Springer Tracts in Advanced Robotics 83

For further volumes:
http://www.springer.com/series/5208

STAR (Springer Tracts in Advanced Robotics) has been promoted under the auspices of EURON (European Robotics Research Network)

Alcherio Martinoli, Francesco Mondada,
Nikolaus Correll, Grégory Mermoud,
Magnus Egerstedt, M. Ani Hsieh,
Lynne E. Parker, and Kasper Støy (Eds.)

Distributed Autonomous Robotic Systems

The 10th International Symposium

Springer

Editors

Prof. Alcherio Martinoli
EPFL ENAC IIE DISAL
Lausanne
Switzerland

Dr. Francesco Mondada
EPFL STI IMT LSRO
Lausanne
Switzerland

Prof. Nikolaus Correll
Department of Computer Science
University of Colorado at Boulder
Boulder, Colorado
USA

Dr. Grégory Mermoud
EPFL ENAC IIE DISAL
Lausanne
Switzerland

Prof. Magnus Egerstedt
Department of Electrical
and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia
USA

Prof. M. Ani Hsieh
Department of Mechanical Engineering
and Mechanics
Drexel University
Philadelphia, Pennsylvania
USA

Prof. Lynne E. Parker
Department of Electrical Engineering
and Computer Science
University of Tennessee
Knoxville, Tennessee
USA

Prof. Kasper Støy
Maersk Mc-Kinney Moller Institute
University of Southern Denmark
Odense
Denmark

ISSN 1610-7438
ISBN 978-3-642-32722-3
DOI 10.1007/978-3-642-32723-0
Springer Heidelberg New York Dordrecht London

e-ISSN 1610-742X
e-ISBN 978-3-642-32723-0

Library of Congress Control Number: 2012946355

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Foreword

Robotics is undergoing a major transformation in scope and dimension. From a largely dominant industrial focus, robotics is rapidly expanding into human environments and vigorously engaged in its new challenges. Interacting with, assisting, serving, and exploring with humans, the emerging robots will increasingly touch people and their lives.

Beyond its impact on physical robots, the body of knowledge robotics has produced is revealing a much wider range of applications reaching across diverse research areas and scientific disciplines, such as: biomechanics, haptics, neurosciences, virtual simulation, animation, surgery, and sensor networks among others. In return, the challenges of the new emerging areas are proving an abundant source of stimulation and insights for the field of robotics. It is indeed at the intersection of disciplines that the most striking advances happen.

The *Springer Tracts in Advanced Robotics (STAR)* is devoted to bringing to the research community the latest advances in the robotics field on the basis of their significance and quality. Through a wide and timely dissemination of critical research developments in robotics, our objective with this series is to promote more exchanges and collaborations among the researchers in the community and contribute to further advancements in this rapidly growing field.

DARS is a well-established single-track conference that gathers every two years the main researchers in *Distributed Autonomous Robotic Systems*. The papers from the last four editions have been published as edited collections by Springer. STAR is proud to welcome the Tenth edition of DARS among the volumes resulting from thematic symposia devoted to excellence in robotics research.

The volume edited by Alcherio Martinoli, Francesco Mondada, Nikolaus Correll, Grégory Mermoud, Magnus Egerstedt, M. Ani Hsieh, Lynne E. Parker and Kasper Støy offers in its forty-three chapters an interdisciplinary collection of technologies, algorithms, system architectures, and applications of advanced distributed robotic systems. The contents are effectively grouped into four thematic parts, each

introduced by an invited contribution by a world-renowned scholar in the field: Part I on distributed sensing, Part II on localization, navigation, and formations, Part III on coordination algorithms and formal methods, Part IV on modularity, distributed manipulation, and platforms.

Rich by topics and authoritative contributors, DARS culminates with this unique reference on the current developments and new directions in the field of distributed autonomous robotic systems. A very fine addition to the series!

Naples, Italy
July 2012

Bruno Siciliano
STAR Editor

Preface

The goal of the Symposium on Distributed Autonomous Robotic Systems (DARS) is to exchange and stimulate research ideas to realize advanced distributed robotic systems. Distributed robotics is a rapidly growing, interdisciplinary research area lying at the intersection of computer science, communication and control systems, and electrical and mechanical engineering. Technologies, algorithms, system architectures, and applications were presented and discussed during a single-track, 3-day symposium. The 10th edition of DARS took place at the École Polytechnique Fédérale de Lausanne (EPFL), in its idyllic location on the shores of Lake Geneva, Switzerland. The symposium also included a great social event in the Lavaux, a UNESCO World Heritage Site, immersed in the beautiful fall colors, just at the end of the grape harvesting period. More details and pictures can be found on http://dars2010.epfl.ch.

DARS 2010 has been an excellent 10th anniversary edition thanks to the high quality of the submissions and selective reviewing process. We received a total of 75 submissions; 30 contributions were presented both orally and as a poster, while 13 uniquely as poster. Each submitted paper was reviewed by at least three reviewers and a technical program co-chair. The editors of this book—four technical program co-chairs (Magnus Egerstedt, M. Ani Hsieh, Lynne E. Parker, and Kasper Støy), two publication co-chairs (Grégory Mermoud and Nikolaus Correll), and two general co-chairs (Alcherio Martinoli and Francesco Mondada)—coordinated the review process with the help of the 99 members of the program committee. We are very grateful to all the reviewers and technical program co-chairs for their thoroughness and constructivism in reviewing the papers. All the accepted papers, including those presented only as poster, were included in the digital pre-proceedings distributed at the event and conditionally accepted for inclusion in this STAR volume, contingent to both presentation of the work at the symposium and proper addressing of the reviewers' and technical co-chairs' remarks. To this purpose authors were encouraged to submit a revised version after the conference together with a cover letter explaining how the reviewers' criticism was addressed. We noticed a drastic improvement in the quality of contributions due to the introduction of this second post-conference

quality control checkpoint; only a few authors were encouraged to take into account final minor suggestions and eventually all the revised papers were accepted in this volume. The overall collection consists therefore of 43 original contributions which are organized in four different parts, each introduced by a different technical program co-chair: distributed sensing (Part I); localization, navigation, and formations (Part II); coordination algorithms and formal methods (Part III); modularity, distributed manipulation, and platforms (Part IV). We feel that this breakdown is indeed representative of the current research activities in distributed robotics and is coarse enough to remain valid over the next few years.

The program of DARS 2010 included several invited keynote talks by world-renowned speakers representing well the four areas of distributed robotics mentioned above: Gaurav S. Sukhatme, University of Southern California for Part I; Raffaello D'Andrea, ETH Zurich and Kiva Systems for Part II; Radhika Nagpal, Harvard University for Part III; and Haruhisa Kurokawa, AIST for Part IV. We include in this volume abstracts and bio-sketches for each invited contribution and speaker, respectively.

DARS 2010 distributed two awards, one for the best student contribution and one for overall best contribution, co-sponsored by the DARS 2008 organizing committee, represented by Haruhisa Kurokawa at the symposium. The award panel was chaired by Hajime Asama (Tokyo University) and included Alan Winfield (University of West England), Radhika Nagpal (Harvard University), Haruhisa Kurokawa (AIST), James McLurkin (Rice University), and Magnus Egerstedt (Georgia Institute of Technology). The award selection process took into account various factors, including the reviewers' score, the revised contribution included in the digital pre-proceedings, the presentation, and related discussion at the symposium. The Best Paper Award was assigned to T.W. Mather, C. Braun and M.A. Hsieh (Drexel University) for their paper entitled "Distributed Filtering for Time-Delayed Deployment to Multiple Sites". The Best Student Paper Award was shared by two contributions, namely that of D. Mellinger, M. Shomin, N. Michael and V. Kumar (University of Pennsylvania) entitled "Cooperative Grasping and Transport using Multiple Quadrotors" and that of Y. Chen, X. C. Ding, A. Stefanescu and C. Belta (Boston University) entitled "A Formal Approach to Deployment of Robotic Teams in an Urban-Like Environment".

Last but not least, we would like to acknowledge the support of our partners in hosting DARS 2010. The Swiss National Science Foundation, the Swiss National Center for Competence in Research for Mobile Information and Communication Systems, the Swiss National Center for Competence in Research for Robotics, the Institute of Environmental Engineering at EPFL, and all of our industrial partners (BlueBotics SA, Cyberbotics S.à.r.l, GCtronic S.à.r.l, K-Team SA, and Skybotix S.à.r.l,) have financially co-sponsored the symposium, while the IEEE Robotics and Automation Society has been involved as technical co-sponsor. We would also like to thank the Editor-in-Chief of the STAR series, Bruno Siciliano, as well as Thomas Ditzinger, responsible coordinator of the series representing Springer Verlag, for affording us the opportunity to publish for the first time the proceedings of a DARS symposium in such prestigious venue. Finally, the symposium would not have been

possible without the hard work of a wonderful local organization team consisting of enthusiastic administrative assistants, PhD students, and research collaborators (see the DARS 2010 website for names and pictures).

We hope that this STAR volume will raise the same excitement and lively discussions that characterized the DARS 2010 symposium!

Lausanne, Switzerland
June 11, 2012

Alcherio Martinoli
Francesco Mondada
Nikolaus Correll
Grégory Mermoud

Invited Keynote Presentations

Termites, Starfish, and Robot Collectives

Radhika Nagpal *Harvard University, USA*

Abstract. Biological systems, from embryos to social insects, get tremendous mileage by having vast numbers of cheap and unreliable individuals cooperate to achieve complex goals. We are also rapidly building new kinds of distributed systems with similar characteristics, from multi-modular robots and robot swarms, to vast sensor networks. Can we engineer collective systems to achieve the kind of complexity and self-repair that nature seems to achieve? In this talk, I will describe several ongoing projects from my group where we use inspiration from nature – termites, starfish, and cells – to design collective robotic systems. For example, simple mobile robots that collectively build structures without explicit communication, self-adaptive modular robots that respond to the environment, and low-cost swarm robots that could self-assemble large-scale shapes. In each case, we use inspiration from biology to design simple decentralized cooperation, and techniques from computer science to analyze and generalize these algorithms to new tasks. A common theme in all of our work is understanding self-organizing multi-agent systems: how does robust collective behavior arise from many locally interacting agents, and how can we systematically program simple agents to achieve the global behaviors we want.

Biography. Radhika Nagpal is a Professor of Computer Science at Harvard University. She received her PhD degree in Computer Science from MIT, and spent a year as a research fellow at Harvard Medical School. She is a recipient of the 2005 Microsoft New Faculty Fellowship award, the 2007 NSF Career award and the 2010 Borg Early Career Award. Her research interests are biologically-inspired engineering principles for multi-agent systems and computational models multicellular biology.

Some Applications of Distributed Estimation and Control

Raffaello D'Andrea *ETH Zurich, Switzerland and Kiva Systems, USA*

Abstract. In this talk I will discuss several applications of distributed estimation and control: Kiva Systems, a company that uses hundreds of mobile robots to move inventory in distribution facilities; the Balancing Cube, a structure that can balance on any one of its edges or corners using six rotating mechanisms on the cube's inner faces; the Distributed Flight Array, a flying platform consisting of multiple autonomous single propeller vehicles that are able to drive, dock with their peers, and fly in a coordinated fashion; the Flying Machine Arena, a research-driven airspace where vehicles teach themselves – and each other – how to fly.

Biography. Raffaello D'Andrea is Professor of Dynamic Systems and Control at ETH Zurich and Technical Co-Founder of Kiva Systems, a company that develops adaptive and self-configuring warehouse automation systems using hundreds of networked, mobile robots. Also a creator of dynamic sculpture, he has shown his work at international venues including the Venice Biennale, the Luminato Festival, Ars Electronica, and ideaCity; two of his pieces are in the permanent collection of the National Gallery of Canada.

Survey of Modular Robotics as DARS Research

Haruhisa Kurokawa *AIST, Japan*

Abstract. Modular robotics has been widely researched over the past 20 years. Modular robots, especially self-reconfigurable ones, have many research topics in common with other research of DARS. Currently, however, most of the claimed prospects seem unfinished dreams. For example, only simple scalability has been obtained. Scalability and fault tolerance is far more difficult to attain by a physical system than an information system, and simple and quantitative scalability, even if attained, will not lead to qualitative one enabling graceful degradation. Joining forces of multiple modules is another difficult problem, though such an ability is indispensable to most robots. Applications of modular robots, especially of lattice-type systems, have not been clear. Endoluminal inspection and surgery will be a good application, but centralized or manual control is better suited for such. The history of modular robotics, with achievements and problems, can anyhow contribute to future DARS research such as in micro or nano scale, and the research, mainly ours, is surveyed in this talk.

Biography. Haruhisa Kurokawa received M.E. in Precision Machinery Engineering in 1981, and Dr. degree in Aeronautical and Astronautical Engineering in 1997, both from the University of Tokyo. He is currently Senior Researcher of the Field Robotics Research Group, Intelligent Systems Institute, National Institute of Advanced Industrial Science and Technology (AIST), Japan. He served as the general chair of DARS 2008. His main research subjects are kinematics of mechanisms, control in space, distributed autonomous systems and nonlinear control.

Monitoring the Coastal Ocean using Underwater Networked Robots: Algorithms and Experiments

Gaurav S. Sukhatme *University of Southern California, USA*

Abstract. We describe recent progress in systems and algorithms for underwater robots with applications to the monitoring of the coastal ocean. We describe a new algorithm for area coverage with a strong theoretical guarantee and a data fusion method for a communication-constrained underwater multi-robot system. Experimental results from sea trials (6 weeks) will be presented. We also give a brief overview of the underlying systems infrastructure that we have built to support the experiments and field trials.

Biography. Gaurav S. Sukhatme is a Professor of Computer Science (joint appointment in Electrical Engineering) at the University of Southern California (USC). He received his undergraduate education at IIT Bombay in Computer Science and Engineering, and M.S. and Ph.D. degrees in Computer Science from USC. He is the co-director of the USC Robotics Research Laboratory and the director of the USC Robotic Embedded Systems Laboratory which he founded in 2000. His research interests are in multi-robot systems, robot networks and aquatic robots. He has published over 200 papers in these and related areas. Sukhatme has served as PI on numerous NSF, DARPA and NASA grants. He is a Co-PI on the Center for Embedded Networked Sensing (CENS), an NSF Science and Technology Center. He is a senior member of the IEEE, and a member of AAAI and the ACM. He is a recipient of the NSF CAREER award and the Okawa foundation research award. He has served on many conference program committees, and is one of the founders of the Robotics: Science and Systems (RSS) conference. He was one of the program chairs of the 2008 IEEE International Conference on Robotics and Automation (ICRA) and is the program chair of the 2010 IEEE/RSJ Intelligent Robots and Systems (IROS) conference. He is the Editor-in-Chief of Autonomous Robots. He has served as Associate Editor of the IEEE Transactions on Robotics and Automation, the IEEE Transactions on Mobile Computing, and on the editorial board of IEEE Pervasive Computing.

Program Committee

Francesco Amigoni	Politecnico di Milano, Italy
Marcelo Ang	National University of Singapore
Ronald Arkin	Georgia Institute of Technology, Atlanta, USA
Filippo Arrichiello	Università degli Studi di Cassino, Italy
Minoru Asada	Osaka University, Japan
Hajime Asama	University of Tokyo, Japan
Jacob Beal	BBN Technologies, Cambridge, USA
Calin Belta	Boston University, USA
Gerardo Beni	University of California, Riverside, USA
Sarah Bergbreiter	University of Maryland, College Park, USA
Spring Berman	Harvard University, Boston, USA
Andreas Birk	Jacobs University, Bremen, Germany
Tim Bretl	University of Illinois at Urbana Champaign, USA
Mathias Broxvall	University of Orebro, Sweden
Wolfram Burgard	Albert-Ludwigs-Universität Freiburg, Germany
Zack Butler	Rochester Institute of Technology, USA
Raja Chatila	Laboratoire d'Architecture et d'Analyse des Systèmes, CNRS, France
Greg Chirikjian	Johns Hopkins University, Baltimore, USA
Anders Lyhne Christensen	Lisbon University Institute, Portugal
Henrik Christensen	Georgia Institute of Technology, Atlanta, USA
Timothy Chung	Naval Postgraduate School, Monterey, USA
Nikolaus Correll	University of Colorado, Boulder, USA
Jorge Cortes	University of California, San Diego, USA
Raffaello D'Andrea	ETH Zurich, Switzerland
Prithviraj Dasgupta	University of Nebraska, Omaha, USA
Carrick Detweiler	University of Nebraska at Lincoln, USA
M. Bernardine Dias	Carnegie Mellon University, Pittsburgh, USA

Marco Dorigo	Université Libre de Bruxelles, Belgium
Magnus Egerstedt	Georgia Institute of Technology, Atlanta, USA
Riccardo Falconi	Università di Bologna, Italy
Francesco Mondada	École Polytechnique Fédérale de Lausanne, Switzerland
Emilio Frazzoli	Massachusetts Institute of Technology, Cambridge, USA
Eric Frew	University of Colorado at Boulder, Boulder, CO, USA
Toshio Fukuda	Nagoya University, Japan
Luca Gambardella	IDSIA Lugano, Switzerland
Simon Garnier	Princeton University, USA
Andrea Gasparri	Università degli Studi Roma Tre, Italy
Veysel Gazi	TOBB Ekonomi ve Teknoloji Universitesi, Ankara, Turkey
Brian Gerkey	Willow Garage, Menlo Park, USA
Maria Gini	University of Minnesota, Minneapolis, USA
Roderich Gross	University of Sheffield, UK
Norihiro Hagita	Advanced Telecommunications Research Institute International, Kyoto, Japan
Adam Halasz	West Virginia University, Morgantown, USA
Heiko Hamann	University of Graz, Austria
Yasuhisa Hirata	Tohoku University, Japan
Ayanna Howard	Georgia Institute of Technology, Atlanta, USA
M. Ani Hsieh	Drexel University, Philadelphia, USA
Auke Ijspeert	École Polytechnique Fédérale de Lausanne, Switzerland
Akio Ishiguro	Tohoku University, Japan
Volkan Isler	University of Minnesota, Minneapolis, USA
Gil Jones	Willow Garage, Menlo Park, USA
Michael Kaess	Massachusetts Institute of Technology, Cambridge, USA
Serge Kernbach	Universitaet Stuttgart, Germany
Chris Kitts	Santa Clara University, California, USA
Yuichi Kobayashi	Tokyo University of Agriculture and Technology, Japan
Kazuhiro Kosuge	Tohoku University, Japan
Navinda Kottege	CSIRO, Pullenvale, Australia
Ryo Kurazume	Kyushu University, Japan
Haruhisa Kurokawa	National Institute of Advanced Science and Technology, Tsukuba, Japan
Kostas Kyriakopoulos	National Technical University, Athens, Greece
Naomi Leonard	Princeton University, USA

Pedro U. Lima	Instituto Superior Tecnico, Lisbon, Portugal
Savvas Loizou	Frederick University, Cyprus
Kevin Lynch	Northwestern University, Evanston, USA
Lino Marques	University of Coimbra, Portugal
Alcherio Martinoli	École Polytechnique Fédérale de Lausanne, Switzerland
Fernando Matia	Universidad Politecnica de Madrid, Spain
James Mclurkin	Rice University, Houston, USA
Grégory Mermoud	École Polytechnique Fédérale de Lausanne, Switzerland
Mehran Mesbahi	University of Washington, Seattle, USA
Nathan Michael	University of Pennsylvania, Philadelphia, USA
Dejan Milutinovic	University of California at Santa Cruz, USA
Nader Motee	California Institute of Technology, Pasadena, USA
Radhika Nagpal	Harvard University, Boston, USA
Daniele Nardi	Università La Sapienza Roma, Italy
Giuseppe Notarstefano	University of Lecce, Italy
Kazuhiro Ohkura	Hiroshima University, Japan
Giuseppe Oriolo	Università La Sapienza, Rome, Italy
Jun Ota	University of Tokyo, Japan
Stefano Panzieri	Università degli Studi Roma Tre, Italy
Nikolaos Papanikolopoulos	University of Minnesota, Minneapolis, USA
Lynne E. Parker	University of Tennessee, USA
Guilherme Pereira	Universidade Federal de Minas Gerais, Brasil
Sameera Poduri	University of Southern California, Los Angeles, USA
Ioannis Rekleitis	McGill University, Montreal, Canada
Wei Ren	Utah State University, Logan, USA
Paul Rybski	Carnegie Mellon University, Pittsburgh, USA
Alessandro Saffiotti	Orebro University, Sweden
Erol Sahin	Middle East University, Ankara, Turkey
Ketan Savla	Massachusetts Institute of Technology, Cambridge, USA
Luca Schenato	Università di Padova, Italy
Thomas Schmickl	Karl-Franzens-Universitat Graz, Austria
Mac Schwager	University of Pennsylvania, Philadelphia, USA
Dylan Shell	Texas A&M University, College Station, USA
Wei-Min Shen	University of Southern California, Los Angeles, USA
Kasper Stoy	University of Southern Denmark
Ken Sugawara	Tohoku Gakuin University, Japan
Gaurav Sukhatme	University of Southern California, Los Angeles, USA
Guy Theraulaz	Université Paul Sabatier and CNRS, Toulouse, France

Vito Trianni	Istituto di Scienze e Tecnologie della Cognizione, Roma, Italy
Elio Tuci	Istituto di Scienze e Tecnologie della Cognizione, Roma, Italy
Richard Vaughan	Simon Fraser University, Burnaby, Canada
Richard Voyles	University of Denver, Denver, USA
Justin Werfel	Harvard University, Boston, USA
Alan F. T. Winfield	University of the West of England, Bristol, UK
Heinz Woern	University of Karlsruhe, Germany
Eiichi Yoshida	Joint Japanese-French Robotics Laboratory, Toulouse, France

Contents

Part I: Distributed Sensing

Part II: Localization, Navigation, and Formations

Part III: Coordination Algorithms and Formal Methods

Part I
Distributed Sensing

Part I: Distributed Sensing

M. Ani Hsieh

A significant advantage of distributed autonomous robotic systems lies in their ability to cover large regions in physical space to achieve sampling in both space and time much more efficiently than single robot counterparts. However, the ability for distributed data collection and processing invariably presents challenges at the intersection of communication, control, perception, and more recently energy management.

Stirling and Floreano investigates the trade-offs between energy consumption and deployment time for an aerial swarm of robots in unknown environments. Specifically, they consider three separate deployment strategies and show that the best approach to reduce energy consumption while maintaining a fast deployment rate is to control the density of the aerial swarm. This is an exciting result since this echoes recent results in the studies of starling and locust flocks where swarming behavior can be correlated with the density of the organisms. Cannata and Sgorbissa presents a strategy for coordinating multi-robot teams to achieve coverage of a specific set of locations in a given workspace by controlling the frequency in which the robots visit the locations. In essence, this work reformulates the multi-robot coverage problem in the time domain and presents an approach that allows the team to achieve uniform frequency coverage of the sites of interest. Cortez, Fierro, and Wood presents the use of mobile communication relays collaborating with mobile sensing agents to extend the reach of a sensor network while maintaining the overall connectivity of sensor network. This work furthers our understanding of the impact of non-uniform sensing and communication ranges on the team's ability to achieve the desired level of coverage and presents a system approach towards deriving connectivity constraints in heterogeneous robot teams. Marjovi and Marques an exploration and mapping strategy that enables robot teams to cooperatively seek out and localize odor sources in unknown environments based on olfactory cues. They show that topological maps of the environment can be generated and odor sources can be

M. Ani Hsieh
Drexel University, Philadelphia, PA 19104 USA
e-mail: mhsieh1@drexel.edu

found in a distributed fashion by combining odor concentration measurements with traditional mobile robot sensors.

In addition to considering the trade-offs between communication, control, perception, and energy needs and the impact they have on the overall distributed system's performance, there is the added challenge of working in complex and dynamic environments. Bhattacharya, Michael, and Kumar presents a generalized Voronoi decomposition for non-convex environments based on geodesic distances. This enables the design of real-time feedback control laws for sensor coverage in realistic environments. This generalized Voronoi decomposition can be extended for unknown environments when coupled with entropy-based metrics. Evans, Bahr, and Martinoli investigates the gap between theory and the real world environmental monitoring applications. They study the algorithm performance of constraint chaining for data collection for sensor networks handling real data collected in the field and propose modifications to improve its performance. Lochmatter, Göl, Navarro, and Martinoli presents a cooperative plume tracking strategy for teams of robots. The approach specifically leverages the multi-robot's inherent ability to simultaneously sample the odor concentrations at different points in space to determine the location of an odor source of interest. Umeda, Sekiyama, and Fukuda investigates the use of an ambiguity index to classify the effectiveness of visual features in a scene for distributed object tracking by a team of cooperating robots. The authors show that the proposed ambiguity index can minimize the amount of cognitive sharing in a team of robots and thus result in more effective cooperative object tracking strategies.

Robust distributed coordination strategies enables more complexity in behaviors and the chance for adaptation. Martin, de la Croix, and Egerstedt presents a software framework for dynamic reassignment of tasks among robots. The novelty of this work lies in the framework's ability to change how sensors and actuators interact and interconnect in the reassignment process achieving dynamic reconfiguration of the networked multi-agent system based on task needs.

The use of distributed autonomous robotic systems to accomplish tasks within a complex environment with limited resources requires strategies that can leverage the inherent redundancies within these systems. The work presented in this section is a nice representation of the ongoing efforts in addressing these challenges.

Energy-Time Efficiency in Aerial Swarm Deployment

Timothy Stirling and Dario Floreano

Abstract. A major challenge in swarm robotics is efficiently deploying robots into unknown environments, minimising energy and time costs. This is especially important with small aerial robots which have extremely limited flight autonomy. This paper compares three deployment strategies characterised by nominal computation, memory, communication and sensing requirements, and hence are suitable for flying robots. Energy consumption is decreased by reducing unnecessary flight following two premises: 1) exploiting environmental information gathered by the robots; 2) avoiding diminishing returns and reducing interference between robots. Using a 3-D dynamics simulator we examine energy and time metrics, and also scalability effects. Results indicate that a novel strategy that controls the density of flying robots is most promising in reducing swarm energy costs while maintaining rapid search times. Furthermore, we highlight the energy-time tradeoff and the importance of measuring both metrics, and also the significance of electronics power in calculating total energy consumption, even if it is small relative to locomotion power.

1 Introduction

Autonomous systems must manage their own energy resources to complete missions successfully [15]. This is notably evident with small aerial robots, which have severely limited flight autonomy (typically 10–15 minutes [19, 23]). Although energy efficient algorithms are paramount in creating truly autonomous aerial robots, prior research is sparse [23]. Previously we developed an algorithm for indoor aerial swarm search [22] that exploited the ability of our robots to attach to ceilings, saving energy [19]. This paper expands this work and compares methods to deploy

Timothy Stirling · Dario Floreano
Laboratory of Intelligent Systems (LIS), Ecole Polytechnique Fédéral de Lausanne (EPFL), Switzerland
e-mail: tim.stirling@epfl.ch

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 5–18.
springerlink.com

a swarm of aerial robots into unknown environments, aiming to reduce the total swarm energy cost with rapid operation for a search task.

A complex problem in swarm robotics is controlling deployment into unknown environments. If robots deploy to unnecessary locations, energy is wasted. Furthermore, they may interfere with other robots, e.g. by increasing collision risk [20]. Conversely, if an area receives insufficient robots the task may be unachievable or performance reduced. Rapid deployment is desired to expedite tasks such as disaster mitigation. However, time and energy are not independent, and often there is a tradeoff [13, 6, 10]. Previous research considered only ground robots. However, aerial robots have significantly different energy dynamics, require substantially more energy to locomote [19], and the small payload entails reduced sensing and processing capabilities. Time and energy were previously either examined independently, or only with multi-objective functions that mask trends in the individual metrics. Prior work also usually neglected the energy consumption of sensors and processors [12].

This paper compares three strategies suitable for aerial swarms, characterised by minimal computation, communication and sensing requirements. They are suitable for microcontrollers rather than powerful CPUs and are simple to implement to avoid further complicating autonomous flight control. Total swarm energy and search time are examined in 3-D simulation using a complete energy model validated on real robots [19]. Finally, scalability performance is examined by increasing the robot group size.

2 Related Work

In work by Rybski *et al.* [21], increasing the number of deployed robots led to the phenomenon of *diminishing returns*, as proposed by economists [1]. Additional robots increased performance by decreasing amounts until a peak was reached, after which additional robots no longer improved performance. Moreover, Rosenfeld *et al.* [20] examined scalability in foraging tasks and noted that after a peak in performance, additional robots usually decreased performance (*negative returns*) due to spatial constraints and interference. Spatial constraints are stronger in confined areas, such as narrow corridors, causing congestion and increased collision risk. Therefore, it is important to control deployment to minimise time and energy costs.

The tradeoff between group size and efficiency was examined by Hayes [6] in a search task. A multi-objective performance function was used incorporating search time, energy and robot initialisation costs. This analysis allowed the prediction of the optimal number of robots to complete the search. However, Hayes' analysis assumed an obstacle-free square arena and ignored spatial constraints and interference. Similarly, Mei *et al.* [11] researched methods to determine the optimal group size under constraints of energy, time and environment area. It was shown that energy limitations significantly affected the required group size. However, the environment size was known *a priori* and a centralised planner used.

For operation in unknown environments, Howard *et al.* [7] developed an algorithm that deployed robots one at a time, thereby avoiding interference. Each subsequent deployed robot exploited environment information acquired from previously deployed robots and was guided to optimal environment locations. However, Howard's approach is slow [7]. Alternatively, Zlot *et al.* [25] present a market economy-based architecture [5] for efficient multi-robot exploration. This maximises search area while minimising total travel distance. However, high bandwidth communication is required. Both Howard's and Zlot's approaches are computationally expensive with centralised processing, usually undesirable in swarm robotics. Unfortunately, neither authors provide quantitative results for energy or time costs.

For simple robots with decentralised control, Chang *et al.* [2] deployed robots based on perceived local environment size, reducing unnecessary locomotion. When larger areas are discovered, additional robots are requested to aid exploration, reducing search time. However, this was assessed with ground robots in a simple discrete 2-D simulator with basic environments that reduced spatial constraints and interference. Additionally, the deposition of artificial pheromones was used to control deployment, but no such sensor currently exists for real flying robots.

Alternatively, researchers have developed mechanisms to improve efficiency by controlling robot activation. For example, Liu *et al.* [9] examined energy efficient task allocation in foraging robots. The ratio of active foragers to resting robots was adjusted based on simple adaptation rules. These rules included internal cues of successful foraging, environmental cues from collisions, and social cues of successful foraging by other robots. However, foraging differs from search because (un)successful foraging over time indicates the robot's utility, which can be used to control activity. Furthermore, foraging often involves retrieval to a communal nest, facilitating global coordination through local communication. Finally, a basic energy model was used and time costs were not examined.

In summary, previous research focused only on ground robots, but aerial robots have considerably different energy characteristics [19]. We compare three strategies that are scalable, decentralised, require no *a priori* environment information, and are suitable for aerial robots with nominal computation, sensing and communication requirements.

3 Aerial Swarm Search without Global Information

The aerial swarm search algorithm considered here [22] is based on principles of sensor networks, which perform distributed processing of local information through wireless communication [7]. This work is based on the quadrotor robots we are developing for indoor swarming [17]. The robots are equipped with infrared distance sensors for obstacle detection [17]. A 3-D relative-positioning sensor gives the range and bearing to nearby robots and low-bandwidth short range communication, facilitating coordination [18]. Wireless LAN provides longer range communication. To

prolong missions the robots can attach to ferromagnetic ceilings [19]. Alternatively, Gecko inspired dry adhesives [14] or mechanical perching could be utilised [8, 4], or simply robots could land (merely loosing their elevated perception capabilities).

Robots operate in two control states: "*Beacons*" or "*Explorers*". Beacons are static robots passively attached to the ceiling to conserve energy and form a robotic sensor network [19]. Explorers are flying robots, deploying into the environment guided by Beacons. Beacons sense their local environment and communicate with neighbouring Beacons to guide nearby Explorers. Explorers start clustered on the ground below a pre-deployed Beacon. When deployed, Explorers take off and follow the guidance signal of the nearest Beacon, flying from Beacon to Beacon across the network. Beacons next to unexplored space indicate adjacent locations where a new Beacon is required. Explorers that arrive at these locations attach to the ceiling and become Beacons. Beacons can revert back to Explorers once an area has been searched and redeploy to unexplored areas. We utilise depth-first search [3] which exhaustively explores a subarea of the environment before searching other unexplored areas. This avoids search duplication and unnecessary locomotion compared with stochastic methods, thus reducing the total swarm flight time [22]. Navigation in unknown environments is afforded by the hop-counts of local communication signals propagated across the network [22]. By exploiting the ceiling attachment capability, the swarm energy cost is reduced by 3–400% [22]. A video demonstrating the search behaviour in simulation is available online[1]. A second video demonstrating the current progress in developing this search strategy on real robots is also available[2], which shows entirely autonomous flight.

4 Deployment Strategies

To reduce energy costs and search time, the initial deployment of robots from the ground and the redeployment of Beacons from the ceiling once an area is searched are controlled. Three strategies are compared that are scalable, decentralised, and require low computational and communication resources. The strategies exploit environment information as it is acquired by the robots to reduce unnecessary locomotion [7, 2], and reduce diminishing returns and interference between robots [20, 9]. A video showing the three deployment strategies is available online[3].

4.1 Linear-Temporal Incremental Deployment (LTID)

The simplest strategy, labelled LTID, deploys robots one at a time with a fixed time interval between consecutive launches. This was used in our prior work [22] and is

[1] `http://lis.epfl.ch/~stirling/videos/Swarm_Search.avi`

[2] `http://lis.epfl.ch/~stirling/videos/Eyebot_Autonomous_Flight.mp4`

[3] `http://lis.epfl.ch/~stirling/videos/Deployment_Strategies.mp4`

similar to the linear dispatching presented by Chang *et al.* [2]. Longer inter-launch intervals (λ) slow deployment, but decrease the number of concurrent flying robots. This reduces spatial interference and unnecessary flight by exploiting environmental information acquired from the expanding Beacon network. Once a subarea of the environment has been searched, Beacons redeploy as Explorers to new unexplored areas. Before this redeployment commences, there may be multiple Explorers flying into this subarea where they are not required, which is reduced with longer inter-launch intervals. Thus, LTID reduces energy consumption by reducing interference and unnecessary locomotion.

To implement LTID, robots are assigned a unique ID $\{1,...,N\}$ and initially launch after $\lambda \times$ID seconds. Redeploying beacons also wait λ before detaching. LTID does not adapt online, but λ could be optimised *a priori* if the type of environment is known [22]. The advantages of LTID are its simplicity and no requirements for sensing, communication or significant processing. Therefore, LTID serves as a baseline strategy from which the other two strategies can be compared.

4.2 Single Incremental Deployment (SID)

SID is similar to LTID and deploys one robot at a time, but waits for the previous robot to become a Beacon before launching the next. This is similar to Howard *et al.*'s [7] approach in that the swarm waits for the previous robot to examine the newly discovered environment, but no centralised processing or map is required. SID reduces unnecessary flight time because the next robot will only (re)deploy once the Beacon network has sensed the environment and perceived if and where a new Beacon is required. Thereby, Explorers always fly directly to the desired deployment location. To implement SID, the swarm communicates if an Explorer is flying. This can be achieved by propagating local messages across the Beacon network. However, here we employ a simplified mechanism using long-range wireless communication. Beacons signal to the whole swarm if they perceive a flying Explorer and robots only (re)deploy if no signal is received. To ensure only a single robot deploys at a time, random timeouts are used. When no flying Explorer signal is present, robots wait a short random time period (typically 1–2 s). If after this period there is no flying Explorer signal, the robot can deploy.

Robots usually deploy more slowly than with LTID, increasing search times. Therefore, although SID may reduce flight energy consumption compared with LTID, the energy consumption of sensors and processors may be elevated due to the increased runtime. Additionally, there may be a robot that is closer to the desired destination due to the redeployment of Beacons, but since the launch selection is based on random timeouts, the closest robot is not guaranteed to deploy. Various strategies exist that would ensure the closest robot is selected [24]. SID is a fixed deployment scheme without adjustable parameters. SID requires no additional sensing or computation, but very low bandwidth communication is required for coordination.

4.3 Adaptive Group Size (AGS)

AGS is a novel strategy that adapts the density of flying robots, inspired by Liu *et al.*'s [9] rules to control robot foraging activity. However, we consider a search task rather than a foraging task (see Sect. 2), and we aim to explicitly avoid collisions. AGS initially rapidly deploys robots, every 2–3 seconds. Flying Explorers measure the density of neighbouring flying robots using their relative-positioning sensor [18] and will probabilistically land if the density is higher than a predefined threshold. This decreases the ratio of flying robots, reducing diminishing returns and interference. Robots which have landed launch again when there are no robots flying in the vicinity. The density of flying robots ρ is given by: $\rho = \sum_{i=1}^{N} \frac{4}{d_i}$, where d_i is the distance to neighbouring robot i. The constant 4 is a normalisation factor such that a single flying robot 4.0 m away (considered a safe flight separation) gives ρ the unit value 1. If ρ is greater than a threshold τ (typically 3.0–8.0) the robot will try to land. To prevent multiple robots landing simultaneously, robots wait a random timeout period (typically 1–2 s) while they signal their intention to land. After this timeout robots can land if no signal of a neighbouring robot's intention to land is received. Otherwise, the robot with the highest ID has priority in landing. Robots could attach to the ceiling [19] instead of landing, but this could interfere with the Beacon network.

AGS uses the perceived density of flying robots to avoid diminishing returns. For example, if robots land in high density areas where collision risk is considerable, interference is reduced. Furthermore, the perceived density implicitly encodes local environment information. If many robots are flying in a confined space the density will be high. Avoiding high densities reduces the deployment of robots to locations where they may not be required. Therefore, energy consumption is reduced by decreasing unnecessary flight time and interference. AGS can be optimised by varying the threshold τ. No significant processing or high bandwidth communication is required. However, a sensor is required to measure the density, which could be simple Time of Flight sensors or local communication instead of relative-positioning [18].

5 Experimental Method

Comparing strategies requires extensive simulation analysis since it is infeasible to gather sufficient data for statistical analysis with real flight tests given the large parameter space and the challenging logistics of conducting numerous flight experiments. Therefore, a realistic 3-D dynamics simulator was utilised [16], as discussed below in Sect. 5.1

Performance was measured over 100 trials with robots clustered in random starting locations in randomly generated maze-like corridor environments. Environments were constructed from 40 connected 3×3 m cells (see [22] for details). Fig. 1 shows

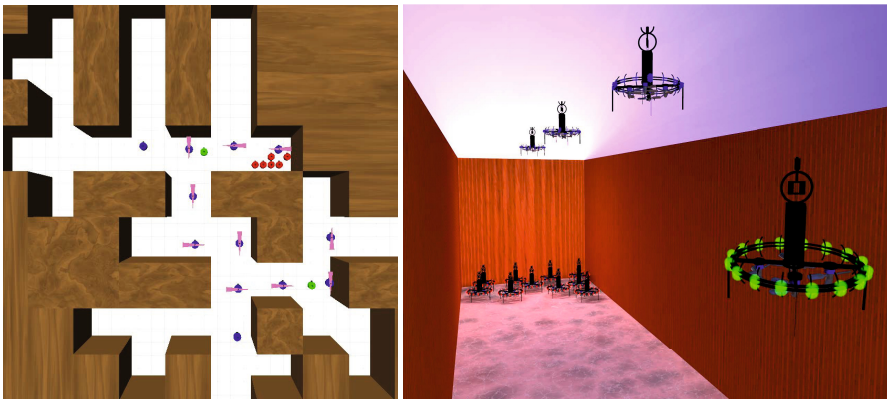


Fig. 1 *Left*: Typical randomly generated maze environment. *Right*: The swarm deploying with robots on the ground, a flying Explorer and Beacons on the ceiling.

a typical environment and the swarm deploying with Beacons and a flying Explorer. For the first experiments 20 robots were available to deploy. Subsequent experiments assessed the scalability performance, so the number of robots was increased from 20 to 30. We measured search time, coverage area and swarm energy consumption, calculated with an energy model of the rotor thrust-power curve of a real quadrotor helicopter [19], shown in Fig. 2. This model facilitated the accurate prediction of flight endurance within a 1% error. The model was extended to include the energy used by sensors and processors, detailed in Table 1. This creates three electronics power consumption rates: when the robot was flying (high power); when a Beacon on the ceiling (medium power); and when resting on the ground (low power). This equates to a power consumption of 120 W for flying Explorers, 5 W for Beacons, and 0.5 W for robots resting on the ground. These rates were validated on real flying robots [19] and are similar to other rotorcraft, e.g. [23]. Moreover, the performance trends are robust to changes in model parameters since the power rates are differentiated by an order of magnitude.

Table 1 Power consumption of components used to develop the energy model for aerial robots

Component	Power (W)	Comment
Total rotor power	100–120	Depends on payload [19]
Flight computer	2.44	Sensors & microprocessor [19]
Microprocessor	0.125	Microchip PIC32 40MHz
802.11a WiFi	1.5, 1.22, 0.01	Send, receive & sleep power [15]
Infrared distance sensor	0.165	Sharp GP2Y0A02YK
Ultrasonic altitude	0.015	MaxBotics LV-MaxSonar-EZ4
3-D Relative positioning	~ 7	For 20 robots [18]

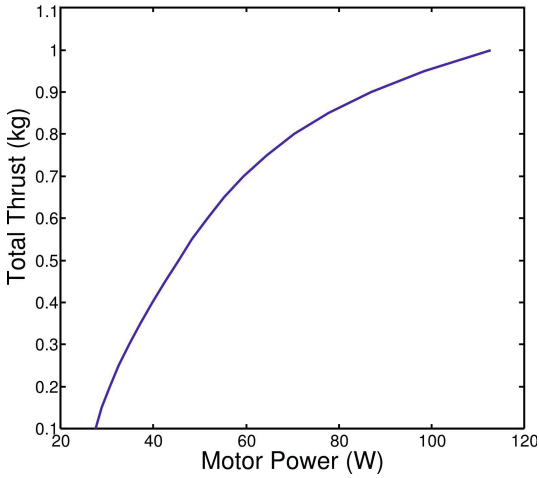


Fig. 2 Thrust-power curve of the quadrotor propulsion system validated in [19]

5.1 Simulation and Flight Dynamics

A custom 3-D dynamics simulation was developed using the Open Dynamics Engine[4] (ODE). An input force vector F_{tot} is applied to a rigid body with mass m giving accelerations: $\dot{V} = \frac{1}{m}F_{tot}$. Angular accelerations and torque were neglected since they are stabilised by the flight controller [17]. Gaussian noise was added to simulate turbulence and imprecise control, with standard deviations (s.d.) set empirically (2.5×10^{-2} N) as the platform has not yet been characterised. However, the altitude fluctuation was modelled from previous work in [17]. F_{tot} is given by:

$$F_{tot} = F_g + F_c + F_d + \varepsilon_N \ , \tag{1}$$

where ε_N is a Gaussian noise vector for roll, pitch and thrust standard deviations: $\varepsilon_N \sim \mathcal{N}(0, \hat{\sigma})$. F_g denotes the platform weight with $F_g = [0,0,m\cdot g]^T$. F_c is the control force vector formed from desired pitch f_p and roll f_p forces, combined with the altitude control f_a from a PID controller [17]: $F_c = [f_p, f_r, f_a]^T$. Drag force is calculated with:

$$F_d = -\frac{1}{2}\rho V^2 A C_d, \tag{2}$$

where ρ is the specific air-density, V is air-speed, A is the frontal reference area and C_d is the estimated drag coefficient.

Sensor noise was Gaussian with s.d. measured from characterisation experiments: 2.5 cm for the ultrasound altitude sensor and 5 cm for infrared distance sensors used for obstacle avoidance. The relative-positioning sensor has been characterised in [18]: the range s.d. is 17 cm and bearing s.d. is 6.1°.

[4] www.ode.org

6 Results

To compare performances both the total swarm energy consumption and search time metrics are examined separately. Subsequently, a multi-objective function is used that linearly combines energy and time into a simple single parameter-free metric, the Energy-Time-Product (ETP), measured in Joule-Seconds (Js) [2]. The ETP is inspired by the Power-Delay Product frequently used in electronics engineering. Both energy and time metrics are taken to have equal unit weighting, which assumes the equal importance of these factors and also avoids arbitrary parameterisation. Alternative weighting of parameters is discussed in Sect. 7. With AGS and LTID we varied the inter-launch interval with $\lambda = 6$, 8, 10, 12, 18 and 24 seconds, and the density-threshold τ from 3.0 to 8.0, respectively. Finally, scalability performance is examined by increasing the robot group size. Shapiro-Wilk tests indicated small deviations from normal-distributions, so Kruskal-Wallace χ^2 and Spearman's Rho r_s non-parametric tests were used to examine the statistical significance of any effects. Medians are shown with standard deviations in parenthesis.

6.1 Overall Comparisons

Importantly, there was no significant difference in median coverage area (99.4%) across all strategies ($\chi^2 = 18.42$, df = 12, p = 0.1), permitting fair comparisons. The mean coverage area was 95.7%. Comparing all strategies over all parameters, the fastest was AGS with $\tau = 6.0$ with a median search time of 307.9 s (57.6) (Fig. 3(a)). The most energy efficient was AGS with $\tau = 4.0$, with a median of 178.5 kJ (33.8) (Fig. 3(b)). The slowest strategy was SID taking 1013.9 s (118.6), 229.3% slower than AGS with $\tau = 6.0$. The least energy efficient was LTID with $\lambda = 6$ with a median of 253.4 kJ (59.2), requiring 42.0% more energy than AGS with $\tau = 4.0$. Comparing ETP performances (Fig. 3(c)), LTID suffers from a tradeoff between search time and energy-efficiency achieving its lowest ETP of 800.5×10^5 Js (26.5) with $\lambda = 10$ s. Since AGS showed low energy consumption with fast search times it produced the lowest overall ETP of 579.7×10^5 Js (183.1), with $\tau = 6.0$. Finally, because SID suffers from slow deployment the median ETP was high, 1866.3×10^5 Js (459.8).

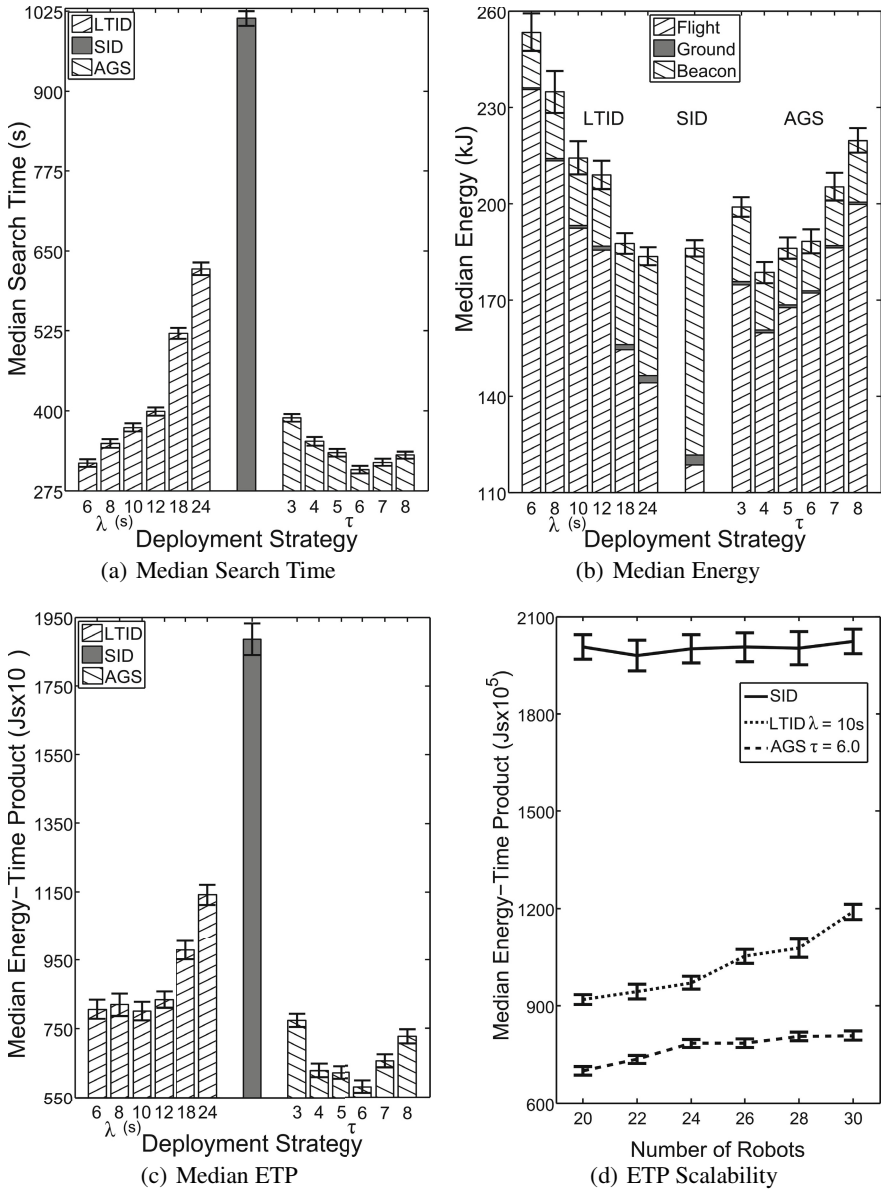
With LTID, increasing the time between consecutive robot launches (λ) significantly decreased the median swarm energy consumption ($r_s = -0.47$, df = 598, $p < 0.001$) and increased median search time ($r_s = 0.80$, df = 598, $p < 0.001$). Increasing λ from 6 s to 24 s reduces the median energy consumption by 27.6% and increases median search time by 95.6%. Energy consumption is deceased by reducing robot deployment to unnecessary locations and decreasing interference, decreasing flight energy. Although energy consumption decreased as λ increased, the search time increased more strongly, so the median ETP increased.

SID has no tunable parameters. SID had the slowest search time since only a single robot flies at a time. Robots deployed only when and where necessary, which minimised unnecessary flight energy over all deployment strategies, confirmed with multiple comparisons at the $p < 0.001$ level (using Wilcoxon ranksum tests). However, SID did not achieve the lowest energy consumption because of the energy consumption of the Beacons' sensors and processors over the long search duration. Therefore, even although the power consumption of electronics is small compared with the rotor power it is important to consider within a complete energy model. The median ETP was high due to the slow search time.

With AGS, varying the threshold τ significantly affected the median search time ($\chi^2 = 116.3$, df = 5, $p < 0.001$) and median swarm energy consumption ($\chi^2 = 102.1$, df = 5, $p < 0.001$). Both energy and time metrics form a $\cup$-shape. This is because at low thresholds flying robots have a higher probability of landing when encountering neighbours, which incurs a time and energy cost. Conversely, increasing τ reduces the effect of controlling the group size and so increases interference and unnecessary flight, thereby increasing energy consumption and search time. Therefore, there is an optimal threshold ($\tau = 6.0$) that minimises the ETP.

6.2 Scalability Performance

To assess scalability performance the median ETP was compared when the number of robots was increased from 20 to 30. For LTID and AGS the parameters that minimised the ETP for 20 robots (λ = 10 s and τ = 6.0, respectively) were the same for 26 and 30 robots, so were used for all group sizes. Since increasing the swarm size can increase the expected coverage area [22] and associated search time and flight energy, we restricted results to trials that achieved 100% coverage, ensuring fair comparisons. Results are shown in Fig. 3(d). Increasing the robot group size significantly increases the median ETP for both AGS ($r_s = 0.28$, df = 598, $p < 0.001$) and LTID ($r_s = 0.41$, df = 598, $p < 0.001$), but not for SID ($r_s = 0.007$, df = 598, $p > 0.86$). LTID increases at a higher rate compared to AGS. AGS minimises the median ETP over all tested group sizes. The median ETP of SID is approximately constant because only a single robot flies at a time, so there is no unnecessary flight time. However, SID never becomes competitive even for large group sizes. The ETP trends for both AGS and LTID mask a decrease in median search time (9.6% and 9.1%, respectively) and an increase in median energy cost (15.6% and 29.4%, respectively). This is due to the increased parallelisation afforded by additional robots accelerating the search and consequently increasing flight energy. Importantly, for all strategies the median ETP (and energy consumption) *per robot* decreases as the group size increases, indicating good scalability performance.



(a) Median Search Time (b) Median Energy

(c) Median ETP (d) ETP Scalability

Fig. 3 Median **a**) search time, **b**) swarm energy cost and **c**) Energy-Time-Product (ETP) tested with 20 robots over 100 trials. Standard error bars (standard deviation divide by square root of sample size) are shown. The energy results show the constituent ground, beacon and flight costs. The inter-launch interval λ of LTID and the flying robot density-threshold τ of AGS were varied. **d**) Median ETP performance as the group size increases. AGS has the lowest energy consumption, fastest search time, lowest ETP and good scalability performance.

7 Conclusion and Future Work

In this paper we compared three strategies to deploy flying robots for a search task in unknown environments using a complete energy model for electronics and motors validated on real flying robots. All strategies were characterised by nominal computation, memory and communication requirements, and were necessarily simple to facilitate implementation on aerial systems without further complexifying autonomous flight control. To summarise:

- LTID demonstrated that slowing deployment facilitates a significant reduction in energy consumption up to 27.6%, but this increased search time by 95.6%. This tradeoff can be optimised by adjusting the inter-launch interval. No communication or additional sensing is required and the implementation is simple. Therefore, LTID serves as a useful benchmark strategy.
- SID ensures only one robot flies at a time and leads to low energy consumption, but a very high search time. This indicates that mitigating deployment of robots to unnecessary locations by exploiting acquired environmental information significantly reduces flight energy. However, the increased energy consumption of sensors and processors prevents SID achieving the best overall energy-efficiency. No additional sensing is required, but very low-bandwidth communication is used for coordination.
- The AGS strategy results showed that, by controlling the density of flying robots, the swarm energy consumption can be reduced while also achieving rapid search. AGS require a sensor to measure the local robot density.

With optimal parameters, AGS ($\tau = 6.0$) has an ETP 27.6% better than LTID ($\lambda = 10\,\text{s}$) and 69.3% better than SID. All strategies showed good scalability performance, with a decreased median ETP *per robot* as group sizes increased; SID displayed constant performance, but AGS consistently achieved the best ETP.

When comparing algorithm performance, the choice of metrics is crucial. Previous researchers often examined either the time cost [20] or energy consumption [9, 7] independently. Alternatively, multi-objective functions are used, combining multiple weighted metrics, e.g. Hayes [6]. However, this is not straightforward due to the choice of metrics, weighting and formulation. Additionally, although comparisons are simplified, individual metric trends are obfuscated. To clearly understand the underlying trends, we have shown both energy and time costs independently as well as the ETP to allow selection of the best energy-time efficient strategy. The ETP has equal weighting of time and energy costs providing a simple metric. However, relative weightings could be easily applied to the provided energy and time results, depending on their relative importance in different applications. The ETP facilitated comparisons between different robot group sizes during the scalability tests.

This work was confined to one type of corridor environment of a fixed size. Properties such as size and complexity or the existence of open areas may affect the performance of the three strategies and response to parameters (λ and τ). These effects were examined with LTID previously in [22]. Summarising, the gains in energy efficiency with LTID are more pronounced in higher complexity

environments with more corridor junctions. This is because robots will redeploy to new areas more frequently and experience greater interference. Therefore, it is expected that these characteristics will generalise to SID and AGS. Such complex environments are common in buildings such as offices, especially in disaster situations. It would also be feasible to autonomously optimise the control parameters based on perceived environmental conditions, a subject of future work. Finally, the effects of small obstructions (e.g., lights) on the relative-positioning sensor was neglected. Current testing indicates the sensor is robust to small obstructions and experiences only slight attenuation, while large obstacles are handled algorithmically [22].

Currently we have developed the autonomous flight behaviours of the underlying swarm search behaviour, validating the feasibility of the approach. In the future we aim to verify the presented results with real flying robots. We are also investigating methods to extend these strategies to further reduce flight energy, e.g. by selection of the closest robot to the desired destination with strategies amenable to swarm robotics [24]. Additionally we will test all strategies in more varied environments aiming to draw more general performance predictions.

In conclusion, aerial swarms are gaining interest due to their suitability for many applications such as search or disaster mitigation because they can rapidly cover obstacle-rich terrain [22]. The work presented here facilitates the future deployment of flying robots with limited autonomy to successfully cover larger environments, while understanding the impact on time costs. The three presented strategies provide different performances with different sensing and communication requirements, facilitating selection according to robot capabilities and application requirements.

Acknowledgements. This work is part of the "Swarmanoid Project", a Future Emerging Technologies (FET IST-022888) project funded by the European Commission. The authors would like to thank Sara Mitri and Thomas Schaffter for their valuable feedback and assistance.

References

1. Brue, S.L.: Retrospectives: The law of diminishing returns. The Journal of Economic Perspectives 7(3), 185–192 (1993)
2. Chang, H.J., Lee, C.S.G., Lu, Y.-H., Hu, Y.C.: Energy-time-efficient adaptive dispatching algorithms for ant-like robot systems. In: International Conference on Robotics and Automation, ICRA 2004, vol. 4, pp. 3294–3299. IEEE, Piscataway (2004)
3. Cormen, T.H., Leiserson, C.E., Rivest, R.L.: Introduction to Algorithms. MIT Press, Cambridge (1990)
4. Cory, R., Tedrake, R.: Experiments in fixed-wing UAV perching. In: Proceedings of the Guidance, Navigation, and Control Conference, pp. 1–12. AIAA, Reston (2008)
5. Dias, M., Zlot, R., Kalra, N., Stentz, A.: Market-based multirobot coordination: A survey and analysis. Proceedings of the IEEE 94(7), 1257–1270 (2006)
6. Hayes, A.T.: How many robots? Group size and efficiency in collective search tasks. In: Proceedings of the 6th Int. Symp. on Distributed Autonomous Robotic Systems, DARS 2002, pp. 289–298 (2002)

7. Howard, A., Mataric, M.J., Sukhatme, G.S.: An incremental self-deployment algorithm for mobile sensor networks. Autonomous Robots 13(2), 113–126 (2002)
8. Kovac, M., Germann, J.M., Hrzeler, C., Siegwart, R., Floreano, D.: A perching mechanism for micro aerial vehicles. Journal of Micro-Nano Mechatronics 5(3–4), 77–91 (2009)
9. Liu, W., Winfield, A., Sa, J., Chen, J., Dou, L.: Strategies for Energy Optimisation in a Swarm of Foraging Robots. In: Şahin, E., Spears, W.M., Winfield, A.F.T. (eds.) SAB 2006 Ws 2007. LNCS, vol. 4433, pp. 14–26. Springer, Heidelberg (2007)
10. Mei, Y., Lu, Y.H., Hu, Y., Lee, C.: Deployment of mobile robots with energy and timing constraints. IEEE Transactions on Robotics 22(3), 507–522 (2006)
11. Mei, Y., Lu, Y.H., Hu, Y.C., Lee, C.S.G.: Determining the fleet size of mobile robots with energy constraints. In: International Conference on Intelligent Robots and Systems, IROS 2004, vol. 2, pp. 1420–1425. IEEE Press, Piscataway (2004)
12. Mei, Y., Lu, Y.H., Hu, Y.C., Lee, C.S.G.: A case study of mobile robot's energy consumption and conservation techniques. In: Proceedings of the 12th International Conference on Advanced Robotics, ICAR 2005, pp. 492–497. IEEE, Piscataway (2005)
13. Moscibroda, T., von Rickenbach, P., Wattenhofer, R.: Analyzing the energy-latency trade-off during the deployment of sensor networks. In: Proceedings of the 25th International Conference on Computer Communications, pp. 1–13. IEEE Press, Piscataway (2006)
14. Murphy, M., Aksak, B., Sitti, M.: Gecko-inspired directional and controllable adhesion. Small 5(2), 170–175 (2009)
15. O'Hara, K.J., Nathuji, R., Raj, H., Schwan, K., Balch, T.: AutoPower: toward energy-aware software systems for distributed mobile robots. In: International Conference on Robotics and Automation, ICRA 2006, pp. 2757–2762. IEEE, Piscataway (2006)
16. Pinciroli, C.: The swarmanoid simulator. Tech. Rep. TR/IRIDIA/2007-025, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium (2007)
17. Roberts, J., Stirling, T., Zufferey, J.C., Floreano, D.: Quadrotor using minimal sensing for autonomous indoor flight. In: Proceedings of the 2007 European Micro Air Vehicle Conference and Flight Competition, EMAV 2007 (2007)
18. Roberts, J., Stirling, T., Zufferey, J.C., Floreano, D.: 2.5D infrared range and bearing system for collective robotics. In: Proceedings of the International Conference on Intelligent Robots and Systems, IROS 2009, pp. 3659–3664. IEEE, Piscataway (2009)
19. Roberts, J., Zufferey, J.C., Floreano, D.: Energy management for indoor hovering robots. In: Proceedings of the International Conference on Intelligent Robots and Systems, IROS 2008, pp. 1242–1247. IEEE, Piscataway (2008)
20. Rosenfeld, A., Kaminka, G.A., Kraus, S.: A Study of Scalability Properties in Robotic Teams. In: Coordination of Large-Scale Multiagent Systems, Part 1, pp. 27–51. Springer (2006)
21. Rybski, P., Larson, A., Lindahl, M., Gini, M.: Performance evaluation of multiple robots in a search and retrieval task. In: Proceedings of the Workshop on Artificial Intelligence and Manufacturing, pp. 153–160. AAAI Press, Menlo Park (1998)
22. Stirling, T., Wischmann, S., Floreano, D.: Energy-efficient indoor search by swarms of simulated flying robots without global information. Swarm Intelligence 4(2), 117–143 (2010)
23. Valenti, M., Bethke, B., How, J.P., Farias, D.P., Vian, J.: Embedding health management into mission tasking for UAV teams. In: American Control Conference, pp. 5777–5783. IEEE, Piscataway (2007)
24. Wang, G., Cao, G., Porta, T.L., Zhang, W.: Sensor relocation in mobile sensor networks. In: Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2005, vol. 4, pp. 2302–2312. IEEE Press, Piscataway (2005)
25. Zlot, R.M., Stentz, A., Dias, M.B., Thayer, S.: Multi-robot exploration controlled by a market economy. In: International Conference on Robotics and Automation, vol. 3, pp. 3016–3023. IEEE Press, Piscataway (2002)

A Distributed, Real–Time Approach to Multi Robot Uniform Frequency Coverage

Giorgio Cannata and Antonio Sgorbissa

Abstract. The article proposes a novel distributed solution to the problem of Multi–Robot Uniform Frequency Coverage (MRUFC in short), in which a team of robots are requested to repeatedly visit a set of pre–defined locations of the environment with uniform frequency. With respect to other algorithms in literature, the approach proposed has extremely low requirements in terms of computational power, does not require inter–robot communication, and can even be implemented on memoryless robots, thus being easily implementable on real, marketable robot swarms.

1 Introduction

The article proposes a novel distributed solution to the problem of Multi–Robot Uniform Frequency Coverage (MRUFC in short) introduced in [4, 8], in which a team of robots are requested to repeatedly visit a set of pre–defined locations of the environment with uniform frequency. The problem has a fundamental importance in many applications, e.g., surveillance and patrolling, continuous cleaning of crowded areas (malls, convention centers, restaurants, etc.), serving food or beverages (in hospitals or in a banquet). However, differently from other problems related to multi–robot coverage and exploration, it has received only a limited attention. MRUFC shares some similarities with the Dynamic Vehicle Routing problem, see for example [3] and the references therein. However the objective of DVR is different, since it aims at minimizing the expected time between the appearance of a target and the time it is visited by one of the vehicles.

Almost all traditional approaches to coverage, either single or *multi–robot*, are based on *space decomposition*, i.e, they rely on the idea that the work–area is

Giorgio Cannata · Antonio Sgorbissa
University of Genova, Via Opera Pia 13, 16145 Genova, Italy
e-mail: {giorgio.cannata,antonio.sgorbissa}@unige.it

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 19–32.
springerlink.com

decomposed into subregions and that each robot is assigned a subregion (or set of subregions) to cover [26, 2, 9, 18, 16, 1]. An exception is *spanning tree–based coverage*, which envisions a situation in which all robots periodically cover the whole environment [13, 14, 11]. *Spanning tree–based coverage* owes its importance also to the fact that, to the authors' best knowledge, it is the only approach dealing explicitly with the problem of guaranteeing that all areas are visited with *uniform frequency*.

All these approaches have the major drawbacks that they require the robots to have a complete map of the environment a priori or to build it in run–time, with obvious consequences on the computational power required on–board for sensor fusion and planning. On the opposite, in order to find solutions that are technologically feasible at the present state–of–the–art, this work makes the assumption that robots have low computational power and memory storage, and are able to operate even when wireless communication is not available or it is severely degraded.

These technological constraints naturally lead to the choice of algorithms belonging to the class of the so–called *real–time search* and *ant–like* algorithms [23, 10, 22, 28]. Algorithms of this class usually assume that robots navigate in a graph–like world which is only locally known, i.e., every robot has only access to information related to the closer vertex (and, in some cases, to adjacent vertices). Starting from these assumptions, different algorithms implement different strategies to find a path to the goal state. The simpler of these methods is perhaps *Random Walk*. Assume that the environment is modeled as an oriented graph: when the robot is in a vertex, it selects a departing edge randomly with uniform probability, which guarantees complete coverage in a statistical sense as the exploration time tends to infinite. *Edge Counting* is a deterministic variant of this idea [21], in which the robot chooses different edges in circular order in subsequent visits, therefore guaranteeing that the relative frequency of choices tends to the uniform distribution. *Node Count* [23] exhibits an improved behaviour by relying on the idea of associating a value with each vertex of the graph, which counts how often each vertex has been already visited so far. When a robot enters a vertex, it increases the value of the vertex by one: next, it moves to the adjacent vertex which has received less visits up to present time. Many variants of this simple idea exist (e.g., [27, 5, 23, 22]).

Real–time search and *ant–like* algorithms are particularly interesting in that they move most of the burden of computing and memorizing from the searching agent to the vertices of the graph: the graph itself is not only a model of the topology of the environment, but it becomes a real physical entity which can be built by leaving chemical trails on the floor [28], by dropping pebbles [10], and so on. Towards this end, some works adopt Radio Frequency IDentification (RFID) tags as a reference technology (e.g., [29, 20, 12]): RFID tags are low–cost, short–range, energetically self–sustained transponders that can be distributed in the environment and can store a limited amount of information. Smart–nodes (either implemented as RFID tags or with a similar technology) can be placed and distributed in the environment prior to robot operations, and used to build the navigation graph: each *smart node* contains navigation directions to neighboring *smart nodes*, thus allowing robots with local communication capability to safely execute paths from and to different locations.

Unfortunately, *real–time search* and *ant–like* algorithms in the literature have two major drawbacks: i) they do not deal explicitly with the problem of guaranteeing that all areas of the environment are visited with *uniform frequency*, which is the main objective of this work, and ii) in some cases, they make assumptions which are in contrast with the constraints that has been put, i.e., the absence of global representations and long–range communication capability. This is definitely the case of *Node Count* and its variants: they unrealistically require that a robot in a given vertex knows how many times neighboring vertices have been visited, which either requires an internal representation shared between all robots, or long–range communication capabilities.

The main contribution of this work is the introduction of the novel *PatrolGRAPH*^{A} algorithm, which is able to solve MRUFC under all constraints. The approach proposed shares some similarities with [17, 6], which investigate optimized stochastic policies for distributing robots among the vertices of a graph according to a prescribed distribution. These works are different in that they allow self–loops, i.e., edges departing from/arriving to the same vertex: under these conditions, the MRUFC problem can be demonstrated to have always a solution, which is not always the case when self–loops are not allowed. An additional difference, both with [17, 6] and with our previous work [4, 8] is the following: in *PatrolGRAPH*^{A} neither a centralized controller nor an off–line phase are required to computer transition rates between vertices or to alter the graph, since adaptation is performed on–line and fully decentralized during robot operation.

The paper is organized as follows. Section 2 describes the MRUFC problem in details, and it introduces the novel *PatrolGRAPH*^{A} algorithm. Section 3 shows experimental results. Conclusions follow.

2 Multi–Robot Uniform Frequency Coverage

The Multi–Robot Uniform Frequency Coverage (MRUFC) problem consists in a decision procedure which allows a team of robot to navigate in a workspace modelled as a navigation graph, ensuring that all vertices of the graph are visited with uniform frequency. Given that:

- G_N is a planar, non–oriented graph of arbitrary order, possibly containing cycles, which represents the topology of the free space, referred to as the *navigation graph*. As usual, the navigation graph is better represented through a strongly connected, oriented graph $\hat{G}_N$, derived from G_N by simply doubling all its edges and assigning them opposite directions (Figure 1).
- $S = \{s_i\}$ denotes the finite set of N vertices in $\hat{G}_N$. Each vertex s_i is associated with a location in the workspace.
- $A_i = \{a_{ij}\} \neq 0$ is the finite, nonempty set of directed edges that leave vertex $s_i \in S$. Each edge a_{ij} is defined through a couple of indices (i, j), which respectively identify the corresponding start and end vertices. $|A_i|$ is the dimension of the set, i.e., the number of edges departing from s_i.

- $R = \{r_i\}$ is a set of M robots. Robots are allowed to move in the workspace from s_i to s_j in $\hat{G}_N$ only if $a_{ij} \in A_i$, i.e., if the two vertices are adjacent.
- $\boldsymbol{\lambda} = [\lambda_1, \cdots, \lambda_N]^T$ is a vector which describes the average robot arrival rate to each vertex $s_i \in S$, expressed as *number of visits* divided by *time* ($\lambda_i \in \Re$).

The following objective must be achieved: the M robots must guarantee uniform coverage of $\hat{G}_N$, i.e., they must move in such a way that, for all s_i, $\lambda_i = \bar{\lambda}$, so that $\bar{\lambda} = \sum_{i=1}^{N} \lambda_i / N$. This basically means that all vertices in $\hat{G}_N$ must be visited with the same frequency as time passes.

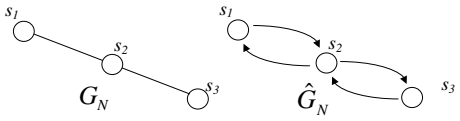


Fig. 1 The oriented graph $\hat{G}_N$ built from G_N

For a real–world implementation, it is assumed that robots are equipped with proper algorithms for vertex–to–vertex navigation, as well as for obstacle avoidance and localization. In particular, it is assumed that vertex s_i is linked to an adjacent vertex s_j through a_{ij} whenever it is possible to reach s_j starting from s_i through a "simple motion", e.g., moving along a straight line[1].

Additional implementation constraints are taken into account, which are not necessarily related to the MRUFC problem, but can play an important role to allow implementation on affordable and dependable commercial robots with minimal computational, memory, and communication capabilities.

- *Low computational cost.* The algorithm which solves MRUFC should be executable in parallel on very simple robots with limited computational power.
- *Local memory.* The graph $\hat{G}_N$ should possibly never be stored in robots memory. Instead, all the information concerning a generic vertex, as well as the edges departing from it, should be stored into a *smart node* opportunely located in the environment[2]. To help robots to physically navigate in the workspace, every *smart node* can store navigation directions to reach neighboring *smart nodes*.
- *Local communication.* Robots should be able to communicate only with *smart nodes* within a very short communication range, and to indirectly communicate with other robots by writing to/reading from *smart nodes*. It is assumed that: a robot cannot directly communicate with another robot; a *smart node* cannot directly communicate with another *smart node*; a robot cannot communicate with two *smart nodes* at the same time.

[1] The intuitive notion of "simple motion" can vary depending on the robot kinematics, localization skills, etc.

[2] Smart nodes can be implemented, for example, as active or passive RFID tags, or similar devices with local communication capabilities and a very limited memory storage.

2.1 The Basic Navigation Algorithm

In this work, the *PatrolGRAPH*^{A} algorithm is introduced, which assumes that M robots execute in parallel a particular instance of Algorithm 1 to move between adjacent vertices within $\hat{G}_N$.

Algorithm 1. Navigation Algorithm - PatrolGRAPH^A

```
1: s_c := s_start
2: while TRUE do
3:     a_cl := choose(s_c, PatrolGRAPH^A)
4:     Move along edge a_cl
5:     s_l := succ(s_c, a_cl)
6:     update(p_lc)
7:     s_c := s_l
8: end while
```

Algorithm 1 itself is straighforward. Line 1 chooses an arbitrary start vertex s_{start}, that can be different for different robots. When the robot is in vertex s_c, the operator $choose(s_c, Alg)$ in Line 3 returns one of the directed edges $a_{cl} \in A_c$, according to a strategy that depends on the Algorithm Alg which identifies a specific navigation strategy. In particular, by temporarily ignoring Line 6, Algorithm 1 constitutes a basis to describe also the well–known *Random Walk* and *Edge Counting* algorithms, whose behaviour in solving the MRUFC problem has been described in [4], as well as the *PatrolGRAPH** algorithm, introduced in [8].

In a real–world implementation, the operator $choose(s_c, Alg)$ requires the robot to communicate with the *smart node* which stores information about vertex s_c, and to retrieve navigation directions that provide guidance to move towards the next vertex. Line 4 summarizes all procedures that are requested for the robot to move to the next vertex (including motion control, obstacle avoidance, localization, etc.). As discussed in the previous Section, it is assumed that each robot is capable to move along edge a_{cl} and to reach the next *smart node* correctly, i.e., it is equipped with proper hardware and software subsystems for achieving this. The operator $succ(s_c, a_{cl})$ in Line 5 returns the successor vertex that results from the traversal of edge $a_{cl} \in A_c$ starting from vertex $s_c \in \hat{G}_N$. In practice the output of this operator is the effect of the actual robot motion.

2.2 PatrolGRAPH^A: Policy to Choose the Next Edge

In order to describe the implementation of the operator $choose(s_c, PatrolGRAPH^A)$, shown in Algorithm 2, it is necessary to define:

- v_i as an integer variable initialized to 0 which counts the overall number of visits to vertex s_i;

- k_{ij} as an integer variable initialized to 0 which counts the number of times that robots have chosen to proceed to s_j after leaving s_i;
- p_{ij} as a real variable which describes the desired ratio of robots that, after visiting s_i, must head towards s_j.

In practice, the set of all p_{ij}, $(i, j = 1 \ldots N)$, describes a transition matrix P associated to the graph $\hat{G}_N$. P is necessarily a *stochastic matrix* [15], i.e., it is subject to the following constraints:

$$\sum_{j=1}^{N} p_{ij} = 1, \quad (i = 1, \ldots, N) \tag{1}$$

$$0 \leq p_{ij} \leq 1, \quad (i, j = 1, \ldots, N). \tag{2}$$

Algorithm 2. Operator $choose(s_c, PatrolGRAPH^A)$

1: $v_c := v_c + 1$
2: **for all** j such that $a_{cj} \in A_c$ **do**
3: $\quad \Delta p_{cj} := k_{cj}/v_c - p_{cj}$
4: **end for**
5: $l := \arg\min_j(\Delta p_{cj})$
6: $k_{cl} := k_{cl} + 1$
7: **return** a_{cl}

The operator $choose(s_c, PatrolGRAPH^A)$ is shown in Algorithm 2: Line 1 updates the number of visits v_c received by s_c; Line 3 computes, for every adjacent vertex, the error Δp_{cj} between the ratio k_{cj}/v_c and the desired relative frequency p_{cj}; Line 5 picks the edge a_{cl} for which Δp_{cj} is minimum; Line 6 updates k_{cl}. It has been demonstrated [8] that Algorithm 2 guarantees that the value $k_{ij}/v_i - p_{ij} \to 0$ as the number of visits $v_i \to \infty$, $(i, j = 1, \ldots, N)$. That is, for every edge a_{ij}, the relative frequency of the choice k_{ij}/v_i tends to the desired transition probability p_{ij}.

In order to analyse the behaviour of $PatrolGRAPH^A$ with M robots executing Algorithm 1 in parallel, it is convenient to model the system as a Closed Queueing Network (CQN), a dynamical model usually found in application domains like process automation, communication networks, etc.[7], with the purpose of describing and analyzing how *service centers* are allocated to *customers* in time. In particular, CQNs are specified by:

- the number N of service centers s_i (corresponding to vertices of $\hat{G}_N$);
- the number M of customers (corresponding to robots);
- the average arrival/departure rate λ_i to/from node s_i (i.e., expressed as *number of arriving customers* divided by *time*). The arrival and departure rate in s_i are the same in a CQN (and obviously correspond to the visiting rate), since customers cannot exit or enter the network from outside;
- the number of servers m_i running in parallel at s_i (i.e., how many robots can perform the assigned task in s_i at the same time), the time t_i requested to complete

a task in s_i and the maximum service rate $\mu_i = 1/t_i$ at node s_i (expressed as *number of robots which can complete the task* per *time unit*);
- a routing policy, expressed through a $N \times N$ transition matrix P.

In a CQN, the average arrival rate at steady state equals the average departure rate for every vertex. Then, it is possible to write flow balance equations:

$$\lambda_i = \sum_{j=1}^{N} p_{ji}\lambda_j, \quad (i = 1,\ldots,N). \tag{3}$$

By referring to the transition matrix P, (3) can be re–written in matrix form as:

$$(P^T - I)\boldsymbol{\lambda} = 0. \tag{4}$$

In our case, in addition to being stochastic, P is irreducible, since $\hat{G}_N$ is strongly connected by definition (i.e., it is possible to reach every vertex from every other vertex in a finite number of transitions). According to the Perron–Frobenius Theorem [15], the eigenvector problem above has ∞^1 solutions associated with the eigenvalue 1 as long as P is a stochastic irreducible matrix. Then, the N components of the eigenvector can be expressed as a function of one arbitrary parameter $\bar{\lambda}$.

Equations (3) and (4) are very important since they state that, in order to achieve a uniform distribution of average arrival rates $\boldsymbol{\lambda}$ at steady state, the only controllable variables are the elements of the transition matrix P. As a corollary, it follows that the parameters m_i (the number of robots m_i allowed to perform the assigned task is s_i at the same time), t_i (the time requested to complete a task in s_i) and t_{ij} (the navigation time from s_i to s_j) do not play any role in determining the mutual relationships between the components of $\boldsymbol{\lambda}$ at steady state. Obviously, m_i, t_i, and t_{ij} play a role in determining the actual value of the arrival rate: for example, if the navigation time t_{ij} between vertices increases, the average arrival rate to each vertex necessarily decreases. However, the fact that, for example, $\lambda_3 = 2\lambda_2$ (i.e., the arrival rate at s_3 is doubled with respect to s_2) depends exclusively on P.

2.3 *PatrolGRAPH^A: Policy to Set the Elements of P*

In [4, 8] it has been shown that *Random Walk* and *Edge Counting* cannot provide a general solution to the MRUFC problem. This also follows from known result in Markov Chains theory [24]: when using of these algorithms, the average rate of visits λ_i that each vertex s_i of $\hat{G}_N$ receives as time passes is proportional to the number of incident edges $|A_i|$ of s_i.

More specifically, it is a known result that $\boldsymbol{\lambda}$ has a uniform distribution if and only if the transition matrix P is a doubly stochastic (bistochastic) matrix [15], i.e., its rows as well as its columns sum up to 1. This requires P to be subject to the following additional constraints:

$$\sum_{j=1}^{N} p_{ji} = 1, \quad (i = 1,\ldots,N). \tag{5}$$

In principle, for a fully connected graph, a matrix P subject to all the constraints in (1), (2), and (5) can be easily found, e.g.:

$$p_{ij} = \frac{1}{N}, \quad (i,j = 1,\ldots,N). \tag{6}$$

In practice, since the MRUFC problem assumes that the topology of the navigation graph $\hat{G}_N$ is given a priori, it is not possible to arbitrary assign all entries of P. In particular, p_{ij} can be assigned a non–zero value only if the couple of vertices s_i, s_j in $\hat{G}_N$ are adjacent; when an edge from s_i to s_j does not exist, the corresponding entry in matrix P is constrained to be zero, and the MRUFC problem might have no solutions. This happens, for example, in the very simple case in Figure 1 on the right, where only p_{12}, p_{21}, p_{23}, and p_{32} can be arbitrarily assigned, whereas all the remaining elements are constrained to be zero. It is easy to verify that s_2 receives more visits than s_1 and s_3, since every path between the latter vertices necessarily passes through the former, which is in the middle between them. Notice also that, if one is allowed to modify the topology of the graph, a doubly stochastic transition matrix can be obtained through the so–called Metropolis rule [24], which basically consists in adding self–loops, i.e., edges which depart from / arrive to the same vertex. However, self–loops are an obvious waste of time, and therefore they should be avoided.

Since an exact solution to the MRUFC problem is not guaranteed to exist when the topology of $\hat{G}_N$ is unmodifiable, one could be tempted to search for a solution in the sense of the least squares. This is the solution adopted by the *PatrolGRAPH** algorithm proposed in [8], which tries to find an approximate solution to the so–called inverse problem [19] under topological constraints, i.e., it searches for a matrix P whose stationary distribution approximately corresponds to the uniform distribution.

By writing $\boldsymbol{\lambda} = \boldsymbol{\lambda}(P)$ to stress the dependency between $\boldsymbol{\lambda}$ and P, the *PatrolGRAPH** algorithm searches for a matrix P which minimizes the following:

$$E = \sum_{i=1}^{N} \left(\lambda_i(P) - \bar{\lambda}\right)^2, \tag{7}$$

subject to a set of constraints which guarantee that, during the minimization process, the topology of $\hat{G}_N$ is preserved and P is always an irreducible stochastic matrix. It can be observed that (7) equals zero if and only if $\lambda_i(P) = \bar{\lambda}$, $(i = 1\ldots N)$.

Unfortunately, in order to compute all elements of the transition matrix P and to store them into the smart nodes, *PatrolGRAPH** requires an off–line centralized phase prior to robot operation, thus partially conflicting with the constraints that have been put at the beginning of this Section: the absence of global representations and long–range communication capabilities.

In order to avoid the off–line computation of the transition matrix P, which is aimed at guaranteeing a uniform steady state distribution, *PatrolGRAPH*^{A} allows

the robots themselves to modify the entries of the transition matrix P as they move along the graph. Consider Line 6 in Algorithm 1. Whenever a robot reaches a new vertex s_l in Line 6, it is now allowed to update the element p_{lc} of the transition matrix P, i.e., the transition probability from s_l to the previously visited vertex s_c. Notice that, since p_{lc} is stored in the *smart node* corresponding to the current vertex s_l, only local communication is required to perform this update.

Specifically, by recalling that $|A_i|$ is the number of edges departing from s_i, elements p_{ij} are initialized as follows: if s_i and s_j are adjacent $p_{ij} = 1/|A_i|$; whereas, if s_i and s_j are not adjacent, $p_{ij} = 0$. During operations, if and only if a robot is in vertex s_l coming from s_c, it is allowed to update p_{lc} through the operator $updateP(p_{lc})$ shown in Algorithm 3.

Algorithm 3. Operator $updateP(p_{lc})$

1: $K_1, K_2 := constant$
2: $\Delta v_{lc} := (v_l - v_c)/v_l$
3: **if** $\Delta v_{lc} > 0$ **then**
4: $\quad p_{lc} := \min(1-\varepsilon, p_{lc} + K_1\ \Delta v_{lc} e^{-K_2\ v_l})$
5: **else**
6: $\quad p_{lc} := \max(\varepsilon, p_{lc} + K_1\ \Delta v_{lc} e^{-K_2\ v_l})$
7: **end if**
8: **for all** $j = 1$ to N **do**
9: $\quad p_{lj} := p_{lj} / \sum_{j=1}^{N} p_{lj}$
10: **end for**

Line 1 chooses a constant value for K_1 and K_2 (to be experimentally tuned). Line 2 computes the difference Δv_{lc} between the number of visits received by the current vertex s_l and those received by s_c (i.e., the last vertex visisted by the robot before heading to s_l). Next, if $\Delta v_{lc} > 0$, Line 4 increases the value of p_{lc}, up to the maximum value $p_{lc} = 1 - \varepsilon$, with $\varepsilon > 0$. Otherwise, Line 6 decreases the value of p_{lc}, down to the minimum value $p_{lc} = \varepsilon$. Notice that the correction made to p_{lc} exponentially decreases (in absolute value) as robots visit s_l in subsequent times, i.e., as v_l increases. The upper and lower bounds in Lines 4 and 6 are motivated by the fact that the topology of the graph should never be altered, which would happen by setting $p_{ij} = 0$ for some edge. If, at some time, p_{ij} is decreased to 0, the graph could be disconnected forever: in fact, if the only path to s_j passes through s_i (i.e., s_j is not connected to any other vertex), it is possible that s_j will never be visited in the following, thus loosing forever the possibility to re–establish the deleted link. Finally, Line 8 normalizes the sum of all probabilities p_{lj} to 1 in order to re–establish the constraints in (1).

The behaviour of *PatrolGRAPH*^{A} is not formally demonstrated here, and only a brief explanation of the underlying ideas is given. These ideas will be validated through experiments in Section 3. Basically, Algorithm 3 searches for a distributed solution to the minimization problem in (7). The rationale for achieveing this is the following: when a robot is in s_l, it increases or decreases the average flow of robots $\lambda_{lc}(P) = p_{lc}\lambda_c(P)$ between s_l and s_c of a quantity which depends on the

difference between $\lambda_l(P)$ and $\lambda_c(P)$, towards the end of achieving $\lambda_l(P) \approx \lambda_c(P)$. In addition, in order to make P converge, similarly to *Simulated Annealing* [25], the quantity added to / subtracted from $\lambda_{lc}(P)$ decreases exponentially as the number of iterations increases, i.e., as s_l receives more visits.

It should be noticed that, by recalling that $\bar{\lambda} = \sum_{i=1}^{N} \lambda_i / N$ and by writing sums in expanded form, the function to be minimized in (7) can be re–written as:

$$\begin{aligned} E = & \frac{(\lambda_1(P) - \lambda_1(P) + \ldots + \lambda_1(P) - \lambda_N(P))^2}{N^2} + \ldots + \\ & + \frac{(\lambda_l(P) - \lambda_1(P) + \ldots + \lambda_l(P) - \lambda_c(P) + \ldots + \lambda_l(P) - \lambda_N(P))^2}{N^2} + \ldots + \\ & + \frac{(\lambda_N(P) - \lambda_1(P) + \ldots + \lambda_N(P) - \lambda_N(P))^2}{N^2}, \end{aligned} \tag{8}$$

which clearly shows that, by decreasing the difference (in absolute value) of $\lambda_l(P) - \lambda_c(P)$ in the second line of (8), one is not guaranteed that the overall value decreases. In fact, when locally operating on p_{lc}, this has a global effect on the eigenvectors of P, and it is possible that another term of the sum increases (in absoulte value). The working hypothesis is that, when locally operating on p_{lc} according to Algorithm 3, the overall value in (8) decreases more (on average) than it increases. This can can be said also as follows: the probability that the overall value in (8) decreases is higher than the probability that it increases. If this hypothesis is true, Algorithm 3 is expected to behave similarly to *Simulated Annealing* with an energy function E corresponding to (7), thus guaranteeing convergence to the global minimum as time tends to infinite.

3 Simulated Experiments

In this Section an assessment of the properties of *PatrolGRAPH*^{A} is presented. Since the focus of this article is on the theoretical properties, simulated experiments have been considered.

Specifically, *PatrolGRAPH*^{A} is compared with the following algorithms: *Random Walk*, which does not solve MRUFC and therefore is used as a reference; *PatrolGRAPH** [8], which yields an optimal solution to the minimization problem in (7), but requires an off–line phase; *Node Count* [23], which has more requirements than all previous algorithms since it needs to know in advance the number of visits received by all neighbouring vertices. Specifically, *PatrolGRAPH** requires to know only the number of visits received by the present vertex, whereas *PatrolGRAPH*^{A} requires to remember this information also for the last visited vertex (Line 2 of Algorithm 3).

Algorithms are compared by running them on a set of 50 randomly generated graphs. Two types of graphs have been considered:

1. graphs with a grid–like topology, built by deleting randomly chosen vertices and edges from a grid (Figure 2 on the left);
2. graphs with a grid–like topology and no local cut nodes, i.e., nodes whose removal disconnects the graph (Figure 2 on the right).

Graphs of type 2 deserve a particular attention, since the absence of local cut nodes is a requirement for *spanning tree–based coverage* [13, 14, 11]. This is the only other approach in the literature, to the best of our knowledge, dealing explicitly with the MRUFC problem.

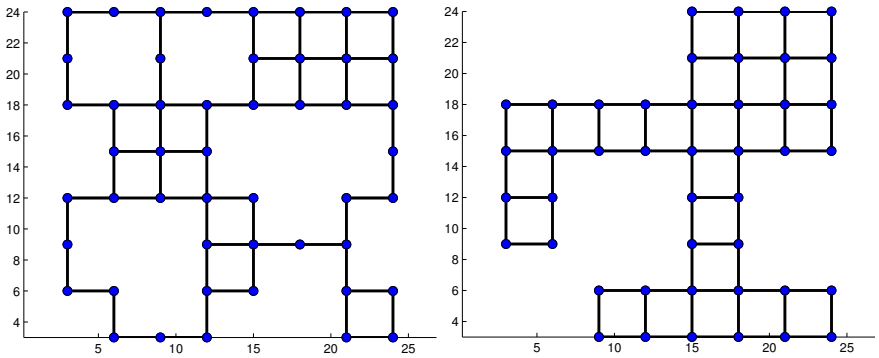


Fig. 2 Randomly generated non–oriented graphs used in experiments. Left: grid–like topology, right: grid–like topology with no local cut nodes.

Experiments are performed in the following way:

- starting positions are randomly chosen for all robots;
- the navigation time between vertices depends on the travelled distance; a random delay possibly due to traffic or navigation errors is added;
- the time required to perform a task in a vertex is assumed to be constant; a delay is randomly computed and added to the task execution time;
- the same experiment is executed with a varying number M of robots ($M = 5, 10, 20$).

The typical result of a MRUFC simulation run is shown in Figure 3, which illustrates experiments with the randomly generated graphs in Figure 2. Figure 3 shows, for every algorithm, the plot of $v_i(t)$ versus t; every curve corresponds to a different vertex s_i, and therefore the ideal situation is when all curves are "almost straight" lines with the same slope.

To compare algorithms, the following quantities are recorded every time step t :

- the number of visits $v_i(t)$ received by every vertex s_i up to time t;
- the visiting rate $\lambda_i^{\Delta t}(t)$ to every vertex s_i, averaged over the last Δt time steps as follows:

$$\lambda_i^{\Delta t}(t) = \frac{v_i(t) - v_i(t - \Delta t)}{\Delta t}. \tag{9}$$

Every t, the mean value $\lambda_m^{\Delta t}(t)$ of $\lambda_i^{\Delta t}(t)$ is computed, averaged over all vertices s_i, as well the corresponding standard deviation $\sigma_{\lambda^{\Delta t}}(t)$. The ideal situation is when the *coefficient of variation* $\sigma_{\lambda^{\Delta t}}(t)/\lambda_m^{\Delta t}(t)$ is almost null every t, i.e., the average visiting rate is almost the same for all vertices. The interval Δt is chosen in such a

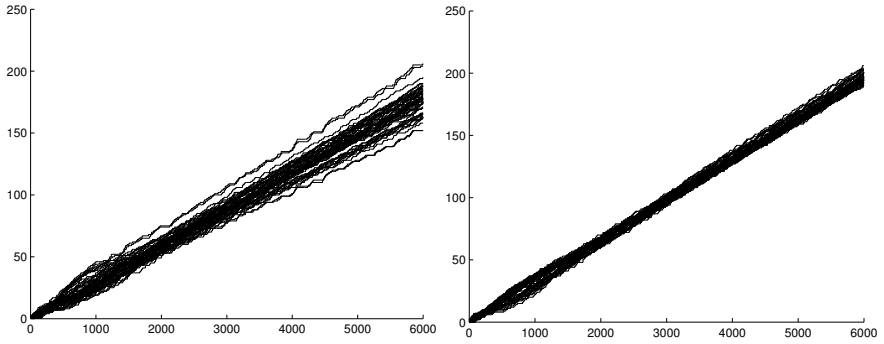


Fig. 3 Number of visits $v_1(t), \ldots, v_N(t)$ received by vertices in G_N (vertical axis) versus time (horizontal axis) with the graphs shown in Figure 2. Every curve corresponds to a vertex s_i. Left: grid–like topology, right: grid–like topology with no local cut nodes.

Table 1 MRUFC – Summary of Simulated Results

	Graph of Type 1			Graph of Type 2		
N. of robots	5	10	20	5	10	20
Random Walk	1.11	0.94	0.80	1.01	0.86	0.74
PatrolGRAPH*	0.30	0.26	0.22	0.28	0.23	0.21
PatrolGRAPH^A	0.42	0.31	0.23	0.40	0.28	0.21
Node Count	0.49	0.54	0.44	0.31	0.30	0.29

way that, on average, every vertex is visited 5 times in Δt. Informally speaking, if a test returns a coefficient of variation $\approx 1/5$, this means that the number of visits received by a vertex in Δt time steps is included in the range 5 ± 1 most of the time.

Table 1 summarizes results: all algorithms are executed on 50 randomly generated graphs of each type in Figure 2, and every test is repeated with 5, 10, and 20 robots. Each cell contains aggregate information about a set of 50 tests executed with a given *algorithm/graph topology/number of robot*. In particular, the values written in the cells correspond to the 90th percentile of the *coefficient of variation* $\sigma_{\lambda^{\Delta t}}(t)/\lambda_m^{\Delta t}(t)$, i.e., meaning that the *coefficient of variation* has been below that value 90% of the time.

It can be observed that, in almost all experiments, *Random Walk* has significantly worse performance than all other algorithms. The performance of *PatrolGRAPH*^{A} gets closer to *PatrolGRAPH** as the number of robots increases: in fact, with a higher number of robots, the convergence to the minimum of (7) is faster. By executing experiments with a higher number of time steps $\gg 6000$, *PatrolGRAPH*^{A} always converges to *PatrolGRAPH**, with a transient behaviour which depends on the values of K_1 and K_2 in Algorithm 3. *Node Count* is usually worse than *PatrolGRAPH*^{A} and *PatrolGRAPH**. Before *PatrolGRAPH*^{A} has converged, *Node Count* sometimes behaves better, thanks to the fact that a robot in s_i irrealistically knows how many times neighbouring vertices have been visited.

4 Conclusions

This article introduces *PatrolGRAPH*^{A}, a distributed real–time algorithm which solves the Multi–Robot Uniform Frequency Coverage problem. *PatrolGRAPH*^{A} can be implemented by distributing *smart nodes* with proper characteristics, such as passive (energetically self–sustaining) RFID tags with reduced memory storage capabilities and communication range. This technology allows one to meet all the additional constraints of *Low computational cost*, *Local memory*, *Local communication*, which can play an important role in real world implementations.

With respect to the *Node Count* and the *PatrolGRAPH** algorithms (whose ability to solve MRUFC has been discussed in [4, 8]), *PatrolGRAPH*^{A} offers many advantages: specifically, it does not assume the look ahead capabilities of *Node Count*, which requires to know in advance the number of visits received by neighbouring vertices, and it does not require an off–line phase to properly compute the elements of the transition matrix *P*. To the best of our knowledge, *PatrolGRAPH*^{A} is the simplest algorithm in the literature guaranteeing a uniform distribution of visits over the vertices of a graph.

The properties of *PatrolGRAPH*^{A} have been validated through simulated experiments, showing a suboptimal behaviour which is however very close to the optimal behaviour achieved by *PatrolGRAPH**.

References

1. Ahmadi, M., Stone, P.: A multi-robot system for continuous area sweeping tasks. In: Proc. of the IEEE Int. Conf. on Robotics and Automation, pp. 1724–1729 (2006)
2. Arkin, E., Fekete, S., Mitchell, J.: Approximation algorithms for lawn mowing and milling. Tech. Rep. 255, Mathematisches Institut, Universität zu Köln (1997), `ftp.zpr.uni-koeln.de/pub/paper/zpr97-255.ps.gz`
3. Arsie, A., Savla, K., Frazzoli, E.: Efficient routing algorithms for multiple vehicles with no explicit communications. IEEE Trans. on Automatic Control 54(10), 2302–2317 (2009)
4. Baglietto, M., Cannata, G., Capezio, F., Sgorbissa, A.: Multi-robot continuous coverage of significant locations in the environment. In: Asama, H., Kurokawa, H., Ota, J., Sekiyama, K. (eds.) Distributed Autonomous Robotic Systems 8, pp. 3–14. Springer, Heidelberg (2009)
5. Balch, T.: Avoiding the past: a simple but effective strategy for reactive navigation. In: Proc. of the IEEE Int. Conf. on Robotics and Automation, vol. 1, pp. 678–685 (1993), doi:10.1109/ROBOT.1993.292057
6. Bermana, S., Halasz, A., Hsieh, M.A., Kumar, V.: Optimal stochastic policies for task allocation in swarms of robots. IEEE Trans. on Robotics 25(4), 927–937 (2009)
7. Bolch, G., Greiner, S., de Meer, H., Trivedi, K.: Queueing Networks and Markov Chains. John Wiley and Sons (1998)
8. Cannata, G., Sgorbissa, A.: Minimalist algorithms for multi–robot continuous coverage. Tech. rep., DIST, Dipartimento di Informatica, Sistemistica e Telematica, Università di Genova, Genova (2010), `http://www.robotics.laboratorium.dist.unige.it/index.php?section=5` (accepted for publication on IEEE Transactions on Robotics)

9. Choset, H.: Coverage path planning: The boustrophedon cellular decomposition. In: Int. Conf. on Field and Service Robotics (1997)
10. Dudek, G., Jenkin, M., Milios, E., Wilkes, D.: Robotic exploration as graph construction. IEEE Trans. on Robotics and Automation 7(6), 859–865 (1991), doi:10.1109/70.105395
11. Elmaliach, Y., Agmon, N., Kaminka, G.: Multi-robot area patrol under frequency constraints. In: Proc. of the IEEE Int. Conf. on Robotics and Automation, pp. 385–390 (2007), doi:10.1109/ROBOT.2007.363817
12. Ferranti, E., Trigoni, N.: Robot-assisted discovery of evacuation routes in emergency scenarios. In: Proc. of the IEEE Int. Conf. on Robotics and Automation, pp. 2824–2830 (2008), doi:10.1109/ROBOT.2008.4543638
13. Gabriely, Y., Rimon, E.: Spanning-tree based coverage of continuous areas by a mobile robot. Annals of Mathematics and Artificial Intelligence 31(1-4), 77–98 (2001)
14. Gabriely, Y., Rimon, E.: Competitive on-line coverage of grid environments by a mobile robot. Comput. Geom. Theory Appl. 24(3), 197–224 (2003), doi:http://dx.doi.org/10.1016/S0925-77210200110-4
15. Godsil, C., Royle, G.: Algebraic Graph Theory Chains. Springer (2001)
16. Guo, Y., Parker, L., Madhavan, R.: A multi-robot system for continuous area sweeping tasks. In: Proc. of the Int. Symp. on Collaborative Technologies and Systems, pp. 235–240 (2004)
17. Hsieh, M.A., Bermana, S., Halasz, A., Kumar, V.: Biologically inspired redistribution of a swarm of robots among multiple sites. Swarm Intelligence 2(2-4), 121–141 (2008)
18. Huang, W.: Optimal line-sweep-based decompositions for coverage algorithms. In: Proc. of the IEEE Int. Conf. on Robotics and Automation, vol. 1, pp. 27–32 (2001), doi:10.1109/ROBOT, 932525
19. Karr, A.F.: Markov chains and processes with a prescribed invariant measure. Stochastic Processes and their Applications 7(3), 277–290 (1978)
20. Kleiner, A., Prediger, J., Nebel, B.: Rfid technology-based exploration and slam for search and rescue. Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS 2006 pp. 4054–4059 (2006), doi:10.1109/IROS.2006.281867
21. Koenig, S., Simmons, R.: Easy and hard testbeds for real-time search algorithms. In: Proc. of the Nat. Conf. on Artificial Intelligence, pp. 279–285 (1996)
22. Koenig, S., Szymanski, B., Liu, Y.: Efficient and inefficient ant coverage methods. Annals of Mathematics and Artificial Intelligence 31(1-4), 41–76 (2001)
23. Korf, R.: Real-time heuristic search. Artif. Intell. 42(2-3), 189–211 (1990), doi:http://dx.doi.org/10.1016/0004-37029090054-4
24. Levin, D.A., Peres, Y., Wilmer, E.L.: Markov chains and mixing times. American Mathematical Society (2008)
25. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E.: Equation of state calculations by fast computing machines. The Journal of Chemical Physics 21(6), 1087–1092 (1953)
26. Ntafos, S.: Watchman routes under limited visibility. Comput. Geom. Theory Appl. 1(3), 149–170 (1992), doi:http://dx.doi.org/10.1016/S0925-77219900022-X
27. Pirzadeh, A., Snyder, W.: A unified solution to coverage and search in explored and unexplored terrains using indirect control. In: Proc. of the IEEE Int. Conf. on Robotics and Automation, vol. 3, pp. 2113–2119 (1990), doi:10.1109/ROBOT.1990.126317
28. Wagner, I., Lindenbaum, M., Bruckstein, A.: Distributed covering by ant-robots using evaporating traces. IEEE Trans. on Robotics and Automation 15(5), 918–933 (1999), doi:10.1109/70.795795
29. Ziparo, V., Kleiner, A., Nebel, B., Nardi, D.: Rfid-based exploration for large robot teams. In: Proc. of the IEEE Int. Conf. on Robotics and Automation, pp. 4606–4613 (2007), doi:10.1109/ROBOT.2007.364189

Connectivity Maintenance of a Heterogeneous Sensor Network

Randy Andres Cortez, Rafael Fierro, and John Wood

Abstract. In this paper we derive connectivity constraints for a heterogeneous sensor network made up of sensing agents and mobile communication relays. With these constraints we develop feasible motion sets that can guarantee network connectivity. We also show how to reduce the number of communication constraints to allow the sensing agents to maximize their feasible motion sets and thus allow for a larger search area while maintaining network connectivity. A technique for shaping the network configuration is also presented that allows for biasing particular communication links within the network. Numerical simulations and preliminary experimental results verify the validity of the proposed approach.

1 Introduction

Recently in the literature, connectivity maintenance has been considered as a constraint on the reconfigurable sensor network. The constraint of maintaining connectivity between sensor nodes is a relaxation to the typically assumed fixed communication topology. The connectivity constraint complicates the motion planning problem for the sensor network in the sense that sensors should only move to areas in the search space where communication can be guaranteed. Typically the connectivity constraint is directly imposed on the sensor network, which may greatly limit its ability to investigate the search space. To overcome this constraint on the reconfigurable sensor network, we propose to add mobile communication "relay" agents

Randy Andres Cortez · John Wood
Mechanical Engineering Department, University of New Mexico,
Albuquerque, New Mexico 87131
e-mail: {acortez,jw}@unm.edu

Rafael Fierro
Electrical and Computer Engineering Department, University of New Mexico,
Albuquerque, New Mexico 87131
e-mail: rfierro@ece.unm.edu

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 33–46.
springerlink.com

to the communication topology. This allows for the sensor network to have a "longer reach" to investigate the search space, with the added difficulty of having to control a heterogeneous team of robots (sensors and relays).

Research in multi-robot coordination typically assumes that the underlying communication topology is fixed and connected, [1], [2]. Recently however, research has begun to focus on a relaxation of this assumption, namely considering the connectivity of the multi-robot group as being a dynamic topology which should maintain some connectivity properties. In [3], Dimarogonas and Johansson present a distributed control law that guarantees connectivity maintenance in a network of multiple mobile agents. The control law is achieved through a potential field approach with guaranteed boundedness on the agents input. Michael *et.al.,* [4] implement a control algorithm that is based on a consensus approach and market-based auctions on a group of mobile robots. The local connectivity of the group is estimated by computing the second smallest eigenvalue of the graph Laplacian, similar to the work in Kim and Mesbahi [5]. In [6], Ji and Egerstedt address maintaining connectivity in rendezvous and formation control problems. Fink and Kumar [7] explore methods for online mapping of Received Signal Strength Indicator (RSSI) with mobile robots where the RSSI map can then be used for control algorithms requiring inter-robot communications. Tekdas *et. al.,* [8] study the problem of computing the minimum number of robotic routers in order to maintain connectivity of a single user to a base station. In [9] the authors derive a flocking controller to regulate the distance between vehicles that address coverage and vehicles that address coordination. The distance requirements for the flocking controller are the communication range of the vehicle types. In [10] the authors develop a distributed controller to position a team of UAVs in a configuration that optimizes communication-link quality to support a team of UGVs performing a collaborative task, however they must assume the UGVs have zero dynamics to guarantee the connectivity of the combined UAV, UGV network. Stachura and Frew [11] use a fixed planning hierarchy for a finite horizon optimization that addresses cooperative target localization with communication considerations.

In our approach to multi-robot connectivity maintenance we propose adding specialized agents that are better equipped (hardware) to relay information over longer distances to the sensor network. This allows the sensor network to have a longer "reach" in the search space. It also allows the sensing agents to be built in a way that the communication hardware can be minimized.

2 Problem Formulation

We begin by considering a heterogeneous team of agents consisting of n sensing agents and m relay agents in two and three dimensions. Assume the n sensing agents are equipped with sensors capable of sensing the environment within a finite radius R_s and communicating within a finite radius $R_c(q) \leq R_{c_{max}}$. Here we assume that the communication radius will change based on the positions of the robots. This

relaxation in the communication range allows us to model, to some degree, the path loss in the communication channel [12]. Incorporating communication channel characteristics, which has been largely ignored in the literature to date, allows for a better system model. Also let us assume that the m relay agents are capable of communicating over a finite radius R_{rc} such that $R_{rc} > R_{c_{max}}$ *i.e.,* the relay robots are better equipped for communication than the sensing agents and the relay robots communication range is not dependent on location. Consider the area of interest Q, assumed to be a simple convex polygon with boundary ∂Q, including its interior. For our mathematical formulation we consider each agent x_i to have the following dynamics:

$$\dot{x}_i = Ax_i + Bu_i, \tag{1}$$

where A is the system matrix, B is the input matrix, u_i is the input, and $i = 1, \cdots, n+m$. We are assuming a linear controllable system under the premise that the dynamics from both ground vehicles as well as aerial vehicles with an autopilot system can be conservatively estimated in such a way. Here we are considering our sensor network to be heterogeneous not only because relay and sensing agents have different communication ranges but also because they play different roles in the sensor network.

3 Communication Constraints

In our scenario there exist three particular communication link possibilities. The first being, *relay/sensor* communication, where a sensor communicates directly to a relay agent. The second, *relay/relay* communication, where a relay shares a communication link with another relay agent. The last communication link possibility is *sensor/sensor* communication where sensors communicate directly with each other. For the following formulation let us consider the case where the communication radius of the sensing robots is not location dependent, *i.e.,* $R_c(q) = R_c$. Also, let each agent's communication range denote the range over which the agent can both send and receive information.

Similar to the work on homogeneous networks of Bullo *et al.,* [13], we now formulate the *connectivity constraint set* for each particular communication link possibility of our heterogeneous network based on the geometry of the communication radii. For the following definitions we will use $\bar{\mathscr{B}}(p,r)$ to denote a closed ball of radius r centered at p in $\mathbb{R}^2$.

Definition 1. (*Relay/Sensor* connectivity constraint set) Consider two agents, one relay agent i located at position p_i and one sensing agent j located at position p_j such that $||p_i - p_j||_2 \leq R_{rc}$. Then the connectivity constraint set of agent i with respect to agent j is

$$\Upsilon_{\mathrm{d}_{rs}}(p_i, p_j) = \bar{\mathscr{B}}(\frac{p_i + p_j}{2}, \frac{R_{rc}}{2}). \tag{2}$$

Definition 2. (*Relay/Relay* connectivity constraint set) Consider two relay agents, one agent i located at position p_i and one agent j located at position p_j such that $\left\|p_i - p_j\right\|_2 \leq R_{rc}$. Then the connectivity constraint set of agent i with respect to agent j is

$$\Upsilon_{\mathrm{d}_{rr}}(p_i, p_j) = \bar{\mathscr{B}}(\frac{p_i + p_j}{2}, \frac{R_{rc}}{2}). \tag{3}$$

Definition 3. (*Sensor/Sensor* connectivity constraint set) Consider two sensing agents, one agent i located at position p_i and one agent j located at position p_j such that $\left\|p_i - p_j\right\|_2 \leq R_c$. Then the connectivity constraint set of agent i with respect to agent j is

$$\Upsilon_{\mathrm{d}_{ss}}(p_i, p_j) = \bar{\mathscr{B}}(\frac{p_i + p_j}{2}, \frac{R_c}{2}). \tag{4}$$

Definition 4. (Connectivity constraint set for relay agent w.r.t. heterogeneous network) Consider a group of agents containing both sensing and relay agents located at $\mathscr{P} = \{p_1, p_2, \ldots, p_{n+m}\}$. Then the connectivity constraint set of relay agent i with respect to all other agents in the group is

$$\Upsilon_{\mathrm{d}_{hr}}(p_i, \mathscr{P}) = \{x \in \Upsilon_{\mathrm{d}_{rr}}(p_i, p_j) | q \in \mathscr{P} \backslash \{p_i\} \text{ s.t. } \|q - p_i\|_2 \leq R_{rc}\}. \tag{5}$$

Before we can state the definition of the connectivity constraint set for a sensing agent with respect to the heterogeneous network we need some preliminaries. Let p_i be a sensing agent, then

$$\Lambda_{ss} = \cap_{j=1}^{n} \Upsilon_{ss}(p_i, p_j), \text{ where } p_j \in \text{ sensors}, \tag{6}$$

$$\Lambda_{sr} = \cap_{k=1}^{m} \Upsilon_{sr}(p_i, p_k), \text{ where } p_k \in \text{ relays}. \tag{7}$$

Definition 5. (Connectivity constraint set for sensor agent w.r.t. heterogeneous network) Consider a group of agents containing both sensing and relay agents located at $\mathscr{P} = \{p_1, p_2, \ldots, p_{n+m}\}$. Then the connectivity constraint set of a sensor agent i with respect to all other agents in the group is

$$\Upsilon_{\mathrm{d}_{hs}}(p_i, \mathscr{P}) = \Lambda_{ss} \cap \Lambda_{sr}. \tag{8}$$

Figure 1 shows an example of a sensors connectivity constraint set w.r.t. the heterogeneous network. The connectivity constraint sets defined in (2) - (8) define the set of allowable positions that each robot may take such that the communication network will remain connected. Thus, the connectivity constraint sets define the feasible motion for each individual robot to remain connected with the network.

3.1 Heterogeneous Proximity Graph

Due to the heterogeneity of our sensor network, we must define an appropriate proximity graph. As a reminder, a proximity graph describes connections between a set

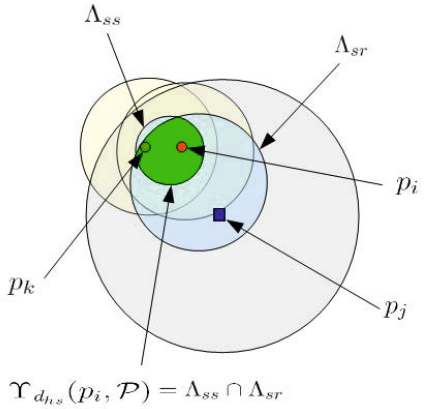


Fig. 1 Motion constraint set for a sensor agent (p_i) w.r.t. the network, $\Upsilon_{d_{hs}}(p_i, \mathcal{P})$. p_j is a relay agent and p_k is a sensing agent. The green area represents the connectivity constraint set that guarantees connectivity for sensor agent p_i w.r.t. the heterogeneous network.

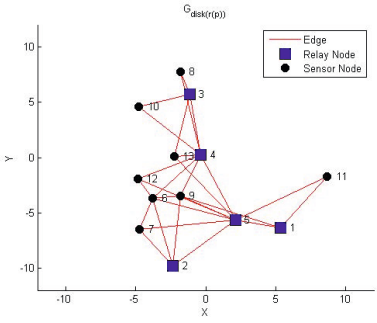


Fig. 2 Example of the $\mathcal{G}_{\text{disk}(r(p))}$ with five relay nodes and eight sensing nodes. Notice that there are many redundant connections between agents.

of vertexes based on their relative distances. Let us now define the proximity graph for our heterogeneous network.

Definition 6. (Heterogeneous r(p)-disk graph, $\mathcal{G}_{\text{disk}(r(p))}(\mathcal{P})$) Two agents p_i and p_j are neighbors if they are located within a distance $r(p) = R_c$ if both p_i and p_j are sensing agents or $r(p) = R_{rc}$ if one of the agents is a relay agent, *i.e.,*

$$(p_i, p_j) \in \mathcal{E}_{\mathcal{G}_{\text{disk}(r(p))}}(\mathcal{P}) \text{ if } \begin{cases} \|p_i - p_j\| \leq R_c \text{ and } p_i, p_j \text{ both sensing agents} \\ \|p_i - p_j\| \leq R_{rc} \text{ and } p_i \text{ or } p_j \text{ is a relay agent.} \end{cases} \tag{9}$$

An example of the $\mathcal{G}_{\text{disk}(r(p))}(\mathcal{P})$ graph is shown in Figure 2. In the $\mathcal{G}_{\text{disk}(r(p))}(\mathcal{P})$ graph, edges depend on the agent distances as well as agent connection combinations.

The heterogeneous r(p)-disk proximity graph, $\mathcal{G}_{\text{disk}(r(p))}(\mathcal{P})$, allows us to represent the communication links for each agent in the network. It can be seen, that depending on the configuration of the network there may exist heavy redundancy in the connections (Figure 2). This redundancy comes at the cost of more constraints on each agent (equations (5) and (8)), therefore reducing the size of the set of possible inputs (positions) that guarantee connectivity of the heterogeneous network.

4 Minimizing Motion Constraints

With a formal way of representing the network communication for each agent with respect to the heterogeneous group, we now are left with trying to minimize the connectivity constraints in such a way that we maximize the feasible motion sets (areas) the agents can choose from that still guarantees connectivity at the next time step. One solution is to take $\mathscr{G}_{\text{disk}(r(p))}(\mathscr{P})$ and run a minimum spanning tree algorithm to determine a subgraph of the r(p)-disk proximity graph that has the minimum number of connections needed to remain connected. A key result from modern graph theory is assuming $\mathscr{G}_{\text{disk}(r(p))}(\mathscr{P})$ is connected there always exist a minimal spanning tree (MST) [14]. The usefulness of the minimum spanning tree approach is that it allows us to weigh connections between agents. This may be useful in enforcing *relay/sensor* connections over *sensor/sensor* connections since *relay/sensor* connections offer a greater motion set for the agents as opposed to the *sensor/sensor* connections because of the larger communication radius. Another reason to bias certain network connections when possible is because relay nodes may be better equipped to handle communication data, *i.e.,* higher bandwidth.

4.1 Shaping the Network Configuration

To help bias *relay/sensor* connections over *sensor/sensor* connections with respect to the Minimum Spanning Tree (MST) we now formulate a weighting factor for *sensor/sensor* connections. From definitions (2) and (4) we see that the motion constraint set for *relay/sensor* connections is larger than *sensor/sensor* connections due to a larger communication radius. With the help of Figure 3 we look at the scenario of one relay and two sensing agents. In terms of the MST, all connections that have a possibility of being biased can be broken down in this way. For ease of notation we will refer to the $MST_{\mathscr{G}_{\text{disk}(r(p))}}$ as just the MST.

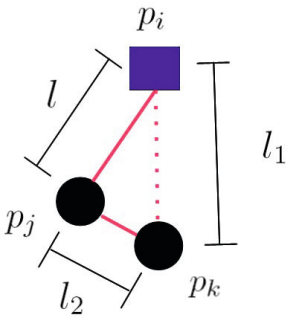


Fig. 3 Figure of one relay agent (blue square) and two sensing agents (black circle) used to formulate weighting factor for *sensor/sensor* connections

Let $\|p_i - p_j\|_2 = l$, $\|p_i - p_k\|_2 = l_1$ and $\|p_j - p_k\|_2 = l_2$. Let us assume that $l < l_1 \leq R_{rc}$ and $l_2 \leq R_c$. From construction of the MST, the red solid edges between p_i, p_j, and p_k in Figure 3 will be chosen since

$$l + l_2 < l_1 + l_2,$$

$$l + l_2 < l + l_1.$$

To bias the *relay/sensor* connection (red dotted line) a weighting factor ξ_1, must be constructed such that when $l_2 = R_c$, $\xi_1 l_2 \geq R_{rc}$. Defining $\xi_1 = \left(\frac{R_{rc}}{R_c} + \delta_1\right)$ with $\delta_1 \geq 0$ we get the following,

$$\begin{aligned} \xi_1 l_2 &= \left(\frac{R_{rc}}{R_c} + \delta_1\right) l_2, \\ \xi_1 l_2 &= R_{rc} + \delta_1 R_c, \\ \xi_1 l_2 &\geq R_{rc}. \end{aligned} \tag{10}$$

Therefore, with the connection weighting factor ξ_1 we now have the following,

$$l + l_1 \leq l + \xi_1 l_2,$$

$$l + l_1 \leq l_1 + \xi_1 l_2.$$

Weighting the *sensor/sensor* connection (edge) by a factor of ξ_1 allows us to bias the MST to chose the *relay/sensor* connections. Figure 2 shows a connected $\mathcal{G}_{\text{disk}(r(p))}$ graph with many redundant connections. Figure 4 and Figure 5 show the difference in network connections between the MST and the connection weighted MST (MST_{CW}) respectively, where *sensor/sensor* connections are weighted by the factor ξ_1. Notice that in the MST_{CW} graph, the *relay/sensor* connections are chosen over *sensor/sensor* connections.

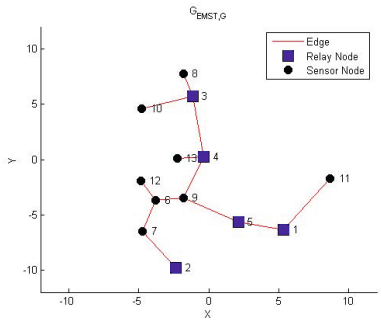


Fig. 4 Example of the MST for thirteen agents

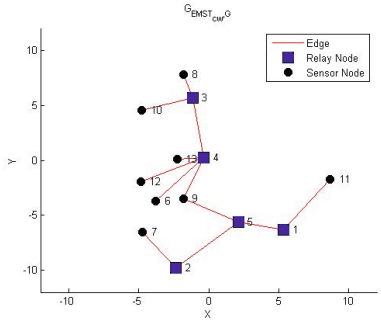


Fig. 5 The MST_{CW} graph for the thirteen agents. Notice how the *relay/sensor* connections are chosen over *sensor/sensor* connections.

We sum the total areas of the feasible motion sets for each graph representation of the network in Figure 4 and Figure 5, and see that the $\mathscr{G}_{\text{disk}(r(p))}$ totalled 167 units^2, the MST totalled 337 units^2, and the MST$_{CW}$ totalled 462 units^2. This gives us a good indication that the MST$_{CW}$ graph "frees" up more area for the sensing agents to investigate than the other network graph representations.

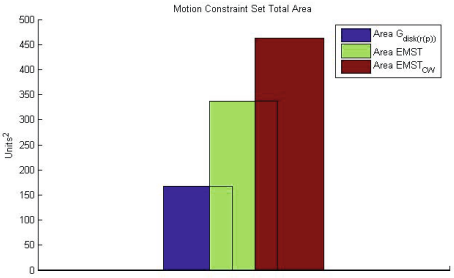


Fig. 6 Figure showing the total area covered by the motion constraint sets by the three different graph representations

In a similar fashion we can bias relay/relay connections. This may be advantageous for certain mission objectives or when large amounts of data may need to be transferred directly to a relay node. It may not be efficient or even possible to send large amounts of data through a sensing node to reach another relay node.

Using Figure 7, let us assume the minimum distance between any two agents is $\varepsilon > 0$. ε can be thought of the agent's physical footprint, *i.e.,* two agents can't

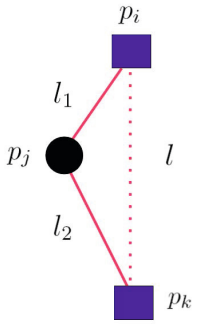


Fig. 7 Figure of two relay agents (blue squares) and one sensing agent (black circle) used to formulate weighting factor for *relay/relay* connections

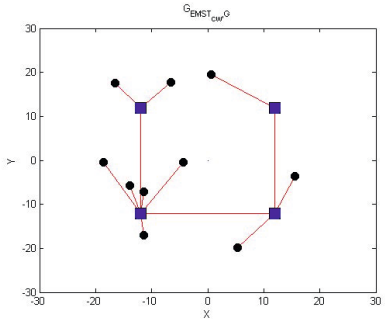


Fig. 8 Example MST$_{CW}$ for fourteen agents with both *sensor/sensor* and *relay/relay* connection weights

occupy the same position. Let us also assume that from Figure 7 that $l, l_1, l_2 < R_{rc}$ and for convenience assume $l_1 < l_2 < l$. From the point of view of the MST the red edges between p_i, p_j, and p_k in Figure 7 will be chosen since

$$l_1 + l_2 < l_1 + l,$$
$$l_1 + l_2 < l + l_2.$$

To bias direct *relay/relay* connections (Figure 7 red dotted line), we use a weighting factor $\xi_2 = \frac{\varepsilon}{l}$. Choosing ξ_2 in this way insures that a direct *relay/relay* connection will be chosen over the multi-hop connection by the MST algorithm in Figure 7, *i.e.,* relay$\rightarrow$ sensor$\rightarrow$ relay. This is seen from the fact that,

$$\xi_2 l = \frac{\varepsilon}{l} l,$$
$$\xi_2 l = \varepsilon.$$

Therefore, now the distance between p_i and p_k is ε from the point of view of the MST algorithm. Since the minimum distance of any two agents is ε the MST will choose the direct *relay/relay* link. Figure 8 shows the network configuration using both ξ_1 and ξ_2 as connection weights.

4.2 Properties of the Heterogeneous Motion Constraints

Theorem 1. *Given a relay agent, p_i, with dynamics described in* (1)*, such that* (1) *is at least stabilizable and having a motion constraint set as defined in* (5)*. If p_i takes a goal point $g_{p_i} \in \Upsilon_{d_{hr}}(p_i, \mathscr{P})$ at time t_1, then p_i will be connected with all agents at time t_2 that it was connected with at t_1 when it reaches g_{p_i}.*

Proof. Given the fact that the dynamics of p_i are at least stabilizable implies that there exists a static control law $u(t) = -Kx(t)$ such that the closed loop system is asymptotically stable, *i.e.,* $\lim_{t\to\infty} x(t) = g_{p_i}$.

By definition $g_{p_i} \in \Upsilon_{\mathrm{d}_{hr}}(p_i, \mathscr{P})$ which implies that

$$\forall q \in \mathscr{P} \backslash \{p_i\} \text{ s.t. } \left\| q - g_{p_i} \right\|_2 < R_{rc},$$

hence p_i at position g_{p_i} at time time t_2 is connected with all $p_j \in \mathscr{P}$ that it was connected to at time t_1. □

Theorem 2. *Given a sensing agent, p_i, with dynamics described in* (1)*, such that* (1) *is at least stabilizable and having a motion constraint set as defined in* (8)*. If p_i takes a goal point $g_{p_i} \in \Upsilon_{d_{hs}}(p_i, \mathscr{P})$ at time t_1, then p_i will be connected with all agents at time t_2 that it was connected with at t_1 when it reaches g_{p_i}.*

Proof. Given the fact that the dynamics of p_i are at least stabilizable implies that there exists a static control law $u(t) = -Kx(t)$ such that the closed loop system is asymptotically stable, *i.e.,* $\lim_{t\to\infty} x(t) = g_{p_i}$.

By definition $g_{p_i} \in \Upsilon_{\mathrm{d}_{hs}}(p_i, \mathscr{P})$ which implies that

$$\forall q \in \mathscr{P} \backslash \{p_i\} \text{ s.t. } \left\| q - g_{p_i} \right\|_2 < R_{rc},$$

if q is a relay agent and

$$\forall q \in \mathscr{P} \backslash \{p_i\} \text{ s.t. } \left\| q - g_{p_i} \right\|_2 < R_c,$$

if q is a sensing agent. Hence p_i at position g_{p_i} at time time t_2 is connected with all $p_j \in \mathscr{P}$ that it was connected to at time t_1. □

Note that Theorem 1 and Theorem 2 guarantee that the exact network topology will be preserved when each agent reaches its respective goal point. This implies that if one minimizes the number of constraints (links) before computing the communication constraint sets, the "minimized" network topology will be preserved.

5 Case Study: Centroidal Heterogeneous Motion Constraint Set Configurations

This section looks at the particular situation when the heterogeneous team moves towards their respective centroid of their feasible motion set. This centroidal configuration is a straightforward way to test the claims of Theorem 1 and Theorem 2.

By construction, the centroid $\mathscr{C}_i^*$ of each agents motion constraint set (MSC_i) lives in the interior of its motion constraint set. Therefore by setting $\mathscr{C}_i^*$ as the goal point for each agent $i, \forall i = 1, \ldots, n+m$, then the heterogeneous team should remain connected when each goal point is reached by the respective agents. Figure 9 depicts the centroidal heterogeneous motion constrain set configuration for two relay agents and one sensing agent. The stars denote the centroid of each agents constraint set.

Algorithm 5.1. Centroidal Behavior ($g_{p_i} = \mathscr{C}_i^*$)

```
while t < t_final do
  for x_i = 1,...,n+m do
    Calculate C_i^* from MSC_i (Equations (2)- (8))
    g_{p_i} ⇐ C_i^*
    while Δt < T_s do
      u_i(t) = −K x_i(t)
    end while
  end for
end while
```

For this simulation, we set $R_c = 3$ m, and $R_{rc} = 10$ m. We use $m = 4$ relay agents and $n = 6$ sensing agents initially in a random configuration but in such a way that the heterogeneous team is initially connected. Each agent $i, \forall i = 1, \ldots, n+m$, then uses its neighbors of the $\mathscr{G}_{MST_{CW}, \mathscr{G}}$ graph to calculate the centroid of their respective

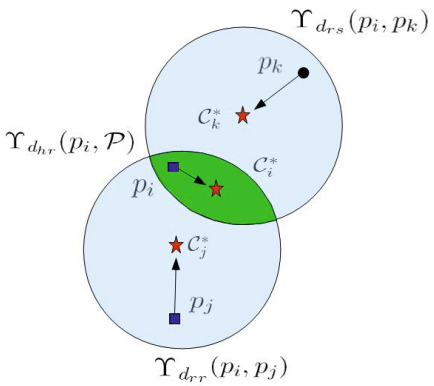


Fig. 9 Figure showing the centroidal heterogeneous motion constraint set configurations. Each respective agent calculates its own centroid w.r.t. its constraint set and then moves towards it.

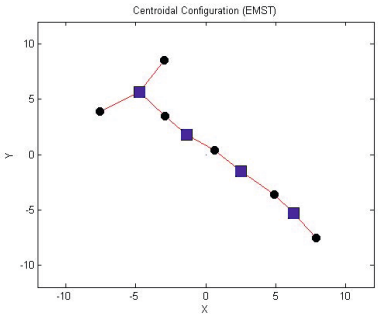


Fig. 10 Centroidal configuration for a heterogeneous team that moves towards goal points that are the centroid of their respective motion constraint set.

motion constraint set. Each agent is modeled as a double integrator (1), and a state feedback control law is used to drive the agents from their current position to their respective goal points ($\mathscr{C}_i^*$). Every $T_s = 0.05$ s, position information is exchanged among the team members and an updated $\mathscr{G}_{MST_{CW},\mathscr{G}}$ graph is calculated. Based on the new information, new centroids are updated and used as the goal point. The simulation lasts for a total of five seconds.

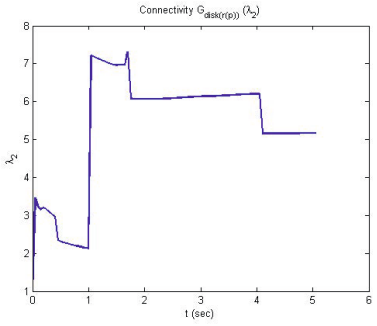


Fig. 11 The second smallest eigenvalue of the $\mathscr{G}_{\text{disk}(r(p))}$ graph during simulation of the centroidal behavior

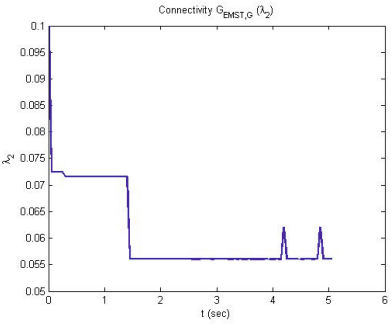


Fig. 12 The second smallest eigenvalue of the $\mathscr{G}_{MST_{CW},\mathscr{G}}$ graph during simulation

Figure 10 shows the final configuration of the heterogeneous team after five seconds of the centroid seeking behavior. The final configuration is reasonable since by design the algorithm tries to keep each agent "equidistant" from agents it shares a communication link with. Since only sensor/sensor connections were weighted for this simulation and due to the networks initial configuration we end up with this uniform mix of relay/sensor links. However this is not always the case. Figure 11 and Figure 12 show the connectivity with respect to the second smallest eigenvalue criteria (*i.e.*, $\lambda_2(\mathcal{G}) > 0 \Rightarrow \mathcal{G}$ is connected) for the $\mathcal{G}_{\text{disk}(r(p))}$ and $\mathcal{G}_{MST_{CW},\mathcal{G}}$ graph respectively. We see that both graphs stay connected at all times ($\lambda_2(\mathcal{G}) > 0$) however, only the $\mathcal{G}_{MST_{CW},\mathcal{G}}$ graph is used to calculate the constraint sets. Note that the larger $\lambda_2(\mathcal{G})$, the more connections exist in the graph. Figure 11 shows that in the $\mathcal{G}_{\text{disk}(r(p))}$ graph there exist redundant links that allows for some robustness to node failures, however it is unclear at this point to what extent this is true.

5.1 Hardware Experiments

For the hardware implementation of the communication constraints, we chose to implement Algorithm 5.1 on a single relay robot due to the lack of a larger experimental space. Two sensing robots were given predefined trajectories and were tasked with taking light intensity measurements along these trajectories. The sensing data is then exchanged between the agents at a rate of 5 Hz. The relay robot calculates its feasible motion set based on the positions of the sensing agents and then moves towards its centroid. Position information of the sensing agents were updated every $T_s = 0.5$ s. For this experiment we used $R_{rc} = 3.2$ m and $R_c = 1.0$ m for the communication radius of the relay and sensing agents respectively. The ad-hoc network consisting of three XBee wireless RF modules were used to

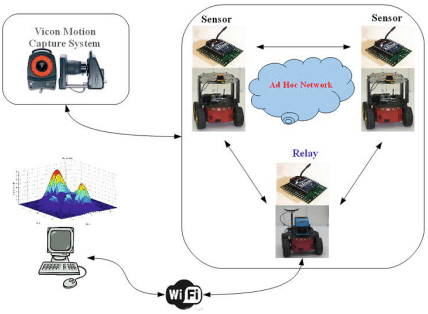


Fig. 13 Diagram of hardware experiments using the centroidal heterogeneous motion constraint set configurations

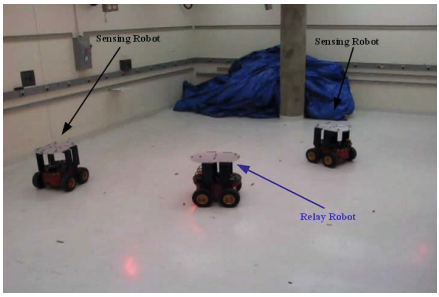


Fig. 14 Experimental snapshot showing the two sensing agents taking light intensity measurements over the search space and a single relay agent maneuvering to keep the network connected

communicate sensing data between robots. This allowed for the light intensity map to be built in a distributed fashion. A wireless local area network (WLAN) was used to send a real-time light intensity map from the relay robot to an end user using a laptop outside the experimental area. Figure 15 shows the evolution of the second smallest eigenvalue of the $\mathcal{G}_{\text{disk}(r(p))}$ graph. Notice that the heterogeneous network stays connected for the entire experiment.

Figure 13 shows a diagram of how the experiment was implemented and Figure 14 shows a snapshot of the hardware setup.

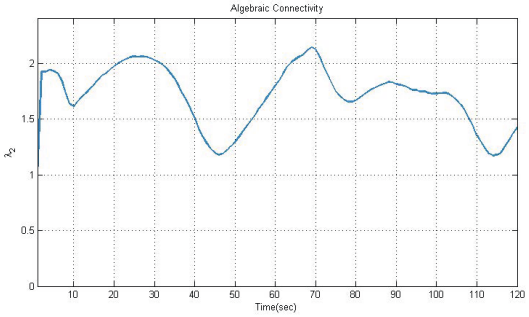


Fig. 15 The second smallest eigenvalue (λ_2) of the $\mathcal{G}_{\text{disk}(r(p))}$ graph during experiments of the centroidal behavior. The network remains connected since $\lambda_2 > 0$ throughout the experiment.

6 Conclusion and Future Work

In this paper we presented a framework for deriving motion constraints for a heterogeneous group of sensing and mobile communication relay agents such that network connectivity can be maintained. We also showed how these constraints can be minimized to allow for a larger motion area for the sensing agents while still maintaining network connectivity. Lastly, weighting factors were derived that allow for biasing particular communication links. This framework leads to the possibility of using heterogeneous sensor networks for prioritized search and surveillance problems where network connectivity is an underlying constraint. Through experimentation we showed that our framework can be implemented in real hardware.

Future work entails incorporating the derived motion constraints with our recent work on prioritized sensing [15]. This integration will allow for a prioritized search of an area while taking into account communication constraints of the network. Robustness of our approach to node failure is also an area of future research.

Acknowledgements. This work was supported by DOE URPR (University Research Program in Robotics) grant DE-FG52-04NA25590, NSF grants ECCS CAREER #0811347, and IIS #0812338. The authors would like to thank Jose-Marcio Luna for his help in conducting the experiments at the MARHES laboratory.

References

1. Bollobàs, B.: Modern Graph Theory. Springer, New York (1998)
2. Bullo, F., Cortés, J., Martínez, S.: Distributed Control of Robotic Networks. Applied Mathematics Series. Princeton University Press (2009), http://coordinationbook.info
3. Cortez, R.A., Fierro, R., Wood, J.: Prioritized sensor detection via dynamic Voronoi-based navigation. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5815–5820 (October 2009)
4. Dimarogonas, D.V., Johansson, K.H.: Decentralized connectivity maintenance in mobile networks with bounded inputs. In: IEEE International Conference on Robotics and Automation, pp. 1507–1512 (2008)
5. Fax, J.A., Murray, R.M.: Graph laplacian and stabilization of vehicle formations. In: Proceeding of the 15th IFAC, pp. 283–288 (2002)
6. Fink, J., Kumar, V.: Online methods for radio signal mapping with mobile robots. In: IEEE International Conference on Robotics and Automation, pp. 1940–1945 (2010)
7. Ghaffarkhah, A., Mostofi, Y.: Communication-aware target tracking using navigation functions - centralized case. In: International Conference on Robot Communication and Coordination (ROBOCOMM), pp. 1–8 (2009)
8. Gil, S., Schwager, M., Julian, B.J., Rus, D.: Optimizing communication in air-ground robot networks using decentralized control. In: IEEE International Conference on Robotics and Automation, pp. 1964–1971 (2010)
9. Hussein, I., Stipanović, D., Wang, Y.: Reliable coverage control using heterogeneous vehicles. In: IEEE Conference on Decision and Control, New Orleans, pp. 6142–6147 (2007)
10. Ji, M., Egerstedt, M.: Distributed coordination control of multiagent systems while preserving connectedness. IEEE Transactions on Robotics 23(4), 693–703 (2007)
11. Kim, Y., Mesbahi, M.: On maximizing the second smallest eigenvalue of a state-dependent graph laplacian. IEEE Transactions on Automatic Control 51(1), 116–120 (2006)
12. Michael, N., Zavlanos, M.M., Kumar, V., Pappas, G.J.: Maintaining Connectivity in Mobile Robot Networks. In: Khatib, O., Kumar, V., Pappas, G.J. (eds.) Experimental Robotics. STAR, vol. 54, pp. 117–126. Springer, Heidelberg (2009)
13. Stachura, M., Frew, E.W.: Reliable coverage control using heterogeneous vehicles. In: AIAA Guidance, Navigation, and Control Conference, Toronto, Canada (2010)
14. Tanner, H.G., Jadbabaie, A., Pappas, G.: Stable flocking of mobile agents, Part I: Fixed topology. In: IEEE Conference on Decision and Control, pp. 2010–2015 (2003)
15. Tekdas, O., Wang, W., Isler, V.: Robotic routers: Algorithms and implementation. The International Journal of Robotics Research 29(1), 110–126 (2010)

Multi-robot Topological Exploration Using Olfactory Cues

Ali Marjovi and Lino Marques

Abstract. This paper presents a distributed multi-robot system to search for odor sources inside unknown environments. The robots cooperatively explore the whole environment and generate its topological map. The exploration method is a decentralized frontier based algorithm that is enhanced by considering odor concentration at each frontier inside its cost/gain function. The robots independently generate local topological maps and by transferring them to each other, they are able to integrate these maps and generate a whole global map. The proposed method was tested and validated in real reduced scale scenarios.

1 Introduction

Search and rescue operations inside buildings, caves, tunnels and mines can be extremely dangerous tasks. An example of an extremely risky situation is the human operation inside an industrial warehouse during a fire. In these cases the human senses can become severely impaired: the visibility will be reduced by the smoke, the noise caused by the fire will make it impossible to communicate, and additionally dangerous vapors and toxins may be released. The use of autonomous robots to assist such tasks in complex environments reduces the risk of these operations. Robots can search for toxic chemicals and other targets while they explore the environment, providing real-time data about the discovered map and the status of the facility. It is well known that multi-robot systems can be faster and more robust doing a given task than single robot systems, but the reverse side of such benefits are the difficulties of efficiently coordinating those multiple robots, and accurately merging all the data gathered individually by each of them.

Ali Marjovi · Lino Marques
Universidade de Coimbra, Coimbra, Portugal
e-mail: {ali,lino}@isr.uc.pt

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 47–60.
springerlink.com

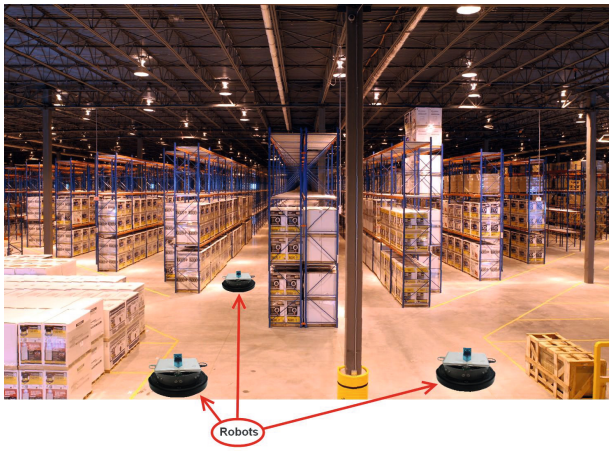


Fig. 1 Robots searching inside a warehouse composed by multiple straight corridors

The problem of search and exploration with multiple robots in an unknown environment can be stated as following; Consider a group of N mobile robots, moving in R^2 that are labeled as $R_1, R_2, ..., R_N$. Each robot $R_i (i = 1, ..., N)$ is able to communicate with the other robots localized at a short distance range Δ. The robots are equipped with sensors for measuring the odor intensity and air flow direction. There are unknown number of odor sources in the environment which emit odor gas into the area. There is no central base-station for the system. The robots should act separately and independently from the others. There is no global positioning system and the odometry of the robots is not very accurate. There is no knowledge about the environment before the mission except that it is similar to a structural building containing corridors, corners, branches, crosses, etc. similar to a warehouse (Fig. 1). The problem is to localize all the odor sources in the environment, explore the whole area, and generate a topological map of the environment.

The authors have been working on the problem of finding odor and fire sources with robots in previous studies [1, 2, 3, 4, 5]. This current project tries to address the problem of odor source searching in unknown environments by complementing it with the environment exploration and mapping.

The cooperation, communication, and management of the robots in a multi-agent system can be done either in centralized way by using a base station as the server, or decentralized way by having a distributed behavioral based algorithm (like in [6, 7]). Lacking a central station makes it difficult to distribute the tasks between the robots. Since the environment is unknown, the robots are not aware of the tasks before exploring the area, i.e. there can not be any kind of task allocation before the mission. Task allocation must be done during the mission automatically by the robots which are participating in the search and exploration mission.

Singh and Fujimura [8] presented a decentralized online approach for heterogeneous robots. In their method, most of the time the robots work independently.

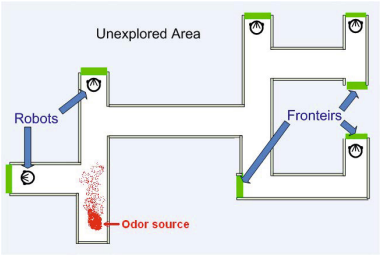


Fig. 2 Multi-robot frontier-based search and exploration

When a robot finds a situation that is difficult to solve by itself, it sends the problem to another robot which may be able to solve the situation. Yamauchi [9] proposed a distributed method for multi-robot exploration, yielding a robust solution even with the loss of one or more vehicles. A key aspect of this approach involves sharing map information among the robotic agents so they execute their own exploration strategy, independently of all other agents. While this technique effectively decentralizes control, exchange of map information is not enough to prevent inefficient cooperative behaviors. As a simple conclusion, "task sharing" is tightly related to two other issues; "cooperative technique" and "map merging".

If the robots know their relative locations and share a map of the area that they have explored so far, then effective coordination can be achieved by guiding the robots into different, non-overlapping areas of the environment [6]. In other words, effective coordination can be achieved by extracting exploration frontiers from the partial maps and assigning robots to frontiers based on a global measure of performance [6, 7]. Frontier based exploration is a simple approach for decentralized multiple robot task allocation (Fig. 2). Frontiers are the borders of the partial map, between explored free space and unexplored area. These borders, thus, represent locations that are reachable from within the partial map and provide opportunities for exploring unknown terrain, thereby allowing the robots to greedily maximize information gain [10]. However, methodologies should have a strategy to not send two robots toward the same frontier.

While multiple robots cooperatively explore an environment, information from individual robots must be integrated to produce a single globally consistent map. This is a difficult problem when the robots do not have a common reference frame or global positioning system [11]. Topological maps provide a brief characterization of the navigability of an environment, and, with measurements easily collected during exploration, the vertices of the map can be embedded in a metric space [11]. These maps use a graph to represent possibilities for navigation through an environment.The current proposed approach employs a topological mapping technique, so the robots only exchange a few environmental features.

Most of the existing approaches to coordinate multi-robot mapping assume that all agents know their locations in a shared (partial) map of the environment [2, 5, 6, 7]. Having a general positioning system is a constraint in unknown areas where there

is no knowledge about the environment. This paper deals with two issues in this aspect, the first dealing with uncertainty in localization of each robot by correcting it by the information of the partial maps, and the second presenting an approach for map merging in the case that the robots do not know their relative locations and their coordinate systems are different with each other.

Using topological maps, the problem of "map merging" is reduced to the "graph merging" problem that is already addressed in several ways [11, 12, 13]. Whereas most approaches to topological map merging and related problems have focused on using either map structure [14] or map geometry [15], our algorithm takes advantage of both. Similar to [11], the use of map structure allows quick identification of potential vertex matches in the maps (and rejection of mismatches), while the use of map geometry enables the algorithm to directly merge maps with multiple (disconnected) overlapping regions. In another words, the algorithm in [11] uses both the structure and the geometry of topological maps to determine the best correspondence between maps with single or multiple overlapping regions. However, if the geometric data of the local maps are not obtained through a unique coordinate system this method (in [11]) will not be functional.

As already stated, the exploration must be done in a way that the group of robots automatically intends to look for the odor sources in the environment. Over the past years, odor localization has become recognized as a valuable area of robotics with practical applications. Most of related works to the problem have focused on mobile robots at a scale of the order of tens of centimeters, operating in open areas free of obstacles with a background fluid flow. Fluid flow in these environments is dominated by turbulence. The odor is carried downwind from the source forming a plume. Due to turbulence, the plume meanders, and the chemical concentration within the plume is patchy [16]. The researchers have developed methods that employ combinations and variations of plume acquisition and plume upwind following using reactive control algorithms. However, most of the significant projects have addressed the problem with a single robot [1, 17, 18, 19] and/or in open, free obstacle spaces [19, 20]. However, this paper attends to fulfill the goals using multiple robots in a structural environment (similar to a real building of a warehouse).

2 The Proposed Method

This section explains the concept of the proposed multi-robot cooperation technique. This method is illustrated in the schematic diagram of Fig. 3. As shown in the diagram, Some of the tasks of the diagram are briefly described below.

2.1 Olfactory Search and Exploration Algorithm

It is desired to find odor sources and cover the whole environment as fast as possible. Therefore, it is essential that the robots share their tasks and individually achieve the objectives through optimal paths towards the odor sources. In an unknown

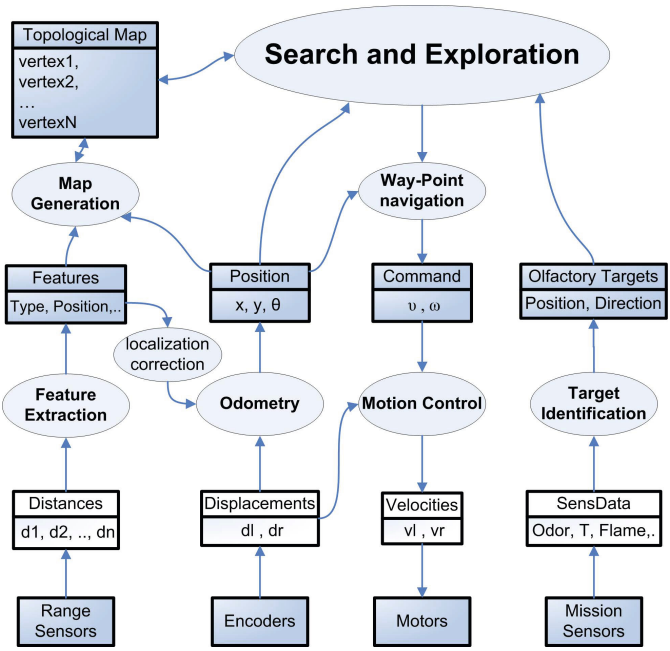


Fig. 3 Software architecture

environment, the immediate goals are the frontiers. Most of the time, when the robots are exploring an area, there are several unexplored regions, which poses a problem of how to assign specific frontiers to the individual robots without existence of a specific task allocator. In the proposed approach, the robots firstly decide to explore the frontiers which indicate higher odor concentration. The robots must avoid selecting the same frontier, this may result in collision concerns. Another problem is the lack of base station, so the robots should be able to explore autonomously and also avoid collisions between themselves. To address these problems, the proposed method is based on a behavioral exploration algorithm that is presented in algorithm 1.

Algorithm.1 describes the core of the decision making technique of the proposed method. To briefly describe this algorithm; the robots start exploring the environment independently. Each robot looks forward to get into new features in the environment. It generates its own local topological map out of the extracted features of the environment and also transmits this local map to the other robots which are working on the same environment. Once the robot is in the situations that has to make a decision to select its future path (e.g. in the branches), it first measures the odor concentration in that place. If the odor concentration is more than a certain threshold, it means that the robot is in the zone of an odor source and it must go to the direction of up-wind in order to localize that. In this case the robot takes an action to go towards up-wind.

Algorithm 1: Odor source searching and map exploration algorithm

```
1  while there is at least one unexplored frontier in the map do
2      repeat
3          Go_forward_and_follow_the_potential_field_algorithm();
4      until getting a different feature in the environment;
5      if the new node exists in the map then
6          Update_map's_data();
7          Correct_the_odometry_error_based_on_the_feature_data();
8      else
9          Add_new_node_to_map();
10     Send_the_new_local_map_to_the_other_robots();
11     if the current node has any unexplored link then
12         M = Measure_the_odor_concentration();
13         D = Detect_the_wind_direction_by_anemometer();
14         if (M > odor_threshold) and (D is not explored) then
15             Set_the_new_Objective_to_go(D);
16         else
17             F=Find_unassigned_unexplored_frontier_with_highest_Utility-Cost();
18             D = Calculate_the_best_path_to_take_based_on_the_A*_algorithm();
19             Set_the_new_Objective_to_go(D);
20     if M > Odor_Source_Threshold then
21         Report_this_place_as_an_odor_source_to_all_other_robots();
22 End of algorithm
```

However if a robot is travelling inside the explored area and wants to select a frontier to explore; the frontier is selected based on the cost of reaching it and the utility it can provide to the exploration. The cost is calculated through the A* method, where it simultaneously determines the optimal path to reach the frontier and its distance. Therefore, the cost is proportional to the distance that the robot has to pass to reach the frontier.

$cost = dist(A^*_{i=0,n}[(X_R, Y_R), (X_{f_i}, Y_{f_i})])$

where (X_R, Y_R) is "position of the robot",
(X_{f_i}, Y_{f_i}) is "position of the frontier i"
and n is "number of frontiers".

The utility depends on the level of odor concentration in that frontier, which means that if there are several frontiers at similar distances, the robot will go to the one that has higher utility, i.e. higher odor concentration;

$utility(i) = Odor\_Concentration(i) \quad \forall i \in \{1..n\}$.

This procedure will make the robots disperse and explore the environment in an efficient way towards odor sources.

The robots generate the topological map of the environment during their search and exploration mission. Within the topological map, besides having information regarding the kind of nodes, their position and the odor concentration in that feature, it also has data describing the location of the robots and their frontier target. Through this data, a robot can see which frontiers are unexplored, their position and if any robot has targeted them as its objective. Therefore, the robots will not attend to explore the same frontiers. Each robot is aware of the frontier that the other robots have aimed to explore, so it will choose another frontier that is unexplored and

unassigned to any other robot. As a result, the robots will autonomously pick up the tasks in a way that not any frontier will be assigned to more than one robot.

2.2 Feature Extraction

With the type of environment considered in this paper, there are five types of features that can be extracted from structured environments; corridors, corners, crosses, branches (3-way junctions) and dead-ends. The robots should recognize environmental features based on the distances from left, right and front measured by range sensors. Fig. 4 is an example that shows how robots detect corridors and branches by their sensors. Since "feature extraction" is not the main theme of this work, this paper does not go to its details.

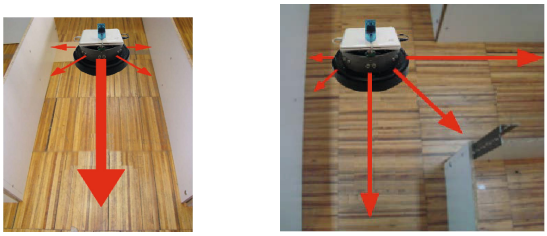


Fig. 4 Feature extraction, left: corridor, right: branch

2.3 Map Merging

Consider several robots exploring a single environment while each one has its own coordinate system. Their axes of X and Y do not match with each other and moreover they do not know where the reference point of the other's localization system is. Each one of them is generating the topological map of its visited local area. They are simultaneously sending these local self-generated topological maps to each other. The problem is "how each one of them can integrate the data coming form the others to its local map and generate a bigger map?". Fig. 5 shows an example of this problem. There is no central station unit for attuning the robots; moreover, there is not any specific landmark in the environment. Therefore, the robots should solve the problem in a distributed way.

Similar to [11], in this method, the generated map represents more than just the structure of the environment. Additional information, such as the degree of vertices, the orientation of edges at vertices, and other attributes, is recorded and stored in annotations of the graph. Fig. 7 shows an example of the topological map's data with brief descriptions.

The next step is to match the topological maps and generate a merged global map out of them. Since the topological map includes geometric data of each node, we

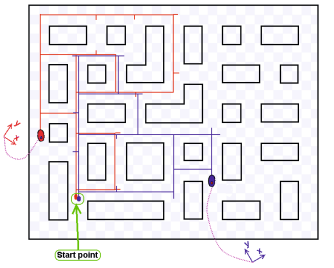


Fig. 5 An environment being explored by two robots with different coordinate systems

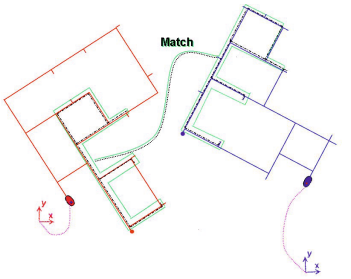


Fig. 6 Matching the generated maps of two robots exploring the environment in Fig. 5

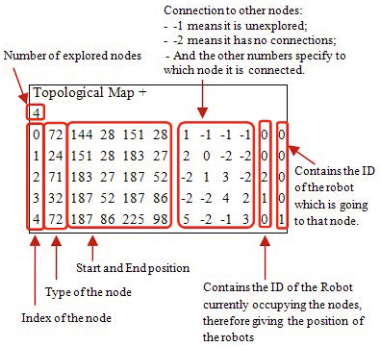


Fig. 7 An example of topological map data

can consider the maps as several subgraphs that belong to a unique graph. Now the problem of map merging is converted to the standard well-known subgraph isomorphism problem.

We considered the problem of "map merging" as a "graph matching" problem (see Fig. 5 and Fig. 6). One of the problems in graph matching is error-tolerant subgraph isomorphism. The robots should identify a common subgraph when there may be missing vertices and edges. Solutions to this problem are generally evaluated in terms of the "edit distance", the smallest sequence of elementary graph operations

(i.e., substitution, deletion, insertion) that transforms one subgraph into the other. When a robot cannot reliably detect "places" or when the environment is dynamic, the resulting topological map may have missing vertices and edges.

Common subgraphs $H_1 = [V,E,k]$ (where V is the set of vertices, E is the set of edges and k is the graph size) and $H_2 = [W,F,k]$ of two given graphs G_1 and G_2 are those, of equal size k, that are isomorphic to each other. This means that there should be such numeration of subgraphs' vertices $x(i)$ and $y(i)$, that :

$$\forall i,j \in \{1...k\}$$
$$Equivalence\_vertex\_function(v_{x(i)}, w_{y(i)}) = true \quad (1)$$
$$Equivalence\_edge\_function(v(e_{(x(i),x(j))}, f_{(y(i),y(j))}) = true$$

"$Equivalence\_vertex\_function$" : Two vertices are equivalent if and only if their environmental features match with each other.

"$Equivalence\_edge\_function$" : Two edges are equivalent if their lengths are approximately equal and their connecting vertices are equivalent.

In other words, all pairs of matching vertices in the subgraphs are connected by matching edges (including the virtual null edges). Now we just look for numbered sets $X = \{x(i)\}_{i=1}^{k}$ and $Y = \{y(i)\}_{i=1}^{k}$, satisfying conditions (1). The major issue here is to define the "Equivalence_vertex_function" and "Equivalence_edge_function" functions. Two vertices are equivalent if and only if their environmental features matches with each other and two edges are equivalent if their length is almost equal and their connecting vertices are equivalent. By that, the subgraphs match with each other regardless to the positions of the vertices but based on the topology of them.

The project has stated that the robots start from the same point (usually an entrance of the building) at the beginning of the mission. By this assumption it is guaranteed that the first vertex of all the local maps is from the same point, therefore the graphs always have at least one common subgraph and the robots are able to merge their maps in any case, while coordinate systems are not matched.

The value of k (common subgraph size) is a critical issue, Based on the experiments that we had in simulations and also in the real world, we figured out that it is good enough to set k to five. This means that if two robots find a common subgraph with five vertices, they can surly merge their local maps together.

Here is a description of the map merging method. Whenever a robot finds a new feature in the environment, it adds this feature as a new node to its local map and sends a message to all the other robots and reports the new map. In the other hand, each robot has a running memory-resident program that always is listening to the network and receives all the messages that are sent by the other robots. When a robot receives a message that shows another robot has found a new feature in the environment, it starts the hypothesis building process by creating the list of all vertices in the local topological map that are structurally compatible with the new map. Vertexes are tested for compatibility by examining their attributes: exactly known attributes (e.g., vertex type) must match perfectly; inexactly known attributes (i.e., due to measurement error) must be compared with a similarity test. Finally, this robot can find a common isomorphic subgraph and will modify its own local map. Therefore all the robots are able to merge their local maps with each other without having a base station.

Once the robot finds the subgraph in another map that is isomorphic to a subgraph of its own map, it will merge these maps with each other after finding the geometric transform function that converts the positions of the vertices of the second map to its current map. Since we have used a position correction method (described in the next section) that corrects the odometry of the robots, a linear transform function is good enough to convert the coordinates of two maps to each other. The transform function is found by a simple geometric calculation. Finally, the robot is able to add all the nodes of the other map to its own map after transforming their positioning data to its coordinate system.

Robots localization is a key issue in multi-robot mapping and exploration. There is no global or local positioning system or any kind of landmark or beacon for localization in this project. The only tool that the robots have for determining their position is their odometry. However, the odometry is unreliable because of uneven floors and wheel slippage. It is therefore necessary to augment the localization accuracy by measuring position of robot relative to known objects in the environment.

Normally odometry errors accumulate incrementally as while as the robot is travelling, therefore, the robot's odometry is more accurate at the beginning of an experiment and it loses its accuracy during the test. If the robot enters to an environmental feature that has been already explored, it can look at the map and find the start position of that feature, then correct its (x, y) based on the data that has been already stored inside the map. It does not matter if this feature was added to the map by the current robot or by another robot in the team, since the feature has been added to the map in the past; it means that the location that was saved in the map is more reliable than the odometry of the robot. This method is only used when the robot is passing an area that had been already explored.

Some other issues in multi-robot search and exploration, namely "feature extraction", "olfactory sensing", "communication" and "motion control" are out of scope of this paper and mostly were already addressed in [1, 2, 5] by the authors.

3 Experiments

The proposed algorithm has been tested and validated both in real reduced scale scenario using iRobot[1] Roomba robots and in simulation using the Payer/Stage environment [21]. The Roomba vacuum cleaning robot is an attractive research platform since it is inexpensive, readily available, and can be fully monitored and commanded through a serial port interface. In the current work, a set of Roombas were upgraded with small laptop computers (ASUS Eee PC 901) running a Linux based operating system and the Player environment. The computers interface through a micro-controller board with a set of five sonars, 2D anemometer and a gas sensing board (see Fig. 8). The gas concentration was measured with a custom sensing board based on metal oxide gas sensors (Figaro[2] TGS2620). The directional anemometer

[1] http://www.irobot.com

[2] http://www.figarosensor.com

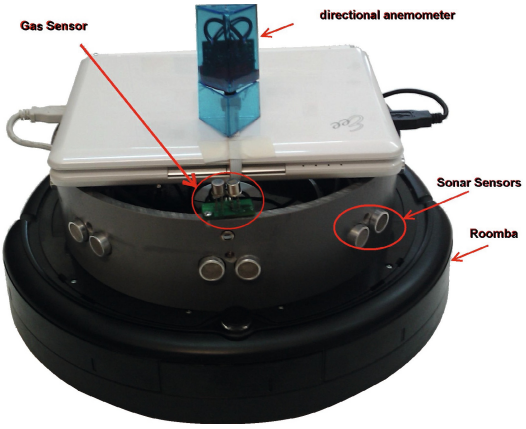


Fig. 8 iRobot Roomba robot equipped with a laptop, a sonar array, gas sensors, and 2D anemometer

was based on self heated NTC[3] sensors placed around a triangular prism and processing the raw measurements with a method similar to the one previously described in order to estimate the wind direction [22]. The Player/Stage driver for Roomba robots makes it possible to run the same code either in simulation or on the real Roomba robots. The sonars range was approximately 1.5 m while the biggest width of reduced scale testing environments is about 40 cm.

During exploration and navigation, the robots are simultaneously acquiring information from the environment. Given the structure of the considered environment, mainly corridors, if a robot is travelling in the upwind direction and the gas concentration decreases suddenly, that means that the robot has just passed through an odor source. On the other hand, if the robot is travelling in the downwind direction and it starts detecting a high concentration of odor, it means that there is an odor source in this location.

The proposed method was tested in different small scale maze-like environments, like the one shown in Fig. 9. The shown testing arena, with $3 \times 4m^2$ area by 0.5 meters height, has controlled ventilation through a manifold that extracts air from the testing environment through a honeycomb mesh integrated into one of the walls. The opposite surface of the environment contains a similar mesh that allows the entrance of clean air that flows through the environment. Controlled gas sources are simulated with ethanol vapor, generated using bubblers and pumped to different localizations of the environment through a set of PVC tubes.

In order to evaluate the method in terms of odor source searching, the algorithm was experimented in the real world once without any odor source and another time with existence of an odor source. By comparing the results and analyzing the behavior of the robots, the functionality of the method was validated.

[3] *Negative Temperature Coefficient.*

Fig. 9 Three robots exploring a gas free environment

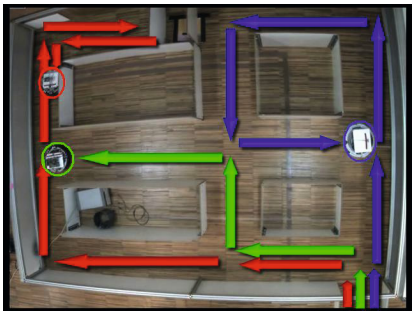


Fig. 10 Three robots exploring the environment and finding the odor sources

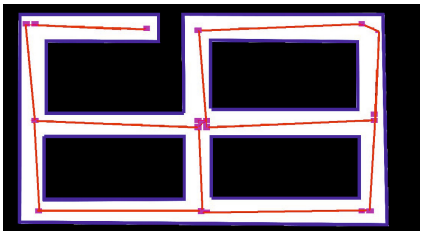


Fig. 11 The generated map for the environment in the Fig. 10

Fig. 9 shows three robots exploring a small maze and finding an odor source. In this experiment there were no odor sources. All robots started from the same point but not at the same time. We intentionally ran the robots a few seconds after each other. The red footprint shows the first robot's path, the blue footprint is related to the second robot and the green shows the footprint of the third robot. Fig. 11 shows the topological man of testing environment generated by the robots. The full algorithm is functional and it works in different maze structures and with different number of robots. For an example of the coordination between the robots, in Fig. 9 when the second robot reached the junction it figured out that the path in the front was already explored so it chose the left path.

The same maze structure was tested with the same robots with adding an ethanol odor source in the left side of the environment. The results show the effect of odor concentration on the behavior of the robots. Fig. 10 shows the path that robots took during exploring the environment. The first robot in the first branch made decision to go to the left-way because of a high clue of the odor and wind speed in that direction (red footprints).

The most important parameter for evaluation of the method is the exploration time. The environment shown in Fig. 9 was tested by one, two and three Roomba robots separately, once without having any odor or gas source, and once with having an odor source releasing gas in the environment. Fig. 12 shows that the exploration

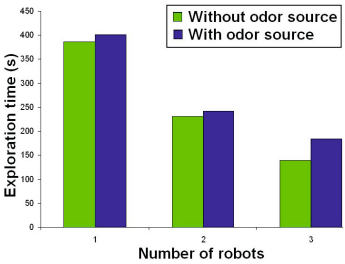


Fig. 12 Exploration time

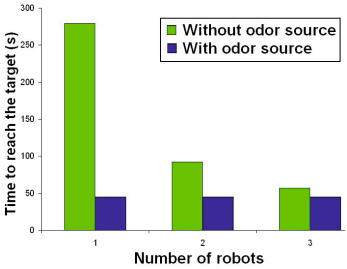


Fig. 13 Reaching the target (the odor source)

time is a bit more, with having gas cues, however it is not a big difference and they are still comparable. Fig. 13 shows the time to reach the target (the location of the odor source) in these two scenarios. The chart shows that the robots reach the target much faster with having gas cues rather that without having it, which proves the functionality of the algorithm. Each result is the average of five similar tests. Different tests with constant conditions had similar results with about eight percent variance. The maximum speeds of the robots were kept constant in all the tests.

4 Conclusions

A proposed method for multi-robot odor source search and exploration in unknown structural environment has been implemented and experimented in realistic reduced scale scenarios. The exploration algorithm is modified by integrating odor sensing cues in the frontiers selection and has been tested in the real world. The robots navigate towards the odor sources and are able to localize them while cooperating with each other by sharing information in their local maps.

In terms of mapping, the robots generate the topological map of environment during exploration. Map sharing is the main tool for automatic distributed task sharing and cooperation in our method. We showed that the robots can merge their topological maps based on common subgraph isomorphism even if they do not have common coordinate systems. After merging the maps, each robot will know the unexplored frontiers and assigns one of the frontiers to itself as a target to explore, so the other robots will not aim to that frontier anymore.

The algorithm was tested in the real world with different configuration and different number of robots and the results show the effect of gas cues on the behavior of the robots and it proves that based on the proposed algorithm, robots firstly explore the area with higher probability of existence of odor sources.

Acknowledgements. This work was partially supported by European project GUARDIANS contract FP6-IST-045269 as well as by the Portuguese Foundation for Science and Technology contract SFRH/BD/45740/2008.

References

1. Marques, L., Almeida, N., de Almeida, A.: Olfactory sensory system for odour-plume tracking and localization. In: IEEE Int. Conf. on Sensors, Toronto, Canada (2003)
2. Marjovi, A., Nunes, J., Marques, L., de Almeida, A.T.: Multi-robot exploration and fire searching. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, St. Louis, USA (2009)
3. Marques, L., Nunes, U., de Almeida, A.: Particle swarm-based olfactory guided search. Autonomous Robots 20(3), 277–287 (2006)
4. Marjovi, A., Nunes, J., Sousa, P., Faria, R., Marques, L.: An olfactory-based robot swarm navigation method. In: Proc. IEEE Int. Conf. on Robotics and Automation, USA (2010)
5. Marjovi, A., Nunes, J.G., Marques, L., de Almeida, A.: Multi-Robot Fire Searching in Unknown Environment. In: Howard, A., Iagnemma, K., Kelly, A. (eds.) Field and Service Robotics. STAR, vol. 62, pp. 341–351. Springer, Heidelberg (2010)
6. Fox, D., Ko, J., Konolige, K., Limketkai, B., Schulz, D., Stewart, B.: Distributed multirobot exploration and mapping. IEEE Special Issue on Multi-Robot Systems 94(7), 1325–1339 (2006)
7. Burgard, W., Moors, M., Stachniss, C., Schneider, F.: Coordinated multi-robot exploration. IEEE Trans. on Robotics 21(3), 376–386 (2005)
8. Singh, K., Fujimura, K.: Map making by cooperating mobile robots. In: Proc. IEEE Int. Conf. on Robotics and Automation (1993)
9. Yamauchi, B.: Frontier-based exploration using multiple robots. In: Proc. of 2nd Int. Conf. on Autonomous Agents (1998)
10. Zlot, R., Stentz, A., Bernardine Dias, M., Thayer, S.: Multi-robot exploration controlled by a market economy. In: Proc. IEEE Int. Conf. on Robotics and Automation, USA (2002)
11. Huang, W., Beevers, K.: Topological map merging. The International Journal of Robotics Research 24(8), 601 (2005)
12. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-completeness. W.H. Freeman, New York (1979)
13. Krissinel, E., Henrick, K.: Common subgraph isomorphism detection by backtracking search. Software: Practice and Experience 34(6), 591–607 (2004)
14. Dudek, G., Jenkin, M., Milos, E., Wilkes, D.: Topological exploration with multiple robots. In: Proc. 7th Int. Symp. on Robotics with Applications, Anchorage, Alaska (1998)
15. Dedeoglu, G., Sukhatme, G.: Landmark-based matching algorithm for cooperative mapping by autonomous robots. In: Proc. the 2000 Int. Symp. on Distributed Autonomous Robotic Systems, Knoxville, TN, pp. 251–260 (2000)
16. Kowadlo, G., Rawlinson, D., Russell, R., Jarvis, R.: Bi-modal search using complementary sensing (olfaction/vision) for odour source localisation. In: Proc. IEEE Int. Conf. on Robotics and Automation, Orlando (2006)
17. Loutfi, A., Coradeschi, S., Lilienthal, A., Gonzalez, J.: Gas distribution mapping of multiple odour sources using a mobile robot. Robotica 27(02), 311–319 (2008)
18. Marques, L., Nunes, U., Almeida, A.: Olfaction-based mobile robot navigation. Thin Solid Films 418(1), 51–58 (2002)
19. Ferri, G., Caselli, E., Mattoli, V., Mondini, A., Mazzolai, B., Dario, P.: SPIRAL: A novel biologically-inspired algorithm for gas/odor source localization in an indoor environment with no strong airflow. Robotics and Autonomous Systems 57(4), 393–402 (2009)
20. Hayes, A., Martinoli, A., Goodman, R.: Distributed odor source localization. IEEE Sensors Journal 2(3), 260–271 (2002)
21. Gerkey, B., Vaughan, R., Howard, A.: The Player/Stage project: Tools for multi-robot and distributed sensor systems. In: Proc. IEEE Int. Conf. on Advanced Robotics, Coimbra, Portugal, pp. 317–323 (2003)
22. Marques, L., de Almeida, A.: ThermalSkin: a distributed sensor for anemotaxis robot navigation. In: Proc. 5th IEEE Int. Conf. on Sensors, South Korea, pp. 1515–1518 (2006)

Distributed Coverage and Exploration in Unknown Non-convex Environments

Subhrajit Bhattacharya, Nathan Michael, and Vijay Kumar

Abstract. We consider the problem of multi-robot exploration and coverage in unknown non-convex environments. The contributions of the work include (1) the presentation of a distributed algorithm that computes the *generalized Voronoi tessellation* of non-convex environments (using a discrete representation) in real-time for use in feedback control laws; and (2) the extension of this method to entropy-based metrics that allow for cooperative coverage control in unknown non-convex environments. Simulation results demonstrate the application of the control methodology for cooperative exploration and coverage in an office environment.

1 Introduction

We are interested in considering the following scenario: a team of robots enter an unknown and non-convex environment. The robots must control to explore the environment for map construction and converge to a formation in the map that disperses the robots to locations that permit them to continue to engage in activities such as persistent surveillance. This description lends itself to a broad class of robotics applications. In this work, we focus on an essential component toward this scenario: the development of decentralized individual robot control laws based on uncertain estimates of the environment that drive the team of robots to explore and cover the environment. The contributions of this work are:

1. the presentation of a distributed algorithm that computes the *generalized Voronoi tessellation* of non-convex environments (using a discrete representation) in real-time for use in feedback control laws; and

Subhrajit Bhattacharya · Nathan Michael · Vijay Kumar
GRASP Laboratory,
University of Pennsylvania,
Philadelphia, PA 19104, USA
e-mail: {subhrabh,nmichael,kumar}@grasp.upenn.edu

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 61–75.
springerlink.com

2. the extension of this method to entropy-based metrics that allow for cooperative coverage control in unknown non-convex environments.

Therefore, to limit the scope of the presentation, we assume the robots are able to (A) localize and build maps in a common frame; and (B) communicate.

The presentation begins by motivating the development of a geodesic Voronoi tessellation in non-convex environments. We detail a search-based algorithm for computing an equivalent tessellation in discrete environments (Sect. 4). We propose a method for computing the centroid of the Voronoi cells given a tessellation of a discrete environment, permitting the application of centroid-based robot control laws for cooperative coverage. In Sect. 5, uncertainty in the environment description is introduced through the development of an entropy-based metric; enabling the computation of the instantaneous geodesic Voronoi tessellation given an uncertain environment. We comment on the coverage and convergence guarantees resulting from this approach and present results that demonstrate through simulation its application in an office environment.

2 Related Literature

This work is most closely related to methods proposed in the cooperative exploration literature and coverage control literature. A common approach toward exploration is frontier-based exploration where control directions seek to minimize entropy or uncertainty in the robot pose or map [16]. Coordination for multi-robot exploration is generally accomplished through explicit coordination designed to reduce redundant exploration or implicit coordination that occurs when robots communicate and coordinate (e.g. share maps) when in close proximity but without considering other robots' history or future plans.

In [15], the authors propose an exploration strategy with feedback control laws that maximize information gain by considering uncertainty in both the robot pose and map. A key contribution of this work (and similar recent works) is the relaxation of the assumption of robot localization. It is for this reason that we believe the first assumption in the prior section is reasonable. A multi-robot exploration strategy is presented in [2, 14], where the robots coordinate to determine targets best served by each robot that maximize the information gain for the team of robots. The authors quantify the performance gain due to explicit coordination and increasing numbers of robots. In [7], experimental results are presented that demonstrate the use of a team of robots to address a scenario similar to the one described at the beginning of this work. The authors do not consider explicit coordination between robots and note that this results in redundant exploration. A similar multi-robot exploration study is presented in [6], where robots explore an indoor office environment while simultaneously localizing and mapping. Coordination is implicit as the robots exchange and merge maps when in close proximity.

A common coverage control approach is through the definition of feedback control laws defined with respect to the centroids of Voronoi cells resulting from the

Voronoi tessellation of an environment. In [4], the authors propose gradient descent-based individual robot control laws that guarantee optimal coverage of a convex environment given a density function which represents the desired coverage distribution. The authors of [12] build upon this idea and develop decentralized control laws that position a mobile sensor network optimally with respect to a known probability distribution. In [13], this approach is extended to consider near-optimal controllers that do not require prior knowledge of a desired coverage distribution. To address the limitation of requiring a convex environment, the authors of [10] propose the use of *geodesic Voronoi tessellations* determined by the geodesic distance rather than the Euclidean distance. An approach that considers both exploration and coverage using Voronoi tessellations is presented in [17]. However, this method differs greatly from ours and assumes that the boundary of the environment is known. The Voronoi tessellation is computed based on this assumed convex polygon and the robots control to the centroids of this tessellation and explore en route.

The primary point of differentiation between our work and existing methods is due to the fact that we are able to rapidly compute the Voronoi tessellation of non-convex environments in a distributed manner. The consequence of this result is that we can use the instantaneous tessellation to compute decentralized robot control laws. By considering exploration and coverage simultaneously, we enforce explicit coordination between the robots and yield optimal solutions.

3 Background

Let $\Omega \subset \mathbb{R}^N$ be a simply connected (in general non-convex) subset of $\mathbb{R}^N$ that represents the environment. There are n mobile robots in the environment with on-board range sensors, and in particular the position of the i^{th} robot is represented by $\mathbf{p}_i \in \Omega$ and the tessellation associated with it by W_i, $\forall i = 1, 2, \ldots, n$. By definition, the tessellations are such that $I(W_i) \cap I(W_j) = \emptyset, \forall i \neq j$, where $I(\cdot)$ denotes the interior of a set, and $\cup_{i=1}^{n} W_i = \Omega$. For a given set of robot positions $P = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$ and tessellations $W = \{W_1, W_2, \ldots, W_n\}$ such that $\mathbf{p}_i \in W_i, \forall i = 1, 2, \ldots, n$, the *coverage functional* is defined as:

$$\mathcal{H}(P, W) = \sum_{i=1}^{n} \mathcal{H}(\mathbf{p}_i, W_i) = \sum_{i=1}^{n} \int_{W_i} f(d(\mathbf{q}, \mathbf{p}_i))\phi(\mathbf{q}) d\mathbf{q},$$

where $d(\cdot, \cdot)$ is a distance function defined on Ω, $f : \mathbb{R} \to \mathbb{R}$ is a smooth and strictly increasing function in the range of d, $\phi : \Omega \to \mathbb{R}$ is a weight or density function, and $d\mathbf{q}$ represents an infinitesimal area or volume element. Throughout this paper we choose $f(x) = x^2$.

Lloyd's algorithm [8] and its continuous-time asynchronous implementations [4] are distributed algorithms for minimizing $\mathcal{H}(P, W)$ with guarantees on completeness and asymptotic convergence to a local optimum when Ω is convex and in an Euclidean distance setting (i.e. $d(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\|_2$).

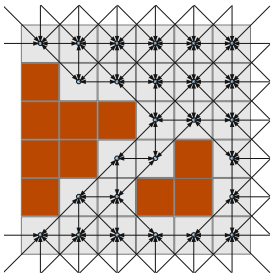


Fig. 1 An 8-connected grid graph created from a uniformly discretized environment

4 Coverage without Uncertainty

4.1 Geodesic Voronoi Tessellation

An extension of the continuous-time Lloyd's Algorithm algorithm to non-convex environments is presented in [10], where the distance function is defined as the *geodesic distance* in Ω. Consequently, the *Voronoi tessellations* under consideration are the *geodesic Voronoi tessellations*. That is, for a given P,

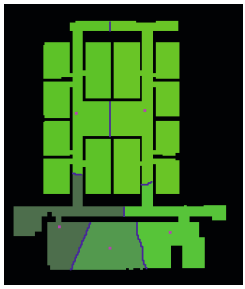
$$V(P) = \underset{W}{\operatorname{argmin}}\, \mathcal{H}(P, W) \iff V_i(P) = \{\mathbf{q} \in \Omega \,|\, d(\mathbf{q}, \mathbf{p}_i) \leq d(\mathbf{q}, \mathbf{p}_j), \forall j \neq i\}\,. \tag{1}$$

Geometric methods for computing such geodesic Voronoi tessellations in non-convex polygonal environments are detailed in [10, 1]. These methods suffer from the inherent drawback of high complexity in modeling cluttered real environments with noise and small obstacles. Moreover, in this work we wish to modify the metric such that it is non-uniform in Ω, and hence $d(\mathbf{q}, \mathbf{p})$ is no longer the Euclidean length of the shortest path between $\mathbf{q}$ and $\mathbf{p}$ lying in Ω.

4.2 A Search-Based Algorithm for Finding Geodesic Voronoi Tessellations

We propose a search-based algorithm for finding the geodesic Voronoi tessellations. We begin by uniformly discretizing Ω and creating a graph such that each node or vertex of the graph corresponds to a cell of the discretization with connections to admissible neighbors (see Fig. 1). This graph, $\mathcal{G}_\Omega$, consists of a vertex set, $\mathcal{V}(\mathcal{G}_\Omega)$, and edge set, $\mathcal{E}(\mathcal{G}_\Omega)$. For a vertex $\mathbf{q} \in \mathcal{V}(\mathcal{G}_\Omega)$, we use the same notation $\mathbf{q}$ to denote the coordinate of the vertex in the configuration space of the agents. $\mathcal{N}(\mathbf{q}) := \{\mathbf{s} \in \mathcal{V}(\mathcal{G}_\Omega) \mid \overline{\mathbf{sq}} \in \mathcal{E}(\mathcal{G}_\Omega)\}$ denotes the set of neighboring vertices of $\mathbf{q}$. The vertices joined by an edge $\varepsilon \in \mathcal{E}(\mathcal{G}_\Omega)$ are denoted by $\mathbf{v}_s(\varepsilon)$ and $\mathbf{v}_t(\varepsilon)$. We associate a cost, $c(\varepsilon)$, with every edge $\varepsilon = \overline{\mathbf{v}_s(\varepsilon)\mathbf{v}_t(\varepsilon)} \in \mathcal{E}(\mathcal{G}_\Omega)$. In particular, for elementary geodesic Voronoi tessellations, the cost of an edge is its Euclidean length (i.e. $c(\varepsilon) = \|\mathbf{v}_s(\varepsilon) - \mathbf{v}_t(\varepsilon)\|_2$).

(a) 3 robots in a simple environment (200 × 200 discretized).



(b) 5 robots in an office environment (170 × 200 discretized).

Fig. 2 The geodesic Voronoi tessellation of non-convex workspaces created using a uniformly discretized 8-connected grid-world. The robot locations are marked by enlarged magenta pixels.

For each of the robots we perform a Dijkstra's search [5, 3] in $\mathcal{G}_\Omega$ starting from the vertex where the robot itself is located, $\mathbf{p}_i$, and expand all the vertices in $\mathcal{G}_\Omega$. Thus, at the end of the expansions, for each vertex $\mathbf{q} \in \mathcal{V}(\mathcal{G}_\Omega)$ we obtain the values of $g_i(\mathbf{q})$, $\forall i = 1, 2, \ldots, n$, such that $g_i(\mathbf{q})$ gives the cost of the shortest path between $\mathbf{p}_i$ and $\mathbf{q}$ in $\mathcal{G}_\Omega$. The geodesic Voronoi tessellation is created by assigning the cells or vertices to the robot which has the least g-value at that node. That is, $V_i = \{\mathbf{q} \in \mathcal{V}(\mathcal{G}_\Omega) \,|\, g_i(\mathbf{q}) \leq g_j(\mathbf{q}), \forall j \neq i\}$ and $\mathbf{q} \in V_i \iff g_i(\mathbf{q}) \leq g_j(\mathbf{q}), \forall j \neq i$. We use the same notation V_i to denote the sub-set of vertices in $\mathcal{V}(\mathcal{G}_\Omega)$ that belong to the Voronoi tessellation V_i.

For the distributed architecture, each robot maintains its own copy of $\mathcal{G}_\Omega$, updating it (for probability of occupancy) using its own sensor readings as well as information acquired from its neighboring robots about parts of their copies of $\mathcal{G}_\Omega$ (described later).

Note that the least cost path between two points and hence the Voronoi tessellation, depends on the discretization of the environment and the definition of connectivity between neighboring vertices.

Figure 2 depicts geodesic Voronoi tessellations created using this algorithm. The boundary between two adjacent tessellations is such that the costs of the least cost paths from any cell on the boundary to either of the two robots that share the boundary are equal.

4.3 Continuous-Time Lloyd's Algorithm for Discrete Non-Convex Environments

The continuous-time Lloyd's algorithm for non-convex environments requires that each mobile robot follows the gradient of $\mathcal{H}(P) := \mathcal{H}(P, V(P))$ given by the formula [10]:

$$\frac{\partial \mathcal{H}}{\partial \mathbf{p}_i} = \int_{V_i(P)} \frac{\partial}{\partial \mathbf{p}_i} f(d(\mathbf{q}, \mathbf{p}_i))\phi(\mathbf{q}) d\mathbf{q}. \tag{2}$$

In a discretized environment, finding the gradient in (2) approximately reduces to searching among the neighboring vertices of the robot's current location such that $\mathcal{H}(P)$ is minimized for the new robot position in the next time-step. That is, we seek to find

$$\begin{aligned} \mathbf{p}_i^{t+1} &= \operatorname*{argmin}_{\mathbf{p} \in \mathcal{N}(\mathbf{p}_i^t)} \int_{V_i(P^t)} f(d(\mathbf{q}, \mathbf{p}))\phi(\mathbf{q}) d\mathbf{q} \\ &\approxeq \operatorname*{argmin}_{\mathbf{p} \in \mathcal{N}(\mathbf{p}_i^t)} \sum_{\mathbf{q} \in V_i(P^t)} f(d(\mathbf{q}, \mathbf{p}))\phi(\mathbf{q}) \Delta\mathbf{q}, \end{aligned} \tag{3}$$

where the superscripts denote the time-step, the summation is over all of the nodes in $\mathcal{V}(\mathcal{G}_\Omega)$ that are inside $V_i(P^t)$, and $\Delta\mathbf{q}$ is the area of the discretization cell at $\mathbf{q}$. Thus, the control law for each mobile robot reduces to driving from the current positions $\mathbf{p}_i^t$ to $\mathbf{p}_i^{t+1}$ as prescribed by (3). In order to find $\mathbf{p}_i^{t+1}$ from (3) for the i^{th} robot, at time instant t we perform Dijkstra's search and expand the states in $V_i(P^t)$ starting for each of the states in $\mathcal{N}(\mathbf{p}_i^t)$. This gives us the values for $d(\mathbf{q}, \mathbf{p}), \forall \mathbf{p} \in \mathcal{N}(\mathbf{p}_i^t), \mathbf{q} \in V_i(P^t)$; with $\mathbf{p}_i^{t+1}$ computed directly from (3) by computing and comparing the summations for each $\mathbf{p} \in \mathcal{N}(\mathbf{p}_i^t)$.

Figures 3(a)-3(d) show the evolution of the geodesic Voronoi tessellations and the trajectories followed by the mobile robots upon following the above control algorithm. In this example, we set $\phi(\mathbf{q}) = 1$. The environment is a 170×200 uniformly discretized 8-connected grid-world. Starting from the shown configuration convergence is achieved in less than 150 iterations. Running on a single Pentium processor (2.1 GHz, 4 GB RAM), each iteration takes on average 0.1 s (including computation of the current tessellations and the desired positions for the next time-step for all robots).

4.3.1 Projection of Centroid Method

In the previous section we do not discuss how to find the centroid, $\mathbf{C}_{V_i}$, of the geodesic Voronoi tessellation, V_i. In general, the direct computation of the *generalized centroid*,

$$\mathbf{C}_{V_i}^{gen} = \operatorname*{argmin}_{\mathbf{p}_i \in V_i} \int_{V_i} f(d(\mathbf{q}, \mathbf{p}_i))\phi(\mathbf{q}) d\mathbf{q} \tag{4}$$

in a non-convex environment is difficult [10]. However, a coverage control law over Voronoi tessellations such as that proposed in [4]:

$$\mathbf{u}_i = k(\mathbf{C}_{V_i} - \mathbf{p}_i), \tag{5}$$

requires knowledge of a centroid for the tessellations. Moreover, for exploration, we desire to implement the standard Lloyd's Algorithm or a semi-continuous version of it, which invariably require the computation of a *centroid*. In order to find an analog

of a centroid for a non-convex tessellation, we project the geometric centroid inside the tessellation. We compute

$$\mathbf{C}_{V_i} = \frac{\int_{V_i} \mathbf{q}\phi(\mathbf{q})d\mathbf{q}}{\int_{V_i} \phi(\mathbf{q})d\mathbf{q}}, \tag{6}$$

and if $\mathbf{C}_{V_i}$ lies outside V_i, find the point in V_i closest to it:

$$\underline{\mathbf{C}}_{V_i} = \underset{\mathbf{q}\in V_i}{\operatorname{argmin}} \|\mathbf{q} - \mathbf{C}_{V_i}\|. \tag{7}$$

This is an approximate method. In order to account for the non-uniformity of ϕ inside V_i, we may compensate by projecting the centroid in a *high ϕ region* in V_i. We modify (7) as,

$$\overline{\mathbf{C}}_{V_i} = \begin{cases} \underset{\mathbf{q}\in V_i, \phi(\mathbf{q})\geq\tau}{\operatorname{argmin}} \|\mathbf{q} - \mathbf{C}_{V_i}\| & \text{if it exists} \\ \underset{\mathbf{q}\in V_i}{\operatorname{argmin}} \|\mathbf{q} - \mathbf{C}_{V_i}\| & \text{otherwise.} \end{cases} \tag{8}$$

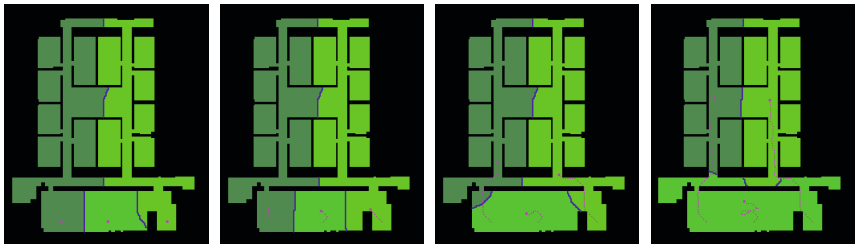
for some threshold τ, and use this projection.

The control law for the i^{th} robot given a discrete formulation reduces to taking one step towards $\overline{\mathbf{C}}_{V_i(P^t)}$ along the shortest path joining $\mathbf{p}_i^t$ and $\overline{\mathbf{C}}_{V_i(P^t)}$ (found via a single Dijkstra's search in V_i). This approach requires less computation than the gradient search method of (3). Further, simulation results suggest that the *projection of centroid* control method always converges and in cluttered environments differs little from the results obtained by the gradient search method. From these observations, we formulate the following conjecture.

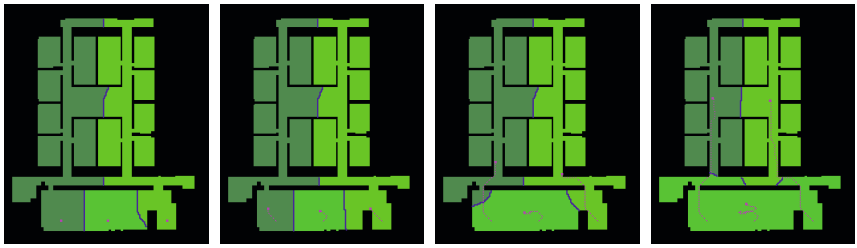
Conjecture 1 (Convergence of Projection of Centroid Method). If $\phi(\mathbf{q}) = k < \tau$ is uniform (constant) for all $\mathbf{q}$, then there exists robot positions $P^* = \{\mathbf{p}_1^*, \ldots, \mathbf{p}_n^*\}$ such that $\overline{\mathbf{C}}_{V_i(P^*)} = \mathbf{p}_i^*$, *i.e.* an equilibrium point, and the Projection of Centroid control method drives the robots to such a configuration.

Proof for a special case. We present a partial proof for a certain type of V_i. For Euclidean metric, if the $\overline{\mathbf{C}}_{V_i} = \mathbf{C}_{V_i}^{gen}$, this conjecture becomes a theorem and the control law described above is guaranteed to converge [10]. We thus investigate the cases where $\overline{\mathbf{C}}_{V_i}$ indeed is the *generalized centroid*. Clearly, $\mathbf{C}_{V_i} = \mathbf{C}_{V_i}^{gen} \Rightarrow \overline{\mathbf{C}}_{V_i} = \mathbf{C}_{V_i}^{gen}$ (since $\mathbf{C}_{V_i}^{gen}$ always lies inside V_i). The condition under which $\mathbf{C}_{V_i} = \mathbf{C}_{V_i}^{gen}$ is that $d(\mathbf{q}, \mathbf{C}_{V_i}) = \|\mathbf{q} - \mathbf{C}_{V_i}\|\ \forall \mathbf{q} \in V_i$. This condition is equivalent to saying that V_i be *star-shaped* [11] with respect to $\mathbf{C}_{V_i}$. A trivial case of this condition is when V_i is convex, when the algorithm becomes equivalent to the continuous-time Lloyd's Algorithm [4].

For comparison, Figs. 3(e)–3(h) show the evolution of the geodesic Voronoi tessellations and the trajectories followed by the mobile robots using this control method. Once again, we use $\phi(\mathbf{q}) = 1$, and convergence takes place in less than 150 iterations. However in this case the computation time per iteration is on an average 0.03 s.



(a) $t = 0$ (b) $t = 10$ (c) $t = 50$ (d) $t = 150$ (converged)



(e) $t = 0$ (f) $t = 10$ (g) $t = 50$ (h) $t = 150$ (converged)

Fig. 3 Continuous-time Lloyd's algorithm in a discretized setting for optimal coverage. Figures 3(a)-3(d) use the gradient search method while Figures 3(e)-3(h) use the projection of centroid method.

5 Simultaneous Exploration and Coverage of Unknown or Partially-Known Environments

In this section, we consider the problem of deploying n mobile robots in an unknown or partially known environment, which upon collaborative exploration of the environment, will converge to an optimal or near-optimal coverage.

5.1 Entropy as Density Function

In order to address this problem each mobile robot maintains and communicates a probability map for the discretized environment such that $p(\mathbf{q})$ is the probability that the vertex $\mathbf{q}$ is inaccessible (*i.e.* occupied or represents an obstacle), for all $\mathbf{q} \in \mathcal{V}(\mathcal{G}_\Omega)$. A threshold on the value of probability determines whether a particular node in $\mathcal{V}(\mathcal{G}_\Omega)$ is occupied/inaccessible for computation of the Geodesic Voronoi tessellations as well as control. Moreover the *Shannon entropy* for each cell can be computed as follows,

$$e(\mathbf{q}) = p(\mathbf{q}) \ln(p(\mathbf{q})) + (1 - p(\mathbf{q})) \ln(1 - p(\mathbf{q})). \tag{9}$$

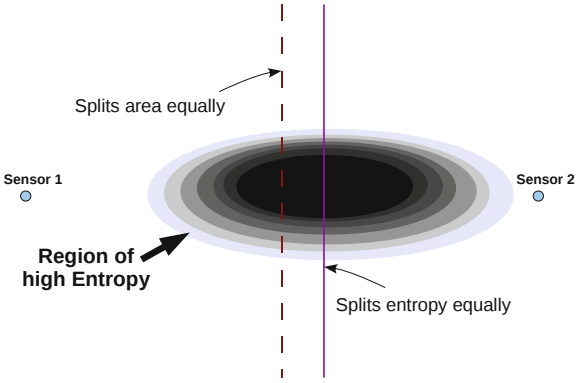


Fig. 4 Entropy-weighted Voronoi tessellation

This gives us an *Entropy map*, *i.e.* a value of entropy associated with each vertex $\mathbf{q}$ - a map that represents uncertainty or the need to gather information within the environment. The Shannon entropy is such that it assumes high values for vertices for which the uncertainty is high (i.e. probability is close to 0.5), whereas it is low for known or visited vertices. Thus, we identify the weight or density function $\phi(\cdot)$ with the entropy $e(\cdot)$. This, by the construction of the control laws described before, will drive the mobile robots towards regions of high entropy within the robot's own tessellation, hence resulting in exploration of the environment.

For exploration of an unknown environment it is desired that we follow an analog of the traditional Lloyd's algorithm, where each robot visits completely a computed projected centroid of a tessellation at an earlier time-step (we call it a *target*), hence exploring the region, and subsequently recompute the next target, which is the projected centroid of the current tessellation.

5.2 Entropy-Based Metric

The geodesic Voronoi tessellation performed according to (1) ensures that the boundaries of the tessellations "bisect" the area lying between the robots. While the metric d for this can be the geodesic distance in the case of the coverage problem, for exploration and for environments with uncertainty the tessellation boundaries need to be such that they "bisect" the uncertainty (or entropy) among the adjacent robots for cooperative exploration. This notion is illustrated in Fig. 4, where a high entropy region is placed asymmetrically between two robots in a convex environment without obstacles. The dashed line shows the boundary of a Voronoi tessellation created using the standard distance metric. However, one mobile robot has a larger unexplored region than the other. An alternate division is depicted with a solid line that splits the unexplored region equally.

We now redefine $d(\cdot,\cdot)$ to accommodate uncertainty in the tessellation. Let $\Gamma(\mathbf{p},\mathbf{q})$ represent the set of all paths in Ω connecting $\mathbf{p}$ and $\mathbf{q}$. Then the original definition of the geodesic distance is written as $d(\mathbf{p},\mathbf{q}) = \min_{\gamma\in\Gamma(\mathbf{p},\mathbf{q})} \int_\gamma dl$, where dl represents an elemental length along γ. We modify the definition as follows:

$$d(\mathbf{p},\mathbf{q}) = \min_{\gamma\in\Gamma(\mathbf{p},\mathbf{q})} \int_\gamma e(\mathbf{r})dl,$$

where $\mathbf{r}$ is a point on γ.

In terms of finding the Voronoi Tessellations by performing Dijkstra's searches in $\mathcal{G}_\Omega$ as described earlier, the only required change is to weigh the edges of the graph $\mathcal{G}_\Omega$ by entropy in those regions instead of the Euclidean length of the edges. In particular, we now define the cost of an edge ε as

$$c(\varepsilon) = \frac{e\left(\mathbf{v}_s(\varepsilon)\right) + e(\mathbf{v}_t(\varepsilon))}{2} + \eta\|\mathbf{v}_s(\varepsilon) - \mathbf{v}_t(\varepsilon)\|_2$$

where we add the second therm with a very small value of η in order to compensate for noise in near-zero values of entropy and to make sure that the cost of an edge doesn't vanish. Performing Dijkstra searches in this weighted graph and creating tessellations using the same procedure as before will split the unexplored regions between two neighboring robots equally.

5.3 Time Dependence of Entropy, Coverage, and Convergence

We now detail how the probability map is updated based on the sensor readings. For the discussion that follows, $p^t(\mathbf{q})$ represents the estimated probability of occupancy of $\mathbf{q}$ at the t^{th} iteration based on all measurements.

5.3.1 Inter-robot Communication

As discussed earlier, for the distributed architecture, each robot maintains its own copy of probability and entropy maps. Each updates its own maps based on readings from its own on-board sensor as well as information acquired from its neighboring robots about parts of their copies of their probability maps. A sensor fusion model (described in next section) is used to aggregate the data. For communicating its own probability map to other robots, each robot broadcasts the *new* information acquired by its own sensor over a *time window* or *phase*. Essentially the broadcasting of probability maps by each robot is done in *phases*. During a phase, a robot broadcasts a constant message (part of its own probability map) with a fixed timestamp over and over (repeatedly). This is to make sure that other robots receive this message. The robot also broadcasts its unique identity along with the message. Also, instead of broadcasting the whole probability map in each phase, each robot broadcasts only whatever new it has sensed during the previous broadcast phase. Thus the broadcasted information actually comprises of a small window in the whole probability

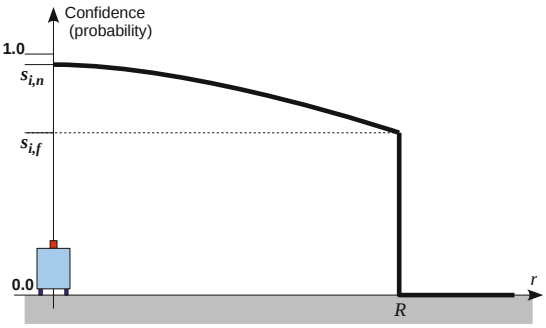


Fig. 5 The sensor model

map as well as in time, inside which the probability readings have changed. This makes each broadcast messages rather small. Essentially each robot maintains two buffers: The *current sensing buffer*, and the *broadcast buffer*. New readings from a robot's own laser sensor are added to the current sensing buffer, while things in the broadcast buffer are broadcasted. At the end of a broadcast *phase* the content of the broadcast buffer is pushed into the main probability map maintained by the robot, the content of the current sensing buffer is copied into the broadcast buffer, and the current sensing buffer is cleared for new sensor data. The information received from other robots about their map are directly added to the main probability map. This *differential* approach of communication significantly reduces the communication overhead required for sharing map data.

5.3.2 Sensor Model

We use a sensor model for each robot, $s_i(r)$, which gives the probability that the i^{th} robot's sensor measures the state of a grid cell located at a distance r from it correctly. In particular, in our simulations we use,

$$s_i(r) = \begin{cases} s_{i,n} + \frac{r^2}{R_i^2}(s_{i,f} - s_{i,n}) & \text{if } r \leq R_i \\ 0 & \text{otherwise,} \end{cases}$$

where R_i is the sensor range, and $0 \leq s_{i,f} \leq s_{i,n} \leq 1$ gives the far and near values of the confidence of the sensor.

Thus, if at time-step t the sensor of the i^{th} robot receives a measurement $z_i^t(\mathbf{q})$ (which is 1 for occupied, and 0 for unoccupied) for the cell $\mathbf{q}$, the probability that the cell is occupied based only on this measurement is given by $u_i^t(\mathbf{q}) = z_i^t(\mathbf{q})\, s_i(\|\mathbf{q} - \mathbf{p}_i^t\|) + (1 - z_i^t(\mathbf{q}))(1 - s_i(\|\mathbf{q} - \mathbf{p}_i^t\|))$. We use a sensor fusion model to compute the net probability of occupancy for the cells based on the individual measured probabilities. In particular, one can compute $p^t(\mathbf{q}) = g^{-1}\left(\frac{\sum_{i,t'} g(u_i^{t'}(\mathbf{q}))}{\sum_{i,t'} 1}\right)$, where g is a strictly increasing function in $[0, 1]$, and the summations are taken over all the measurements by all sensors over all time instants [16]. For our experiments we

choose $g(x) = x^m,\ m > 0$. We note that by choosing $m \to \infty$, the value of $p^t(\mathbf{q})$ essentially becomes the *supremum* of all the measurements for $\mathbf{q}$. Alternatively, choosing $g(\cdot) = \log(\cdot)$ gives the geometric mean of the measurements, which has also been used in [16].

In order to compensate for the sensor noise the entropy map is smoothed by passing it through a min-filter. The smoothed entropy map is consequently used for computing the density function.

5.3.3 Time-Varying Density Function

A consequence of updating the probability map is that the entropies, and hence the weight function, ϕ, becomes a function of time:

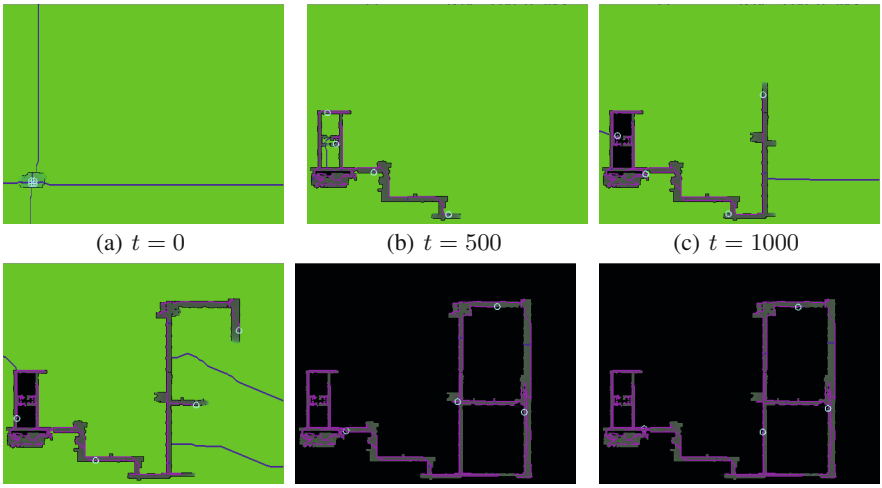
$$\phi(\mathbf{q}, t) = e(\mathbf{q}, t) = p^t(\mathbf{q}) \ln(p^t(\mathbf{q})) + (1 - p^t(\mathbf{q})) \ln(1 - p^t(\mathbf{q})).$$

Conjecture 2 (Exploration and Convergence Guarantee). Assuming Conjecture 1 is true and the individual robot motion at each time step is determined by the Projection of Centroid Method (8), there exists a τ', $0 < \tau' \leq \tau$ such that choosing $\phi(\mathbf{q}, t) = \max(\tau', e(\mathbf{q}, t))$ ensures complete exploration of the environment and convergence of the algorithm.

Proof. We assume there exists an ϵ radius around each mobile robot such that it is able to sense the occupancy and reduce the entropy of the cells within this radius below the value of τ' in a permanent manner. Due to the choice of our control method, each mobile robot drives closer to $\overline{\mathbf{C}}_{V_i}$ at every time-step. However, as long as there exists at least one cell $\mathbf{q} \in \mathcal{V}(\mathcal{G}_\Omega)$ such that $e(\mathbf{q}, t) \geq \tau$, a robot will drive to that cell where $\phi(\mathbf{q}, t) = \max(\tau', e(\mathbf{q}, t)) \geq \tau$ (due to equation (8) and since $\tau' \leq \tau$), and subsequently reduce the entropy of the region around $\mathbf{q}$ below τ'. This process continues until the entropy of all the cells in the map goes below τ such that $\phi(\mathbf{q}, t_{covered}) = \max(\tau', e(\mathbf{q}, t_{covered})) \in [\tau', \tau)$ for all $\mathbf{q}$. This guarantees exploration of the environment with all cells having final entropy less than τ. Note that while τ' is sensor specific, τ is a design variable. Thus, if we choose $\tau = \tau'$, the density function becomes $\phi(\mathbf{q}, t_{covered}) = \tau' = \tau$, which is constant and independent of time throughout the environment. Consequently by Conjecture 1 convergence is achieved at some $t_{converged} \geq t_{covered}$.

5.3.4 The Overall Algorithm

So far we have described the various components of the algorithm. To put those in perspective, the steps below are what goes on at a higher level on each robot in sequence while exploring and covering an unknown or partially known environment in a distributed fashion.



(a) $t = 0$ (b) $t = 500$ (c) $t = 1000$

(d) $t = 1600$ (e) $t = 2600$ (complete map built) (f) $t = 2750$ (convergence)

Fig. 6 Exploration and coverage of a large unknown environment. Green indicates uncertainty.

i. Each robot maintains its own probability, entropy and obstacle maps.
ii. Each robot use sensor data as well as communicate with its neighbors to update the maps. They also communicate their locations.
iii. Each robot computes its own entropy-weighted Voronoi tesellation and the corresponding weighted projected centroid, and take a step towards that along the shortest path.

5.4 Results

Figure 6 shows the screenshots from a simulation of four robots exploring a large (1000×783 uniformly discretized) cluttered environment. The boundaries of the tessellations are shown by the bold blue lines. The robot positions are encircled by cyan circles. The dark lines show the robot trajectories. The intensity of the pixels in the environment represent the entropy, and the unreachable regions are colored in black. The mobile robots begin at the room in the lower left with no prior knowledge about the environment, hence the highest value of entropy, $\ln(0.5)$, is assigned to each cell. Besides collaboratively exploring the environment the robots distribute themselves in such a way that they maintain proper coverage of the explored environment both during exploration and after completely building the map. The mobile robots attain full exploration, coverage, and convergence within $t = 2750$ iterations. Each iteration, which involves computing the voronoi tessellations as well as the control commands for all the robots, takes about $1.7s$ running on a single processor as described in earlier results.

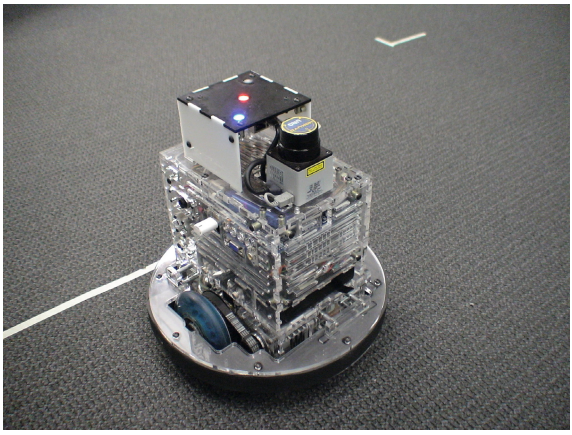


Fig. 7 The Scarab mobile robot platform

6 Conclusion and Future Work

We presented a search-based algorithm for computing a geodesic Voronoi tessellation in discrete environments. We propose a method for computing the centroid of the Voronoi cells given a tessellation of a discrete environment, permitting the application of centroid-based robot control laws for cooperative coverage. Uncertainty in the environment description is introduced through the development of an entropy-based metric; enabling the computation of the instantaneous geodesic Voronoi tessellation given an uncertain environment. We comment on the coverage and convergence guarantees resulting from this approach and present results that demonstrate through simulation its application in an office environment.

There are a few limitations in this paper that we are currently working on. First, as we move to experimentation with real robots, we must address real world issues surround localization and state estimation for the robots as well as inter-robot communication. To this end, we have already integrated our simulation model within the ROS (Robot Operating System) framework, and have started extending the implementation for running preliminary experiments on multiple Scarab robots [9] (see Fig. 7) which allow for on-board computation and localization using laser range sensors and monocular cameras. We are incorporating the anisotropy and finite field-of-view constraints that are characteristics of these sensors within our sensor model and the uncertainty associated with localization in our entropic measure. In addition, we are exploring models for inter-robot communication to relax the current assumption of a complete communication graph in the paper. Finally, we are also addressing algorithmic improvements to allow distributed computation and to enhance efficiency.

References

1. Aronov, B.: On the geodesic voronoi diagram of point sites in a simple polygon. In: Proc. of the Sym. on Computational Geometry, pp. 39–49. ACM, New York (1987), doi:http://doi.acm.org/10.1145/41958.41963
2. Burgard, W., Moors, M., Stachniss, C., Schneider, F.: Coordinated multi-robot exploration. IEEE Trans. Robot. 21(3), 376–378 (2005)
3. Cormen, T.H., Stein, C., Rivest, R.L., Leiserson, C.E.: Introduction to Algorithms. McGraw-Hill Higher Education (2001)
4. Cortes, J., Martinez, S., Karatas, T., Bullo, F.: Coverage control for mobile sensing networks. IEEE Trans. Robot. Autom. 20(2), 243–255 (2004)
5. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik 1, 269–271 (1959)
6. Fox, D., Ko, J., Konolige, K., Limketkai, B., Schultz, D., Stewart, B.: Distributed multirobot exploration and mapping. Proc. of the IEEE 94(7), 1325–1339 (2006)
7. Howard, A., Parker, L.E., Sukhatme, G.S.: Experiments with a large heterogeneous mobile robot team: Exploration, mapping, deployment and detection. Intl. J. Robot. Research 25(5-6), 431–447 (2006)
8. Lloyd, S.P.: Least squares quantization in PCM. IEEE Trans. Inf. Theory 28, 129–137 (1982)
9. Michael, N., Fink, J., Kumar, V.: Experimental testbed for large multi-robot teams: Verification and validation. IEEE Robot. Autom. Mag. 15(1), 53–61 (2008)
10. Pimenta, L.C.A., Kumar, V., Mesquita, R.C., Pereira, G.A.S.: Sensing and coverage for a network of heterogeneous robots. In: Proc. of the IEEE Conf. on Decision and Control, Cancun, Mexico, pp. 3947–3952 (2008)
11. Preparata, F.P., Shamos, M.I.: Computational geometry: an introduction. Springer-Verlag New York, Inc., New York (1985)
12. Schwager, M., McLurkin, J., Rus, D.: Distributed coverage control with sensory feedback for networked robots. In: Proc. of Robot.: Sci. and Syst., Philadelphia, PA (2006)
13. Schwager, M., Slotine, J.E., Rus, D.: Decentralized, adaptive control for coverage with networked robots. In: Proc. of the IEEE Intl. Conf. on Robot. and Autom., Rome, Italy, pp. 3289–3294 (2007)
14. Stachniss, C.: Exploration and mapping with mobile robots. Ph.D. thesis, University of Freiburg, Freiburg, Germany (2006)
15. Stachniss, C., Grisetti, G., Burgard, W.: Information gain-based exploration using rao-blackwellized particle filters. In: Proc. of Robot.: Sci. and Syst., Cambridge, MA, pp. 65–72 (2005)
16. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). The MIT Press (2005)
17. Wu, L., Garcia, M.A., Puig, D., Sole, A.: Voronoi-based space partitioning for coordinated multi-robot exploration. J. Physical Agents 1(1), 37–44 (2007)

Evaluating Efficient Data Collection Algorithms for Environmental Sensor Networks

William C. Evans, Alexander Bahr, and Alcherio Martinoli

Abstract. Although there exists a large body of work on efficient data collection in sensor networks, the vast majority of proposed techniques have not been implemented on real networks or thoroughly studied on real data. As algorithm performance is highly dependent on the characteristics of the data being reported, it is very difficult to make suggestions as to the relative performance of any particular method. In this work we seek to compare and evaluate existing approaches to efficient data gathering in the specific context of environmental monitoring. We examine a choice algorithm that has not, to the best of our knowledge, been thoroughly studied on real data. We detail a number of algorithmic modifications necessary to bring it from theory to reality, and study the algorithm's performance in simulation using extensive traces from real world sensor network deployments.

1 Introduction

Low-cost sensor networks are becoming ubiquitous, as they have a broad range of applications from target tracking [2, 7] to health monitoring [10, 11], and our specific focus in this paper, environmental monitoring. This paper is part of an ongoing effort to deploy distributed intelligent algorithms into sensor networks consisting of resource-constrained nodes. The overall idea is that data collected by a network should be not only be used by the end user but should also at the same time allow more intelligent control of the activities of its nodes, possibly achieving the same level of data accuracy with less power consumption. Ultimately, our goal is to validate and subsequently deploy such algorithms in real world scenarios.

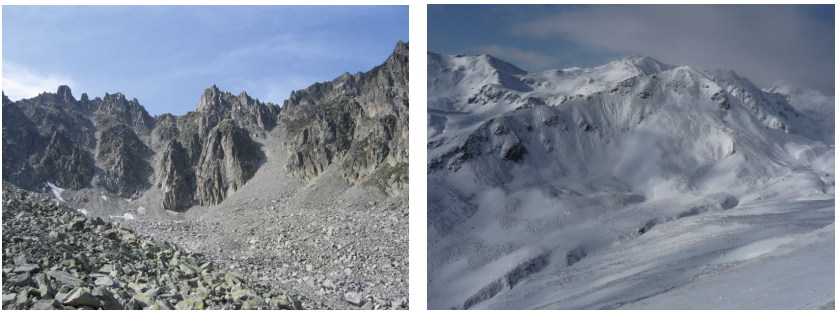
We base our study on typical configurations used in environmental campaigns in our local Alpine region, where 10-20 sensor nodes are placed such that they are able

William Evans · Alexander Bahr · Alcherio Martinoli
Ecole Polytechnique Fédérale de Lausanne (EPFL), 1015 Lausanne, Switzerland
e-mail: firstname.lastname@epfl.ch

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 77–89.
springerlink.com

to communicate via short range wireless transceivers. These networks always include at least one sink that uses long range communication (e.g., GPRS) to forward the data to a central location.

Currently there is a significant gap between theory and real world usage. While many algorithms for efficient data collection from sensor networks have been proposed, many existing systems simply take the naive ”always broadcast“ approach [4]. Indeed, the mention of efficient monitoring techniques in data-oriented sensor network deployments is rare, in general authors do not discuss the possibility of an intelligent data collection approach. The naive approach is proven and reliable, as uniform sampling with high relative frequency (e.g., one sample per minute) provides environmental engineers with easily manipulated data that have built-in redundancy due to the fact that many environmental fields change slowly the majority of the time (i.e. on the order of tens of minutes to days). However, there is much to gain by reducing energy consumption; doing so allows developers to lengthen sensor network autonomy, increase sampling frequency, lower reliance on expensive power sources, and so on.



(a) (b)

Fig. 1 SensorScope has seen several deployments under diverse conditions, including (a) a two-month deployment on Le Génépi comprised of 16 stations, and (b) a three-month deployment in the Wannengrat using 18 stations

Fig. 2 Three weeks of sensor data were collected on the rooftop of EPFL’s GR building

This gap is in part to due to a lack of comprehensive analysis of existing algorithms. Authors tend to test their algorithms on small datasets that may not be representative for more general applications, and hardware implementations are rare. We seek to perform an exhaustive comparison of a large number of such algorithms as they apply to environmental monitoring in terms of vulnerability to node failure and message loss, communication overhead, and data accuracy.

2 Related Work

In temporal suppression schemes, each node uses its own history of measurements to determine if a new value can be inferred by the network sink (i.e., it does not need to be transmitted). A simple example would be transmitting measurements only when they differ from the previous value. Typically these approaches make use of much more complex models, often providing bounded error.

The Probabilistic Adaptable Query (PAQ) system is one notable such scheme based on time series forecasting [21]. It uses autoregressive models maintained locally per sensor in order to keep from sending data directly to the sink. Instead, nodes communicate model parameters as necessary in order to keep the sink's predictions within some defined error bound. Tulone and Madden extend this work with their Similarity-based Adaptive Framework (SAF) [20], adding robustness to quick changes in data trends as well as a location-independent clustering technique that allows the detection of redundant nodes.

On the other hand, spatial suppression exploits spatial correlations between nearby sensor nodes in order to reduce communication load.

Many spatial suppression algorithms attempt to detect and deactivate sets of redundant nodes. Prorok et al. study hierarchical network topologies based on spatial clustering [13]. In this approach, cluster heads may choose to prune their children if the part of the monitored field they represent is highly isotropic as defined by some statistically computed threshold. Arici and Altunbasak propose using a first-order model to determine the predictability of particular nodes [1]. They define some of the nodes in the network as *macronodes* which attempt to fit a plane over their neighbors' positions and data, commanding easily predictable nodes to stop reporting measurements for some period of time. Similarly, Willett et al. define the idea of a *fusion center* that is responsible for estimating a field based on received sensor measurements and then directly deactivating redundant nodes [22].

Chu et al. propose the use of replicated dynamic probabilistic models between the sink and *disjoint cliques* of data sources [6]. The sink then uses these models to predict future sensor data. If the *root* of a clique observes data inconsistent with the sink's current prediction model, a subset of the clique's recent observations are sent and the sink's model is updated as necessary.

A third and wholly separate approach, compressed sensing, draws on recent advances in signal processing that have interesting implications in sensor networks. This technique allows the accurate reconstruction of a signal while sampling at

a rate that does not satisfy the Nyquist sampling theorem. Note that unlike other approaches, compressed sensing reduces the number of measurements needed, an obvious advantage when using active sensors, i.e., sensors that require considerable power just to sample. However compressed sensing also imposes strict requirements on the properties of measured data and makes in-network processing very difficult.

Haupt and Nowak develop the idea of compressed sensing in a multi-sensor scenario [8]. They envision using a uniform array of sensors to measure some phenomenon, each transmitting its processed results to a common destination, using inter-signal correlations to reduce power usage. Baron et al. generalize this idea, allowing the use of irregular spatial sampling and additionally taking advantage of sensors' recent history [3].

Outside of continuous monitoring, many approaches to efficient data collection seek to reduce overall message volume by eliminating uninteresting data in-network. TinyDB [9] provides such functionality, returning sensor data to the sink in response to simple aggregation queries such as SUM or MAX. Other techniques use in-network triggers to decide when data should be sent to the sink. Yang et al. present a Two-Phase Self-Join scheme that accepts complex monitoring queries from the user and informs the sink should an appropriate event be detected [23].

In [14], Sadagopan et al. compare their query resolution algorithm for sensor networks, ACQUIRE, with other more basic approaches. They conduct a theoretical analysis using mathematical models, using the results to tune algorithm parameters and draw performance estimates. While this approach allows theoretical insight into algorithmic performance, the authors make a number of assumptions that make their conclusions unlikely to extend to the uncontrolled conditions considered in this paper (e.g., uniform deployment and communication range). Our work seeks to compare algorithms experimentally, specifically evaluating algorithmic performance in a real-world context.

3 Materials and Methods

SensorScope stations [4] are a replacement for traditional large, centralized weather stations that may be time-consuming and expensive to deploy (see Figure 2). Instead they are a distributed array of smaller, cheaper stations that leverage greater coverage area and ease of deployment to provide more valuable data. SensorScopes guiding principle is to provide environmental scientists with real-time, total access to the data collected by their stations. This is best reflected by their online data browser, Climaps [17], which serves as an interface for both retrieving a particular dataset (from a specific time period, collection of sensors and/or set of stations), as well as monitoring the battery levels and communication links of users' stations.

The stations themselves are built around 2-3m metal poles. Up to seven sensors may be attached along with the main controller, protected by a hermetically sealed

container. Each station is attached to an energy source, in our case consisting of a large NiMH battery that is recharged by a small solar panel. Network sinks have an attached GPRS module for sending data from the network to off site SensorScope servers for viewing and archival. In this paper we use data collected by an attached SHT75 air humidity/temperature sensor and Zytemp TN901 infrared thermometer for sensing surface temperature.

The station itself is controlled by a ShockFish TinyNode 584 [18] running TinyOS 2.x. Local communication is performed on the 868MHz band using a Semtech XE1205 radio transceiver, with a range of one kilometer given strong line of sight.

SensorScope stations are an ideal platform for deploying efficient monitoring algorithms. The hardware platform has already proved itself successful in many previous deployments in various locales (see Figure 1), with many further deployments in planning by environmental scientists around Switzerland. Currently SensorScope stations take the naive approach to data collection, i.e., stations simply broadcast every sensor measurement made at regular time intervals. By implementing a data collection algorithm that takes advantage of the strong spatial and temporal correlations often present in environmental fields, we can reduce SensorScope stations' reliance on expensive power systems, driving down per-station cost.

SensorScope was specifically developed for the purpose of monitoring environmental phenomena. Environmental fields tend to lend themselves well to the suppression-based approaches discussed in the previous section. Strong temporal patterns are often present in environmental data, which may be described as having *trend* and *seasonal* components [21]. One clear example of a seasonal pattern is the typical day/night cycle, which has a clearly visible effect on ambient temperature as seen in Figure 3. Environmental data is also prone to high degrees of spatial correlation (see Figure 4); it is clear that spatio-temporal suppression is highly appropriate in this domain.

4 Constraint Chaining

In this section we explain *constraint chaining* (CONCH) [19], a technique that monitors constraints between adjacent nodes rather than the absolute values of nodes themselves. Like other recent work in efficient monitoring, CONCH uses a combination of spatial and temporal suppression to reduce network traffic. The in-network computational cost of this approach is minimal, making it a strong candidate for real world usage on the SensorScope platform. The algorithm provides real-time, accurate monitoring at a potentially greatly reduced communication cost while being flexible enough to tailor to specific data collection scenarios.

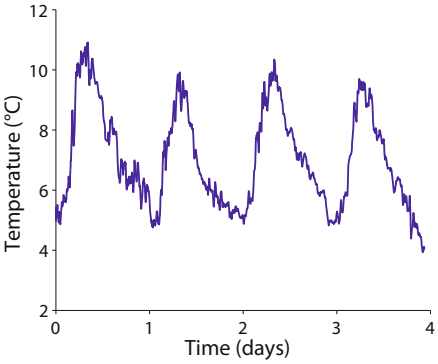


Fig. 3 Temperature data over a four-day period from a SensorScope deployment in the Génépi. This plot illustrates the seasonal component often present in environmental data.

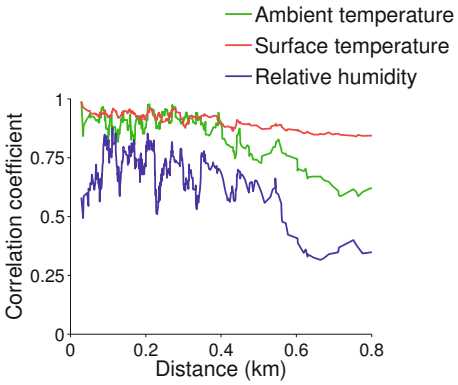


Fig. 4 Spatial correlation varies greatly depending on the type of environmental process in question. We calculate the correlation between each pair of sensor nodes within each deployment listed in Table 1 and plot these values directly according to distance.

4.1 Basic Algorithm

This approach revolves around a subset of the network's communication edges that we will refer to as the monitored edges. The nodes adjacent to each monitored edge are given different responsibilities for that edge. One node is assigned the role of *reporter*, responsible for sending a message to the sink every time the relative value of the two nodes along that edge changes. The opposite node is called the *updater*, and sends a message to the aforementioned reporter whenever its own value changes. Note that messages are not sent to the sink if both nodes change by the same amount simultaneously or if neither node changes at all. Nodes determined to be outliers, i.e. those that do not change in a pattern similar to any of their neighbors, are monitored directly and thus report their measured values to the sink whenever they change- we refer to these as monitored nodes.

A particular set of monitored edges, monitored nodes and updater/reporter assignments is called a *Conch plan* (see Figure 5 for an example). Ideally, we would like our plan to be configured such that the values along monitored edges change as infrequently as possible. Note that in order for a plan to be valid, each monitored

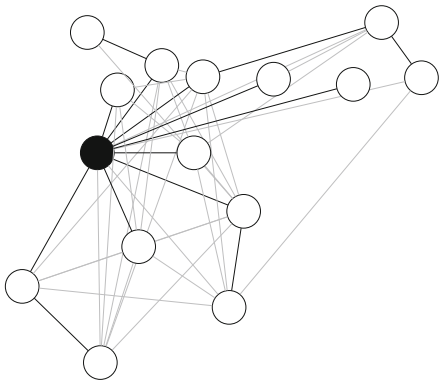


Fig. 5 Example CONCH plan from simulation over data from a real world SensorScope deployment in the Génépi. The sink is filled in black. Monitored edges are shown in black, and communication edges are shown in light gray.

edge must be connected to a tree that includes at least one directly monitored node (or the sink). In this way the sink is able to infer the values of all members of the network by chaining relative values along monitored edges, starting from this subset of nodes where absolute values are known.

Our search for a near-optimal CONCH plan is carried out in three steps. First, an arbitrary plan is chosen and used for a given period of time. In this phase network performance will be suboptimal, however note that as long as the plan is valid, no data will be lost. During this period we build up a data history that we will use to create a plan better suited to the observed environmental patterns. Once a hopefully representative dataset has been gathered (discussed further in Section 4.2), we assign a cost to all communication edges in the network. The cost of monitoring edge e is $freq(e) \times dist(e)$, where $freq(e)$ is the number of times the relative value along edge e changes throughout the dataset, and $dist(e)$ is the edge's distance from the sink. We also add an imaginary edge from every station s directly to the sink with cost $freq(s) \times dist(s)$. The minimum spanning tree over this graph represents the set of monitored edges and monitored nodes.

All that remains is our choice of updaters and reporters. The authors of this algorithm propose a mixed integer program that seeks to minimize network traffic given the data observed thus far. Their program accounts for the per-message and per-byte sending and receiving costs for the radios used as well as the number of messages required given a particular configuration. Again, the details of this formulation are left to [19].

4.2 Modifications

In this section we explain two algorithmic modifications necessary before CONCH can operate in the kind of network configuration described previously.

The first change deals with CONCH plan optimization. The linear program originally proposed contains in its objective function a number of variables that increases

exponentially with the number of time steps and stations. As previously stated, it is important to optimize over a length of time that is representative for the phenomenon being measured, e.g., one day/night cycle. In many deployments, SensorScope stations are configured to take a sensor reading once per minute. For a modestly sized network of 10 nodes over one day of measurements at the aforementioned rate, the objective function contains over 25,000 variables, with an even larger number of constraints. The result is an intractable linear program that is simply not solvable in a time frame appropriate for this problem.

Consider that only reporters are responsible for communicating data to the sink, and an optimal CONCH plan will be likely to place reporters closer to the sink than their corresponding updaters. Bearing this in mind, we simply iterate through all edges in the plan, marking the adjacent node closest to the sink as its reporter, and the other as its updater. This simplication may be cause excessive energy usage if the node chosen as updater changes value significantly more often than its reporter, however we do not observe such scenarios in our experiments.

Second, we introduce a strategy for repeatedly updating the CONCH plan. In evaluating their algorithm, the authors of CONCH build a CONCH plan using data collected over an initial training period. This plan is then used for the remainder of the test. In other words, they assume that relationships between nodes will never change. In a real environment it may be beneficial to explore more complicated techniques for building and maintaining an optimal CONCH plan; outdoor environments can be extremely dynamic, with unpredictable local and regional weather patterns playing a significant role on the spatial relationships present.

As creating and disseminating a new CONCH plan is a cheap operation compared to sensor reporting, we examine the performance of a scheme that builds an optimal plan using the previous N hours of network data, repeating this process after another N hours have passed.[1] One could imagine more dynamic schemes in which CONCH plan generation is triggered by some condition detected at the sink; however for the sake of brevity we leave such approaches to further study.

5 Simulation

5.1 Existing Datasets

We have collected a number of datasets from previous SensorScope deployments throughout Switzerland. We specifically selected deployments of about 10 or more nodes where the network was dense enough to be connected, yet sparse enough to require multi-hop communication. The results in this paper are found using datasets from three such deployments (see Table 1), downloaded with geographical metadata from Climaps [17], a data visualization and archiving system for environmental data.

[1] During the first N hours of operation we have no available data from which to guess stable relationships between nodes, so we simply use the routing tree as a temporary measure.

Table 1 Past SensorScope deployments provide over five years of individual station data appropriate for our algorithm evaluation framework

Location	Nodes	Duration (days)
Génépi	15	65
Great St. Bernard Pass	16	43
GR Rooftop	8	17

It is the focus of this paper to examine algorithm performance via realistic simulation. Unfortunately, it is rare for data-oriented sensor network deployments to record the data necessary to accurately do so. In particular, we note a lack of useful topology and link quality information. We evaluated a number of approaches for generating simulated topology information, however we found such techniques to be inappropriate due to the nature of these deployments (i.e., in outdoor, uncontrolled environments, some times even including manual, undocumented redeployment over the experiment period). Indeed, as one may observe in Figure 5, wireless communication is highly unpredictable under our circumstances.

Incorporating datasets from new sources into our framework is a straightforward process. There are many publicly available environmental datasets that would be useful in our simulations, and we suggest where such datasets may be found in Section 6.

5.2 Results

We have developed a modeling framework sufficient for simulating a wide variety of algorithms over datasets from any external source. While we currently only make use of three datasets and three algorithms, we have established a common evaluation platform for performing deeper studies in the future. For further discussion, see Section 6.

All algorithms use error bounds according to individual sensor accuracy as listed on the relevant datasheet. Thus, for the Sensiron SHT75, we report ambient temperature to an accuracy of $\pm 0.3^{\circ}$C and relative humidity to $\pm 1.8\%$ [16]. We report surface temperature as measured by the Zytemp TN901 to an accuracy of $\pm 0.6^{\circ}$C [24]. We speak about algorithmic efficiency in terms of the overall reduction in transmitter power used, as calculated using the TinyNode datasheet [18]. Note that we do not talk about algorithmic overhead as it is negligible for all algorithms, especially when compared to the sampling frequency. However, it is accounted for in our results.

Our first significant result is that while CONCH may at first appear to yield significant power savings (see Table 2), in fact its built-in temporal sampling behavior is doing almost all of the work. It is likely that SensorScope stations simply sample so frequently that the probability of two nodes simultaneously perceiving a change in

Table 2 Algorithm performance by average power savings over all previously listed datasets. Here we set the training duration to $N = 4$ hours, and the sampling interval to 6 minutes. Temporal suppression yields tremendous savings (i.e., power consumed in transmitting sensor values is 43% of the original), while CONCH yields a small additional savings.

Algorithm	Tx power reduction	Standard deviation
Naive (SensorScope)	0.0%	0.0%
Temporal suppression	57.3%	6.8%
CONCH	62.2%	11.5%

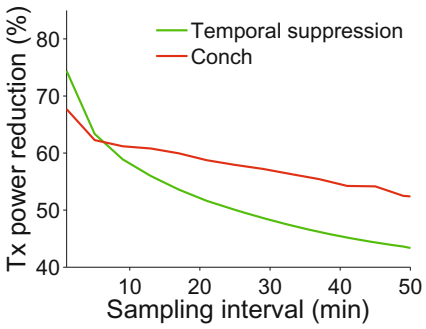


Fig. 6 As we decrease the sampling frequency, CONCH gains a greater advantage over temporal sampling. However, as subsequent measurements become less temporally correlated, the performance of both approaches drops significantly.

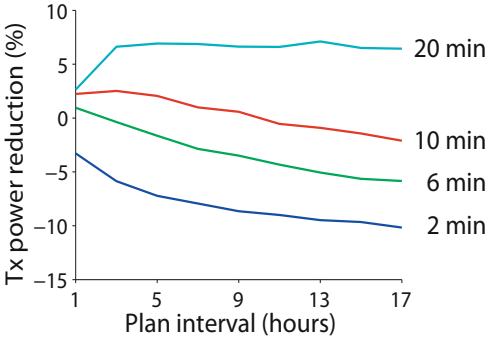


Fig. 7 CONCH's decrease in energy usage relative to temporal suppression as a function of plan duration (x-axis) and sampling interval (right). We show that a short training interval is more suited to high sampling frequencies, while longer plans only benefit very low sampling frequencies. Additionally, high sampling frequencies (2 and 6 minutes) fail to out perform temporal suppression, while lower sampling frequencies (10 and 20 minutes) see increased benefit from both CONCH and longer plan durations.

the environment is small. Indeed, as we increase the time between samples, CONCH gains a greater advantage over temporal sampling (see Figure 6).

The duration and frequency of CONCH plan optimization have a clear effect on the algorithm's ability to suppress messages. As previously described, we generate

a new plan using the previous N hours of network data every N hours. In general, we observe that long training intervals are best suited to low sampling frequencies (see Figure 7). When sampling with high frequency, the unstable edge constraints make CONCH less performant than temporal suppression. However, as before, CONCH shows an increasing advantage over temporal suppression as we lower the sampling frequency.

6 Conclusion and Future Work

In this paper we presented our approach to evaluating data collection algorithms specific to environmental monitoring. We selected an algorithm likely suited to common patterns observed in environmental fields, CONCH, adapted it for deployment on real hardware, and studied its performance using simulations over a number of datasets collected from real world sensor network deployments. While temporal suppression yields large power savings (see Table 2), CONCH fails to bring a significant amount of additional performance at high sampling frequencies. However, CONCH was originally presented as simply a modeling framework. The models used in this paper could be replaced by something more complex that may even be specifically suited to different types of sensors. For example, in the case of CONCH, it may be beneficial for nodes to maintain edge constraints based on the parameters of the autoregressive models proposed in [21], rather than directly comparing sensor measurements. We plan to explore integrating such orthogonal approaches to efficient monitoring in the future.

Many research groups release data from internal sensor network deployments to the community. While we currently use datasets from long term SensorScope deployments, UCLA makes sensor network data available via SensorBase [5], PermaSense releases data to the public on their online repository [12], and often an individual researchers' datasets are made available upon request. We also plan to incorporate classic datasets such as indoor monitoring data from the Intel/Berkeley laboratory [15]. Such data can be used to obtain a more thorough understanding of algorithm performance.

Our laboratory has a number of SensorScope stations ready for use in implementing and testing algorithms under real-world conditions (one such deployment took place on the EPFL campus in Spring 2010, see Figure 2). We are currently implementing CONCH on real stations in order carry out further campaigns that investigate energy savings as a function of the spatial distribution of the stations and anisotropy of the monitored field.

Acknowledgements. This work was partially funded by "The Swiss Experiment" of the Competence Center Environment and Sustainability of the ETH Domain (CCES), and by the Swiss National Science Foundation's Stabilization Measures Program, associated with the project "Tamperproof Monitoring Solution for Weather Risk Management" managed by the National Center of Competence in Research in Mobile Information and Communication Systems (NCCR-MICS).

References

1. Arici, T., Altunbasak, Y.: Adaptive sensing for environment monitoring using wireless sensor networks. In: IEEE Wireless Communications & Networking Conf., Atlanta, GA, USA, pp. 2347–2352 (2004)
2. Arora, A., Dutta, P., Bapat, S., Kulathumani, V., Zhang, H., Naik, V., Mittal, V., Cao, H., Demirbas, M., Gouda, M.: A line in the sand: A wireless sensor network for target detection, classification, and tracking. Computer Networks 46(5), 605–634 (2004)
3. Baron, D., Wakin, M., Duarte, M., Sarvotham, S., Baraniuk, R.: Distributed compressed sensing (2005) (Preprint)
4. Barrenetxea, G., Ingelrest, F., Schaefer, G., Vetterli, M.: The hitchhiker's guide to successful wireless sensor network deployments. In: ACM Conf. on Embedded Networked Sensor Systems, Raleigh, NC, USA, pp. 43–56 (2008)
5. Center for Embedded Networked Sensing: SensorBase data repository, `http://www.sensorbase.org/`
6. Chu, D., Deshpande, A., Hellerstein, J., Hong, W.: Approximate data collection in sensor networks using probabilistic models. In: Int. Conf. on Data Engineering, Atlanta, GA, USA, pp. 48–48 (2006)
7. Gui, C., Mohapatra, P.: Power conservation and quality of surveillance in target tracking sensor networks. In: ACM Int. Conf. on Mobile Computing and Networking, Philadelphia, PA, USA, pp. 129–143 (2004)
8. Haupt, J., Nowak, R.: Signal reconstruction from noisy random projections. IEEE Transactions on Information Theory 52(9), 4036–4048 (2006)
9. Madden, S., Franklin, M., Hellerstein, J.: TinyDB: an acquisitional query processing system for sensor networks. ACM Transactions on Database Systems 30(1), 122–173 (2005)
10. Milenkovic, A., Otto, C., Jovanov, E.: Wireless sensor networks for personal health monitoring: Issues and an implementation. Computer Communications 29(13-14), 2521–2533 (2006)
11. Otto, C., Milenkovic, A., Sanders, C., Jovanov, E.: System architecture of a wireless body area sensor network for ubiquitous health monitoring. Journal of Mobile Multimedia 1(4), 307–326 (2006)
12. PermaSense: PermaSense data frontend, `http://data.permasense.ch/`
13. Prorok, A., Cianci, C., Martinoli, A.: Towards optimally efficient field estimation with threshold-based pruning in real robotic sensor networks. In: IEEE Int. Conf. on Robotics and Automation, Anchorage, AK, USA, pp. 5453–5459 (2010)
14. Sadagopan, N., Krishnamachari, B., Helmy, A.: Active query forwarding in sensor networks. Ad Hoc Networks 3(1), 91–113 (2005)
15. Madden, S.: Intel Berkeley Lab data, `http://db.csail.mit.edu/labdata/labdata.html`
16. Sensirion: Sht75 digital humidity sensor data sheet (2010), `http://www.sensirion.com/`
17. Sensorscope Sàrl: Climaps weather monitoring system, `http://sensorscope.epfl.ch/climaps/`
18. ShockFish: TinyNode 584 user's manual (2010), `http://tinynode.com/`
19. Silberstein, A., Braynard, R., Yang, J.: Constraint chaining: On energy-efficient continuous monitoring in sensor networks. In: ACM SIGMOD Int. Conf. on Management of Data, Chicago, IL, USA, pp. 157–168 (2006)
20. Tulone, D., Madden, S.: An energy-efficient querying framework in sensor networks for detecting node similarities. In: ACM Int. Symp. on Modeling, Analysis and Simulation of Wireless and Mobile Systems, Torremolinos, Malaga, Spain, pp. 191–300 (2006)

21. Tulone, D., Madden, S.: PAQ: Time series forecasting for approximate query answering in sensor networks. In: European Conf. on Wireless Sensor Networks, Zurich, Switzerland, pp. 21–37 (2006)
22. Willett, R., Martin, A., Nowak, R.: Backcasting: adaptive sampling for sensor networks. In: ACM Information Processing in Sensor Networks, Berkeley, CA, USA, pp. 124–133 (2004)
23. Yang, X., Lim, H., Özsu, T., Tan, K.: In-network execution of monitoring queries in sensor networks. In: ACM SIGMOD Int. Conf. on Management of Data, Beijing, China, pp. 95–105 (2007)
24. Zytemp: Tn9 infrared thermometer user manual (2010), http://www.zytemp.com/

A Plume Tracking Algorithm Based on Crosswind Formations

Thomas Lochmatter, Ebru Aydın Göl, Iñaki Navarro, and Alcherio Martinoli

Abstract. We introduce a plume tracking algorithm based on robot formations. The algorithm is inherently designed for multi-robot systems, and requires at least two robots to collaborate. The robots try to keep themselves centered around the plume while moving upwind towards the source, and share their odor concentration and wind direction measurements with each other. In addition, robots know the relative poses of other team members. Systematic experiments with up to 5 real robots in a wind tunnel show that the robots achieve close-to-optimal performance in our scenario, and by far outperform previous approaches. The performance gain is attributed to the fact that robots continuously share information about the plume (odor concentration, wind direction) without spatially competing for acquiring it.

1 Introduction

With the advances in robotics and chemical sensor research in the last decade, odor sniffing robots have become an active research area. Notably the localization of odor sources would allow for very interesting robotic applications, such as search and rescue operations, safety and control operations on airports or industrial plants, and humanitarian demining [15][12][3]. Many of these applications are time-critical, i. e. odor sources should be found as fast as possible. Moreover, as the structure of plumes in the air is intermittent in both time and space [17], tracking plumes is a challenging problem.

Thomas Lochmatter · Ebru Aydın Göl · Alcherio Martinoli
Distributed Intelligent Systems and Algorithms Laboratory (DISAL),
École Polytechnique Fédérale de Lausanne (EPFL), Station 2,
1015 Lausanne, Switzerland
e-mail: alcherio.martinoli@epfl.ch

Iñaki Navarro
Universidad Politécnica de Madrid, ETSI Industriales,
c/ José Gutiérrez Abascal 2, E-28006 Madrid, Spain

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 91–102.
springerlink.com

In previous work [9], we analyzed three bio-inspired plume tracking algorithms based on upwind surge, casting, and spiraling, and carried out experiments with real robots and in simulation. Experiments with the multi-robot versions of these algorithms [8] thereby revealed that robots are competing for space even when they are communicating and collaborating, a limitation which is especially important in narrow plumes. When robots avoid each other (to prevent collisions), they often lose the plume and switch to plume reacquisition, which obviously results in performance degradation. Reasons for this are the state-machine nature of the bio-inspired algorithms on one hand, but also the fact that robots do not plan their path with respect to the positions of the other robots. The low-level controller (obstacle avoidance) interferes with the high-level controller (bio-inspired odor source localization algorithm).

In this paper, we present a novel algorithm to tackle exactly this problem. The algorithm is based on a loose robot formation which collaboratively moves through the plume towards the source. Robots thereby communicate with each other, and share all observations (wind and odor concentration) as well as their relative positions. A simple reactive control scheme keeps the robot formation centered (in crosswind direction) around the plume while the robots are moving upwind towards the source.

This algorithm is *not* an extension of some existing single-robot algorithm, but inherently designed for multi-robot system. (At least 2 robots are necessary for the algorithm to work.) *By design*, robots do not compete for space, and no low-level avoidance algorithm is necessary to prevent robots from bumping into each other[1]. In addition, robots truly and continuously collaborate: every single observation is shared, and the control algorithm uses the observations of all robots in the formation as input.

The remainder of this paper is structured as follows. In Section 2, we present existing approaches for multi-robot odor source localization. Section 3 formally introduces the crosswind formation algorithm. After depicting the experimental setup in Section 4, we show real-robot experiments with a static source (Section 5) and a moving source (Section 6). Finally, we conclude in Section 7.

2 Related Work

Most odor source localization algorithms found in the scientific literature are intended for single-robot systems. To our knowledge, there only exist four multi-robot approaches to date.

Hayes et al. applied a bio-inspired algorithm based on spiraling and upwind surge to multiple robots [2] [1], and studied the effect of a primitive broadcasting communication scheme. In particular, they studied a communication scheme called KILL in which all robots stop as soon as one perceives an above-threshold odor

[1] An obstacle avoidance algorithm may still be necessary to prevent robots from bumping into obstacles, of course.

concentration, and a scheme called ATTRACT whereby robots that do not perceive any odor join others that have found plume information. Experiments were carried out with real robots and in an embodied simulation, and the performance metric was a linear combination of time and group energy, the latter being proportional to the sum of the distances the robots traveled. The KILL strategy was found to save significant power, whereas the ATTRACT strategy did not reveal any advantage in their setup. The multi-robot experiments we carried out in simulation [8] are conceptually similar.

Second, all algorithms based on PSO [13] [5] [6] [4] are intended for use in multi-robot systems. PSO requires robots to communicate at least locally, and robots must be aware of each other's position. In the standard PSO algorithm, collaboration is however limited. The only variable they share is the (locally or globally) highest concentration, and the position where it was measured. As a result, robots will have a tendency to move towards the same local optimum if communication is global, and may bump into each other. Jatmiko et al. therefore introduced an extension called CPSO [6] in which robots share their positions and use a repulsive force to avoid collisions and make sure robots remain spread over some area. Robots however do not directly take concentration measurements of other robots into account.

Another algorithm inherently designed for multi-robot systems is the *fluxotaxis* algorithm [16]. As its name suggests, it is based on the chemical mass flux, which is a product of the chemical density and the flow velocity. Fluxotaxis is therefore a hybrid between chemotaxis and anemotaxis. As positive divergence of mass flux indicates a source, this algorithm basically climbs up the mass flux gradient. This is done with a flock of robots that share their observations (concentration and airflow) with a central controller. Fluxotaxis has been shown to perform well even in complex scenarios with obstacles.

Finally, the infotaxis algorithm has been extended to multiple robots [14]. Robots thereby share all their observations (concentrations and positions) to collaboratively infer the location of the source. All information from all robots is integrated into a single model, which is the maximum amount of information robots can share. The authors reported that in some scenarios, super-linear performance increase can be achieved by using multiple robots.

3 The Crosswind Formation Algorithm

The underlying idea of the algorithm presented in this paper is to keep some robots on the left side of the plume, and some robots on the right side of the plume. While they are going upwind, they try to stay centered in the plume, i. e. keep the (average) concentrations on the left and on the right approximately equal. The formation chosen here is a line formation in crosswind direction.

Each robot can measure the odor concentration and the wind direction (relative to its pose) at its current location, and shares this information with all members of the formation. In addition, robots can measure the relative poses of each other. A global

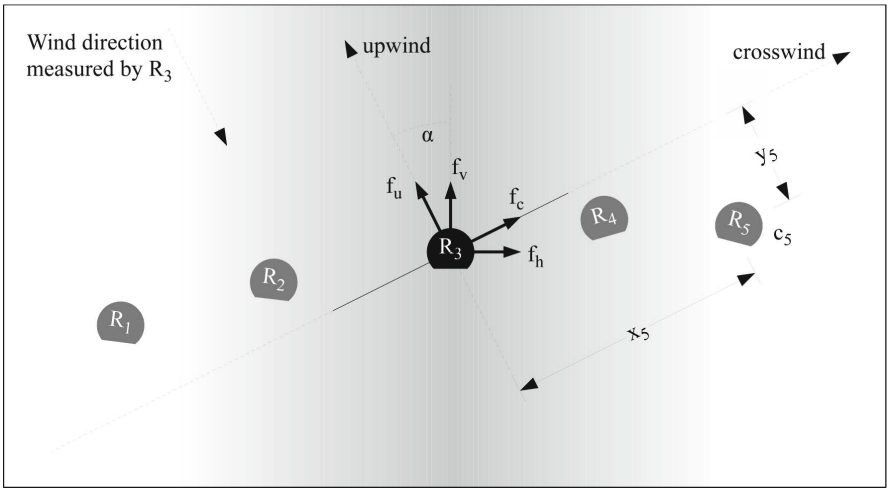


Fig. 1 Sketch of the formation algorithm from the perspective of robot 3. Calculation of the forces f_u and f_c is carried out in a reference system defined by the wind direction. The resulting force vector is then rotated into the robot's reference system. All robots carry out the exact same calculation, but from their own perspective.

reference system is not required, but the wind measurement actually provides the robots with an estimate of a global direction.

In a reference system defined by the wind direction, each robot calculates a (virtual) crosswind and a (virtual) upwind force, as depicted in Figure 1. The upwind force, f_u is

$$f_u = u + \frac{1}{N}\sum_i y_i \tag{1}$$

where N denotes the number of robots and u the constant upwind drag, a parameter of the algorithm. f_u keeps the robot aligned with the other robots, such that they all have approximately the same downwind distance from the odor source. If, for instance, one robot is behind, the y_i tend to be more positive in the coordinate system of this robot and the resulting force is stronger.

The crosswind force, f_c, is a weighted difference (with weights a and r) between an attractive and a repulsive force. Formally,

$$f_c = a f_a - r f_r \tag{2}$$

$$f_a = \frac{\sum_i x_i c_i}{\sum_i c_i} \tag{3}$$

$$f_r = \frac{1}{N}\sum_{i, i\neq \mathrm{me}} \frac{1}{x_i} \tag{4}$$

The attractive force, f_a, takes into account the odor concentrations, c_i, measured by all other robots and is responsible for keeping the formation centered in the plume.

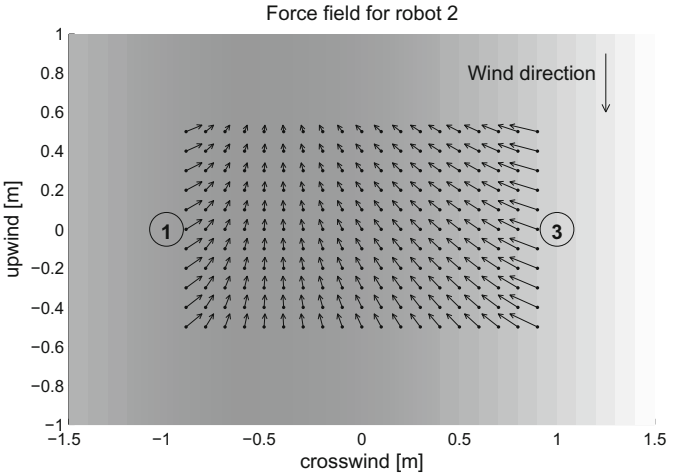


Fig. 2 Crosswind and upwind forces (scaled) for different positions of robot 2 in a formation with 3 robots. Robot 1 is currently measuring a 4 times higher concentration as compared to robot 3. The gray shading stands for the odor concentration, which attains its maximum at $x = -0.5\,\mathrm{m}$.

Robots measuring a high concentration contribute more weight, and therefore pull the other robots towards them. The repulsive force, f_r, keeps the robot at a certain distance from all other robots.

The vector (f_u, f_c) is then rotated into the reference system defined by the robots heading,

$$f_v = f_u \cos(-\alpha) - f_c \sin(-\alpha) \tag{5}$$
$$f_h = f_u \sin(-\alpha) + f_c \cos(-\alpha) \tag{6}$$

where α denotes the wind angle relative to the robots heading. The resulting vector (f_v, f_h) is finally transformed into differential drive wheel speeds as follows:

$$s_l = s(k_v f_v + k_h f_h) \tag{7}$$
$$s_r = s(k_v f_v - k_h f_h) \tag{8}$$

k_v and k_h are thereby factors to scale the forward and differential speed appropriately, and s denotes the mean forward speed.

All robots keep executing these steps continuously in a loop. The loop speed is approximately the same on all robots, but robots are not tightly synchronized. In each iteration of the loop, a robot takes one measurement with the wind direction sensor and one with the odor sensor, and broadcasts the latter to all other robots. To calculate the forces, it uses the last received odor concentration and relative position values of each robot.

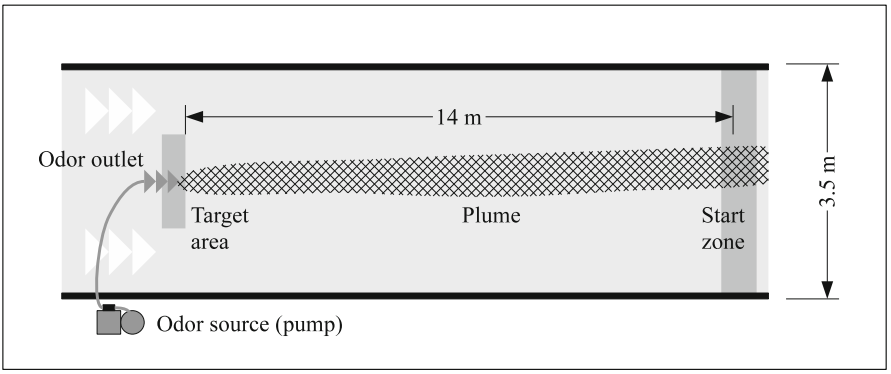


Fig. 3 Schematic drawing of the arena (not to scale) with the approximate location of the plume

4 Experimental Setup

The experiments were carried out in a 16 m long and 4 m wide wind tunnel. The setup was the same as described in [10], except that the arena inside the tunnel was enlarged to approximately 15 m by 3.5 m. The same setup was also used to analyze three bio-inspired algorithms [7]. In the following paragraphs, we briefly repeat the most important figures.

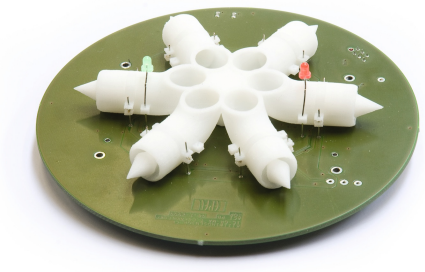
The wind field in the wind tunnel was laminar at roughly 1 m/s speed. The ethanol odor plume was therefore a straight line (see Figure 3), and the concentration peaks were slightly decreasing as the plume moved downwind. A constant amount of ethanol vapor was released by means of a pump. To reduce the turbulence created by the odor source, the pump was placed outside of the arena and connected with a tube to the source outlet. Nevertheless, the outlet created some turbulence right downwind the source, which sometimes disturbed the laminar wind flow in that area. The starting area was 14 meters downwind from the outlet, as depicted in Figure 3.

4.1 Robotic Platform

The robot used in the experiments was a Khepera III robot (K-Team SA, Switzerland) equipped with an odor sensor and a wind sensor board, as depicted in Figure 4 (a).

The odor sensor was a MiCS-5521 volatile organic compound (VOC) sensor, which has a very fast response time (≈ 0.1 s). This sensor reacts to a wide range of organic compounds in the air, with an sensitivity to ethanol comparable to that of a human nose (≈ 10 ppm). To take advantage of the sensor's low response time, air was taken in and released with a small pump.

(a)



(b)

Fig. 4 (a) The Khepera III robot with the wind sensor and the odor sensor board. (b) Picture of the wind sensor board. The two LEDs on top were used for tracking the robots with overhead cameras.

A new version of the wind sensor board with 6 thermistors was used on the robots. The board reports the wind direction at a rate of ≈ 10 Hz with a standard error of 4^{o} in our flow.

4.2 Relative Positioning

Relative positions were emulated using a camera system and sent to the robots via wireless LAN at a 10 Hz update rate. The camera system consisted of 6 overhead cameras, which recorded the whole arena. Each robot was equipped with two LEDs, and SwisTrack [11] was used to detect these markers on the recorded images. The global accuracy after calibration was about 8 cm, while the precision was around 4 cm.

5 Experiments with a Static Source

We tested the crosswind formation algorithm with the following settings:

	Algorithm	Robots	Start position	Runs
A	Crosswind Formation OSL	3	left	10
B	Crosswind Formation OSL	3	middle	10
C	Crosswind Formation OSL	3	right	10
D	Crosswind Formation OSL	5	middle	5

With the start position *left* (resp. *right*), the robots started slightly at the left (resp. right) of the plume, and only the rightmost (resp. leftmost) robot was measuring an above-baseline odor concentration. With the start position *middle*, one robot was placed in the plume center at the beginning of the experiment, while an equal

number of robots started on the left and on the right of the plume. The mean forward speed of the robots was $s = 7.1$ cm/s, and the parameters of the force model were set as follows:

	u	a	r	k_v	k_h
Experiments with 3 robots (A, B, C)	1	1	0.1225	1	1
Experiments with 5 robots (D)	1	1	0.4225	1	1

Note that no attempt was made to systematically optimize these parameters, as the main objective is to demonstrate odor source localization using formations, and not formation control itself.

Before each run, the odor concentration baseline was determined individually for each robot by taking a few measurement samples in fresh air. The sensitivities of the odor sensors were not systematically calibrated, but believed to be approximately

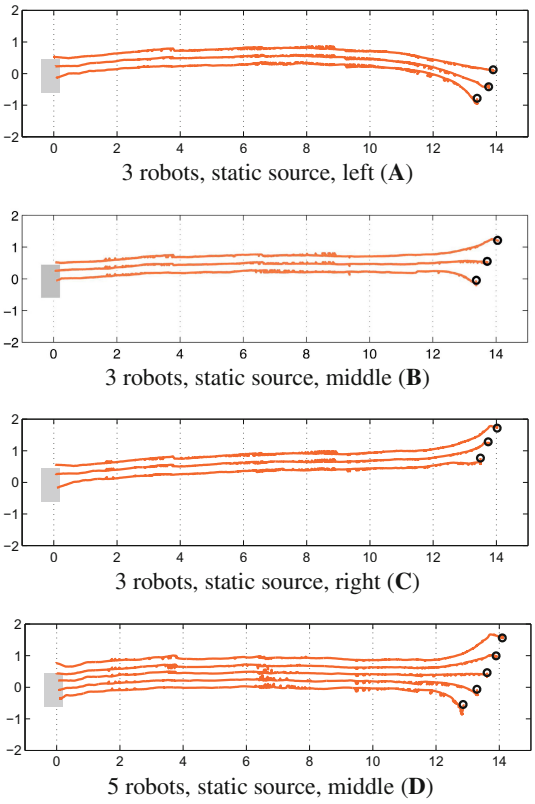


Fig. 5 Real-robot trajectories produced by crosswind formation algorithm. The gray rectangle represents the target area with the odor source while the black circles denote the starting positions of the robots. The robots go almost straight upwind towards the source, yielding very low distance overheads and high success rates.

the same. Slight sensitivity differences would result in a little drift of the formation in crosswind direction, but hardly affect the performance.

Two metrics were calculated for each run:

- The *success rate* of the team is the number of successful runs (where robots keep the formation and reach the source) divided by the total number of runs.
- The *distance overhead* of a robot is its traveled distance divided by the distance of the shortest path to the source (straight line). The distance overhead of the team is the average over the distance overheads of all team members.

Figure 5 shows one run of each setting. No matter where the robots started, the robots found the center of the plume within the first 2 m upwind distance and then continued going straight upwind. The distance overheads are therefore extremely low, as shown in Figure 6. The success rate was 100 % in all settings.

In our setup, this algorithm clearly outperforms all bio-inspired algorithms [7] [9] in terms of distance overhead and success rate. The distance overhead for most runs was below 2 %, and for some runs even below 1 %. Included in this overhead is the initial phase in which robots get into the predefined formation shape. Without this, the results would be even closer to the optimal performance.

This is not surprising: with sensors on the left and on the right of the plume, the formation obtains direct feedback about its position with respect to the plume, and can correct for that long before leaving the plume completely. A plume reacquisition

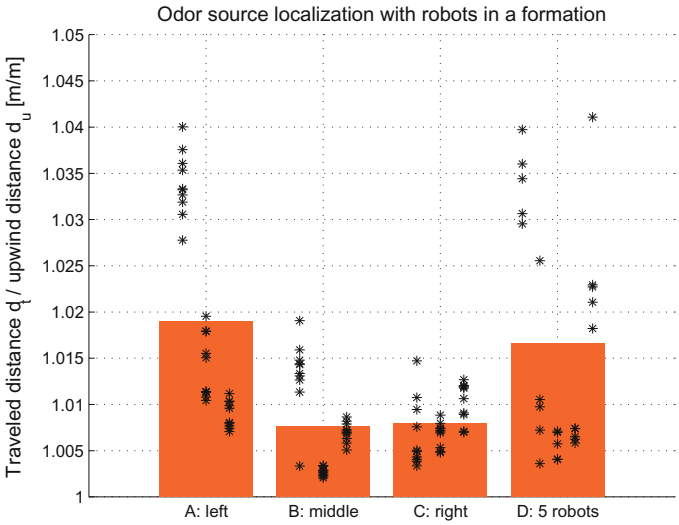


Fig. 6 Distance overheads (by starting position) of the experiments with three robots. The dots stand for the distance overhead of the robots in individual runs, and are classified by the robot's position within the formation. The bars indicate the mean distance overhead over all runs and all robots.

phase, as it is used with the bio-inspired algorithms is not necessary any more. The robots keep going upwind without ever losing the plume.

Using 5 instead of 3 robots did not improve the performance in our experiments. There is no a priori reason for which 5 robots should yield worse results. After all, 5 robots collect even more information about the plume than 3 robots, and should therefore do at least an equally good job. However, the information gain when switching from 3 to 5 robots might be tiny, and therefore irrelevant in our setup. Main reason for the performance drop here are presumable the outermost robots, which started at a suboptimal position quite far away from the plume center, and first had to move closer to the center. We believe, however, that increasing the number of robots would be advantageous in settings with a sparser plume.

6 Experiments with a Moving Source

Since this algorithm measures the odor concentration at several points at the same time, it is particularly well suited for scenarios with moving sources. With a single sensor, an algorithm is unable to tell with a single measurement in which direction the source moved. Such information can only be deduced from multiple (sequential or parallel) measurements at different locations.

Multi-robot algorithms provide just this: taking several measurements at the same time. This allows the formation to know immediately whether the source moved towards the left or towards the right. The force model takes advantage of that information in that it tries to keep the robots centered around the plume.

We carried out 5 runs with the same algorithm tracking a moving source. The source was thereby moved back and forth by 92 cm in crosswind direction at constant speed. All other parameters of the setup and the algorithm were kept the

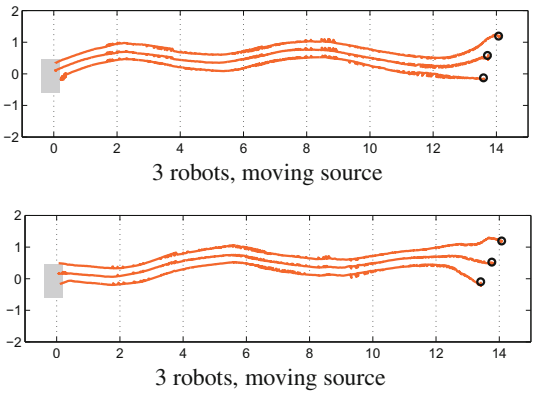


Fig. 7 Real-robot trajectories produced by the crosswind formation algorithm tracking a source moving in crosswind direction. The gray rectangle represents the target area with the odor source while the black circles denote the starting positions of the robots. The robots nicely follow the movement of the plume.

same. Two of these runs are drawn in Figure 7. While all runs were successful, it is not possible to calculate the distance overhead (as defined above) in this case. The trajectories however reveal that the algorithm works fine.

7 Conclusion

In this paper, we presented the *crosswind formation* algorithm, an odor source localization algorithm which is inherently designed for multi-robot systems. In its present form, the algorithm requires at least 2 robots to collaborate. Experiments were carried out with 3 and 5 real robots in a scenario with laminar flow and no obstacles, and showed that this algorithm achieves close-to-optimal performance in terms of distance overhead and success rate. We also demonstrated the algorithm's robustness in a scenario with a moving source.

The algorithm performs substantially better than previous attempts with bio-inspired algorithms that were extended to multi-robot algorithms. With the formation-based approach used here, a single controller takes care of avoiding collisions among robots and tracking a plume at the same time. Hence, robots do not compete for space, and do not block each other's way. Instead, robots take advantage of their relative positions, as well as their wind direction and concentration measurements, and strive for a common goal: finding the source. Nevertheless, the algorithm is simple and has — except for relative localization — low requirements.

Future work could address different formations or failing robots. In addition, the positioning requirements may be reduced to direct-neighbor-only information. The algorithm should also be tested in more complex scenarios, such as scenarios with obstacles, turbulence, more complex source motions models, or multiple sources. Also of interest is a comparison with single-robot algorithms based on multiple odor sensors, which could be regarded as multi-robot algorithms with perfect formations.

Acknowledgements. This work was supported by the National Competence Center in Research on Mobile Information and Communication Systems NCCR-MICS, a center supported by the Swiss NSF under grant number 5005-67322.

References

1. Hayes, A.T., Martinoli, A., Goodman, R.M.: Distributed odor source localization. IEEE Sensors Journal 2(3), 260–271 (2002)
2. Hayes, A.T., Martinoli, A., Goodman, R.M.: Swarm robotic odor localization: Off-line optimization and validation with real robots. Robotica 21, 427–441 (2003)
3. Ishida, H., Nakamoto, T., Moriizumi, T., Kikas, T., Janata, J.: Plume-tracking robots: A new application of chemical sensors. Biological Bulletin (200), 222–226 (2001)
4. Jatmiko, W., Sekiyama, K., Fukuda, T.: A pso-based mobile sensor network for odor source localization in dynamic environment: Theory, simulation and measurement. In: IEEE Congress on Evolutionary Computation, CEC 2006, pp. 1036–1043 (2006)

5. Jatmiko, W., Ikemoto, Y., Matsuno, T., Fukuda, T.: Distributed odor source localization in dynamic environment. In: Proceedings of the 4th IEEE Conference on Sensors (Sensors 2005), pp. 254–257 (2005)
6. Jatmiko, W., Sekiyama, K., Fukuda, T.: A pso-based mobile robot for odor source localization in dynamic advection-diffusion with obstacles environment. IEEE Computational Intelligence Magazine, 37–51 (2007)
7. Lochmatter, T., Martinoli, A.: Tracking Odor Plumes in a Laminar Wind Field with Bio-inspired Algorithms. In: Khatib, O., Kumar, V., Pappas, G.J. (eds.) Experimental Robotics. STAR, vol. 54, pp. 473–482. Springer, Heidelberg (2009)
8. Lochmatter, T., Martinoli, A.: Understanding the potential impact of multiple robots in odor source localization. In: Proceedings of the 9th International Symposium on Distributed Autonomous Robotic Systems (DARS 2008), Tsukuba, Ibaraki, Japan, vol. 8, pp. 239–250 (2008)
9. Lochmatter, T., Martinoli, A.: Theoretical analysis of three bio-inspired plume tracking algorithms. In: ICRA 2009: Proceedings of the 2009 IEEE International Conference on Robotics and Automation, Piscataway, NJ, USA, pp. 3195–3202 (2009)
10. Lochmatter, T., Raemy, X., Matthey, L., Indra, S., Martinoli, A.: A comparison of casting and spiraling algorithms for odor source localization in laminar flow. In: Proceedings of the 2008 IEEE International Conference on Robotics and Automation (ICRA 2008), pp. 1138–1143 (2008)
11. Lochmatter, T., Roduit, P., Cianci, C., Correll, N., Jacot, J., Martinoli, A.: SwisTrack: A flexible open source tracking software for multi-agent systems. In: Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2008), pp. 4004–4010. IEEE/RSJ (2008)
12. Long, M., Gage, A., Murphy, R., Valavanis, K.: Application of the distributed field robot architecture to a simulated demining task. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA 2005), pp. 3193–3200 (2005)
13. Marques, L., Nunes, U., de Almeida, A.T.: Particle swarm-based olfactory guided search. Autonomous Robots 20(3), 277–287 (2006)
14. Masson, J.-B., Bailly Bechet, M., Vergassola, M.: Chasing information to search in random environments. Journal of Physics A: Mathematical and Theoretical 42 (2009)
15. Settles, G.S.: Sniffers: Fluid-dynamic sampling for olfactory trace detection in nature and homeland security—the 2004 Freeman scholar lecture. Journal of Fluids Engineering, Transactions of the ASME 127, 189–218 (2005)
16. Spears, D.F., Thayer, D.R., Zarzhitsky, D.V.: Foundations of swarm robotic chemical plume tracing from a fluid dynamics perspective. International Journal of Intelligent Computing and Cybernetics 2(4), 745–785 (2009)
17. Vergassola, M., Villermaux, E., Shraiman, B.I.: 'Infotaxis' as a strategy for searching without gradients. Nature 445, 406–409 (2007)

Cooperative Distributed Object Tracking by Multiple Robots Based on Feature Selection

Takayuki Umeda, Kosuke Sekiyama, and Toshio Fukuda

Abstract. This paper proposes a cooperative visual object tracking by multi-robot system, where robust cognitive sharing is essential between the robots. However, one of the main issues in vision-based distributed observation is the significant differences in the background image for the interested object. According to the observing point of the robot, effective invariant feature to identify the interested object is different. In this paper, we propose an ambiguity index to select better feature algorithm for object tracking. Experimental result shows promising result for the effective multi-robot cognitive sharing.

1 Introduction

In conventional research on cooperative behaviors of distributed autonomous robots, it has been premised that robots engaged in the same mission are capable of sharing recognition of the observation target. Such information is often set up to be available by using a landmark [1] or an RFID [2] tag attached to the object. However, this is not the case in more general situations in which the cooperation is more dynamic and a priori cognitive sharing cannot be taken for granted. Hence, more challenging cognitive issues have to be considered.

To achieve cooperative work, the following basic technology elements are necessary: a self-deployment algorithm for robots to arrange themselves in a formation suitable to the task at hand [3], a recognition algorithm using vision sensors, a method of sharing common perceptions between robots [4], and a cooperative decision-making algorithm for task allocation. In this paper, we focus on the third element

Takayuki Umdeda · Kosuke Sekiyama · Toshio Fukuda
Department of Micro-Nano Systems Engineering Nagoya University,
1 Furo-cho, Chikusa-ku, Nagoya, Aichi, Japan
e-mail: {umeda,sekiyama,fukuda}@mein.nagoya-u.ac.jp

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 103–114.
springerlink.com

and propose a cooperative visual object-tracking method for a multi-robot system in which robust cognitive sharing between the robots is essential. Visual cognitive sharing in multi-robot cooperation is a particularly important issue since each robot is assumed to be in a different position, hence their views of even the same object would be significantly different. There is very little research literature which deals with this problem. In the research on vision-based object recognition, various techniques and efficient features have been proposed [5, 6, 7], but whether a particular feature is effective depends on the robot's viewpoint, which is a significant problem.

In this paper, we attempt to explore the issues of cognitive sharing by considering the visual tracking of a moving object using multi-robot cooperation. In this situation, the effectiveness of a visual feature will highly depend on the background pattern relative to the observation target, meaning that effectiveness will dynamically change as the robot moves.

In order to realize more robust cognitive sharing, we propose a framework that we call the Hierarchical Invariant Perception Model in which we regard mutual understanding as the sharing of an appropriate invariant between robots. The invariant may be shape, color, name, or relation, which each have a different level of abstractness.

In order to evaluate the appropriateness of a feature selection, we define an indicator called "ambiguity". Ambiguity is dynamically evaluated to select better features for recognition according to the current situation.

Experimental results demonstrate the effectiveness of the feature selection method for robust object tracking in a visually confusing environment.

2 Approach to the Cognitive Sharing

2.1 Hierarchical Invariants Perception Model

To achieve cognitive sharing between robots, robots need to evaluate autonomously which feature is effective within an environment. For example, suppose a robot wants to recognize a blue target but the robot is surrounded by many blue objects. In this situation, the robot cannot recognize the target using color information only, and the robot understands that the color information is not effective.

There are many invariants that robots can extract from an image, including the target's color, shape, name, function, and the geometric or semantic relation between target and other objects. We can build a model with these invariants allocated hierarchically (Fig. 1). We define ambiguity with respect to these features, and if the ambiguity is low, then a robot can evaluate the feature effectively. Therefore, a robot can evaluate its surroundings using ambiguity and if it knows the ambiguities of other robots, then it can relate their environments to its own. In this paper, we

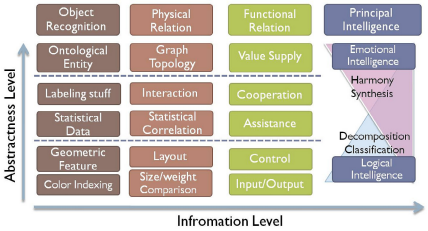


Fig. 1 Hierarchical Invariants Perception Model

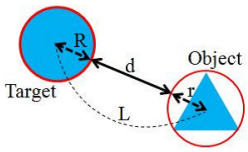


Fig. 2 Definition of the distance between objects

focus on information on color and contour. We use the mean-shift tracker for color information and pattern matching for contour information.

2.2 Ambiguity of the Color Feature

2.2.1 Definition of Ambiguity

First of all, we evaluate how objects of the same color as the target are distributed within the input image by using color histograms. Normalizing the target color histogram H_{target} and the whole image color histogram H_{image}, we calculate histogram intersection C from both histograms using eq. (1), obtaining a value for C of between 0 and 1. Here, i is the histogram bin number. The more objects of the same color are in the input image as the target, the larger the value of C.

$$C = \sum_{i} min(H_{target}(i), H_{image}(i)) \tag{1}$$

The mean-shift tracker calculates the distribution of the histogram in a window and the center of gravity of the distribution. Then, the tracker shifts the window's center of gravity to that of the calculated distribution. Because the tracker searches for objects within the window, if there are objects of the same color as the target in the window, the tracker will make a false recognition. Therefore, it is necessary to evaluate the distance between the target and objects of the same color. To evaluate distance, we use the Poisson distribution. The Poisson distribution, shown in eq. (2), gives a probability distribution based only on expected number of occurrences λ. Here, k is the number of occurrences of an event.

$$P_{(k)} = \frac{e^{-\lambda}\lambda^{k}}{k!} \tag{2}$$

First, we define the distance between the target and other objects by following the procedure below, applied to Fig. 2.

- Area of the target: A_{target}
- Radius of the target when approximated by a circle: $R = \sqrt{A_{target}/\pi}$
- Area of an object of the same color: A_{object}
- Radius of an object when approximated by a circle: $r = \sqrt{A_{object}/\pi}$
- Distance between the center of the target and the object when both are approximated by circles: L
- Distance: $d = L - r - R$

Next, we interpret the distance as k.

$$\begin{gathered} d < 5 : k = 1 \\ 5 \cdot (n-1) \leq d < 10 \cdot n : k = n \ (2 \leq n \leq 19) \\ 95 \leq d : k = 20 \end{gathered}$$

If k is small, the distance is short and there is a high probability of a false recognition. We can evaluate distance by calculating the cumulative probability in $k < 6$ showed in eq. (3).

$$P = \int_0^5 P_{(k)} dk \tag{3}$$

Finally we define the ambiguity of color feature A_{color} by calculating the weighted geometric mean of C and P because whether a false recognition occurs greatly depends on the distance between objects.

$$A_{color} = C^{\frac{n}{n+m}} \cdot P^{\frac{m}{n+m}}, \ n : m = 1 : 3 \tag{4}$$

2.2.2 Evaluation of Ambiguity

In this section, we examine the correlation between our ambiguity and the actual recognition rate. We evaluated the recognition performance of designated targets for 50 scenes in total: 5 each of ideal and real scenes for each ambiguity value range, which are 0 to 19, 20 to 39, 40 to 59, 60 to 79, and 80 or more.

Figures 3 to 5 show example scenes. The target to be recognized is a blue ball, the red circle indicates object recognized by the robot, and the gray circles indicate the approximating circle of other objects of the same color. In Figs. 3 and 4, the histogram intersection C is low, and P is low because all the distances between the objects, d, are high Thus, A_{color} is low and the robot can recognize the target. However, in Fig.5, although C is similar to in Figs. 3 and 4, P is high because there are blue objects near the target, and all distances d are low. Thus, A_{color} is high and the robot fails to recognize target.

We conducted an experiment as above for each ambiguity range; results are shown in Figs. 6 and 7. Here, the horizontal axis is ambiguity and the vertical axis is the success probability of recognition, max value is 5. From these results, it is obvious that false recognition occurs when ambiguity exceeds 60 and robots cannot recognize a target using a color feature when ambiguity exceeds 80.

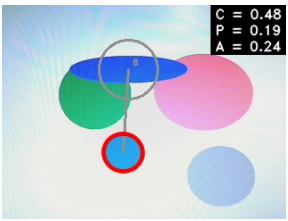


Fig. 3 Success in an ideal scene

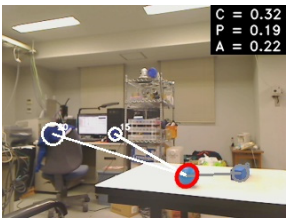


Fig. 4 Success in a real scene

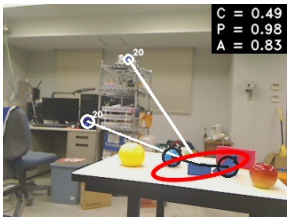


Fig. 5 Failure in a real scene

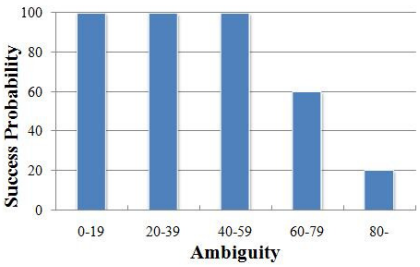


Fig. 6 Success probability in an ideal scene

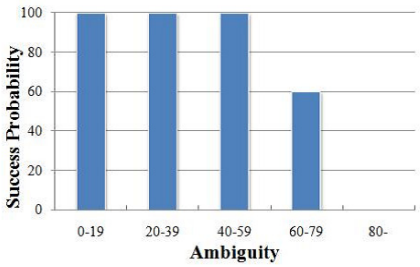


Fig. 7 Success probability in a Real Scene.

2.3 Ambiguity in the Contour Feature

2.3.1 Definition of Ambiguity

We use a pattern-matching algorithm for recognition of a target based on the contour feature. The similarity S_i of each contour is defined as in eq. (4) using Hu moment h_k [8]. Additional character t indicates the target and i is a contour.

$$\begin{aligned} m_k^t &= sign(h_k^t) \cdot \log |\, h_k^t \,| \\ m_k^i &= sign(h_k^i) \cdot \log |\, h_k^i \,| \\ S_i &= \sum_{k=1}^{7} |\, m_k^t - m_k^i \,| \end{aligned} \tag{5}$$

If $S_i = 0$, then both contours are perfectly matched, and if S_i is large, then they are not well matched. Therefore, transforming S_i to S_i' by the sigmoid function as eq. (6), $S_i' = 1$ corresponds to perfect matching, making it suitable for defining ambiguity.

$$\begin{aligned} S_i' &= \frac{1}{1 + \exp^{-\alpha(S-\beta)}} \\ \alpha &= -15\ ,\ \beta = 0.3 \end{aligned} \tag{6}$$

Calculating S_i' for all contours obtained from an input image and numbering them from 1 to N, we define ambiguity of contour $A_{contour}$ as eq. (7). The ambiguity ranges from 0 to 1.

$$A_{contour} = \frac{\sum_{i=1}^{N} S_i'}{N} \tag{7}$$

However, if there are no contours similar to those of the target, which is to say no contour has an S' over a threshold value or the robot is unable obtain any contours from the input image, then the robot cannot recognize the target at all using the contour feature and the robot sets $A_{contour} = 1$.

2.3.2 Evaluation of Ambiguity

In this section, we examine the correlation between our proposed ambiguity and the actual recognition rate in the same way as for the color feature. Figures 8 to 10 show example experiments. In Figs. 8 and 9, $A_{contour}$ is low and the robot successfully recognizes the target. However, in Fig. 10, because $A_{contour}$ is high, the robot fails to recognize the target.

We conduct experiments using the above definitions for each ambiguity range; the results are shown in Figs. 6 and 7. Although false recognitions occurs when ambiguity exceeds 60 for the color feature, in this case false recognition occurs when ambiguity exceeds 40 and the robot cannot recognize the target using the contour feature when ambiguity exceeds 60.

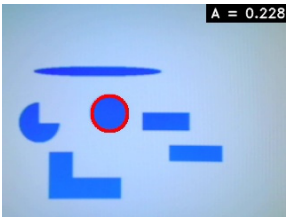


Fig. 8 Success in an ideal scene

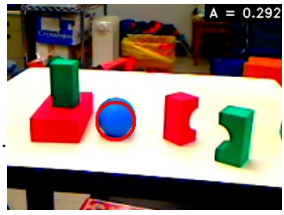


Fig. 9 Success in a real scene

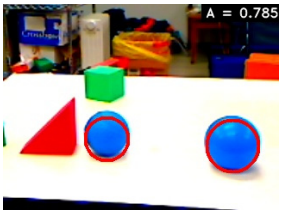


Fig. 10 Failure in a real scene

2.4 Selecting a Feature Algorithm

This section deals with selecting a feature algorithm according to the environment. In short, robots select one feature with low ambiguity by comparing ambiguities. However, from the experiments described in the above sections dealing evaluation of ambiguity, it is obvious that the ambiguity value at which a robot fails to recognize

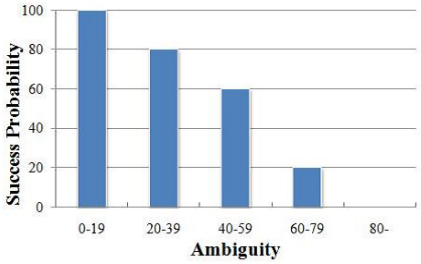


Fig. 11 Success Probability in an ideal scene

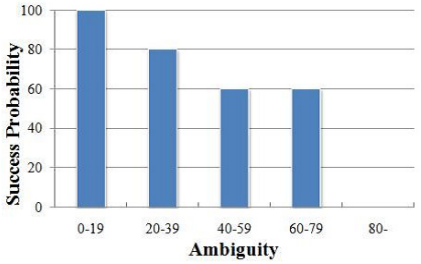


Fig. 12 Success Probability in a real scene

a target varies between features. Therefore, using a heuristic offset for $A_{contour}$, the new ambiguity is $A_{contour} \cdot 1.2$ is used.

The robots calculate ambiguity for each image frame and select a feature. Because of the robots are moving, ambiguity and the selected feature change rapidly. This can lead to recognition failure even though, for a given feature, the robot could successfully recognize the target in this environment. In addition, when the two ambiguities have almost the same value, this also causes the selected feature to change rapidly. To avoid these problems, two processes are added to feature selection, as follows.

(1) Robots save the ambiguities of the last 10 frames and calculate the means for each feature. Robots select the feature by comparing the mean at the next frame.
(2) If the difference between the two ambiguities is less than 0.1, the robots do not change the feature.

3 Experiment of Cognitive Sharing through Object Tracking

3.1 Cognitive Sharing on Robots Communication

3.1.1 Sharing Target Infomation

This section discusses cognitive sharing through object tracking based on ambiguity.

When a robot cannot maintain the tracking of a mobile target, it needs to make a tracking request to peripheral robots. Because the environment surrounding each robot is different, it is not obvious whether the effective feature for one robot is appropriate for another robot. Therefore, robots need to modify differences in the effective feature caused different environments by communicating. For example, when robot A tracking a target makes a tracking request to robot B, the robots modify their differences as follows.

(1) Robot A sends a packet including information on which feature has low ambiguity. The packet on a color/contour feature is named a Color/Contour Packet.

(2) Robot B start to track the target on the basis of the received feature. If its ambiguity is low, the robot maintains tracking. But if its ambiguity is high, robot B may fail to recognize the target and request another feature from Robot A by sending a Request Packet.
(3) When robot A receives a Request Packet from robot B, robot A sends a packet not sent at step (1).
(4) When robot B gets the new feature, robot B restarts tracking on the basis of both features. If both ambiguities are high, robot B maintains tracking, but without confidence in successfully recognizing the target.

Additionally, when robots receive any packet, they send an Answer Packet. After receiving an Answer Packet, the original sending robot stops sending its packet.

3.1.2 Identifying a Target between Robots

Identifying a target when multiple robots track a target cooperatively is more complicated. If only one robot tracks a target, whether the robot has successfully recognized the target or not, it must keep tracking. However, when multiple robots try to recognize a target at the same time, there are three cases: both robots successfully recognize the target, both robots fail to recognize the target, and exactly one robot successfully recognizes the target. This problem can be solved by using the context information shown in Fig. 1. In this paper, we approach this problem by using the position relation between the target and a landmark as the basic context.

When a robot recognizes the target, it sends a Recognition Packet, which includes three pieces of information: the type of selected feature, its ambiguity, and the position relation between the target and a landmark. The position relation is classified into whether the landmark is near the target or on which side of the landmark the target can be found.

For example, robot A sends a Recognition Packet and also receives one from other robot. Robots identify the target as follows.

CASE1. There is a contradiction in the position information for the target relative to the landmark.

(1) If robot A's ambiguity is less than the other robot's, robot A maintains tracking.
(2) If robot A's ambiguity is higher than the other robot's, robot A starts to search its surroundings.
(3) If the two robot's ambiguities are almost the same, the robots cannot evaluate which robot has successfully recognized the target and both robots maintain tracking.

CASE2. There is no contradiction in the position information for the target relative to the landmark.

(1) If both robot's ambiguities are low, the robots conclude that they are recognizing the same object and maintain tracking.
(2) If one or both robot's ambiguity is high, the robots conclude they may be recognizing same object and maintain tracking.

3.2 Experimental Conditions

The experimental environment is shown in Fig. 13. The target is the blue ball placed on the mobile robot operated manually, and the landmark is the red block. Because of the dummy, which is the same color as target, the ambiguity of color is high on the left side of robots B and C's area.

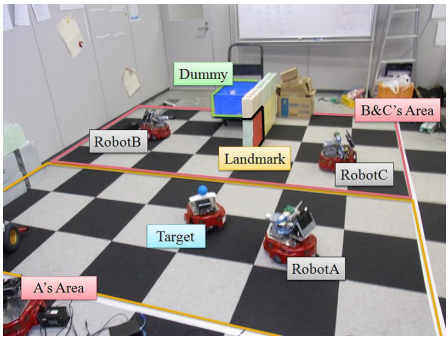


Fig. 13 Experimental Environment

The experimental procedure is as follows, where step (1) is carried out manually but the robots run autonomously starting from step (2).

(1) Robot A gets color and contour features of the target by selecting the target from an input image.
(2) Robot A tracks target as long as the target is in robot A's area.
(3) When the target leaves robot A's area, robot A sends a tracking request to robots B and C.
(4) Robots share target information and identify the target by the methods shown in previous sections.

3.3 Experimental Results

Scenes of the Experiment are shown in Figs. 14 to 16. In addition, ambiguity and the selected feature as a function of frame are shown in Figs. 17 to 19, and the communication log between robots is shown in Fig. 20.

In Fig.14, the target leaves robot A's area, and thus robot A sends a tracking request to the other robots by sending a Color Packet. In Fig. 15, although robot B tries to track the target, it requests another feature from robot A because A_{color} is high. However, from robot B's viewpoint, there is no contour similar to that of the target; therefore, robot B starts to track using the color feature and reports that the landmark is to the right of the target. In contrast, robot C can track the target by using the color feature because its ambiguity is low; therefore, robot C does not request another feature and reports that the landmark is to the right of the target. Robots B and C both send a Recognition Packet and try to identify the target, but there is a contradiction in the position information for the target relative to the landmark. Thus, robot B, which has high ambiguity, stops tracking and starts searching its surroundings. In Fig.16, robot B recognizes an object again and the landmark is to the left of the object. Because this time there is no contradiction and both robot's ambiguities are low, the robots report that they have identified the target.

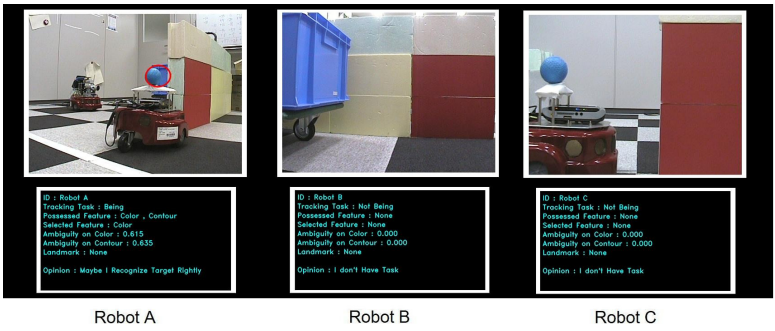


Fig. 14 Experimentarl Result: Frame = 218. The target leaves robot A's area, and robot A sends a tracking request to the other robots.

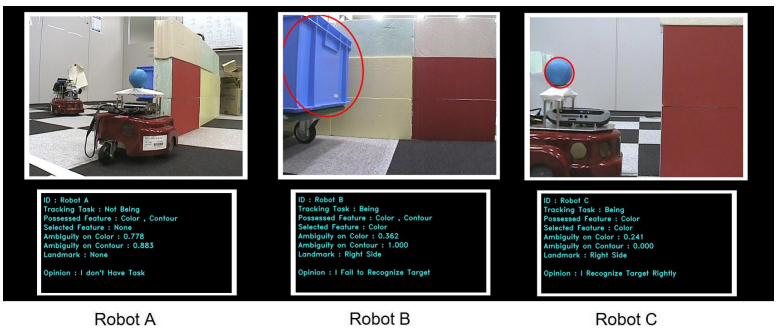


Fig. 15 Frame = 269. Robots B and C try to identify the target. Robot B, which has high ambiguity, fails to recognize the target, and robot C successfully recognizes the target.

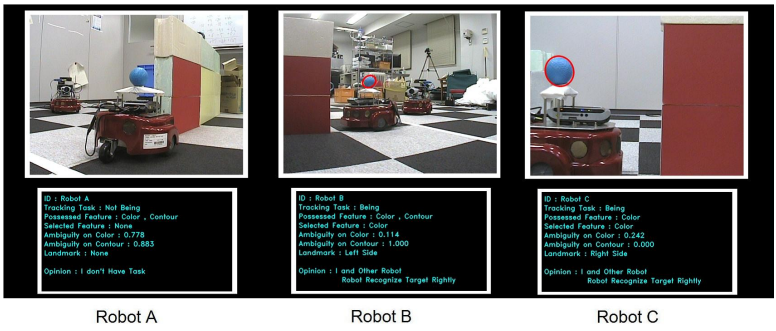


Fig. 16 Frame = 308. Robot B searches its surroundings and successfully recognizes the target again. This time, there is no contradiction, and thus both robots successfully recognize the target.

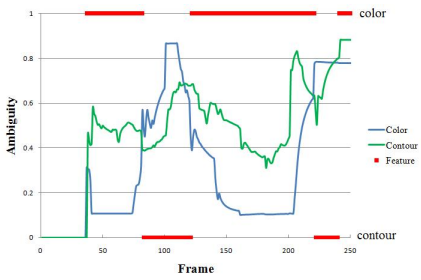


Fig. 17 Transitions: robot A

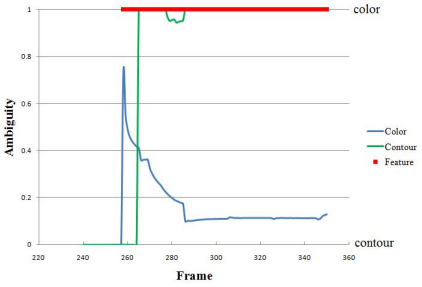


Fig. 18 Transitions: robot B

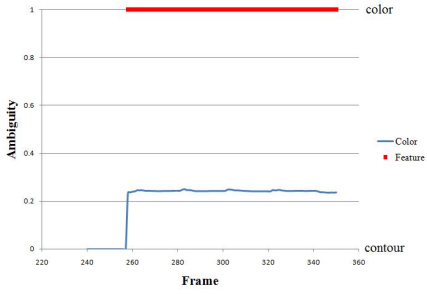


Fig. 19 Transitions: robot C

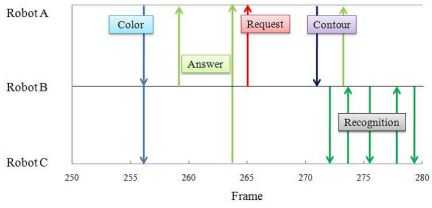


Fig. 20 Communication log

4 Conclusion

In this paper, we proposed ambiguity for evaluating effectiveness of each feature in various environments to enable cognitive sharing and identify a target, which

are the main problems in multi-robot cooperative tasks. We developed an algorithm selecting the best feature according to the environment, based on the ambiguity. Furthermore, through the experiment of cooperative distributed object tracking by multiple robots, we show that communication between robots based on ambiguity corrected differences in the effective feature caused by differences in environment and position relation between target and landmark, which allows the target to be identified.

Acknowledgements. This paper was partially supported by MEXT (KAKENHI 22500174) and Toyota Physical and Chemical Research Institute.

References

1. Zhao, L., Li, R., Zang, T., Sun, L.-N., Fan, X.: A Method of Landmark Visual Tracking for Mobile Robot. In: Xiong, C.-H., Liu, H., Huang, Y., Xiong, Y.L. (eds.) ICIRA 2008, Part I. LNCS (LNAI), vol. 5314, pp. 901–910. Springer, Heidelberg (2008)
2. Tan, K.G., Wasif, A.R., Tan, C.P.: Objects Tracking Utilizing Square Grid Rfid Reader Antenna Network. Journal of Electromagnetic Waves and Applications 22(12), 27–38 (2008)
3. Takahashi, J., Sekiyama, K., Fukuda, T.: Cooperative Object Tracking with Mobile Robotic Sensor Network. Distributed Autonomous Robotic Systems 8, 51–62 (2008)
4. Gohring, D., Homann, J.: Multi Robot Object Tracking and Self Localization Using Visual Percept Relations. In: Proceedings of IEEE/RSJ International Conference of Intelligent Robots and Systems, pp. 31–36 (2006)
5. Tussaubatafan, P., Suter, D.: Object Tracking in Image Sequences Using Point Feature. Pattern Recognittion 38(1), 105–113 (2005)
6. Yilmaz, A.: Object Tracking by Asymmetric kernel Mean Shift with Automatic Scale and Orientation Selection. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–6 (2007)
7. Okuma, K., Taleghani, A., de Freitas, N., Little, J.J., Lowe, D.G.: A Boosted Particle Filter: Multitarget Detection and Tracking. In: Pajdla, T., Matas, J(G.) (eds.) ECCV 2004, Part I. LNCS, vol. 3021, pp. 28–39. Springer, Heidelberg (2004)
8. Hu, M.: Visual pattern recognition by moment invariants. CIRE Transaction on Information Theory, 179–187 (1962)
9. Comanisiu, D., Ramesh, V., Meer, P.: Real-time tracking og non-rigid objects using mean shift. In: IEEE Computer Vision and Pattern Recognision, vol. 2, pp. 142–149 (2000)
10. Comanisiu, D., Meer, P.: Mean shift analysis and applications. In: IEEE International Conference on Computer Vision, pp.1197-1203 (1999)
11. Fukunaga, K., Hostetler, L.: The estmation of the gradient of a density function, with applications in pattern recognision. IEEE Trans. Inf. Theory 21(1), 32–40 (1975)

Pancakes: A Software Framework for Distributed Robot and Sensor Network Applications

Patrick Martin, Jean-Pierre de la Croix, and Magnus Egerstedt

Abstract. The development of control applications for multi-agent robot and sensor networks is complicated by the heterogeneous nature of the systems involved, as well as their physical capabilities (or limitations). We propose a software framework that unifies these networked systems, thus facilitating the development of multi-agent control across multiple platforms and application domains. This framework addresses the need for these systems to dynamically adjust their actuating, sensing, and networking capabilities based on physical constraints, such as power levels. Furthermore, it allows for sensing and control algorithms to migrate to different platforms, which gives multi-agent control application designers the ability to adjust sensing and control as the network evolves. This paper describes the design and implementation of our software system and demonstrates its successful application on robots and sensor nodes, which dynamically modify their operational components.

1 Introduction

The increasing use of wireless sensor networks in distributed control applications, such as unmanned surveillance or building automation, results in the deployment of heterogenous, mobile computing platforms into new environments. These systems are usually connected with wired or wireless interfaces, such as Ethernet, Wi-Fi, or ZigBee, to enable the sharing of local information among the devices comprising the network. One important development for utilizing these distributed control networks

Patrick Martin
York College of Pennsylvania, York, PA 17403 USA
e-mail: pjmartin@ycp.edu

Jean-Pierre de la Croix · Magnus Egerstedt
School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, GA 30312 USA
e-mail: jdelacroix@gatech.edu, magnus@ece.gatech.edu

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 115–127.
springerlink.com

is the incorporation of mobile robots, as noted by LaMarca *et al.* [14] and Saffioti *et al.* [17]. Allowing robots to interact with sensor networks provides new functionality in military, industrial, and consumer applications. In [14], the authors deployed a robot to maintain an office garden and its wireless sensors. The authors developed a software framework that couples their robot with the sensor nodes embedded into the office garden. The robot successfully maintained energy resources of the sensors as well as detected failures. Furthermore, the authors of [17] developed a software framework that connects robots to distributed sensor networks so that mobile robots may assist humans in a residential environment.

To make multi-agent robotics applications, such as the prior examples, work across different types of robots and wireless sensors, developers need software frameworks that help manage the complexity introduced by the heterogeneity of computational platforms and communication interfaces. Furthermore, the robotic and sensing devices on the network need to respond dynamically to physical changes (i.e. battery power). Providing the ability to dynamically adjust the framework at runtime opens up the possibility of extending operational lifetime, as well as adapting the system to reflect changes in the environment.

In this paper, we propose and demonstrate a software framework that unifies robotic and sensor networks in a seamless way, much like the prior efforts in [14, 17]. This framework, called *Pancakes*, gives developers several key features that facilitate the design of multi-agent control applications. First, Pancakes *abstracts sensing, actuation, and networking capabilities* such that high-level controllers can be implemented without worrying about low-level hardware management. Furthermore, this framework provides a structured way to *dynamically adjust* the runtime behavior of the sensor and robotic platforms according changes on the local system, as well as the operational environment. Complementary to this dynamic adjustment feature, Pancakes allows for the *migration of executable components* (i.e. sensing and control algorithms) from one platform to another. This software framework was inspired by the current literature in distributed and software control middleware, e.g. [7, 10, 18], robotics control software, e.g. [9, 11, 13, 16, 6], and actor-oriented design principles, e.g. [8, 12, 15].

In [7], Abdelzaher *et al.* developed a software framework that enabled the dynamic adjustment of a web server using feedback control. Their middleware exposed software "knobs" that could be adjusted to get better quality of service. Moving beyond this idea of modifying parameters of software components is the idea of reflective middleware [18]. This work describes a system where the pieces of the middleware dynamically adapt their capabilities as changes occur within the software. The work in [10] proposed a larger distributed embedded system framework that enables the development of software across many different types of computing platforms, from embedded controllers to desktop systems. In a similar manner, Pancakes gives robot and sensor network application designers the ability to dynamically change how their system operates at runtime. Furthermore, it allows for the migration of system components across deployed platforms.

The work in robotics software architectures made the control of heterogeneous systems easier by abstracting the sensors and actuators. For example, [11] created

a common interface to the sensors and actuators of the robots so that users could write control software that works on different types of robots without having to know every detail of the robot's implementation. The authors of [9] took this idea a step further by separating the capabilities of a robot into discrete, re-usable components that can be assembled into a larger robot control application. Additionally, the work in [13] applied multi-agent software design to create a platform for developing distributed robotics applications. The newer software package, ROS [6], provides an operating system-like framework in an attempt to standardize robotics software development for many robotics platforms. Pancakes provides the same sensor, actuator, and network abstraction that are commonplace in recent robotics software frameworks, which allow it to unify distributed robots with sensor networks.

To the best of the authors' knowledge, the combination of dynamic adjustment and migration of system components with hardware abstraction is a novel contribution to the distributed robotics community. These features allow us to create dynamic applications that leverage the capabilities of these heterogeneous robot and sensor networks. The structure of this paper is as follows: in Section 2 we provide a high-level description of how Pancakes works using an example application. Following this overview, we discuss the architecture design and implementation in Section 3. In Section 4 we deploy the Pancakes architecture onto mobile robots that must encircle a region monitored by a sensor node. We conclude with some final remarks in Section 5.

2 Pancakes Overview

Each system deployed with Pancakes is treated at its highest level as a software agent. However, Pancakes is not a general purpose agent-based software framework, such as CybelePro [2] or JADE [4]. Instead, Pancakes focuses on providing an infrastructure for the distributed control of robots and sensor networks. The result is a Java-based system that can be deployed on embedded computers, such as ARM-based platforms, as well as full desktop environments.

Pancakes provides the necessary hardware and network abstractions that have become a common practice in current robotics software frameworks. These abstractions let users utilize the system devices that interact with the environment, such as actuators and sensors. Also, the networking services let each agent share local information by passing messages over the network interface without having to micromanage the low-level communication protocols.

Internally, each Pancakes agent is composed of the Pancakes *kernel* and a collection of actor-like software components that communicate with each other using input and output channels provided by the Pancakes kernel. The two types of components in Pancakes are *tasks* and *services*. Tasks carry out a particular function for the Pancakes agent, such as reading sensor data or performing agent discovery. They publish their results onto their output channels for other tasks or services to use.

Services spawn and manage tasks that the agent requires for execution. Services submit periodic tasks to the kernel's *scheduler* for execution. Additionally, event-driven tasks are configured to listen to their input channels for new messages, use these messages to carry out their computation, and eventually send results to an output channel. The services also enable the dynamic reconfiguration of the middleware by starting new tasks, adjusting task schedules, stopping current tasks, migrating tasks across platforms, or shutting down an entire service. Dynamic reconfiguration is especially important when we construct power- and communication-aware applications.

Consider the mobile robots and wireless motion sensor shown in Figure 1, which are deployed in a building for security monitoring. The two mobile robots, Agents 1 and 2, need to communicate between themselves and the sensor node in order to share information necessary for completing the desired monitoring mission. Some important questions an application designer needs to consider are: what happens when a mobile robot is low on power? and how can robot tasks be transferred from one agent to another?

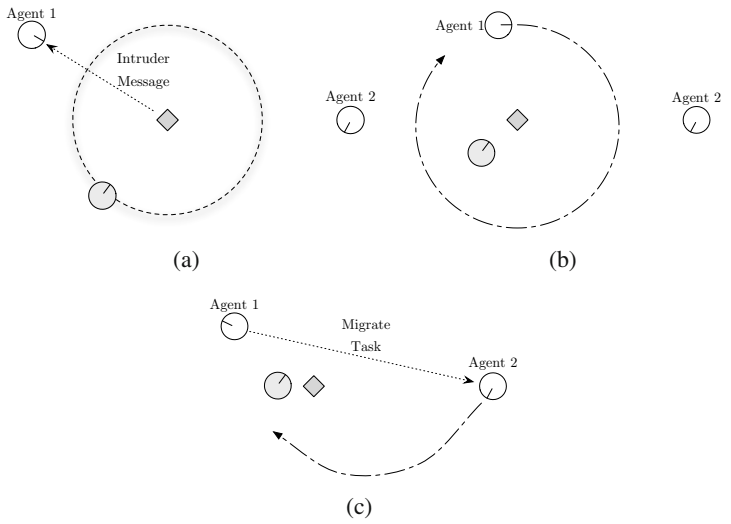


Fig. 1 An illustrative example of our desired control application for mobile robots (white circles) working with a sensor node (grey diamond) to isolate an intruding agent (grey circle)

Using Pancakes, we can create a control application that allows us to address these questions in the following way. When the sensor node (grey diamond) detects motion from an intruder, it sends a message to Agent 1. This robot initiates a task that encircles the region where the intruder was detected, as illustrated in Figure 1(b). While executing this task, Agent 1 can use Pancakes to monitor its power consumption and actively adjust its speed or communication rates to conserve power. If Agent 1 consumes too much energy, it needs to ensure that the region is still

monitored by *migrating* its currently running task to Agent 2 (Figure 1(c)). This capability would effectively lengthen the operational time of the network by letting Agent 2 wait until absolutely necessary before executing a task. The details on how Pancakes is designed to facilitate the implementation of this application is the subject of the next section.

3 The Pancakes Architecture

In this section, we describe the architecture of the Pancakes software framework and how they work together to facilitate the development of multi-agent robotics applications. Each Pancakes agent is composed of a collection of executable components, or *tasks*, which are the "workhorses" of Pancakes. These tasks are associated with a *service* that maintains a collection of related tasks. Since we adopt the actor-oriented model of programming [8, 12, 15], tasks and services communicate with each other through a collection of channels, the *information stream*. Combining these pieces with a scheduler allows for the construction of parallel and dynamic control applications for multi-agent systems.

3.1 Information Stream

The information stream sets up the communication channels that services and tasks use to publish new information or subscribe to receive information from other Pancakes components. This stream contains five core channels: `system`, `sysctrl`, `ctrl`, `network`, and `log`. Additionally, services can create specialized channels at runtime that are used to pass service specific information among tasks within the service.

The `system` channel provides a channel for services and tasks to publish system information to user-made and other system tasks. For example, a mobile robot's sonar sensor task would publish its most recent data points to the `system` channel, which is subscribed to by a control task. The `sysctrl` channel serves as a control messaging channel among the services and tasks. Messages sent over this channel facilitate the dynamic rescheduling, shut-down, or migration of tasks. To issue control commands to actuators, tasks send messages over the `ctrl` channel. A task or service that requires network communication publishes its network messages to the `network` channel. Finally, the `log` channel allows any Pancakes component to perform error, debug, or data logging, which helps in the post-run analysis and debugging of complex distributed applications.

It is important to note that the existence of the five core channels stems from our preference to semantically organize the flow of information within Pancakes. Information is communicated with Packets that indicate the destination and type of information. This approach is analogous to networking over TCP/IP, where packets can be transmitted to a destination independent of the information contained in

each packet. When using TCP/IP, one can choose to transmit `http` over port 80 or `ssh` over port 22 in order to organize such information flow over the network. Similarly, the information stream in Pancakes can be organized by adding or removing channels as needed.

3.2 Tasks

Task components are the main "actors" in Pancakes: they produce and consume information in order to affect a change in the deployed system. Tasks can execute in time-driven, event-driven, or a combination of both modes depending on the desired functionality set by the designer. At startup, a time-driven task is submitted to the scheduler and is executed at its specified frequency. The event-driven tasks wait for a message to arrive on one of its incoming channels. Since tasks communicate via the Pancakes stream channels, there is no need to synchronize on shared variables. Instead, the data necessary for execution is transmitted through the channels and delivered to subscribing Pancakes components.

Tasks are a natural way to abstract how different pieces in the system should interact. For instance, as shown in Figure 2, a robot can have several sensing tasks, such as sonar, IR, or local pose, and a control task that takes the output from these sensors and computes a control input for the actuation system. Furthermore, there can be a supervisor task that executes in parallel, monitors the output of all of the other tasks, and makes higher level decisions. In this example, the Supervisor Task examines the sensor input *and* the output from the control task in order to adjust sampling rates of different sensor managed by the Device Service. The advantage of this approach is that the application designer can focus on developing the input/output behavior of each individual task, rather than deal with complicated thread management.

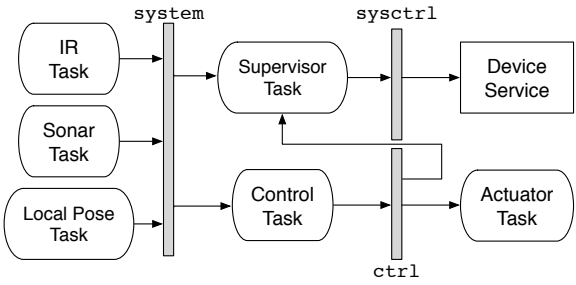


Fig. 2 In this example Pancakes application, a robot is deployed with sonar, IR, and local pose sensors. This sensor information is communicated to the system via the `system` channel, which is subscribed to by the Control Task and Supervisor Task. The Control Task computes a command for the actuators and sends it over the `ctrl` channel. Additionally, the Supervisor Task sends commands over `sysctrl` to the Device Service to adjust runtime components (e.g. the sampling rate of the IR task).

3.3 Services

The duties of services are 1) to maintain a registry of its tasks that are currently running, 2) to manage the startup, shutdown, or migration of tasks, and 3) to manage the shutdown or restart of the service itself. Pancakes has four default services that are loaded at startup: the Device Service, the Network Service, the Log Service, and the Client Service. Furthermore, developers can create new services that can supply additional functionality for their system.

The Device Service creates the system device tasks, which are the hardware abstractions for sensors and actuators, and schedules any that require timed execution. The Network Service enables communication with other Pancakes agents on the network. This service creates a network client task that listens to the `network` channel for any outgoing network messages and transmits them to the intended target. The Log Service listens to the `log` channel and displays error and debug messages to the console; additionally, it can record messages to a file for later analysis. Finally, the Client Service spawns user tasks and channels for inter-task communication. Users implement tasks to carry out communication and control algorithms, which are then loaded into the Client Service at system startup.

3.4 Dynamic Adjustment and Migration of Components

One of the key features of Pancakes is the ability to adjust services and tasks as an application executes on a deployed system. This feature lets the application adjust the capabilities within the architecture according to dynamic effects from software (i.e. logic statements, software controllers) or the physical environment (i.e. power consumption, sensing data). For example, a task can request that network discovery be slowed down to reduce the number of network transmissions; therefore, it can reduce the rate of power consumption of the application. Also, a more drastic power savings could be achieved by requesting the Network Service to shut down temporarily.

We enable this feature by establishing a messaging protocol for tasks and services to request a change the runtime behavior of other tasks and services. The currently supported control operations are *stop*, *restart*, *start*, or *reschedule*. For a task or service to initiate one of these controls, it must send a control message over the `sysctrl` channel within Pancakes. All services subscribe to this channel and inspect the message to determine if it has to change its behavior or that of one of its tasks. Once the message is received at the target service, the service calls on the scheduler to stop, start, or reschedule the task. Additionally, if the service is requested to stop or restart, it shuts down all of its currently running tasks and requests the scheduler to stop or restart itself.

A complementary feature to dynamic adjustment is the ability to migrate tasks among deployed Pancakes systems. As illustrated in the example of Section 2, mobile robots and sensor networks can use this capability to achieve a longer mission lifetime. Task migration is managed by the Task Migrator task in the Network

Service, since it must communicate with neighboring agents to find a suitable candidate for migration. The migration protocol involves sending the task and its dependencies for execution (i.e. required sensors and/or actuators) to all neighbors of the current agent. Once candidate agents are found, the migrating agent chooses the one that has the lowest execution "cost." In its current implementation, our cost metric is based on the system load of the candidate agent, for example, the number of tasks running on the deployed system.

3.5 Implementation

To enable the concurrent operation of multiple hardware and networking devices, Pancakes makes use of the actor-oriented programming model as described in [8, 12, 15]. This model creates software components that focus on concurrency and communication rather than interface methods, such as remote method invocation: a common technique in object oriented software [15]. By using an actor-oriented approach, Pancakes avoids the common issues of thread blocking in concurrent applications, since information is shared via message passing among the components, rather than through direct function calls.

We implemented Pancakes in Java to ensure its operation on several types of computational platforms and operating systems. In particular, the robots and sensors nodes in our lab use low-power ARM processors and an embedded Linux/GNU OS. We use the open source virtual machine JamVM [3], which can be compiled for several processor architectures. Another reason we use Java as our implementation language is the existence of robust Java libraries that enable concurrency and message passing. The Java SE 6 standard library has new concurrency tools that efficiently handle multiple threads using specialized thread pools. Complementary to this library, we make use of the Jetlang [5] library, which provides messaging services for multi-threaded applications.

Our choice to implement Pancakes in Java currently restricts us to JVM capable platforms, such as ARM- and PC-compatible systems. While most robotic systems are capable of supporting a JVM, the smaller processors of sensors are unlikely to support a JVM as well. However, since sensor networks commonly interface to a larger computational unit, we can incorporate these smaller sensor nodes as virtual (networked) devices into a larger unit, like the BUG [1], which can communicate via a low-power radio, such as ZigBee.

4 Experimental Results

In this section, we use Pancakes to implement the example application described in Section 2. The platforms used in our experiment are shown in Figure 3. Pancakes is deployed on two Khepera III robots, which perform the target encirclement behavior with a BUG sensor node as described in the example scenario of Section 2. To

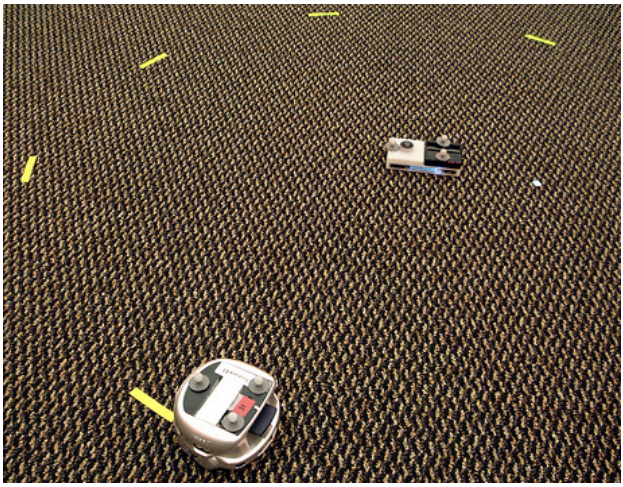


Fig. 3 This figure shows the hardware devices used in our experiment. The figure shows one of the two K-Team robots circling a BUGLabs BUGBase, which is our sensor node to detect intruders. These systems were provided with indoor localization data from the motion capture systems shown.

determine the local pose of each system we use a Vicon motion capture system. Since this local pose data is produced off-board by the motion capture system, it is a "virtual" local sensor on each robot. The Vicon system tracks the reflective points on each robot and transmits the local pose data to each robot, where the data is received and handled by a Local Pose task in Pancakes.

The following experiment uses two robots to perform surveillance, and a BUG sensor node for motion monitoring. The robots have two tasks available: 1) a `ScanTarget` task, which implements the boundary tracking algorithm of [19] and 2) a `GoHome` task, which drives the robot back to its home station. The BUG sensor has a task, `MotionDetection`, that monitors its motion detection sensor and transmits its location to its neighbors when motion is detected. Initially, the BUG is set near the center of the monitored area and both robots are initialized. Agent 1 starts with its `ScanTarget` task initialized and agent 2 is held idle for reserve.

Figure 4 shows how the systems execute during the initial phase of the experiment. Agent 1, the $\triangle$ symbol, starts from its initial position in the top right of the area. It converges to a circle around the BUG sensor, denoted by the $\square$ near the origin. Agent 2, denoted by the $\circ$, waits in the bottom right of the region to be assigned a task. Once agent 1's battery level drops below a particular threshold, it begins its *task migration*, such that agent 2 can take over the `ScanTarget` task.

Agent 1 sends a migration message to all of its neighbors, which includes the task itself and a list of dependencies the task needs to execute. For now, these

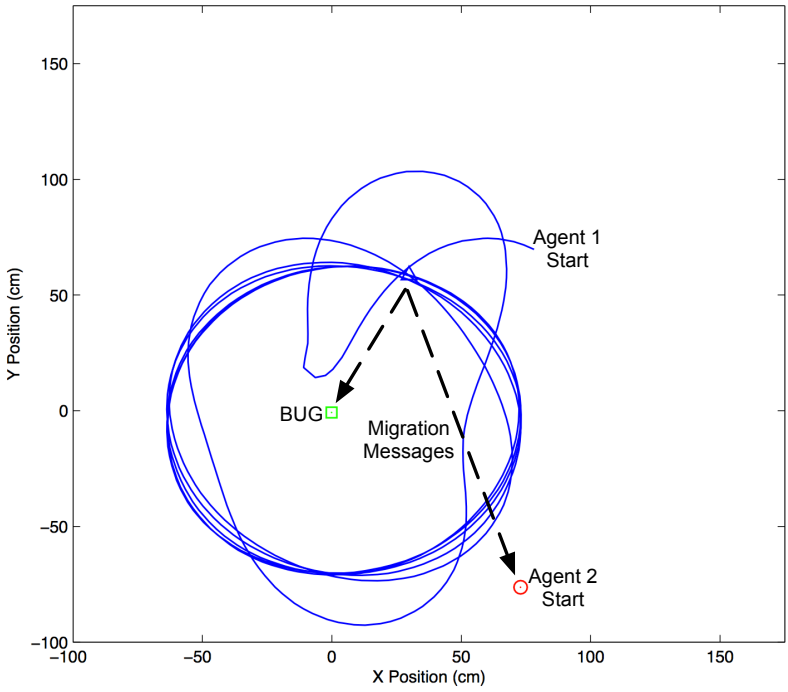


Fig. 4 This figure shows the agent 1, denoted by △ with trajectory, encircling the BUG sensor (□ near the origin). Agent 2, denoted by the ○, is idle to the bottom right.

dependencies are the types of devices the platform supports (i.e. sonar, local pose, motors). When a neighbor receives the message, it checks for dependencies and, if compatible, returns a positive reply with a "cost" value. This cost is currently calculated by counting the number of active tasks running on the platform; however, the framework is flexible enough that a more complicated cost could be calculated from power levels, communication rates, or other important properties of the platform. Agent 1 inspects all of its valid replies, chooses the agent with the lowest cost, and migrates the task to that agent.

Figure 5 shows the trajectories of the systems after migration has taken place. Agent 1 has started its `GoHome` task and agent 2 is encircling the BUG sensor node using the migrated `ScanTarget` task. Agent 2 continues to circle the sensor node, as shown in Figure 6, and agent 1 has returned to its home position for the duration of the mission. This experiment shows how the Pancakes framework enabled the creation of a dynamic control application for a small team comprised of mobile robots and a sensor node.

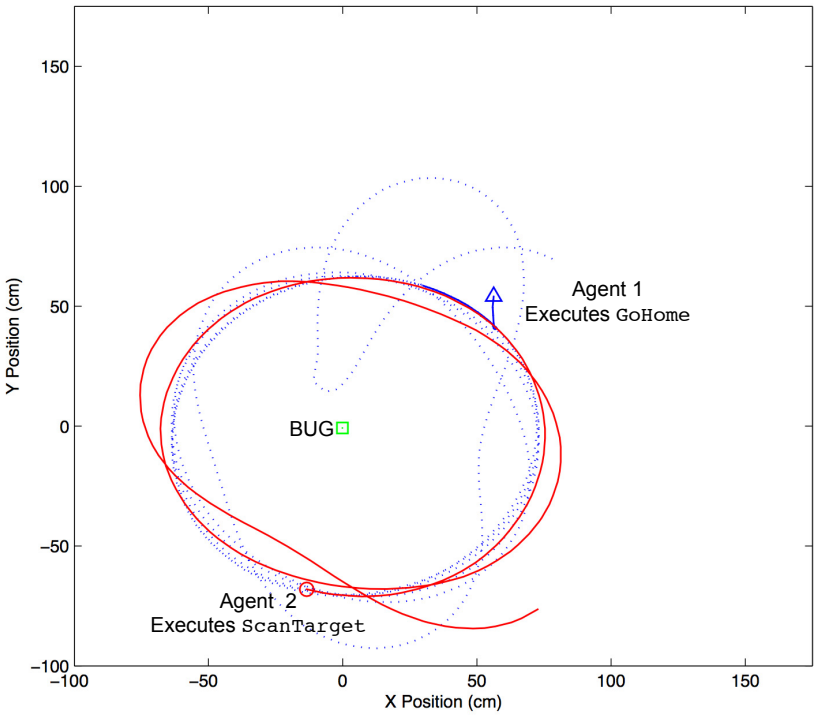


Fig. 5 Once the first agent (△) runs low on its battery level, it sends out a migration message to the other two agents. Since Agent 2 (○) has the necessary devices needed to execute, it accepts the task and starts the same encirclement algorithm. At the same time, Agent 1 begins its `GoHome` behavior.

5 Conclusion

In this paper we designed and demonstrated a new software infrastructure for developing control applications for mobile robots and sensor networks. The benefits of this system are its ability to abstract the sensors, actuators, and network devices as well as the ability to dynamically change how these components operate. Furthermore, the systems allows for the migration of tasks to other agents that have the necessary capabilities to execute them. Our experimental results show that this framework facilitates the development of dynamic control and sensing applications that can incorporate heterogeneous distributed systems.

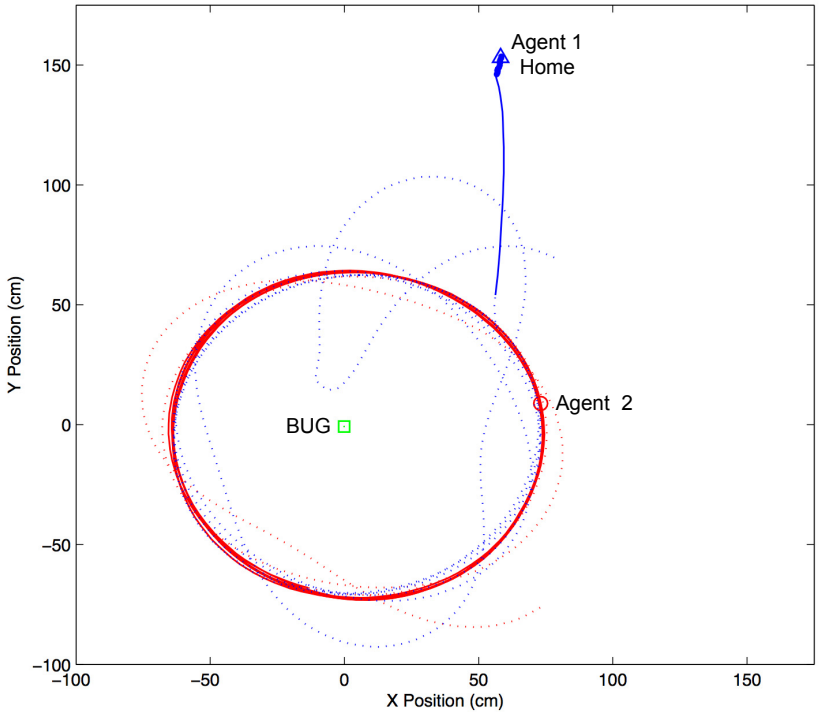


Fig. 6 At this point, agent 1 has finished its `GoHome` behavior. Agent 2 continues to encircle the BUG sensor.

Acknowledgements. This work was sponsored by the US Office for Naval Research through the grant MURI HUNT.

References

1. Bug Labs (2010), `http://www.buglabs.net`
2. CybelePro (2010), `http://products.i-a-i.com`
3. JamVM (2010), `http://jamvm.sourceforge.net`
4. Java Agent Development Framework (2010), `http://jade.tilab.com`
5. Jetlang (2010), `http://code.google.com/p/jetlang`
6. ROS (2010), `http://www.ros.org`
7. Abdelzaher, T., Stankovic, J., Lu, C., Zhang, R., Lu, Y.: Feedback performance control in software services. IEEE Control Systems Magazine, 74–90 (2003)
8. Agha, G.: Concurrent object-oriented programming. Communications of the ACM 33(9), 125–140 (1990)
9. Brooks, A., Kaupp, T., Makarenko, A., Williams, S., Orebäck, A.: Towards component-based robotics. In: Proceedings of the International Conference on Intelligent Robots and Systems (IROS), pp. 163–168 (2005)

10. Cornea, R., Dutt, N., Gupta, R., Kreuger, I., Nicolau, A., Schmidt, D., Shukla, S.: Forge: A framework for optimization of distributed embedded systems software. In: Proceedings of the 17th IEEE/ACM International Parallel and Distributed Processing Symposium (2003)
11. Gerkey, B., Vaughan, R., Howard, A.: The player/stage project: tools for multi-robot and distributed sensor systems. In: Proceedings of the International Conference on Advanced Robotics (ICRA), pp. 317–323 (2003)
12. Hewitt, C.: Viewing control structures as patterns of passing messages. Journal of Artificial Intelligence 8(3), 323–363 (1977)
13. Kulis, Z., Manikonda, V., Azimi-Sadjadi, B., Ranjan, P.: The distributed control framework: A software infrastructure for agent-based distributed control and robotics. In: Proceedings of American Control Conference (2008)
14. LaMarca, A., Brunette, W., Koizumi, D., Lease, M., Sigurdsson, S.B., Sikorski, K., Fox, D., Borriello, G.: Making Sensor Networks Practical with Robots. In: Mattern, F., Naghshineh, M. (eds.) PERVASIVE 2002. LNCS, vol. 2414, pp. 152–166. Springer, Heidelberg (2002)
15. Lee, E.A.: Model-driven development - from object-oriented design to actor-oriented design. In: Workshop on Software Engineering for Embedded Systems: From Requirements to Implementation (a.k.a. The Monterey Workshop) (2003)
16. Montemerlo, M., Roy, N., Thrun, S.: Perspectives on standardization in mobile robot programming: the carnegie mellon navigation (carmen) toolkit. In: Proceedings of the International Conference on Intelligent Robots and Systems (IROS), pp. 2436–2441 (2003)
17. Saffiotti, A., Broxvall, M., Gritti, M., LeBlanc, K., Lundh, R., Rashid, J., Seo, B., Cho, Y.: The peis-ecology project: vision and results. In: Proceedings of the International Conference on Intelligent Robots and Systems (IROS), pp. 2329–2335 (2008)
18. Wang, N., Kircher, M., Schmidt, D.: Applying reflective middleware techniques to optimize a qos-enabled corba component model implementation. In: Proceedings of 24th Annual International Computer Software and Applications Conference, pp. 492–499 (2000)
19. Zhang, F., Justh, E., Krishnaprasad, P.: Boundary following using gyroscopic control. In: Proceedings of the 43rd IEEE Conference on Decision and Control (2004)

Part II
Localization, Navigation and Formations

Part II: Localization, Navigation, and Formations

Magnus Egerstedt

Mobility algorithms for multi-robot systems must inevitably be tailored to the information available to the individual robots. This information is typically limited, which in turn has implications on the mobility algorithms themselves. In other words, they must use only locally available information and, as such, be distributed. At the same time, they should still achieve the desired global properties and this part of the book focuses on this interplay between local sensing, mobility, communications, and cooperation algorithms.

Before one can even start designing algorithms that make teams of mobile robots act in an effective manner, the individual robots must have access to sufficiently rich and reliable information. The *localization* problem deals with a manifestation of this problem, where a team of robots must try to infer their localization from a distributed set of sensors. In Cristofaro, Renzaglia, and Martinelli, the localization problem is studied for a team of micro aerial vehicles. A new information filter is introduced that can be distributed across the individual aerial vehicles, and that allows for the team to localize itself based on noisy GPS signals and inertial data. The SLAM (simultaneous localization and mapping) problem constitutes a generalization of the localization problem in that one needs to simultaneously build up a map of the environment in which the robots are deployed and to localize the robots in this map. The multi-robot SLAM problem is investigated in Abrate, Bona, Indri, Rosa, and Tibaldi, where map updates are managed in a novel way in order to ensure that the inter-agent computational burden remains light.

The *navigation* problem is concerned with the design of algorithms that ensure that a team of mobile robots achieves some goal while avoiding collisions. Otte and Correll investigate how one can structure the computation of collision-free paths in a distributed manner. What is proposed is a separation of time-scales, where a rough, suboptimal path is produced, and this path is then refined over time, subject to communication constraints. In Masehian and Sedighizadeh, a related problem is tackled using distributed particle swarm optimization, whereby global and local planners operate sequentially. An alternative view of the navigation problem – not in terms of path planning but in terms of control laws – is presented in Roussos and

Magnus Egerstedt
Georgia Institute of Technology, Atlanta, GA 30332, USA
e-mail: magnus@gatech.edu

Kyriakopoulos, where the robots move according to a potential field-based feedback law. These potential fields are obtained through decentralized navigation functions and they ensure that collisions between robots as well as with obstacles are avoided. In Alonso-Mora, Breitenmoser, Rufli, Beardsley, and Siegwart, the collision-avoidance problem is further investigated for teams of non-holonomic mobile robots. And, based on the concept of optimal reciprocity, smooth and collision-free paths can be guaranteed.

In Nebot and Cervera, sensing and mobility is combined, and a visual-aided guidance strategy is proposed for driving teams of robots into desired, geometric *formations*. Key to this strategy is the explicit heterogeneity among the robots, whereby one robot acts as a "conductor" in that it drives the formation while the remaining robots ensure that the formation is maintained based on the available visual information. Formation control is also the topic in Mullen, Monekosso, Barman, and Remagnino, where a decentralized formation control strategy is coupled with reactive coordination and control laws. The resulting set of algorithms achieve so-called lattice cohesion in the team in a robust manner. Formations are also considered in Mikkelsen, Jespersen, and Ngo, where potential forces are used to establish the formations in a distributed manner, based on probabilistic inter-robot communication models. A specialized application scenario is considered in Aro, Hu, Vainio, and Halme, where a collection of floating robots have to coordinate in shallow seas. These types of environments are very hard to manage due both to the inherently limited control authority that the individual robots have over their motions, and to the severe bandwidth limitation that under-water communications imply.

Multi-robot systems can abstractly be viewed as a *network* of agents, where the edges in the network encode the information flow in-between agents. As a consequence, it is important to not only focus on the primary mission objectives, such as geometric formation-based objectives, but also on secondary objectives pertaining to the underlying network itself. Simonetto, Keviczky, and Babuška investigate how one key such objective can be achieved, namely the so-called algebraic connectivity of the network, which essentially measures how well-connected the network is. The proposed approach is optimization-based and it operates solely on locally available information. Another network-level objective is synchronization, where the team of robots must coordinate some state in the sense that the individual instantiations of that state should be synchronized, rather than reach particular goal points or inter-agent distances. In Hauert, Leven, Zufferey, and Floreano, the synchronization problem is tackled for a team of fixed-wing flying robots. The key idea is to use beat-based synchronization, whereby the robots' headings align over time, which corresponds to all the robots flying in the same direction.

What makes control, coordination, and sensing across networks of mobile robots so challenging is the tight interplay between the limitations on the sensing and communication capabilities and the mobility algorithms. This part of the book highlights some of these challenges and presents some novel and effective solutions across the entire spectra of problem domains, including navigation, localization, formation control, and network-level coordination and synchronization.

Distributed Information Filters for MAV Cooperative Localization

Andrea Cristofaro, Alessandro Renzaglia, and Agostino Martinelli

Abstract. This paper introduces a new approach to the problem of simultaneously localizing a team of micro aerial vehicles (MAV) equipped with inertial sensors able to monitor their motion and with exteroceptive sensors. The method estimates a delayed state containing the trajectories of all the MAVs. The estimation is based on an Extended Information Filter whose implementation is distributed over the team members. The paper introduces two contributions. The former is a trick which allows exploiting the information contained in the inertial sensor data in a distributed manner. The latter is the use of a projection filter which allows exploiting the information contained in the geometrical constraints which arise as soon as the MAV orientations are characterized by unitary quaternions. The performance of the proposed strategy is evaluated with synthetic data. In particular, the benefit of the previous two contributions is pointed out.

1 Introduction

In recent years, flying robotics has received significant attention from the robotics community. The ability to fly allows easily avoiding obstacles and quickly having an excellent birds eye view. These navigation facilities make flying robots the ideal platform to solve many tasks like exploration, mapping, reconnaissance for search and rescue, environment monitoring, security surveillance, inspection etc. In the framework of flying robotics, micro aerial vehicles (MAV) have a further advantage. Due to the small size they can also be used in narrow out- and indoor environment and they represent only a limited risk for the environment and people living in it. One of the main prerequisite for the successful accomplishment of many tasks is

Andrea Cristofaro · Alessandro Renzaglia · Agostino Martinelli
INRIA Rhône-Alpes, Grenoble, France
e-mail: {andrea.cristofaro,alessandro.renzaglia}@inrialpes.fr
agostino.martinelli@inrialpes.fr

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 133–146.
springerlink.com

a precise vehicle localization. Since micro aerial vehicles are equipped with low computational capabilities an efficient solution must be able to distribute the computation among all the agents in order to exploit the computational resources of the entire team. Distributing the computation has also another key advantage. It allows us to make the solution robust with respect to failures. On the other hand, distributing the computation must also account for the limited communication capabilities.

The cooperative localization problem was formulated in [11] and it has been faced by many authors so far. Fox and collaborators [3] introduced a probabilistic approach based on Markov localization. Their approach has been validated through real experiments showing a drastic improvement in localization speed and accuracy when compared to conventional single robot localization. Other approaches take advantage of relative observations for multi-robot localization [4, 5, 9, 16, 17, 20]. In [5] a method based on a combination of maximum likelihood estimation and numerical optimization was introduced. This method allows to reduce the error in the robot localization by using the information coming from relative observations among the robots in the team. In [17], a distributed multi robot localization strategy was introduced. This strategy is based on an Extended Kalman Filter to fuse proprioceptive and exteroceptive sensor data. In [13], the same approach was adapted in order to deal with any kind of relative observations among the robots. In [17], it was shown that the equations can be written in a decentralized form, allowing the decomposition into a number of smaller communicating filters. However, the distributed structure of the filter only regards the integration of the proprioceptive data (i.e. the so called prediction phase). As soon as an observation between two robots occurs, communication between each member of the team and a single processor (which could be embedded in a member of the team) is required. The same communication skill is required when even an exteroceptive measurements which only regards a single robot occurs (e.g. a GPS measurement). Furthermore, the computation required to integrate the information coming from this observation is entirely performed by a single processor with a computational complexity which scales quadratically with the number of robots. Obviously, the centralized structure of the solution in dealing with exteroceptive observations becomes a serious inconvenience when the communication and processing capabilities do not allow to integrate the information contained in the exteroceptive data in real time. In particular, this happens as soon as the number of robots is large, even if each robot performs very few exteroceptive observations. In [14] this problem was considered. However, the structure of the filter was maintained the same as in [17] (namely centralized in dealing with exteroceptive data). Each robot was supposed to be equipped with several sensors and the optimal sensing frequencies were analytically derived by maximizing the final localization accuracy. The limit of this approach is that as the number of robots increases, the sensing frequencies reduce. In other words, by performing the estimation process in a centralized fashion it is necessary to reduce the number of observations to be processed as the number of robots increases. Hence, distributing the entire

estimation process can provide a great improvement. Very recently a decentralized cooperative localization approach has been presented in [12].

The information filter is very appealing in this framework since the integration of the exteroceptive data is very simple and could be easily distributed. On the other hand, the equations which characterize the prediction step are much more complex and their distributed implementation seems to be forbidden. This is a serious inconvenience since the proprioceptive data run at a very high frequency.

Eustice et al. [2] and Caballero et al. [1] have recently shown that by using a delayed state also the prediction step has some nice properties. In particular, in [2] a solution to the SLAM problem by using an Extended Information Filter (EIF) to estimate a delayed state has been proposed. In [1] the tracking problem has been considered.

In this paper we consider the problem of cooperative localization in $3D$ when the MAVs are equipped with inertial sensors and exteroceptive sensors (e.g. range sensors and GPS). We adopt a delayed state and we perform its estimation by using an Extended Information Filter. We introduce a simple trick which allows us to mathematically express the quantities measured by the IMU (Inertial Measurement Unit) as a function of the delayed state (i.e. the state to be estimated). In other words, by using this trick, the link between sensor-state for the IMU (which are typically proprioceptive sensors) has the same mathematical expression of the one which characterizes an exteroceptive observation. This allows us to use the equations of the integration of the exteroceptive data also to integrate the IMU data. In this way the equations of the EIF prediction step are never used and the overall estimation process can be easily distributed.

The second contribution of this paper is related to another important issue which arises when dealing with a $3D$ environment. The orientation of a MAV which moves in $3D$ is provided by 3 parameters. On the other hand, the MAV dynamics become very easy by adopting quaternions. However, this parameterization is redundant. This means that part of the information is frozen in a geometrical constraint. Without using this constraint part of the information is not exploited and the overall precision gets worse. To the best of our knowledge, this issue has never been considered in the framework of flying robotics. On the other hand, the problem of exploiting the information contained in geometrical constraints is not new in the mobile robotics literature. In particular, it has been considered in SLAM when using a relative map. To this regard a new filter, the projection filter, has been introduced [15]. In this paper we will adopt the same approach. In particular, we consider the geometrical constraint (expressing that the quaternion must be unitary) as an ideal observation.

The paper is structured as follows. In Section 2 it is given a detailed description of the dynamics, the measurement model and the estimation process with the EIF for a single MAV. Section 3 is dedicated to the extension of the previous results to multi robot systems; in particular a distributed EIF algorithm is presented, taking into account relative observations between the robots. In Section 4 we present some simulation results to illustrate the efficiency of the estimation algorithm.

2 The Case of One Single MAV

For the sake of clarity, we begin our analysis by the description of the model for a single MAV. The extension of the presented dynamics and measurement model to multi robot systems is straightforward.

2.1 The System

We provide here a mathematical description of our system. We introduce a global frame, whose z-axis is the vertical one. Let us consider a MAV equipped with IMU proprioceptive sensors (one tri-axial accelerometer and one tri-axial gyroscope) as well as some suitable exteroceptive sensors (GPS, range sensors). In this paper we assume that the IMU data are unbiased. From a practical point of view, unbiased data can be obtained by continuously calibrating the IMU sensors (see for instance [6]). The configuration of the MAV is described by a vector $(r, v, \theta) \in \mathbf{R}^9$ where $r = (r_x, r_y, r_z) \in \mathbf{R}^3$ is the position, $v = (v_x, v_y, v_z) \in \mathbf{R}^3$ is the speed and $\theta = (\theta_\mathrm{r}, \theta_\mathrm{p}, \theta_\mathrm{y}) \in \mathbf{R}^3$ assigns the MAV orientation: θ_r is the roll angle, θ_p is the pitch angle and θ_y is the yaw angle. We will adopt lower case letters to express a quantity in the global frame, while capital letters for the same quantity expressed in the local frame (i.e. the one attached to the MAV).The system description can be simplified adopting a quaternions framework. We recall that the quaternions space $\mathbf{H}$ is the non-commutative set of elements

$$\mathbf{H} = \left\{ q_t + q_x i + q_y j + q_z k : \quad q_t, q_x, q_y, q_z \in \mathbf{R}, \quad i^2 = j^2 = k^2 = ijk = -1 \right\}.$$

For an arbitrary quaternion $q = q_t + q_x i + q_y j + q_z k$, we define the conjugate element $q^* = q_t - q_x i - q_y j - q_z k$ and the norm $||q|| = \sqrt{qq^*} = \sqrt{q^* q} = \sqrt{q_t^2 + q_x^2 + q_y^2 + q_z^2}$.

Let us denote by a_g the gravity acceleration (i.e. $a_g = -(0,0,g)$ with $g \simeq 9.81 m/s^2$) and by A, Ω the acceleration and the angular speed provided by the IMU; regarding the acceleration, the one perceived by the accelerometer (A) is not simply the MAV acceleration ($\overline{A}$): it also contains the gravity acceleration (A_g). In particular, we have $A = \overline{A} - A_g$ since, when the camera does not accelerate (i.e. $\overline{A} = 0$) the accelerometer perceives an acceleration which is the same of an object accelerated upward in the absence of gravity.

The continuous-time dynamics of the MAV is given by the following system of ordinary differential equations

$$\dot{r} = v \tag{1}$$

$$\dot{v} = q \cdot \overline{A} \cdot q^* = q \cdot A \cdot q^* + a_g \tag{2}$$

$$\dot{q} = \frac{1}{2} q \cdot \Omega \tag{3}$$

where r, v, Ω, A are purely imaginary quaternions, while q is a unitary quaternion. The following relations for roll, pitch and yaw angles $\theta_{\mathrm{r}}, \theta_{\mathrm{p}}, \theta_{\mathrm{y}}$ hold

$$\theta_{\mathrm{r}} = \frac{q_t q_x + q_y q_z}{1 - 2(q_x^2 + q_y^2)}$$

$$\theta_{\mathrm{p}} = q_t q_y - q_x q_z$$

$$\theta_{\mathrm{y}} = \frac{q_t q_z + q_y q_x}{1 - 2(q_y^2 + q_z^2)}.$$

During the exploration, the MAV performs measurements thanks to its exteroceptive sensors equipment; such measurements can be individual (i.e. GPS-based measurements) as well as relative to other MAVs poses or to the position of fixed landmarks. The general single MAV observation equation is given by

$$z = h(r, v, q) \tag{4}$$

where $h(\cdot, \cdot, \cdot)$ is a known function.

In the case the exteroceptive sensor is a GPS, the observation equation is very simple as it is linear

$$z_{GPS} = r. \tag{5}$$

2.2 Estimation with the EIF: The Integration of the Exteroceptive Data

Let us denote with Σ and ξ the information matrix and the information vector respectively; in addition let R be the covariance matrix characterizing the measurement error for an exteroceptive sensors. The update equations at the time step i are (see [21]):

$$\Sigma_i = \overline{\Sigma}_i + \Sigma_{obs}, \qquad \Sigma_{obs} = H_i^T R^{-1} H_i, \tag{6}$$

$$\xi_i = \overline{\xi}_i + \xi_{obs}, \qquad \xi_{obs} = H_i^T R^{-1} \left[z_i - h(\overline{\mu}_i) + H_i \overline{\mu}_i \right], \tag{7}$$

where $\overline{\Sigma}_i, \overline{\xi}_i$ are the predicted information matrix and information vector, $\overline{\mu}_i = \overline{\Sigma}_i^{-1} \overline{\xi}_i$ is the predicted mean value and H_i is the Jacobian of the observation function h evaluated at $\overline{\mu}_i$. The structure of such equation is very simple as the update consists only in summing the new information from the exteroceptive sensors to the predicted values.

The case of GPS observations is even easier to treat; since the function h is linear we have $h(\overline{\mu}_i) = H_i\overline{\mu}_i = \overline{\mu}_i$ and hence the update equation for the information filter is

$$\xi_i = \overline{\xi}_i + H_i^T R^{-1} z_i.$$

In particular the explicit computation of the mean value is not performed and this is a key advantage since the information matrix inversion requires in general a high computational burden.

2.3 Estimation with the EIF: The Integration of the Proprioceptive Data

Denoting by Q a noise term affecting the system dynamics, the prediction steps are given by

$$\overline{\Sigma}_i = \left[F_i \Sigma_{i-1}^{-1} F_i^T + Q\right]^{-1}, \tag{8}$$

$$\overline{\xi}_i = \overline{\Sigma}_i F_i \Sigma_{i-1}^{-1} \xi_{i-1}, \tag{9}$$

where F_i is the Jacobian of the dynamics evaluated at the estimated mean value $\mu_{i-1} = \Sigma_{i-1}^{(-1)} \xi_{i-1}$.

Remark 1. In a multi robot scenario, where Σ and ξ characterize the probability distribution of all the MAVs, a distributed algorithm for the implementation of update equations (6)-(7) can be designed (see Section 3 and [1]). On the other hand, the prediction equations (8)-(9) are more complicated and they cannot be easily distributed. Nevertheless we will show that, once a delayed-state is considered, data obtained from proprioceptive sensors can be integrated using only the update equations (6)-(7).

Let us introduce the delayed-state

$$X_i = (r_0, q_0, r_1, ..., r_i, q_i)$$

containing all MAV poses until the i-th time step. The discretization of the dynamics equations over a Δt time-step interval gives

$$r_{i+1} = r_i + v_i \Delta t \tag{10}$$

$$v_{i+1} = v_i + q_i \cdot \int_i^{i+\Delta t} A dt \cdot q_i^* + a_g \Delta t \tag{11}$$

$$q_{i+1} = q_i + \frac{1}{2} q_i \cdot \int_i^{i+\Delta t} \Omega dt \tag{12}$$

From Equation (10) we can get

$$v_i = (r_{i+1} - r_i)/\Delta t$$

and hence the following recursive formula holds

$$r_{i+1} = 2r_i - r_{i-1} + \Delta t \left(q_i \cdot \int_i^{i+\Delta t} A dt \cdot q_i^* + a_g \Delta t \right), \tag{13}$$

corresponding to a second order continuous-time evolution. Setting

$$\tilde{A}_i = \int_i^{i+\Delta t} A dt \tag{14}$$

and

$$\tilde{\Omega}_i = \int_i^{i+\Delta t} \Omega dt, \tag{15}$$

the proprioceptive measurements can be regarded as delayed-state dependent functions:

$$\tilde{A}_i = h_A(r_{i-2}, r_{i-1}, r_i, q_i) = \frac{q_i^*(-a_g \Delta t^2 + r_i - 2r_{i-1} + r_{i-2})q_i}{\Delta t}$$

$$\tilde{\Omega}_i = h_\Omega(q_{i-1}, q_i) = 2q_{i-1}^*(q_i - q_{i-1}).$$

In other words, $\tilde{A}_i$ and $\tilde{\Omega}_i$ are functions of the state X_i to be estimated; moreover, since we are considering the discrete dynamics given by (12)-(13), there is no need to include the MAV speed v into the state vector X_i.

Due to these considerations, we are allowed to integrate proprioceptive data using (6)-(7) instead of (8)-(9), with a consequent reduction of computational cost in the estimation algorithm.

For nonlinear measurements equation (7) involves the mean value and hence information matrix inversion is required; nevertheless in many situation, due to the sparsity of such matrix, a partial state recovery is sufficient in order to guarantee a good estimate (see [2]). Whole state recovering can be obtained using for example the Conjugate Gradients algorithm (see [19]) or the Givens rotations factorization (see [8]). We point out that at any update step, i.e. when a true exteroceptive measurement is performed, the size of the delayed-state vector X increases by $3+4=7$.

2.4 Projection Filter: Integration of Ideal Constraints

As mentioned in the introduction, the quaternion structure is redundant for the problem we are considering and this may lead to a loss of information. To avoid this problem we have assumed that the quaternion q is unitary. On the other hand, if the discrete dynamics (12) is considered, such property is no longer preserved. Anyway, we can take into account the norm invariance of q_i imposing an ideal constraint with a fake observation given by the function

$$h_0(q) = 1 - q_t^2 + q_x^2 + q_y^2 + q_z^2;$$

in other words, we can regard the norm constraint as the measurement

$$z_i = h_0(q_i) = 0.$$

Integration of such fake measurement can be performed with the projection filter (see [15]).

3 The Cooperative Case

3.1 The System

We consider now a fleet of $N > 1$ MAVs, each one having the characteristics described in Section 2. Let us denote by $(r^{(k)}, q^{(k)})$ the coordinates of the k-th MAV; the discrete dynamics is given by

$$r_{i+1}^{(k)} = 2r_i^{(k)} - r_{i-1}^{(k)} + \Delta t \left(q_i^{(k)} \cdot \int_i^{i+\Delta t} A^{(k)} dt \cdot (q_i^{(k)})^* + a_g \Delta t \right) \tag{16}$$

$$q_{i+1}^{(k)} = q_i^{(k)} + \frac{1}{2} q_i^{(k)} \cdot \int_i^{i+\Delta t} \Omega^{(k)} dt. \tag{17}$$

Each MAV, in addition to the measurement model (4), may perform relative observation; the general multi robot observation equation can be written as

$$z_i^{(k)} = h^{(k)}(r_i^{(1)}, q_i^{(1)}, ..., r_i^{(k)}, q_i^{(k)}, ..., r_i^{(N)}, q_i^{(N)}). \tag{18}$$

Simple and common examples of relative observations are distance measures. If the k-th MAV measures its own distance from the j-th MAV, the observation is given by

$$z_i^{(k)} = (r_{i,x}^{(k)} - r_{i,x}^{(j)})^2 + (r_{i,y}^{(k)} - r_{i,y}^{(j)})^2 + (r_{i,z}^{(k)} - r_{i,z}^{(j)})^2.$$

3.2 The Distributed EIF

In [1] it is shown that delayed-states allow to distribute the estimation process over the entire MAVs network. In particular the authors explain how to recover the global belief from the local belief of each network node and remark that the same operation with standard (non delayed) states is not possible at all. We will follow a similar approach, with a slightly different communication and data fusion algorithm.

When the exploration starts, each MAV begins to integrate the information provided by its own sensors by equation (6)-(7) as described before. In particular for any measurement, the incoming data are stored in the bottom-right block of the information matrix and, as a consequence, in the last entries of the information vector:

$$\Sigma_{i-1} \to \Sigma_i = \begin{pmatrix} \Sigma_{i-1} & 0^{7(i-1)\times 7} \\ 0^{7\times 7(i-1)} & 0^{7\times 7} \end{pmatrix} + \begin{pmatrix} 0^{7(i-3)\times 7(i-3)} & 0^{7(i-3)\times 21} \\ 0^{21\times 7(i-3)} & \Sigma_{obs} \end{pmatrix}$$

$$\xi_{i-1} \to \xi_i = \begin{pmatrix} \xi_{i-1} \\ 0^{7\times 1} \end{pmatrix} + \begin{pmatrix} 0^{7(i-3)\times 1} \\ \xi_{obs} \end{pmatrix}.$$

Suppose that after i_1 updating time-steps for the j_1-th MAV and i_2 steps for the j_2-th MAV a relative measurement occurs and for sake of simplicity suppose that $j_1 < j_2$. Each MAV has to increase the size of the information matrix and information vector in order to store the new data. The process is carried out following the steps described below:

1. *State augmentation*. The states of the two MAVs are increased in order to have the same size $7(i_1 + i_2)$; this can be done adding a suitable number of zeros in the information matrix and information vector.

$$\Sigma_{(j_1),i_1} \to \begin{pmatrix} \Sigma_{(j_1),i_1} & 0^{7i_1\times 7i_2} \\ 0^{7i_2\times 7i_1} & 0^{7i_2\times 7i_2} \end{pmatrix}, \quad \xi_{(j_1),i_1} \to \begin{pmatrix} \xi_{(j_1),i_1} \\ 0^{7i_2\times 1} \end{pmatrix}$$

$$\Sigma_{(j_2),i_2} \to \begin{pmatrix} 0^{7i_1\times 7i_1} & 0^{7i_1\times 7i_2} \\ 0^{7i_2\times 7i_1} & \Sigma_{(j_2),i_2} \end{pmatrix}, \quad \xi_{(j_2),i_2} \to \begin{pmatrix} 0^{7i_1\times 1} \\ \xi_{(j_2),i_2} \end{pmatrix}$$

2. *Relative estimation*. The information from relative observations are integrated using the standard update equations (6)-(7). Correlation between the estimates on the last poses of the MAVs may appear, so that the updated matrices may be not block-diagonal.

$$\Sigma_{(j_1),i_1} \to \begin{pmatrix} \Sigma_{(j_1),i_1} & * \\ * & * \end{pmatrix}, \quad \xi_{(j_1),i_1} \to \begin{pmatrix} \xi_{(j_1),i_1} \\ * \end{pmatrix}$$

$$\Sigma_{(j_2),i_2} \to \begin{pmatrix} * & * \\ * & \Sigma_{(j_2),i_2} \end{pmatrix}, \quad \xi_{(j_2),i_2} \to \begin{pmatrix} * \\ \xi_{(j_2),i_2} \end{pmatrix}$$

3. *Data fusion*. A communication is established between the MAVs and they exchange their stored data. The data fusion scheme is a non negligible theoretical issue: as a matter of fact, if the process is carried out taking simply the sum of the contributions from each MAV, estimation errors may arise due to adding several

times the same information. Following [1], we have adopted a fusion algorithm based on a convex combination of the data:

$$\Sigma_{(j_1),i_1} \rightarrow \omega\Sigma_{(j_1),i_1} + (1-\omega)\Sigma_{(j_2),i_2}, \quad \xi_{(j_1),i_1} \rightarrow \omega\xi_{(j_1),i_1} + (1-\omega)\xi_{(j_2),i_2}$$

$$\Sigma_{(j_2),i_2} \rightarrow (1-\omega)\Sigma_{(j_1),i_1} + \omega\Sigma_{(j_2),i_2}, \quad \xi_{(j_2),i_2} \rightarrow (1-\omega)\xi_{(j_1),i_1} + \omega\xi_{(j_2),i_2}$$

As proved in [7], for any $0 < \omega < 1$, the above convex combinations lead to unbiased and consistent estimates, i.e. no overconfident estimate is performed and there is no overlapping of information. This allows us to limit communication. In particular, two robots must communicate only when a relative measurements between them occurs.

4 Performance Evaluation

In order to validate our approach we perform simulations that are described in the following sections.

4.1 The Simulated Environment

The trajectories of the MAVs are generated randomly and independently one from each other. In particular, for every MAV, the motion is generated by generating randomly the linear and angular acceleration at $100Hz$. Specifically, at each time step, the three components of the linear and the angular acceleration are generated as Gaussian independent variables with mean values μ_a and $\mu_{\dot{\Omega}}$ and with covariance matrices P_a and $P_{\dot{\Omega}}$. By performing many simulations we remarked that the precision of the proposed strategy is almost independent of all these parameters. The simulations provided in this section are obtained with the following settings: $\mu_a = \mu_{\dot{\Omega}} = [000]^T$, $P_a = \begin{bmatrix} (5ms^{-2})^2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ and $P_{\dot{\Omega}} = \begin{bmatrix} (10deg\ s^{-2})^2 & 0 & 0 \\ 0 & (10deg\ s^{-2})^2 & 0 \\ 0 & 0 & (10deg\ s^{-2})^2 \end{bmatrix}$

We adopt many different values for the initial MAV positions orientations and speeds. We also consider different scenarios corresponding to a different number of MAVs.

Starting from the accomplished trajectories, the true angular speed and the linear acceleration are computed at each time step of $0.01s$ (respectively, at the time step i, we denote them with Ω_i^{true} and A_i^{true}). Starting from them, the IMU sensors are simulated by generating randomly the angular speed and the linear acceleration at each step according to the following: $\Omega_i = N\left(\Omega_i^{true}, P_{\Omega_i}\right)$ and $A_i = N\left(A_i^{true} - A_{g\,i}, P_{A_i}\right)$ where N indicates the Normal distribution whose first

entry is the mean value and the second one its covariance matrix and P_{Ω_i} and P_{A_i} are the covariance matrices characterizing the accuracy of the IMU; finally, A_g is the gravity acceleration expressed in the local frame. In all the simulations we set both P_{A_i} and P_{Ω_i} diagonal matrices. In the results here provided they are set as follows: $P_{A_i} = \begin{bmatrix} (0.1ms^{-2})^2 & 0 & 0 \\ 0 & (0.1ms^{-2})^2 & 0 \\ 0 & 0 & (0.1ms^{-2})^2 \end{bmatrix}$ and $P_{\Omega_i} = \begin{bmatrix} (10deg\ s^{-1})^2 & 0 & 0 \\ 0 & (10deg\ s^{-1})^2 & 0 \\ 0 & 0 & (10deg\ s^{-1})^2 \end{bmatrix}$ for every step i.

The MAVs are also equipped with GPS and range sensors. The GPS provides the position of the MAV with a Gaussian error whose covariance is a diagonal matrix and whose components are equal to $25m^2$. The GPS data are delivered at $5Hz$. Finally, the range sensors provide the distances among the MAVs at $2Hz$ and the measurement errors are normally distributed with variance $(0.01m)^2$.

All the previous parameters were set in order to be close to a real scenario [10].

4.2 Results

We provide some of the results obtained with the previous settings and by simulating N MAVs. In particular, we consider the case of $N = 3$ and $N = 5$. Furthermore, we consider separately the cases when the estimation is performed by only integrating the IMU data, by combining the IMU data with the GPS data and by combining all the sensor data. Finally, in order to evaluate the benefit of using the projection filter discussed in Section 2.4, we consider separately the cases when this filter is adopted and when it is not adopted.

Fig. 1a-b show the results obtained with three MAVs. The blue dots represent the ground truth. In fig. 1-a the magenta dots represent the GPS data and the black circles the trajectories estimated by only integrating the IMU data. It is clear that both IMU and GPS are very noisy and cannot be used separately to estimate the MAV trajectories. In fig. 1-b the green dots represent the trajectories estimated by fusing the IMU data and the GPS data with our proposed approach (EIF and projection filter). Finally, the red dots represent the result obtained by also fusing the range measurements. We remarked that the use of the range measurements further reduce the error. In particular, for the simulation in fig. 1a-b the position error averaged on all the three MAV and on all the time steps is equal to $0.6m$ without the range measurements and $0.45m$ with them. As expected, this improvement is still larger by increasing the number of MAVs (see for instance [18]). In fig. 1c-d the results obtained by using 5 MAVs is shown. The position error obtained by also fusing the range measurements reduces to $0.2m$.

Fig. 2 shows the benefit of using the Projection filter discussed in Section 2.4. In particular, in fig. 2a the red circles represent the trajectories estimated by fusing all the sensor data and by running the Projection Filter at $5Hz$ while in fig. 2b the red circles represent the trajectories estimated without the use of the Projection Filter.

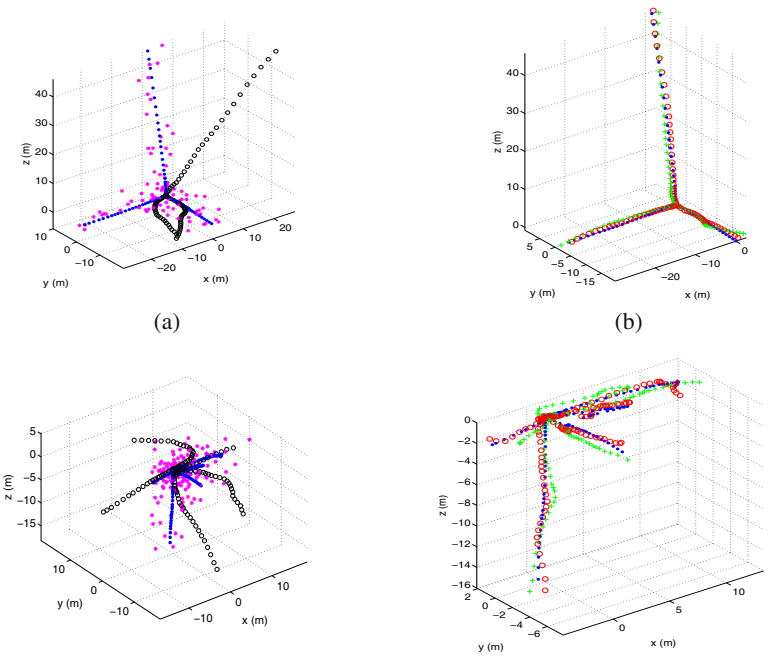


Fig. 1 Blue points represent true MAVs trajectories, black circles are the trajectories with only odometric estimates, magenta stars are the GPS data, green stars are the trajectory estimates without taking into account relative observations and red circles are the estimates with the complete distributed EIF. Figures (1a)-(1b) are the simulation of 3-MAV scenario, while in Figures (1c)-(1d) is plotted the evolution of a 5-MAV system.

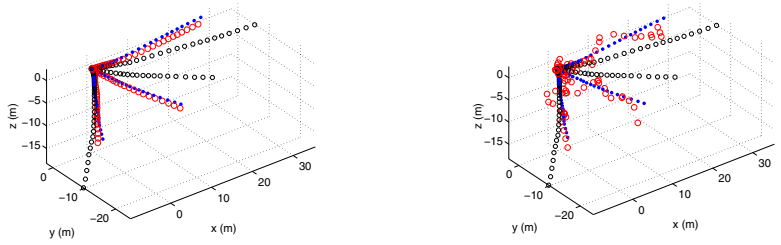


Fig. 2 Blue points represent true MAVs trajectories, black circles are the estimated trajectories via odometry and red circles are the estimated trajectories with the EIF. Figure (2a) represents the simulation of a 3-MAV system; Figure (2b) represents the same scenario without taking into account the information provided by the projection filter.

As in the previous figures, the ground truth is represented with blue dots and the black dots represent the trajectories obtained by a simple integration of the IMU data.

5 Conclusions

In this paper we have discussed an approach to perform cooperative localization of a team of micro aerial vehicles equipped with inertial sensors (one accelerometer and one gyroscope) and exteroceptive sensors (GPS and range sensors). The approach is based on an Extended Information Filter whose implementation is distributed over the team members.

Two original contributions have been introduced. The former consists of a simple trick which allowed us to avoid the equations which characterize the prediction phase of the extended information filter. In particular, the information contained in the data provided by the inertial sensors is exploited by using the equations which characterize the perception step of the EIF. This allowed us to easily distributing the entire estimation process over all the team members. The latter contribution is the use of a projection filter which allowed exploiting the information contained in the geometrical constraints which arise as soon as the MAV orientations are characterized by unitary quaternions.

The performance of the proposed approach was evaluated by using synthetic data.

Acknowledgements. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n. 231855 (sFly).

References

1. Capitan, J., Merino, L., Caballero, F., Ollero, A.: Delayed-state Information Filter for Cooperative Decentralized Tracking. In: IEEE International Conference on Robotics and Automation, Kobe, Japan, pp. 3865–3870 (2009)
2. Eustice, R.M., Singh, H., Leonard, J.J.: Exactly Sparse Delayed-State Filters for View-Based SLAM. IEEE Trans. on Robotics 22(6), 1100–1114 (2006)
3. Fox, D., Burgard, W., Kruppa, H., Thrun, S.: A Probabilistic Approach to Collaborative Multi-Robot Localization. Autonomous Robots 8, 325–344 (2000)
4. Grabowski, R., Navarro-Serment, L.E., Paredis, C.J.J., Khosla, P.K.: Heterogeneous Teams of Modular Robots for Mapping and Exploration. Autonomous Robots 8(3), 325–344 (2000)
5. Howard, A., Mataric, M.J., Sukhatme, G.S.: Localization for Mobile Robot Teams Using Maximum Likelihood Estimation. In: International Conference on Intelligent Robot and Systems (IROS 2002), Lausanne, Switzerland, pp. 2849–2854 (2002)
6. Jones, E., Vedaldi, A., Soatto, S.: Inertial Structure from Motion with Autocalibration. In: ICCU Workshop (2007)
7. Julier, S.J., Uhlmann, J.K.: A Non-divergent Estimation Algorithm in the Presence of Unknown Correlations. In: American Control Conference, Albuquerque, New Mexico, pp. 2369–2373 (1997)

8. Kaess, M., Ranganathan, A., Dellaert, F.: iSAM: Incremental Smoothing and Mapping. IEEE Trans. on Robotics 24(6), 1365–1378 (2008)
9. Kato, K., Ishiguro, H., Barth, M.: Identifying and Localizing Robots in a Multi-Robot System Environment. In: International Conference on Intelligent Robot and Systems, IROS 1999 (1999)
10. Kneip, L., Martinelli, A., Weiss, S., Scaramuzza, D., Siegwart, R.: A Closed-Form Solution for Absolute Scale Velocity Determination Combining Inertial Measurements and a Single Feature Correspondence. Accepted as Contributed Papers at the IEEE International Conference on Robotics and Automation (ICRA 2011), Shanghai, China (2011)
11. Kurazume, R., Nagata, R., Hirose, S.: Cooperative Positioning with Multiple Robots. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 1250–1257 (1994)
12. Leung, K.Y., Barfoot, T.D., Liu, H.H.: Decentralized Localization of Sparsely-Communicating Robot Networks: A Centralized-Equivalent Approach. IEEE Trans. on Robotics 26(1), 62–77 (2010)
13. Martinelli, A., Pont, F., Siegwart, R.: Multi-Robot Localization Using Relative Observations. In: International Conference on Robotics and Automation (ICRA), Barcelona, Spain (2005)
14. Mourikis, A.I., Roumeliotis, S.I.: Optimal Sensing Strategies for Mobile Robot Formations: Resource-Constrained Localization. In: Robotics: Science and Systems (RSS), Cambridge, Massachusetts, USA (2005)
15. Newman, P.: On the structures and solution of simultaneous localization and mapping problem, PhD thesis, Australian Center for Field Robotics, Sidney (1999)
16. Rekleitis, I.M., Dudek, G., Milios, E.E.: Multi-robot cooperative localization: a study of trade-offs between efficiency and accuracy. In: International Conference on Intelligent Robot and Systems (IROS 2002), Lausanne, Switzerland (2002)
17. Roumeliotis, S.I., Bekey, G.A.: Distributed Multirobot Localization. IEEE Transaction on Robotics and Automation 18(5) (October 2002)
18. Roumeliotis, S.I., Rekleitis, I.M.: Propagation of Uncertainty in Cooperative Multirobot Localization: Analysis and Experimental Results. Autonomous Robots 17(1), 41–54 (2004)
19. Schewchuk, J.: An Introduction to Conjugate Gradient Method without Agonizing Pain. Carnegie-Mellon University, Pittsburgh, PA, Tech. Report CMU-CS-94-125 (1994)
20. Spletzer, J.R., Taylor, C.J.: A Bounded Uncertainty Approach to Multi-Robot Localization. In: International Conference on Intelligent Robot and Systems (IROS 2003), Las Vegas, USA (2003)
21. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press, Cambridge (2005)

Multi-robot Map Updating in Dynamic Environments

Fabrizio Abrate, Basilio Bona, Marina Indri, Stefano Rosa, and Federico Tibaldi*

Abstract. Multi-robot systems play an important role in many robotic applications. A prerequisite for a team of robots is the capability of building and maintaining updated maps of the environment. The simultaneous estimation of the trajectory and the map of the environment (known as SLAM) requires many computational resources. Moreover, SLAM is generally performed in environments that do not vary over time (called *static* environments), whereas real applications commonly require navigation services in *dynamic* environments. This paper focuses on long-term mapping operativity in presence of variations in the map, as in the case of robotic applications in logistic spaces, where rovers have to track the presence of goods in given areas. In this context classical SLAM approaches are generally not directly applicable, since they usually apply in static environments or in dynamic environments where it is possible to model the environment dynamics. This paper proposes a methodology that allows the robots to detect variations in the environment, generate maps containing only the persistent variations, propagate thiem to the team and finally merge the received information in a consistent way. The team of robots is also exploited to assure the coverage of areas not visited for long time, thus improving the knowledge on the present status of the map. The map updating

Fabrizio Abrate
Istituto Superiore Mario Boella, via P.C. Boggio 61 10129 Torino, Italy
e-mail: fabrizio.abrate@ismb.it

Basilio Bona · Marina Indri · Stefano Rosa · Federico Tibaldi
Dipartimento di Automatica e Informatica, Politecnico di Torino,
Corso Duca degli Abruzzi 24, 10129 Torino, Italy
e-mail: {basilio.bona,marina.indri,stefano.rosa,
federico.tibaldi}@polito.it

* This work was supported by Regione Piemonte under the "MACP4Log" grant (RU/02/26), the "Piattaforma Tecnologica for the Internet of Things Project" and by Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR) under MEMONET National Research Project.

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 147–160.
springerlink.com

process is demonstrated to be computationally light, in order to be performed in parallel with other tasks (*e.g.*, team coordination and planning, surveillance).

1 Introduction

Mobile robot systems have been involved in many successful applications including museum guide robots, surveillance, planetary exploration, search and rescue [13]. To successfully accomplish these tasks, the robots shall be able to build maps of unknown environments and to localize therein. The joint estimation of both the position and the map model is referred to as Simultaneous Localization And Mapping (SLAM). While the maturity of SLAM in single robot scenarios is recognized in many recent works, many issues arise when trying to extend these approaches to multi-robot scenarios. One of the first multi-robot approaches is given in [8], where a cooperative SLAM algorithm is proposed to merge sensor and navigation information from multiple autonomous vehicles, on the basis of stochastic estimation and feature-based landmark extraction from the environment. In [16] the Constrained Local Submap Filter (CLSF) is exploited to create a local submap of the features in the immediate vicinity of the vehicle, periodically fused into the global map of the environment. This representation reduces the computational complexity of maintaining the global map estimates as well as it improves the data association process. Some approaches, as [5] and [9], are based on Rao-Blackwellized particle filters (RBPF), while others [17] are based on Kalman filtering. The approach proposed in [10] is based on manifold representation of maps. This approach has been mainly designed to overcome limitations of existing SLAM methods, especially the sensitivity to false data associations. Other approaches like [4] speed up mapping by using multiple robots exploring different parts of the environment. In general, the problems in multi-robot systems are still related to the need for team coordination strategies and to the high computational and memory requirements depending on the number of robots and the map size. Moreover service robotic applications have to cope with intrinsically dynamic environments. Realistic applications require updated maps of the environments that vary over time, starting from a given initial condition. This is for instance the case of robotic applications in logistic spaces, where robots have to track the presence of goods in certain areas. The goods are stored in appropriate places, but during the day they can be removed and substituted by other items many times. In these scenarios classical SLAM approaches are not suitable, as it could be at least difficult or even impossible to model the dynamics of the environment. Furthermore when dealing with very large environments the memory requirements for multi-robot SLAM could become too high. The problem of keeping an updated map of the environment in order to preserve the robots localization, without investigating any specific goods tracking procedure, has been faced in [3] for the single-robot case.

In this paper this solution is extended to a multi-robot scenario. The concept of *time-map* is introduced to assign to each cell in the map a value representing its

reliability. This time-map is used to merge in an appropriate way the changes detected locally by a robot and the updated maps received from the other team members. The effectiveness of the approach is improved by a simple team coordination strategy, which we propose to actively search for modifications in the map. Finally experimental results of simulated and real tests are carried out to evaluate the effectiveness of the algorithm and its computational load.

2 Problem Formulation

A team of mobile robots, each endowed with a laser rangefinder and wireless connectivity, is supposed to be correctly localized with respect to the available environment map. Each robot is assumed to be in the *position tracking* state, as defined in [1] and [2].

Each robot uses an occupancy grid map of the environment in the localization algorithm to track its position over time. Such a map could have been manually created or previously built by a SLAM algorithm.

At discrete time instants k the environment changes, and the robots have to modify their map to take into account the variation. This phase is called Δ*-mapping* step.

The set of new maps collected up to time k is defined as

$$\mathscr{M}(k) = \{M_k\},\ k = 0, \ldots, K.$$

M_0 is the initial map, obtained by the SLAM procedure. The goal of the developed algorithm is to provide for each robot an estimate $\hat{M}_k$ of the map at each time step k. In order to take advantage of the multi-robot scenario these updated maps must be shared with the other robots, and this information has to be merged in order to create a map that is a good estimate of the current state of the environment.

Correct map merging is not sufficient; a coordination strategy of the team of robots it also needed to maximize the number of detected variations, balancing at the same the number of Δ*-mapping* processes among the robots.

3 The Approach

The guidelines of the proposed approach are described hereafter, whereas details about the specific processes of variations awareness, local Δ*-mapping* and map merging are given in Subsection 3.1.

In the proposed Δ*-mapping* approach the concept of *time-map* is introduced to merge properly the changes detected locally by a robot and the updated maps received from the other team members.

In a grid map each cell represents the belief on the occupation value of the corresponding area. Since the environment changes over time, the reliability of the stored

value for the cells decreases over time. Therefore to each cell in the map a value in the range $[0-1]$ is assigned, related to the time passed since the cell has been visited for the last time. The set of these values at each time step is called *time-map* and defined as T_t.

The outline of the Δ-*mapping* algorithm, which runs on board of each rover, is described in Algorithm 1.

Input: $\hat{M}_{k-1}$, T_{t-1}, p, l, P, L
Output: $\hat{M}_k$, T_t
T_t = **updateTimeMap**(T_{t-1}, p, l);
if *received map* $\hat{M}'$ T' **then**
    $[\hat{M}'_{k-1}, T_t]$ = **mergeMap**($\hat{M}_{k-1}$, T_t, $\hat{M}'$, T');
    $\hat{M}_{k-1} = \hat{M}'_{k-1}$;
end
if $\Delta -$ *awareness* **then**
    $P = P + p$;
    $L = L + l$;
else
    if $P! = \emptyset$ **then**
        $[\hat{M}_k]$ = **updateMap**($\hat{M}_{k-1}$, P, L);
        $P = \emptyset$;
        $L = \emptyset$;
        dispatchUpdatedMap($\hat{M}_k$, T_t);
    end
end

Algorithm 1. The Δ-mapping algorithm

The algorithm takes as inputs the previous map $\hat{M}_{k-1}$ and the time-map T_{t-1}. p and l are the current robot pose and the current laser range reading respectively, P and L are two matrices collecting the values of p and l

$$P = \begin{bmatrix} \hat{x}^1, \hat{y}^1, \hat{\theta}^1 \\ \vdots \\ \hat{x}^n, \hat{y}^n, \hat{\theta}^n \end{bmatrix} \tag{1}$$

$$L = \begin{bmatrix} l^1 \\ \vdots \\ l^n \end{bmatrix} \tag{2}$$

where the n-th entry is the last element stored. These matrices are used to create a local Δ-map containing the changes in the environment detected by the robot.

The time-map T_t is updated every time a laser scan is available to the robot; a ray tracing procedure is applied for each angle of the scan, assigning a maximum value

equal to 1 to every cell crossed by a ray. At each time step all the values in T_t are updated according to

$$T_t(i,j) = T_{t-1}(i,j) \cdot \left(1 - \frac{\Delta t}{C_t}\right) \tag{3}$$

where Δt is the time elapsed from the last update of T_t, and C_t is a time constant which defines the forgetting speed.

The time-map update depends only on Δt, therefore a common timebase among the team members is not required, avoiding the need of synchronization techniques over the net.

The algorithm is divided in two parts. The first part (lines 2-4) is performed only when the robot receives a map from another robot member of the team, while the second part (lines 6-15) is performed only if a variation in the environment has been detected.

If a robot receives a new map $\hat{M}'$ and the relative time-map T', it updates the state of its map and its time-map by merging them with $\hat{M}_{k-1}$ and T_t respectively (line 3). At this point the resulting map contains the modifications perceived by the other robots (line 4).

If a modification is detected by the Δ-*awareness* block, recalled in Section 3.1, the algorithm stores the current robot pose and the relative laser range reading (lines 7-8).

If the Δ-*awareness* block does not detect any modification and P and L are not empty, a local Δ-*mapping* is performed, following the approach recalled in subsection 3.1 (line 10). The content of these vectors is used to create a local Δ-map $\Delta\hat{M}$, then $\Delta\hat{M}$ is aligned and merged with the old map $\hat{M}_{k-1}$, to obtain an updated map $\hat{M}_k$.

Finally the resulting map $\hat{M}_k$ and the current time-map T_t are dispatched to the other team members.

3.1 Δ-Awareness, Local Δ-Mapping and Map Merging

In [3] the authors presented a single-robot approach that maintains an updated grid map of a dynamic environment, assuming an initial occupancy grid map available. The algorithm detects persistent variations in the environment and merges them with the previous map by using limited computational resources. It is composed by four blocks as shown in Figure 1.

The Δ-*awareness* block detects persistent variations in the environment, using a technique called *weighted recency averaging*, normally applied in tracking non-stationary processes.

In this setting, the weighted recency averaging recognizes changes in the environment, under the hypothesis that the robot is correctly localized and never kidnapped.

The purpose of the *Store Scan* block is to select the laser scan readings suitable for building the local updated sub maps. These readings are stored in L with the corresponding robot poses stored in P.

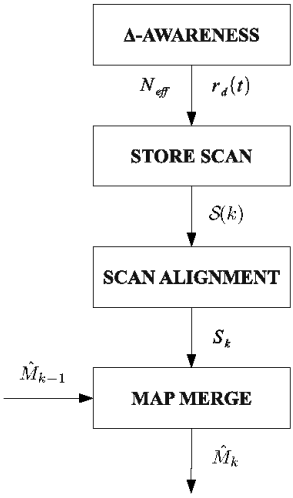


Fig. 1 The local Δ-mapping architecture

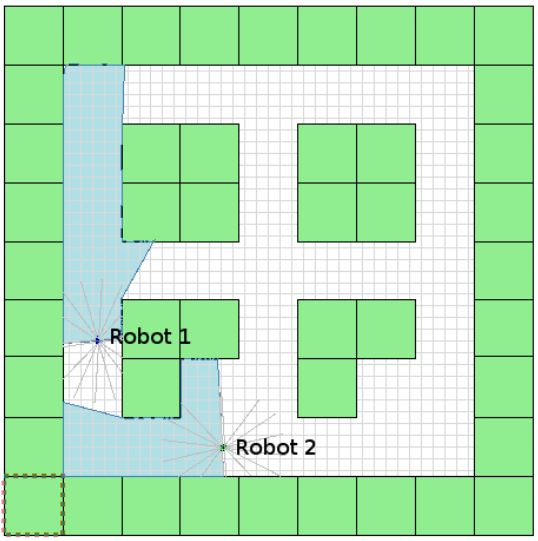
The *Scan Alignment* block produces a Δ-map performing a consistent registration of the collection of scan readings contained in L. The approach maintains all the local frames of data as well as the relative spatial relationships between local frames, modeled as random variables and derived from matching pairwise scans or from rover poses stored in P.

The *Map Merge* block merges the output of the *Scan Alignment* block with the map $\hat{M}_{k-1}$. The goal of this block is to find a rigid transformation that overlaps Δ-map and $\hat{M}_{k-1}$, to create the current environment occupancy map $\hat{M}_k$. We adopted the algorithm proposed in [6], based on Discretized Hough transform and bidimensional correlation. The Discretized Hough transform finds the rotation that aligns Δ-map with $\hat{M}_{k-1}$, then the bidimensional correlation is applied to compute the translation that overlaps the two maps.

Local Δ*-mapping* in this work is the application of the *Scan Alignment* and *Map Merge* blocks.

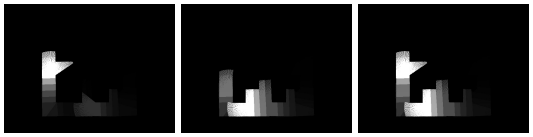
In the *updateTimeMap* function in line 3 of Algorithm 1 the current maps $\hat{M}_{k-1}$ and T_t are updated according to M' and T' received from the other robots. For all the couples i, j every cell $\hat{M}_{k-1}(i, j)$ is updated if its value is older than the corresponding cell $\hat{M}'(i, j)$, so that the most recent (hence reliable) value is used. The information about the reliability is given by the time-maps T_t and T'.

When a robot receives a new map $\hat{M}'$ and a time-map T' from another robot, it merges the received time map with the previous map $\hat{M}_{k-1}$ and the local time-map T_t in order to produce $\hat{M}'_{k-1}$. T_t is also updated. For all the couples i, j the value of the cell $\hat{M}'_{k-1}(i, j)$ is set equal to the cell $\hat{M}'(i, j)$ if $T'(i, j) > T_t(i, j)$, otherwise it is set equal to $\hat{M}_{k-1}(i, j)$. The value of the cell $T_t(i, j)$ is set equal to $T'(i, j)$ if



(a) Environment state and robots pose

(b) $\hat{M}_{k-1}$ (c) M' (d) $\hat{M}'_{k-1}$



(e) T_t (f) T' (g) T_t

Fig. 2 Figures show the map merging in a typical case: 2(a) shows the pose of the robots, Robot 1 receives a map from Robot 2 and it uses it to update its map; 2(b) is the current map, 2(e) is the current time-map, 2(c) is the received map, 2(f) is the received time-map, 2(d) and 2(g) are the resulting map and time-map after the merging process.

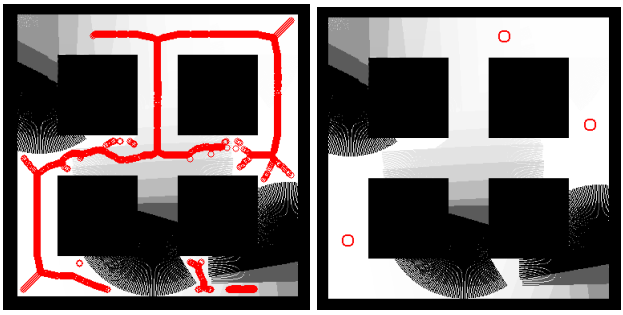
$T'(i,j) > T_t(i,j)$, otherwise it is not modified. Figure 2 shows the map merging process in a typical case. It can be noticed that changes received from another robot and local changes detected by the local Δ-*mapping* are merged in a consistent way. Cells belonging to areas that have been recently mapped have high corresponding time-map values (close to 1), so recent changes in the map resulting from a local Δ-*mapping* process are not discarded.

4 Coverage Strategy

A team coordination strategy that actively searches modifications in the map has been developed. Without any coordination strategy all the robots could follow the same path or leave some areas not visited for a long time. This problem can be treated in partial similarity with the problem of multi-robot exploration. In the exploration approaches the aim is to discover a map starting from a completely unknown environment. In the case considered, the initial map is known, as well as the robot pose, but since the environment is persistently changing (pallets are added and removed), the reliability of the initial map decreases over time on the basis of the number of changes in the environment. For this reason, areas that have not been recently visited may become completely unknown, as the reliability of the map in those areas is very low.

Areas that need to be covered are the ones for which the corresponding value of the time-map is below a given threshold. For each robot, a set of points is extracted to feed the path planning algorithms from a topological map, which is constructed from the grid-map representing the areas to be visited.

Many approaches obtain a topological representation from a grid-map, such as Voronoi diagrams [15] or topological operations [7]. The skeleton of an image is a good representation of the geometrical and topological properties of its shape, hence a morphological skeleton representation of the map is extracted using the algorithm described in [12], which is proven to be fast. Aset of points belonging to the skeleton is identified, with the constraint that each point has to be at a minimum distance from every other point. Each point becomes one goal point for the wavefront algorithm [11]. These goal points are then allocated to the team members by a distributed market-based task allocation algorithm described in the following subsection. Figure 3 shows how the goal points are obtained. In the time-map the black cells have the highest reliability and the white ones have the lowest reliability. In Figure 3(a) red points belong to the skeleton of the areas with reliability below a given threshold. In this case the team is composed by three robots, so three goal points are obtained as indicated in Figure 3(b).



(a) (b)

Fig. 3 Time-map and skeleton of areas to cover (a) and final goal points for three robots (b)

4.1 Distributed Auction-Based Task Allocation

Each goal point generated by the coverage strategy must be efficiently assigned to one of the robots in order to minimize travel time.

The *Hungarian method* performs a combinatorial optimization to solve the assignment problem in polynomial time. It guarantees the optimal solution, but it is a centralized algorithm, that requires a supervisor node and a matrix containing a row for each robot and a column for each task. Each cell contains the cost for the relative task. Moreover this approach requires the ability for all the robots to communicate, but this condition is not assured due to unreliable WiFi communication.

The used approach is then based on auctions, and it has been developed starting from the one proposed in [14]. Every goal point is assigned to an auction over a multicast network channel; the robots that receive the auction compute and send back a bid. The auctioneer assigns the task to the robot with the best bid. The bid is computed according both to the robot's current position and to its queue of pending tasks. This approach does not guarantee the optimal solution, but it is robust to communication failures. The auctioneer is always a different robot, thus avoiding the problem of single point of failure.

5 Simulation Tests

The simulated environment of a logistic area already used in [3] and shown in Figure 4 is considered. The occupied green areas can be thought as containers or similar items stored before distribution. The environment is 35×35 m, the green areas in the center are 10×10 m and the corridors are 5 m wide.

$n = 3$ rovers are endowed with wheel encoders, a laser range finder and a WiFi board, and are able to localize themselves in the given environment. It is assumed that, once the rovers are correctly localized, a virtual fork-lift adds or removes one container every minute.

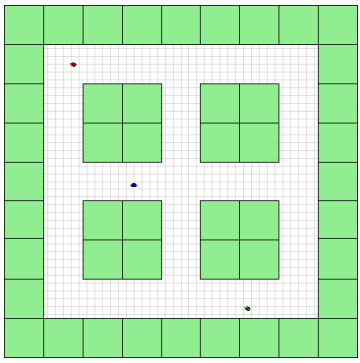


Fig. 4 The simulation environment

The rovers start moving with a simple obstacle avoidance policy. When the Δ-*mapping* process starts, the rovers move according to the coverage strategy described in Section 4. The quality of the map over time and the localization error are measured. The error on the estimate of the robot pose is strictly related to the quality of the map. Every Δ-*mapping* process induces some degradation of the map, due to the localization error which cannot be fully compensated by the *Map Merge* block.

Even after a consistent number of changes in the environment the rovers keep a map that is consistent with the environment and therefore the localization error remains low.

5.1 Simulation Test 1

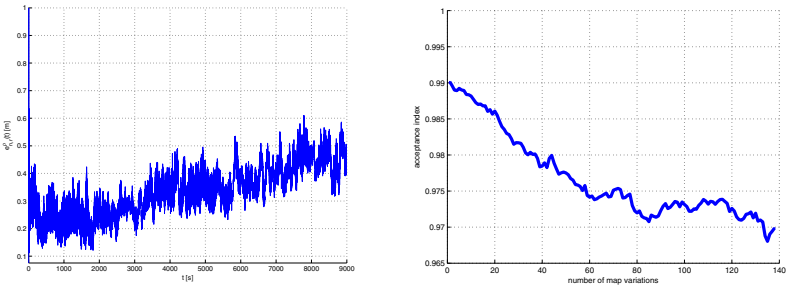
To demonstrate the effectiveness of the proposed approach first results related to $r = 10$ averaged runs are provided, where the Δ-*mapping* updating process lasts for approximately two hours each run.
The localization error of the i-th robot is defined as the distance between the ground-truth Cartesian position $(x_i^{gt}(t), y_i^{gt}(t))$ and its Cartesian position estimation as

$$e_i^{\rho}(t) = \sqrt{(x_i^{gt}(t) - \hat{x}_i(t))^2 + (y_i^{gt}(t) - \hat{y}_i(t))^2}. \tag{4}$$

We then define the average localization error for n robots over r runs as

$$e_{n,r}^{\rho}(t) = \frac{1}{r} \sum_{j=1}^{r} \sum_{i=1}^{n} \frac{e_i^{\rho}(t)}{n} \tag{5}$$

The localization error is reported in Figure 5(a). It can be noticed that the mean localization error remains lower than 0.6 m after approximately 2.5 hours. The quality of the map for the duration of the test is also inspected. Visual inspection is often



(a) (b)

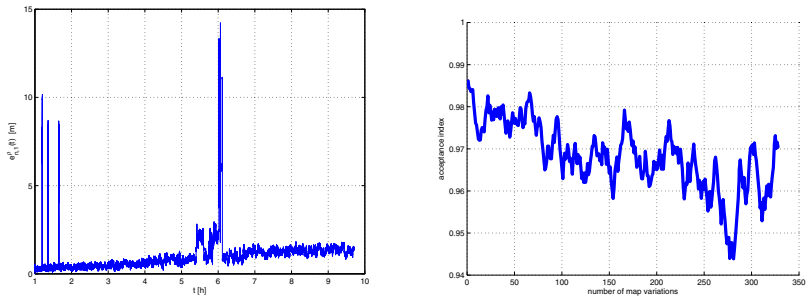
Fig. 5 Localization error and acceptance index for test 1

used, and numerical results by using the acceptance index described in [6] are also provided. They can be used as a measure of similarity between the real map and the estimated map.

Figure 5(b) shows the acceptance index mediate over the $n = 3$ robots and over $r = 10$ runs. After 140 variations the value obtained is 0.97, which is comparable with the one obtained with a typical grid-based SLAM algorithm (0.98).

5.2 Simulation Test 2

Here the performances of the Δ-*mapping* process in long term operativity are tested. The simulation scenario is the same as for the previous test, but the virtual fork-lift adds and removes containers every two minutes. In this test the map updating process lasts for approximately 9.5 hours, for a total number of 328 variations. Figure 6(a) shows the localization error for a single run, while Figure 6(b) shows the acceptance index over 328 variations. The sudden increase of the localization error after approximately 6 hours is due to one of the robots losing its localization for a short period of time. However as the robot receives an updated map it is able to recover itself. After 328 variations the acceptance index is still comparable with the one obtained in the previous test (see figure 5.1). Moreover, this acceptance index decreases to 0.97 after 9.5 hours, while in [3] the same error occurs after only 6 hours.



(a) (b)

Fig. 6 Localization error and acceptance index for test 2

5.3 Simulation Test 3

In this test the $n = 3$ robots perform different actions. The first robot performs Δ-*mapping* and sends map variations to the others team members; the second one only receives map variations but does not perform Δ-*mapping*; the last one neither perform Δ-*mapping* nor receives changes from the other team members. This test demonstrates the advantage in receiving map updates from other robots.

Figure 7 shows the localization error $e^p(t)$ for the three robots during a single run. Robot 1 remains well localized, while for robot 3 the error increases after approximately 3800 seconds; localization error for robot 2 starts to increase after 4720 seconds. This is due to the fact that robot 2 is able to merge the map updates received from robot 1, but this is still not sufficient in order to maintain a consistent map of the environment.

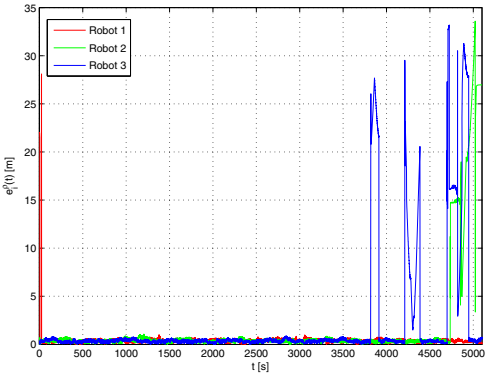


Fig. 7 Localization error for test 3

5.4 Computational Load

In Figure 8 the CPU usage and memory occupation for each robot are reported. The algorithm runs on an Intel Core 2 Duo 2.4 Ghz with 2 GB of RAM. After approximately one minute the simulated fork lift starts to remove and add pallets, and the Δ-*mapping* process starts. The peaks in CPU usage and memory occupation

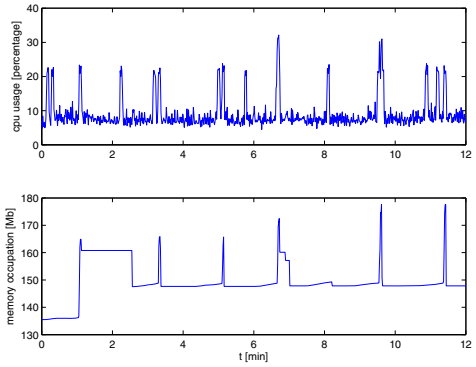


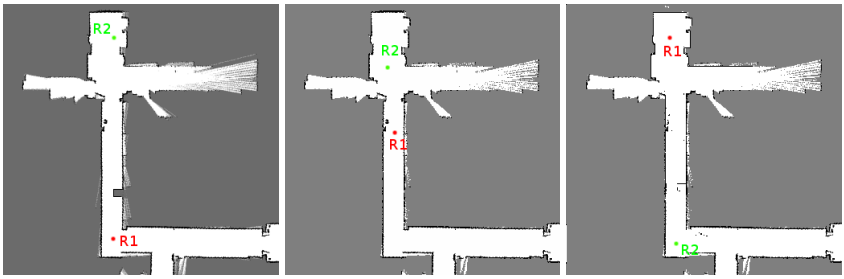
Fig. 8 CPU usage (upper plot) and memory usage (lower plot) in a simulated experiment

refer to the end of each local *Δ-mapping*, while the peaks in CPU usage only refer to the computation and assignment of the points to be visited. It can be noticed that after the beginning of the *Δ-mapping* process the memory usage steadily increases by only 12 MB, with peaks corresponding to the last phase of each local *Δ-mapping* process.

6 Experimental Tests

An experiment in a real environment using two Pioneer P3DX robots has been carried out. Each robot is endowed with a SICK LMS200 laser rangefinder and a WiFi board. A 1×1 m box has been placed in a 30×3 m corridor, and a classical Rao-Blackwellized SLAM process is first performed to obtain the map of the environment, as shown in Figure 9 (a). Then, the box is removed, R1 detects the absence of the box while travelling in the corridor, performs a *Δ-mapping* process and dispatches the map to R2, which updates its map (see Figure 9 (b)). Finally, the box is placed again in the previous place and R2 detects the presence of the box while travelling in the corridor, performs a *Δ-mapping* process and dispatches the map to R1, which updates its map (see Figure 9 (c)). It is worth noting that maps in Figure 9 (a),(b),(c) are the same for R1 and R2, even if they have not all perceived the same variations at the same time.

This preliminary test demonstrates the effectiveness of the proposed methodology in a simple but real scenario, since robots are able to merge the received maps from team members in a consistent way.



(a) Initial map (b) Dispatched map after the first change (c) Dispatched map after the second change

Fig. 9 The maps obtained during the experimental test

7 Conclusions

In this work a methodology which is able to perform map updating in multi-robot applications dealing with dynamic environments is proposed. This methodology enables robots to detect variations in an environment, to generate an updated map

containing only the persistent variations, to send this map to the other team members and to merge received maps in a consistent way. The approach is suitable for applications such as logistic applications, where a long-term operativity is required and the algorithm has to be computationally light and to use limited memory, in order to allow concurrent execution of other higher level services. Future works will be devoted to extensive experimental tests in real environments and to improvements of the coordination strategy.

References

1. Abrate, F., Bona, B., Indri, M., Rosa, S., Tibaldi, F.: Switching multirobot collaborative localization in symmetrical environments. In: IEEE International Conference on Intelligent Robots Systems (IROS 2008), 2nd Workshop on Planning, Perception and Navigation for Intelligent Vehicles, PPNIV (2008)
2. Abrate, F., Bona, B., Indri, M., Rosa, S., Tibaldi, F.: Three state multirobot collaborative localization in symmetrical environments. In: Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions, pp. 1–6 (2009)
3. Abrate, F., Bona, B., Indri, M., Rosa, S., Tibaldi, F.: Map updating in dynamic environments. In: Proceedings of the 41st International Symposium on Robotics, pp. 296–303 (2010)
4. Birk, A., Carpin, S.: Merging occupancy grid maps from multiple robots. Proceedings of the IEEE 94(7), 1384–1397 (2006), doi:10.1109/JPROC.2006.876965
5. Carlone, L., Ng, M.K., Du, J., Bona, B., Indri, M.: Rao-blackwellized particle filters multi robot slam with unknown initial correspondences and limited communication. In: Proceedings of IEEE International Conference on Robotics and Automation (2010)
6. Carpin, S.: Fast and accurate map merging for multi-robot systems. Auton. Robots 25(3), 305–316 (2008), doi:http://dx.doi.org/10.1007/s10514-008-9097-4
7. Fabrizi, E., Saffiotti, A.: Extracting topology-based maps from gridmaps. In: IEEE Intl. Conf. on Robotics and Automation (ICRA), pp. 2972–2978 (2000)
8. Fenwick, J., Newman, P., Leonard, J.: Cooperative concurrent mapping and localization, vol. 2, pp. 1810–1817 (2002), doi:10.1109/ROBOT.2002.1014804
9. Howard, A.: Multi-robot simultaneous localization and mapping using particle filters. In: Robotics: Science and Systems, pp. 201–208 (2005)
10. Howard, A., Sukhatme, G.S., Matarić, M.J.: Multi-robot mapping using manifold representations. Proceedings of the IEEE - Special Issue on Multi-Robot Systems 94(9), 1360–1369 (2006)
11. LaValle, S.: Planning Algorithms. Cambridge University Press (2004)
12. Maragos, P., Saffiotti, A.: Morphological skeleton representation and coding of binary images. IEEE Trans. on Acoustics, Speech, and Signal Processing (1986)
13. Siciliano, B., Khatib, O. (eds.): Springer Handbook of Robotics. Springer, Heidelberg (2008), http://dx.doi.org/10.1007/978-3-540-30301-5
14. Smith, R.G.: The contract net protocol: High-level communication and control in a distributed problem solver. IEEE Transactions on Computers C-29(12), 1104–1113 (1981)
15. Thrun, S., Bücken, A.: Integrating grid-based and topological maps for mobile robot navigation. In: Proc. of the National Conference on Artificial Intelligence (1996)
16. Williams, S., Dissanayake, G., Durrant-Whyte, H.: Towards multi-vehicle simultaneous localisation and mapping, pp. 2743–2748 (2002), doi:10.1109/ROBOT.2002.1013647
17. Zhou, X.S., Roumeliotis, S.I.: Multi-robot slam with unknown initial correspondence: The robot rendezvous case. In: Proceedings of IEEE International Conference on Intelligent Robots and Systems, pp. 1785–1792

Any-Com Multi-robot Path-Planning: Maximizing Collaboration for Variable Bandwidth

Michael Otte and Nikolaus Correll

Abstract. We identify a new class of algorithms for multi-robot problems called "Any-Com" and present the first algorithm belonging to that class: "Any-Com intermediate solution sharing" (or Any-Com ISS) for multi-robot path planning. Any-Com algorithms find a suboptimal solution quickly and then refine that solution subject to communication constraints. This is analogous to the "Any-Time" framework, in which a suboptimal solution is found quickly, and refined as time permits. The current paper focuses on the task of finding a coordinated set of collision-free paths for all robots in a common area. The computational load of calculating a solution is distributed among all robots, such that the robotic team becomes a distributed computer. Any-Com ISS is probabilistically/resolution complete and a particular robot contributes to the global solution as much as communication reliability permits. Any-Com ISS is "Centralized" in the planning-algorithmic sense that all robots are viewed as pieces of a composite robot; however, there is no dedicated leader and all robots have the same priority. Previous centralized multi-robot navigation algorithms make assumptions about communication topology and bandwidth that are often invalid in the real world. Any-Com allows for collaborative problem solving with graceful performance declines as communication deteriorates. Results are validated experimentally with a team of 5 robots.

1 Introduction

Autonomous navigation is a key capability for enabling both industrial and consumer robotics to perform their work effectively. In fact, many of today's state-of-the-art systems are being commercialized, and will become increasingly deployed into mainstream settings in the near future. As robot traffic becomes more

Michael Otte · Nikolaus Correll
University of Colorado at Boulder, Colorado, USA
e-mail: {michael.otte,nikolaus.correll}@colorado.edu

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 161–173.
springerlink.com

congested, tomorrow's systems must be capable of coordinated interaction within a multi-robot society. This imposes a need for multi-robot navigation solutions that can plan efficient, coordinated, and collision-free paths for a collection of robots.

Complete solutions to multi-robot problems can be computationally complex. Although less expensive methods can enable practical performance in many real-world situations, these are incomplete and can fail in the most challenging circumstances (see Section 2.1). Often, each robot in a team is equipped with its own computer and the ability to communicate. Given these resources, it makes sense to divide computational effort among all robots a solution will benefit. That is, a networked team of robots can be re-cast as a distributed computer to solve the problems encountered by its composite robots. This is particularly useful for complex communal tasks such as centralized multi-robot path-planning.

In practice, wireless bandwidth is environment dependent and often beyond the control of the user or a system. Yet, algorithms for coordinating networked robot systems usually rely on a minimum quality of service and fail otherwise. We are therefore interested in distributed algorithms able to utilize unreliable communication, and coin the term "Any-Com" to describe them. The idea is to find a suboptimal solution quickly, and then refine toward optimality as communication permits. This is analogous to the "Any-Time" paradigm, in which algorithms adapt to the available computation *time* (Boddy and Dean, 1989). In this paper we present an algorithm called *Any-Com Intermediate Solution Sharing* (or Any-Com ISS) for performing centralized multi-robot path-planning within the Any-Com framework. In previous work, centralized solutions have either been calculated on a single robot and then disseminated, or solved by each robot individually (see Section 2.2).

In general, Any-Com algorithms exploit perfect communication and have gracefully performance declines otherwise. However, just as Any-Time algorithms cannot calculate a solution in 0 time, Any-Com ISS may not find a solution when communication totally fails. Worst-case scenarios aside, Any-Com ISS is robust to a high degree of communication disruption.

A brief survey of related work is presented in Section 2. Algorithmic details are provided in Section 3. In Section 4 we conduct a series of experiments both in simulation and on real robots. In Section 5 we discuss our results, and conclusions are given in Section 6.

2 Related Work

Here we briefly discuss a few multi-robot algorithms located along the communication, computation, and completeness spectra. Recall that a *complete* algorithm is guaranteed to find a solution when one exists and will also report failure in finite time if a solution does not exist. A *resolution complete* algorithm is an algorithm that is complete to within a predefined granularity of the world representation. A *probabilistically complete* algorithm is an algorithm that will find a solution, if one exists, in finite time with probability approaching 1.

2.1 Incomplete Methods

In the *cocktail party* model each agent maintain its own world-view, goals, and navigation function, while remaining ignorant of other robots and their intentions (Lumelsky and Harinarayan, 1997; van den Berg et al, 2009). Each agent alternates sensing, planning, and movement, and there is no direct coordination between robots. While this algorithm is incomplete, it is popular due to simplicity, scalability, and minimal communication requirements.

In *prioritized planning* each robot's path is calculated separately, subject to the movement constraints imposed by the paths of higher-priority robots (Erdmann and Lozano-Perez, 1987; Warren, 1990; Hada and Takasa, 2001; Clark and Rock, 2001). Higher priority robots follow optimal to near-optimal trajectories while lower priority robots may be unable to find a solution. Prioritized planning has also been used to periodically create a line-of-sight communication chain while performing the somewhat related coverage task (Hollinger and Singh, 2010).

Decoupled planning breaks planning into two phases. In phase-1 each robot calculates its own path to the goal. In phase-2 the space-time positions of the robots along these paths are calculated such that no collisions occur (Kant and Zuker, 1986; Aronov et al, 1998; Leroy et al, 1999; Guo and Parker, 2002). Although decoupled planning can be distance-optimal, it is incomplete because each robot's path is completely determined after phase-1 (and they may pathologically conflict) (Sanchez and Latombe, 2002).

2.2 Complete Methods: Centralized Planning

In *Centralized planning* all robots are considered individual pieces of a single composite robot. Solutions are calculated in the resulting high dimensional configuration space. Robot paths are found by projecting the high-dimensional solution down into the relevant subspace per each robot. (Xidias and Aspragathos, 2008; Bonert, 1999; Schwartz and Sharir, 1985; Clark et al, 2003; Sanchez and Latombe, 2002). Previously, the high-dimensional solution has either been calculated by a single agent or at the same time on each robot (thus robots must communicate with this agent or each other, respectively). Centralized planning is theoretically complete but practical algorithms are usually probabilistically or resolution complete; nonetheless, it provides the best completeness guarantees of any multi-robot planning method.

2.3 Relevance to Our Work

Our Any-Com ISS algorithm (presented in Section 3) is centralized, and therefore shares many similarities to the work described above. One major difference is that previous work has not considered what happens when communication deteriorates—this is a main contribution of our work. Another important difference is that our algorithm leverages the distributed-computing power of the robotic team

to help find better solutions more quickly. In contrast, previous work has required *each* agent to calculate an entire solution completely on its own.

Distributed versions of both prioritized planning and decoupled planning exist. For instance, in prioritized planning each robot can calculate its own path (assuming it respects robots of higher priority), and in decoupled planning each robot can individually calculate its own phase-1 solution (although these must be assembled by a single agent in phase-2). However, both prioritized planning and decoupled planning are incomplete, while Any-Com ISS is probabilistically/resolution complete.

We believe Any-Com ISS is most applicable to the complicated planning situations in which the incomplete planning methods fail, and advocate using the (less computationally complex) incomplete ideas under most circumstances. For this reason we only compare Any-Com ISS to state-of-the-art *centralized* planning techniques in Section 4—as these are the only other algorithms available when incomplete methods fail.

3 Methodology

Let the robot workspace $\mathbf{W}$ exist in $\mathbb{R}^2$. To guarantee probabilistic/resolution completeness, the entire team is considered a single composite robot. Each individual robot contributes 2 dimensions to the combined configuration space $\mathbf{C}$, in the form of position (x, y), and search occurs in a $\mathbb{R}^{2n}$ configuration space where n is the number of robots. We assume resolution accuracy δ is defined for the configuration state vector. δ is the minimum distance allowable between any two configurations per dimension and thus defines the resolution of the search. In a pragmatic sense, δ keeps the search-tree from being populated with essentially duplicate configurations, and focuses effort on finding (significantly) better solutions. We assume circular robots that can pivot in place, but note our algorithms can be generalized to arbitrary robots.

We use a heavily modified version of an any-time rapidly expanding random tree (RRT) inspired by Ferguson and Stentz (2006). Our underlying RRT differs from previous work (LaValle and Keffner, 2001) in two significant ways. First, instead of connecting a new node to the tree using the shortest possible edge, we use the edge that gives the new node the shortest possible distance-to-root. Second, instead of restarting each subsequent tree from scratch (i.e. while time remains to find a better solution), we prune the existing tree such that it only contains nodes that can possibly lead to better solutions—then continue growing the same tree subject to the constraint that new nodes must be able to lead to better solutions.

In general, we seek to utilize the distributed computational power of a team of mobile robots. We want algorithms that function in environments where communication is unreliable, but take advantage of reliable communication when it exists. To these ends, each agent maintains its own randomly created tree. Assuming n robots, the union of all trees is a $O(n)$ times larger tree maintained collectively by the entire team. Any-com is achieved by having robots share their individual intermediate solutions *during* path-planning so that all agents can prune globally sub-optimal

branches from their local trees. This enables each robot to focus effort on finding only better solutions than those currently known to *any* robot. It also gives all robots a chance to directly refine the best intermediate solution. We call this idea Any-Com *Intermediate Solution Sharing* (Any-Com ISS).

Theoretically, allowing more agents to work on a random-tree problem will increase the chances a good solution is found quickly, regardless of whether or not the sharing of intermediate solutions has any affect. Therefore, to determine how much (if any) advantage Any-Com ISS provides, we compare Intermediate Solution Sharing to having *each* agent individually find a unique solution to the complete problem, then broadcasting them so the team can use the best one. We refer to the latter method as *Voting*, and note that similar ideas have been explored in the past (Clark et al, 2003). Finally, to give context to the relative performance of Any-Com ISS vs. Voting, we compare both of them to a client-server framework. In the client-server system, which we call *Baseline*, the server is charged with calculating a complete solution using a single random tree, and then sharing it with the other robots. Any-Com ISS, Voting, and Baseline use the same underlying random tree algorithm (Figure 1-Left). To demonstrate that it performs well vs. previous work, we additionally compare results to Any-Time RRT (Ferguson and Stentz (2006)).

We assume the existence of an admissible heuristic function $h(p_1, p_2)$ that returns the distance between configurations p_1 and p_2 ignoring any collisions. The value *bstln* stores the length of the shortest path known at any particular time. On line 4 we pick a new configuration p_1 to add to the tree—chosen as the goal with probability ρ and uniformly at random otherwise. On line 5, we check both if p_1 exists in $\mathbf{C}_{free}$, the collision free portion of the configuration space, and also if using p_1 can possibly lead to a better solution based on the start and goal configurations and *bstln*. Note that $\mathbf{C}_{free}$ is calculated with respect to both robot-robot collisions and robot-obstacle collisions. On line 7 we find the best node p_2 in the tree to use as a parent of p_1. We record $S_{dist}(p_1)$, the actual distance-to-start of p_1 through p_2, and then add p_1 to the tree on lines 10 and 11. If p_1 is the goal (and $p_2 \neq$ **null** on line 8) then the new path-to-goal is the best intermediate solution found so far, so we update *bstln* on line 13. On line 14 we use the function **findShortcuts**() to see if other nodes in the tree can reach the start more quickly via p_1 instead of their current parent. If so, we change the tree to reflect this, and update S_{dist} values of the descendants.

Lines 16-20 are only executed when Any-Com ISS is used. On line 16 we check for incoming messages from other agents that may contain better paths. If a better path is received, then it is added to the search-tree and *bstln* is updated (lines 18 and 19). Finally, we send messages to other agents on line 20.

While searching for p_1 in **findBestAndPruneTree**() (Figure 1-Right-Top) we simultaneously prune any nodes that cannot possibly lead to solutions shorter than *bstln*, and also check if p_1 is more than δ away from configurations already in the tree. Keeping the tree as small as possible focuses effort on finding better solutions.

Search continues until time μ, after which the most recent (and therefore best) intermediate solution is recorded as an agent's final solution. In Baseline, this is when the server distributes its final solution to the client robots, and also when individual solutions are compared in Voting.

RandomTree()

```
1:  bstln = ∞
2:  add start as root of search-tree
3:  while time < μ do
4:      pick a point p1 ∈ C, where
        p1 = goal with probability ρ
5:      if p1 ∉ C_free
        or h(start, p1) + h(p1, goal) ≥ bstln
        then
6:          continue
7:      p2 = findBestAndPruneTree(p1)
8:      if p2 = null then
9:          continue
10:     S_dist(p1) = S_dist(p2) + h(p1, p2)
11:     add p1 to search-tree as a child of p2

12:     if p1 = goal then
13:         bstln = S_dist(p1)
14:     FindShortcuts(p1)
15:     if using Any-Com ISS then
16:         check for messages at rate ω
17:         if received better path then
18:             add that path to search-tree
19:             update bstln
20:         send message with best-path
```

p_2 = **findBestAndPruneTree**(p_1)

```
1:  p2 = null
2:  g_p2 = bstln
3:  for each node p_i ∈ Tree do
4:      if S_dist(p_i) + h(p_i, goal) > bstln
        then
5:          remove p_i
6:      else if p1 is within δ of p2 then
7:          return null
8:      if S_dist(P_i) + h(p_i, p1) < g_p2 then
9:          if edge (p_i,p1) ∈ C_free then
10:             p2 = p_i
11:             g_p2 = S_dist(P_i) + h(p_i, p1)
12: return p2
```

FindShortcuts(p_1)

```
1:  for each node p_i ∈ Tree do
2:      if S_dist(p_i) + h(p_i, p1) < S_dist(p1)
        and edge (p_i,p1) ∈ C_free then
3:          S_dist(p1) = S_dist(p_i) + h(p_i, p1)
4:          reroute p_i through p1
5:          for descendants of p1 do
6:              update S_dist()
```

Fig. 1 Random tree algorithm with flags indicating functionality native to Any-Com ISS (Left). Subroutine for finding p_2 (the best neighbor of p_1 already in the tree) and pruning the tree (Top-Right). Subroutine for checking if old nodes would do better by using p_2 as their parent (Bottom-Right).

We hypothesize Intermediate Solution Sharing will produce better solutions than the other two methods because it allows the entire team to have tighter search-tree pruning—focusing search toward new and improved solutions. Additionally, Any-Com ISS gives each agent the opportunity to improve the best solution found so far. Any-Com ISS is robust to packet loss because dropped messages do not affect an agent's ability to eventually find a solution. On the other hand, successful communication focuses search in beneficial ways and helps the team find better solutions more quickly. Even out-of-date messages have the potential to be beneficial, as long as the solution they contain is better than the receiving agent's current best.

Each search-tree is generated randomly and each solution is drawn from a distribution over all possible solutions. Theoretically, both Any-Com ISS and Voting should increase the team's collective chances of finding a desirable solution, vs. Baseline, because n random samples are drawn from this distribution instead of 1.

Both Any-Com ISS and Voting use the same underlying message-passing protocol to disseminate information within the group. The idea is simple: each robot broadcasts information to every other robot at a predefined rate ω using the User

Datagram Protocol (UDP). UDP drops unsuccessful messages, which keeps the information flowing through the network up-to-date. Each message contains the following information about the state of the global solution, based on the sending robot's current knowledge:

- Best solution (currently known to the sender)
- Best solution's length
- ID of the robot that generated the best solution
- List of robots that have submitted a final solution
- Movement flag
- List of robots that support best solution.

Each robot keeps a copy of what it believes to be the best solution found by any robot. Each robot is responsible for adding itself to the appropriate lists. In order to keep the network up-to-date, messages are dropped if they contain paths that are worse than the best path known to the receiving agent. Planning halts after time μ, at which point robots begin adding themselves to the list of robots that have submitted a final solution. Any robot can correctly deduce which agreement has occurred if it knows all robots have submitted a final solution (regardless of algorithm). This is because better solutions are no longer being generated and the best solution known to the sending robot is always sent in every message—the actual best solution must have been passed along with the knowledge that the robot who generated it has submitted a final solution. In the unlikely event of a tie, the solution found by the robot with the lower ID is used. Once a robot knows an agreement has been reached, it sets the moving flag to TRUE, begins moving along its path per the best solution, and rebroadcasts the best solution at ω. If a robot receives a message with a TRUE movement flag, it also starts moving and rebroadcasts that solution at ω.

Baseline modifies the method described above by setting the movement flag to TRUE as soon as time μ occurs. Therefore, each robot begins moving as soon as the solution is received from the server. In order to keep Baseline as naive as possible, the client robots do not rebroadcast the solution to each other, but the server continues to rebroadcast at ω.

Any-Com ISS also has an additional method of reaching an agreement. By carefully tracking which partial solutions the other robots most recently support (during the planning phase), it is possible to approximately forecast the final vote at time μ. After time μ, if a particular robot believes all robots currently support its most recent solution, then it starts moving on that solution and rebroadcasts it at ω (along with the moving flag set to TRUE). This information is propagated through the network as usual (with disagreements broken toward solutions from robots with lower IDs). Although this protocol may allow a suboptimal solution to be chosen, it is unlikely. Further, if an agent erroneously believes *all* robots currently support its solution, then it must have had the best solution in the past, so the cost of erroneously picking a suboptimal solution is mitigated. A scenario where different robots move along different *incompatible* solutions is impossible because two or more robots cannot simultaneously believe all robots support their most recent solution. This is due to the fact that *only* the robot that generated a solution can initiate movement along it.

If two different robots generate competing solutions, neither will initiate movement until one robot advertises support for the other's solution—and they cannot both support the other's solution because one solution is guaranteed to be better than the other (or, in the case of ties, come from the robot with lower ID).

4 Experiments

We perform two experiments with 5 robots in an office environment. Experiment 1 is conducted in simulation to evaluate theoretical performance over a wide range of parameters. Experiment 2 uses real robots to validate that the algorithms function in practice. Our robotic platform is the iRobot create, and we use the ROS operating system by Willow Garage. Robots are equipped with the Stargazer Indoor Localization System. Our Computational Units are System 76 Netbooks with built-in wireless networking capabilities.

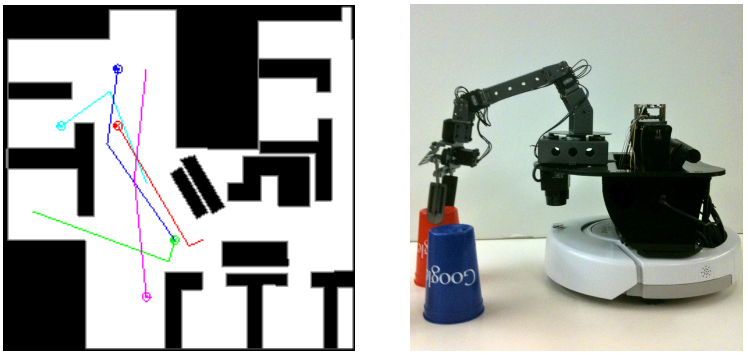


Fig. 2 A solution from Experiment 1 (Left). The Prairiedog Robotic Platform (Right).

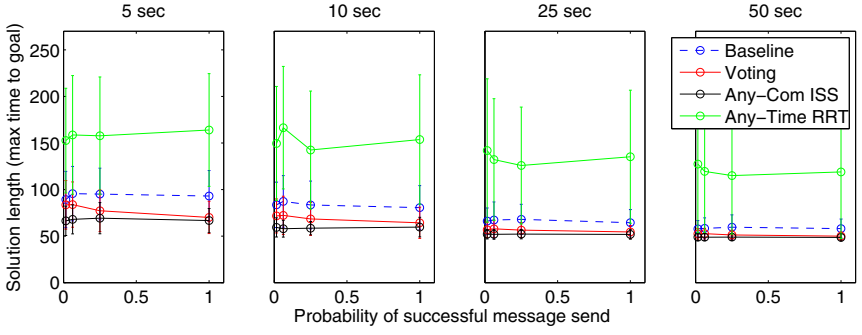


Fig. 3 Average Solution Lengths from Experiment 1. Sub-plots show different planning times μ.

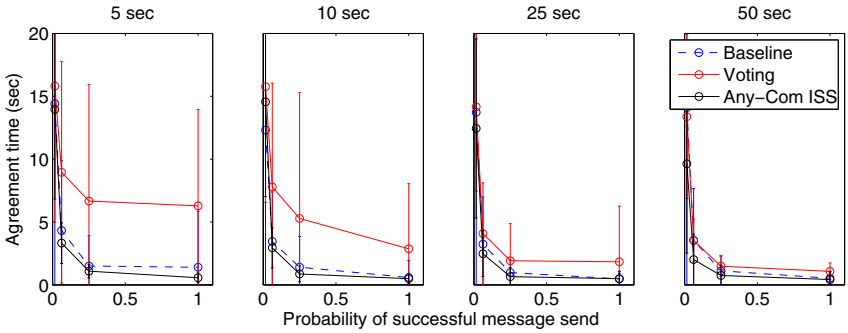


Fig. 4 Average agreement time from Experiment 1. Sub-plots show different planning times μ.

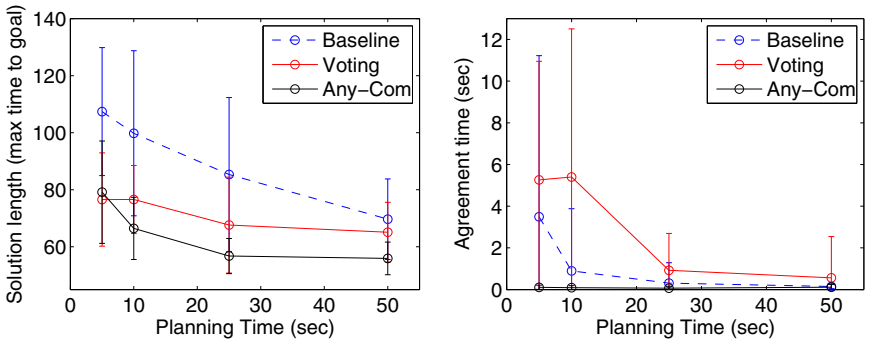


Fig. 5 Average Solution Lengths (Left) and average agreement time (Right) from Experiment 2

Experiment 1 evaluates the relative performance of Any-Com ISS, Voting, Baseline, and Any-Time RRT (Figure 2). Note that Any-Time RRT is run on a single robot. We evaluate performance of all four algorithms vs. message success probability τ vs. planning time μ. We use $\tau = \{1, 1/4, 1/16, 1/64\}$ probability of success and $\mu = \{5, 10, 25, 50\}$ seconds. We perform 100 runs per each combination of parameters to facilitate statistical analysis of results. Mean and standard deviations of the resulting solution lengths are displayed in Figure 3 and agreement times in Figure 4 (agreement time is the time after μ and before movement). Note that agreement times are not presented for Any-Time RRT, since no message passing is required.

Experiment 2 is conducted on 5 actual robots and is similar to Experiment 1. Robot speed is 0.2 meters per second. During planning $\omega = 4$, and during the agreement phase $\omega = 32$. The change is due to the preliminary results in Experiment 1, where it is clear that the agreement phase can become lengthy in terms of messages sent. Also, path-planning is computationally intensive while the agreement phase is not, and robots are able to spare additional resources to increase ω. The same μ are used as in Experiment 1. Each data-point represents 20 runs. We plot solution

quality and agreement time vs. planning time in Figure 5 Left and Right, respectively. Signal quality was relatively good in this experiment, the observed packet loss rate was less than 50%. We forgo comparison vs. Any-Time RRT due to the positive performance of the other methods in Experiment 1.

5 Discussion of Results

With regard to solution quality, both Any-Com ISS and Voting out-perform Baseline, and Any-Com ISS outperforms Voting. All three methods outperform Any-Time RRT. Using a two-sample Kolmogorov-Smirnov test, we compare algorithms based on solution lengths, and find statistically significant ($p < .05$) differences between any two algorithms for all but one method-parameter combinations in Experiment 1 (i.e. for one method vs. another with μ and τ held constant), and all but one parameter combination in Experiment 2 (Voting vs. Any-Com ISS at $\mu = 5$ sec). In fact, $p < 0.001$ for most data-points in either experiment. When the results from all experiments are considered together, p becomes vanishingly small. These results validate our original hypothesis.

Examining the solution quality vs. planning time for the various methods in Figures 3 and 5-Left illustrate just how well Any-Com ISS performs. Voting finds similar quality solutions using less than half the planning time as Baseline, on average, while Any-Com ISS finds similar quality solutions in $\leq 1/n$ of the time! This is strong evidence the robotic team is functioning as an effective distributed computer. Given we are using n times as much computational power, the expected ratio of required planning time is $1/n$. Therefore, the super-efficient observed value of $< 1/n$ in Experiment 2 is impressive, especially given the minimal data shared between agents. Whether or not this trend will continue for larger groups of robots is a question we hope to answer in future work.

It often takes longer than 5 seconds (or even 10) for an agent to find a solution. That is, $\mu = 5$ is not enough time to guarantee that all robots have found a solution. In such a case, after 5 seconds has passed, Any-Com ISS uses the best solution found by any robot so far, while Baseline must wait until the server finds its first solution, and Voting must wait until all robots have found a solution. This has two interesting affects. First, the agreement times of Baseline and Voting are greater than Any-Com ISS because all robots must wait until the server or all robots have found a solution, respectively, before an agreement can be reached. Second, by waiting extra time until n solutions exits, Voting has an increased chance of finding a "good" solution vs. Any-Com ISS. While this may initially seem desirable, we note that Any-Com ISS is able to start movement at the expected time, while the other algorithms suffer unexpected delays. We believe this is why the results for Voting and Any-Com ISS are similar for $\mu = 5$ sec in Experiment 2 (i.e. $p > .05$), and also why the agreement times for Voting are inflated for $\mu = 5$ and $\mu = 10$ in Experiment 1.

Another interesting trend is that Any-Com ISS solution quality does not get much worse as communication becomes unreliable. Theoretically, as $\tau \to 0$ the results of

Any-Com ISS will approach those of Voting. There is a hint of this in Experiment 1, where τ is controlled, especially for longer planning times. However, it appears communication must drastically deteriorate before Any-Com ISS begins to suffer. In fact, packet loss rates as high as 98% have little affect on solution quality.

The most noticeable effect of poor communication is an increase in the time it takes the robots to agree on a single solution. Assuming that communication failure is strictly Poisson-distributed, increasing the messaging rate ω during the agreement phase can mitigate the effects of communication deterioration (as we did in Experiment 2). In any case, the bandwidth will eventually become saturated, and further diminishing τ will eventually prevent an agreement from taking place within a useful time. Therefore, Any-Com ISS should not be used when $\tau \approx 0$. That said, it is impossible for *any* complete algorithm to function when $\tau \approx 0$. As a practical measure, the $\tau \approx 0$ case could be handled using a time-out. After which, robots start moving based on the best solutions known to them individually. Assuming on-board sensors exist, conflicts could then be resolved using the cocktail-party model. Although this 'worst-case-scenario' forces the algorithm to become incomplete until communication is resumed, it is arguably better than letting the team remain motionless forever. Further discussion on this idea is beyond the scope of this paper.

The simulated experiments predict Baseline should have similar agreement times to Any-Com ISS, while the real experiments show Any-Com ISS as the clear winner. The fact that these benefits do not extend to the Voting method (even for $\mu > 10$) suggests some other mechanism is responsible for the relatively quick agreement time of Any-Com ISS. We credit this improvement to the auxiliary vote-forecasting agreement method available to Any-Com ISS.

6 Conclusions

We coin the term "Any-Com" to describe algorithms that use multiple agents to collaboratively refine a solution toward optimality as communication permits. The motivation behind the general Any-Com idea is that distributed robots should adapt to use as much collaborative problem solving as communication quality permits. This is useful for solving computationally intensive problems, and especially well suited to problems with solutions of value to multiple agents. The problem domain of centralized multi-robot rover navigation has both of these qualities.

We present a practical Any-Com multi-robot path-planning algorithm called *Any-Com Intermediate Solution Sharing* (Any-Com ISS) in which agents share intermediate solutions so that the entire team can focus remaining effort on finding even better solutions. This works because it allows all robots to prune globally suboptimal branches from their local search trees based on the best solution known to any member of the team. It also gives each robot an opportunity to directly improve the best solution. Intermediate Solution Sharing is Any-Com because dropped messages do not prohibit a solution from eventually being found, while successful messages improve solution quality (both in overall path quality, and the time it takes

to reach an agreement). We envision Any-Com ISS as one tool among many in the multi-robot planning arsenal—useful in the specific case when a complete algorithm must be used (i.e. when a group of robots finds itself confronted with a difficult problem that cannot be solved by less expensive incomplete planning methods).

We perform 2 experiments using a team of $n = 5$ robots, and compare results to a basic server-client model as well as a voting method (in the server-client framework one agent plans and then distributes the solution to the other robots, while in voting each agent is allowed to plan separately and then the team uses the best solution found by any single agent). We find Any-Com ISS requires *less then* $1/n$ of the time required by the client-server framework to find a solution of similar quality, and less than $1/2$ the time required by the voting method, on average.

As bandwidth approaches 0 the solution quality of Any-Com ISS theoretically declines gracefully to that of the voting method, while both remain better than the server-client model. In fact, we find that communication loss as high as 98% has little affect on solution quality. Unfortunately, the time it takes to reach consensus approaches infinity as communication approach 0. This is not unexpected, as all complete algorithms are inherently vulnerable to *total* communication failure. Ignoring this worst-case-scenario, we find that Any-Com ISS is robust to a high degree of communication interference.

While this paper is a focused case-study on Any-Com applied to multi-robot navigation, we stress that the Any-Com idea is not limited to this particular domain. In particular, Any-Com ISS is applicable to any random-tree search through a metric space. We hope that the Any-Com concept will spread to other problems, and envision a world in which mobile robots dynamically take advantage all available computational resources to solve complex problems.

References

Aronov, B., de Berg, M., van der Stappen, A.F., Svestka, P., Vleugels, J.: Motion planning for multiple robots. In: Proceedings of the Fourteenth Annual Symposium on Computational Geometry, Minneapolis, USA, pp. 374–382 (1998)

Boddy, M., Dean, T.L.: Solving time-dependent planning problems. In: Proc. Eleventh International Joint Conference on Artificial Intelligence, pp. 979–984 (1989)

Bonert, M.: Motion planning for multi-robot assembly systems. MS dissertation, University of Toronto (1999)

Clark, C.M., Rock, S.: Randomized motion planning for groups of nonholonomic robots. In: Proc. International Symposium of Artificial Intelligence, Robotics and Automation in Space (2001)

Clark, C.M., Rock, S.M., Latombe, J.C.: Motion planning for multiple mobile robots using dynamic networks. In: Proc. IEEE International Conference on Robotics and Automation, pp. 4222–4227 (2003)

Erdmann, M., Lozano-Perez, T.: On multiple moving objects. Algorithmica, 477–521 (1987)

Ferguson, D., Stentz, A.: Anytime rrts. In: Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5369–5375 (2006)

Guo, Y., Parker, L.D.: A distributed and optimal motion planning approach for multiple mobile robots. In: Proc. IEEE International Conference on Robotics and Automation, pp. 2612–2619 (2002)

Hada, Y., Takasa, K.: Multiple mobile robot navigation using the indoor global positioning system (igps). In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, Hawaii, United States, pp. 1005–1010 (2001)

Hollinger, G., Singh, S.: Multi-robot coordination with periodic connectivity. In: Proc. IEEE International Conference on Robotics and Automation (2010)

Kant, K., Zuker, S.W.: Toward efficient trajectory planning: the path-velocity decomposition. The International Journal of Robotics Research 5, 72–89 (1986)

LaValle, S., Keffner, J.: Rapidly-exploring random trees: Progress and prospects. In: Algorithmic and Computational Robotics: New Directions, pp. 293–308 (2001)

Leroy, S., Laumond, J.P., Simeon, T.: Multiple path coordination for mobile robots: a geometric algorithm. In: Proc. International Conference on Artificial Intelligence (1999)

Lumelsky, V.J., Harinarayan, K.R.: Decentralized motion planning for multiple mobile robots: The cocktail party model. Autonomous Robots 4, 121–135 (1997)

Sanchez, G., Latombe, J.C.: Using a prm planner to compare centralized and decoupled planning for multi robot systems. In: Proc. IEEE International Conference on Robotics and Automation (2002)

Schwartz, J.T., Sharir, M.: On the piano mover's problem iii. coordinating the motion of several independent bodies: the special case of circular bodies amidst polygonal barriers. In: Proc. IEEE International Conference on Robotics and Automation, pp. 514–522 (1985)

van den Berg, J., Guy, S.J., Lin, M., Manocha, D.: Reciprocal n-body collision avoidance. In: Proc. International Symposium on Robotics Research (2009)

Warren, C.W.: Multiple robot path coordination using artificial potential fields. In: Proc. of IEEE International Conference on Robotics and Automation, Cincinnati, OH, pp. 500–505 (1990)

Xidias, E.K., Aspragathos, N.A.: Motion planning for multiple non-holonomic robots: a geometric approach. Robotica 26, 525–536 (2008)

An Improved Particle Swarm Optimization Method for Motion Planning of Multiple Robots

Ellips Masehian and Davoud Sedighizadeh

Abstract. Multi robot motion planning is a challenging problem in the robotics field due to its complexity and high computational costs induced by the number of robots. In this paper a new heuristic method is presented for solving this problem through a decentralized approach with global coordination. The method is based on a new improved variant of the Particle Swarm Optimization (PSO) metaheuristic, which serves as a global planner. Alternatively, for local planning and avoiding obstacles in narrow passages, the Probabilistic Roadmap Method (PRM) is employed. The global and local planners act sequentially until all robots reach their goals. The algorithm iteratively and simultaneously minimizes two main objectives, shortness and smoothness of the paths. The proposed algorithm is simulated and compared with the standard (basic) PSO, as well as the standard Probabilistic Roadmap methods. The experimental results show a meaningful advantage of the new method regarding computational time and path quality.

1 Introduction

The robot motion planning discipline experienced a boost specifically after the advent of the Configuration Space (C-space) notion by Lozano-Pérez and Wesley in the mid 70's [1]. While early motion planning algorithms were mainly developed for single robots, the multi robot motion planning problem remained untackled until a decade later, when the prioritization and coordination concepts were developed, as in [2].

The general single robot motion planning problem is defined as the problem of finding a collision-free path for a robot navigating among various obstacles, and is classified as a PSPACE-hard and NP-hard problem [3]. This complexity is further increased for multi-robot motion planning as the larger number of robots creates difficult problems regarding their coordination, cooperation and obstacle avoidance. Thus, as a challenging problem, the multi robot motion planning problem increasingly attracts the attention of roboticists and researchers.

Ellips Masehian · Davoud Sedighizadeh
Faculty of Engineering, Tarbiat Modares University, Tehran, Iran

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 175–188.
springerlink.com

The primary approaches for path planning of single and multiple robots were generally based on computational geometry and handled deterministic low-dimensional problems. These methods, also known as classic methods, are variations of a few general techniques: Roadmaps (including Visibility Graph, Voronoi diagrams, and Silhouette), Cell Decomposition, Potential Fields, and Mathematical programming (including operations research and game theory models) [4].

Due to the complexities of the motion planning problem and its progressive increase for the multi-robot case, many heuristic and metaheuristic methods have been developed or applied extensively over the recent years, generally showing better performance than the classic methods in terms of computational burden. However, it should be noticed that heuristic methods do not guarantee to find a solution, but if a solution is found, it is done in much shorter time than exact methods.

1.1 Multi Robot Motion Planning

The Multi Robot Motion Planning (MRMP) problem has been solved through two main approaches: *centralized* and *decentralized* (or *decoupled*) [5].

The centralized planning considers all of the robots concurrently; that is, paths for all robots are planned simultaneously by searching the C-space of a hypothetical multi-arm robot consisted of all the robots, in which collisions between robots are considered as self-collisions of the multi-arm robot. The degrees of freedom (dof) of this hypothetical robot equals to the sum of the dof's of all individual robots. The main advantage of the centralized planning is that it is complete; i.e., it is guaranteed to find a solution if one exists. However, it is potentially expensive and typically requires searching high-dimensional spaces and the knowledge of goals and states of all robots, meaning that it hardly can be applied for online situations.

The decoupled planning performs the motion planning of each robot independently and sequentially, and has two phases: first a collision-free path τ_1 is generated for each robot considering only obstacles (ignoring other robots) in its space, and then, in order to prevent collisions between the robots, the robots' motions along their pre-generated paths are coordinated via two main techniques, namely *prioritization*, and velocity tuning. Each robot is restricted to move along its previously-generated path, although it may stop, retreat or change velocity to allow coordination with other robots [5].

The two main coordination approaches are *pairwise* and *global* coordination. In the pairwise coordination, the paths τ_1 and τ_2 of the first two robots are coordinated in their 2-dimensional coordination space. The process is repeated for paths $\tau_{1,\,2}$ and τ_3 resulting in a coordinated path $\tau_{1,\,2,\,3}$. Eventually, a collision-free coordinate path $\tau_{1,\,2,\,\ldots,\,m}$ is generated that defines a valid coordination of all m robots. In global coordination, the paths of all m robots are coordinated in an m-dimensional coordination space, resulting in a collision-free path $\tau_{1,\,2,\,\ldots,\,m}$.

The decoupled planning is generally less computationally expensive than the centralized planning since lower dimensional spaces are searched [6]. However, it is not complete, and failures usually occur in the second phase as it might not be possible to coordinate the paths generated in the first phase without collision

between robots [7]. Nevertheless, some attempts have been made to combine the centralized and decoupled approaches [8].

A trend of applying metaheuristic algorithms such as simulated annealing (SA), genetic algorithms (GA), and ant colony optimization (ACO) to the MRMP problem is noticeable especially among more recent contributions, as in [9], [10], and [11], respectively. Also, the particle swarm optimization (PSO) algorithm has found some applications MRMP. The first work in this regard is due to [12] in which the PSO is used for single and multiple target tracing applications for multiple robots. In [13] obstacle avoidance is done for a single robot in dynamic environments, in [14] bio-inspired group behaviors for deployment of a swarm of robots to multiple destinations are proposed. Other fresh works in this regard are [15], [16], and [17]. In [18] a PSO-inspired algorithm is proposed as a framework for robots to work together to find their targets. In [19] an asynchronous mechanism is proposed for information exchange and position update of small robots with limited sensing capabilities. In [20] a PSO-based method is developed for searching operations by a large number of mobile robots, with small inter-robot communications.

In this paper, a new PSO-based algorithm is developed for MRMP. The reason of employing the PSO is that as a population-based metaheuristic, it is very consistent with the distributed nature of multi robot systems. Moreover, although both the PSO and GA are population-based metaheuristics, the PSO proved to be more efficient and faster than the GA algorithm as reported in [21], after they were analyzed and compared statistically from both efficiency (speed) and effectiveness (quality) perspectives for eight optimization functions. The advantage of PSO over GA is also mentioned in [22].

A distinctive feature of the presented work, as compared to the previous works, is that the PSO is combined with a well-known and fast motion planning technique, called Probabilistic Road Map method (PRM), to produce obstacle-free paths in shorter times. Also, in order to enhance the quality of the produced paths, a multi-objective fitness function has been developed to minimize the path length while discouraging the robot to make sharp and abrupt turns, thus maintaining its smoothness.

2 Overview of the New Method

After analyzing many PSO-based algorithms and examining their components, it was found out that PSO is more successful in *diversification* rather than *intensification* due to high distribution of the particles in the space [23]. Intensification forces the algorithm to search a specific area with more depth and within a local scope, while diversification forces exploration of completely new regions, acting in a global scope. Therefore, the PSO component of the proposed algorithm was considered as a global planner, with the responsibility of searching and exploring new areas. This idea was first used in our previous work for a single robot [24].

Our analysis also showed that the PSO is not sufficiently efficient in obstacle avoiding, especially when a large number of obstacles populate the workspace densely, or there are narrow passageways in the workspace. Although thanks to the probabilistic nature of the PSO it can eventually find a collision-free path from the

robot's start to goal, this usually happens after so many unsuccessful attempts and thus takes considerable time. To remedy this drawback, we took advantage of a fast planner, namely, the *Probabilistic Road Map* (PRM) method, which is based on searching a graph with randomly-generated nodes and edges and is more powerful in local search (i.e. intensification). This component is described in section 4.

In addition to the abovementioned speed issue, the quality of the paths is also of great importance. Considering that the two major attributes of a high-quality path are its shortness and smoothness, we tried to incorporate these dual objectives in the fitness function, and concurrently minimize the length and maximize the smoothness of the path. This issue is addressed in section 3.

As the speed and efficiency are of specific importance in this work, the decentralized approach was employed: in fact, for m robots, m PSO algorithms are performed sequentially but independently in each iteration to determine the positions of the robots. This process is iterated until all robots reach their goals.

The combination and interaction of the PSO and PRM methods is a new concept in the field of multi-robot motion planning. As computational results have shown, PSO and PRM act very coherently since both have probabilistic elements and parameters. More specifically, the notions of particles in the PSO and random nodes generated in the PRM complement each other and unify these methods. The algorithm iteratively shifts from PSO to PRM until all robots' goals are reached.

For each robot, the following steps are executed:

1. A preset number of particles are generated around the robot's initial position and within its sensing range.
2. Each particle takes a new velocity and position based on the constantly updated improved PSO equations. A candidate for the robot's next position is determined by the position of the best particle (i.e. the one nearest to the goal).
3. If the robot's current position can be directly connected to the candidate best particle obtained in Step 2, then set it as the robot's next position and go to Step 2, otherwise continue with Step 4.
4. If the candidate best particle is located beyond an obstacle (i.e. the line connecting the best position to the robot's current position intersects an obstacle), a probabilistic roadmap is formed and searched for the shortest path. As a result, the current position of the robot is connected to a node of the PRM which is nearest to the goal through their shortest path.
5. Steps 2 to 4 are executed until the goal is within the robot's sensing range and can be accessed via a straight collision-free line.

The above steps are executed for every robot separately and concurrently until the last robot reaches its goal.

As mentioned earlier, the decentralized planning consists of two phases: the first phase specifies a collision-free start-to-goal path for each robot without considering other robots, and the second phase deals with velocity tuning, in which the robots' velocities along their generated paths are coordinated in order to avoid collision among the robots. In the proposed algorithm the *global coordination* approach is implemented for the velocity tuning, in which the paths of all m robots are coordinated in an m-dimensional space. Each robot is limited to move along its

previously generated path although it may stop and vary its speed for coordination with other robots. More precisely, whenever two robots get closer than a limit to each other, moving priorities are assigned to them at random, after which the robot with lower priority reduces speed to let the robot with higher priority pass.

Depending on the robot's start and goal positions, each robot reaches its goal at different times and after different number of iterations, and since the algorithm runs for each robot in parallel with others, at a specific moment, the planning phases underway for each robot might be different from other robots. Therefore, another factor called *action mode* was introduced to precisely describe the situation of each robot at a given time. This concept facilitates the robots' coordination and increases the algorithm's speed and efficiency.

There are five modes for each robot as explained below, in which $gbest^i$ is the position of the robot at the i-th iteration:

Mode 1 is for when a robot can move from $gbest^i$ to $gbest^{i+1}$ via a straight line without colliding with any obstacle. In other words, the robot's motion is planned by the global algorithm (PSO).
Mode 2 is for when Mode 1 does not hold due to collision with obstacles. As a result, the robot moves from $gbest^i$ to the nearest node in the PRM network.
Mode 3 is for when robot moves between two nodes of the PRM network. In other words, the robot's motion is totally planned by the local algorithm (PRM).
Mode 4 is for when the robot abandons the PRM network and moves to $gbest^{i+1}$ straightforwardly.
Mode 5 is for when the robot's goal is within its line of sight and can be reached directly without collision with obstacles.

It should be noted that the robots traverse the lines between two successive points with regard to their own speed and step size, and sequentially move to the intermediate points obtained from interpolating the line. For any robot, if after taking a step towards its desired point, it is estimated that a collision with another robot is imminent, the robot will automatically reduce its step size such that the collision is avoided. Therefore, a global coordination is performed at each iteration.

Also, at the end of each action mode an attempt is made to connect the robot's current position to its goal via a straight line. If it fails, the robot will continue moving to the $gbest^{i+1}$ directly or through a PRM network. If the attempt is successful, then the robot reaches its goal and the algorithm terminates unconditionally for that robot. However, since the robots do not essentially reach their goal at the same time, the termination criterion of the algorithm is satisfied whenever the last robot gets to its goal.

In the following two sections the details of the PSO and PRM components are described in detail.

3 PSO: The Global Planner

In the proposed MRMP method, the Particle Swarm Optimization method is employed as the global motion planner; that is, it is used for planning the large-scale,

'gross' motions of the robots. In this section, an overview of the basic (standard) PSO algorithm is presented.

The basic Particle Swarm Optimization algorithm was proposed by Kennedy and Eberhart in 1995 [25], inspired by the collective behavior of swarms of fish, birds, etc. Each member of the swarm is denoted by a particle, which shows a solution candidate. The particles start their fly from random positions in a search area, and in each iteration, they update their positions and velocities according to equations (1) and (2) below, and move to another position. Flying is affected by a fitness function that assesses the quality of each solution.

$$prtpos_j^i = prtpos_j^{i-1} + prtvel_j^i \tag{1}$$

$$prtvel_j^i = \chi \cdot \begin{bmatrix} w \cdot prtvel_j^{i-1} + \\ c_1 \cdot r_1 \cdot (pbest_j^{i-1} - prtpos_j^{i-1}) + \\ c_2 \cdot r_2 \cdot (gbest^{i-1} - prtpos_j^{i-1}) \end{bmatrix} \tag{2}$$

in which:

$prtpos^i_j$ = the position of the j-th particle in j-th iteration,
$prtvel^i_j$ = the velocity of the j-th particle in j-th iteration,
$pbest^{i-1}_j$ = the best position of the j-th particle at the end of $(i-1)$-th iteration,
$gbest^{i-1}$ = the best position in the swarm at the end of $(i-1)$-th iteration,

$$\chi = 2 / \left| 2 - \varphi - \sqrt{\varphi^2 - 4\varphi} \right|, \quad \varphi > 4.$$

The PSO has some dependent parameters: c_1 is a constant called *cognitive acceleration coefficient*, and c_2 is another constant named *collective acceleration coefficient*. These factors balance the effect of self-knowledge and social knowledge when particles move towards the target, and are usually set to a value of 2, although good results have been also produced with $c_1 = c_2 = 4$ [26]. r_1 and r_2 are random numbers between 0 and 1, different at each iteration, and χ is the *constriction factor*, which limits the velocity. w is a weight that regulates the global search behavior, set to an upper bound w_{max} in the beginning of the searching process and dynamically reduced during the optimization to a lower bound w_{min}, (which emulates a deeper local search behavior). Its range is suggested to be [0.2, 0.4].

The first term of equation (2), i.e. ($prtvel^{i-1}_j$), considers the velocity of the particle in the previous iteration, which produces a momentum needed for particles to fly all over the search space. The second term, ($pbest^{i-1}_j - prtpos^{i-1}_j$), which is known as the cognitive part, simulates the 'personal memory' of a particle: it encourages the particles to fly towards the best position they have found till now (i.e. *pbest*). Finally the third term, ($gbest^{i-1} - prtpos^{i-1}_j$), called the *collective part*, presents the effect of the particles' cooperation in finding the global optimum: it always directs the particles towards the best position ever found among all the members of the swarm.

The overall procedure of the PSO method has a main nested loop terminated when the total number of iterations exceeds a certain limit or a minimum error

threshold is achieved. In each iteration, particles are generated and best fitness values for each particle (*pbest*) and for the whole swarm (*gbest*) are calculated. Particles' positions and velocities are then updated based on (1) and (2).

To improve the performance of the basic PSO and increase its efficiency, we propose a modified, improved variant of the PSO algorithm. The new variant incorporates two new criteria for the particles' velocity updating equation, as shown by equation (3). The particles' positions are still updated by equation (1).

$$prtvel_j^i = \chi \cdot \begin{bmatrix} w_1 \cdot (prtvel_j^{i-1}) + \\ w_2 \cdot c_1 \cdot r_1 \cdot (pbest_j^{i-1} - prtpos_j^{i-1}) + \\ w_3 \cdot c_2 \cdot r_2 \cdot \alpha_1 \cdot (gbest^{i-1} - prtpos_j^{i-1}) + \\ w_4 \cdot c_3 \cdot r_3 \cdot \alpha_2 \cdot (pbest_{rand}^{i-1} - prtpos_j^{i-1}) + \\ w_5 \cdot c_4 \cdot r_4 \cdot \alpha_3 \cdot (prtvel_{rand}) \end{bmatrix} \tag{3}$$

In this equation, $pbest^i_{rand}$ is the best position of a randomly selected particle in the i-th iteration, $prtvel_{rand}$ is a random velocity vector with a size between $V_{\min}$ and $V_{\max}$, w_1 is the inertia weight, w_2–w_5 are control weights within [0.4, 0.9], c_1–c_4 are acceleration constants within [1.5, 4], r_1–r_4 are random numbers different at each iteration and in the range of [0, 1], and α_1–α_3 are respectively the influence factors of *gbest*, *pbest*, and $prtvel_{rand}$, in the ranges of [0, 10], [0, 20], and [0, 1].

We added two new terms based on the following logic: the fourth term of the velocity update equation, which we call the '*random self-cognition part*', sends the particles towards one of the best positions found randomly by particles ($pbest^{i-1}_{rand}$). This scheme gives an opportunity for reasonably good local positions of other randomly selected particles in the swarm to influence the velocities of other particles. The fifth term, which is enforced through the random velocity parameter $prtvel_{rand}$, increases the variety in the swarm and leads to a better and more effective movement of the swarm in narrow and complicated search areas.

All but the first term of equation (3) contribute to the overall velocity updating process in random proportions at each iteration. Consequently, the particles' positions spread all over the search space and the goal is reached quickly. If a particle lies inside an obstacle, it is simply deleted from the swarm and replaced by random particles in the free space.

In the basic PSO algorithm a number of particles are required to be created and positioned randomly in the search space. In our proposed method, a set of particles are generated for each robot with respect to its initial position and regarding its sensing range. The initial population is generated such that along each sensing direction, a particle is created at a certain distance from the robot, determined by the range of the used sensor. If any obstacle point is within the sensing range at that direction, a point near the obstacle's border is selected as the particle at that direction. Thus, the number of created particles depends on the number of sensors (or in a virtual space, on the number of divisions of the circumferential circle). Fig. 1 illustrates the creation of 36 particles around the robot's starting point. The larger

the number of divisions on the circle is, the larger the number of particles would be, and therefore the planning accuracy would be higher.

This innovative procedure has the advantage that the initial particles are generated around the robot's start point such that the movement from the start position to the next best position can be made through a fast, straightforward and safe connection within the sensing range. In existing PSO-based approaches, the initial positions are generated randomly, whereas in our method, while maintaining the centralization of the robot's start point, the obstacles' distribution around it is also considered.

3.1 Multiple-Objective Fitness Function

Most path planners aim to generate an optimal path considering a single criterion like path travel time or path length. However, in practice, a path is feasible if it meets several conditions, such as safety, estimated needed time for navigation, energy consumption, etc.

For robots needing to reach their destination as early as possible, a minimum-time path might seem desirable, but it may require a lot of time to be traversed due to uneasy terrain. Categorically, there exist various feasible paths between start and goal points being neither short nor fast but providing reasonable tradeoffs between shortness and fastness. These are generally desirable paths, while a path optimal for a single criterion without considering other equally important criteria is not desirable [27]. This is just one type of problems for which our multi-objective approach has been designed. In the developed method, the Simple Additive Weighting (SAW) technique is employed, in which a weighted sum of multiple objectives is expressed as a conventional single-objective function.

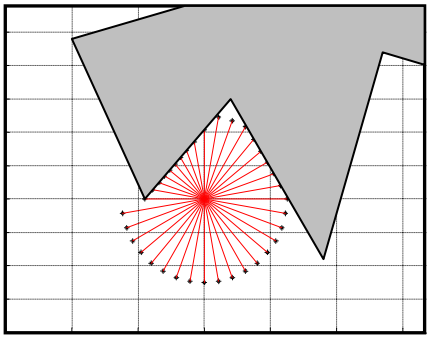


Fig. 1 The particles' initial population is generated based on the borders of the robot's sensed area

In our proposed method, the criterion for path shortness is defined as the Euclidean distance between each particle and the goal point in each iteration, and the criterion for path smoothness is defined as the angle between two hypothetical lines connecting the goal point to the robot's positions in two successive iterations, i.e. $gbest^{i}$ and $gbest^{i-1}$, in which i is the iteration number. The definition of

path smoothness in this way is a novel idea. The first objective function, i.e. the shortest path for the k-th robot, and the second objective function, the smoothest path for the k-th robot, are mathematically expressed in equations (4) and (5),

$$F_{\text{short}}(k)_j^i = \left\| prtpos(k)_j^i - goal(k) \right\| \tag{4}$$

$$F_{\text{smooth}}(k)_j^i = \frac{\cos^{-1}\left[(x_{prtpos(k)_j^i} - x_{goal(k)}) \cdot (x_{gbest(k)^{i-1}} - x_{goal(k)}) + (y_{prtpos(k)_j^i} - y_{goal(k)}) \cdot (y_{gbest(k)^{i-1}} - y_{goal(k)}) \right]}{\sqrt{(x_{prtpos(k)_j^i} - x_{goal(k)})^2 + (y_{prtpos(k)_j^i} - y_{goal(k)})^2} \times \sqrt{(x_{gbest(k)^{i-1}} - x_{goal(k)})^2 + (y_{gbest(k)^{i-1}} - y_{goal(k)})^2}} \tag{5}$$

in which (4) shows the distance of the particles' position to the goal point, and $k = 1, \ldots, m$ is the number of robot. The overall fitness (or objective) function is obtained by the weighted sum of the above shortest and smoothest objectives:

$$Fitness_j^i = \lambda_1 \cdot F_{\text{short}}{}_j^i + \lambda_2 \cdot F_{\text{smooth}}{}_j^i \tag{6}$$

By minimizing the overall fitness function with the assigned weights of each criterion, a shortest path with the least oscillations is obtained. The weights of the shortest and smoothest fitness functions, λ_1 and λ_2, are tuned through extensive simulation and try and errors, with best found values of $\lambda_1 = 0.7$ and $\lambda_2 = 0.3$.

4 PRM: The Local Planner

Due to its ease of implementation and ability to plan in high dimensional configuration spaces, the Probabilistic Roadmap method (PRM) has drawn considerable attention in recent motion planning works. Initial PRMs succeeded in solving a number of complex problems with high-dimensional configuration spaces which had not been solved efficiently until that time [28]. The PRM was enhanced later into some variant forms like Medial Axis PRM (MAPRM), Obstacle-Based PRM (OBPRM), and Visibility-based PRM, which improved the process of random node generation and made it more effective [29]. The PRM has also been applied in the multi robot systems [30].

The PRM has three phases: (1) generating random nodes in free configuration space, (2) connecting the nodes via some edges such that the edges lie in the free space and the nodes are connected through a single graph, and (3) searching the graph to find the shortest path between the start and goal nodes.

In the second phase, an edge is generated between two nodes by first trying to connect them via a straight line, and if this fails, a simple local planner is employed to connect them through a few intermediate newly generated nodes. The path planning is done by searching this graph.

In our version of PRM, four groups of nodes lying in free space are considered as the set of PRM nodes:

(*i*) a number of randomly generated nodes,
(*ii*) the robot's current position,
(*iii*) the best particles generated in the PSO,
(*iv*) two points around each corner of the obstructing obstacle.

The above combination of nodes is proposed for the first time in the literature, and provides a subtle intertwining of the PSO and PRM methods. In addition to the randomly generated nodes (group (*i*)) which are typical in the PRM method, about 30% – 40% of PSO particles with highest fitness values (*pbests*) are also integrated in the PRM graph. The group (*iv*) helps in circumnavigating obstacle vertices naturally and easily by creating nodes at safe clearances from both sides of a vertex.

After creating the necessary nodes, new edges are generated in the second phase of the PRM by connecting nodes to each other and deleting invalid edges (i.e., those intersecting with obstacles).

The shortest path between the robot's current position and the point *gbest* (calculated based on the best position among particles) is then found using the Dijkstra's search algorithm. As a result, the robot can move from $gbest^{i}$ to $gbest^{i+1}$ and get closer to the goal, while avoiding the obstacles that locally intercept its path to the goal. Once the robot is located on its new position, the PSO particles' velocities and positions are updated again, as described in equations (1) and (2).

5 Experimental Results

In order to analyze the function of the proposed new algorithm, numerous simulations were run for 2-, 3-, 4-, and 5-robot problems through which the algorithm's parameters were tuned to their best values. A few simulations for problems with simple to complex obstacles are illustrated in Fig. 2.

For comparing the algorithm's performance with other efficient and well-known algorithms, the standard PRM method was selected. Ten sample problems with 20 to 414 vertices were designed and solved for 2, 3, 4, and 5 robots using the proposed Improved PSO+PRM, Standard PSO+PRM, and standard PRM methods. Regarding that all these algorithms are heuristic and incorporate random parameters, we solved each problem set 5 times and calculated the mean value of runtimes. Note that the runtime is calculated based on the time needed for the last robot to reach its goal. In total, (10 problems) × (4 sets of robots) × (3 methods) × (5 times each) = 600 instances were run on an Intel 3.0 GHz processor.

The standard PSO against which we tested our algorithm was coded based on the basic PSO algorithm proposed in [25], combined with the PRM method. Also, the standard PRM was coded according to the explanations in section 4. In the above three methods, whenever a probabilistic roadmap was constructed (either in combination or stand-alone), it was searched by the Dijkstra's method to yield a shortest path on the roadmap. The results of solving the test problems are shown in Fig. 3 for 2, 3, 4, and 5 robots (from left to right, up to dawn), summarized in Table 1.

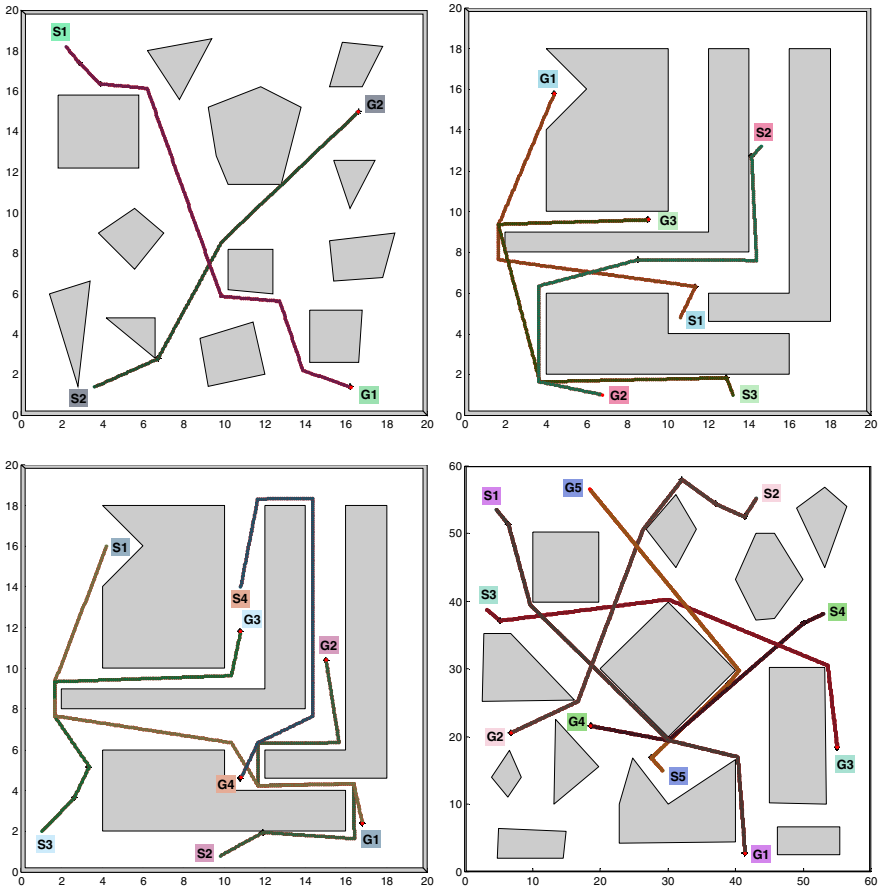


Fig. 2 Some simulations for 2-, 3-, 4-, and 5-robot motion planning. S_i and G_i indicate the start and goal of the i-th robot, respectively.

Table 1 Comparison of the average runtimes of the three methods and their standard deviations

	2 robots		3 robots		4 robots		5 robots		Total
	Avg.	SD	Avg.	SD	Avg.	SD	Avg.	SD	Avg.
Improved PSO + PRM	22.28	30.76	28.95	36.59	34.65	41.58	38.77	43.69	31.16
Standard PSO + PRM	31.27	41.35	34.36	43.26	40.23	47.53	43.71	47.54	37.39
Standard PRM	40.86	49.77	44.16	54.43	49.17	56.33	53.71	56.75	46.98

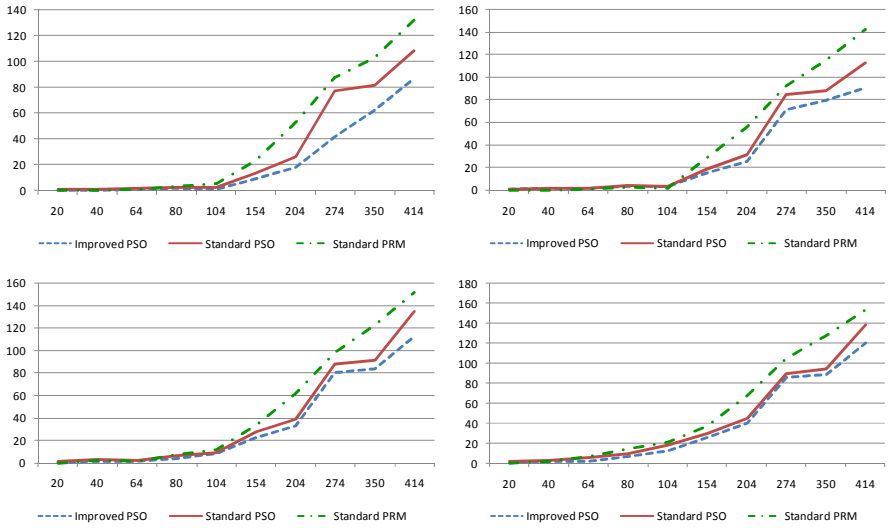


Fig. 3 Average runtime (s) vs. number of obstacle vertices for 2-, 3-, 4-, and 5-robot problems

The results of 600 solutions show that the proposed Improved PSO+PRM compound method was averagely about 17% and 34% faster than the Standard PSO+PRM and PRM methods, respectively, with considerably smaller standard deviation. Furthermore, the PSO+PRM method satisfied a bi-objective fitness function while in the PRM such a possibility is absent. Also, it is observed that runtime differences in the three methods increase as the number of vertices grows, showing the success of the new method in this type of problems.

6 Conclusions

In this paper, a new Improved PSO-based heuristic method is presented for multi-robot motion planning, which satisfies shortest and smoothest path objectives. The algorithm consists of a global planner (PSO) as well as a local planner (PRM). The multi robot motion planning problem is solved by this algorithm through decentralized planning with a global coordination model. For each robot, the proposed algorithm is run separately and then their motion coordination is performed all together and online. Also, five action modes were defined to describe the accurate situation of each robot at a given time. As a result, each robot moves one step toward its goal in each iteration.

The algorithm provides a novel and unique method to combine and coordinate the PSO and PRM algorithms by incorporating four groups of nodes within a single population. These nodes include: the best particles of the PSO, random nodes generated by PRM, current and next positions of the robot, and a pair of particles around each obstacle vertex. The set of these nodes form a network by being connected through straight edges. The shortest path between two consecutive robot

positions is then searched using a graph searching algorithm like the Dijkstra's method. As a result the free spaces around the obstacles can be searched in much less time than in the classic PRM algorithm.

After running and simulating 600 problem instances, the results showed that the proposed algorithm runs about 17% and 34% faster than the standard PSO+PRM and PRM methods, respectively, while two objectives are also optimized.

Considering the possibility of extending the PSO algorithm to high-dimensional spaces, we believe that the proposed method can be used for motion planning in high dimensional spaces provided that a proper distance metric is used.

References

[1] Lozano-Perez, T., Wesley, M.A.: An algorithm for planning collision-free paths among polyhedral obstacles. Communications of the ACM 22, 560–570 (1979)

[2] Warren, C.W.: Multiple robot path coordination using artificial potential fields. In: Proc. IEEE Int. Conf. Robotics and Automation, pp. 500–505 (1990)

[3] Canny, J.F.: The Complexity of Robot Motion Planning. The MIT press, Cambridge (1988)

[4] Hwang, Y.K., Ahuja, N.: Gross motion planning – A survey. ACM Computing Surveys 24(3), 219–291 (1992)

[5] Latombe, J.C.: Robot Motion Planning. Kluwer Academic Publishers, London (1991)

[6] Chun, L., Zheng, Z., Chang, W.: A decentralized approach to the conflict-free motion planning for multiple mobile robots. In: Proc. IEEE Int. Conf. Rob. Autom., vol. 2, pp. 1544–1549 (1999)

[7] Fujimura, K.: Motion Planning in Dynamic Environments. Springer, New York (1991)

[8] Arai, T., Ota, J.: Motion planning of multiple robots. In: Proc. IEEE Int. Conf. on Intelligent and Robotic Systems, pp. 1761–1768 (1992)

[9] Sanchez-Ante, G., Ramos, F., Frausto, J.: Cooperative Simulated Annealing for Path Planning in Multi-Robot Systems. In: Cairó, O., Cantú, F.J. (eds.) MICAI 2000. LNCS, vol. 1793, pp. 148–157. Springer, Heidelberg (2000)

[10] Sheng, G., Jie, Z., Hegao, C.: Genetic algorithm based path planning of coordinated multi-robot manipulators. In: Proc. IEEE Int. Conf. on Rob. Intell. Sys. & Signal Proc., pp. 763–767 (2003)

[11] Liu, S., Mao, L., Yu, J.: Path planning based on ant colony algorithm and distributed local navigation for multi-robot systems. In: Proc. IEEE Int. Conf. on Mech. and Autom., pp. 1733–1738 (2006)

[12] Doctor, S., Venayagamoorthy, G.K., Gudise, V.G.: Optimal PSO for collective robotic search applications. In: Proc. IEEE Congress on Evolutionary Computation, pp. 1390–1395 (2004)

[13] Min, H.Q., Zhu, J.H., Zheng, X.J.: Obstacle avoidance with multi-objective optimization by PSO in dynamic environment. In: Proc. IEEE Int. Conf. on Mach. Learning and Cyber., pp. 2950–2956 (2005)

[14] Berman, S., Halasz, A., Kumar, V., Pratt, S.: Bio-inspired group behaviors for the deployment of a swarm of robots to multiple destinations. In: Proc. IEEE Int. Conf. Rob. and Autom., pp. 2318–2323 (2007)

[15] Rigatos, G.G.: Distributed gradient and particle swarm optimization for multi-robot motion planning. Robotica 26(3), 357–370 (2008)
[16] Parhi, D.R., Pothal, J.K., Singh, M.K.: Navigation of multiple mobile robots using swarm intelligence". In: World Congress on Nature and Biologically Inspired Computing, pp. 1145–1149 (2009)
[17] Kim, S.H., Lee, G., Hong, I., Kim, Y.J., Kim, D.: New potential functions for multi robot path planning: SWARM or SPREAD. In: Proc. IEEE/ICCAE, vol. 2, pp. 557–561 (2010)
[18] Pugh, J., Martinoli, A.: Inspiring and modeling multi-robot search with particle swarm optimization. In: Proc. IEEE Swarm Intelligence Symp., pp. 332–339 (2007)
[19] Akat, S.B., Gazi, V., Marques, L.: Asynchronous particle swarm optimization-based search with a multi-robot system: simulation and implementation on a real robotic system. Turkish Journal of Electrical Engineering & Computer Science 18(5), 749–764 (2010)
[20] Hereford, J.M.: A distributed particle swarm optimization algorithm for swarm robotic applications. In: Proc. IEEE Congress on Evolutionary Computation, pp. 1678–1685 (2006)
[21] Hassan, R., Cohanim, B., de Weck, O.: A comparison of particle swarm optimization and the genetic algorithm. In: Proc. 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference (2004)
[22] Matsui, T., Kato, K., Sakawa, M., Uno, T., Matsumoto, K.: Particle swarm optimization for nonlinear integer programming problems. In: Proc. International Multi-Conference of Engineers and Computer Scientists, pp. 1874–1877 (2008)
[23] Sedighizadeh, D., Masehian, E.: Particle swarm optimization methods, taxonomy and applications. International Journal of Computer Theory and Engineering 1(5), 482–499 (2009)
[24] Masehian, E., Sedighizadeh, D.: Multi-objective robot motion planning using a particle swarm optimization model. Journal of Zhejiang University–Science C 11(8), 607–619 (2010)
[25] Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proc. IEEE Int. Conf. on Neural Networks, pp. 1942–1948 (1995)
[26] Shi, Y., Eberhart, R.C.: Particle swarm optimization with fuzzy adaptive inertia weight. In: Proc. Workshop on Particle Swarm Optimization, Indianapolis, pp. 101–106 (2001)
[27] Fujimura, K.: Path planning with multiple objectives. Journal of IEEE Robotics and Automation Society 3(1), 33–38 (1996)
[28] Kavraki, L., Svestka, P., Latombe, J.C., Overmars, M.: Probabilistic Roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation 12(4), 566–580 (1996)
[29] Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L.E., Thrun, S.: Principles of Robot Motion: Theory, Algorithms, and Implementations. MIT Press, Boston (2005)
[30] Sanchez, G., Latombe, J.C.: Using a PRM planner to compare centralized and decoupled planning for multi-robot systems. In: Proc. IEEE Int. Conf. on Rob. and Automation, pp. 2112–2119 (2002)

Decentralized and Prioritized Navigation and Collision Avoidance for Multiple Mobile Robots

Giannis Roussos and Kostas J. Kyriakopoulos

Abstract. We present an algorithm for the decentralised navigation of multiple mobile robots. Completely decentralised Navigation Functions are constructed, creating a potential field for each robot that gives rise to a feedback control law. The construction of the potential field incorporates limited sensing and explicit prioritisation in the form of priority classes. A non-circular sensing area creates asymmetrical sensing by reducing the influence of robots and obstacles behind each robot, introducing implicit priorities resembling "rules of the road". Static and moving obstacles are also taken into account, as well as malfunctioning robots that are unable to maneuver. A decentralised feedback control law based on the gradient of the potential field ensures convergence and collision avoidance for all robots, while respecting a lower speed bound. Simulation results demonstrate the efficacy of the proposed algorithm.

1 Introduction

Decentralised navigation has become popular in the field of robotics, as it is a prerequisite for a wide variety of applications involving multiple mobile robots. This problem is also being investigated from the point of view of multi-agent systems. In most multi-robot applications an increased level of decentralisation is desired to allow for greater performance, flexibility and computational efficiency. Moreover, a properly decentralised approach can offer some level of robustness with respect to single robot failures, limiting their effect on the rest of the robots.

A wide variety of methods for robot navigation has emerged, employing various techniques. One class of methods handles the problem in a two step approach [8]: the workspace is initially divided into cells, which are then used to formulate the

Giannis Roussos · Kostas Kyriakopoulos
Control Systems Lab, School of Mechanical Engineering, National Technical University of Athens, 9 Heroon Polytechniou Street, Zografou 15780, Greece
e-mail: {jrous,kkyria}@mail.ntua.gr

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 189–202.
springerlink.com

navigation problem as a graph search problem. Artificial potential or vector fields are then employed to guide the robots between cells, following the sequence provided by the graph search. An extension to multi-robot navigation is presented in [2]. Although this class of solutions is based on an intuitive line of thought, it requires considerable pre-calculations and thus a-priori knowledge. Moreover, performing the cell decomposition in the combined state space of all robots and solving the graph search problem can become very challenging computationally for large groups of robots.

A different class of methods uses artificial potential fields [6] to directly derive feedback controllers steering the robots over the entire workspace. A common weakness of these methods is the appearance of local minima away from the goal that can prevent convergence. A special class of potential fields, Navigation Functions (NFs) [7], can ensure the existence of a single, global minimum. The NF methodology has been developed for a wide class problems. The main advantages of this class of methods are the formal performance guarantees they can provide, computational efficiency and their real-time feedback nature, that can compensate for measuring and modeling errors. Navigation Functions have been so far applied to multi-agent problems ranging from robotic navigation [3] to Air Traffic Control (ATC) applications [11].

In this paper we further refine the concept of limited sensing in the proven NF methodology, which combined with a feedback control law designed on the principles of [11] yields a completely decentralised solution for multi-robot navigation and collision avoidance in a workspace with obstacles. Our approach requires no a-priori computation or knowledge and does not rely at all on centralised controllers. The only information that each robot needs is its position within the workspace and knowledge about other robots and obstacles within a sensing area around it. Thus, our algorithm is completely distributed and its computational cost does not depend on the total number of robots.

The construction of the potential fields incorporates priorities, both in explicit and implicit form. The former is achieved by assigning priority classes to the robots and allowing high priority robots to maintain right of way versus the lower priority ones. Moving and static obstacles are assigned the highest priority. Moving obstacles have been also considered in [1], but their motion is assumed to be known a-priori, as the algorithm pre-calculates the complete trajectories of the robots. Malfunctioning robots can also be treated as moving obstacles, thus offering some fault tolerance. Implicit priorities resembling "rules of the road" are introduced by using a non-circular sensing area, so that the potential of each robot is mostly influenced by robots and obstacles in front of it.

The rest of this paper is organised as follows: Section 2 formally defines the problem considered here, followed by Section 3 where the construction of the proposed potential field is described. In Section 4 the feedback control scheme is presented and discussed, while simulation results are given in Section 5. Finally, the conclusions of the paper are summarized in Section 6.

2 Problem Statement

We assume a scenario involving N mobile robots modeled as kinematic unicycles:

$$\dot{\boldsymbol{q}}_i = \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \end{bmatrix} = \mathbf{J}_i \cdot u_i,$$
$$\dot{\phi}_i = \omega_i,$$

where $\boldsymbol{q}_i = \begin{bmatrix} x_i & y_i \end{bmatrix}^\top$ is the position vector of robot i with respect to a global frame $\mathscr{E}$, ϕ_i its heading angle, i.e. the angle between the robot's longitudinal axis and the global x axis, and $\boldsymbol{J}_i = \begin{bmatrix} \cos(\phi_i) & \sin(\phi_i) \end{bmatrix}^\top$. Each circular robot i of radius r_i is driven via the linear velocity u_i and the angular velocity ω_i. For the linear motion a desired speed u_{di} is assumed, acting as a lower bound for the absolute linear velocity. All robots are operating inside a common workspace around the origin of $\mathscr{E}$ with radius R_w[1], while the information available to each of them is restricted to other robots and/or obstacles within its sensing area $\mathscr{A}_i$.

The control objective here is to drive each robot i to its destination $\boldsymbol{q}_{di}$ while avoiding all collisions with other robots and obstacles. While doing so, we want to enforce some form of prioritization, so that robots with high priority can maintain right of way versus lower priority ones. Our aim is to derive a completely decentralised solution that will also consider static and moving obstacles.

3 Completely Decentralised Navigation Functions

Decentralisation in the Navigation Functions (NFs) methodology has been introduced by allowing each robot to ignore the targets of other robots and navigate independently using its own NF-generated potential field. Limited sensing is a key factor for decentralisation: it takes into account the finite range of real sensors and greatly limits the information that each robot needs to acquire and process, significantly improving the applicability and scalability of the algorithm in large scenarios. A number of approaches to introduce limited sensing in the NF framework have been been presented. In [3] the authors implement limited sensing range in a C^0 fashion, but assume a priori knowledge of the total number of agents. This requirement has been eliminated in [4], where a switching sensing graph is used, resulting in a hybrid system. However, this approach does not ensure global stability, as blocking situations may be reached. Thus, convergence occurs only if the switching of the sensing graph eventually stops. A completely locally computable NF has been presented in [9], but only for single-agent problems and with the assumption that at each time instant there is at maximum one visible obstacle. This effectively means that the

[1] In the case of a non-circular workspace the algorithm presented here can still be applied by employing an appropriate transformation to a spherical workspace, as shown in [12].

algorithm solves the collision with one obstacle at a time, which is not optimal in a multi-agent scenario.

A completely decentralised scheme for a NF has been presented by the authors in [10], incorporating limited sensing range and explicit priorities in an absolutely locally computable potential field that can take into account multiple robots according to their priorities, as well as static and moving obstacles. The work presented here further develops this concept, by using non-circular sensing areas for each robot (see Figure 1a). By doing so, we allow each robot to use its full sensor range in the forward direction in order to exploit as much information as possible to plan its trajectory, while the effective sensing range is reduced in the rear direction. Such a sensing scheme introduces implicit prioritisation, as it can create situations with "asymmetrical" sensing between robots (see Figure 1b). In graph theory terms, this means that the communication graph is no longer undirected. Moreover, the shape of the sensing area improves the computational efficiency of the algorithm, as it allows a finer selection of the neighboring obstacles and robots that contribute to the potential field and influence the motion of each robot.

The potential field presented here can be combined with a control scheme similar to the one presented in [11] to provide decentralised, non-cooperative navigation for multiple robots. In fact, any controller that can ensure a decreasing rate for the potential's value over time is applicable. Thus, the use of the potential field presented here is not limited to unicycle-like robots, but can also be applied to other types of kinematics (holonomic or non-holonomic), when combined with an appropriate control scheme.

The decentralised Navigation Function (NF) we use is of the form [5]:

$$\Phi_i = \frac{\gamma_i + f_i}{\left((\gamma_i + f_i)^k + G_i \cdot \beta_i\right)^{1/k}}, \tag{1}$$

where γ_i is the target function, f_i the cooperation function, G_i the obstacle function and finally β_i is the workspace boundary function. Functions G_i and β_i fade on the boundary of collisions with other agents or the workspace boundary respectively and take positive values otherwise, while function γ_i is 0 on the destination and positive away from it. Finally, f_i is always non-negative. The potential Φ_i attains its maximum value of 1 on the boundary of collisions and has a single minimum of 0 at the destination. Our contribution focuses in the construction of the obstacle function G_i, where we incorporate the non-circular sensing scheme presented in section 3.2, which combined with the priority classes described in section 3.1 allows the use of both explicit and implicit prioritisation between the robots.

3.1 Priority Classes

Explicit prioritisation is introduced in the Navigation Functions (NFs) algorithm by assigning each robot i, $i \in \{1,\ldots,N\}$ a priority class $c_i \in \mathbb{N}$. Lower values of c_i represent higher priority robots, with $c_i = 0$ denoting either uncontrolled or otherwise

unable to maneuver robots, or obstacles (stationary or moving). The assignment of priorities can be performed independently of the navigation algorithm presented here. Thus, aspects like the importance of each robot's task, the robot's capabilities, etc can be taken into account for the classification. We define the *threat set* T_i of each robot i with $c_i > 0$ as the set of all obstacles and robots of the same or higher priority class, i.e. with the same or lower c_i:

$$T_i \triangleq \left\{ j \in \{1,.,N\} \setminus \{i\} \,\middle|\, c_j \leq c_i \right\}. \tag{2}$$

For the robots belonging to the highest priority class, i.e. with $c_i = 0$, the respective threat set is empty, $T_i = 0$. Priorities are used to define the sensing relations between neighboring robots: each robot i ignores any other robots that belong to lower priority classes, i.e. any robot j with $c_j > c_i$, and only considers robots and obstacles belonging to its threat set T_i. Thus, robots performing high priority tasks can maintain right of way, while lower priority robots steer around them.

Moving and static obstacles are handled like uncontrolled robots; they are assigned the maximum priority class, $c_i = 0$ and are avoided by all normally operating robots. Moreover, if a robot i is known to suffer a degradation of its navigation and collision avoidance capabilities it is assigned the highest priority class $c_i = 0$ and is treated as an obstacle by other robots[2]. This classification scheme means that two robots i and j have mutual sensing between them, i.e. they both take each other into account to navigate, $i \in T_j$ and $j \in T_i$, if and only if $c_i = c_j \neq 0$, i.e. they belong to the same priority class, other than the highest one. Otherwise, if one of the robots, say i, belongs to a higher priority class (even the highest one), $0 \leq c_i < c_j$, then $i \in T_j$ but $j \notin T_i$. Thus, at all combinations of c_i, c_j where at least one of them is nonzero, i.e. $max(c_i, c_j) > 0$, there is at least one-way sensing between robots i and j. This ensures that all collisions are avoided, at least by one of the two involved robots. Of course, in the unfortunate case that 2 robots i and j malfunction simultaneously, i.e. $c_i = c_j = 0$, any collisions between them can not be avoided, though all other normally operating robots will still manoeuvre around both of them.

3.2 Limited Sensing

The effective sensing area used by each robot consists of a semicircle of radius R_{sr} in the rear semi-plane and a semi-ellipse with semimajor and semiminor axes R_{sf}, R_{sr} (with $R_{sf} > R_{sr}$) respectively in the forward semiplane, as shown in Figure 1a. Range R_{sf} should be less or equal to the maximum range allowed by the robot's sensors to exploit as much information as possible, while R_{sr} should be enough to allow effective collision avoidance in all directions. The boundary of the sensing area around the robot is then given by:

[2] Online priority reassignment is outside the scope of this work, as it is assumed that it will be handled by an independent fault detection system.

$$R_s(\theta) = \begin{cases} \frac{R_{sr}R_{sf}}{\sqrt{(R_{sr}\cos(\theta))^2 + (R_{sf}\sin(\theta))^2}}, & \theta \in (-\frac{\pi}{2}, \frac{\pi}{2}) \\ R_{sr}, & otherwise \end{cases} \tag{3}$$

The angle $\theta \in (-\pi, \pi]$ is measured from the forward direction of the robot, as shown in Figure 1a. Similarly, for each neighbor j of robot i we define the bearing angle θ_{ij} between the relative position vector $\boldsymbol{q}_{ij} = \boldsymbol{q}_j - \boldsymbol{q}_i$ and the forward direction of robot i, as shown in Figure 1b. The effective sensing range of robot i in the direction of $\boldsymbol{q}_{ij}$ is given by $R_s(\theta_{ij})$, using (3). In the special case that $R_{sf} = R_{sr}$ the sensing zone becomes a circle, as in [10]. The elliptical shape offers a simple way for an adjustable forward sensing range in a C^1 fashion, though other C^1 curves may be used if required by specific applications.

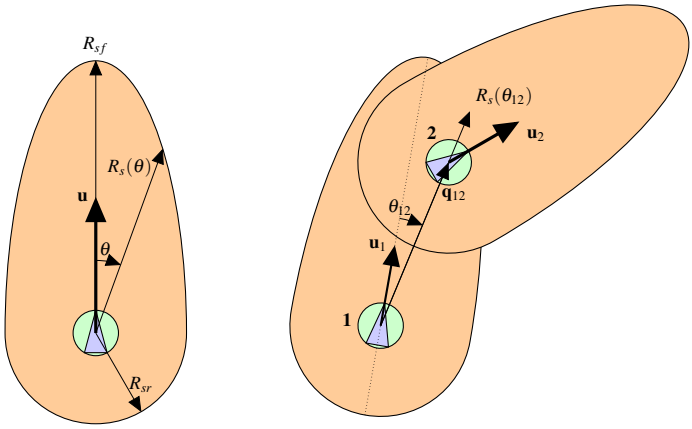


Fig. 1 (a): The non-circular sensing area used for each robot, consisting of a semicircle in the rear and a semi-ellipse in the front of the robot.
(b): Implicit prioritisation of robot 2 wrt robot 1: Robot 1 is outside robot's 2 sensing area, while robot 2 is insides robot's 1 sensing area. Thus, only robot 1 senses robot 2 (but not vice versa) and only robot 1 will manoeuvre.

The contribution of robot j to the potential field of robot i is based on the basic obstacle function $\hat{g}_{ji}$, which is defined as in previous NF approaches:

$$\hat{g}_{ij} = \left|\left|\boldsymbol{q}_{ij}\right|\right|^2 - r_{ij}^2 \tag{4}$$

where $r_{ij} \triangleq r_i + r_j$. By the above definition, $\hat{g}_{ij}$ is zero when robot j touches robot i, i.e. when $\left|\left|\boldsymbol{q}_{ij}\right|\right| = r_{ij}$, and increases as the robots move away from each other.

Each robot can sense other robots or obstacles that are inside the above sensing area, i.e. whenever $\left|\left|\boldsymbol{q}_{ij}\right|\right| \leq R_s(\theta_{ij})$. The effective sensing range $R_s(\theta_{ij})$ is used, as presented in [10], to derive the normalised obstacle function $\bar{g}_{ij}$:

$$\bar{g}_{ij} = \frac{\hat{g}_{ij}}{R_s(\theta_{ij})^2 - r_{ij}^2} \tag{5}$$

Finally, the contribution g_{ij} of robot (or obstacle) j to robot i's potential is derived:

$$g_{ij} = \begin{cases} L(\bar{g}_{ij}), & \left|\left|\boldsymbol{q}_{ij}\right|\right| \leq R_s(\theta_{ij}) \\ 1, & \left|\left|\boldsymbol{q}_{ij}\right|\right| > R_s(\theta_{ij}) \end{cases} \tag{6}$$

where the shaping function $L(x)$ is $L(x) = x^3 - 3x^2 + 3x$, chosen to satisfy:

$$L(0) = 0 \qquad L(1) = 1$$
$$L'(x) > 0 \; \forall x \in [0,1) \qquad L'(1) = L''(1) = 0$$

By the above definition, g_{ij} is zero when robots i and j collide, i.e. $\left|\left|\boldsymbol{q}_{ij}\right|\right| = r_{ij}$ and increases up to 1 at the boundary of the sensing area, i.e. when $\left|\left|\boldsymbol{q}_{ij}\right|\right| = R_s(\theta_{ij})$. Outside the sensing area of robot i, g_{ij} is constantly 1. Using the above properties of $L(x)$ it can be verified that g_{ij} is by construction C^2 in the interior of the free space, i.e. away from collisions, where $\hat{g}_{ij} \in [0, +\infty)$. This allows the potential Φ_i to be C^2, as it is required for it to be a Navigation Function [7]. Function $g_{ij} = g_{ij}\left(\left|\left|\boldsymbol{q}_{ij}\right|\right|\right)$ is plotted in Figure 2a. Since g_{ij} is constantly 1 when $\left|\left|\boldsymbol{q}_{ij}\right|\right| \geq R_s(\theta_{ij})$, each robot i is only affected by other robots $j \in T_i$ inside its sensing area. It should be noted here that although $\hat{g}_{ij} = \hat{g}_{ji}$, the nondimensional functions g_{ij} and g_{ji} are not equal in general, since θ_{ij} and θ_{ji} are different. This difference between g_{ij} and g_{ji} introduces the asymmetrical sensing in the construction of the potential fields of robots i and j.

The complete obstacle function G_i is then constructed:

$$G_i = \prod_{j \in T_i} g_{ij}. \tag{7}$$

The priority classes defined in 3.1 are used here to allow robot i to ignore j when $c_i < c_j$, forcing robot j to manoeuver around i. Essentially, this G_i construction means that only knowledge about those robots in T_i that are within the sensing area of i is required:

$$G_i = \prod_{j \in \tilde{T}_i} g_{ij}, \tag{8}$$

where $\tilde{T}_i$ is the "close threat" set that comprises threats in the sensing area of i:

$$\tilde{T}_i = \left\{ j \in T_i \,\middle|\, \left|\left|\boldsymbol{q}_{ij}\right|\right| < R_s(\theta_{ij}) \right\} \subset T_i. \tag{9}$$

Similarly to g_{ij}, β_i is designed to contain the influence of the workspace boundary in a zone of width R_{sf}. The dimensional workspace boundary function $\hat{\beta}_i$ is:

$$\hat{\beta}_i = (R_w - r_i)^2 - ||\boldsymbol{q}_i||^2$$

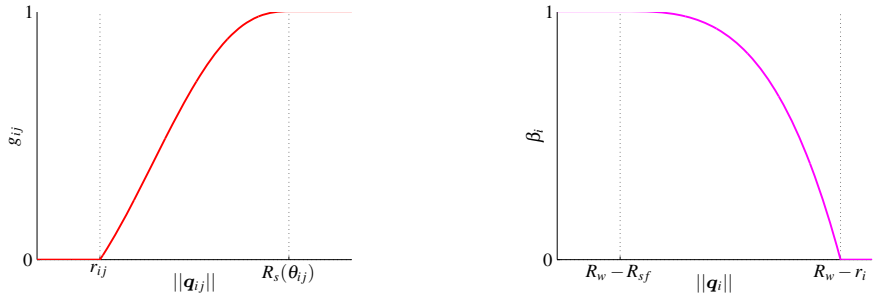


Fig. 2 (a): Obstacle function g_{ij} wrt distance $\left|\left|\boldsymbol{q}_{ij}\right|\right|$ between robots i and j
(b): Workspace boundary function β_i wrt $||\boldsymbol{q}_i||$

The corresponding normalised boundary funtion $\bar{\beta}_i$ is:

$$\bar{\beta}_i = \frac{\hat{\beta}_i}{\left(R_w - r_i\right)^2 - \left(R_w - R_{sf}\right)^2} \tag{10}$$

The effective boundary function β_i used in Φ_i is then defined similarly to g_{ij}:

$$\beta_i = \begin{cases} L\left(\bar{\beta}_i\right) & ||\boldsymbol{q}_i|| \geq R_w - R_{sf} \\ 1, & ||\boldsymbol{q}_i|| < R_w - R_{sf} \end{cases} \tag{11}$$

Thus, β_i becomes zero when robot i touches the workspace boundary, i.e. when $||\boldsymbol{q}_i|| = R_w - r_i$, and varies in a C^2 fashion to exactly 1 when robot i is at a distance equal to or higher than R_{sf} away from the boundary, i.e. when $||\boldsymbol{q}_i|| \leq R_w - R_{sf}$, see Figure 2b.

3.3 Potential Construction

For the target function γ_i we use the following nondimensional form:

$$\gamma_i = \frac{||\boldsymbol{q}_i - \boldsymbol{q}_{di}||^2}{R_w^2} \tag{12}$$

Since the largest distance between any two positions $\boldsymbol{q}_i$, $\boldsymbol{q}_{di}$ inside the spherical workspace of radius R_w is $2R_w$, γ_i is equal to or lower than 4 for any combination of $\boldsymbol{q}_i$, $\boldsymbol{q}_{di}$.

The cooperation function f_i is used here as in [3]:

$$f_i(G_i) = \begin{cases} a_0 + \sum_{l=1}^{3} a_l G_i^l, & G_i \leq X \\ 0, & G_i > X \end{cases} \tag{13}$$

where $a_0 = Y$, $a_1 = 0$, $a_2 = \frac{-3Y}{X^2}$, $a_3 = \frac{2Y}{X^3}$ and X, Y are positive parameters. X sets a threshold for G_i, so that the cooperation function f_i is activated when $G_i < X$. Parameter Y defines the maximum value of f_i, attained when $G_i = 0$.

The final result of using the above defined G_i, β_i and γ_i in (1) for a setup with 3 obstacles is shown in Figure 3. The target $\boldsymbol{q}_{di}$ is set in the center of the workspace and 3 obstacles are included. Figure 3b presents the potential field in the workspace, while Figure 3a shows the values of G_i, β_i, γ_i and Φ_i along the positive x axis, that crosses through the center of one of the obstacles that is placed between the target and the workspace boundary. In this example we have assumed that the cooperation function f_i is not activated, i.e. $f_i = 0$ everywhere. Moreover, since we cannot plot the potential for all possible robot orientations, we have used $R_s(\theta_{ij}) = R_{sf}$ everywhere for simplicity in the figure. As figure 3a demonstrates, G_i and β_i become less than 1 only within the sensing range R_{sf} of the obstacle and workspace boundary, respectively. The dotted blue line represents the value of Φ_i for $G_i = \beta_i = 1$ everywhere, i.e. without the effect of any obstacles or the workspace boundary. As expected, this coincides with the actual Φ_i outside the sensing area of obstacles and the workspace boundary.

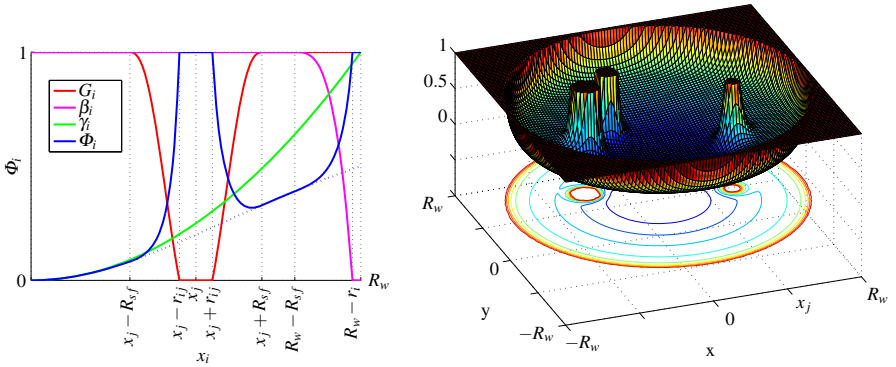


Fig. 3 Left: Obstacle function G_i, workspace boundary function β_i, target function γ_i and the resulting Navigation Function Φ_i on the line $y_i = 0$, $x_i \in [0, Rw]$.
Right: Navigation Function potential field in a workspace with 3 obstacles and local sensing.

The invariance of the properties of Navigation Functions (NFs) under diffeomorphisms has been used in [10] to show that the construction of the potential field described here results in a proper Navigation Function. Therefore, it can provide almost global navigation and collision avoidance for all values of k higher than a finite lower bound k_0. Moreover, since in the construction presented here only robots and

obstacles inside the non-circular sensing area $\mathscr{A}_i$ affect the potential Φ_i, the number of g_{ij} that contribute to Φ_i at any given time is significantly reduced wrt [10] and previous approaches with global sensing. This significantly boosts the computational efficiency of the algorithm, especially in scenarios involving many robots. Simulation experience with NFs indicates that the minimum value of the exponent k required to eliminate local minima and render (1) a NF increases with the number of contributing robots and obstacles. Thus, the exponent k needed for the potential presented here is in most cases lower than the one required in [4].

4 Completely Decentralised Navigation

The proven navigation properties of the potential field Φ_i described above can be used to drive each robot to its destination while avoiding collisions. In fact, any controller that can maintain a decreasing rate for each potential field Φ_i, i.e. $\dot{\Phi}_i < 0$ can be employed in combination with the potential field presented previously to stabilise each robot to its target, while also avoiding collisions. Such a controller has been presented in [11] for unicycle-like vehicles moving in 3D space . We can derive a similar controller for planar unicycles by neglecting the vertical velocity input. The resulting control scheme employs the projection of the gradient $\nabla_i \Phi_i = \left[\Phi_{ix} \ \Phi_{iy} \right]^\top$ on robot's i longitudinal (heading) direction:

$$P_i = \boldsymbol{J}_i^\top \cdot \nabla_i \Phi_i \tag{14}$$

where $\boldsymbol{J}_i = \left[\cos(\phi_i) \ \sin(\phi_i) \right]^\top$. Moreover, we use the partial derivative $\frac{\partial \Phi_i}{\partial t}$, which sums the effect of all but the i^{th} robots' motion on Φ_i:

$$\frac{\partial \Phi_i}{\partial t} = \sum_{j \neq i} \nabla_j \Phi_i^\top \cdot \boldsymbol{J}_j u_j$$

where $\nabla_j \Phi_i = \frac{\partial \Phi_i}{\partial \boldsymbol{q}_j}$ is the gradient of Φ_i with respect to $\boldsymbol{q}_j$.

The proposed control law for the linear velocity u_i is:

$$u_i = \begin{cases} -\operatorname{sgn}(P_i) U_i, & \frac{\partial \Phi_i}{\partial t} \leq U_i \left(|P_i| - \varepsilon \right) \\ -\operatorname{sgn}(P_i) \frac{U_i \varepsilon + \frac{\partial \Phi_i}{\partial t}}{|P_i|}, & \frac{\partial \Phi_i}{\partial t} > U_i \left(|P_i| - \varepsilon \right) \end{cases}, \tag{15}$$

where U_i is the nominal velocity:

$$U_i = \begin{cases} u_{di}, & ||\boldsymbol{q}_i - \boldsymbol{q}_{di}|| > d_i \\ \frac{||\boldsymbol{q}_i - \boldsymbol{q}_{di}||}{d_i} \cdot u_{di}, & ||\boldsymbol{q}_i - \boldsymbol{q}_{di}|| \leq d_i \end{cases},$$

while $\mathrm{sgn}(x)$ is a modified sign function:

$$\mathrm{sgn}(x) \triangleq \begin{cases} 1, & \text{if } x \geq 0 \\ -1, & \text{if } x < 0 \end{cases}$$

and ε is a small positive constant used to ensure a decreasing rate of Φ_i. The nominal speed U_i matches identically the desired value u_{di} away from the target $\boldsymbol{q}_{di}$ and reduces continuously to 0 inside a ball of radius d_i around $\boldsymbol{q}_{di}$. The angular velocity used is:

$$\omega_i = \begin{cases} 0, & M_i \geq \varepsilon_\phi \\ \Omega_i \cdot \left(1 - \frac{M_i}{\varepsilon_\phi}\right), & 0 < M_i < \varepsilon_\phi \\ \Omega_i, & M_i \leq 0, \end{cases} \tag{16}$$

$$\text{where:} \qquad \begin{aligned} M_i &\triangleq \dot{\phi}_{\boldsymbol{nh}_i}\left(\phi_i - \phi_{\boldsymbol{nh}_i}\right), \\ \Omega_i &\triangleq -k_\phi\left(\phi_i - \phi_{\boldsymbol{nh}_i}\right) + \dot{\phi}_{\boldsymbol{nh}_i}. \end{aligned}$$

The *nonholonomic* heading angle $\phi_{\mathbf{nh}i}$ represents the heading of $\mathrm{sgn}(p_i)\nabla_i\Phi_i$:

$$\phi_{\mathbf{nh}i} \triangleq atan2\left(\mathrm{sgn}\left(p_i\right)\Phi_{iy}, \mathrm{sgn}\left(p_i\right)\Phi_{ix}\right), \tag{17}$$

where: $\mathrm{atan2}(y,x) \triangleq \arg(x,y), \quad (x,y) \in \mathbb{C}$ and $p_i = \mathbf{J}_{di}^{\top} \cdot (\boldsymbol{n}_{i1} - \boldsymbol{n}_{i1d})$ is the position vector with respect to the destination, projected on the longitudinal axis of the desired orientation. Consequently, $\mathrm{sgn}(p_i)$ is equal to 1 in front of the target configuration and -1 behind it. Finally, ε_ϕ is a small positive constant and k_ϕ a positive gain.

4.1 Stability and Convergence Analysis

The principles of this control scheme can be found in detail in [11]. Since the stability analysis presented there does not rely on the specific Navigation Function used, one can follow the same line of thought to prove that the above control scheme ensures a decreasing rate for all Φ_i over time:

$$\dot{\Phi}_i \leq -u_{di}\varepsilon \tag{18}$$

Thus convergence and collision avoidance are guaranteed. It should be noted though that here we have not included in the Navigation Function (1) a nonholonomic obstacle H_{nh} to render it Dipolar [12]. Thus, the integral lines of the resulting potential field approach the destination with arbitrary orientation and consequently the final orientation of each unicycle is not predefined, i.e. only its position is stabilised.

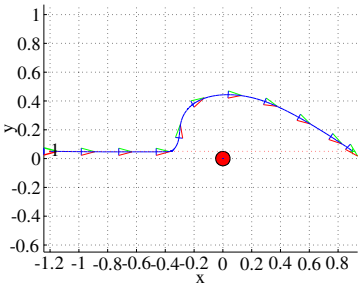
The use of the priority scheme described in 3.1 means that collisions between any two robots i, j are avoided when at least one of them has non-zero priority,

$max(c_i, c_j) > 0$, i.e. one of them is able to manoeuvre. This holds because by construction a NF is transverse on the boundary of collisions with other robots or obstacles. One can easily show similarly to [11] that the above control scheme ensures that $\nabla_i \Phi_i u_i \leq 0$ holds always, i.e. all robots steer to align with the gradient's heading and move towards the direction that decreases their potential. Thus, when there is at least one-way sensing between any two neighboring robots, at least one of the robots moves away from the other and collisions between them are avoided. Of course, when both robots are uncontrolled, $c_i = c_j = 0$, no collision avoidance can be performed between them. Thus, the proposed control scheme combined with the priority rules in section 3.1 ensures that all collisions between two controlled robots, or a controlled and an uncontrolled one or an obstacle are avoided.

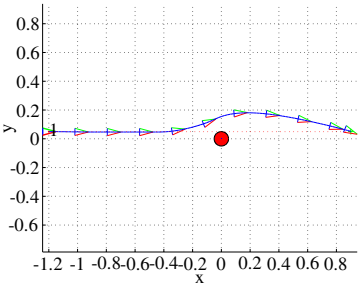
5 Simulation Results

In order to demonstrate the effect of the noncircular sensing area to the performance of the algorithm we present simulation results below. The first simulation scenario is a simple example with a robot navigating around one static obstacle. Although this is not a challenging scenario, it is useful to give a clear view of the performance and efficiency improvements that our algorithm achieves compared to the circular sensing scheme in [10]. A maximum sensing range of 0.5 length unit is assumed, so for the new sensing we have used $R_{sr} = 0.15$, $R_{sf} = 0.5$, i.e. the full sensor range is exploited in the forward direction, but only 30% of it in the rear. Results from the same example using circular sensing are included, using two different sensing ranges, $R_s = 0.5$ and $R_s = 0.15$. The resulting paths are shown in Figures 4a-4c, along with some statistical information in Table 4d. Compared to the full sensor range in 4a, the new sensing scheme path in 4b is significantly less conservative. In both 4a and 4b cases the robot starts turning at around the same position, near $x = -0.3$, as the forward sensing range is the same. However, the significantly shorter sensing range to the sides and rear of the robot with the new sensing scheme results in a much smaller deviation from the straight line path. Using a reduced circular sensing of radius 0.15 in 4c results in a more aggressive turn as the robot starts maneuvering later, and eventually covers longer distance to reach the target. The improvements of the algorithm presented here are reflected in the total length of the paths shown in table 4d, as well as the computation time, because of the reduced interaction between the robot and the obstacle. Finally the total absolute turning angle $A = \int |\omega| \, dt$ is reduced, as less maneuvering is used with the new algorithm.

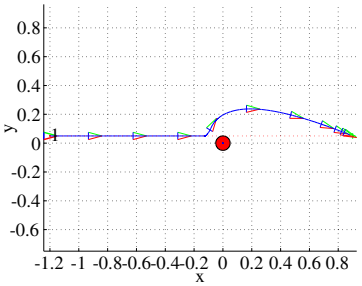
The second simulation example is a multirobot scenario similar to the one used in [10]: 4 low priority robots are moving in parallel, while a high priority one is crossing their paths. Results are presented in Figure 5a for a circular sensing $R_s = 0.5$ and in 5b for the noncircular sensing scheme with $R_{sr} = 0.15$ and $R_{sf} = 0.5$. As noted on the figures, the new algorithm results in much smaller deviations, allowing the robots to reach their targets quicker.



(a) Circular sensing with $R_{sr} = R_{sf} = 0.5$



(b) Noncircular sensing with $R_{sr} = 0.15$, $R_{sf} = 0.5$

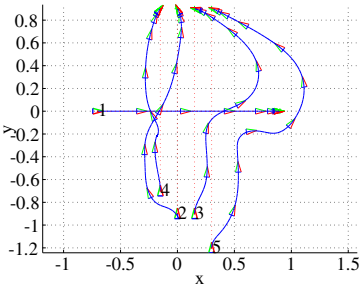


(c) Circular sensing with $R_{sr} = R_{sf} = 0.15$

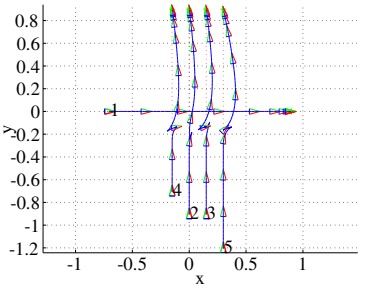
Sensing Scheme	(a)	(b)	(c)
Computation Time	25*sec*	20*sec*	26*sec*
Path Length	7.56	5.13	6.59
Total steering angle	6.11	3.66	5.00

(d)

Fig. 4 Simulation results: Obstacle avoidance using various sensing schemes



(a) Circular sensing, $R_s = 0.5$. Path length: 23.89



(b) Noncircular sensing, $R_{sf} = 0.5$, $R_{sr} = 0.15$. Path length: 20.9

Fig. 5 Simulation Results: A high priority robot crosses the paths of 4 lower priority ones moving in parallel

6 Conclusions

We have presented an algorithm for multi-robot navigation and collision avoidance using the Navigation Functions framework. A novel sensing scheme is implemented in the method, allowing implicit prioritisation in a rules-of-the-road fashion. Explicit prioritisation is also taken into account, as well static or moving obstacles and uncontrollable robots. Simulation results show this new algorithm to offer significant performance and efficiency improvements with respect to previous work utilising the NF methodology. Future work in this area is directed towards the extension of the algorithm to 3D navigation and further verification against complex scenarios.

Acknowledgements. The authors of this paper want to acknowledge the contribution of the European Commission through project iFLY.

References

1. Ahmadzadeh, A., Motee, N., Jadbabaie, A., Pappas, G.: Multi-vehicle path planning in dynamically changing environments. In: 2009 IEEE International Conference on Robotics and Automation, pp. 2148–2153. IEEE Press (2009)
2. Ayanian, N., Kumar, V.: Decentralized feedback controllers for multi-agent teams in environments with obstacles. In: IEEE Int. Conf. on Robotics and Automation, pp. 1936–1941 (2008), doi:10.1109/ROBOT.2008.4543490
3. Dimarogonas, D.V., Kyriakopoulos, K.J.: Decentralized navigation functions for multiple robotic agents with limited sensing capabilities. Journal of Intelligent and Robotic Systems 48(3), 411–433 (2007)
4. Dimarogonas, D.V., Kyriakopoulos, K.J., Theodorakatos, D.: Totally distributed motion control of sphere world multi-agent systems using decentralized navigation functions. In: 2006 IEEE International Conference on Robotics and Automation, pp. 2430–2435 (2006)
5. Dimarogonas, D.V., Loizou, S.G., Kyriakopoulos, K.J., Zavlanos, M.M.: A feedback stabilization and collision avoidance scheme for multiple independent non-point agents. Automatica 42(2), 229–243 (2006)
6. Khatib, O.: Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. The International Journal of Robotics Research 5(1), 90–98 (1986), doi:10.1177/027836498600500106
7. Koditschek, D.E., Rimon, E.: Robot navigation functions on manifolds with boundary. Advances in Applied Mathematics 11(4), 412–442 (1990)
8. Lindemann, S., Lavalle, S.: Smoothly blending vector fields for global robot navigation. In: 44th IEEE Conference on Decision and Control and European Control Conference, ECC 2005. Omnipress (2005)
9. Lionis, G., Papageorgiou, X., Kyriakopoulos, K.J.: Locally computable navigation functions for sphere worlds. In: Proceedings of the 2007 IEEE International Conference on Robotics and Automation, pp. 1998–2003 (2007)
10. Roussos, G., Kyriakopoulos, K.J.: Completely decentralised navigation of multiple unicycle agents with prioritization and fault tolerance. In: 49th IEEE Conference on Decision and Control (2010) (to appear)
11. Roussos, G., Kyriakopoulos, K.J.: Decentralised navigation and collision avoidance for aircraft in 3D space. In: 2010 American Control Conference, Baltimore, USA (2010)
12. Tanner, H.G., Loizou, S., Kyriakopoulos, K.J.: Nonholonomic navigation and control of cooperating mobile manipulators. IEEE Transactions on Robotics and Automation 19(1), 53–64 (2003)

Optimal Reciprocal Collision Avoidance for Multiple Non-Holonomic Robots

Javier Alonso-Mora, Andreas Breitenmoser, Martin Rufli,
Paul Beardsley, and Roland Siegwart

Abstract. In this paper an optimal method for distributed collision avoidance among multiple non-holonomic robots is presented in theory and experiments. Non-holonomic optimal reciprocal collision avoidance (NH-ORCA) builds on the concepts introduced in [2], but further guarantees smooth and collision-free motions under non-holonomic constraints. Optimal control inputs and constraints in velocity space are formally derived for the non-holonomic robots. The theoretical results are validated in several collision avoidance experiments with up to fourteen e-puck robots set on collision course. Even in scenarios with very crowded situations, NH-ORCA showed to be collision-free for all times.

1 Introduction

Multi-robot systems are designed to achieve tasks by collaboration. A key requirement for their efficient operation is good coordination and reciprocal collision avoidance. Moving a vehicle on a collision-free path is a well-studied problem in robot navigation. The work in [4], [6] and [8] presents representative examples of collision avoidance methods for single mobile robots. Basically, similar approaches as in the single robot cases can be applied in the context of collision avoidance for multiple robots. However, the increase in robot density and collaborative interaction needs methods that scale well with the number of robots. The collision avoidance approaches are extended in [11] among others for multiple robots by decoupling path planning and coordination. Other work investigated potential fields [5] and

Javier Alonso-Mora · Andreas Breitenmoser · Martin Rufli · Roland Siegwart
Autonomous Systems Laboratory (ASL), ETH Zurich, Tannenstrasse 3,
8092 Zurich, Switzerland
e-mail: {jalonso,andrbrei,ruflim,rsiegwart}@ethz.ch

Javier Alonso-Mora · Paul Beardsley
Disney Research Zurich, Clausiusstrasse 49, 8092 Zurich, Switzerland
e-mail: {jalonso,pab}@disneyresearch.com

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 203–216.
springerlink.com

cooperative control laws [14] to direct a group of robots to their objectives while avoiding collisions. Decentralized control helps lowering computational cost and introduces additional robustness and flexibility to the multi-robot system.

In this paper, we develop and formally analyze a new collision avoidance strategy for a group of non-holonomic robots. Mobile robots we see being deployed nowadays in research or industry are mostly non-holonomic. Therefore installations with multiple robots in real world scenarios, such as multiple vacuum cleaners or collaborative monitoring and maintenance vehicles, require collision avoidance methods that take the non-holonomic constraints of the robots into account.

Our approach builds on Optimal Reciprocal Collision Avoidance (ORCA) [2] and extends it toward *non-holonomic* reciprocal collision avoidance. The robots are controlled to stay within a maximum tracking error $\mathscr{E}$ of an ideal holonomic trajectory. Control inputs for optimal tracking are derived from mapping holonomic onto non-holonomic velocities. We focus on differential-drive robots in the following work, even though our approach applies more generally for the class of feedback-linearizable vehicles with non-holonomic kinematics, such as car-like robots or differentially-driven robots with trailer.

Reciprocal Collision Avoidance (RVO) [3], a collaborative collision avoidance method based on velocity obstacles, was reformulated as ORCA [2] and shown to be solved efficiently through a low-dimensional linear program, which results in completeness and a speed-up of the algorithm. Each robot makes a similar collision avoidance reasoning and collision-free motion is guaranteed all time, but holonomic robots are assumed and oscillations in the form of reciprocal dances can occur. The extension in [12] combines both the concepts of basic velocity obstacles and RVO to reduce the amount of oscillations. In addition, robot kinematics and sensor uncertainty are included by enlarging the velocity cones, even though a formal proof of collision-free motion is not given. The work in [15] generalizes RVO for robots with non-holonomic constraints by testing sampled controls for their optimality. As the method requires extensive numeric computation and relies on probabilistic sampling, it may fail to find an existing feasible solution. The latest extension [13] introduces a solution for differential-drive robots by applying ORCA on the robot's virtual center. This is in contrast to our approach of extending the robot's radius, which allows to decrease its extension to zero in crowded scenarios. [13] also relies on the mapping between desired holonomic and non-holonomic velocities, but is different from ours in how it is derived; moreover it further constrains the motion of the robots. Another reactive collision avoidance method for unicycles based on velocity obstacles was presented in [9], where inputs are obtained by a weighted combination of the closest collision in normal and tangential directions.

The paper is organized as follows. We start with the problem formulation in Section 2 and review the main concepts of ORCA. Then the proposed algorithm for collision avoidance in a group of non-holonomic robots is presented in Section 3. In Section 4, we give a formal analysis of the non-holonomic controls that lead to optimal tracking of holonomic velocities and prove collision-free motion. Section 5 demonstrates the method in experiments with up to fourteen robots and shows successful collision avoidance and smooth trajectories.

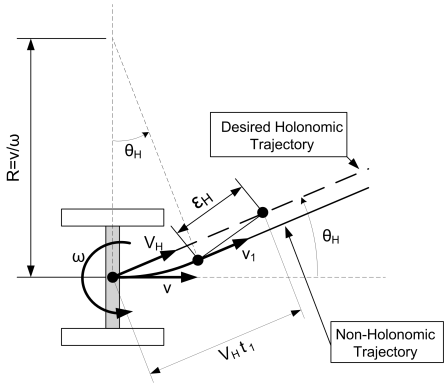


Fig. 1 Non-holonomic tracking error. The holonomic trajectory is tracked by the differential-drive robot moving along the non-holonomic trajectory within tracking error ε_H.

2 Problem Formulation

2.1 Kinematic Model of Differential-Drive Robot

First the kinematic model for the differential-drive robot is introduced. The basic trajectories of the non-holonomic robots considered in this work are defined by two sections, an arc of circumference covered with constant speed v, followed by a straight line path with constant speed v_1, as illustrated in Fig. 1. The basic non-holonomic controls $(v(t), \omega(t))$ consist of the linear and angular velocities

$$v(t) = \begin{cases} v = \omega R, \text{ for } 0 \le t \le t_1 \\ v_1, \text{ for } t > t_1 \end{cases}, \qquad \omega(t) = \begin{cases} \omega, \text{ for } 0 \le t \le t_1 \\ 0, \text{ for } t > t_1 \end{cases}. \tag{1}$$

Note that in our formulation the robots have no constraints in acceleration, nevertheless, these could be easily included by adding to the complexity of the formulation. Although the planned trajectory is a circular sector followed by a straight line segment, the robots perform only a part of the circular segment and then recompute, which results in final trajectories that are much more complex.

The kinematic constraints are given by $|v(t)| \le v_{max,\omega} = v_{max} - |\omega(t)|\frac{l_w}{2}$, $|\omega(t)| \le \omega_{max} = \frac{2v_s^{max}}{l_w K_{vs}}$ and $v_{max} = \frac{v_s^{max}}{K_{vs}}$, where the wheel speed is bounded by $-v_s^{max} \le v_s(t) = \left(v(t) \pm \frac{l_w}{2}\omega(t)\right) K_{vs} \le v_s^{max}$, with $v_s(t)$ the angular velocity of the right and the left wheel respectively, l_w is the distance between the wheels and K_{vs} a conversion factor. The system parameters that are relevant for the locomotion of the e-puck robot (refer to Section 5) are given by: $l_w = 0.0525$ m, $v_s^{max} = 1000$ steps/s, $K_{vs} = 7674.6$ steps/m, $v_{max} = 0.13$ m/s and $\omega_{max} = 4.96$ rad/s.

The *set of non-holonomic controls* $S_{NHC} = \{(v(t), \omega(t)) \,|\, \text{Eq. (1) and kinematic constraints}\}$ is defined as the feasible subset of the controls $(v(t), \omega(t))$ given by Eq. (1), i.e. the controls satisfying the kinematic constraints.

2.2 Set of Allowed Holonomic Velocities

The underlying idea of the approach here presented is that a particular non-holonomic robot is able to track a certain set of holonomic motions within a given maximum tracking error $\mathscr{E}$. Therefore, increasing the radius of each robot by its fixed value $\mathscr{E}$ guarantees collision-free trajectories, even in the case of non-holonomic robots. The tracking error ε_H is quantified by consideration of the robot's kinematics and can be bounded by a certain value $\mathscr{E}$ through limiting the set of holonomic trajectories to be tracked.

In Fig. 1 the trajectories for both holonomic and non-holonomic robots are presented. If the velocity v_1 of the non-holonomic robot in Eq. (1) is fixed to the speed of the holonomic robot V_H, the maximum error in tracking a holonomic trajectory at constant velocity $\mathbf{v}_H = V_H(\cos(\theta_H), \sin(\theta_H))$ is given at time t_1, and represented by ε_H. Note that the tracking error might as well be decreased with a more complex control scheme. However, it never increases under the non-holonomic controls according to Eq. (1). Let us fix $v_1 = V_H$ in the following.

Thus, for a given holonomic velocity $\mathbf{v}_{H_i} = \mathbf{v}_H$ and control inputs (v, ω) at time $t = k\Delta t$, where $k \in \mathbb{N}$ is the iteration index and Δt the time step, the value of the tracking error ε_H is given by simple geometry

$$\begin{aligned}\varepsilon_H^2(v, w, V_H, \theta_H) &= (V_H t_1 - R\sin(\theta_H))^2 + (R(1-\cos(\theta_H)))^2 \\ &= V_H^2 t_1^2 - \frac{2V_H t_1 \sin(\theta_H)}{\omega} v + \frac{2(1-\cos(\theta_H))}{\omega^2} v^2. \end{aligned} \tag{2}$$

For non-holonomic robots and fixed a maximum tracking error $\mathscr{E}$, the *set of allowed holonomic velocities* S_{AHV} is given by the velocities $\mathbf{v}_H$ for which there exists a control input within the set of non-holonomic controls S_{NHC} that guarantees a tracking error lower or equal than the given maximum tracking error $\mathscr{E}$ at all times. The set of allowed holonomic velocities is defined as

$$S_{AHV} = \{\mathbf{v}_H \in \mathbb{R}^2 \mid \exists (v(\tau), \omega(\tau)) \in S_{NHC},\ ||\mathbf{p} + \tau \cdot \mathbf{v}_H - \hat{\mathbf{p}}^k(\tau)|| \leq \mathscr{E}\ \forall \tau \geq 0\}, \tag{3}$$

where $\hat{\mathbf{p}}^k(\tau)$ is the expected robot position at time $k\Delta t + \tau$ if controls $(v(\tau), \omega(\tau))$ are applied at time $k\Delta t$.

In order to obtain smooth trajectories, the time t_1 to achieve the correct orientation θ_H can be fixed to a minimum value T. Note that this value must be at least equal to the time step Δt of the controller. t_1 is kept fixed for the following sections.

In Section 4 the closed form of S_{AHV} and the mapping between the sets S_{AHV} and S_{NHC}, as well as the proof of collision-free motion, are derived.

2.3 Optimal Reciprocal Collision Avoidance

ORCA [2] is a velocity-based approach to collision avoidance that provides a sufficient condition for guaranteeing collision-free motion among multiple holonomic

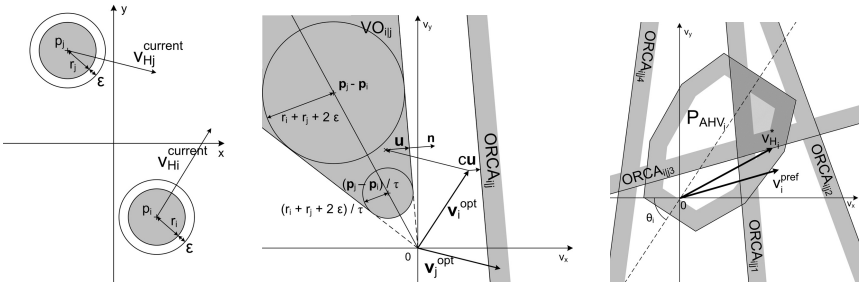


Fig. 2 Left: configuration with two non-holonomic robots. Center: $VO^{\tau}_{i|j}$ and $ORCA^{\tau}_{i|j}$ for a holonomic robot at $\mathbf{p}_i$ with $r_i + \varepsilon$ and $\mathbf{v}^{current}_{H_i}$, generated by a holonomic robot at $\mathbf{p}_j$ with $r_j + \varepsilon$ and $\mathbf{v}^{current}_{H_j}$. Right: constraints in velocity generated by $ORCA^{\tau}_{i|j}$ from multiple robots, together with the set P_{AHV_i} taking into account the kinematics of the robot. The region of collision-free velocities $ORCA^{\tau}_i$ is highlighted and $\mathbf{v}^*_{H_i}$ is displayed.

robots. Given a group of n disk-shaped robots with radius r_i and velocity $\mathbf{v}_i \in \mathbb{R}^2$ at position $\mathbf{p}_i$ in the plane $\mathbb{R}^2$, each robot tries to reach an assigned goal point $\mathbf{g}_i$ by selecting a preferred velocity $\mathbf{v}^{pref}_i \in \mathbb{R}^2$. The objective is to choose an optimal $\mathbf{v}_i$, which lies as close as possible to $\mathbf{v}^{pref}_i$, such that collisions among the robots are avoided for at least a time horizon τ.

In the case of holonomic robots with velocities $\mathbf{v}_H \in \mathbb{R}^2$, the velocity obstacle for robot $i \in [1,n] \subset \mathbb{N}$ with r_i at $\mathbf{p}_i$ induced by any robot $j \in [1,n]$, $j \neq i$, with r_j at $\mathbf{p}_j$ is defined as the set of relative velocities $\bar{\mathbf{v}} = \mathbf{v}_{H_i} - \mathbf{v}_{H_j}$ between robots i and j

$$VO^{\tau}_{i|j} = \left\{ \bar{\mathbf{v}} \,|\, \exists t \in [0,\tau], t \cdot \bar{\mathbf{v}} \in D(\mathbf{p}_j - \mathbf{p}_i, r_i + r_j) \right\}, \tag{4}$$

with $D(\mathbf{p}, r) = \{\mathbf{q} \,|\, \|\mathbf{q} - \mathbf{p}\| < r\}$ the open ball of radius r. The set of collision-free velocities $ORCA^{\tau}_{i|j}$ for robot i with respect to robot j can geometrically be constructed from $VO^{\tau}_{i|j}$ (see Fig. 2 left and center). First, the minimum change

$$\mathbf{u} = (\underset{\bar{\mathbf{v}} \in \partial VO^{\tau}_{i|j}}{argmin} \|\bar{\mathbf{v}} - (\mathbf{v}^{opt}_i - \mathbf{v}^{opt}_j)\|) - (\mathbf{v}^{opt}_i - \mathbf{v}^{opt}_j), \tag{5}$$

which needs to be added to $\bar{\mathbf{v}}$ to avoid a collision, is computed. $\mathbf{v}^{opt}_i$ is the optimization velocity, set to the current velocity $\mathbf{v}^{current}_{H_i}$ of the robot. This gives good results as shown in [2]. Then $ORCA^{\tau}_{i|j} = \{\mathbf{v}_{H_i} | (\mathbf{v}_{H_i} - (\mathbf{v}^{opt}_i + c\mathbf{u})) \cdot \mathbf{n} \geq 0\}$ follows as described in [2]. $\mathbf{n}$ denotes the outward normal of the boundary of $VO^{\tau}_{i|j}$ at $(\mathbf{v}^{opt}_i - \mathbf{v}^{opt}_j) + \mathbf{u}$, and c defines how much each robot gets involved in avoiding a collision. $c = \frac{1}{2}$ means both robots i and j help to equal amounts to avoid colliding with each other; $c = 1$ means robot i fully avoids collisions with a dynamic obstacle j. Likewise, the velocity obstacle can be computed for static obstacles following [2].

The set of collision-free velocities for robot i, $ORCA_i^\tau$, is given by

$$ORCA_i^\tau = S_{AHV_i} \cap \bigcap_{j \neq i} ORCA_{i|j}^\tau, \tag{6}$$

with S_{AHV_i} the set of allowed holonomic velocities under the kinematic constraints of robot i. For holonomic robots, $S_{AHV_i} = D(0, V_{H_i}^{max})$. Fig. 2 on the right shows the set $ORCA_i^\tau$ for a configuration with multiple robots, where S_{AHV_i} is approximated by the convex polygon P_{AHV_i} for a differential-drive robot.

The optimal holonomic velocity for robot i is to be found as

$$\mathbf{v}_{H_i}^* = \underset{\mathbf{v}_{H_i} \in ORCA_i^\tau}{argmin} \|\mathbf{v}_{H_i} - \mathbf{v}_i^{pref}\|. \tag{7}$$

3 NH-ORCA: Optimal Reciprocal Collision Avoidance under Non-Holonomic Constraints

In each time-step NH-ORCA consists of the following three main steps: first, $VO_{i|j}^\tau$ and $ORCA_{i|j}^\tau$ are computed for holonomic robots of radius $r_i + \mathscr{E}_i$, $r_j + \mathscr{E}_j$ at $\mathbf{p}_i$, $\mathbf{p}_j$ with velocity $\mathbf{v}_{H_i}^{current}$, $\mathbf{v}_{H_j}^{current}$. Second, S_{AHV_i} is computed for fixed $\mathscr{E}_i$ and T_i and approximated by a convex polygon P_{AHV_i}. Moreover, $ORCA_i^\tau$ is generated with respect to the neighboring robots and an optimal holonomic velocity is selected from the set of collision-free velocities defined by Eq. (6); thereby, the preferred velocities of the robots are taken into account. This is represented in Fig. 2 where $\mathscr{E} = \mathscr{E}_i = \mathscr{E}_j$. Finally, the selected holonomic velocity is mapped to the corresponding non-holonomic control inputs, which guarantee collision-free motion. A detailed description of the algorithm is provided in Algorithm 1.

The closed-form expression from Eq. (13) (in Section 4) is evaluated to compute the maximum allowed holonomic velocities, this is the set S_{AHV_i}. In general, S_{AHV_i} is not convex for a given T_i. In our implementation of NH-ORCA, the area of S_{AHV_i} is approximated by a convex polygon P_{AHV_i} that lies inside S_{AHV_i}. This simplifies the optimization problem. Note that P_{AHV_i} can be precomputed due to rotational invariance and at each iteration be aligned with the current orientation of the robot. As $ORCA_i^\tau$ is a convex region formed by linear constraints, a quadratic optimization problem with linear constraints is formulated. Eq. (7), where S_{AHV_i} is substituted by P_{AHV_i}, can efficiently be solved by methods from computational geometry. The optimization velocity $\mathbf{v}_i^{opt}$ that is used in the optimization is set to the current holonomic velocity $\mathbf{v}_{H_i}^{current}$ of the agent, but other choices are possible. The mapping to non-holonomic optimal control inputs follows from Eq. (9) (in Section 4).

NH-ORCA can be applied to heterogeneous groups of robots with different kinematic constraints, sizes, maximum tracking errors $\mathscr{E}_i$ and lower bounds T_i.

Algorithm 1. Non-Holonomic Reciprocal Collision Avoidance.

Require: Fixed $\mathscr{E}_i$ and T_i. Group of differential-drive robots $i \in [1,n]$ provided with:
- internal parameters: $\mathbf{p}_i$, $\mathbf{v}_{H_i}^{current}$, θ_i, $\mathbf{v}_i^{pref}$, r_i, $\mathscr{E}_i$, T_i.
- external parameters (obtained from sensing or communication): $\mathbf{p}_j$, $\mathbf{v}_{H_j}^{current}$, $r_j + \mathscr{E}_j$ with $j \neq i$.

Compute $P_{AHV_{i,0}}$ from closed-form expression of $S_{AHV_{i,0}}$ and zero orientation, Eq. (13).
loop
  for $i \in \{1,...,n\}$ **do**
    Compute P_{AHV_i} by rotating $P_{AHV_{i,0}}$ to match orientation θ_i.
    for $j \in \{1,...,n\}$, $j \neq i$ **do**
      Compute $VO_{i|j}^{\tau}$ for holonomic robots of radius $r_i + \mathscr{E}_i$ and $r_j + \mathscr{E}_j$ at p_i and p_j with $\mathbf{v}_{H_i}^{current}$ and $\mathbf{v}_{H_j}^{current}$.
      Compute $ORCA_{i|j}^{\tau}$.
    end for
    Construct $ORCA_i^{\tau} = P_{AHV_i} \cap \bigcap_{i \neq j} ORCA_{i|j}^{\tau}$.
    Compute optimal collision-free holonomic velocity $\mathbf{v}_{H_i}^*$ following Eq. (7).
    Map $\mathbf{v}_{H_i}^*$ to (v_i, ω_i) following Eq. (9).
    Apply controls.
  end for
end loop

4 Formal Analysis

In our analysis the symmetry of the tracking with respect to both axis and its rotational invariance is exploited. Therefore, the considerations are limited to the case of tracking holonomic velocities in $\mathbb{R}_+^2$ and zero orientation of the agent. It is clear that the analysis extends likewise to entire $\mathbb{R}^2$ and general orientation of the robot.

4.1 Selection of Non-Holonomic Controls

In this section, the control inputs (v, ω) for optimal tracking of a given holonomic velocity $\mathbf{v}_H$ are found. The controls for the non-holonomic robot are chosen as those that minimize the tracking error ε_H, while achieving the correct orientation in the fixed given time T. If this is impossible due to the robot's constraints, the robot performs a turn in place by rotating at maximum speed until the correct orientation is reached, i.e. $\omega = \min\left(\frac{\theta_H}{T}, \omega_{max}\right)$. In general, t_1, θ_H and ω are related by $\omega = \frac{\theta_H}{t_1}$.

With everything else fixed, the linear velocity that minimizes Eq. (2) is given by

$$v^* = \frac{V_H t_1 \sin(\theta_H)\omega}{2(1-\cos(\theta_H))} = V_H \frac{\theta_H \sin(\theta_H)}{2(1-\cos(\theta_H))}. \tag{8}$$

The optimal linear velocity might not be feasible due to the limits on the linear and angular velocities. Therefore, the optimal controls are

$$
\begin{aligned}
R_{A1}: \ \omega &= \frac{\theta_H}{T} \leq \omega_{max} \ \text{ and } \ v = v^* \leq v_{max,\omega} \\
R_{A2}: \ \omega &= \frac{\theta_H}{T} \leq \omega_{max} \ \text{ and } \ v = v_{max,\omega} \\
R_B: \ \omega &= \omega_{max} \ \text{ and } \ v = 0.
\end{aligned} \tag{9}
$$

If the optimal controls are chosen, the maximum tracking error $\varepsilon_H^2(\mathbf{v}_H)$ committed in each of the regions are derived from Eq. (2) and Eq. (8) and given by

$$R_{A1}: \ \varepsilon_H^2 = \left(\frac{2(1-\cos(\theta_H)) - \sin^2(\theta_H)}{2(1-cos(\theta_H)} \right) T^2 V_H^2 \tag{10}$$

$$R_{A2}: \ \varepsilon_H^2 = V_H^2 T^2 - \frac{2V_H T^2 \sin(\theta_H)}{\theta_H} v_{max,\omega} + \frac{2T^2(1-\cos(\theta_H))}{\theta_H^2} v_{max,\omega}^2 \tag{11}$$

$$R_B: \ \varepsilon_H = V_H t_1 = V_H \frac{\theta_H}{\omega_{max}}. \tag{12}$$

4.2 Construction of S_{AHV}

The closed form of the set of allowed holonomic velocities S_{AHV} is derived for fixed $\mathscr{E}$ and T in this section. For a given orientation θ_H of the holonomic velocity, the maximum holonomic speed V_H that can be successfully tracked with $\varepsilon_H \leq \mathscr{E}$ is computed (see Fig. 3). Note that for feasibility, the maximum holonomic speed is limited by the robot's maximum linear velocity $V_H \leq v_{max}$. Otherwise the tracking error would increase after time t_1.

Theorem 1. *Both the optimal linear velocity $v(V_H)$ and the tracking error $\varepsilon_H(V_H)$ are monotonically increasing with respect to the holonomic speed V_H for fixed θ_H.*

Proof. From Eq. (8)-(9) it directly follows that, with everything else fixed, the optimal linear velocity v is monotonically increasing with respect to the holonomic speed V_H. The monotonicity of $\varepsilon_H(V_H)$ is derived from Eq. (10)-(12). Due to limited space, the proof is omitted. □

Theorem 2. *The maximum holonomic speed V_H^{max} that can be tracked with $\varepsilon_H \leq \mathscr{E}$ for a fixed θ_H is given by*

$$V_H^{max} = \begin{cases} \min\left(\frac{\mathscr{E}}{T} \sqrt{\frac{2(1-\cos(\theta_H))}{2(1-\cos(\theta_H)) - \sin^2(\theta_H)}}, v_{max} \right) & \text{if} \quad \begin{cases} \frac{\theta_H}{T} \leq \omega_{max} \\ v_{\mathscr{E}}^* \leq v_{max,\omega} \end{cases} \\ \min\left(\frac{-\beta + \sqrt{\beta^2 - 4\alpha\gamma}}{2\gamma}, v_{max} \right) & \text{if} \quad \begin{cases} \frac{\theta_H}{T} \leq \omega_{max} \\ v_{\mathscr{E}}^* \geq v_{max,\omega} \end{cases} \\ \min\left(\frac{\mathscr{E}\omega_{max}}{\theta_H}, v_{max} \right) & \text{if} \quad \frac{\theta_H}{T} \geq \omega_{max}, \end{cases} \tag{13}$$

where $v^*_{\mathscr{E}}$, α, β, γ *are given by*

$$v^*_{\mathscr{E}} = \frac{\mathscr{E}}{T}\frac{\theta_H \sin(\theta_H)}{2(1-\cos(\theta_H))}\sqrt{\frac{2(1-\cos(\theta_H))}{2(1-\cos(\theta_H))-\sin^2(\theta_H)}}, \tag{14}$$

$$\alpha = T^2, \quad \beta = -\frac{2T^2\sin(\theta_H)}{\theta_H}v_{max,\omega}, \quad \gamma = \frac{2T^2(1-\cos(\theta_H))}{\theta_H^2}v^2_{max,\omega} - \mathscr{E}^2. \tag{15}$$

Proof. Denote $v_{\mathbf{v}_H^{max}}$ and $\omega_{\mathbf{v}_H^{max}}$ the linear and angular velocities for optimal tracking of the maximum holonomic velocity $\mathbf{v}_H^{max}$, given by V_H^{max} and θ_H.

The proof is divided for regions R_{A1}, R_{A2} and R_B. Recall from Theorem 1 that, $v(V_H)$ and $\varepsilon_H(V_H)$ are monotonically increasing with respect to V_H. This is implicitly used in the proof. In all cases the value of the maximum holonomic speed V_H^{max} must be limited to v_{max} following $V_H \leq v_{max}$.

- Region R_{A1}: Assume $\omega_{\mathbf{v}_H^{max}} = \frac{\theta_H}{T} < \omega_{max}$. Consider the case where $v_{\mathbf{v}_H^{max}} < v_{max,\omega}$. The holonomic speed which gives a tracking error equal to the maximum $\varepsilon_H = \mathscr{E}$ is found by solving Eq. (10), which gives the top value V_H^{max} of Eq. (13). The linear velocity for optimal tracking of $\mathbf{v}_H^{max}$ is then given by Eq. (14), obtained by substituting V_H^{max} into Eq. (8), which is feasible if $v^*_{\mathscr{E}} \leq v_{max,\omega} = v_{max} - \frac{\theta_H l_W}{2T}$. If this holds, $v_{\mathbf{v}_H^{max}} = v^*_{\mathscr{E}}$. Otherwise, the solution is found in region R_{A2}.

- Region R_{A2}: Assume $\omega_{\mathbf{v}_H^{max}} = \frac{\theta_H}{T} < \omega_{max}$. Consider the case where $v_{\mathbf{v}_H^{max}} = v_{max,\omega}$. The tracking error is given by Eq. (11) and from Theorem 1, the maximum holonomic speed V_H^{max} satisfies $\varepsilon_H = \mathscr{E}$. The solution is given by solving, $0 = \alpha(V_H^{max})^2 + \beta V_H^{max} + \gamma$, where from Eq. (11), $\alpha = T^2$, $\beta = -\frac{2T^2\sin(\theta_H)}{\theta_H}v_{max,\omega}$ and $\gamma = \frac{2T^2(1-\cos(\theta_H))}{\theta_H^2}v^2_{max,\omega} - \mathscr{E}^2$. From Theorem 1, the maximum holonomic speed is given by the solution of largest value, hence the middle value V_H^{max} of Eq. (13). The associated linear velocity for optimal tracking is given by $v_{\mathbf{v}_H^{max}} = v_{max,\omega} = v_{max} - \frac{\theta_H l_W}{2T}$. Finally, the value of the maximum holonomic speed V_H^{max} must be limited to v_{max} following $V_H \leq v_{max}$.

- Region R_B: Assume $\omega_{\mathbf{v}_H^{max}} = \omega_{max}$. In this case, a rotation in place is performed. Therefore $v_{\mathbf{v}_H^{max}} = 0$. Recalling Eq. (12) and Theorem 1, the maximum holonomic speed V_H^{max} from Eq. (13) bottom is obtained. □

Similar results are derived for the case where the angular velocity ω is limited by $\hat{\omega} < \omega_{max}$. This leads to smoother trajectories and, independently of the chosen T, in place rotations are avoided.

Theorem 3. *Consider* $\omega \leq \hat{\omega} < \omega_{max}$. *The maximum holonomic speed* V_H^{max} *that can be tracked with* $\varepsilon_H \leq \mathscr{E}$ *for a fixed* θ_H *is given by*

$$V_H^{max} = \begin{cases} \min\left(\frac{\mathscr{E}}{T}\sqrt{\frac{2(1-\cos(\theta_H))}{2(1-\cos(\theta_H))-\sin^2(\theta_H)}}, v_{max}\right) & \text{if} \quad \begin{cases} \frac{\theta_H}{T} \leq \hat{\omega} \\ v^*_{\mathscr{E}} \leq v_{max,\omega} \end{cases} \\ \min\left(\frac{-\beta+\sqrt{\beta^2-4\alpha\gamma}}{2\gamma}, v_{max}\right) & \text{if} \quad \begin{cases} \frac{\theta_H}{T} \leq \hat{\omega} \\ v^*_{\mathscr{E}} \geq v_{max,\omega} \end{cases} \\ \min\left(\frac{\mathscr{E}\hat{\omega}}{\theta_H}\sqrt{\frac{2(1-\cos(\theta_H))}{2(1-\cos(\theta_H))-\sin^2(\theta_H)}}, v_{max}\right) & \text{if} \quad \begin{cases} \frac{\theta_H}{T} \geq \hat{\omega} \\ v^*_{\mathscr{E}} \leq v_{max,\omega} \end{cases} \\ \min\left(\frac{-\hat{\beta}+\sqrt{\hat{\beta}^2-4\hat{\alpha}\hat{\gamma}}}{2\hat{\gamma}}, v_{max}\right) & \text{if} \quad \begin{cases} \frac{\theta_H}{T} \geq \hat{\omega} \\ v^*_{\mathscr{E}} \geq v_{max,\omega} \end{cases} \end{cases} \tag{16}$$

where $v^*_{\mathscr{E}}$, α, β, γ *are given by Eq. (14) and (15).* $\hat{\alpha}$, $\hat{\beta}$, $\hat{\gamma}$ *are given by*

$$\hat{\alpha} = \frac{\theta_H^2}{\hat{\omega}^2}, \quad \hat{\beta} = -\frac{2\theta_H \sin(\theta_H)}{\hat{\omega}^2} v_{max,\omega}, \quad \hat{\gamma} = \frac{2(1-\cos(\theta_H))}{\hat{\omega}^2} v_{max,\omega}^2 - \mathscr{E}^2. \tag{17}$$

Proof. The proof is analogous to that of Theorem 2, where the optimal controls are given by $\omega = \min(\frac{\theta_H}{T}, \hat{\omega})$ and $v = \min(v^*, v_{max,\omega})$. □

Remark 1 *Maximal* S_{AHV}. The maximal set of allowed holonomic velocities S^{max}_{AHV} is given by a maximization of the maximal holonomic speed V_H^{max} over T for a fixed orientation θ_H, $S^{max}_{AHV} = \bigcup_{T\in[\Delta t,\infty)} S_{AHV}$. In this case the time T is not constant, but varies as a function of the orientation θ_H.

Remark 2 *Polygonal approximation* P_{AHV}. Due to the particular non-convex shape of the S_{AHV} two options are described. First, the best approximation is obtained by dividing S_{AHV} in two complementary regions, one for forward and one for backward driving. Then, the problem is solved for one region (the one pointing towards the desired goal) and if unfeasible, for the opposite region in a second step. This region is represented by $P_{AHV,A}$ in Fig. 3. Alternatively, a faster but more restrictive implementation is obtained by using the biggest rectangle contained inside S_{AHV}. This region is represented by $P_{AHV,B}$ in Fig. 3 on the right.

Remark 3 *Behavior in the limits.* Two limit cases might be considered:

- Limit $T \to 0$. For $\theta_H = 0$ trajectories are straight lines; in fact, $\omega = 0$ holds independent of T and therefore perfect tracking is achieved for $V_H \leq v_{max}$. For $\theta_H \in (0, \frac{\pi}{2}]$ and fixed θ_H, $\frac{\theta_H}{T} \to \infty$ is obtained; therefore, rotation in place with $\omega = \omega_{max}$ and $v = 0$ is always the chosen trajectory. This reduces to time optimal trajectories, each composed of straight line segments alternating with turns in place as seen in [1].
- Limit $\mathscr{E} \to 0$. For $\theta_H = 0$ trajectories are straight lines; again, $\omega = 0$ holds independent of T and therefore perfect tracking is achieved for $V_H \leq v_{max}$. For $\theta_H \in (0, \frac{\pi}{2}]$, it can be seen from Theorem 2 that trajectories are reduced to turning in place at angular velocity $\omega = \min\left(\frac{\theta_H}{T}, \omega_{max}\right)$ and $v = 0$.

Remark 4 *Variable maximum tracking error* $\mathscr{E}$. NH-ORCA guarantees collision-free trajectories for non-holonomic robots, that is $r_i + r_j \leq d(\mathbf{p}_i, \mathbf{p}_j)$. To guarantee feasibility of the computation of the $VO^{\tau}_{i|j}$, $r_i + r_j + \mathscr{E}_i + \mathscr{E}_j \leq d(\mathbf{p}_i, \mathbf{p}_j)$ must be

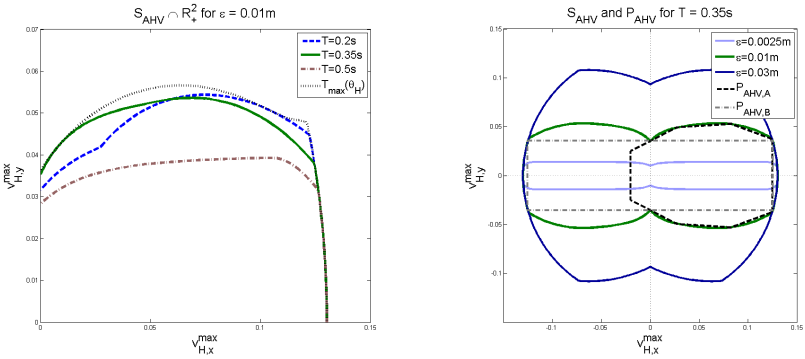


Fig. 3 Left: S_{AHV} for fixed $\mathscr{E}$ and varying T. $T_{max}(\theta_H)$ denotes the variable T that results in the maximal set S^{max}_{AHV}. Right: S_{AHV} for fixed T and varying $\mathscr{E}$. Two polygonal approximations $P_{AHV,A}$ and $P_{AHV,B}$ are shown for $\mathscr{E} = 0.01$ m and $T = 0.35$ s.

satisfied, i.e. the extended radii of the robots must not be in collision. This might happen for fixed $\mathscr{E}_i$ and $\omega_i \neq 0$ but is assuredly avoided by having $\mathscr{E}_i$ and $\mathscr{E}_j$ stepwise decreasing when robots are close to each other.

4.3 Collision-Free Motion

Finally, the proof that NH-ORCA guarantees collision-free motions among multiple non-holonomic robots is presented.

Theorem 4. *The trajectories of all robots are guaranteed to be collision-free.*

Proof. First, planned trajectories for the holonomic robots of radius $r_i + \mathscr{E}_i$ are collision-free, if solutions of ORCA exist, as proven in [2]. Otherwise the constraints given by $ORCA^{\tau}_{i|j}$ must be relaxed by decreasing τ until the problem becomes feasible, thus becoming a 3D optimization [2]. Second, planned trajectories for non-holonomic robots stay within distance $\mathscr{E}_i$ of the planned trajectories for extended holonomic robots, if $T_i \geq \Delta t$. Note that this only guarantees that the distance between two non-holonomic agents is greater than the sum of their radii. To guarantee feasibility of the velocity obstacles computation, and thus completion of the method, Remark 4 must hold in addition.

Trajectories planned for the non-holonomic robot are collision-free. Due to the time-discrete implementation, after each time-step a new collision-free trajectory is computed. Therefore, the trajectories of all agents, given as concatenation of segments, are collision-free. □

Remark 5 *Deadlocks.* NH-ORCA guarantees collision-free trajectories for non-holonomic robots but convergence to a goal destination is not fully guaranteed.

While robots are in movement, deadlocks will not appear (as seen in our experiments in Section 5. Nevertheless, when robots reach their goal, their behavior is close to that of static obstacles. If they are approached by another robot, a deadlock situation may result as the robot's velocity that is closest to its preferred velocity might become zero in order to avoid collisions. This is inherited from the original method for holonomic agents [2] and can be resolved by waypoint navigation [7].

5 Experimental Results

We have evaluated the proposed collision avoidance method and the theoretical results by experiments with real robots. A group of fourteen e-puck robots [10] is used in the experiments. The e-puck is a small disk-shaped differential-drive robot. To enable reliable communication and tracking of the e-pucks, the robots were enhanced with a generic radio receiver and eight infrared LEDs. Red-colored disks were further added on top of each robot for better visual appearance. The following parameters for the NH-ORCA computation are selected: $\mathscr{E} = 0.01$ m, $T = 0.35$ s, $\tau = 7$ s, $V^{pref} = 0.1$ m/s and $r = 0.05$ m the radius of the modified e-puck.

The test setup consists of a central workstation with radio transmitter and an overhead camera mounted on a frame above a flat floor plate of 1.2 m x 1.4 m. The robots' positions and orientations are detected and read into the workstation, where the NH-ORCA is computed for each robot in a decentralized way. The resulting velocities are then broadcasted to the e-pucks in each iteration. The e-puck robots and the workstation form a closed control loop running at a frequency of 10 Hz.

The results of two experiments are presented, which confirm the theoretical findings from Section 4. In the first experiment four e-puck robots are placed in square shape and consecutively exchange positions with each other. Fig. 4 on the left illustrates a subsequence of the robots exchanging positions in diagonal directions. On the right, the trajectories for two out of the four robots are shown when moving along the square's vertical edge to swap positions. As can be seen from the trajectories of the first experiment, not only collision-free but also smooth and visually appealing motions are obtained for the differential-drive robots with the NH-ORCA algorithm.

In cases of symmetry and in order to avoid reciprocal dances [12], the closest point on the velocity obstacle $VO^{\tau}_{i|j}$ is selected clockwise for Eq. (5). This gives preference to right-side avoidance in cases of full symmetry.

In the second experiment, fourteen e-pucks are lined up on a circle and move all together to their antipodal positions on the circle's boundary. This experiment demonstrates that the distributed NH-ORCA algorithm scales with the number of robots, and that it can moreover be applied without any change in the set of parameters for scenarios with many robots (the same parameters as in the first experiment with only four robots are used). The robots successfully solve a very crowded scenario while avoiding collisions at all times. In such scenarios with many robots, a slow-down of the robots can be noticed in areas of increased robot density. This

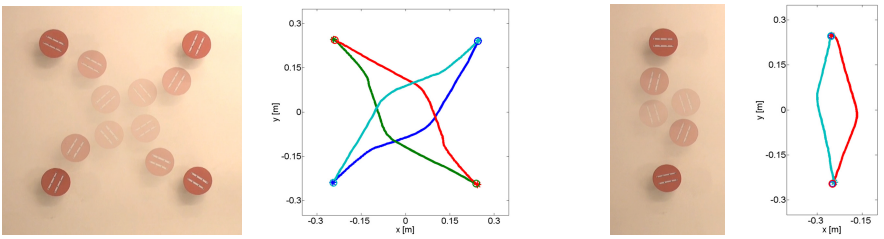


Fig. 4 Experiment 1 with four e-puck robots. Left: e-pucks exchanging positions in diagonal direction. Right: e-pucks exchanging positions vertically. Both sequences and trajectories are smooth and collision-free.

Fig. 5 Experiment 2 with fourteen e-puck robots. The sequence shows collision-free transition of the e-pucks through the circle center to the antipodal position on the circle's boundary.

results from the stronger constraints on the feasible set of velocities, and is in correspondence with Theorem 2 and Remarks 3 and 4 (tendency of increasingly turning in place).

In cases where the optimization becomes unfeasible, zero inputs can be selected for the robots. Alternatively, implementation of Remark 4 and of the 3D optimization guarantee feasibility while leading to a decrease in the time of collision τ. As a result, faster motions are achieved for the robots in Experiment 2. The robots can get infinitely close from the fact that no safety area is added, but collisions are avoided. Further experiments studied different scenarios, including scenarios with dynamic obstacles. A video showing the conducted experiments in full length accompanies the paper.

6 Conclusion and Outlook

In this work, a fast and distributed method for local collision avoidance among non-holonomic robots, so-called NH-ORCA, is presented on the basis of multiple differential-drive robots. Formal proofs of collision-free motion (valid both for continuous and discrete control) are derived and several experiments are performed verifying the results. NH-ORCA achieves smooth and visually appealing trajectories for non-holonomic robots, as demonstrated in the first experiment. Furthermore, the method successfully deals with very crowded situations, as shown in the second experiment with a larger group of fourteen robots.

In future work, it would be interesting to extend the method here presented to other non-holonomic vehicle dynamics. We believe this can be achieved by modifying the set of allowed holonomic velocities S_{AHV}. In accordance with [2], another line of research is to combine NH-ORCA with global path planning and to look closer at the avoidance of deadlock situations. For less controlled environments, or full integration of sensing and actuation, the method should be extended to compensate for uncertainties. Eventually, the method could be generalized for higher dimension and applied to underwater or aerial robots in $\mathbb{R}^3$.

References

1. Balkcom, D.J., Mason, M.T.: Time Optimal Trajectories for Bounded Velocity Differential Drive Vehicles. Int. J. Robot. Res. 21(3), 199–218 (2002)
2. van den Berg, J., Guy, S.J., Lin, M.C., Manocha, D.: Reciprocal n-body Collision Avoidance. In: Proc. Int. Symp. Robot. Res. (2009)
3. van den Berg, J., Lin, M.C., Manocha, D.: Reciprocal Velocity Obstacles for real-time multi-agent navigation. In: Proc. IEEE Int. Conf. Robot. Autom., pp. 1928–1935 (2008)
4. Borenstein, J., Koren, Y.: The vector field histogram - fast obstacle avoidance for mobile robots. IEEE Trans. Robot. Autom. (7), 278–288 (1991)
5. Chang, D.E., Shadden, S., Marsden, J.E., Olfati Saber, R.: Collision Avoidance for Multiple Agent Systems. In: Proc. IEEE Conf. Dec. Contr. (2003)
6. Fiorini, P., Shiller, Z.: Motion planning in dynamic environments using velocity obstacles. Int. J. Robot. Res. 17(7), 760–772 (1998)
7. Guy, S.J., Chhugani, J., Kim, C., Satish, N., Lin, M., Manocha, D., Dubey, P.: ClearPath: Highly Parallel Collision Avoidance for Multi-Agent Simulation. In: Proc. ACM SIGGRAPH Eurographics Symp. Comput. Animat. (2009)
8. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. Int. J. Robot. Res. (5), 90–98 (1986)
9. Lalish, E., Morgansen, K.A.: Decentralized Reactive Collision Avoidance for Multivehicle Systems. In: Proc. IEEE Conf. Decision and Control (2008)
10. Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.-C., Floreano, D., Martinoli, A.: The e-puck, a Robot Designed for Education in Engineering. In: Proc. Conf. Aut. Rob. Syst. Compet., pp. 59–65 (2009)
11. Siméon, T., Leroy, S., Laumond, J.-P.: Path coordination for multiple mobile robots: a resolution complete algorithm. IEEE Trans. Robot. Autom. 18(1) (2002)
12. Snape, J., van den Berg, J., Guy, S.J., Manocha, D.: Independent navigation of multiple mobile robots with hybrid reciprocal velocity obstacles. In: Proc. IEEE Int. Conf. Intell. Rob. Syst., pp. 5917–5922 (2009)
13. Snape, J., van den Berg, J., Guy, S.J., Manocha, D.: S-ORCA: Guaranteeing Smooth and Collision-Free Multi-Robot Navigation Under Differential-Drive Constraints. In: Proc. IEEE Int. Conf. Robot. Autom. (2010)
14. Stipanović, D.M., Hokayem, P.F., Spong, M.W., Šiljak, D.D.: Cooperative Avoidance Control for Multiagent Systems. ASME J. Dyn. Sys. Meas. Control 129(5), 699–707 (2007)
15. Wilkie, D., van den Berg, J., Manocha, D.: Generalized velocity obstacles. In: Proc. IEEE Int. Conf. Intell. Rob. Syst., pp. 5573–5578 (2009)

Visual-Aided Guidance for the Maintenance of Multirobot Formations

Patricio Nebot and Enric Cervera

Abstract. Among other multirobot formation topics, one of the most important is the maintenance of the initial formation while the robots are moving through the environment. This paper presents a new approach based on the cooperation among a team of heterogeneous robots for the maintenance of multirobot formations. In this case, one of the robots, the *conductor*, drives the formation and the rest of robots must follow it maintaining the formation. To do that, the use of "virtual points" and Bezier curves are introduced. Moreover, in order to solve the odometry problem of the robots, visual information is introduced, which allows one robot, the *leader*, to monitor the positions of the "blind" robots and help them to maintain the formation. In this way, a new method based on visual-aided guidance for the maintenance of formations have been developed. The results prove that this new approach is suitable for the maintenance of formations of multiple robots.

1 Introduction

Coordination among a group of robots can be very useful for many applications. One of the most important task is, in motion coordination, how to move a team of robots in an ordered way, such as maintaining a predefined formation.

One of the first approximations to multirobot formations is the leader-follower approach [3, 4, 7, 8] where one robot is selected as the leader and must be followed by the rest of robots. This task is usually implemented as only a single robot following some other autonomous agent. The follower must attempt to track the leader even though the leader undergoes possibly rapid random motion changes [5].

Patricio Nebot · Enric Cervera
Robotic Intelligence Laboratory
Universitat Jaume I
E-12071 Castellón de la Plana, Spain
e-mail: {pnebot,ecervera}@uji.es

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 217–228.
springerlink.com

In this field, two works [2, 11] were developed at the Robotic Intelligence Lab of the Jaume I University. These works were useful as the basis for the work developed in this paper.

Chiem [2] presented a simple method for the development of leader-follower formations of multiple autonomous robots. Each follower robot estimates the position and orientation of its leader with a color-tracking vision system, and builds a Bezier curve that describes the trajectory between its current position, and the position of the leader robot. This approach can be extended to more follower robots, while keeping a line formation. Also, Chiem stated that by the introduction of virtual points formations of different shapes can be generated, but it was not implemented.

Renaud [11] worked on robot formation control strategies based on a vision-based follow-the-leader scenario, but he concentrated on the reliability of the system. In that way, perception is enhanced by the control of a pan-tilt-zoom camera, which gives the follower robot a large field of view and improvement of the leader detection.

The application described in this paper is based on the previous ones but extended to multiple types of formations and groups of heterogeneous robots by using virtual points as suggested in [2]. The main idea is to make it feasible for a heterogeneous team of four robots to navigate through an environment in such a way that the robots with sensory power help the robots without it.

Before continuing, it is necessary to define some specific terms. As explained in [1, 6], after analyzing different types of geometric formations that multirobots can create, they agree that a geometric formation consists of three main parts: conductor, leader and followers. *Conductor* is the robot at the head of a group in a formation, and is the responsible of leading the group and all the other robots will follow it. *Leader* is the robot who takes the decisions about the formation. Finally *Follower* is any robot in a formation except the conductor, including the leader.

In order to get the robots moving in a coordinated way, a *conductor-referenced system* is used, but applying displacements of the conductor's position, in the form of "virtual points", to create the desired formation. That is, each robot will follow a virtual point, which is determined by the position of the conductor at any time. To determine the relative location of the robots, the leader uses the visual information obtained from the pan-tilt-zoom on-board camera. Camera images are used to detect other robots and to determine the relative position of the detected robot and its orientation with respect to the leader. Each robot carries a colored target, as it can be seen in [10], that helps the leader to recognize it and calculate their position and orientation. Moreover, the zoom is used to enhance the perception and get a higher accuracy and a larger field of view.

The control of the maintenance of the formation while the robots are moving is performed using a decentralized process, where each follower decides which movements must be performed in order to follow the movements of the conductor. The conductor is specialized in navigation because it is using a laser range-finder to build maps and achieve the localization and navigation tasks. The actual position of the conductor while it is moving is sent to every follower. When receiving the position, each follower calculates the virtual point that it must follow. Once the virtual point

is calculated, the follower computes the trajectory it must follow in order to arrive from the current position to the estimated conductor relative position. For the calculation of this trajectory, Bezier curves are considered. Moreover, as it will be seen next, when the robots have problems due to the odometry errors, the leader uses the visual information to place the robots in the formation again.

2 Design of the Application

The main objective in this paper is to develop a system able to maintain a formation while the robots are moving. It is important to remark that the available team for the development of the application is composed of robots with different capabilities. All the robots of the team are Pioneers-2, but each one with different capabilities. Thus, there is one robot with the camera, which also plays the role of leader. There is one robot which has a laser range-finder and plays the role of conductor, that is, it is in charge of driving the team by a previously defined path. Finally, there are also two plain robots, which are the *followers*, and their only mission is to follow the conductor and obey the orders of the leader. In this case, the leader also acts as follower, because it must maintain the formation in order to get the information for the plain robots, so it must follow the conductor as well. Moreover, all the robots have the capacity to communicate by means of a wireless network.

The formation control is developed in Acromovi [9], a framework specifically designed for the development of distributed applications for teams of heterogeneous mobile robots. The software architecture gives us the ease of development of cooperative tasks among robots, using an agent-based platform. In particular, communication between robots can be easily integrated to the control scheme.

In the application, the conductor indicates the movements to be followed by the rest of robots in the formation and each follower decides which movements it must perform in order to follow the movements of the conductor. At each step, the actual position of the conductor is sent to every follower. The follower computes the trajectory to follow in order to get from its current position to the virtual point derived from the conductor position.

With this approach the robots are able to move in formation, in simulation environments, where the odometry error is controlled. When this approach is used in physical robots, the odometry errors make it useless. To improve the performance of the approach, a new agent is introduced in the system, the camera that is carried by the leader is used to reduce the odometry errors of the robots while they are following the conductor. The leader can detect the position of the conductor and the rest of robots and at the same time that the formation is moving, it monitors the position of each robot and indicates to them the possible variations that each one must perform in its movement in order to maintain the formation. This is repeated until the robots arrive at the desired position.

3 Odometry-Based Formation Control

In this section it is described how a formation can be maintained while the robots are moving in the environment. The formation is composed of a maximum of four robots, where one of them, the conductor, is equipped with a laser range-finder that allows it to navigate in buildings and follow a path determined from the map of the building, the goal point, and the obstacles.

In the formation, the position of the robot followers can be controlled by the position of the conductor if all the robots are arranged in a line formation. In other cases, when the followers are positioned to the right or left of the conductor, virtual points are added to the system, as pointed in [2]. These ”virtual points” are calculated by applying a displacement in the conductor position to the left or to the right, depending on the desired formation. The followers, in this case, instead of following the conductor must follow these virtual points. This arrangement is shown in figure 1.

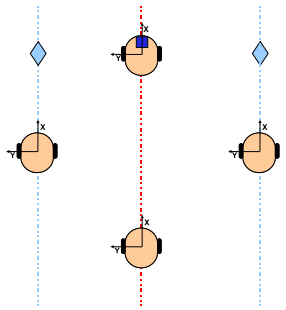


Fig. 1 Formation using virtual points as reference for followers

Following this approach, it is desired that robots perform the task using only the odometry in conjunction with communication to estimate the position of the conductor with respect to each follower.

The conductor, by means of a localization agent is always correctly localized in the map, and it is assumed that it will follow a predefined trajectory, with a constant, known linear velocity. In order for another robot to follow the conductor, the linear and angular velocities need to be computed at each time step. It must be noted that the linear velocity of the follower robots is not constant, due to the different radius of their respective trajectories or because their position may be relatively further back or further forward in the formation. Thus, the linear velocity can be computed by simply adding a gain factor proportional to the distance to the conductor robot. The angular velocity can be computed by means of the curvature of the Bezier trajectory in the origin position (P_0). These velocities are updated at each time step, when the new position of the conductor is sent to all the followers.

In order to calculate the linear and angular velocities that allow the followers to move following the movement of the conductor and maintaining the formation, it is

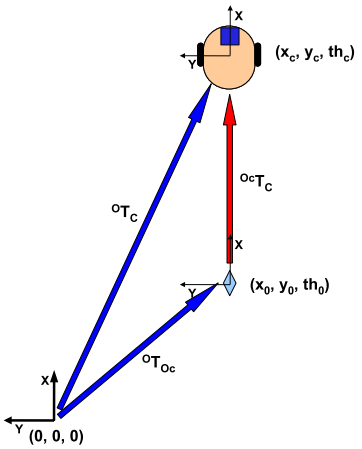


Fig. 2 Calculation of the local position of the conductor

necessary to construct the corresponding Bezier curve that defines the trajectory to be followed by the robot. To compute the Bezier curve two positions are needed, the current position of the follower and the current position of the conductor in relation to the follower.

The conductor, at each step sends its own position to the followers. The conductor is always localized in the map by means of a localization agent, so, when it requires information about its position, this is done with the coordinates of the map. The followers are not localized on the map, so the only information they have available is their position on their own local system which is determined by the origin that is fixed by their initial position. So, the conductor must transform their global position into a position in its local system. In figure 2, the necessary relationships to transform the global position into a local position can be seen.

From the figure 2, it can be deducted that to calculate the position in the local system it is necessary to know the origin of the trajectory in the global system and the actual position of the robot in the global system.

$$^{Oc}T_C = \left(^{O}T_{Oc}\right)^{-1} \cdot {}^{O}T_C \tag{1}$$

where $^{Oc}T_C$ is the translation matrix representing the actual position of the conductor in the local system, $^{O}T_{Oc}$ is the translation matrix representing the origin of the trajectory in the global system, and $^{O}T_C$ is the translation matrix representing the actual position of the conductor in the global system. This last position is updated continuously by the Localization task so that the robot have always its current position on the map.

The conductor sends its current position calculated in this way to each follower. When the followers receive the position of the conductor, they need to calculate the

position of the conductor in relation to themselves in order to use that information to generate the Bezier curve necessary to calculate the velocities which allow them to move maintaining the formation.

Each follower knows its original displacement in the formation. This information in conjunction with the current position of the follower and the current position of the conductor allows the robots to calculate the relative position of the conductor in relation with the follower. In figure 3, the necessary relationships to compute the conductor position are shown.

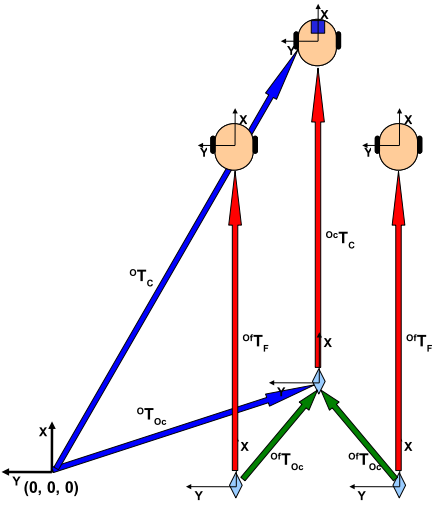


Fig. 3 Calculation of the conductor's position in relation to the followers

In this case, from the figure, the following equation to calculate the conductor position in relation with the follower ${}^{F}T_{C}$ can be obtained,

$$ {}^{F}T_{C} = \left({}^{Of}T_{F}\right)^{-1} \cdot {}^{Of}T_{Oc} \cdot {}^{Oc}T_{C} \tag{2} $$

where ${}^{Of}T_{F}$ is the translation matrix representing the position of the follower in its local system, ${}^{Of}T_{Oc}$ is the translation matrix representing the position of the origin of the conductor's trajectory in relation to the origin of the follower's trajectory. This value is indicated by the displacements assigned to the position of each robot in the creation of the formation. And finally, ${}^{Oc}T_{C}$ is the translation matrix representing the position of the conductor in its local system. This is the position that the conductor sends every step to the followers.

Once this position is calculated, it is possible to compute the Bezier curve between the current position of the follower and the position calculated for the conductor. It is at this moment when the virtual points are calculated, if applicable, applying the following formulas,

$$\begin{aligned} x &= x - \sin(\delta) \cdot dy \\ y &= y + \cos(\delta) \cdot dy \end{aligned} \tag{3}$$

where (x, y, δ) is the position of the conductor in relation to the follower, and dy is the displacement in the y axis to be applied. This generates a new point, the virtual point, which will be used to compute the Bezier curve that drives the movement of the follower.

From the Bezier curve computed, the linear and angular velocities can be obtained. The angular velocity can be computed from the curvature of the curve, and the linear velocity is computed in order to maintain the distance from the conductor, applying a gain factor proportional to the current distance from the conductor robot.

This approach has been tested in simulation and in real robots with different results. In figure 4, some examples executed using the simulator are presented together with the formation in which the robots are organized.

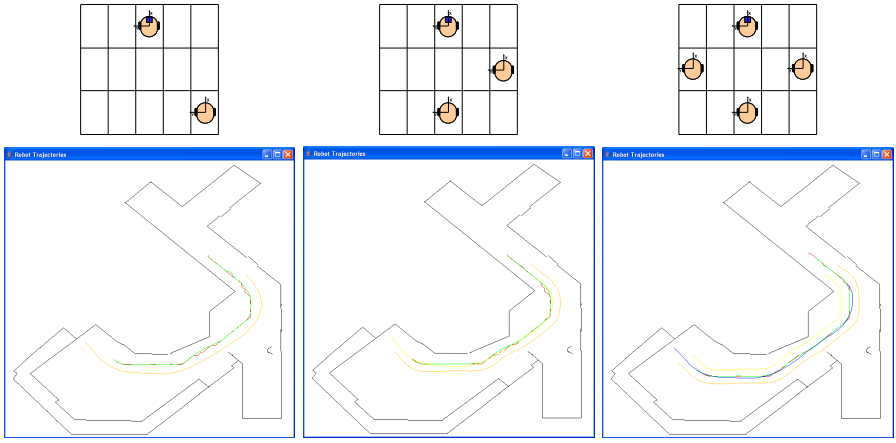


Fig. 4 Examples of executions of the application in the simulator with formations with different number of robots

As it can be seen, the odometry-based approach explained in this section is suitable for the development of applications using multirobot formations. From the different examples, it can be infered that the behaviour of the robots is as desired, and all the follower robots can follow the movements of the conductor robot by the utilization of virtual points. Moreover, it can be noticed that the robots follow the predefined path imposed by the conductor while maintaining the formation at all times.

After testing in the simulator with different formation shapes and verifying that the approach produces the desired behaviour, it was the turn of testing that approach in real robots. The change from the simulation to the real robots is automatic, because no modification in the system or in the code is needed.

However, after several tests, it was noticed that something was wrong in the execution in real robots. In the simplest case of formation, where one robot must follow the conductor, the follower was not able to follow the conductor as desired. The problem was due to the odometry errors of the robots. In simulation, no odometry errors were presented, but when the application was executed in real robots, these errors meant that it was not suitable for the execution in real robots. These odometry errors influence in the behaviour of the follower, making it impossible for it to follow the conductor maintaining the formation.

4 Visual-Based Formation Control

To solve the odometry problem, the decision was to use the camera of the leader robot to improve the navigation of the robots and reduce the odometry errors. Monitoring the position of the followers while the formation is moving, it is possible to indicate to them when they have lost the position in the formation and what is their current position. The followers, in that case, must perform a modification of their trajectory in order to put themselves in the correct position in the formation again.

In order to calculate those real positions, the leader uses the targets that each follower carries and the target that the conductor also carries, as explained in [10]. In this approach, the leader must be at the back of the formation and all of the followers and the conductor being visible, and uses the camera to localize the rest of robots.

Determination of the position and orientation of any robot can be achieved by estimating its distance and relative orientation with regard to the leader robot. The position of the observed robot can be obtained from its estimated distance and the pan angle of the camera. The relative orientation of the robot can be obtained by means of the observation of the target, this calculation is explained in [10].

Because the leader is also a follower, its position must be calculated, and for this, it uses the target of the conductor. Localizing the position of the conductor it is possible to establish the real position of the leader. So, two possibilities in the calculation of the position of the followers are implemented.

In the case of the calculation of the position of the leader, it simply localizes the position of the conductor, and by means of this position, its real position can be calculated. Thus, to calculate its position, the leader needs to know three values, the position of the conductor in relation to itself, represented by the translation matrix ${}^{L}T_{C}$, the current position of the conductor in its local system, represented by the translation matrix ${}^{Oc}T_{C}$, and the position of itself in relation with the conductor in the creation of the formation, represented by the translation matrix ${}^{Oc}T_{Ol}$. Knowing that the current position of the conductor in its local system can be calculated as

$$ {}^{Oc}T_{C} = ({}^{O}T_{Oc})^{-1} \cdot {}^{O}T_{C} \tag{4} $$

the actual position of the leader is calculated as

$$^{Ol}T_L = (^{Oc}T_{Ol})^{-1} \cdot ^{Oc}T_C \cdot (^{L}T_C)^{-1} \tag{5}$$

In figure 5, this calculation can be seen graphically.

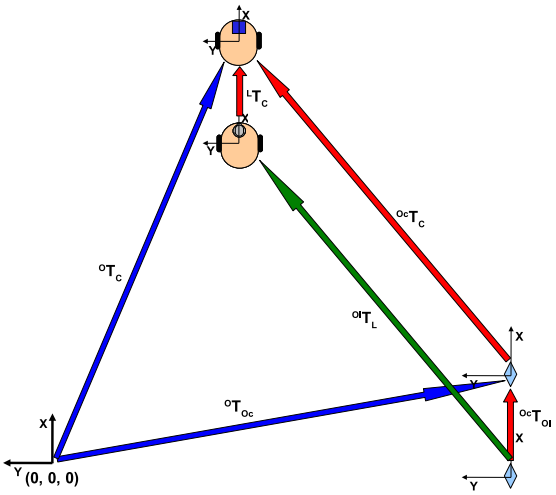


Fig. 5 Calculation of the conductor's position in relation to the followers

In order to calculate the position of any of the rest of followers, it is also necessary to know the position of this robot in relation to the leader, which is calculated by the leader using the camera. Thus, to calculate its position, the follower needs to know four values, the position of the follower in relation to the leader, represented by the translation matrix $^{L}T_F$, the position of the conductor in relation to the leader, also calculated with the cam, represented by the translation matrix $^{L}T_C$, the current position of the conductor in its local system, represented by the translation matrix $^{Oc}T_C$, and the position of the follower in relation to the conductor in the creation of the formation, represented by the translation matrix $^{Oc}T_{Of}$. Knowing that the current position of the conductor in its local system can be calculated using equation 4, the actual position of the follower is calculates as

$$^{Of}T_F = (^{Oc}T_{Of})^{-1} \cdot ^{Oc}T_C \cdot (^{L}T_C)^{-1} \cdot ^{L}T_F \tag{6}$$

In figure 6, it can be graphically seen.

Once one follower has calculated its real position, it changes the value of its actual position, that is, its belief about the position where it thinks it is in that moment. In this way, in the following step, when the follower calculates the Bezier trajectory to follow the conductor, it will use the real position in those calculations, and

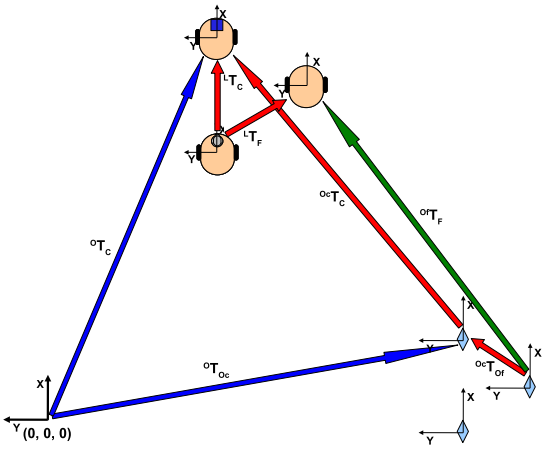


Fig. 6 Calculation of the conductor's position in relation to the followers

automatically, following the trajectory of that Bezier curve, it will go to its correct position in the formation while it moves following the conductor.

5 Testing and Results

Some experiments have been performed in order to validate the new approach. Based on these experiments, it can be concluded that the maintenance of the formation has been improved considerably, and now the robots are able to maintain their positions in the formation while following the conductor in any path.

In figures 7 and 8, two experiments using different numbers of robots are depicted. In the case of figure 7, only two robots are used in a classical leader-follower configuration, with a distance of 2 meters between the two robots. Figure 8 depicts a new experiment with four robots in a diamond formation. In this case, there's a distance of 2 meters between the two robots in the X-axis, and 2 meters between the robots in the Y-axis. In the tow cases, the velocity of the conductor is $0,1m/s$. This velocity guarantees that the leader can monitor the robots in real time. As it can be seen, the robots behave as expected, solving in this way the odometry errors seen in the case of the odometric guidance of the formation. In the case of the odometric maintenance, in the simplest case where one robot follows another one, the errors can arrive to 50 cm in the Y-axis in the middle of the trajectory. With the introduction of the visual information, these errors are drastically reduced due to the robots are continuously coming back to their correct position in the formation. The deviation in this case depends on the time that the leader localizes the follower again, but it is not more than a few centimeters (10 - 20 cm), which are automatically reduced by the follower by the new calculation of the trajectory to follow.

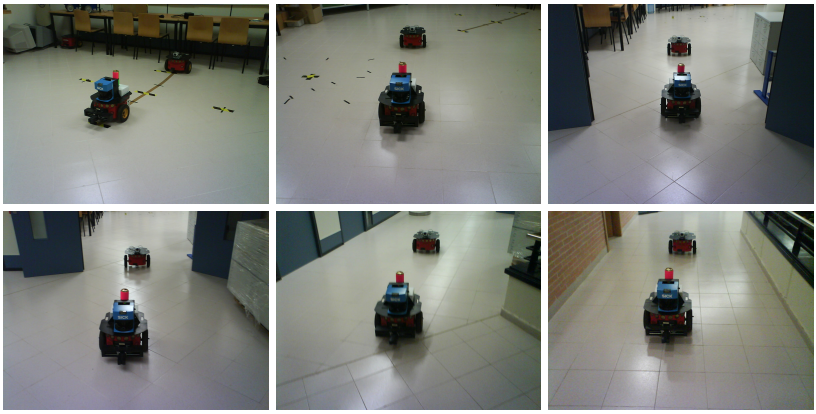


Fig. 7 Test with two-robot leader-following formation using visual repositioning

Fig. 8 Test with four-robot diamond formation using visual repositioning

6 Conclusions

The paper describes a new method for a team of heterogeneous robots to navigate maintaining any type of predefined geometric formation. In this application, the robots without sensory power get support for the navigation from the robot with navigation and localization facilities. Moreover, in the case that any robot, due to the odometry errors, lose the original path, the robot with the camera helps it to come back to the good trajectory. In this way, all the robots are able to follow a predefined path getting help from others if necessary, that is, cooperating among them.

In this paper, it is presented a new design of formation navigation based on a ”follow the conductor” approach, with a series of virtual points with a displacement of the leader for constructing the formation, and Bezier curves for the movement of each robot. Moreover, it has been implemented a cooperative method in the formation task

in order to eliminate the odometry errors of the robots and exploit the resources of each robot for the benefit of the whole team. The cooperation is carried out in such a way that the robots without sensors get help from the resources of the others to navigate and maintain the formation. For this reason, the robot with the laser sensor indicates the path to be followed and the one with the camera produces information about the errors of odometry to the robots so that they can correct these errors.

The results have demonstrated that the approach is suitable for getting the maintenance of formations, from 2 to 4 robots. It has been impossible to test with more robots, but the definition of the method can be applied to any number of robots. However, due to the restriction of the visual field of the leader, this only can be applied to a number of robots which cannot produce occlusions among them to be in the vision of the leader.

Acknowledgements. This paper describes research carried out at the Robotic Intelligence Laboratory of Universitat Jaume-I. Support is provided in part by the Generalitat Valenciana under project GV/2010/087, and by the Fundació Caixa Castelló - Bancaixa under project P1-1A2008-12.

References

1. Balch, T., Arkin, R.: Behaviour-based formation control for multiagent robot teams. IEEE Transactions on Robotics and Automation 14(6), 926–993 (1998)
2. Chiem, S., Cervera, E.: Vision-based robot formations with bezier trajectories. In: Proceedings of the 8th Conference on Intelligent Autonomous System (2004)
3. Das, K., Fierro, R., Kumar, V., Ostrowski, J.P., Spletzer, J., Taylor, C.: A vision-based formation control framework. IEEE Transactions on Robotics and Automation 18(5), 813–825 (2002)
4. Desai, J., Ostrowski, J., Kumar, V.: Modelling and control of formations of nonholonomic mobile robots. IEEE Transactions on Robotics and Automation 17(6), 905–908 (2001)
5. Dudek, G., Jenkin, M., Milios, E.: A taxonomy of multirobot systems. In: Robot Teams: From Diversity to Polymorphism, pp. 3–22. AK Peters (2002)
6. Fredslund, J., Mataric, M.: A general, local algorithm for robot formations. IEEE Transactions on Robotics and Automation (Special Issue on Advances in Multi-Robot Systems) 18(5), 837–846 (2002)
7. Li, S., Boskovic, J., Mehra, R.: Globally stable automatic formation flight control in two dimensions. In: Proceedings of the AIAA Guidance, Navigation and Control Conference (2001)
8. Liu, S., Tan, D., Liu, G.: Distributed formation control of robots with directive visual measurement. In: Proceedings of the IEEE International Conference on Mechatronics & Automation, pp. 1760–1765 (2005)
9. Nebot, P., Cervera, E.: A framework for the development of cooperative robotic applications. In: Proceedings of the 12th International Conference on Advanced Robotics, pp. 901–906 (2005)
10. Nebot, P., Cervera, E.: Self-localization of a team of mobile robots by means of common colored targets. In: Proceedings of the 6th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2009), pp. 274–279 (2009)
11. Renaud, P., Cervera, E., Martinet, P.: Towards a reliable vision-based mobile robot formation control. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3176–3181 (2004)

Reactive Coordination and Adaptive Lattice Formation in Mobile Robotic Surveillance Swarms

Robert J. Mullen, Dorothy Monekosso, Sarah Barman, and Paolo Remagnino

Abstract. We present here a set of decentralised control laws to facilitate lattice formation and reactive coordination and control of a swarm of mobile ground based robots. The control laws rely on local, indirect communication, which we implement in the form of virtual forces governed by physics based laws, computed from range and bearing measurements relative to the individual robots. Furthermore, we introduce the Virtual Robot Node (VRN) architecture to extend the capabilities of the cooperative formation control in terms of lattice cohesion and reactive dynamic abilities. The characteristics of the control laws are analysed through a number of 3D physics-based simulation experiments. We show that the basic proposed methods exhibit robustness to simulated sensor noise. We further show a number of improvements made by employing the VRN architecture, in terms of reducing errors in specified formation constraints, and additional dynamic capabilities.

1 Introduction

Swarm robotics is a relatively new area of research and development that has emerged from the swarm intelligence paradigm [4]. Much of the inspiration behind swarm intelligence based systems comes from observations of biological swarms in nature, and in particular the self-organising, emergent properties they exhibit, for example the foraging behaviours of a colony of ants [6][9].

Swarm robotics largely considers systems of multiple, relatively simple, homogeneous robots, with local and limited sensing and communication abilities, that work collectively to achieve some unified goal [7]. The use of multiple robots

Robert J. Mullen · Sarah Barman · Paolo Remagnino
Digital Image Research Centre, CISM, Kingston University, London, England
e-mail: {r.mullen,s.barman,p.remagnino}@kingston.ac.uk

Dorothy Monekosso
School of Computing and Mathematics, Ulster University, Northern Ireland
e-mail: dn.monekosso@ulster.ac.uk

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 229–242.
springerlink.com

necessitates the system to be scalable in nature, and the local and limited sensing and communication abilities provide a focus on a decentralised approach.

Research in the area of swarm robotics, and indeed multi-robot systems in general, remains very active. Many different methods are being developed to achieve autonomous multi-robot control, for a wide range of applications and robotic platforms. We present here a brief overview of a selection of related methods.

From the swarm intelligence paradigm, the concept of stigmergy [9] has been used, for example in [20][8], to achieve multi-robot coordination and control by using artificial pheromones to guide the robots. Similarly, the well established potential fields method [10] is still widely used for robot control/coordination. Both of these methods involve computing virtual force vector-fields which guide the robots movement. Another similar method uses physics based laws to compute local artificial forces [21][16] to guide the robots movements. The latter method does not require the computation of a global vector field of artificial forces, rather each robot computes the forces it experiences locally, thus offering a more distributed approach.

The well established nature inspired technique of flocking [19] has been used in many variations to achieve multi-robot coordination and control (for example [12][22][13]), and often use virtual forces to achieve the desired behaviours. One of the potential limitations of the typical flocking approach is that it required inter-robot communication, for example to share heading information [22] or velocity [13] between the robots.

A more engineering based method is reported in [2], which uses a kinematics control approach and merges a number of elementary behaviours into one final behaviour to facilitate the entrapment and escorting of a target by multiple robots.

The latter mentioned approach, along with the methods using physics based laws to compute virtual forces, tend more towards a deterministic nature, as opposed to the stochastic nature of the ant-algorithm approaches of [20][8]. The stochastic nature of many swarm intelligence methods makes them particularly good at solving unpredictable and dynamic problems (in [11] stochasticity is introduced in a Lyapunov-based flocking controller and shown to improve performance). This can also however make it difficult to fully predict the behaviour of the solution itself [23], which, in particular safety critical applications, might be an undesirable feature.

In this paper, we employ a variation of the swarm intelligence technique of stigmergy to facilitate self-organisation, but implement this in a deterministic way, to maintain a predictable system as much as possible. We develop multi-robot coordination and control laws based on physics-based virtual forces, similar to those presented in [21][16]. Other similar approaches include [18], where multi holonomic agent formation shape and orientation is studied in the setting of tensegreity structures, and [5] which presents a discrete-time optimisation framework for target tracking with multi-agent systems (which uses inter-robot communication, in contrast to our proposed approach). We introduce a new control architecture that uses Virtual Robot Nodes (VRNs) to better define and maintain specific formation constraints. We develop the control laws to facilitate formation control and maintenance and extend the capabilities to allow for dynamic formation control with

changing geometric environment constraints, similar to the work presented in [13]. We however maintain robot anonymity by designing our controllers to not require inter-robot communication, and place emphasis on making the dynamic control elements sensor driven.

2 System Overview

The proposed method uses indirect observatory communication to create a number of control laws that facilitate reactive self-organising behaviour within a swarm of multiple robots. The indirect communication is implemented as virtual forces which are calculated based on range and bearing measurements relative to the individual robot. The Lennard-Jones potential is used to govern the virtual forces, with additional constraints introduced to induce specific behavioural characteristics.

2.1 Formation Control

The basic control law yields the self-organisation of the swarm into a quasi regular spaced repeating lattice formation, based on the *Physicomimetics* framework represented in [21]. This is achieved by each individual robot measuring the range, r, and bearing, θ, to any neighbouring robots $n \in N$, within a given 'visible range', r_{vis}. The visible range is user-set, however there may also be physical constraints imposed depending on the sensor method used to determine the range to neighbouring robots. The inter-robot force, F_R, experienced due to neighbouring robot, n, is given by:

$$F_{Rn} = 4\varepsilon \left[\left(\frac{R}{r} \right)^{\sigma} - \left(\frac{R}{r} \right)^{\tau} \right], \tag{1}$$

where ε is the maximum allowed attractive force, R is the desired separation distance between neighbouring robots, and σ and τ are control parameters. We also apply an additional constraint to limit the allowed overall force, such that $F_{Rn} = \varepsilon$ if $F_{Rn} > \varepsilon$ and $F_{Rn} = -\varepsilon$ if $F_{Rn} < -\varepsilon$. The x and y components are given by: $F_{Rnx} = F_{Rn} cos(\theta_n)$ and $F_{Rny} = F_{Rn} sin(\theta_n)$. The total x and y components of the force experienced are then given by: $F_{Rtotalx} = \sum_{n=1}^{N} F_{Rnx}$ and $F_{Rtotaly} = \sum_{n=1}^{N} F_{Rny}$. We employ a discrete-time approximation and calculate the resultant x and y components of the velocity as: $v_x = F_{Rtotalx} \Delta t$ and $v_y = F_{Rtotaly} \Delta t$. From v_x and v_y we calculate the new displacement vectors as: $\Delta x = v_x \Delta t$ and $\Delta y = v_y \Delta t$. The calculated Δx and Δy values give the desired robot displacement at time t, due to the virtual forces acting upon it.

2.2 Virtual Robot Nodes

The basic control law described in Section 2.1 is limited in being only capable of creating certain lattice formations, which are based on a repeating lattice of

equilateral triangles. The desired result is however not always achieved, due to, for example the 'clustering' problem [21][17], and the formation topology can be unpredictable. This also makes formation maintenance problematic, especially when dealing with dynamic environments.

We aim to remove this limitation by introducing a novel method of adaptive formation control, which we achieve by extending the notion of virtual forces to include VRNs. VRNs can be used not only to create specific user defined formations, but can also be used to facilitate self-adapting formation control, as well as sensor directed collective movement.

Any robot can place a VRN within its neighbourhood at a given r and θ, which will then result in the same virtual force being calculated as if there had been a real robot detected at that range and bearing. VRNs are only visible to the robot that placed them; maintaining the distributed property of the system. The idea is to allow individual robots to manage their own spatial requirements in a reactive, dynamic manner.

By setting particular conditions and constraints on when and where an individual robot places a VRN it is possible to achieve a range of different formations, for example line, column and wedge. Additionally this approach can be used to induce self-organised flocking behaviours. Due to space limitations however, we concentrate here on the application of the VRN architecture to the control law detailed in Section 2.1, in order to create and maintain a double wedge formation, and alleviate some of the limitations of the original control law.

The robots neighbourhood is divided into J segments, $\Psi_j \in J$, with each segment defined by a radius of r_{seg} and two bounding bearings ψ_a and ψ_b (See Fig. 1, left). When the robot obtains the information regarding the range, r, and bearing, θ, of any neighbouring robots, the robot checks which segments contain neighbouring robots. Those segments which do not contain any neighbouring robots receive a VRN at range r_{VRN} and bearing $\theta_{VRN} = (\psi_a + \psi_b)/2$, such that:

$$VRN(j) = \begin{cases} FALSE \text{ if any } \theta_{n \in N} \in \Psi_j \\ TRUE \text{ otherwise} \end{cases} \tag{2}$$

For the double-wedge formation we have $J = 6$ segments, with $\theta_{VRN\,j} = j\pi/6$, $\psi_{a\,j} = (j\pi/3 - 4\pi/9)$ and $\psi_{b\,j} = (j\pi/3 - \pi/18)$.

Robots on the periphery of the lattice thus create a VRN boundary around the outer region of the current formation, which effects the overall shape of the formation, in this case, with 8 robots, resulting in a double wedge formation as depicted in Fig. 1, right.

2.3 Dynamic Directed Movement Behaviour

This behaviour extends the basic control law to enable sensor driven dynamic, collective movement through a given environment. To facilitate this behaviour we modify the e-puck robots in 3D simulation by providing each robot with 16 additional range finding sensors (equivalent to long-range IR, or laser), with approximate

equal spacing around the circumference of the robots outer body, to provide 360 degree sensing of a walled environment. The sensors are placed above the height of the robots to avoid the main body of neighbouring robots being detected as the environment.

Any robot that does not detect any real robots ahead of its current position (with respect to the forwards direction through the environment), considers itself a *leading* robot. This means that instead of attempting to maintain formation the robot attempts to move through the centre of the environment, guided by measurements taken from the on-board distance sensors. The robot will place a VRN with a forwards/left/right bias with respect to its current heading, to induce an attractive force in the desired direction. Any given robot will not know whether or not a neighbouring robot is a leading robot or not, maintaining the anonymous property of the system.

Any robot not leading will dynamically adjust its desired inter-robot distance R with respect to measurements taken from the on-board range finders.

In the most basic set-up, using only the formation control law given in Section 2.1, we set $\xi(t) = range_{min}(t)$, where $\xi(t)$ represents the robots perceived range to the wall at time t, and $range_{min}(t)$ is the minimum range measured from the on-board distance sensors at time t.

We then set the following conditions to adapt the robot's R value:

$$R(t) = \begin{cases} R(t-1) - 1 \text{ if } \xi(t) < (R(t-1) - (R(t-1)/\chi)) \\ R(t-1) + 1 \text{ if } \xi(t) > (R(t-1) + (R(t-1)/\chi)) \end{cases} \tag{3}$$

where χ is a sensitivity weighting parameter.

Changing the R value in this way allows the swarm to dynamically expand and contract with changing geometric environment characteristics as observed by on-board sensors. The resultant movement is however somewhat non-cohesive, with no specific formation being maintained (as shown in Section 3.2).

In order to achieve a more cohesive swarm movement, with the ability to achieve and maintain a specific formation, we implement the VRN architecture described in Section 2.2 and add additional constraints on calculating the dynamic R value. From

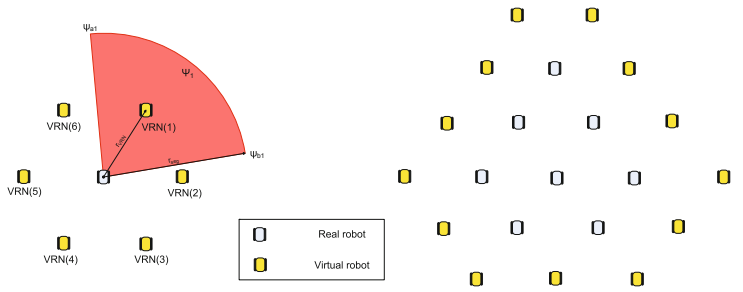


Fig. 1 Left: A schematic diagram of the VRN positions and segments; right: a schematic diagram of the real robot and VRN positions for a double wedge formation

the VRN segment scanning, each robot estimates its current position with respect to the desired double wedge formation, and adjusts ξ accordingly, in order to maintain formation within the constraints of the environment. This is implemented as follows:

$$\xi(t) = \begin{cases} range_{min}(t) & \text{if } VRN(1,2,3) = TRUE \\ range_{min}(t) & \text{if } VRN(4,5,6) = TRUE \\ range_{min}(t) - R(t) & \text{if } VRN(1,2,3,4,5,6) = FALSE \\ range_{min}(t) - (R(t)/2) & \text{otherwise} \end{cases} \quad (4)$$

where $VRN(n) = TRUE$ if a VRN is placed in segment n, and is *FALSE* otherwise.

These constraints allow for the offset of $range_{min}$, depending on where the robot is in the ideal formation.

The VRN range is then set as $r_{VRN} = R + (R - r_{min})$, where r_{min} is the minimum range to a neighbouring robot.

To further improve the swarm cohesion we take inspiration from the well established *flocking* phenomena. In recent years it has been shown [3] that when birds in the wild exhibit flocking behaviours, each bird interacts on average with a fixed number of neighbours (six of seven), rather than with all neighbours within a fixed radius. We implement a similar behavioural characteristic by imposing each robot to choose the nearest 6 robots and/or VRNs to compute as neighbouring robots for inclusion in the control law.

The resultant behaviour allows the swarm lattice to self-organise into and maintain the double wedge formation and dynamically expand and contract with the changing geometric characteristics of the environment, while collectively moving along the passageway. The movement(s) of the leading robot(s) cause(s) a chain reaction through the rest of the swarm as neighbouring robots are attracted to those moving away, along the passageway, with the VRNs maintaining the desired double wedge formation.

2.4 Experimental Set-Up

Experiments are carried out in a 3D physics-based simulation environment, using the Webots professional mobile robot simulation package [14]. We use the e-puck [15] as our robotic platform, which is a small-scale differential drive laboratory robot approximately 7cm in diameter.

Each time-step in the experiment is equivalent to approximately $0.1s$ real-time. During each time-step, each robot calculates the range and bearing to all neighbouring robots. Each robot then executes one of the control laws, which will in turn output a displacement vector based on the virtual forces acting upon the robot. The displacement vector then needs to be converted into left and right wheel velocities in order to drive the e-puck robot towards the desired location. We have developed a motor control law derived from the one reported in [1] for this purpose, which provides smooth closed-loop steering for a differential-drive robot towards the desired location.

For all experiments reported in this paper, unless stated otherwise, the control parameters for the basic control law are set with: $\sigma = 0.1$ and $\tau = 0.05$.

3 Results

We present a number of experiments designed to show the effectiveness of the proposed control laws for a number of multi-robot cooperative coordination and control scenarios. In particular we aim to show a number of advancements to the existing method of utilising artificial forces to govern robot formation control.

3.1 VRN Formation Control

To demonstrate this functionality we consider a limitation of the basic control law described in Section 2.1. Given 7 robots the resultant formation would be a well formed hexagon with a robot at the centre, maintaining the repeating lattice of equilateral triangles. If however we had 8 robots, the resultant formation would be less cohesive, with one robot 'trapped,' resulting in a less uniform lattice, not resembling a repeating lattice of equilateral triangles. This clustering phenomena has been previously reported (in for example [21][17]), and is known to be caused by local minima of the inter-robot forces.

We carry out an experiment in 3D simulation with 8 robots starting in a pseudo-random cluster (Fig. 2, bottom-left), running the basic formation control law of Section 2.1, with $R = 50cm$, $\varepsilon = 20.0$ and $r_{vis} = 3R/2$. The resultant lattice formation can be seen in Fig. 2, bottom-centre. Although the robots have manoeuvred from their initial positions into a formation, there is one robot 'trapped' in the centre of the formation, preventing the swarm from achieving the desired repeating equilateral triangle lattice.

By using the VRN architecture we enable each of the robots on the periphery of the lattice to behave as if there were another layer of robots beyond them, causing the formation to expand into the desired repeating equilateral triangle lattice, making a double wedge formation. Repeating the same experiment with the VRN enabled control laws with $r_{VRN} = 60cm$, we observe a final lattice formation with greater geometric cohesion, showing near uniform inter-robot separation (Fig. 2 bottom-right). The VRN architecture is shown here to overcome the clustering problem in a single merged behaviour.

In Fig. 2, top, we show a plot of the error in average minimum inter-robot distance (calculated against the desired inter-robot distance of 50cm) for both the formation experiments. As expected, we indeed see the VRN control law yielding a smaller error as the formation converges, in comparison to the basic control law.

The smaller, positive error observed when the swarm has converged using the VRN control law can be attributed to the constant attractive force from the VRN robots on the periphery of the formation. To reduce this error further still, we might

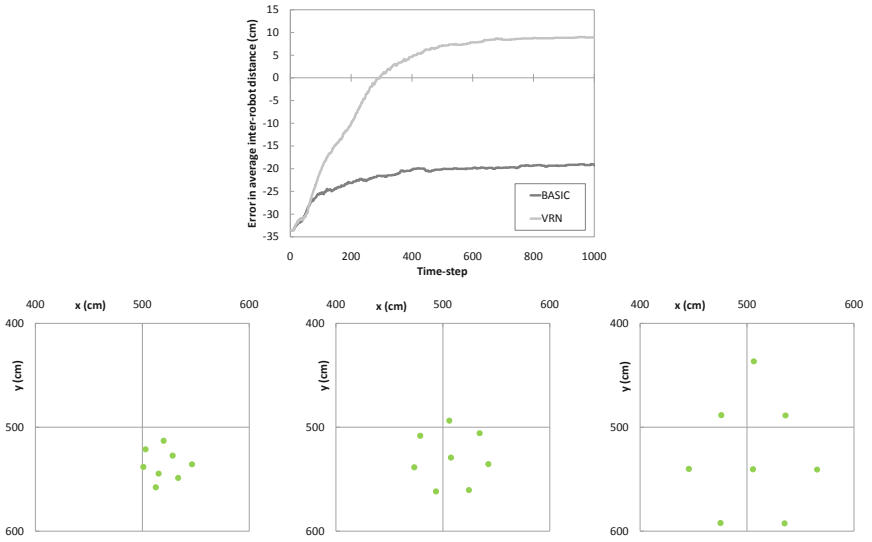


Fig. 2 Results for the VRN formation comparison experiment. Top: A typical plot of the error of the average minimum inter-robot distance versus time-steps, for both the basic and VRN enabled control laws in 3D simulation. The error is calculated against the desired inter-robot distance of 50cm. Bottom-left: A Plot showing the positions (green circles) of the robot starting positions in 3D simulation; bottom-centre: the final positions from running the basic formation control law; bottom-right: the final positions from running the basic formation control law with the addition of the VRN architecture.

introduce a dynamic r_{VRN} that changes according to the robots perceived formation accuracy. This could reduce the force experienced due to the VRNs in a dynamic fashion, as the formation becomes complete. This is however beyond the scope of this paper and will feature in future work.

Fig. 3 gives a comparison between the VRN and basic control laws for formation control with varying numbers of robots, showing the average Standard Deviation (SD) of minimum inter-robot distance (left) and the average error in minimum inter-robot separation distance (right). The experiments are run with $R = 50cm$, for $N = 4, 8$ and 16 robots. For each method and for each set number of robots, we run the experiments 10 times, changing the pseudo-random starting positions each time.

For each N value we see that the VRN method has yielded lower errors in comparison to the basic method. Again the errors present in the basic method are mainly due to the 'clustering' problem, as mentioned previously, with the VRN method overcoming this problem. Fig. 3 also shows the positive error present with the VRN method to be less than the error due to the clustering problem, over the range of N.

We also note that the difference in error for increasing N with the VRN method is relatively small, showing a level of scalability for small groups of robots (future work will investigate scalability for larger group sizes of $N = 1000+$). The basic

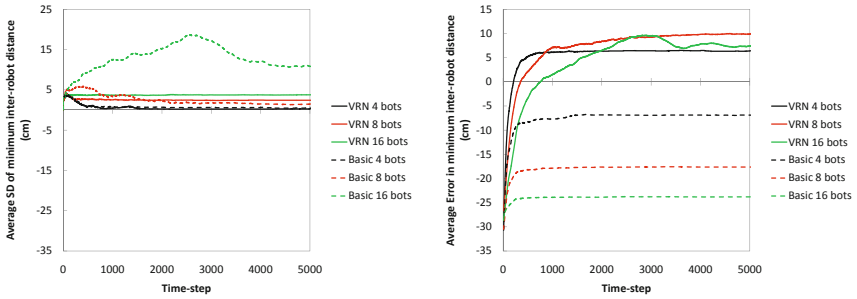


Fig. 3 A comparison between the VRN and basic control laws for formation control with varying numbers of robots. Right; showing the average error in minimum inter-robot separation distance, and left; showing the S.D. of inter-robot separation distance.

method however shows a significant increase in error with increasing N (approximately $+10.7cm$ between $N = 4$ and $N = 8$ and approximately $+6.1cm$ between $N = 8$ and $N = 16$).

The SD for the VRN method shows a small (and relatively uniform) increase with increasing N. For the basic method there is minor increase in SD from $N = 4$ to $N = 8$ and then a significant increase in SD (to a gap of approximately $9.3cm$ at $t = 5000$) from $N = 8$ to $N = 16$.

The distributed nature of our proposed system, in particular the non-use of inter-robot communication, means that each individual robot is reliant entirely on its own sensors to gather the required information to execute the desired behaviours. It is therefore important that we assess the robustness of the proposed system to the effects of sensor noise.

We carry out another Webots simulation-based formation experiment, with $N = 8$ robots, $R = 30$ and $r_{virtual} = 40$. We carry out the experiment for increasing levels of simulated sensor noise effecting the robots' θ and r measurements, for $0\%, 1\%, 10\%, 30\%$ and 50% levels of noise with respect to the measured value. For each level of noise we repeat the experiment 10 times and average the results.

Fig. 4, left, shows plots of the average SD of minimum inter-robot distance, and right, the average error in minimum inter-robot separation distance. For noise levels upto 10% we observe the levels of cohesion to remain relatively unchanged (characterised by a low SD). For noise levels upto 30% the overall formation remains connected despite the observed increase in SD of inter-robot separation distance. This increase is due to the robots 'vibrating' about thier equilibrium positions. For the average error (Fig. 4, right) we observe an increases in error of approximately $0.8cm$, $5.8cm$ and $8.5cm$, for 1%, 10% and 30% noise levels respectively.

Given that in these experiments the levels of cohesion remain relatively unchanged for levels of sensor noise upto 10%, and the overall formation management remains stable with levels of noise upto 30%, this provides example evidence of robustness of the system with respect to coping with noisy sensor data. Preliminary work carried out on real e-puck robots (results not included here due to space

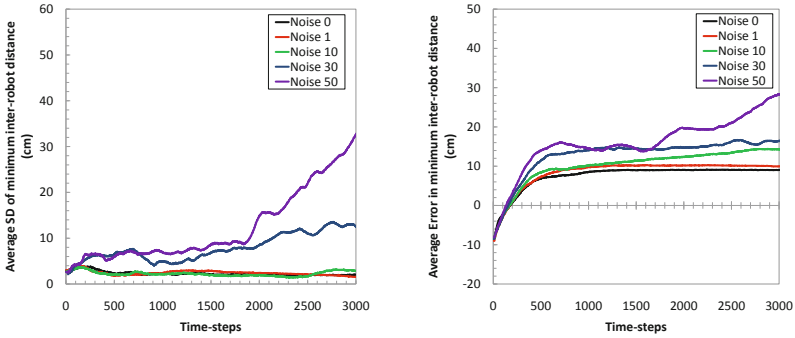


Fig. 4 Plots showing the effects of sensor noise on the VRN-based formation management. Right; showing the average error in minimum inter-robot separation distance, and left; showing the S.D. of inter-robot separation distance, for increasing levels of noise.

limitations) have shown very similar trends when compared to identical experiments carried out in 'ideal' simulation settings, again providing evidence of robustness to noisy sensor data, as well as robustness to actuator noise such as wheel-slip.

3.2 Dynamic Directed Movement

To test the dynamic directed movement extension to the proposed control law, along with the VRN architecture, we consider an experiment where the challenge is for the robot swarm to collectively move though a walled passageway, distributing dynamically with regards to the changing width of the passageway (with a coverage based sweep search scenario in mind), while maintaining maximum cohesion within the formation. The 3D environment in this experiment consists of a narrow walled passageway which opens into a wider passageway. The robots start in a pseudo-random cluster in the narrow section, and move into the wider section. The relevant parameters for this experiment are set with $\varepsilon = 20.0$, $r_{vis} = 3R/2$ and $\chi = 3$.

Fig. 5 shows example results from a typical experiment run for both the basic control law and VRN enabled dynamic directed movement algorithm. In Fig. 5, bottom-left, we see the robot positions and trajectories throughout the experiment with just the basic control law. We see that the robot swarm does effectively adapt to the increasing width of the environment, however the swarm does not appear to maintain any specific formation throughout. Fig. 5, bottom-right, shows results of the same experiment with the VRN enabled control law. Again we see the swarm effectively adapting to the changing width of the environment, and we also see in this case that the swarm maintains the desired double wedge formation after forming this lattice from the initial pseudo-random starting positions and adapting to the increase in environment width.

Fig. 5, top left, shows a plot of the SD of the minimum inter-robot separation distance measured by each robot, versus time-steps, comparing the basic and VRN

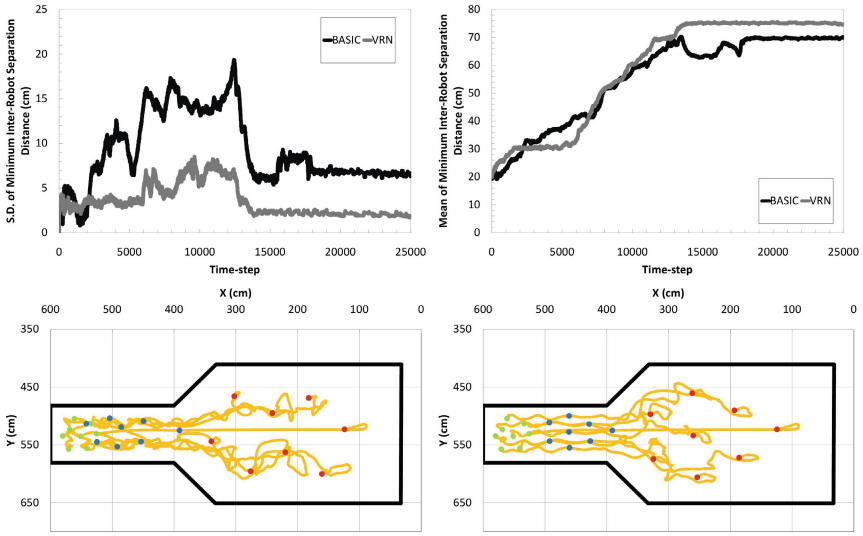


Fig. 5 Results of the dynamic directed movement experiment. Top left: A plot of the standard deviation of the minimum inter-robot separation distance versus time-steps, comparing the basic and VRN-enables control laws. Top-right: A plot of the mean of the minimum inter-robot separation distance versus time-steps, comparing the basic and VRN-enabled control laws. Bottom-left: A plot of robot positions (coloured circles) and past trajectories (orange lines) at different stages in the experiment using the basic control law. Green circles show the starting positions, blue circles show intermediate positions and red circles show the final positions. Solid black lines denote the environment walls. Bottom-right: the same using the VRN-enabled control law.

enabled control laws. We would expect the SD to be small when the robots are in a cohesive formation, with similar inter-robot separation distances (with an ideal formation having equal inter-robot distances and thus a SD of zero).

Firstly we observe that the VRN experiment exhibits a lower SD for the majority of the experiment. The area of relatively large peaks seen for both curves between approximately 6000 and 13000 time-steps is the region where the swarm is adapting to the increase in environment width. The experiment shows the basic control law to yield a larger increase in SD during this self-adapting process, suggesting a less cohesive collective behaviour.

Fig. 5, top right, shows a plot of the mean of the minimum inter-robot separation distance measured by each robot, versus time-steps, comparing the basic and VRN control laws. For the VRN curve, the plateau between approximately 1500 and 6000 time-steps corresponds to where the swarm has self-organised into the double wedge formation (as can be seen in Fig. 5, bottom-right, blue circles), and is moving along the narrow section of the environment before the wider section. This correlates to the maintained low SD throughout this section as seen in Fig. 5, top left.

For the basic control law, the mean of the minimum inter-robot separation distance continues to increase from 0 to approximately 13000 times-steps, during

which period the SD is seen the fluctuate to comparatively high values. This can be attributed to the observation that for the basic control law, the swarm does not achieve a cohesive formation from the initial starting positions. As can be seen in Fig. 5, bottom-left, the blue circles show a non-cohesive formation expanding lengthways along the narrow section of the environment.

As the curves reach a plateau at 14000 and 18000 time-steps for the VRN and basic control laws, respectively, this corresponds to the final formations reached as the swarms have expanded into the larger section of the environment (Fig. 5, bottom row, red circles). We observe that the VRN control law has maintained the double wedge formation after the adaptive process, and achieved a final SD of approximately $0.69cm$. The swarm under the basic control law successfully manages to expand with the changing environment, but the final formation appears less cohesive from visual inspection, with a final SD of approximately $7.1cm$.

4 Discussion and Conclusion

We have presented a swarm robotics system that utilises relative range/bearing sensing capabilities to facilitate indirect communication, using artificial force laws to induce self-organising behaviour to guide the robots collectively into lattice formations.

Furthermore, we have introduced an architecture called VRNs, which can be used to overcome the 'clustering' problem inherent in the artificial force laws method, and moreover, can be used to create specifically structured formations without requiring explicit communication.

We have also extended the proposed control laws to achieve adaptive formation control for reactive collective movement through simple changing environments, while maintaining a fully distributed approach. These reactive behaviours, while not requiring inter-robot communication, do rely heavily on sensor readings. We have shown the proposed method to be relatively robust to simulated sensor noise. Further work will involve extensive testing on real robots to assess the effects of real hardware sensor and actuator noise.

Additional ongoing and future work includes further assessment of the use of VRNs to create different types of formations with varying numbers of robots and with varying geometric environment constraints, as well as carrying out a quantitative comparisons to related methods. We are also further investigating the use of VRNs for emergent flocking behaviours and will compare this approach to other flocking methods in future work. Preliminary work suggests our proposed system scales well with increasing numbers of robots. We are also addressing the robustness of the proposed system to robot failure.

Ultimately we are interested in applying a learning strategy to allow the robots to learn when and where best to position VRNs given specific sensory input about the local environment, in order to evolve a number of cooperative movement strategies for behaviours such as searching, obstacle avoidance and environment/structure

inspection. Although the presented framework is not specific to any given application, as the title of the paper suggests, we are interested in surveillance scenarios. In a wider context, we aim to develop the proposed VRN architecture to control teams of mobile robots equipped with video cameras, and to optimise cooperative searching formations and movements in order to maximise visual information retrieval of a given environment in minimum time.

References

1. Aicardi, M., Casalino, G., Bicchi, A., Balestrino, A.: Closed loop steering of unicycle-like vehicles via Lyapunov techniques. IEEE Robotics and Automation Magazine 2(1), 27–35 (1995)
2. Antonelli, G., Arrichiello, F., Chiaverini, S.: The entrapment/escorting mission - an experimental study using a multirobot system. IEEE Robotics and Automation Magazine, 22–29 (2008)
3. Ballerini, M., Cabibbo, N., Candelier, R., Cavagna, A., Cisbani, E., Giardina, I., Lecomte, V., Orlandi, A., Parisi, G., Procaccini, A., Viale, M., Zdravkovic, V.: Interaction ruling animal collective behaviour depends on topological rather than metric distance: Evidence from a field study. Proc. Natl. Acad. Sci. USA (PNAS) 105(4), 1232–1237 (2008)
4. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, New York (1999)
5. Derenick, J., Spletzer, J., Hsieh, A.: An optimal approach to collaborative target tracking with performance guarantees. J. Intell. Robot. Syst. 56, 47–67 (2009)
6. Dorigo, M., Gambardella, L.M.: Ant Colony System: A cooperating learning approach to the travelling salesman problem. IEEE Transactions on Evolutionary Computation 1(1), 53–66 (1997)
7. Dorigo, M., Sahin, E.: Swarm robotics - special edition editorial. Autonomous Robots 17(2-3), 111–113 (2004)
8. Garnier, S., Tache, F., Combe, M., Grimal, A., Theraulaz, G.: Alice in pheromone land: An experimental setup for the study of ant-like robots. In: Proceedings of the 2007 IEEE Swarm Intelligence Symposium (SIS 2007), pp. 37–44 (2007)
9. Grasse, P.P.: La reconstruction du nid et les coordinations interindividuelles chez bellicositermes natalensis et cubitermes sp. la theorie de la stigmergie: Essai dinterpretation du comportement des termites constructeurs. Insectes Sociaux 6, 41–81 (1959)
10. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. The International Journal of Robotics Research 5(1), 90–98 (1986)
11. Kumar, M., Milutinovic, D., Garg, D.P.: Role of stochasticity in self-organisation of robotic swarms. In: Proceedings of the 2008 American Control Conference (2008)
12. Labonte, G.: Canadian Arctic Sovereignty: Local Intervention by Flocking UAVs. In: Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defence Applications (CISDA 2009) (2009)
13. Manh, H., Sheng, W.: Adaptive flocking control for dynamic target tracking in mobile sensor networks. In: Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2009), pp. 4843–4848 (2009)
14. Michel, O.: Webots: Professional mobile robot simulation. Journal of Advanced Robotics Systems (2004)
15. Mondada, F., Bobani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.-C.: The e-puck, a robot designed for education in engineering. In: Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions, pp. 59–65 (2009)

16. Morgan, D.S., Schwartz, I.B.: Dynamic coordinated control laws in multiple agent models. Physics Letters A 340, 121–131 (2005)
17. Navarro, I., Pugh, J., Martinoli, A., Matia, F.: A distributed scalable approach to formation control in multi-robot systems. In: Proceedings of the International Symposium on Distributed Autonomous Robotic Systems, DARS 2008 (2008)
18. Pais, D., Cao, M., Leonard, N.E.: Formation shape and orientation control using projected collinear tensegrity structures. In: Proceedings of the 2009 American Control Conference (2009)
19. Reynolds, C.: Flocks, herds and schools: A distributed behavioral model. In: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 1987), pp. 25–34 (1987)
20. Sauter, J.A., Matthews, R., Parunak, H., Brueckner, S.A.: Performance of digital pheromones for swarming vehicle control. In: Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 903–910 (2005)
21. Spears, W.M., Spears, D.F., Hamann, J.C., Heil, R.: Distributed, physics-based control of swarms of vehicles. Autonomous Robots 17, 137–162 (2004)
22. Turgut, A.E., Celikkanat, H., Gokce, F., Sahin, E.: Self-organized flocking in mobile robot swarms. Swarm Intelligence 2, 97–120 (2008)
23. Winfield, A.F.T., Liu, W., Nembrini, J., Mertinoli, A.: Modelling a wireless connected swarm of mobile robots. Swarm Intelligence 2, 241–266 (2008)

Probabilistic Communication Based Potential Force for Robot Formations: A Practical Approach

Simon Bjerg Mikkelsen, René Jespersen, and Trung Dung Ngo

Abstract. We introduce a new method of artificial potential forces based on probabilistic communication, called *'Probabilistic Communication based Potential Forces'*- PCPF. The potential forces provides a locally distributed control for a formation of a large volume of self-regulated mobile robots. While models of sensing and communication so fare mostly have been with simple assumptions that are far away from the physical properties of sensors and communication mechanisms, the method here is realistic because both attractive and repulsive forces are only based on probability of communication which are empirically measured and approximately estimated between robots. The method is demonstrated through non-trivial examples of robot formation and formation transformation. Analysis is provided to facilitate understanding of the elements of the probabilistic method.

1 Introduction

Robot formations has recently been investigated as an idea for deploying multi-robot systems in dynamical environments for various applications, e.g., exploration, victim searching, environmental patrol and monitoring, and surveillance and reconnaissance. The advantage of using robot formations is that the system is robust as they can assist each other through inter-sensing and inter-communication, flexible as they can adapt to changes of the environment by transforming their formation shapes, and scalable as their formation is not depending on the number of participants.

In a distributed scheme, to maintain the position in the formation, a robot needs to estimate the relative distance and orientation to the neighbors using its sensor and/or communication mechanism. While simulating robot formations, it mostly works

Simon Bjerg Mikkelsen · Rene Jespersen · Trung Dung Ngo
Aalborg University, Aalborg, Denmark
e-mail: simbjemik@gmail.com, rene84@control.aau.dk, dungnt@es.aau.dk

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 243–253.
springerlink.com

very well because assumptions of highly accurate sensing and communication are used to generate their relative distance and bearing. However, when transitioning the controller from simulation to the physical robots, the difference between the simulated and the real-world sensing-models, lead to the simulated controller not performing as good. At least not without further tuning.

This research focuses on a new method for developing potential forces that is very realistic to implement in real robot systems. The method is based on the probability of communication between robots for both attractive and repulsive forces.

In the perspective of real-world robot formation applications, we propose a general method, including obstacle avoidance and goal targeting. We can separately develop a potential function for obstacle avoidance based on capability of obstacle detection, and an external potential function as global force that may be applied to guide the robots to target a goal. However, this paper restricts in the method of Probabilistic Communication based Potential Forces (PCPF) to the distributed controllers of robot formations without considering obstacle avoidance or goal targeting.

The remainder of this paper is organized as follows: First, we catalogize and charatertize the artificial potential field of robot formation into centralized and distributed schemes, based on previous work. Then, we briefly present our general principle for a probabilistic potential force. A case study of robot formations using our custom-made robots show how the probabilistic method has been generated and applied to make realistic controllers. A sequence of examples shows how the robots can self-organize and self-regulate their formation. Theoretical analysis is carried out to facilitate deeper understanding of the method. We conclude with summation of contribution and future work.

2 Related Work

The Artificial Potential Field [1] was originally found for single robot navigation. The method represents interactions of the robot with environmental obstacles as a vector field. The vector field is made up by attractive and repulsive forces. The attractive forces may include constraint forces that create geometric shapes of robot formations or attaches the robots to the target location, while the repulsive forces may embrace environmental force to pull the robot away from obstacles. The sum of attractive and repulsive vector fields is used to decide the robot behavior.

The Artificial Potential Field has been extended in to various dimensions in multi-robot systems. However, it can be sorted out in two catalogues; centralized potential field and distributed potential field.

In the centralized paradigm the whole environment is globally assigned potential forces where obstacles are given repulsive forces, and the goal is attributed an attractive force. The robot is lead to follow the path resulting as the minimum of subtraction of attractive and repulsive forces. Examples of centralized potential fields

can be found in [1, 2, 5, 4, 3]. The drawback of the centralized method is that it is not robust and flexible because the path is not changeable after being initialized by the global force field of the environment, therefore it is hard to apply for a multi-robot system where dynamics must be considered in real-time.

In the distributed paradigm the potential force is locally represented in the robot's vicinity, usually based on the robots' perception of their surroundings. The advantage of this method is that since the potential force is calculated individually for each agent it can be updated in real-time. Because of this, the distributed potential field is popularly used to generate controllers for robot formation which requires high robustness, flexibility and scalability. For examples, the *social potential fields* [7] creates force-laws to allow for different forces between agents. Similarly, different social potential forces inspired by molecular form crystals is used to enable formations of scalable multi-robot systems [6]. The *light weight methodology* [8] used the springs law to create the potential forces with constraints of virtual leader to maintain the robot formation. All were under the assumption of perfect perception among the robots and the results were only demonstrated in simulation. On the other hand, the *heading alignment and proximity control* [11] and the *artificial physics* [9] are a few examples demonstrating usability of potential forces for formation of the physical robots. The former combines heading alignment which is based on digital compass and wireless communication and IR proximity measurements, to generate virtual forces for robot controllers. The later is inspired by natural law of gravitational forces in generating the potential forces to each robots. The method is analyzed deeply in terms of robustness, flexibily and scalability, and demonstrated in simulation and with 7 real robots as well.

In general, the above mentioned methods calculate potential forces based on sensing of relative positioning between robots (the wireless communication used in [11] is apart from the estimation of relative positioning). Also, it is assumed that robots have perfect sensing systems being able to estimate accurate distance to all neighbors.

3 General Principle of Probabilistic Potential Force

In this section, we describe our general method of probabilistic potential forces that we propose for building robot controllers in a swarm. This is illustrated in the Equation (1) which is based on each robots inter-communication with neighbors, observation of environmental obstacles, and goal targeting. Because the controller is based on the probability of communication and perception of individual robots, it allows designing a fully distributed control for swarm formations. A robot r calculates its distributed potential force as.

$$\vec{F}_r = \sum_{i=1}^{n} \vec{F}_{att}\,(i) + \sum_{i=1}^{n} \vec{F}_{rep}\,(i) + \sum_{o=1}^{m} \vec{F}_{obst}\,(o) + \sum_{g \in G} \vec{F}_{swarm}\,(g) \qquad (1)$$

where n is the number of communication channels on the robot, m is number of obstacles appearing in the robot's perceptual vicinity and g is sub goal of G the swarm goal.

The two first items, $\vec{F}_{att}$ and $\vec{F}_{rep}$, are a robot's probabilistic potential forces based on directional communication with its neighbors in the local vicinity. The combination of attractive force and repulsive forces induces coherence in the swarm in accordance to relative distance and orientation.

The third item, $\vec{F}_{obst}$, is a probabilistic repulsive force between the robot and environmental obstacles based on perceptual capability of the robots. This repulsion ensures all robots to avoid obstacles when swarming.

The fourth item ,$\vec{F}_{swarm}$, is a potential force to control all robots of the swarm towards the desired goal.

4 The Approach to Robot Formation

This study is based on the physical design of our custom-made robots illustrated in Figure 1.

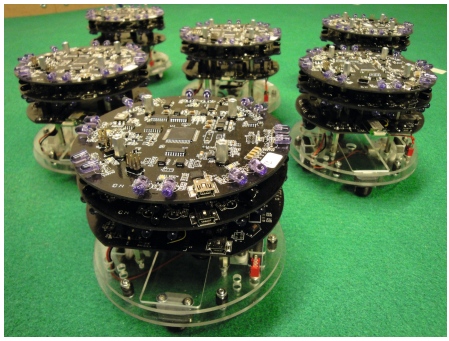


Fig. 1 Developed robots to test the theory in practice

Equipped with 8 pairs of infrared emitting and receiving diodes located symmetrically at 45 °, the robots can communicate and sense around at 8 different directions. Infrared sensing and communication is not reliable and far from free of bit errors. Furthermore, there is no sensing and communication within the blind spots of two nearby infrared beams as illustrated in Figure 2.

4.1 Communication Model

To achieve a realistic model of the communication between two robots a set of experiments has been carried out. The bit error rate depends on the strength of the

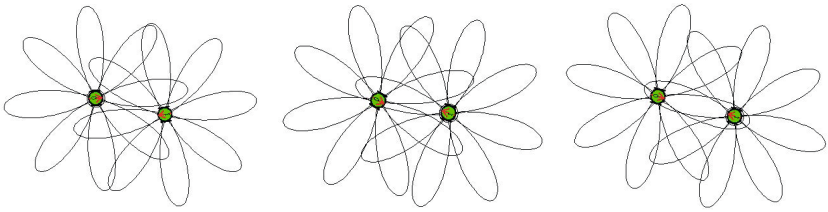


Fig. 2 Interrelation of sensor beams and communication possibilities between two robots. In this figure, each robot has 8 "flower petals" representing the angular view of both infrared transmitters and receivers. The two robots can establish line of sight communication based on their relative position: (a) no robot can communicate each other, (b) only left robot receives the signal, (c) they can mutually communicate.

signal received, which depends on relative angle and distance between the robots. Based on the experiment data a function was approximated to model the voltage in the receiver as a function of distance and angle. This model is shown in Equation 2 to 5

$$U_r(a,d) = \begin{cases} \text{U(a,d) if } U(a,d) < 4.7 \\ 4.7 \quad \text{if } U(a,d) \geq 4.7 \end{cases} \tag{2}$$

$$U(a,d) = M_r \frac{K(a)}{d^{n(a)}} \tag{3}$$

$$K(a) = \frac{c_K}{\sqrt{2\pi} \cdot \sigma_K} e^{\frac{-a^2}{2 \cdot \sigma_K^2}} \tag{4}$$

$$n(a) = \frac{c_N}{\sqrt{2\pi} \cdot \sigma_N} e^{\frac{-a^2}{2 \cdot \sigma_N^2}} \tag{5}$$

Where the following values where determined. $M_r = 0.5271$, $c_K = 518.71 \cdot 10^3$, $\sigma_K = 6.336$, $c_N = 163.106$ and $\sigma_N = 25.768$.

U is the voltage level on the receiving diode as a function of angle a and distance d. U_r is that same voltage with maximum limit of 4.7 Volt.

Some simplifications were made to the model. We assume the petal of transmitter and receiver are identically shaped. This allows the simplification of the model to only take, the sum of the absolute values of the robots' angles to the direct line between them into account. Hence U_r is a function of only two parameters. All influences from the physical surroundings of the robots are ignored. The tolerance of the components is also an unknown factor, and the model was therefore based on a set of measurements from all 8 different infrared transmitters on the robots.

The voltage is used as an input to determine the probability of a bit error. $P0$ in equation (6) is the probability of a bit error when sending a "0", $P1$ in in equation (7) the probability of bit error when sending a "1". The bit error is changed into a byte error using 50% ones and 50% zeros in byte. The model only provides a value

for the probability and does not distinguish between the robot being either far away or the relative angle very large.

$$P0(U_r) = \begin{cases} 0 & \text{if } U_r < 1.3640 \\ \alpha_0 \cdot U_r + \beta_0 & \text{if } U_r \geq 1.3640 \end{cases} \tag{6}$$

$$P1(U_r) = \begin{cases} 1 & \text{if } U_r < 0.0549 \\ \alpha_1 \cdot U_r + \beta_1 & \text{if } 0.0549 \leq U_r < 0.2976 \\ 0 & \text{if } U_r \geq 0.2976 \end{cases} \tag{7}$$

Where:
$\alpha_0 = 6.9 \cdot 10^{-3}$, $\beta_0 = -9.4 \cdot 10^{-3}$, $\alpha_1 = -4.1202$, $\beta_1 = 1.226$,

Figure 3 is an illustration of the signal spectrum relative to the probabilistic characteristic of the communication model, showing the error rate when sending a random bit.

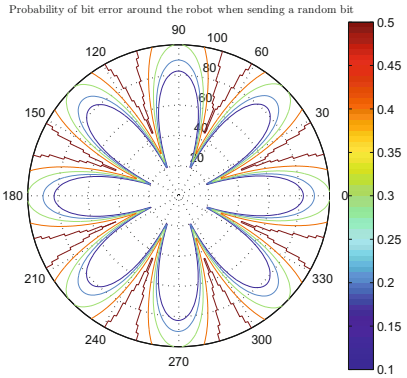


Fig. 3 Bit error probability of random bits [10]

4.2 Potential Force

To limit this study, $\vec{F}_{obst}$ is not implemented meaning no obstacles is introduced in the simulation and the common goal $\vec{F}_{swarm}$ of the swarm is simply set to zero. Because we have 8 directional communication channels on each robots, the sum of potential forces established on 8 communication channels of a robot is stated in Equation (8).

$$\vec{F}_r = \sum_{i=1}^{8} \vec{F}_{att}\,(i) + \sum_{i=1}^{8} \vec{F}_{rep}\,(i) \tag{8}$$

Taking an example of gathered empirical data, the sum in Equation (8) is depicted in Figure 4.

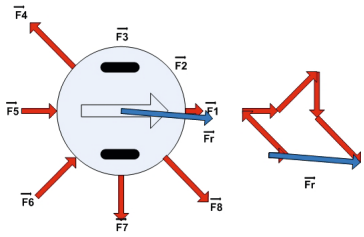


Fig. 4 Summation of directional forces on a robot

Attractive Force

Whenever a robot receives a transmission from another robot, a virtual force of constant magnitude *atf* is set in the direction the receiving sensor points.

The attractive force on each communication channel can be stated as Equation (9).

$$\vec{F}_{atf} = \begin{cases} \text{atf if byte received} \\ 0 \quad \text{if byte not received} \end{cases} \tag{9}$$

where the attractive force *atf* is set to 3.

Repulsive Force

To maintain a constant distance between robots a repulsive force is needed. The wanted distance between robots will be maintained when the repulsive force and the attractive force completely subtract each other. Robots will be repelled by measuring the strength of infrared light received on each sensor and calculating a repulsive force as linear proportional to this using Equation 10.

$$\vec{F}_{rep} = U_r \cdot ref + L_0 \tag{10}$$

where U_r is calculated using Equation 2, *rep* is negative in the order of -40 and L_0 is 50.

4.3 Simulator

The simulator is constructed in Matlab Simulink(TM) using a kinematic model of the robots and the probabilistic potential field. The probabilistic communication model developed in Section 4.1 is used to determine when a transmission is succesfull by comparing the probability to a random generator. Since Simulink is a graphically based each robot is implemented as a separate block. As the robot block contains the kinematic model and probabilistic controllers, inputs to the robot blocks

are sensor values and output are the robot acting coordinator (x, y, θ). The kinematic model of differentially driven robots used in the simulation is described as follows

$$\dot{X}_i = \hat{v}_{i_x} \equiv v_i \cdot cos(\theta_i) \tag{11}$$

$$\dot{Y}_i = \hat{v}_{i_y} \equiv v_i \cdot sin(\theta_i) \tag{12}$$

where $\hat{v}_{i_y}$ and $\hat{v}_{i_x}$ are the velocity of v_i along the X and Y axises. The angle θ is the derivative of the angular velocity of the robot ω_i: $\dot{\theta} = \omega_i$

The linear velocity v can be calculated as the mean of the forces applied on the wheels. The environmental influence is implicitly modeled in a separate block on each robot. This block takes all other robots output and compare them to make sure robots will not transmit through each other.

5 Experiments and Results

Two different initial scenarios have been tested using potential function control to determine if the PCPF can obtain and maintain a hexagonal lattice structure. Scenario 1 where the robots initially are very close together, and scenario 2 with larger distances. 5 simulations was run for each scenario with different starting positions and each simulation was run for 20 minutes. In Figure 5 there is one example plot from each scenario. Each plot is of one simulation and illustrate positions over time where the + marks the initial position, the line marks the path traveled and the robot marks the final position after 20 minutes.

Figure 6 shows a box plot of inter robot distances for all five simulations in scenario 1. One plot for every 30th second. The robots very quickly move away from each other and after just one minute the upper and lower quartile are stable at approximately 75 cm and 40 cm, with a mean of approximately 65. The mean does

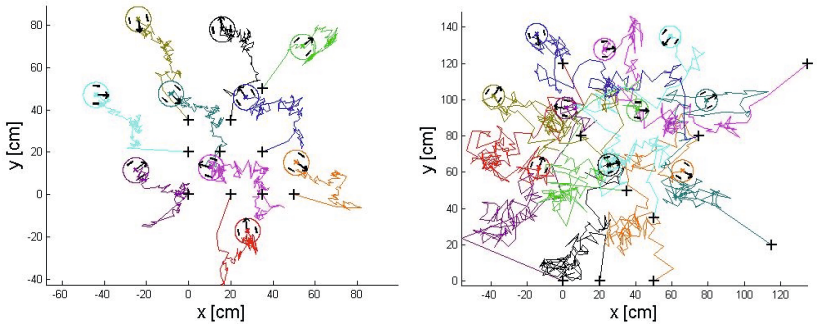


Fig. 5 PCPF run for 20 minutes example positions plotted with path starting in the +'s and ending in robots:(a) Scenario 1, Robots initially close (b) Scenario 2, Robots initially spaced further apart

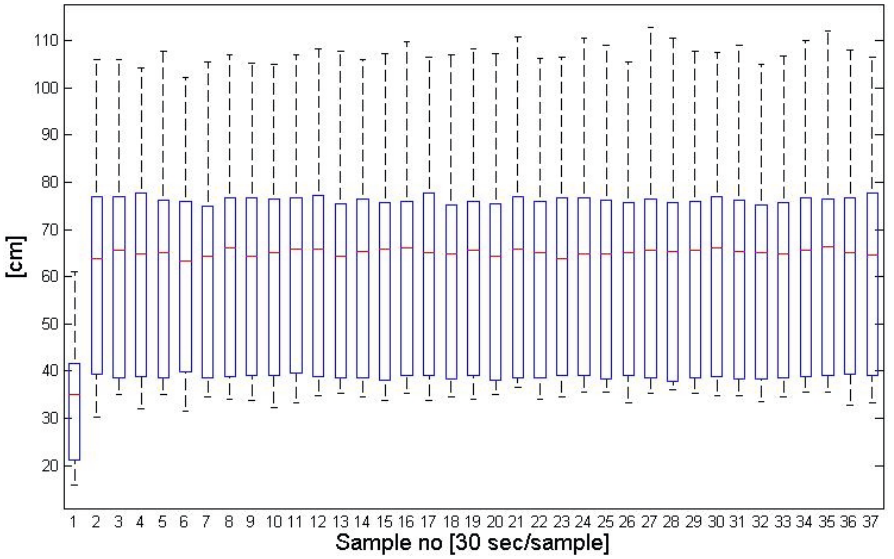


Fig. 6 All inter robot distances (in cm) from all 5 simulations of scenario 1 are plotted in a box plot once every 30th sec.

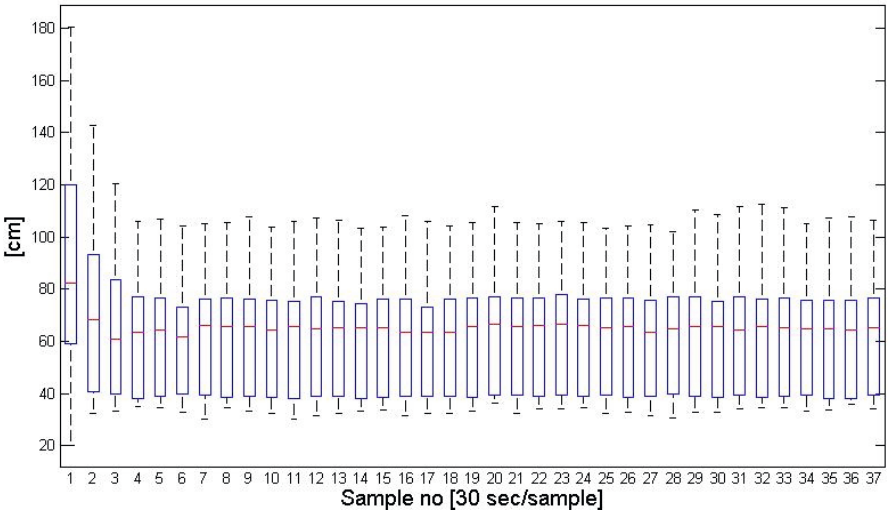


Fig. 7 All inter robot distances (in cm) from all 5 simulations of scenario 2 are plotted in a box plot once every 30th sec.

vary over time as would be expected given the randomness introduced but not much. The equivalent plot for scenario 2 is depicted in Figure 7. The settling takes longer, but the quartiles and mean settle on approximately the same values as in scenario 1 supporting the claim that it is stable.

6 Analysis

The study of this paper, that lies beyond the current state of the art is about a realistic potential field for robot formation. Because the potential field is a sensor-based method [1], the sensor model is therefore extremely important in representing the world in the robot mind. However, most of nowadays sensors do not provide absolutely realistic measurement, especially in a dynamically changing environment. Hence, a probabilistic sensor model that is realistic and adaptable to the empirical world is needed to overcome many other presumed models. In our approach, the sensing and communication models are absolutely based on empirical experiments of our sensor board without any presumption. The robot controller is generated based on those probabilistic models.

It is clear from the results of the simulations that the probabilistic potential function based controller is capable of achieving hexagonal lattice formations within a couple of hundreds seconds. Both scenarios shows hexagonal lattice formations. In scenario 1 the robots starts close together and spreads out, and in scenario 2 the robots are placed further apart and move closer to each other to obtain the formation. It supports the statement that, it does not matter whether the formation needs to extract or expand.

For videos go to the webpage.[1]

7 Conclusion

The paper presents a new approach to robot formation based on Probabilistic Communication based Potential Forces. A number of novelties presented in this paper are: 1) The potential force is based on the communication model and signal intensity only, instead of a sensor-based model as tradition; 2) The probabilistic model is based on the approximation of the empirically data from experiments on the physical robots. And these have only *flower* shaped communication capabilities, as illustrated in Figure 3. Hence 360°communication or perception is not available as many other assume; 3) The robot controller is naturally synthesized based on the probability of those communication without any modification.

The results demonstrated that the approach is sufficient to form and maintain the robot formation. We examined that phase synchronization might not be needed to maintain the formation as the PCPF can do it very well. We still believe it is necessary if we want to reduce the randomness when controlling the movement of a whole swarm. A compatible integration between the PCPF and phase synchronization needs to be further investigated. We do not, nor do we intend to prove that assuming perfect communication is wrong in all cases. We merely suggest that to bridge the gap, that often occurs between simulation and real world, a simulator with more nuance is a very good idea. To back this up, in the near future, the good simulation results the algorithm is going to be implemented and verified on the

[1] http://vnbotics.blogspot.com/2010/07/probabilistic-potential-field-for-robot.html

physical robots. On the other hand, even though robot model and communication models are quite accurate with respect to the physical robot platform, the simulator can then be made more accurate by tuning the models further. Obstacles should then be introduced into the simulation to achieve a proper model of the environment. Ultimately, comparison between simulation and real experiments should be carried out to support the final statement of the novel approach.

References

1. Khatib, O.: Real time obstacile avoidance for manipulators and mobile robots. International Journal of Robotics Research 5(1), 90–98 (1986)
2. Scheider, F.E., Wildermuth, D., Wolf, H.L.: Motion coordination in formations of mutliple mobile robots using a potential field appraoch. Distributed Autonomous Systems 4, 305–314 (2000)
3. Horward, A., Mataric, M., Sukatme, G.: Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In: Sixth International Symposium on Distributed Autonomous Robotics Systems, Fukuoka, Japan, pp. 229–208. ACM (2002)
4. Kraus, S., Shehery, O., Yadgar, O.: Emergent cooperative goal-satisfaction in large scale automated-agent systems. Artificial Intelligence 110, 1–15 (1999)
5. Carlson, B., Gupta, V., Hogg, T.: Controlling agents in smart matter with global constraints. In: AAAI 1997 Workshop on Constraints and Agents (1997)
6. Balch, T., Hybinette, M.: Behaviour based coordination of large-scale robot formation. In: Proceedings of the Fourth Intenational Conference on Multiagent Systems, Boston, MA, USA, pp. 363–364 (2000)
7. Reif, J., Wang, H.: Social potential fields: A distributed behavioral control for autonomous robots. Robotics and Autonomous Systems 27(3) (1999)
8. Elkaim, G., Siegel, M.: A Lightweight Control Methodology for Formation Control of Vehicle Swarms. In: IFAC (2005)
9. Spears, W., Spears, D., Hamann, J., Heil, R.: Distributed, Physics-Based Control of Swarms of Vehicles (2002)
10. Lyhne, C., Sørensen, E., Jespersen, L., Krogen, M.: Network protocol stack for robots in a scalable swarm. In: AAU SEMCON 2009, Aalborg, Denmark (2009)
11. Turgut, A., Celikkanat, H., Gökce, F., Sahin, E.: Self-Organized Flocking with a Mobile Robot Swarm. In: AAMAS (2008)

Coordinating a Group of Autonomous Robotic Floats in Shallow Seas

Eemeli Aro, Zhongliang Hu, Mika Vainio, and Aarne Halme

Abstract. Shallow seas are extremely difficult environments for autonomous underwater profiling floats. These robots possess no thrusters and only one actuator for their buoyancy control, and are thus entirely dependent on sea currents for lateral motion. As a further restriction, underwater acoustic communication is very limited. Taking into account these challenges, a novel co-operative underwater multi-robot system has been designed and implemented for use in shallow waters. A coordination strategy and a localization method have been developed and tested using a detailed simulation of the Baltic Sea. These methods allow the system to operate safely and to map underwater currents and other environmental variables with relatively high accuracy.

1 Introduction

A profiling float is a freely drifting oceanographic measurement platform with buoyancy control [6]. In slightly different terms, it is an autonomous robot moving in a three dimensional fluid with only one actuator that lets it control its depth. Most floats are designed for and deployed in deep water, providing near-real-time information from depths of 1000–2000 m [1]. The environmental variability of the deep oceans is such that the depths and distances at which these floats operate allows for sufficient precision to be achieved by single floats that surface every ten days. On the other hand, operation closer to shore and in shallow seas is significantly more difficult for profiling floats: the spatial scale of environmental features is reduced from hundreds or thousands of meters to tens or hundreds of meters, and changes in the environment occur on the order of hours instead of days.

Eemeli Aro · Zhongliang Hu · Mika Vainio · Aarne Halme
Finnish Centre of Excellence in Generic Intelligent Machines Research,
Aalto University, Finland
e-mail: firstname.lastname@tkk.fi

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 255–266.
springerlink.com

The Autonomous Underwater Multi-probe System for Coastal Area/Shallow Water Monitoring (SWARM) was an EU-funded (FP5, 2003–2005) project aiming to design, implement and test a multi-robot system that could measure local and transient biological and physical variability in the Baltic Sea and similar areas, at the scale relevant for single events [20]. The system consists of multiple homogeneous, robust and easy to use co-operative intelligent profiling floats (See Fig. 1). These floats and their operating environment are described in more detail in [3].

The SWARM floats communicate with a control station via Iridium satellite communication and use inter-robot acoustic ranging and communication for localization and data exchange while underwater. In addition to measuring the standard variables (conductivity, temperature, and pressure; commonly abbreviated as CTD), the system can track and observe sea currents due to deformations of the group as a whole. While on the surface, each float may track its position using GPS, but underwater its only direct position information comes from its pressure sensor, from which its depth may be determined.

This paper first gives an overview of the environment and the hardware, and develops from these the necessary parameters of a co-operative system of underwater floats. The specific implementation chosen in this project is explained, including the developed localization method. These methods are then tested using a detailed simulation of the Gulf of Finland.

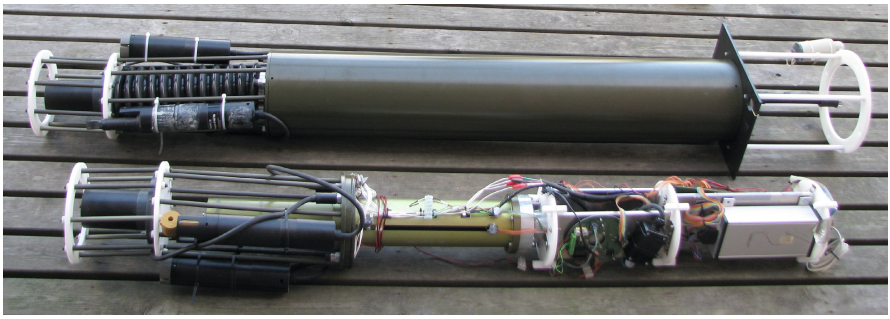


Fig. 1 Two third generation SWARM floats. The foremost unit shows the inside structure including diving engine, computer board, acoustic and Iridium modems but is lacking the battery pack. The unit behind has the outer shell and Iridium antenna in place. Weighing less than 40 kg the units are easy to transport and deploy. In water they float in a vertical orientation.

2 The Vertical Movement of Floats

A float moves by changing its own volume, and thereby changing its density. This allows it to move vertically in the water, as the water density increases with depth. This indirect form of position control is used due to its high energy efficiency, a requirement for a system that needs to function autonomously underwater for extended periods.

The relationship between depth and density is rather complex, depending also on water temperature and salinity [8]. Given that the water density varies rather little (less than 5 kg/m^3 from the surface to the bottom of the Gulf of Finland at 100 m), even small changes in temperature or salinity may result in a significant change in the depth corresponding to a set density. Due to these environmental factors, the SWARM floats use a mechanical piston for very accurate float volume control. Additionally, an algorithm [3] has been developed that allows the floats to consistently reach most depths with a precision of ± 1 m with a probability of over 98% that after the initial piston movement, at most one adjustment will be required to reach the goal depth. As the algorithm has not been tuned for speed but energy efficiency, moving for example from the surface to a depth of 50 m may take 20 minutes or longer, if multiple piston adjustments are required. Vertical movement is limited by the sea bottom, which the floats detect and avoid using an echosounder.

3 A Group of Floats

Given the task of gathering data from a 3D environment that changes with time, and the limitation of the available sensors only having local reach, a multi-platform approach is essential: A conventional float profiling operation can only result in measurements from a single water column, as well as being unable to accurately estimate sub-surface sea currents. In order for such a system to work, the units need to form a cohesive whole that is able to communicate between units, as well as determine where they are, both in absolute terms as well as in relation to other units. Further, the particular environment in which the system needs to function imposes limitations that strongly shape the system structure.

One method for the estimation of sea currents using the measured flock distortion of a descending group of vertically spaced floats was presented in [13], but that approach is limited to mapping a single profile of current estimates. A similar approach is taken in [7], where floats dive from the surface to a set depth and rise while broadcasting signals that are received by surrounding floats; the received signals are post-processed to estimate the diving floats' path while underwater. The approach used in this paper, on the other hand, aims to map sea currents across a significant volume of water rather than just at one location. Other distributed anchor-free underwater methods for the localization of a group of floats have been developed [5], but these do not take into account the unpredictable changes in environmental conditions experienced during an extended mission, as well as resulting in inadequate localization accuracy, with average error 40% or more of the signal range [9].

Alternatively, specialized instruments may be used to directly measure the velocity difference between the float and the water as in [17], but the precision of such measurements is not sufficient for the relatively slow currents of the Baltic Sea. The localization of underwater floats is also possible using fixed buoys that transmit regular acoustic pulses, but this requires external, anchored hardware that is more difficult to deploy than a group of freely drifting floats.

3.1 The Requirement for Cyclical Operations

Practical underwater communication for distances greater than tens or hundreds of meters is only possible using acoustic communication, which is dependent on the speed of sound. The speed of sound is not uniform in water, influencing the path of acoustical signals. In some seas, such as the Baltic and the Mediterranean [19], the sound speed profile may be such that a depth exists near the surface at which acoustic signals will reflect internally due to a minimum in the speed of sound. In the Baltic, this channel is present mostly during the summer, and varies in depth from 20 m to 80 m [10]. Using this sound channel and the available hardware, floats may theoretically achieve communications and ranging over distances of up to 10 km. Communication outside the sound channel is unlikely to succeed beyond hundreds of meters.

Due to communication requiring a float's presence at a specific depth, communication needs to be scheduled. As noise and other factors make communication uncertain and low-bandwidth, continuous operation of such a system naturally leads to having a communication cycle with a preset period. More specifically, the range of current velocities, the maximum range of acoustic communication, and the vertical speed of a float combine to limit the length of a communication cycle to a few hours at most. Such a time frame limits a float to only one or a few actions during a cycle.

The operational cycle (120 min) of each float will therefore consist of a brief period (15 min) of communication and planning at the sound channel depth, followed by a longer period of performing a set of actions: moving to a set depth, waiting there, and possibly communicating with a base station while on the surface, as well as getting a GPS position fix (See Fig. 3.1). The actions of a float are for the most part dependent on the actions of other floats and its relation to them, meaning that changes to a set plan should be considered exceptional. This also allows for the floats to estimate each other's actions in a reliable manner.

3.2 Group Cohesion

Operating a group of floats as a swarm requires balancing two competing forces: the floats need to stay together in order to communicate and by ranging, localize themselves, but the floats also need to stay separated from each other to cover the greatest volume of water. As a float can't move laterally by its own power, it needs to map and make use of the sea currents—which map is in fact an integral part of the system's scientific output.

In general, the variability and the magnitude of sea currents are at their greatest near the surface. In the Baltic Sea, the magnitude of the surface currents may easily be many times greater than those just a few tens of meters below. Therefore, in order to measure these deeper currents with sufficient accuracy the floats will need to minimize the time spent on the surface. This leads to requiring the maintenance of a baseline current estimate at a deeper depth against which other depths' currents

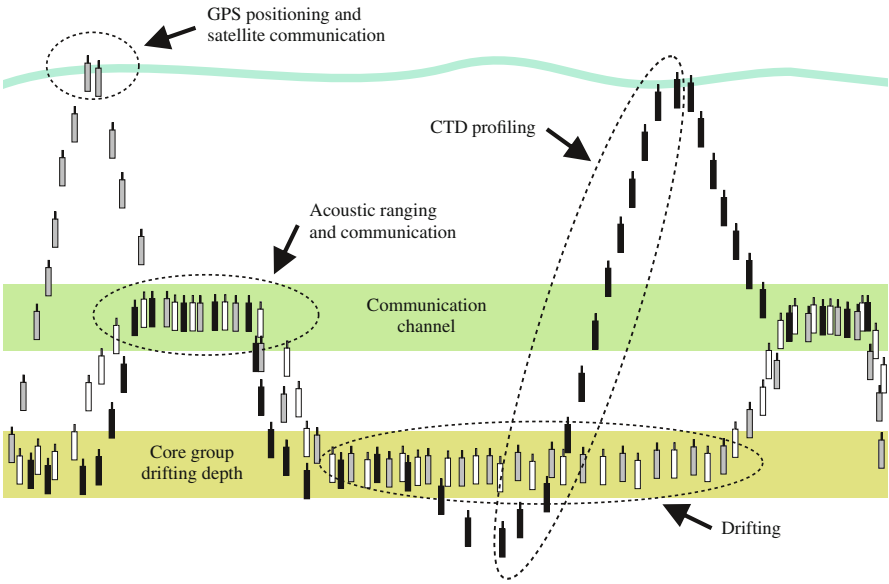


Fig. 2 Illustrative diagram of the actions of three floats during an operational cycle

may be compared using acoustic ranging information. The absolute current at that depth will need to be measured by occasional GPS fixes at the surface.

The simplest way to both maintain a current estimate at a depth and to keep the swarm together would be to keep all of the floats at the same depth, but this is obviously unsatisfactory for mapping the whole of the water column. Instead, some sufficient number of floats will need to form a *core group* that maintains the baseline for estimates of both current velocities and float positions; the rest of the floats will need to act as *scouts*, mapping the environment at different depths. The depth of the baseline will need to be adjusted during the mission, as the current velocity at that depth won't necessarily match the requirements of the mission as a whole, e.g. mapping a specific volume of water. The methods and algorithms for coordinating the swarm as a whole are beyond the scope of this paper, but the relative and absolute localization of the floats will be considered.

3.3 Float Cycle Implementation

SWARM floats operate with a cycle of set length that always starts and ends with all the floats at the sound channel depth (determined from measurements [3]). At the beginning of each cycle, each float has a communication slot during which it first attempts to establish a range estimate by sending a "ping" message to which other floats will automatically respond. After this ranging, the float may broadcast data.

The only localization information that is sent includes the rangings to other units and any recent GPS position information.

In total, acoustic ranging and communications will take at most ten minutes, the exact length depending on the number of floats and the length of each communication slot. At its end, each float will have the most up-to-date information, and may decide what actions to take until the end of the cycle. For now, these actions are limited to diving to and drifting at a specified depth, optionally followed by a profiling ascent from the bottom of the sea to the surface at the end of the cycle. While at the surface, the float will communicate with the controller via satellite after establishing a new GPS position fix. At the end of a cycle, the float will return to the sound channel depth for the next communication slot.

The normal operation of a float may of course be interrupted by exceptional circumstances, such as problems with satellite communication, avoidance of the sea bottom, detection of entrapment to a fishing net, beaching on shallow island waters, being picked up by somebody, etc.

4 Float Localization

The localization of a group of floats, as described above, can be split into three parts. The *core group* here refers to the floats that have each performed the same actions during the previous cycle, and therefore their relative motion is likely to be small.

1. Determine the relative positions of the core group
2. Localize other floats with respect to the core group
3. Estimate the absolute position, orientation, and velocity of the swarm

The approach presented here to solving these related problems makes use of two particular techniques: mass–spring optimization for self–localization by the core group of floats and multilateration for localization with respect to the core group.

As a part of the localization process, estimates of sea current velocities at the various depths visited by the swarm may be generated and maintained. For an alternative approach combining all aspects of localization and mapping, a method based on an extended Kalman filter has also been implemented [12].

4.1 Relative Self-localization

The problem of determining the relative positions of a group of underwater floats based on incomplete measurements of inter-float distances is remarkably similar to the problem of self-localization in ad-hoc wireless sensor networks, with the relaxation that each unit is occasionally able to get an absolute GPS position fix. The approach chosen here is based on mass–spring optimization (MSO)—a type of mesh relaxation—that has previously been used in such networks [16], along with a number of other approaches (e.g. [4], [14], and [15]). The primary reason for choosing MSO is its support for using previous position information in building a new

estimate. MSO has previously been used for robot localization in [11], where laser range-finder scans of reflective beacons from multiple poses were combined to a single map. The implementation presented here differs from previous work as the localization is repeated each cycle, as well as integrating information with different modalities: relative distances from acoustic ranging and GPS position data.

As all floats need to be at the same depth for communication and ranging, the localization is done in two dimensions rather than three. Additionally, the orientation of the floats may be ignored due to their rotational symmetry around the vertical axis.

It should be noted that as MSO is a type of gradient descent method, it may reach a local minimum rather than a global minimum, i.e. the localization may produce an incorrect topology. To decrease this possibility, the quality of the initial estimate needs to be improved. In practice, MSO will first be applied only to the core group of floats that have followed the same set of actions during the previous cycle; the relative positions of these floats are unlikely to vary significantly from the previous estimate.

Determining the relative positions of the core group is possible when ranging information between the floats is available, i.e. just after the communication period at the beginning of the float cycle. Each distance measurement presents a constraint to the possible positions of the floats, all of which need to be satisfied by the localization method. In the context of MSO, each such measurement is seen as a spring, with energy inversely proportional to how well the constraint is satisfied; if a constraint is satisfied, its energy reaches zero.

The specific implementation of MSO used by the floats starts with an initial estimate of the relative positions of other floats in the core group, centered on the float itself. This initial estimate is based on previous estimates of the floats' positions, at start from GPS data and later from the previous cycle's estimates. Next, the estimate is iteratively improved to minimize the differences between the estimated and measured distances between floats.

To update the estimate of float positions, we iterate over all distance measurements $d_{i,j}$ between pairs of floats i and j. First, we take $\mathbf{s}_{i,j}$ as the estimated separation vector of the floats with (x,y) positions $\mathbf{pos}_i$ and $\mathbf{pos}_j$,

$$\mathbf{s}_{i,j} = \mathbf{pos}_j - \mathbf{pos}_i \ . \tag{1}$$

Then, using a spring constant of 1, the total virtual force $\mathbf{f}_i$ acting on each float may be defined as the sum of the forces $\mathbf{f}_{i,j}$ (in the direction of $\mathbf{s}_{i,j}$) between that float and its neighbours due to the differences between the measured and estimated distances between them:

$$\mathbf{f}_i = \sum_j \mathbf{f}_{i,j} = \sum_j \frac{\mathbf{s}_{i,j}}{|\mathbf{s}_{i,j}|} \left(|\mathbf{s}_{i,j}| - d_{i,j}\right) \ . \tag{2}$$

Following [16], the estimated position of each float is then updated by

$$\mathbf{pos}_i \mapsto \mathbf{pos}_i + \frac{\mathbf{f}_i}{2n_i} \tag{3}$$

where n_i is the number of distance measurements between float i to its neighbours.

The iteration is ended when the global energy of the springs, i.e. the sum of squared errors

$$E = \sum_{i,j} |\mathbf{f}_{i,j}|^2 = \sum_{i,j} (|\mathbf{s}_{i,j}| - d_{i,j})^2 \ , \tag{4}$$

has reached a plateau and ceases to improve.

4.2 From Relative to Absolute Coordinates

Once the relative positions of the core group of floats have been estimated, other floats may be localized with respect to the core using multilateration [2]. This includes scouts—floats that have drifted at other depths than the core group—as well as floats from the core group that have visited the surface. Once these initial position estimates have been included, a second MSO round is performed, incorporating all of the floats.

As each float's local map of the float positions is centred on itself, the relationship between the relative and absolute positions may be expressed via an estimate of the absolute position of the float itself, together with a rotation for the swarm as a whole. The float's absolute position may be corrected from a single GPS fix (either by the float itself, or by another float for which the relative position is known).

The rotation of the swarm is corrected when recent GPS positions of at least two floats are available. This is done by determining the difference between the estimated and measured bearings of each pair of floats for which GPS positions are known, and correcting the swarm rotation factor by the average of these angles.

5 The Sea Simulator

The algorithms and procedures described above have been developed and verified with a simulation of the Gulf of Finland, using a simulator platform developed for this project. The simulator is a platform for testing and developing autonomous floats. It uses a server-client architecture, with each float provided with the same interfaces as the actual float has to its sensors and actuators—including realistic instrument errors. After a float is initialized, its movement in the water is controlled by the simulator according to a model taking into account its buoyancy and drag forces, as well as the three-dimensional current flow vectors. Horizontal and vertical positions and velocities are not discretized. The movement of the float's piston is modelled by limiting the rate of change of the float's volume.

The simulator is based on data from the Finnish Marine Research Institute's BalEco ecosystem model [18], which has a variable depth resolution ranging from 3 meters near the surface to 30 meters at a depth of 150 meters. Horizontal resolution for sea currents, salinity and temperature is roughly 11 km$\times$11 km, and temporal resolution is 6 hours. The sea depth data has a resolution of about 1.8 km$\times$1.8 km. The simulator uses a linear interpolation for continuous values.

The simulator is in no way limited to modelling the Baltic Sea, as long as the requisite environmental data is available. The development of this novel simulator (see Fig. 3) was necessary as no other platform was found with the required capabilities of realistically modelled fluid dynamics and support for simultaneous modelling of multiple floats (tens or even hundreds at a time) as well as hardware-in-the-loop testing. The simulator and its experimental verification with tests at sea will be presented in a later paper.

Acoustic communication is modelled in the simulator by using separate models for the maximum travel distance of a message depending on whether both the sender and recipient are in the sound channel. In the sound channel, the probability of message transmission reaches zero at 7 km, while outside the channel the maximum range is 500 m. The message travel time has an added error of up to 5%. These values are slightly pessimistic approximations, based on at-sea tests done with the acoustic modem used in the SWARM floats.

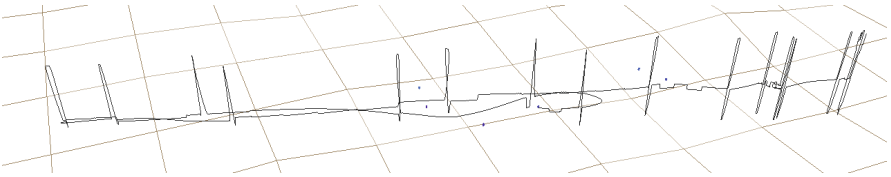


Fig. 3 3D view of the track of a simulated float over seven days, with vertical depth exaggerated by a factor of 20. Dots indicate the positions of other floats. Each surfacing is separated by twelve hours. The shape of the sea bottom is shown in the background, with a horizontal resolution of 1.8 km. The total distance traveled by the float during the time shown is roughly 20 km.

6 Test Setup

The operational cycle and the presented localization method have been tested using the sea simulator described above and a data set from August 2008. At that time of year, the Baltic sound channel is present in the Gulf of Finland at a depth of 55 m to 65 m, which precludes testing in the immediate vicinity of the coast.

For this test, the operational cycle of each float has been set to a length of two hours. At the beginning of the cycle, each float dives to the sound channel for communication and ranging, followed by a dive up to a waiting depth of 10 m. Every six cycles, near the end of the cycle, each float will descend until it detects the sea bottom using its echosounder and then rise to the surface. The surfacings are staggered by assigning each float a variable length for its first dive, with equal distribution across the group from one to six cycles. While at the surface, each float will try to get a GPS fix and communicate that via satellite to the simulated human operator. This setup allows the operator to get a confirmation of each float's continuing operation twice a day as well as an update on the group as a whole every two hours.

The floats are started from a random position near 22.35°E 59.35°N, chosen due to the sea bottom depth in that vicinity keeping at about 100 m; deep enough to have little influence on the sound channel. The starting time of each simulation run is randomly chosen from the available month of data.

Five different scenarios were tested, with variable numbers of floats and variable starting separations: groups of 6, 12, and 18 floats were each set to start within 3 km of each other, and groups of 12 floats were additionally started within 300 m and 1 km of each other. The number of floats surfacing during each cycle is always one sixth of the group size. Roughly 90 week-long deployments of each scenario were simulated, with the simulation runs taking in total about 5 days to complete using a single desktop computer.

7 Results and Discussion

During a mission of one week, each float will drift a distance of 12–25 km, almost always changing its drifting direction by 180° at least once. The average distance between floats will only increase by a factor of 1.5–2.5, as each float will experience roughly the same set of currents. One float's movement during such a deployment is shown in Fig. 3.

Overall, the localization filter is able to consistently localize each float within a few hundred meters of its actual position while drifting 1–2 kilometers underwater from its previous GPS position fix. The median position error ranges from 70 m to 200 m, depending on the number of floats and the range of deployment. The relative positions of other floats are accurate most of the time, as indicated by the low errors in the bearing and range estimates of other floats. These and other error figures are shown in Table 1.

The increased accuracy in a setup using twelve floats instead of six floats is due to the occasional availability of two simultaneous GPS measurements, which allows for the rotation of the local frame of reference to be corrected. This may be seen in the radical decrease of the bearing error (see Table 1) when moving from six to twelve floats.

The decrease in accuracy when the group size is increased to 18 was unexpected, and appears to be due to the increased complexity of the network of floats. With more floats, there are more opportunities for the imperfect distance estimates to cause an error in the estimated topology of the floats. This may also be seen in the increase of the bearing error as the separation between the units decreases: shorter distances between floats allow for more changes in the topology, resulting in more mistakes.

The accuracy of each float's self-localization may also be gauged by comparing the position estimate error to the distance traveled from the latest GPS position fix. Effectively, we may compare the results of the MSO localization to the assumption of always being exactly where the last GPS fix was made. Depending on the magnitude and variability of the currents, this may in fact be a reasonable estimate to make. For estimates made when the float has traveled at least 500 m from the

Table 1 MSO localization errors

	Group size	6	12	12	12	18
Separation at start (km)		3.0	0.3	1.0	3.0	3.0
	Sample size	45 344	85 727	80 406	79 585	135 900
Position error (m) [a]	mean	478.8	119.9	141.5	216.2	410.6
	median	222.2	66.7	69.8	107.3	136.1
	P_{90}	1184.0	255.4	256.4	342.4	1032.2
Bearing error (rad) [b]	mean	0.287	0.363	0.214	0.115	0.315
	median	0.179	0.087	0.045	0.032	0.067
	P_{90}	0.637	1.177	0.674	0.193	1.068
Range error (m) [c]	mean	131.0	16.7	33.3	108.6	123.4
	median	98.3	14.3	29.7	94.8	101.8
	P_{90}	204.0	27.4	50.0	160.6	191.8

P_{90} indicates the 90th percentile.

[a] Distance from each float's estimate of its position to its actual position

[b] Average absolute error in the estimated bearings from each float to all other floats

[c] Average absolute error in the estimated ranges from each float to all other floats

previous GPS position (true for roughly half of the samples), the position estimate is closer to the float's true position in 80–98% of the cases[1].

8 Conclusions and Future Work

The operational environment of SWARM floats is challenging, especially when combined with the requirements of an extended mission. In particular, the limited energy available to each float forces the operational cycle to such a length that the water currents and other environmental variables may change radically between measurements. Despite this, float localization may be achieved at a sufficiently high accuracy for the applications in question.

In order to develop and test the operation of the swarm as a whole, a simulator platform and autonomous float software have been developed. These tools will be used for further research, including future work on even more accurate localization methods. The accuracy of these tools will also need to be tested, with deployments of the actual hardware at sea.

Acknowledgements. This research has been supported by the Academy of Finland Centre of Excellence in Generic Intelligent Machine Research (GIM). We would also like to thank the SWARM consortium, Janne Paanajärvi at GIM, and Tapani Stipa at the Finnish Meteorological Institute.

[1] 6 floats, 3 km: 80.4%; 12 floats, 0.3 km: 98.2%; 12 floats, 1 km: 96.7%; 12 floats, 3 km: 95.1%; 18 floats, 3 km: 87.0%.

References

1. Argo Steering Team, On the design and Implementation of Argo - an initial plan for the global array of profiling floats. International CLIVAR Project Office Report, vol. 21 (1998)
2. Arnold, J., Bean, N., Kraetzl, M., Roughan, M.: Node localisation in wireless ad hoc networks. In: Proc. ICON 2007, pp. 1–6 (2007), doi:10.1109/ICON.2007.4444052
3. Aro, E., Vainio, M., Hu, Z., Halme, A.: Diving in density: Controlling the depth of a profiling float in coastal waters. In: Proc. IASTED-CA 2010 (2010)
4. Biswas, P., Ye, Y.: Semidefinite programming for ad hoc wireless sensor network localization. In: Proc. IPSN 2004, pp. 46–54. ACM (2004), doi:10.1145/984622.984630
5. Chandrasekhar, V., Seah, W.K., Choo, Y.S., Ee, H.V.: Localization in underwater sensor networks: survey and challenges. In: Proc. WUWNet 2006, pp. 33–40. ACM (2006), doi:10.1145/1161039.1161047
6. Davis, R.E., Sherman, J.T., Dufour, J.: Profiling alaces and other advances in autonomous subsurface floats. Journal of Atmospheric and Oceanic Technology 18(6), 982–993 (2001)
7. Erol, M., Vieira, L.F.M., Gerla, M.: Localization with dive'n'rise (dnr) beacons for underwater acoustic sensor networks. In: Proc. WuWNet 2007, pp. 97–100. ACM (2007), doi:10.1145/1287812.1287833
8. Fofonoff, N.P., Millard, R.C.: Algorithms for computation of fundamental properties of seawater. UNESCO Technical papers in marine science, vol. 44 (1983)
9. Garcia, J.: Positioning of sensors in underwater acoustic networks. In: Proc. OCEANS 2005, vol. 3, pp. 2088–2092 (2005), doi:10.1109/OCEANS.2005.1640068
10. Hela, I.: The sound channel of the baltic sea. Geophysica 5(4), 153 (1958)
11. Howard, A., Mataric, M., Sukhatme, G.: Relaxation on a mesh: a formalism for generalized localization. In: Proc. IROS 2001, vol. 2, pp. 1055–1060 (2001), doi:10.1109/IROS.2001.976308
12. Hu, Z., Aro, E., Stipa, T., Vainio, M., Halme, A.: Localization in an autonomous underwater multi-robot system design for coastal area monitoring. In: Proc. ICINCO 2010. INSTICC Press (2010)
13. McFarland, D., Honary, E.: Flock distortion: A new approach in mapping environmental variables in deep water. Robotica 21(4), 365–383 (2003), doi:10.1017/S0263574703004958
14. Moore, D., Leonard, J., Rus, D., Teller, S.: Robust distributed network localization with noisy range measurements. In: Proc. SenSys 2004, pp. 50–61. ACM (2004), doi:10.1145/1031495.1031502
15. Moses, R.L., Krishnamurthy, D., Patterson, R.: A self-localization method for wireless sensor networks. EURASIP Journal on Applied Signal Processing 4, 348–358 (2003), doi:10.1155/S1110865703212063
16. Priyantha, N.B., Balakrishnan, H., Demaine, E., Teller, S.: Anchor-free distributed localization in sensor networks. Tech. Rep. 892, MIT Laboratory for Computer Science (2003)
17. Sanford, T., Dunlap, J., Carlson, J., Webb, D., Girton, J.: Autonomous velocity and density profiler: Em-apex. In: Proc. CMTC 2008, pp. 152–156 (2005), doi:10.1109/CCM.2005.1506361
18. Stipa, T., et al.: Short-term effects of nutrient reductions in the North Sea and the Baltic Sea as seen by an ensemble of numerical models, MERI, Report series of the Finnish Institute of Marine Research, vol. 49 (2003)
19. Vadov, R.: The discovery of the underwater sound channel, experimental studies, and regional differences. Acoustical Physics 53, 268–281 (2007), doi:10.1134/S1063771007030049
20. Vainio, M., et al.: Autonomous underwater multiprobe system for coastal area/shallow water monitoring (swarm). In: Proc. Eurocean 2004, pp. 407–408 (2004)

Distributed Algebraic Connectivity Maximization for Robotic Networks: A Heuristic Approach

Andrea Simonetto, Tamás Keviczky, and Robert Babuška

Abstract. We consider a weighted communication graph in a network of mobile robots, and its associated Laplacian whose entries depend on the pairwise distance between the robots. We propose a heuristic distributed solution for the maximization of the algebraic connectivity of the graph by moving the robots to appropriate positions. Our approach is optimization-based and can be extended to handle various constraints, such as the robots' dynamics. Our proposed distributed solution uses local algorithms that utilize information only from nearby neighboring robots. Numerical simulations show the applicability and effectiveness of the algorithm and indicate that in certain cases the proposed distributed solution can perform better than the centralized version.

1 Introduction

Groups of autonomous mobile robots that communicate with one another to achieve a common goal are considered as a key enabling technology in several applications ranging from underwater and space exploration [1, 2], to search and rescue [3], fire monitoring [4] and other surveillance applications [5]. These robotic teams are envisioned to possess on-board processing capability, but the common task can only be achieved through information exchange among the members and possibly a base station. Such multi-vehicle teams are thus often referred to as robotic networks. Among the several engineering and research questions these applications pose, maintaining connectivity between the individual robots and increasing the communication quality given the environmental constraints and objectives, have fundamental importance. Many different types of coordination and control frameworks that have been proposed recently for cooperating robotic teams rely on some type of agreement protocol or consensus process that leads to coordinated team actions [6, 7, 8].

Andrea Simonetto · Tamás Keviczky · Robert Babuška
Delft Center of Systems and Control, Delft Technical University, Mekelweg 2,
2628 CD Delft, The Netherlands
e-mail: {a.simonetto,t.keviczky,r.babuska}@tudelft.nl

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 267–279.
springerlink.com

Since these protocols typically assume only local communication among "neighboring" units, the interconnection topology of the underlying communication graph influences their effectiveness profoundly. Motivated by the significant role it plays in the performance of many distributed control methods, we study distributed solutions for maximizing the algebraic connectivity of the communication graph (often denoted as λ_2) in mobile robotic networks. This parameter is the second smallest eigenvalue of the communication graph's Laplacian matrix, and it dictates the convergence properties of consensus protocols [9, 10]. We focus on distance-based connectivity maximization with minimum separation constraints, as opposed to ensuring line-of-sight connectivity in an obstacle-rich environment [11]. Maximization of λ_2 is also important for collaborative target tracking [12], where a network of mobile robots strive for increased accuracy of the joint position estimate of one or more moving objects [13, 14, 15, 16]. Besides an increase in accuracy, a positive λ_2 also ensures that the network stays connected during the collective motion.

A few examples of decentralized λ_2 maximization have appeared in the literature so far. These are typically either limited to only specific scenarios, or imply heavy communication requirements. Often the proposed approaches are not derived from a centralized solution, in other words the formulated local problems are not directly related to the solution of the centralized one. Without such a consistency, there are typically no guarantees that the algebraic connectivity is maximized. The approach in [17] uses a two-step distributed solution, which relies on supergradients and potential functions. The required communication load scales with the square of the graph diameter. Other approaches proposed in [18] and [19] make use of auctions and game theory, respectively, and consider maintaining connectedness of the graph as the main priority. Although the communication requirement is limited in these algorithms, they are designed via a bottom-up approach, i.e., starting from local problems, and a potential increase of λ_2 is usually a simple by-product of their solution without analytical guarantees.

In this paper, we present a heuristic distributed approach for the λ_2 maximization problem as formulated by [20, 21, 12] in a centralized framework. Our perspective is model-based optimization, which allows additional constraints (e.g., the dynamics of the robots) to be included explicitly in the problem formulation. Moreover, we believe that this approach can eventually lead to a certain type of consistency with regards to the centralized solution. The proposed solution can also be extended to incorporate other interesting scenarios, such as collaborative target tracking. The proposed distributed approach relies on local problems that are solved by each robot using information only from nearby neighbors. Specifically, two communication policies are introduced to respect the potentially limited communication and computation capabilities of the robots. Simulation results support the efficacy of the approach and show interesting properties of the algorithms. For instance, given the nonlinear/nonconvex nature of the problem, in certain scenarios the distributed solutions converge to a higher λ_2 value than the centralized ones.

The paper is organized as follows. Section 2 formulates the centralized problem as suggested by [20, 21, 12]. The proposed distributed approach and communication policies are described in Section 3. Numerical simulations are shown in Section 4

to assess the performance of the distributed solutions with respect to centralized schemes. Conclusions and open issues are discussed in Section 5.

2 Problem Formulation

We consider a network of N agents. The agents represent mobile robots and the network encodes undirected communication links, meaning that if two agents are connected, they can communicate with each other. As a general notation $a_i(k)$ represents the value of the variable a for agent i at time k. Let $x(k) \in \mathbb{R}^{2N}$ be the collection of the agents' positions on a 2-D plane, i.e., $x(k) = (x_1^\top(k), \ldots, x_N^\top(k))^\top$. Although our scheme can be extended to more complicated robot dynamics, for simplicity of exposition we consider agents with the following discrete-time dynamics

$$x_i(k) = x_i(k-1) + v_i(k-1)\Delta t \tag{1}$$

where $v_i(k)$ is the velocity control input and Δt the sampling time. We use graph-theoretical tools to model the network. The set $\mathscr{S}$ contains the indices of the mobile agents (nodes), with cardinality $N = |\mathscr{S}|$. We use $\mathscr{E}$ to indicate the set of communication links, i.e., the edges $\{(i,j)|i,j \in \mathscr{S}\}$. The graph $\mathscr{G}$ is then expressed as $\mathscr{G} = (\mathscr{S}, \mathscr{E})$. Let the graph be connected initially, the agent clocks synchronized, and assume perfect communication (no delays or packet losses). The agents with which agent i communicates are called neighbors and are contained in the set $\mathscr{N}_i$. Note that node i is not included in the set $\mathscr{N}_i$. We define $\mathscr{J}_i = \mathscr{N}_i \cup \{i\}$ and $N_i = |\mathscr{J}_i|$.

We define a set of Laplacian matrices $\mathscr{L}$ associated with $\mathscr{G}$ as

$$\mathscr{L} = \{L \in \mathbb{R}^{N\times N} | L = L^\top, \ell_{ij} = 0 \text{ iff } (i,j) \notin \mathscr{E}, L\mathbf{1} = \mathbf{0}\}$$

The entries of a Laplacian matrix L are defined as

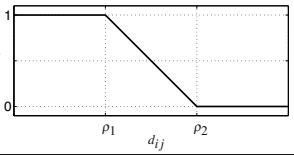
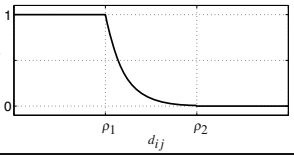
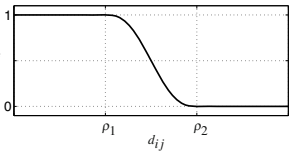
$$\ell_{ij} := \begin{cases} 0 & (i,j) \notin \mathscr{E} \\ -w_{ij} & (i,j) \in \mathscr{E}, i \neq j \\ \sum_{l\neq i} w_{il} & i = j \end{cases} \tag{2}$$

where the positive weights w_{ij} represent the "connection strength" between agents i and j. The weights themselves depend on the physical distance between the agents. For this purpose we introduce the square distance matrix D, whose entries d_{ij} are defined as

$$d_{ij} = ||x_i(k) - x_j(k)||^2. \tag{3}$$

The value of the normalized weights will be 1 representing a "strong connection" if d_{ij} is less than a certain threshold, i.e., $d_{ij} \leq \rho_1$, with $\rho_1 > 0$. On the other hand, agents will not be connected at all ($w_{ij} = 0$) for $d_{ij} > \rho_2$, with $\rho_2 > \rho_1$. For $\rho_1 < d_{ij} \leq \rho_2$ the agents are connected with a connection strength that decreases smoothly with their distance. Typically, spatially decaying functions are used for the weights w_{ij} [22], and a few of them are shown in Table 1. Case (1) is a linear representation which is continuous but not differentiable, case (2) is the exponential function of [12], which is not differentiable and also discontinuous at ρ_2, while case (3) is a

Table 1 Possible choices of weighting functions. Case (1) is a linear representation, case (2) is the exponential function of [12], while case (3) is a 5-th order polynomial description.

Case	Function	Figure
(1)	$w_{ij} := \begin{cases} 1 & d_{ij} < \rho_1 \\ \frac{1}{\rho_2-\rho_1}(\rho_2 - d_{ij}) & \rho_1 \le d_{ij} < \rho_2 \\ 0 & d_{ij} \ge \rho_2 \end{cases}$	
(2)	$w_{ij} := \begin{cases} 1 & d_{ij} < \rho_1 \\ \exp\left(-\frac{5(d_{ij}-\rho_1)}{\rho_2-\rho_1}\right) & \rho_1 \le d_{ij} < \rho_2 \\ 0 & d_{ij} \ge \rho_2 \end{cases}$	
(3)	$w_{ij} := \begin{cases} 1 & d_{ij} < \rho_1 \\ \sum_{p=0}^{5} \alpha_p d_{ij}^p & \rho_1 \le d_{ij} < \rho_2 \\ 0 & d_{ij} \ge \rho_2 \end{cases}$	

polynomial description, which for a suitable choice of the coefficients α_p is both continuous and twice-differentiable.

As a direct consequence of the above definitions, the entries of the Laplacian matrix (2) will depend on the pairwise distance and therefore the position states of the robots, making it state-dependent, which we will denote by $L(x)$. We are interested in the maximization of the algebraic connectivity of the weighted graph by moving the robots to appropriate positions. This goal can be formulated as the following optimization problem [21]:

$$\mathbf{P}(L(x)): \quad \max_{x,\gamma} \ \gamma \tag{4a}$$

$$\text{s.t.} \ \ \gamma > 0 \tag{4b}$$

$$L(x) + \mathbf{1}\mathbf{1}^T \succ \gamma I \tag{4c}$$

where the decision variables are γ and the robot locations x. The optimal value of γ will be the maximum λ_2 for $L(x)$.

This problem would be convex if L was the decision variable, but it is non-convex given that we are optimizing over the positions x and the entries of L are nonlinear functions of x. However, we can obtain an iterative convex approximation following the steps of [20]. First we differentiate (3) with respect to time as

$$2(\dot{x}_i(k-1) - \dot{x}_j(k-1))^\top (x_i(k-1) - x_j(k-1)) = \dot{d}_{ij}(k-1)$$

and then we employ Euler's first-order discretization method to rewrite (3) as

$$d_{ij}(k) = -d_{ij}(k-1) + 2(x_i(k) - x_j(k))^\top (x_i(k-1) - x_j(k-1))$$

In the same way, the weights of the state-dependent Laplacian $L(x)$ are discretized as

$$\begin{aligned} w_{ij}(k) &= w_{ij}(k-1) + \left.\frac{\partial w_{ij}}{\partial d_{ij}}\right|_{d_{ij}(k-1)} (d_{ij}(k) - d_{ij}(k-1)) \\ &= w_{ij}(k-1) + 2\left.\frac{\partial w_{ij}}{\partial d_{ij}}\right|_{d_{ij}(k-1)} (x_i(k) - x_j(k) - x_i(k-1) + x_j(k-1))^\top (x_i(k-1) - x_j(k-1)) \end{aligned}$$

This allows us to consider the maximization of the algebraic connectivity of L as the following iterative convex semi-definite programming (SDP) problem:

$$\mathbf{P}_k\left(L(x), x(k-1), D(k-1), v_{\max}\right):$$

$$\max_{x(k), D(k), \gamma(k)} \quad \gamma(k) \tag{5a}$$

$$\text{s.t.} \quad \mathscr{Q}_1 : \begin{cases} \gamma(k) > 0 \\ L(x(k)) + \mathbf{1}\mathbf{1}^T \succ \gamma(k) I \end{cases} \tag{5b}$$

$$\mathscr{Q}_2 : \begin{cases} \mathscr{Q}_{2.1} : d_{ij}(k) + d_{ij}(k-1) - 2(x_i(k) - x_j(k))^\top (x_i(k-1) - x_j(k-1)) = 0 \\ \mathscr{Q}_{2.2} : \quad d_{ij}(k) > \rho_1, \quad \forall (i,j) \in \mathscr{E} \\ \mathscr{Q}_{2.3} : \quad ||x_i(k) - x_i(k-1)|| \leq v_{\max} \Delta t \quad i = 1, \ldots, N \end{cases} \tag{5c}$$

where the constraint $\mathscr{Q}_{2.2}$ is used both to avoid agents getting too close to each other and to restrict the distances to be positive. This is not automatically ensured by $\mathscr{Q}_{2.1}$ if the agents could move arbitrarily fast. The constraint $\mathscr{Q}_{2.3}$ on the velocity represents the physical limitations of the agents.

The problem $\mathbf{P}_k(L(x), x(k-1), D(k-1), v_{\max})$ in (5) is solved iteratively in each sampling time step and its decision variables are $x(k)$, $D(k)$, and $\gamma(k)$. The problem formulation depends on the values $x(k-1)$ and $D(k-1)$ from the previous time step $k-1$. Here k stands both for the iteration counter and for the discrete time index since in this problem the two concepts are equivalent. Since $x(k)$ and $D(k)$ are considered independent, there could be a possible inconsistency between the distances and the actual position of the agents. This effect can be diminished if in addition to constraint $\mathscr{Q}_{2.1}$, $D(k-1)$ is recomputed based on $x(k-1)$ before each optimization step. Although the original problem (5) can be proven to converge to a local maximum [20], this property may be lost when recomputing $D(k-1)$. This is due to the fact that the λ_2 of $L(x(k-1))$, based on the recomputed $D(k-1)$, may be smaller than $\gamma(k-1)$. However, in the simulated scenarios we consider in Section 4, the algorithm using recomputed $D(k-1)$ has always converged.

Remark 1. We note that in [20] the requirement that the distances $d_{ij}(k)$ form the entries of a square Euclidean Distance Matrix is considered as an additional convex constraint. This would help in reducing the inconsistency effect between $D(k)$ and $x(k)$. However, our experience indicates that this introduces extra rotational rigidity to the graph in the numerical simulations and as a result it has not been included in our problem setup.

The optimization problem that has been described in this section attempts to solve the connectivity maximization problem in a centralized manner using linearization, discretization and an iterative solution approach. In realistic application scenarios

however, computing the desired positions and the corresponding motion commands for the robots cannot be performed in a single centralized location due to computational and communication constraints. In the next section, we describe a solution approach that allows the problem to be solved in a distributed fashion, using local computations and limited communication resources, which increases the flexibility of the robotic network and is thus appealing in practice.

3 The Proposed Distributed Approach

In this section we present a distributed approach to solve (5). First, we introduce necessary notation and definitions, then describe our heuristic method and argue why solving local problems leads to a non-decreasing sequence of algebraic connectivity, when considering the linearized Laplacian of the overall network.

In order to describe the local problems each agent will be solving, we define subgraphs that correspond to the agents and their neighborhood. Let $\mathscr{M}_i$ denote the *enlarged* neighborhood for each agent i defined as

$$\mathscr{M}_i = \bigcup_{l \in \mathscr{J}_i} \mathscr{J}_l, \qquad i = 1,\dots,N \tag{6}$$

whose cardinality will be M_i. We denote the vector containing all the positions of the agents in the set with $x_{\mathscr{M}_i}$, while we call the set of agents belonging to $\partial\mathscr{M}_i$, the bordering agents of $\mathscr{M}_i$ defined as

$$\partial\mathscr{M}_i = \{l | l \in \mathscr{M}_i, l \notin \mathscr{J}_i\}, \qquad i = 1,\dots,N \tag{7}$$

Figure 1 provides a graphical illustration of this notation. In some situations we will consider a randomly selected connected subset of $\mathscr{M}_i$ that includes agent i. This set will be denoted by $R(\mathscr{M}_i)$ with cardinality RM_i. Following suit, we also define random versions of the border set $R(\partial\mathscr{M}_i)$ and neighborhood set $R(\mathscr{J}_i)$ with cardinality RN_i as

$$R(\partial\mathscr{M}_i) = \{l | l \in R(\mathscr{M}_i), l \notin \mathscr{J}_i\}, \qquad i = 1,\dots,N \tag{8}$$

$$R(\mathscr{J}_i) = \{l | l \in R(\mathscr{M}_i), l \notin R(\partial\mathscr{M}_i)\}, \qquad i = 1,\dots,N \tag{9}$$

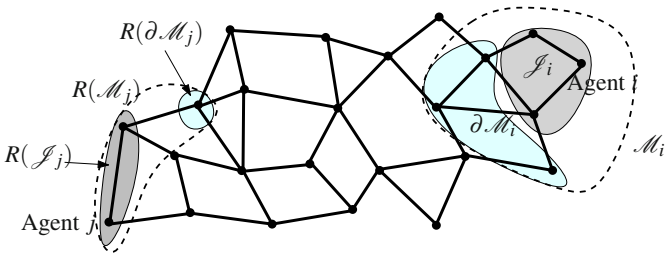


Fig. 1 Notation for the distributed solution

Graphical examples of these definitions are also shown in Figure 1. Finally, we will denote the graph Laplacian associated with subgraph $\mathscr{M}_i$ as L_i with corresponding distance matrix D_i, while the one associated with $R(\mathscr{M}_i)$ as $R(L_i)$ and $R(D_i)$, respectively. We also introduce a scaled maximum velocity $\tilde{v}_{\max,i}$ defined as

$$\tilde{v}_{\max,i} = \left(\sum_{j \in \mathscr{M}_i} \frac{1}{N_j} \right)^{-1} v_{\max}, \quad i = 1, \ldots, N \tag{10}$$

whose value varies from agent to agent. The use of this quantity will be explained later in this section. We consider two possible strategies:

1. Full neighborhood (FN) strategy: the agents are allowed to communicate within the whole enlarged neighborhood $\mathscr{M}_i$. In this case the proposed distributed solution will lead to monotonically increasing connectivity and more importantly, it will respect the constraints on D, meaning that $d_{ij} > \rho_1$ for all i and j. However, the communication requirements and local problem size will increase as the agents get closer to each other and increase their connectivity.
2. Random neighborhood (RN) strategy: the agents are allowed to communicate only with a randomly selected subset of their extended neighborhood $R(\mathscr{M}_i)$. In this case, the overall connectivity may no longer increase monotonically and constraints on D may not always be fulfilled. However, the communication and local problem size can be significantly reduced.

Our algorithms consist of two steps. First, each agent solves the problem $\mathbf{P}_{k,i}^{\mathrm{FN}}$ defined as

$$\mathbf{P}_k(L_i(x_{\mathscr{M}_i}), x_{\mathscr{M}_i}(k-1), D_i(k-1), \tilde{v}_{\max,i}) \tag{11a}$$

$$\text{s.t.} \quad \mathscr{Q}_3 : \; x_j(k) = x_j(k-1), \qquad \text{for } j \in \partial\mathscr{M}_i \tag{11b}$$

computing the solution $\hat{x}_{\mathscr{M}_i}(k)$, which is composed of $\hat{x}_{ij}(k)$ for each $j \in \mathscr{M}_i$. Thus, we will call $\hat{x}_{ij}(k)$ the position of agent j as computed by agent i. Note the importance of the extra constraint $\mathscr{Q}_3$ that guarantees monotonically increasing connectivity as will be explained later in this section.

As the second step, the solutions $\hat{x}_{\mathscr{M}_i}(k)$ are shared within the enlarged neighborhood $\mathscr{M}_i$ and averaged according to

$$x_i(k) = x_i(k-1) + \sum_{j \in \mathscr{M}_i} \frac{1}{N_j} \left(\hat{x}_{ji}(k) - x_i(k-1) \right), \qquad i = 1, \ldots, N \tag{12}$$

Algorithm 3 summarizes the method for the FN strategy as described above. For the RN strategy, the algorithm follows the same scheme with the following substitutions: $\mathbf{P}_{k,i}^{\mathrm{FN}} \to \mathbf{P}_{k,i}^{\mathrm{RN}}$, $\mathscr{M}_i \to R(\mathscr{M}_i)$, $\partial\mathscr{M}_i \to R(\partial\mathscr{M}_i)$, $\hat{x}_{\mathscr{M}_i} \to \hat{x}_{R(\mathscr{M}_i)}$, $L_i(\cdot) \to R(L_i(\cdot))$, $D_i \to R(D_i)$, $M_i \to RM_i$, and $N_j \to RN_j$.

The heuristics presented in the above algorithm lead to a solution with monotonically increasing connectivity, i.e., if we consider the resulting global position vector $x(k) = (x_1^\top(k), \ldots, x_N^\top(k))^\top$, the algebraic connectivity of the corresponding global linearized Laplacian $L(x(k))$ would be monotonically increasing in each iteration. In order to justify our algorithm and ensure this property, the extra constraint $\mathscr{Q}_3$ on the border set is necessary. It allows us to show that $L(x(k)) - L(x(k-1)) \succeq 0$ where $x(k)$ is the collection of the locally averaged $x_i(k)$ solutions. This property

Algorithm 1. Distributed Algebraic Connectivity Maximization for FN strategy

1: Input: $x_i(k-1)$, $x_j(k-1)$, $j \in \mathscr{M}_i$
2: Compute: $d_{ij}(k-1)$ from input based on (3)
3: Solve: $\mathbf{P}^{\mathrm{FN}}_{k,i}$ computing $\hat{x}_{ij}(k)$, $j \in \mathscr{M}_i$
4: Communicate: $\hat{x}_{ij}(k)$ among members of $\mathscr{M}_i$
5: Average: $x_i(k) = x_i(k-1) + \sum_{j \in \mathscr{M}_i} \frac{1}{N_j} \left(\hat{x}_{ji}(k) - x_i(k-1)\right)$
6: Output: $x_i(k)$

follows from the following line of arguments. Consider the local problem $\mathbf{P}^{\mathrm{FN}}_{k,i}$ and its solution comprised of $\hat{x}_{ij}(k)$ for all $j \in \mathscr{M}_i$. Construct a global vector as

$$\hat{x}^{(i)}(k) = (x_1^\top(k-1), \ldots, \hat{x}_{ij}^\top(k), \ldots, x_N^\top(k-1))^\top \tag{13}$$

where we keep those agent positions that have not been optimized fixed, and we update the rest from the solution of the local problem. It is relatively straightforward to see that due to constraint $\mathscr{Q}_3$, $L(\hat{x}^{(i)}(k)) - L(x(k-1)) \succeq 0$, meaning that the new positions $\hat{x}^{(i)}(k)$ do not decrease the algebraic connectivity of the Laplacian matrix. This trivially implies $(L(\hat{x}^{(i)}(k)) - L(x(k-1)))/N_i \succeq 0$ for all i. Thus summing over all agents leads to

$$\sum_{i=1}^{N} \frac{1}{N_i} (L(\hat{x}^{(i)}(k)) - L(x(k-1))) \succeq 0 \tag{14}$$

Considering the weighted sum $x_i(k)$ in (12), and the associated global vector $x(k)$, it can be shown that

$$L(x(k)) = \sum_{i=1}^{N} \frac{1}{N_i} L(\hat{x}^{(i)}(k)) \tag{15}$$

which leads to the desired monotonicity property

$$L(x(k)) \succeq \sum_{i=1}^{N} \frac{1}{N_i} L(x(k-1)) \succeq L(x(k-1)) \tag{16}$$

given that $\sum_{i=1}^{N} \frac{1}{N_i} \geq 1$. With similar arguments, it is possible to argue that feasibility of the local problem constraints imply feasibility of the centralized problem. The above discussion also elucidates the reason for scaling the maximum velocity in the local problems by $(\sum_{j \in \mathscr{M}_i} \frac{1}{N_j})^{-1}$.

Remark 2. In our numerical experiments we have not encountered any infeasibility when using the FN strategy and the original $\mathscr{Q}_{2.2}$ constraint in the local problems (mainly due to the particular choice of the weighting functions and only a few neighbors for each robot). However, in principle, the constraint $\mathscr{Q}_{2.2}$ should be tightened as well by introducing local $\tilde{\rho}_{1_{ij}} \leq \rho_1$. We are currently investigating the most suitable way of incorporating these tightened constraints in the local problem formulations.

Finally, we present a justification for the choice of the enlarged neighborhood set $\mathscr{M}_i$ using the following example. Consider the interconnection shown in Figure 2

and assume that instead of the enlarged neighborhood $\mathscr{M}_i$, the smaller neighborhood $\mathscr{J}_i$ is used. In that case the consistency constraint in $\mathbf{P}_{k,i}^{\mathrm{FN}}$ requires agents $i+1$ and $i-1$ to be fixed. It is easy to see that if the distance between the agents is already at the lower limit $\sqrt{\rho_1}$, all the agents will remain stationary. However, if we considered the enlarged neighborhood $\mathscr{M}_i$ in the local problem instead, the situation would be different. The bordering agents 1 and N would rotate towards the center of the string, to connect with 3 and $N-2$, respectively.

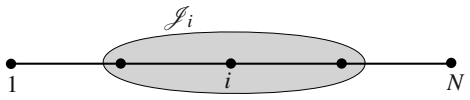


Fig. 2 Illustrative example for justifying the choice of the extended neighborhood set

4 Simulation Results

In this section, we present numerical simulation results to illustrate how the different algorithms perform with respect to the centralized scheme. In particular, we will analyze first the FN strategy and observe that, in some cases, it converges to a higher λ_2 value than the centralized solution. Then we proceed to investigate RN strategies, which lead to reduced communication load for the price of losing the monotonically increasing connectivity property and persistent feasibility of the minimum distance constraint $\mathscr{Q}_{2.2}$. We use the benchmark problem of [20] to relate our results to the literature. This scenario starts with $N=6$ agents on a line forming a connected graph. The initial position vector is $x_i(0)=[1+1.05(i-1),y_i]^\top$, with $y_i \sim (0,\sigma)$, meaning that y_i is drawn from a Gaussian distribution $(0,\sigma)$, with mean 0 and standard deviation $\sigma=0.1$. Randomness is added to test the algorithms' sensitivity to slightly different initial conditions. The other simulation parameters include a weight function of type (3) in Table 1, $\rho_1=0.5$, $\rho_2=2$, velocity bound of 0.2, and final time $T=100$. We collected 50 simulation runs for 4 different cases: (1) FN strategy, (2,3,4) RN strategy with the ratio RM_i/M_i set to $0.75, 0.50$, and 0.25, respectively. We call r_{λ_2} the ratio between the converged λ_2 of the distributed solution and the one from the centralized solution. Therefore, if $r_{\lambda_2}>1$ the distributed solution has better performance than the centralized one. In Figure 3 an example of the trajectories of the centralized and the distributed solutions for the FN strategy is depicted. The initial positions are marked with squares. The final positions are marked with circles. The bold lines represent the final communication graph and the thin lines the agent trajectories. The values of $\sqrt{\rho_1}$ and $\sqrt{\rho_2}$ are also depicted for comparison. Figure 4 shows, in the same simulation, the evolution of the algebraic connectivity as a function of the sampling time k. We can observe that although in this case the centralized solution converges faster to the final configuration, the distributed approach eventually converges to a higher final algebraic connectivity value. We can also notice "plateaus" during the convergence of the algebraic connectivity, where the agents are rotating and λ_2 is not changing significantly.

Figure 5 represents the ratio between the final distributed and centralized solutions, i.e., r_{λ_2} in all four cases. The disconnected label refers to situations in which

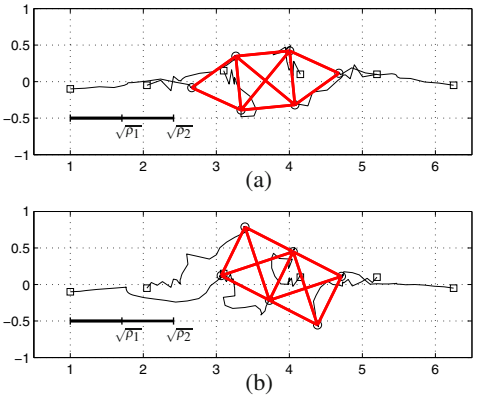


Fig. 3 Trajectories for the centralized solution (a) and for the distributed approach using the FN strategy (b). The initial positions are marked with squares. The final positions are marked with circles. The bold lines represent the final communication graph and the thin lines the agent trajectories.

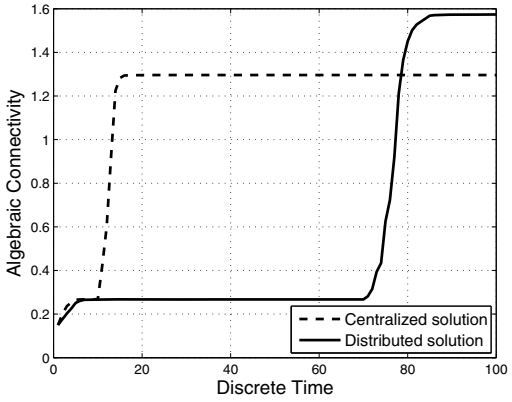


Fig. 4 Algebraic connectivity as a function of discrete time k for both the centralized and the distributed solutions

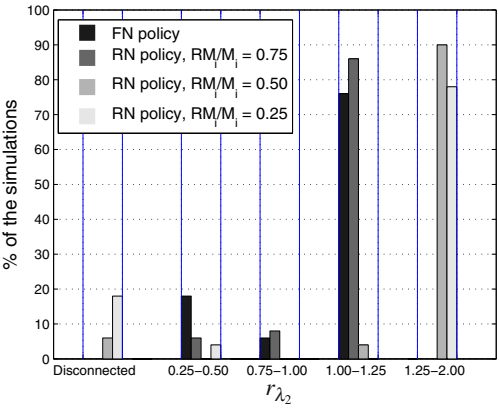


Fig. 5 Ratio between the final distributed and centralized solutions, i.e., r_{λ_2} and the percentage of corresponding simulations. (There are no cases that fall in the segment $0.50 - 0.75$).

the RN strategies lead to a disconnected graph. From the simulation results, we can observe that the FN strategy has performance comparable to the centralized solution in most cases. It may even converge to a higher λ_2 value in some instances, and it could get stuck in local minima in certain cases, which are not present in

the centralized algorithm. Investigation of these local minima is a topic of ongoing research. The behavior of the RN strategies differs from the FN strategy for two reasons: one is the absence of a monotonically increasing connectivity property and the other is a possible infeasibility of the local problems. One consequence of this is that decreasing the ratio RM_i/M_i is more likely to lead to an increasing number of disconnected final graphs. We can also observe a clear increase in performance for $RM_i/M_i = 0.50$ and 0.25. This can be expected since the minimum distance constraints are no longer enforced in a consistent manner, and some agents are allowed to be arbitrarily close to each other if they are excluded from the local problem formulation. This also means that, in some cases, local problems can become infeasible. Infeasible local problems were handled in the simulations by keeping the previous positions, i.e., $x_i(k) = x_i(k-1)$. For small RM_i/M_i ratios this led to all local problems eventually becoming infeasible and all pairwise distances becoming smaller than $\sqrt{\rho_1}$. The ratio RM_i/M_i can be considered as a tuning parameter to obtain a reduction of communication. On one hand, for $RM_i/M_i = 1$ we have possibly high communication load, on the other hand for $RM_i/M_i \to 0$, we have limited communication for the price of sacrificing the monotonically increasing connectivity property. This loss however does not necessarily lead to either disconnected graphs or distances smaller than $\sqrt{\rho_1}$. The choice of 0.75 serves as an example for this phenomenon.

5 Future Developments and Open Questions

We have presented a heuristic distributed solution for the maximization of algebraic connectivity in a network of mobile robots. The method is optimization-based and can be further extended by including other types of constraints. Our approach may be used to obtain a monotonically increasing connectivity property and it can be easily understood based on the existing centralized solution. We presented simulation results for different communication strategies to assess the performance of the method, and we highlighted cases in which the distributed solution converges to a higher λ_2 value than the centralized scheme, along with cases in which it converges to local minima. Several open issues still remain and will be the focus of our future research. In particular, a study of the inconsistency between real and linearized distance $D(k)$, a more realistic dynamical model for the agents, and an investigation of the theoretical properties of both the FN and RN strategies will be considered along with experimental validations. Furthermore, we will investigate possible dual decomposition methods to distribute (5) among the robots, while expecting that the resulting iterative solutions could compromise real-time applicability. Such a dual decomposition approach would typically provide primal feasible solutions only asymptotically, and would require investigating various issues, such as the duality gap. On a longer time scale, other interesting topics of research are how to extend the problem formulation to handle more realistic scenarios, such as obstacle avoidance and environment-dependent connectivity.

References

1. Bhatta, P., Fiorelli, E., Lekien, F., Leonard, N.E., Paley, D.A., Zhang, F., Bachmayer, R., Davis, R.E., Fratantoni, D.M., Sepulchre, R.: Coordination of an Underwater Glider Fleet for Adaptive Sampling. In: Proceedings of the International Workshop on Underwater Robotics, Genoa, Italy, pp. 61–69 (2005)
2. Izzo, D., Pettazzi, L.: Autonomous and Distributed Motion Planning for Satellite Swarm. Journal of Guidance, Control and Dynamics 30(2), 449–459 (2007)
3. Lau, H.Y.K., Ko, A.W.Y.: Coordination of Cooperative Search and Rescue Robots for Disaster Relief. In: Proceedings of IFAC World Congress, Seul, Korea, July 6-11 (2008)
4. Casbeer, D.W., Li, S., Beard, R.W., Mehra, R.K.: Forest Fire Monitoring With Multiple Small UAVs. In: Proceedings of the ACC, Portland, OR, USA, June 8-10 (2005)
5. Mathews, G., Durrant-Whyte, H.: Decentralised Optimal Control for Reconnaissance. In: Proceedings of the Conference on Information, Decision and Control, Adelaide, Australia, February 12-14 (2007)
6. Bullo, F., Cortés, J., Martínez, S.: Distributed Control of Robotic Networks. Applied Mathematics Series. Princeton University Press (2008)
7. Keviczky, T., Johansson, K.H.: A Study on Distributed Model Predictive Consensus. In: Proceedings of IFAC World Congress, Seul, Korea, July 6-11 (2008)
8. Ren, W., Beard, R.: Distributed Consensus in Multi-Vehicle Cooperative Control: Theory and Applications. Springer, London (2008)
9. Olfati-Saber, R., Murray, R.: Consensus Problems in Networks of Agents with Switching Topology and Time-Delays. IEEE Transactions on Automatic Control 49(9), 1520–1533 (2004)
10. Olfati-Saber, R., Fax, J.A., Murray, R.: Consensus and Cooperation in Networked Multi-Agent Systems. Proceeding of IEEE 95(1), 215–233 (2007)
11. Anisi, D.A., Hu, X., Ögren, P.: Connectivity Constrained Multi-UGV Surveillance. In: Proceedings of the 17th IFAC World Congress, Seoul, Korea (July 2008)
12. Derenick, J., Spletzer, J., Hsieh, A.: An Optimal Approach to Collaborative Target Tracking with Performance Guarantees. Journal of Int. Rob. Syst. 56, 47–67 (2009)
13. Grocholsky, B., Makarenko, A., Durrant-Whyte, H.: Information-Theoretic Coordinated Control of Multiple Sensor Platforms. In: Proceedings of the 2003 IEEE International Conference on Robotics and Automation, Taipei, Taiwan, September 14-19 (2003)
14. Chung, T.H., Gupta, V., Burdick, J.W., Murray, R.M.: On a Decentralized Active Sensing Strategy using Mobile Sensor Platforms in a Network. In: Proceedings of the 43rd IEEE Conference on Decision and Control, Paradise Island, the Bahamas, December 14-17 (2004)
15. Martínez, S., Bullo, F.: Optimal sensor placement and motion coordination for target tracking. Automatica 42, 661–668 (2006)
16. Olfati-Saber, R.: Distributed Tracking for Mobile Sensor Networks with Information-Driven Mobility. In: Proceedings of the American Control Conference, New York City, USA (2007)
17. De Gennaro, M.C., Jadbabaie, A.: Decentralized Control of Connectivity for Multi-Agent Systems. In: Proceedings of the 45th IEEE Conference on Decision and Control, San Diego, CA, USA, December 13-15 (2006)
18. Zavlanos, M.M., Pappas, G.J.: Distributed Connectivity Control of Mobile Networks. In: Proceedings of the 46th IEEE Conference on Decision and Control, New Orleans, LA, USA, December 12-14 (2007)
19. Schuresko, M., Cortés, J.: Distributed Motion Constraints for Algebraic Connectivity of Robotic Networks. Journal of Intelligent Robotics Systems 56, 99–126 (2009)

20. Kim, Y., Mesbahi, M.: On Maximizing the Second Smallest Eigenvalue of a State-Dependent Graph Laplacian. IEEE Transactions of Automatic Control 51(1), 116–120 (2006)
21. Boyd, S.: Convex Optimization of Graph Laplacian Eigenvalues. In: Proceedings of the International Congress of Mathematicians, Madrid, Spain, pp. 1311–1319 (2006)
22. Cohen, E., Kaplan, H.: Spatially-decaying Aggregation over a Network. Journal of Computer and System Sciences 73, 265–288 (2007)

Beat-Based Synchronization and Steering for Groups of Fixed-Wing Flying Robots

Sabine Hauert, Severin Leven, Jean-Christophe Zufferey, and Dario Floreano

Abstract. Groups of fixed-wing robots can benefit from moving in synchrony to share sensing and communication capabilities, avoid collisions or produce visually pleasing choreographies. Synchronous motion is especially challenging when using fixed-wing robots that require continuous forward motion to fly. For such platforms, performing trajectories with forward speed lower than the minimum speed of the robot can only be achieved by acting on its heading turn rate. Synchronizing such highly dynamical systems would typically require position information and entail frequent sensing and communication among robots within the group. Instead here we propose a simple controller that reacts to regular beats received through wireless transmissions. Thanks to these beats, robot headings synchronize over time. Furthermore, these controllers can easily be parameterized to steer and regulate the global progression speed of groups of robots. Experiments are performed both in simulation and using up to five fixed-wing flying robots.

1 Introduction

Flying robots are often required to follow trajectories that can be easily steered and speed-regulated for real-world applications such as teleoperation, visiting areas of interest, exploration and tracking [14]. In the case of fixed-wing platforms, this means setting the turn rate of the robot such that it will perform loitering trajectories with a given global speed and direction. Indeed, unlike ground robots or rotorcrafts, fixed-wing robots need to maintain their flight velocity within a certain limit to avoid stalling. Loitering allows robots to slow down their global progression speed.

Moreover, some applications can benefit from deploying several robots rather than a single one to increase the number of sensors in the air, produce different

Sabine Hauert · Severin Leven · Jean-Christophe Zufferey · Dario Floreano
Laboratory of Intelligent Systems, Ecole Polytechnique Fédérale de Lausanne, Switzerland
e-mail: {sabine.hauert,severin.leven}@epfl.ch,
{jean-christophe.zufferey,dario.floreano}@epfl.ch

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 281–293.
springerlink.com

points of view and for increased mission robustness [14]. Synchronizing the heading of the robots within a group can further allow them to move coherently in a given direction, which can help them avoid collisions, maintain relative distance among robots for sensor fusion and favor communication [1].

Synchronizing loitering trajectories in real-time across robots while respecting commands in terms of global motion direction and speed is challenging. Work on formation path following for unicycle-type vehicles has so far concentrated on mathematical models and simulations built upon the assumption that robots know the precise relative position of neighbors (range and bearing) and sometimes their heading and speed [2, 3, 8, 10]. Using this knowledge, robots continuously align their position to that of their neighbors and to the trajectory they need to follow. However, so far no results have been demonstrated with real flying robots and simulations only depict scenarios without sensor noise or low forward speeds and limited turning rates unrealistic for flying robots. Furthermore, inferring position in a robust and dependable manner is one of the main challenges in aerial robotics [5, 6].

Instead, here we present a positionless strategy to synchronize and steer groups of robots that does not require memory, computation or high-bandwidth communication. This work is inspired from the idea of emergent synchronization studied in nature [11] and the discovery of synchronized controllers for flying robots using artificial evolution [6]. We consider robots that fly at constant speed and rely on a heading sensor and a low-level autopilot that is able to regulate turn rate with some precision [7]. Based on these assumptions, we propose a minimal controller for flying robots where synchronization emerges from interactions between each robot and rhythmic beats sent using a radio-emitter from a base station on the ground or one of the robots. Each beat is a step function composed of an "on" and "off" phase of fixed duration. Notice that in the most economical mode, only "on" and "off" signals need to be sent. Based on this beat and their heading, robots will change their turn rate to achieve adequate loitering trajectories that display two essential features seen in Fig. 1. First robots, regardless of their initial heading, converge to identical headings over time (synchronization). Second, the direction and speed of the loitering trajectories can be changed to a desired value by mathematically determining the parameters of the controller. These two features are presented in this order in the paper because the steering of the robots has as a prerequisite their synchronization. Finally, results in this paper are shown both in simulation and with up to five physical fixed-wing flying platforms that are fully autonomous.

2 Heading Synchronization

Synchronization is essential to make robots move coherently in groups. To achieve this we developed a controller that reacts to beat signals and heading information. In particular we define a simple controller where robots receiving the "on" phase of a beat of duration t_1 perform a fixed turn rate of ω_1. The "off" phase is then initiated for a duration t_2. During this phase, robots perform a turn rate of ω_1 or ω_2 depending

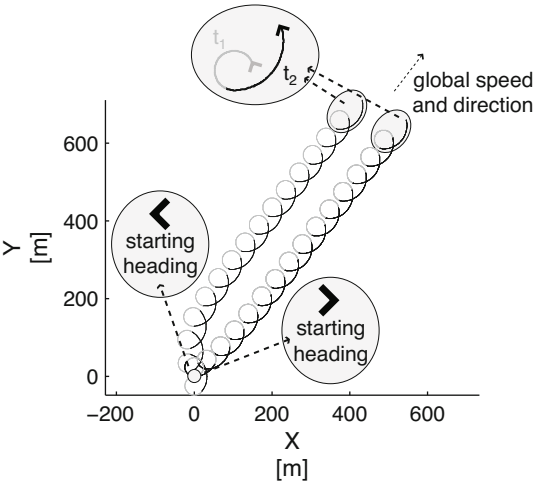


Fig. 1 Example of synchronized steering. Robots launched from (0,0) in opposite directions receive a beat composed of an "on" phase of duration t_1 (grey) and an "off" phase of duration t_2 (black). Over time, the robot headings synchronize. This can be seen by the fact that at each start of a beat, the headings of the two robots are identical. Furthermore, the trajectories converge to a fixed global velocity (speed and direction).

on whether the angle $\widehat{h_{lim}h}$ between a predefined heading limit h_{lim} and their current heading h is positive or negative. ω_1 and ω_2 are assumed to be of same sign. The resulting controller is described below and a possible trajectory is shown in Fig. 2.

$$\omega = \begin{cases} \omega_1, & \text{if beat on} \\ \omega_1, & \text{if beat off and } \widehat{h_{lim}h} < 0 \\ \omega_2, & \text{if beat off and } \widehat{h_{lim}h} > 0 \end{cases} \tag{1}$$

This controller has the property of converging to identical headings at the beginning of each beat. This can be explained by the fact that the amount of time t_{lim} spent between the moment the beat is turned off and the robot reaches the heading limit h_{lim} depends on the initial heading of the robot. If the robot starts at a heading as shown in Fig. 3 (left), it will perform more than 2π within one beat ($t_1 + t_2$), thereby changing its starting heading for the next beat. However, if the robot starts at the

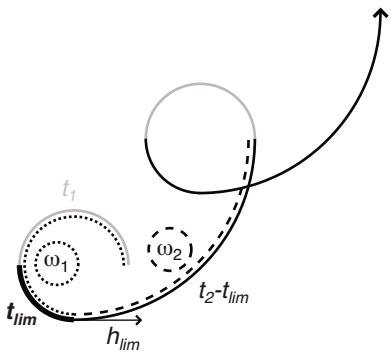


Fig. 2 Example of a robot trajectory implementing the controller described in equation 1. Here a robot receives a beat composed of an "on" phase of duration t_1 (grey) and an "off" phase of duration t_2 (black). Using this, the robot controller sets the turn rate to ω_1 or ω_2 depending on the beat and the heading of the robot with respect to a predefined heading h_{lim} (see dashed lines).

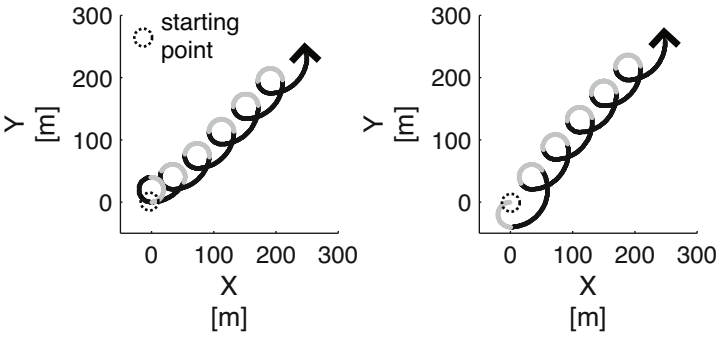


Fig. 3 Robot trajectories synchronize over time by converging to a state where at the beginning of each beat of duration t_1 (grey) + t_2 (black), the robot returns to the same heading. Synchronization is achieve independently of the initial heading of the robot as shown in these two examples with opposite initial headings. Notice that at the end of the trajectory, robot headings are identical.

heading shown in Fig. 3 (right), it will perform less than 2π within one beat. Instead, once synchronized, the robot will perform 2π during one beat, meaning that it will start the next beat with the same heading.

The overall effect is that robots listening to identical beats and using the same controller parameters will synchronize over time. In the particular case shown in Fig. 4, robots synchronize after 2 beats.

Assuming $\omega_1 > \omega_2$, suitable parameters (t_1, t_2, ω_1 and ω_2) that lead to trajectories that perform 2π during one beat must be such that the minimum value for t_{lim} named t_{lim_min} produces trajectories that perform less than 2π during one beat while the maximum value t_{lim_max} produces trajectories that perform more than a full revolution during one beat. These conditions can be mathematically described as:

$$|\omega_1| \cdot t_1 + |\omega_1| \cdot t_{lim\_min} + |\omega_2| \cdot (t_2 - t_{lim\_min}) < 2\pi \tag{2}$$

$$|\omega_1| \cdot t_1 + |\omega_1| \cdot t_{lim\_max} + |\omega_2| \cdot (t_2 - t_{lim\_max}) > 2\pi \tag{3}$$

where

$$t_{lim\_min} = t_2 - min(t_2, \frac{\pi}{|\omega_2|}) \tag{4}$$

$$t_{lim\_max} = min(t_2, \frac{\pi}{|\omega_1|}) \tag{5}$$

Notice that setting $\omega_2 > \omega_1$ would simply result in reverting the inequalities.

As an advantage, this controller is able to compensate for perturbations and resynchronize. This is shown in Fig. 5 where we introduce 8 large perturbations to the system by increasing or decreasing the turn rate ω_1 and ω_2 by 0.05 rad/s and 0.1 rad/s during an entire beat.

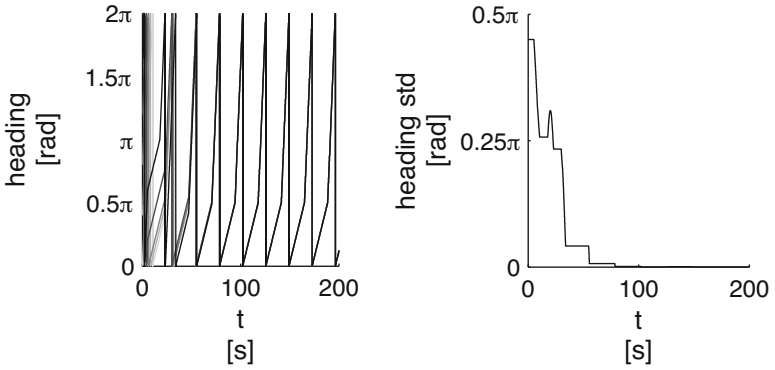


Fig. 4 Headings (left) of 5 simulated robots initialized at headings 0, $\frac{2\pi}{5}$, $\frac{4\pi}{5}$, $\frac{6\pi}{5}$, $\frac{8\pi}{5}$. Notice that over time, the standard deviation (right) across robot headings goes down to 0, meaning the robot are synchronized.

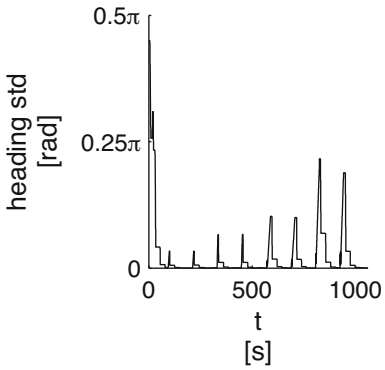


Fig. 5 Capacity of the robot controller to synchronize after 8 large perturbations to its turn rate

3 Global Motion

For real-world applications, such as target tracking or exploration, robots will be required to change their global direction and speed. Here we derive equations to identify how parameters can act on the direction α and global speed v_{avg} of robots implementing the controller presented in Eq. 1. In particular, using symbols in Fig. 6 and knowing that robots with forward speed v will perform 2π during one beat, we can calculate

$$\alpha = h_{lim} + tan^{-1}\frac{a}{b} - \beta + \frac{3\pi}{2} \tag{6}$$

and

$$v_{avg} = \frac{\sqrt{a^2+b^2}}{t_1+t_2} \tag{7}$$

where

$$a = \frac{v}{\omega_1} \cdot sin(\beta) + \frac{v}{\omega_2} \cdot sin(\gamma) \tag{8}$$

$$b = \frac{v}{\omega_2} - \frac{v}{\omega_2} cos(\gamma) - (\frac{v}{\omega_1} - \frac{v}{\omega_1} cos(\beta)) \tag{9}$$

$$\beta = \omega_1 \cdot (t_1 + t_{lim}) \tag{10}$$

$$\gamma = \omega_2 \cdot (t_2 - t_{lim}) \tag{11}$$

$$t_{lim} = \frac{2\pi - |\omega_1| \cdot t_1 - |\omega_2| \cdot t_2}{|\omega_1| - |\omega_2|} \tag{12}$$

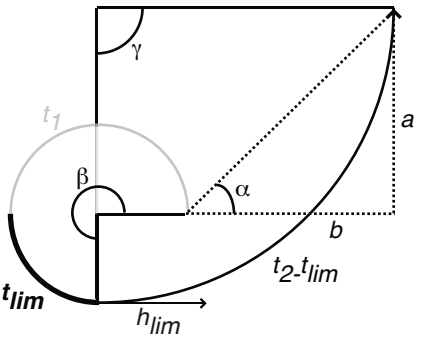


Fig. 6 Symbols used to determine the average advancement speed and direction of the robot trajectories.

While these equations allow to predict in what direction and at what speed each robot will move, it is challenging to set the parameters in order to achieve a desired command because the parameters (ω_1, ω_2, t_1 and t_2) can not be isolated analytically. The problem can however be simplified by only considering trajectories where

$$\beta = \frac{3}{2}\pi \tag{13}$$

$$\gamma = \frac{\pi}{2} \tag{14}$$

which can be achieved if

$$t_1 = \frac{\pi}{|\omega_1|} \tag{15}$$

$$t_2 = \frac{\pi}{2|\omega_1|} + \frac{\pi}{2|\omega_2|} \tag{16}$$

leading to trajectories where

$$\alpha = h_{lim} + \frac{\pi}{4} \tag{17}$$

$$v_{avg} = \frac{\sqrt{2}(\frac{v}{|\omega_2|} - \frac{v}{|\omega_1|})}{t_1 + t_2} \tag{18}$$

Thanks to Eq. 17 and 18, the parameters can easily be modified to modulate the global motion direction and speed of robot trajectories. In particular, the direction can be changed by modifying h_{lim}. Furthermore, increasing or decreasing the global speed of each robot can be done by increasing or decreasing the difference between ω_2 and ω_1 respectively within the boundaries set by Eq. 2. For negative turn rates we can use $\beta = -\frac{3}{2}\pi$, $\gamma = -\frac{\pi}{2}$ and $\alpha = h_{lim} + \pi - \frac{\pi}{4}$.

4 Experimental Setup

For the purpose of this experiment, we use up to five fixed-wing platforms that are light-weight (420 g, 80 cm) and safe (Fig. 7). Each robot is equipped with an autopilot for the control of altitude, airspeed and turn rate that provides an interface for receiving commands from a navigation controller. Embedded in the autopilot is a micro-controller that runs a minimalist control strategy based on input from only 3 sensors: one gyroscope and two pressure sensors [7].

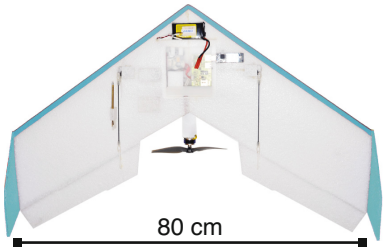


Fig. 7 Safe flying wing (450g, 80 cm) for outdoor experiments made out of soft material and with a back-mounted propeller. The robot is equipped with an autopilot, embedded Linux, WiFi dongle and GPS (only for logging purposes).

The controller presented in this paper is implemented on a Toradex Colibri PXA270 CPU board running Linux, connected to an off-the-shelf USB WiFi dongle. The output of this high-level computer, namely a desired turn rate, is sent as control command to the autopilot. Altitude is set to a different constant value for each robot between 50 m and 90 m and separated by 10 m. The airspeed is also constant at 12 m/s. In order to log flight trajectories, the robot is further equipped with a u-blox[1] LEA-5H GPS module.

For emitting and receiving the beat, robots use a Netgear[2] WNDA3100 dongle implementing the 802.11n standard and transmitting in the 5 GHz band. This is interesting with respect to transmissions in the 2.4 GHz band because it allows for less interference with the considerable number of devices currently used in this band. Dongles are configured for ad-hoc mode and have a communication range of nearly 500 m line-of-sight.

[1] http://www.u-blox.com
[2] http://netgear.com

5 Results

To validate the synchronization and steering of groups of robots we perform a set of in-flight experiments with up to five physical fixed-wing robots described in section 4. During these experiments, one of the robots sends beats by emitting heartbeat messages at an interval of 5 ms during the "on" phase and no messages during the "off" phase. This was done to increase the robustness of beat signals to communication failure.

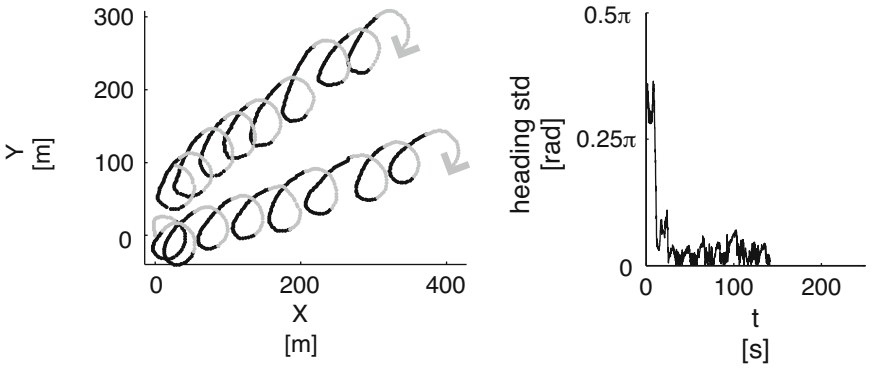


Fig. 8 Demonstration of synchronization on board two real flying robots in an outdoor experiment. Left: trajectories of the robots. Right: standard deviation on the robot headings.

The first experiment shown in Fig. 10 is aimed at demonstrating that two robots that start with different initial headings will synchronize over time. Parameters for this experiment are based on Eq. 1 with $h_{lim} = 5.4$ rad $\omega_1 = -0.7$ rad/s and $\omega_2 = -0.1$ rad/s with t_1 and t_2 set following Eq. 15 and 16 respectively. Notice how the standard deviation of robot headings rapidly goes down to nearly zero, thereby indicating synchronization.

Beyond synchronization, we aim at showing that a group of two robots can be steered and speed regulated. In particular, we propose three mission goals. In the first, robots are directed to go towards the North. h_{lim} is then changed, thereafter directing the group to the South (phase II). In the third phase the turn rate ω_2 is changed to slow down the global progression speed of the group. As a result, Fig. 9 shows how the speed and direction of the robots can be changed while remaining synchronized. Parameters for this experiment are given in Table 1. Notice that because of wind to the South of around 3.5m/s, the desired speed and direction of the group in not exact with respect to theoretical calculations. Good synchronization and group steering is however achieved.

Finally, in Fig. 10 we show that this method scales to five flying robots. For this experiment, we propose two mission goals. In the first, robots are directed to go

Table 1 Controller parameters used to achieve trajectories shown in Fig. 9

	h_{lim} [rad]	ω_1 [rad/s]	ω_2 [rad/s]
phase I	5.8	-0.7	-0.1
phase II	2.7	-0.7	-0.1
phase III	2.7	-0.7	-0.3

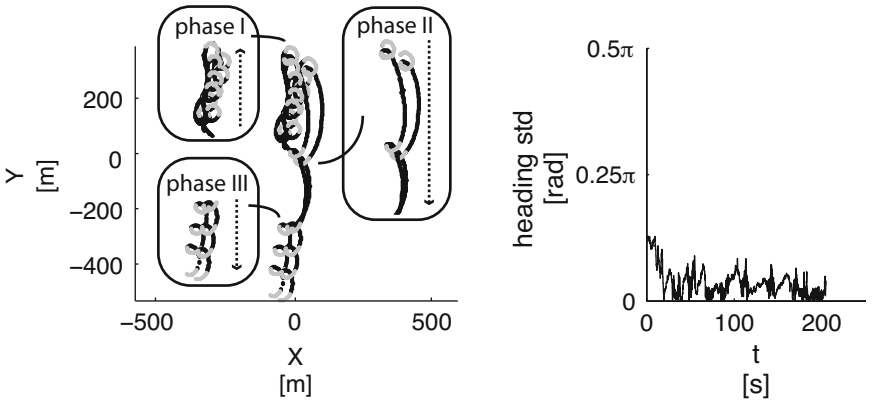


Fig. 9 Demonstration of synchronization and steering of two real flying robots in an outdoor experiment. Left: trajectories of the robots. Right: standard deviation on the robot headings. Three phases are shown here, in the phase I, the robot group is directed to the North against the wind. In phase II, robots are directed to turn around and proceed South. Phase III then shows how the robots can be slowed down. Notice that the robots remain synchronized throughout the experiment.

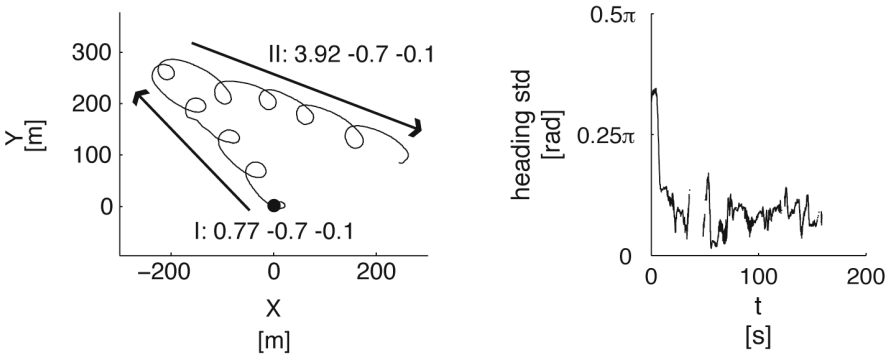


Fig. 10 Demonstration of synchronization and steering of five real flying robots in an outdoor experiment. Left: mean trajectory of the robots. Right: standard deviation on the robot headings. Two phases are shown here, in phase I, the robot swarm is directed to the West. In phase II, robots are directed to turn around and proceed East. Parameters represent h_{lim}, ω_1 and ω_2. Notice that the robots remain synchronized throughout the experiment which is why it is possible to plot such a mean trajectory.

towards the West. h_{lim} is then changed, thereafter directing the swarm to the East (phase II). Fig. 10 (right) shows the standard deviation on the heading of the robots which rapidly decreases over time as robots synchronize. The five robot trajectories are summarized by their mean which highly resembles the individual trajectories because all robots are synchronized. In this experiment wind of around 1 m/s to the North-East was present. Overall, robots are able to achieve good synchronization and steering. Because of wind and the dynamics of the robots, which prevent them from rapidly changing their turn rate, the actual direction performed by the swarm is slightly shifted with respect to the initial goal. A video showing the synchronization of five robots can be seen on our project webpage (http://lis.epfl.ch/smavs).

6 Discussion

While group steering with synchronization is a first step towards deploying flying robots in real-world applications, two main challenges persist, namely the difference in flight dynamics across platforms and the question of what application-oriented outer-loop control could be used to determine where to steer the robots.

Indeed, two robots implementing identical turn rate commands and speed commands will generally not perform identical trajectories due to sensor noise and hardware differences. The effect of turn rate bias on the synchronization and steering of the robots can be seen in Fig. 11 where we show the simulated trajectories resulting from turn rates of value $\omega_1 = [0.6, 0.65, 0.7]$ rad/s and $\omega_2 = 0.1$ for $t_1 = 4.488$ s and $t_2 = 17.9520$ s. However, robots that display different turn rates will still perform one revolution during one beat if they meet requirements described in Eq. 2 and 3. Therefore, the shift in heading among the robots is stable over time.

To achieve robot behaviors as similar as possible, some feedback would be required to calibrate the robot turn rates over time. This feedback could be based on comparing the robot headings at each beat with the desired heading and modifying the turn rate accordingly.

Finally, in scenarios with wind and without any position information, it becomes challenging to know where to steer the group of robots. For this purpose, one can implement an outer-loop responsible for issuing commands for the steering and speed regulation of the robots. This outer-loop can reactively increase or decrease the speed of the swarm and make it turn more or less based on sensory input from the robots. Reactive controllers that do not use position have been developed in the past to allow flying robots to remain leashed or track a base station on the ground [4] or avoid obstacles [13]. As an example, we consider in simulation a scenario where the swarm must remain within the communication range of a base station on the ground. Each time a robot loses its connection to the base station, it records its heading and broadcasts a new set of controller parameters to all robots that make them pursue a global direction opposite from its disconnection heading. In that manner it becomes the “leader” of the swarm. Results in Fig. 12 show that robots starting from different

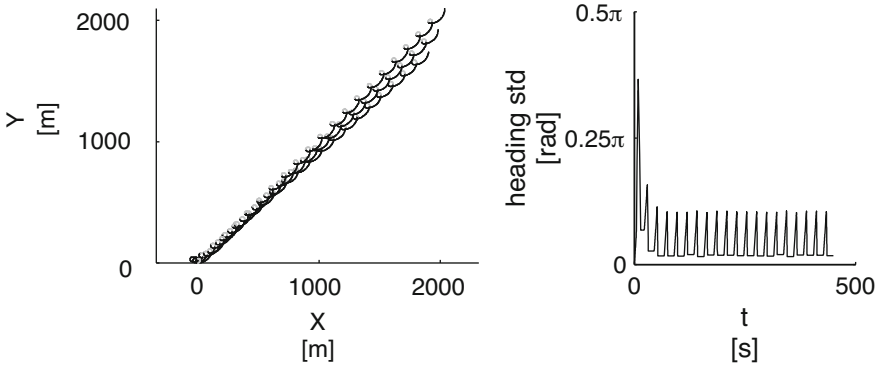


Fig. 11 Effect of turn rate bias in simulation on the robot trajectories (left) and heading (right) of robots with turn rates equal to $\omega_1 = [0.6, 0.65, 0.7]$ rad/s and $\omega_2 = 0.1$ for $t_1 = 4.488$ s and $t_2 = 17.9520$ s. Notice that while the robots implement different controllers, their headings still synchronize with a constant shift. The direction and advancement speed of the group is also slightly modified across robots.

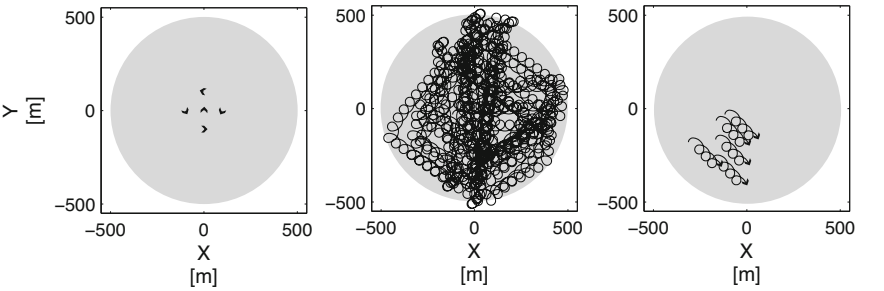


Fig. 12 Simulated swarms synchronize and move in groups while remaining connected to a base station on the ground. The grey area represents the communication range of the base station in (0,0). The figure to the left shows the starting position and heading of the robots, the center image shows all trajectories over a 30 min trial and the figure to the right shows the end of the trial and the synchronized heading of the robots.

headings are able to synchronize and move in groups while remaining connected to the base station.

7 Conclusion

Steering groups of robots while keeping them synchronized is an essential building block for the deployment of flying robots in real-world applications. However, challenges arise when controlling robots with tightly constrained motion dynamics, such

as fixed-wing platforms whose speed must remain within a certain limit. Rather than relying on complex controllers based on position, frequent sensing and communication, or memory, we proposed a minimal controller that modulates the turn rate of robots based on their current heading and a beat signal sent from a radio emitter on a ground station or a robot. This controller has the property of synchronizing the robot headings to the beats, regardless of their starting heading. Furthermore, it can easily be parametrized to steer a group of robots and change its global speed. Finally, we demonstrated synchronization and steering in reality using up to five fully autonomous fixed-wing flying robots.

To improve results in reality, two directions were suggested. The first consists in using feedback on heading to improve the accuracy of turn rate commands across robots. The second includes applying such mechanisms to real-world applications by adding an outer-loop responsible for emitting the higher-level steering commands. In the future, efforts should also be made to describe controllers in terms of synchronized oscillators [9, 12]. Such an endeavor would allow for stability proofs, more formal mathematical models and a large range of extensions based on different forms of synchronization states found in the literature.

Acknowledgements. This work is supported by armasuisse, competence sector Science + Technology for the Swiss Federal Department of Defense, Civil Protection and Sports. Sincere thanks to Steffen Wischmann for contributing his expertise in dynamical systems.

References

1. De Nardi, R., Holland, O., Woods, J., Clark, A.: SwarMAV: A swarm of miniature aerial vehicles. In: Proceedings of the 21st International UAV Systems Conference (2006)
2. Ghabcheloo, R., Pascoal, A., Silvestre, C., Kaminer, I.: Non-linear co-ordinated path following control of multiple wheeled robots with bidirectional communication constraints. International Journal of Adaptive Control and Signal Processing 21(2-3), 133–157 (2007)
3. Ghommam, J., Saad, M., Mnif, F.: Formation path following control of unicycle-type mobile robots. In: IEEE International Conference on Robotics and Automation, pp. 1966–1972. IEEE Press, Piscataway (2008)
4. Hauert, S., Leven, S., Zufferey, J.C., Floreano, D.: Communication-based leashing of real flying robots. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 15–20 (2010)
5. Hauert, S., Winkler, L., Zufferey, J.-C., Floreano, D.: Ant-based swarming with positionless micro air vehicles for communication relay. Swarm Intelligence 2(2-4), 167–188 (2008)
6. Hauert, S., Zufferey, J.C., Floreano, D.: Evolved swarming without positioning information: an application in aerial communication relay. Autonomous Robots 26(1), 21–32 (2009)
7. Leven, S., Zufferey, J.C., Floreano, D.: A minimalist control strategy for small UAVs. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2873–2878. IEEE Press, Piscataway (2009)
8. Li, Q., Jiang, Z.P.: Formation tracking control of unicycle teams with collision avoidance. In: Conference on Decision and Control, pp. 496–501. IEEE Press, Piscataway (2008)

9. Mirollo, R.E., Strogatz, S.H.: Synchronization of pulse-coupled biological oscillators. SIAM Journal on Applied Mathematics 50, 1645–1662 (1990)
10. Moshtagh, N., Jadbabaie, A., Daniilidis, K.: Vision-based distributed coordination and flocking of multi-agent systems. In: Proceedings of Robotics: Science and Systems (2005)
11. Strogatz, S.H.: Sync: The Emerging Science of Spontaneous Order. Hyperion Press (2003)
12. Strogatz, S.H., Stewart, I.: Coupled oscillators and biological synchronization. Scientific American 269(6), 102–109 (1993)
13. Zufferey, J.C., Beyeler, A., Floreano, D.: Autonomous flight at low altitude with vision-based collision avoidance and GPS-based path following. In: Proceedings of the IEEE International Conference on Robotics and Automation. IEEE Press, Piscataway (2010)
14. Zufferey, J.C., Hauert, S., Stirling, T., Leven, S., Roberts, J., Floreano, D.: Handbook of Collective Robotics, chap. Aerial collective systems. Pan Stanford Publishing (in press)

Part III
Coordination Algorithms and Formal Methods

Part III: Coordination Algorithms and Formal Methods

Lynne E. Parker

In distributed robot systems, coordination algorithms are used to control the motions and/or actions of the individual robots, such that the team as a whole achieves a globally coherent behavior. Typically, these coordination algorithms are themselves distributed, requiring each robot to act based only on a partial knowledge of the global system state. The key research challenge is designing the distributed control approach such that optimal, or near-optimal, overall system behavior is achieved. In the early days of distributed robot systems research, system developers would typically select an application, explore various local control strategies, and compare and contrast them to illustrate control techniques that work well in practice. However, many of these early techniques were ad hoc and empirical, and not based on formal methods.

More recent research has recognized this shortcoming, and now focuses on the development of analytical techniques that can synthesize distributed controllers that achieve the desired macro-level system behaviors. Mather, Braun, and Hsieh address this challenge by proposing a technique that first identifies robot-robot interactions at the macroscopic level; they then use this analysis to improve local robot control policies by filtering out spurious robot-robot interactions. Another top-down design approach is presented by Chen, Ding, Stefanescu, and Belta, who show how to automatically synthesize control and communication strategies for a robot team based on global specifications of the desired system-level behavior, stated using regular expressions. The resulting control strategies are formally proven to correctly achieve the desired global behavior. The work of Melo and Veloso takes a different approach to the synthesis of local decision policies by making use of the decentralized sparse-interaction Markov decision process. This technique allows agents to recognize states when interactions with other robots might occur, thus enabling them to choose better motions based on possible future inter-robot actions. Tsiotras and Castro synthesize local controllers by generalizing the standard consensus algorithm, applied to geometric pattern formation. Finally, Milutinović presents a centralized, rather than a distributed, approach to defining local robot motion control, in which each robot's motion is specified by a stochastic hybrid automaton model.

A common theme in most of the works cited above is that they first make use of formal methods to describe the desired macro-level behavior, and then show how to use this macro-level goal to synthesize individual robot controllers. Another use of

Lynne E. Parker
Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, TN USA
e-mail: parker@eecs.utk.edu

formal methods is to show how the individual goals of robot team members can be considered collectively, with the objective of maximizing the system's achievement of individual goals. Toward this end, game-theoretic techniques have been shown useful in a variety of distributed robot formulations. Dasgupta and Cheng make use of the game-theoretic technique called Weighted Voting Games to address the problem of multi-robot team formation control amidst obstacles. Taheri, Afshar, and Asadpour also make use of game-theoretic principles, building upon the Local Interaction Game diffusion model to investigate how a small number of agents can influence the global society's behavior through local interactions.

An interesting question in the design of distributed robot coordination mechanisms is the extent to which identical controllers can lead to diversity, specialization, or changes in robot behavior. Halász, Liang, Hsieh, and Lai study the emergence of specialization in robot swarms by making use of a distributed adaptation algorithm. They present a top-down analytical approach that defines the system equilibrium using waiting time parameters, and then present adaptive optimization strategies that converge to the optimal configurations that achieve system equilibrium. Temporal changes in system-level swarm behavior are addressed by Hoff, Wood, and Nagpal, who show how a swarm can change and improve its foraging behavior by switching between algorithms based on the environment in which the swarm finds itself.

Once the distributed controller is synthesized, most of the works cited thus far presume that individual robots execute their controller successfully. The typical presumption is that large swarms of interchangeable robots automatically result in robust and scalable swarm behavior. However, this presumption is challenged by Bjerknes and Winfield, who illustrate that overall swarm reliability quickly falls in the presence of worst-case, partially failed robots. They conclude that future large scale swarm systems must develop new approaches for achieving high levels of fault tolerance.

A complicating issue when designing distributed coordination algorithms is dealing with robot heterogeneity. In many distributed robot teams, robots are intentionally designed to be heterogeneous. Such robots have overlapping abilities to address the team's tasks, but will typically vary in the quality with which they are able to accomplish tasks. The coordination challenge is then one of task allocation, determining which tasks each robot should address in order to maximize the overall system utility. This task allocation issue has been studied since the early days of distributed robot systems research. Recent work addresses more complex allocation scenarios, such as the need to form multi-robot coalitions to address the same task. Hawley and Butler present a hierarchical market-based approach to task allocation that allows robots to form coalitions; this approach is illustrated in an exploration task. In other heterogeneous robot task allocation work, Walker and Wilson present an endocrine-based system that makes use of adaptive robot task sensitivity parameters to enable dynamic task reallocation.

Together, the research works in this part represent important new contributions to the challenge of coordination algorithms and formal method in distributed robot systems.

Distributed Filtering for Time-Delayed Deployment to Multiple Sites

T. William Mather, Christopher Braun, and M. Ani Hsieh

Abstract. We address the synthesis of distributed control policies for a homogeneous robot ensemble assigned to monitor multiple locations. Our approach uses an appropriate macroscopic description of the ensemble dynamics to first identify spurious behaviors exhibited by the ensemble as a result of robot-robot interactions that arise from operating within a shared environment. This macroscopic analysis is then used to design and improve the agent-level control policies and enhance the overall team performance. In particular, we consider the time-delayed task assignment problem where deterministic (or near deterministic) task execution times result in extraneous robot-robot interactions which degrades the ensemble efficiency. The main contribution is a novel approach towards synthesizing distributed filters to smooth out spurious interactions. We validate our distributed filter both in simulation and in actual robotic experiments.

1 Introduction

We address the dynamic allocation of a homogeneous ensemble of robots to a set of spatially distributed tasks, which is relevant for large scale environmental monitoring, surveillance, and automated warehouse distribution systems. In these applications, the ensemble must have the ability to autonomously distribute among the variouslocales/tasks and redistribute to ensure task completion and/or coverage that may be affected by robot failures or changes in the environment. Furthermore, we are interested in the design of distributed allocation strategies that can be implemented with little to no inter-agent wireless communication. This approach is relevant in situations where communication may be unreliable or non-existent, *e.g.*,

T. William Mather · Christopher Braun · M. Ani Hsieh
Drexel University, Philadelphia, PA 19104
e-mail: {twm32,cwb34,mhsieh1}@drexel.edu

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 299–312.
springerlink.com

underwater or underground environments, or when bandwidth must be preserved to enable transmission of critical data.

This is similar to the multi-task (MT) robots, single-robots (SR), time-extended assignment (TA) problem (Gerkey and Mataric, 2004). In the multi-robot domain, market-based approaches (Gerkey and Mataric, 2002; Vail and Veloso, 2003; Guerrero and Oliver, 2003; Dias, 2004; Lin and Zheng, 2005; Jones et al, 2006, 2007) have been successful and can be further improved when learning is incorporated (Dahl et al, 2006). Another approach is the formulation of the allocation strategy as a distributed constraint satisfaction problem, which requires the explicit modeling of the task requirements (Shen and Salemi, 2002). The main disadvantage of these methods is that they often scale poorly in terms of team size and number of tasks (Dias et al, 2006; Golfarelli et al, 1997). Additionally, in applications where inter-agent wireless communication may be unreliable or non-existent, it is often difficult to devise reliable strategies to ensure timely communication of the various local costs and utilities required by existing allocation approaches.

Recently, there has been increasing interest in the use of macroscopic continuous models to describe the ensemble dynamics of a robot swarm (Martinoli et al, 2004; Lerman et al, 2006; Hsieh et al, 2009). These continuous ensemble models are derived by representing the individual robot controllers as probabilistic finite state machines and approximating the dynamics of an ensemble of discrete Markov processes as a continuous-time Markov process (Martinoli et al, 2004; Lerman et al, 2006; Hsieh et al, 2008). These macroscopic models are used to analyze the effects of microscopic, or agent-level, transition probabilities on ensemble performance. Martinoli et al (2004) studied the impact of individual robot wait times on collaborative stick-pulling. The distribution of a swarm of robots without the use of inter-agent wireless communication was achieved by Lerman et al (2006) for a multi-robot foraging task by specifying the spatial distribution of the tasks within the workspace. Halasz et al (2007); Hsieh et al (2008) used the macroscopic models to synthesize stochastic agent-level control policies to enable the dynamic allocation of a team of robots to multiple locales in predefined proportions without explicit inter-agent wireless communication. Different from Lerman et al (2006), the desired allocation was achieved through appropriate selection of the individual robot transition rates. These results where then extended to account for navigation delays by Berman et al (2008).

In this work, we consider the multi-site allocation problem, first presented by Halasz et al (2007), subject to deterministic task execution times. Different from previous work, we present an approach to reduce the spurious behaviors exhibited by an ensemble of interacting agents. In other words, we identify and filter the "noise" generated from the intended or unintended interactions of the existing coordination strategy. This is achieved by employing an appropriate macroscopic model to analyze the ensemble dynamics and to determine the appropriate filtering technique. From this analysis, we develop a set of agent-level filters to remove the spurious behaviors that degrade the ensemble performance. Different from Goldberg and Matarić (1997); Rosenfeld et al (2004), we provide an analytical approach towards the analysis of the impact of inter-robot interference on team performance.

We consider the coordination problem from an estimation point-of-view and present a novel approach to characterize and control the sources of noise within an ensemble of interacting agents.

This paper is organized as follows: Section 2 presents the development of the macroscopic model for an ensemble of robots executing a collection of tasks with deterministic task execution times. Section 3 describes our analysis and distributed filter synthesis methodology. We present our simulation and experimental results in Section 4. We conclude with a brief discussion of our results in Section 5, and some thoughts for future work in Section 6.

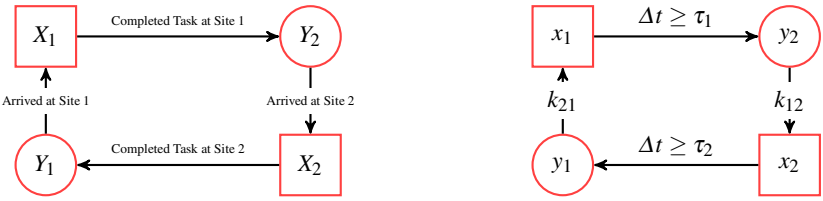
2 Problem Formulation

Consider the assignment of N robots to M distinct locales where robots must execute similar tasks at each locale. Let τ_i denote the amount of time it takes a robot to execute the task at site i. We consider the surveillance scenario where the ensemble is tasked to equally distribute across the M sites. This can be achieved by choosing $\tau_i = \tau$ for all i along a single cycle path. At each site, robots are programmed to spend τ amount of time monitoring the location. Once the task is completed, robots randomly choose the next adjacent site to visit based on a uniform distribution. Fig. 1(a) shows a finite state automaton representation of the individual robot controller for $M = 2$. Robots transition from one controller state to another based on the completed guard condition.

Given a set of M distinct locales, we model the interconnection topology of the sites using a directed graph, $\mathscr{G} = (\mathscr{V}, \mathscr{E})$. The set of vertices, $\mathscr{V}$, represent sites $1, \ldots, M$ and the set of directed edges, $\mathscr{E}$, represent the set of precedence constraints between sites. Two nodes $i, j \in \mathscr{V}$ are *adjacent* if an edge exists between sites i and j, and we represent this relation by the ordered pair, $(i, j) \in \mathscr{V} \times \mathscr{V}$ with the set $\mathscr{E} = \{(i, j) \in \mathscr{V} \times \mathscr{V} | (i, j)\}$. We assume $\mathscr{G}$ is a strongly connected graph, *i.e.*, a directed path exists for any $u, v \in \mathscr{V}$.

Let $X_i(t)$ and $Y_i(t)$ denote the number of robots at site $i \in \{1, \ldots, M\}$ executing task i and the number of robots that have met the guard condition at site i respectively. We define the system state as $\mathbf{y}(t) = [y_1(t), \ldots, y_M(t)]^T$ where $y_i(t) = n_i(t)/N$ denotes the fraction of the N robots that have met the guard condition at site i. Similarly, $x_i(t)$ denotes the fraction of the population at site i. The specification in terms of fractions rather than absolute numbers is to provide a team size invariant formulation and is practical for both scaling purposes and in situations where losses of robots to attrition and breakdown are common.

To model the variability in navigation times between sites, we assign a set of constant *transition rates*, $k_{ij} > 0$, to each edge in $\mathscr{G}$ where k_{ij} defines the transition probability per unit time for one robot that left site i to arrive at site j. Fig. 1(b) shows the graphical representation of the ensemble model where deterministic state transitions are modeled by guard conditions and stochastic transitions are modeled by the probabilistic rates k_{ij} for $M = 2$.



(a) Robot Controller (b) Ensemble Model

Fig. 1 (a) The robot controller. The robot changes controller states dependent on guard conditions. (b) The ensemble model.

In this formulation, the steady-state distribution of the population across the M sites is specified through the selection of the task execution times τ_i for $i = 1,\ldots,M$. Furthermore, all task controllers are equivalent, and a transition from one site to another implies the execution of the navigation controller by the individual robot. As such, we assume each robot has complete knowledge of $\mathscr{G}$, the ability to localize within the workspace, and is capable of navigating from one site to another and execute the local task. Finally, we leverage on classical frequency-domain analysis of linear systems to model the ensemble dynamics and to synthesize the individual robot controllers. As such, given a real-valued function $f(t)$ for all real numbers $t \geq 0$, we denote the Laplace transform of $f(t)$ as $F(s) = \mathscr{L}[f(t)]$.

2.1 Time-Delayed Ensemble Model

Hsieh et al (2008) and Berman et al (2008) have shown that the time evolution of the population fraction executing task i can be modeled as a continuous-time Markov process in the absence of task execution times, *i.e.*,

$$\frac{d}{dt}y_i(t) = \sum_{(j,i)\in\mathscr{E}} k_{ji}y_j(t) - \sum_{(i,j)\in\mathscr{E}} k_{ij}y_i(t). \tag{1}$$

To account for the task execution times, we reformulate the above linear model as a delayed differential equation given by

$$\frac{d}{dt}y_i(t) = \sum_{(j,i)\in\mathscr{E}} k_{ji}y_j(t-\tau_j) - \sum_{(i,j)\in\mathscr{E}} k_{ij}y_i(t). \tag{2}$$

We note that the ensemble state variables do not contain the $x_i(t)$ terms since the deterministic delays are accounted for by the τ_j's in (2).

Both (1) and (2) are macroscopic models of the ensemble activity. In Hsieh et al (2008); Berman et al (2009), the desired distribution across the M sites was achieved by using (1) to optimize the k_{ij} terms to meet certain ensemble performance metrics. In this work, we use (2) to model the dynamics of an ensemble of

non-communicating robots executing a multi-site surveillance task subject to uncertainty. In other words, we use (2) to characterize the "uncertainty model" for the ensemble of interacting, non-communicating robots and develop distributed filtering techniques to filter out the noise.

3 Methodology

3.1 Characterizing the Ensemble Noise Model

In situations where the task execution times are stochastic, Berman et al (2008) showed that the ensemble dynamics given by (2) can be approximated using an equivalent expanded linear system, which we refer to as the Multi-Pole approximation. The expanded linear system is obtained by introducing additional dummy transitions between states to approximate the effects of the stochastic delay times, *i.e.*, adding additional vertices and edges for every edge in $\mathscr{E}$ to account for the delay. When delay times are deterministic or near deterministic, the number of additional dummy transitions required to appropriately capture the macroscopic effects of these delays can be significant.

For deterministic time delays, a more appropriate approximation technique is the use of Padé approximants to approximate the delay Mather and Hsieh (2010). Consider the Laplace Transform of (2) given by

$$sY_i = -\sum_{(j,i)\in\mathscr{E}} k_{ij}Y_i + \sum_{(j,i)\in\mathscr{E}} k_{ji}e^{-s\tau_j}Y_j \text{ for } i = 1,\ldots,M. \tag{3}$$

In general, the Laplace transform converts a differential equation in the time domain into an algebraic equation in the frequency domain where the resulting equations are purely sums of polynomials of s, thus simplifying the analysis. However, the time delay introduces an exponential term which makes the rate equations transcendental. To retain the algebraic structure, a Padé approximation of the form

$$R(s) = \frac{1-\alpha_1 s+\ldots+(-1)^q\alpha_q s^q}{1+\alpha_1 s+\ldots+\alpha_q s^q} = \prod_{i=1}^{q}\frac{1-p_i s}{1+p_i s} \tag{4}$$

is employed for the exponential term in the frequency domain (Silva et al, 2004). Here q denotes the order of the approximation.

In practice, the effect of the delay is to push the phase response of the system without altering the magnitude, and thus the phase error becomes worse as frequency increases. In the frequency domain, the time delay is modeled as an exponential variable as shown in (3). This is equivalent to an exponential delay in the phase response of the output signal. As the frequency increases, the system delays the output signal by more and more periods.

3.1.1 Example Problem

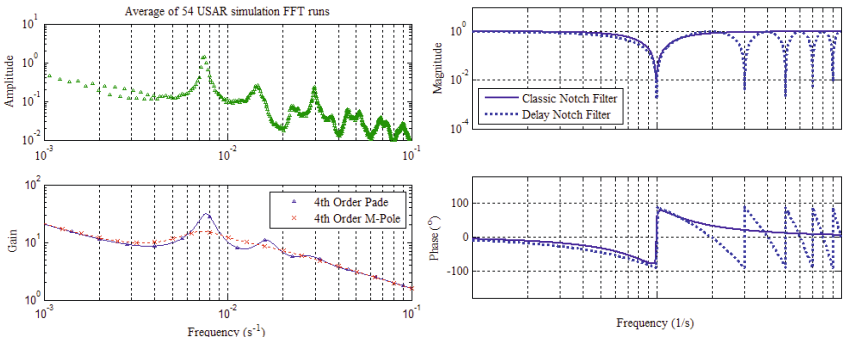
Consider the deployment of an ensemble of 10 robots moving in the plane to 2 distinct locations/sites. Initially, robots are randomly assigned to each of the two sites. Once a robot reaches the targeted site, it circles the site in a clockwise direction. The task execution time, $\tau_i = \tau$, is chosen to reflect the amount of time it takes the robot to circle the site twice. After accomplishing its task, the robot moves to the next site and performs the same task. Robots navigate from one site to another using a potential field controller. The variability in each robot's site-to-site navigation times depends on the amount of traffic on the road, which is affected by the number of collision avoidance maneuvers each robot must execute. Collision avoidance is achieved through a combination of gyroscopic forces and potential functions, (Chang et al, 2003; Hsieh et al, 2007). The transition rates and initial conditions for the system are summarized in Table 1.

Table 1 System transition parameters obtained from running agent-based simulations

State	Initial Condition	Transition Rate	τ Delay	Description
x_1	0	-	$\tau_1 = 50$	Monitoring Site 1
x_2	0	-	$\tau_2 = 50$	Monitoring Site 2
y_1	5	$k_{1,2} = \frac{1}{16.38}$	-	Traveling from 2 to 1
y_2	5	$k_{2,1} = \frac{1}{20.79}$	-	Traveling from 1 to 2

Fig. 2(a) shows the Fast Fourier Transform (FFT) of the average output of 54 agent-based simulations performed in USARSim (USARSim, 2007). The frequency response of the agent-based simulations was obtained by logging the population fractions at each site over time and applying the FFT to these variables for each run. The FFT results were then averaged over all 54 runs. The agent-based system exhibits a maximum gain at approximately 7.5 mHz and while both the Padé and Multi-Pole macroscopic models exhibit peaks at approximately the same frequency. However, the Padé model shows larger gain.

These spurious frequency components in the surveillance application manifest themselves as oscillations in the ensemble distribution in the time domain. This suggests that the robots are clustering together as they travel from one site to another. Consider the extreme case of a single robot, where the robot always travels in a "pack". In this case, the frequency content will be large because there is no traffic and all states will be 1 to 0 square waves. For the single robot there is no adverse effect of this frequency peaking. However, for the team of robots traveling together, collision avoidance becomes a significant concern because the local traffic is always high. This leads to degraded performance as the average transit time between sites will increase due to these traffic concerns. If we only consider the steady-state behavior of the system, one would expect the majority of the N robots to be



(a) Ensemble Response (b) Notch Filter Response

Fig. 2 (a) Top: Average of the FFT of the population fraction at building 2 obtained from 54 micro-discrete simulations. Bottom: Magnitude portion of the Bode plots relating to the number of Robots at building 2. For the 4^{th} order Padé, and a 4^{th} order Multi-Pole macro-continuous systems. (b) Frequency response of the classical 2^{nd} order notch filter and the delayed notch filter given by $H_\tau(s) = \frac{1}{2}(1+e^{-s\omega_\tau})$.

executing their surveillance tasks with only a small fraction of them traveling between sites. However, the presence of the unwanted frequency components can result in a significant imbalance between robots at sites and those traveling between them. The Padé approximated macro-continuous model has the ability to better predict the spurious frequency component that is present in the agent-based simulations and can provide insight into the synthesis of agent-level controllers to filter out these spurious frequencies.

As a final note, it is difficult, in general, to directly compare the macroscopic results with the microscopic results. This is because the FFT of the system states only considers the outputs of the system. The magnitude portion of the Bode plots, on the other hand, gives the response of the ratio of the output to input of the system for all frequencies. In other words, the macroscopic frequency response is based on a unity gain input at all frequencies. The difference between the two plots is dependent on the form of the noise input to the system and is related by the shape of the frequency spectrum of the noise input to the system.

3.2 Controller Synthesis

In general, delayed systems of the form (2) will exhibit a strong peak at a particular frequency, ω_p, which depends on both the transition rates and the task service times. When the deterministic time delays are significantly larger than the mean stochastic travel times, the loop time can be approximated by $T_{LOOP} = \frac{1}{k_{1,2}} + \frac{1}{k_{2,1}} + \tau_1 + \tau_2$.

The actual loop frequency of the system is given by the phase of the open loop gain. When the open loop phase hits 2π radians, the system produces positive feedback. The frequency where this pure positive feedback occurs is given by,

$$\omega_p = \left\{ \omega \quad \middle| \quad \measuredangle \left[\frac{k_{1,2}k_{2,1}e^{-j\omega(\tau_1+\tau_2)}}{(j\omega+k_{1,2})(j\omega+k_{2,1})} \right] = -2\pi \right\},$$

and corresponds to robots bunching together as they travel from one site to another. This causes the system to oscillate as groups of robots all change state at roughly equivalent times.

3.2.1 Ensemble Notch Filter

To smooth the response of the system, a common approach is to implement a notch filter to get rid of the spurious behavior. A notch selectively filters out a specific frequency while leaving other frequency components unchanged, effectively reducing the gain of the single spurious frequency component. A typical 2^{nd} order notch controller has the transfer function $H_1(s)$ given by

$$H_1(s) = \frac{s^2 + 2\zeta_1\omega_N + \omega_N^2}{s^2 + 2\zeta_2\omega_N + \omega_N^2}$$

where ω_N, ζ_1, and ζ_2 set the location and magnitude of the notch.

While applying control on the model of the macro-continuous system is straightforward, it is not clear how such a controller can be implemented at the individual robot level. Careful inspection of the closed-loop time domain equations suggest implementation of the filter will require individual robots to estimate the higher order derivatives of the populations at the various sites. In the following section, we propose an approximate solution, where the spurious frequency response can be removed without extra knowledge of the system states by the individual robots.

3.2.2 Agent-Level Implementation

Given the existing robot controller, there are two ways to modify the agent-level control policy without requiring any inter-agent communication. The addition of either a stochastic or a deterministic time delay after a task execution. By adding an extra delay path in the robot controllers, a distributed ensemble notch filter can be constructed without requiring explicit knowledge of the higher order derivatives of the macroscopic states by the individual robots. This can be done by splitting the team into two sub-teams where one team purposely enacts the additional delay at a given site for each cycle path. This approach can eliminate a frequency by adding a signal to a copy of itself, 180^o degrees out of phase. The transfer function for the proposed notch filter is $H_\tau(s) = \frac{1}{2}(1 + e^{-s\omega_\tau})$.

The frequency response plot for the notch filter and its 2^{nd} order classical controller are shown in Figure 2(b). This delay-constructed filter cancels every odd harmonic of the primary notched frequency. With the added notch filter, the new closed loop macroscopic equations of the notched system dynamics are,

$$\begin{aligned} \dot{y}_1(t) &= -k_{1,2}\, y_1(t) + \tfrac{1}{2} k_{2,1}\, y_2(t-\tau_2) + \tfrac{1}{2} k_{2,1}\, y_2(t-\tau_2-\tau_{NOTCH}) \\ \dot{y}_2(t) &= -k_{2,1}\, y_2(t) + k_{1,2}\, y_1(t-\tau_1) \end{aligned} \tag{5}$$

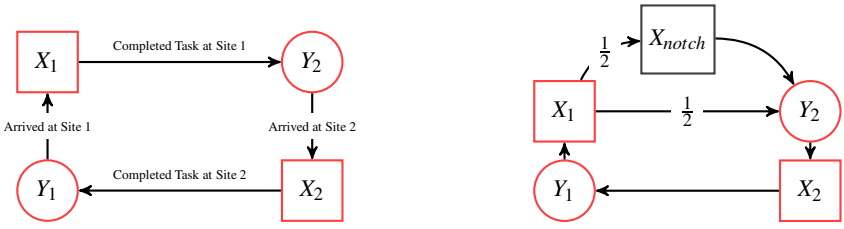
The addition of a single notch filter will suppress a single spurious population behavior. If the task precedence graph has multiple cycles with spurious loops, multiple notches are required to eliminate spurious behavior.

In general, the introduction of a delay into a system with feedback can be dangerous since it can lead to enough phase lag to turn negative feedback into positive feedback resulting in unstable oscillations. However, the models discussed here fall into a family of systems that are stable independent of delay (Chen and Latchman, 1995). The intuition behind this inherent stability is that the loop gain of the linear system, $G(s)$, is never greater than unity, which results in an undefined phase margin. Thus, no amount of extra phase delay can drive the system unstable.

4 Results

4.1 Simulation Results

An agent-based simulation for an ensemble of 10 SRV-1 robots was performed in USARSim (USARSim, 2007). The SRV-1 are differential-drive robots equipped with an embedded processor, color camera, and 802.11 wireless capability. The ensemble was tasked to survey two separate sites that are represented by rectangular blocks in the workspace. We assume the robots have a map of the environment and navigate from one site to another using potential functions. Once a robot reaches the



(a) Original Controller (b) Notched Controller

Fig. 3 (a) Original robot controller. (b) Agent-level implementation of the ensemble notch filter. Robots monitor site 1 then decide with probability $^1/_2$ whether to travel to site 2 or delay for τ_{notch}.

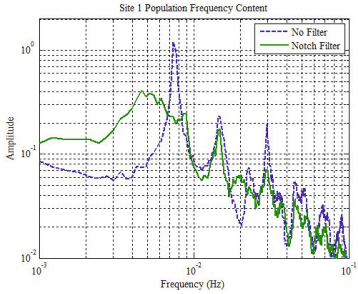


Fig. 4 Frequency response results for the simulated system. This plot shows the average response of 50 simulations of the unfiltered (no notch) system and 50 simulations of the notched system described by (5). The notched system shows a depression in the frequency response at the active notching point.

target site, it circles the site in a clockwise direction until the task execution time, τ_i, has been reached.

We implemented the distributed notch filter on the system with parameters given in Table 1. After completing the timed surveillance at Site 1, each robot randomly chooses with probability $\frac{1}{2}$ to either move to the next site or stay and continue monitoring the site for half of the loop time, the time to complete one cycle in the system. Based on the values in Table 1, this extra wait time, τ_{notch}, was set to 68 seconds.

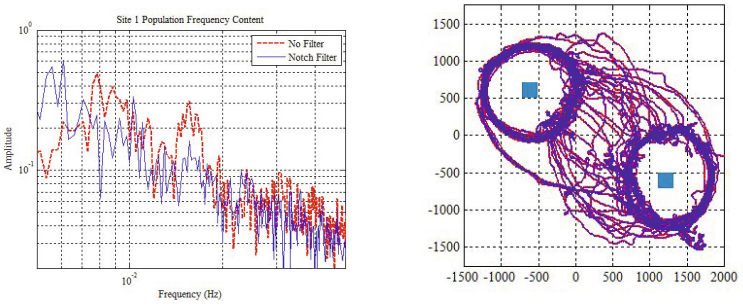
We ran 50 micro-discrete simulations for each ensemble of robots executing the original controllers and the distributed implementation of the notched filter. The frequency response of these results is shown in Fig. 4. Our results show that the distributed notch filter suppressed the undesired frequency component by 70%. While there is limited frequency peaking at the lower frequencies (see Fig. 4) because of the larger loop time introduced by the notch filter, the resulting spurious frequency is much lower in magnitude than the original peak.

4.2 Experimental Results

We also implemented our original and notched robot controllers on our multi-robot testbed. The testbed consisted of seven Surveyor SRV-1 robots operating within a 4.8x5.4 meter workspace. The robots were tasked to navigate towards and surround two separate beacons, similar to our simulations. Overhead localization for the team

Table 2 Variance in the site populations compared to those predicted by the macro-discrete simulation

Simulation Type	$\mathbf{Var}(X_1)$	$\mathbf{Var}(X_2)$
Macro-Discrete (unfiltered)	2.31	2.26
Micro-Discrete (unfiltered)	3.04	2.93
Macro-Discrete (notched)	2.13	2.54
Micro-Discrete (notched)	2.16	2.74



(a) (b)

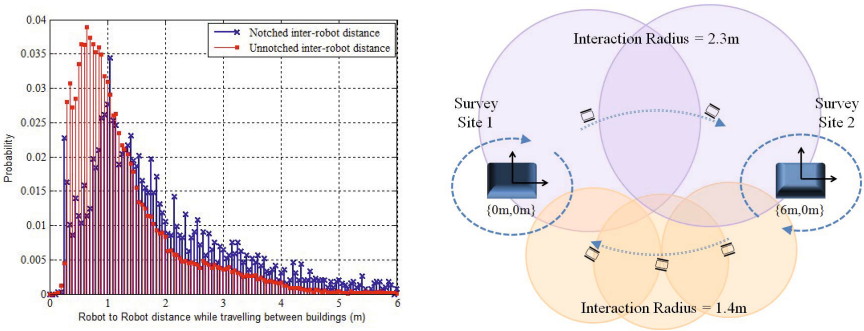
Fig. 5 (a) Frequency response of the initial experimental trials. The plots shows the frequency content of 3 averaged response for the unfiltered and notched system. The peak, though small, is properly located according to the transition times. The high peak at 6.1mHz in the notched response is due to the round trip time if all agents went along the delay route that includes the τ_{NOTCH} delay. (b) A single robot trajectory for the two site surveillance experiment. Sources of stochasticity include collision avoidance, different exit locations, and errors in the localization.

was provided using four cameras. Each experiment ran for roughly 45 to 50 minutes with the robots executing 750 state transitions. The frequency response for both the original and the notched system is shown in Fig. 5(a). The path of a single robot in one experimental run is shown in Fig. 5(b).

5 Discussion

To verify that the spurious frequency component led to traffic congestion as robots move from one site to another, we analyzed the inter-robot distance between the robots traveling between sites for both the original and the notched system. Fig. 6(a) shows the distribution of the pairwise distances between traveling robots. The average distance from a robot to its neighbor in the original system is 1.36 meters with a standard deviation of 1.05 meters. The notch filter increased the mean and standard deviation to 2.29 and 1.41 respectively. Fig. 6(b) shows that this leads to significantly less congestion.

The observed congestion also manifests itself in the on site distribution statistics because robots tend to travel in "pack". This results in higher amplitude oscillations in the population variables at each site which increases the variance of the population fraction at each site. In the extreme case, when all robots travel as a single unit, this leads to a two peak distribution. To demonstrate this, we ran 20 marco-discrete simulations based on Gillespie's Direct Method (Gillespie, 1976). Due to the high level of abstraction of these simulations, inter-robot collision avoidance is completed encapsulated within the stochastic transition rates and provides theoretical values for the variance at each site. These values and the variance observed



(a) (b)

Fig. 6 Proximity distribution of the filtered and unfiltered behavior. The notched proximity distribution has a significantly higher mean than the unfiltered system. The robots in the unfiltered system all begin to operate in sync with each other. Thus causing them to form lines of traveling robots which looks like oscillation in the population variables.

in our micro-discrete simulations are summarized in Table 2. We note that the site variance of the micro-discrete simulations are much larger for the unfiltered system. The notched system shows closer agreement in site variance to the theoretical value as shown in Fig. 6.

6 Conclusions and Outlook

In this work, we presented a method for synthesizing distributed notch filters through the analysis of the macroscopic ensemble models. The macroscopic analysis allowed for the identification of the spurious behavior exhibited by the ensemble, which resulted from excessive robot-robot interactions brought on by robots randomly synchronizing their task transition times. The macroscopic analysis was then used to synthesize an appropriate distributed filtering strategy that could be implemented without requiring any inter-robot wireless communication nor estimation of population variables. The effectiveness of the distributed filtering strategy was shown in both simulations and physical robotic experiments.

An immediate direction for this work is a formal analysis of the stability properties of our filtering strategy. We are also interested in extending this framework to allow for quantitative analysis of more complex coordination strategies, which may lead to the synthesis of further distributed filters for ensembles. Finally, we would like to extend the existing macroscopic models to enable the encoding of higher fidelity spatial information for the ensemble, as to provide guarantees on coverage.

Acknowledgements. We gratefully acknowledge the support of the South Korean Ministry of Economy CIRT, and of NSF grant DGE-0947936.

References

Berman, S., Halasz, A., Hsieh, M.A., Kumar, V.: Navigation-based optimization of stochastic deployment strategies for a robot swarm to multiple sites. In: Proc. of the 47th IEEE Conference on Decision and Control, Cancun, Mexico (2008)

Berman, S., Halasz, A., Hsieh, M.A., Kumar, V.: Optimized stochastic policies for task allocation in swarms of robots. IEEE Transactions on Robotics 25(4) (2009)

Chang, D.E., Shadden, S., Marsden, J.E., Olfati-Saber, R.: Collision avoidance for multiple agent systems. In: Proc. IEEE Conference on Decision and Control, Maui, Hawaii (2003)

Chen, J., Latchman, H.: Frequency sweeping tests for stability independent of delay. IEEE Transactions on Automatic Control 40(9), 1640–1645 (1995), doi:10.1109/9.412637

Dahl, T.S., Matarič, M.J., Sukhatme, G.S.: A machine learning method for improving task allocation in distributed multi-robot transportation. In: Braha, D., Minai, A., Bar-Yam, Y. (eds.) Understanding Complex Systems: Science Meets Technology, pp. 307–337. Springer, Berlin (2006)

Dias, M.B.: Traderbots: A new paradigm for robust and efficient multirobot coordination in dynamic environments. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA (2004)

Dias, M.B., Zlot, R.M., Kalra, N., Stentz, A.T.: Market-based multirobot coordination: a survey and analysis. Proceedings of the IEEE 94(7), 1257–1270 (2006)

Gerkey, B.P., Mataric, M.J.: Sold!: Auction methods for multi-robot control. IEEE Transactions on Robotics & Automation 18(5), 758–768 (2002)

Gerkey, B.P., Mataric, M.J.: A formal framework for the study of task allocation in multi-robot systems. IJRR 23(9), 939–954 (2004)

Gillespie, D.: A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. Journal of Computational Physics 22(4), 403–434 (1976)

Goldberg, D., Matarić, M.J.: Interference as a tool for designing and evaluating multi-robot controllers. In: Proceedings, AAAI 1997, pp. 637–642. AAAI Press (1997)

Golfarelli, M., Maio, D., Rizzi, S.: Multi-agent path planning based on task-swap negotiation. In: Proceedings of the 16th UK Planning and Scheduling SIG Workshop, PlanSIG, Durham, England (1997)

Guerrero, J., Oliver, G.: Multi-robot task allocation strategies using auction-like mechanisms. In: Artificial Research and Development, Frontiers in Artificial Intelligence and Applications, vol. 100, pp. 111–122. IOS Press (2003)

Halasz, A., Hsieh, M.A., Berman, S., Kumar, V.: Dynamic redistribution of a swarm of robots among multiple sites. In: Proceedings of the Conference on Intelligent Robot Systems (IROS 2007), San Diego, CA, pp. 2320–2325 (2007)

Hsieh, M.A., Loizou, S., Kumar, V.: Stabilization of multiple robots on stable orbits via local sensing. In: Proc. of ICRA 2007, Rome, Italy (2007)

Hsieh, M.A., Halasz, A., Berman, S., Kumar, V.: Biologically inspired redistribution of a swarm of robots among multiple sites. Swarm Intelligence (2008)

Hsieh, M.A., Halasz, A., Cubuk, E.D., Schoenholz, S., Martinoli, A.: Specialization as an optimal strategy under varying external conditions. Accepted the 2007 International Conference on Robotics and Automation (ICRA 2007), Kobe, Japan (2009)

Jones, E.G., Browning, B., Dias, M.B., Argall, B., Veloso, M., Stentz, A.T.: Dynamically formed heterogeneous robot teams performing tightly-coordinated tasks. In: Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA 2006), pp. 570–575. IEEE, Los Alamitos (2006)

Jones, E.G., Dias, M.B., Stentz, A.: Learning-enhanced market-based task allocation for oversubscribed domains. In: Proceedings of the Conference on Intelligent Robot Systems (IROS 2007), pp. 2308–2313. IEEE, Los Alamitos (2007)

Lerman, K., Jones, C., Galstyan, A., Matarić, M.J.: Analysis of dynamic task allocation in multi-robot systems. International Journal of Robotics Research (2006)

Lin, L., Zheng, Z.: Combinatorial bids based multi-robot task allocation method. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA 2005), pp. 1145–1150. IEEE, Los Alamitos (2005)

Martinoli, A., Easton, K., Agassounon, W.: Modeling of swarm robotic systems: a case study in collaborative distributed manipulation. International Journal of Robotics Research: Special Issue on Experimental Robotics 23(4-5), 415–436 (2004)

Mather, T.W., Hsieh, M.A.: Analysis of stochastic deployment policies with time delays for robot ensembles. Submitted to IJRR Special Issue: Stochasticity in Robotics and Biological Systems (2010)

Rosenfeld, A., Kaminka, G.A., Kraus, S.: Adaptive robot coordination using interference metrics. In: Proc. of the 16th Eureopean Conference on Artificial Intelligence (ECAI 2004), Valencia, Spain, pp. 910–916 (2004)

Shen, W.M., Salemi, B.: Towards distributed and dynamic task reallocation. In: Gini, M., Shen, W.M., Torras, C., Yuasa, H. (eds.) Intelligent Autonomous Systems 7, pp. 570–575. IOS Press International, Marina del Rey (2002)

Silva, G.J., Datta, A., Bhattacharyya, S.P.: PID Controllers for Time Delay Systems. Birkhäuser, Boston (2004)

USARSim, Unified system for automation and robot simulation (2007), `http://usarsim.sourceforge.net`

Vail, D., Veloso, M.: Multi-robot dynamic role assignment and coordination through shared potential fields. In: Schultz, A., Parker, L., Schneider, F. (eds.) Multi-Robot Systems, pp. 87–98. Kluwer (2003)

A Formal Approach to Deployment of Robotic Teams in an Urban-Like Environment

Yushan Chen, Xu Chu Ding, Alin Stefanescu, and Calin Belta

Abstract. We present a computational framework for automatic synthesis of control and communication strategies for a robotic team from task specifications given as regular expressions about servicing requests in an environment. Our approach is based on two main ideas. First, we extend recent results from formal synthesis of distributed systems to check for the distributability of the task specification and to generate local specifications, while accounting for the service and communication capabilities of the robots. Second, by using a technique inspired from LTL model checking, we generate individual control and communication strategies. We illustrate the method with experimental results in our Robotic Urban-Like Environment.

1 Introduction

The goal in robot motion planning and control is to be able to specify a motion task in a rich, high level language and have the robot(s) automatically convert this specification into a set of low level primitives, such as feedback controllers and communication protocols, to accomplish the task [13, 5, 14]. In most of the existing works, the motion planning problem is simply specified as "go from A to B while avoiding obstacles" [13]. However, there are situations in which this is not enough to capture the nature of the task. Consider, for example, the miniature Robotic Urban-Like Environment (RULE) shown in Fig. 1, where a robot might be required to "Visit Road R_1 or Road R_2 without crossing Intersection I_3, and then park in an available parking space," while at same time obeying the traffic rules. Such a "rich" specification cannot be trivially converted to a sequence of "go from A to B" primitives.

Yushan Chen · Xu Chu Ding · Calin Belta
Boston University, Boston, MA, US
e-mail: {yushanc,xcding,cbelta}@bu.edu

Alin Stefanescu
University of Pitesti, 110040 Pitesti, Romania
e-mail: alin.stefanescu@upit.ro

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 313–327.
springerlink.com

When several robots are available, the problem becomes even more interesting and challenging. Assume that several service requests occur at different locations in the city, and they need to be serviced subject to some temporal and logical constraints. Some of these requests can be serviced by one (possibly specific) robot, while others require the collaboration of two or more (possibly specific) robots. For example, assume that the task is to assemble a piece of machinery in location P_1 or P_2 from two components that can be found at P_3 and P_4. The assembly requires the cooperation of two robots, and the collection of the components needs to be performed in parallel. Can we generate provably-correct individual control and communication strategies from such rich, global specifications? This is the problem that we address in this paper.

It has been advocated as far back as [1] and more recently in [15, 8, 24] that temporal logics, such as Linear Temporal Logic (LTL) and Computation Tree Logic (CTL) [6], can be used as "rich" specification languages in mobile robotics. All of the above works suggest that the corresponding formal verification (model checking) algorithms can be adapted for motion planning and controller synthesis from such specifications. Some related works show that such techniques can be extended to multi-agent systems through the use of parallel composition [18, 12] or reactive games [9]. However, such bottom-up approaches are expensive and can lead to state-space explosion even for relatively simple problems. As a result, one of the main challenges in the area of motion planning and control of distributed teams based on formal verification is to create provably-correct, top-down approaches in which a global, "rich" specification can be decomposed into local (individual) specifications, which can then be used to automatically synthesize robot control and communication strategies. In such a framework, the construction of the parallel composition of the individual motions is not necessary, and therefore the state-space explosion problem is avoided.

In this paper, we draw inspiration from the area of distributed formal synthesis [17] to develop such a top-down approach. We consider a team of robots that can move among the regions of a partitioned environment, and which have known capabilities of servicing a set of requests that can occur in the regions of the partition. Some of these requests can be serviced by a robot individually, while some require the cooperation of groups of robots. We present an algorithm that allows for the fully automatic synthesis of robot control and communication strategies from a task specification given as a regular expression over the set of requests. For simplicity of presentation, we model the environment as a graph and the robots as agents that

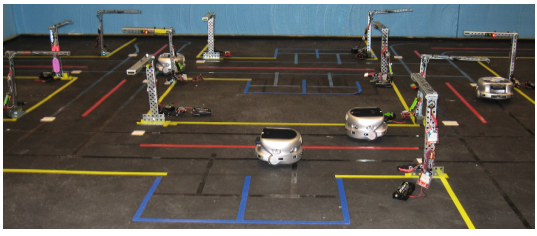


Fig. 1 Robotic Urban-Like Environment (RULE). Khepera III car-like robots move autonomously on streets while staying in their lanes, obeying traffic rules, and avoiding collisions.

can move between adjacent vertices and can communicate only when at particular vertices. This framework is quite general and can be used in conjunction with cell decomposition motion planning techniques [5]. In particular, by using feedback controllers for facet reachability in polytopes [10, 2], this scenario can be extended to robots with continuous dynamics moving in environments with polytopic partitions.

The contribution of this work is threefold. First, we develop a top-down computational framework for automatic deployment of mobile agents from global specifications given as regular expressions over environmental requests. This is a significant extension of our recent work [3] by enlarging the class of specifications for which a solution exists. Second, we provide a relaxation to the standard problem of distributed synthesis modulo synchronous products and language equivalence [17]. Specifically, we show how a satisfying distributed execution can be found when the global specification is only a traced-closed language, rather than a product language. This extends our previous work [22], in which we provided two heuristics for the case of asynchronous automata. Third, we implement and illustrate the computational framework in our Khepera-based Robotic Urban-Like Environment (Fig. 1). In this experimental setup, the robots can be automatically deployed from specifications given as regular expressions over requests occurring at regions in the city.

2 Preliminaries

Throughout this paper, we assume that the reader is familiar with automata theory [11,20]. In this section, we merely review some concepts and introduce the notation.

For a set Σ, we use $|\Sigma|$ and 2^{Σ} to denote its cardinality and power set, respectively. A collection of subsets $\Delta = \{\Sigma_i \subseteq \Sigma, i \in I\}$ is called a *distribution* over Σ if $\bigcup_{i\in I} \Sigma_i = \Sigma$, where I is an index set. A word is a sequence of symbols from Σ. We denote Σ^* as the set of all finite words over Σ. A *language* is a set of words.

Definition 1. A finite state automaton (FSA) is a tuple $A = (Q, q_0, \Sigma, \rightarrow_A, F)$, where Q is the set of states, $q_0 \in Q$ is the initial state, Σ is the set (alphabet) of actions, $\rightarrow_A \in Q \times \Sigma \times Q$ is the transition relation, and $F \subseteq Q$ is the set of accepting states. We also write $q \xrightarrow{\sigma}_A q'$ to denote $(q, \sigma, q') \in \rightarrow_A$.

We denote $\mathscr{L}(A)$ as the language accepted by an FSA A. The language of an FSA is called a *regular language*, which can be concisely represented by a *regular expression* (RE). Given an RE, an FSA accepting all and only the words satisfying the RE can be constructed by using an off-the-shelf tool, such as JFLAP [19].

Definition 2. The synchronous product of n FSAs $A_i = (Q_i, q_{0_i}, \Sigma_i, \rightarrow_{A_i}, F_i)$, denoted by $\|_{i=1}^{n} A_i$, is an FSA $A = (Q, q_0, \Sigma, \rightarrow_A, F)$, where $Q = Q_1 \times Q_2 \times \ldots \times Q_n$, $q_0 = (q_{0_1}, q_{0_2}, \ldots, q_{0_n})$, $\Sigma = \cup_{i=1}^{n} \Sigma_i$, and $F = F_1 \times F_2 \times \ldots \times F_n$. The transition relation $\rightarrow_A \subseteq Q \times \Sigma \times Q$ is defined by $q \xrightarrow{\sigma}_A q'$ iff $\forall\, i \in I_\sigma : q[i] \xrightarrow{\sigma}_{A_i} q'[i]$ and $\forall\, i \notin I_\sigma : q[i] = q'[i]$, where $q[i]$ denotes the ith component of q and $I_\sigma = \{i \in \{1, \ldots, n\} \mid \sigma \in \Sigma_i\}$.

For convenience, in the particular case when $\Sigma_1 = \Sigma_2 = \Sigma$, we use $A_1 \times A_2$ to denote $\|_{i=1}^{2} A_i$. Moreover, $\mathscr{L}(A \times B) = \mathscr{L}(A) \cap \mathscr{L}(B)$ ([20]). An FSA $\neg A$ is defined as an FSA that accepts the language $\overline{\mathscr{L}(A)}$ where $\overline{\mathscr{L}(A)} := \Sigma^* \backslash \mathscr{L}(A)$.

For a word $w \in \Sigma^*$ and a subset $S \subseteq \Sigma$, we denote by $w \restriction_S$ the *projection* of w onto S, which is obtained by erasing all actions σ in w that do not belong to S. For a language $L \subseteq \Sigma^*$ and a subset $S \subseteq \Sigma$, we denote by $L \restriction_S$ the projection of L onto S, which is given by $L \restriction_S := \{w \restriction_S \mid w \in L\}$. Starting from the observation that the projection of a regular language is a regular language, the projection of an FSA A on a subset $S \subseteq \Sigma$ is another FSA (denoted by $A \restriction_S$) accepting the language $\mathscr{L}(A) \restriction_S$, through ε-closure, determinization and minimization ([21]).

Definition 3. Given a distribution Δ of Σ, the product of a set of languages L_i over Σ_i is denoted by $\|_{i \in I} L_i$ and defined as $\|_{i \in I} L_i := \{w \in \Sigma^* \mid w \restriction_{\Sigma_i} \in L_i \text{ for all } i \in I\}$. A *product language* over a distribution Δ of Σ is a language L such that $L = \|_{i \in I} L_i$, where $L_i = L \restriction_{\Sigma_i}$ for all $i \in I$.

Definition 4. Given a distribution Δ of Σ and $w, w' \in \Sigma^*$, we say that w is trace-equivalent to w' ($w \sim_\Delta w'$) iff $w \restriction_{\Sigma_i} = w' \restriction_{\Sigma_i}, \forall i \in I$. We denote by $[w]_\Delta$ the trace-equivalence class of $w \in \Sigma^*$. A *trace-closed language* over a distribution Δ of Σ is a language L such that for all $w \in L$, $[w]_\Delta \subseteq L$.

The class of trace-closed languages is closed under the operations of union, intersection and complementation. Note that a product language is trace-closed (but the converse is not true) ([16, 21, 23]).

3 Problem Formulation and Approach

Let

$$\mathscr{E} = (V, \rightarrow_{\mathscr{E}}) \tag{1}$$

be an environment graph, where V is the set of vertices and $\rightarrow_{\mathscr{E}} \subseteq V \times V$ is a relation modeling the set of edges. For example, $\mathscr{E}$ can be the quotient graph of a partitioned environment, where V is a set of labels for the regions in the partition, and $\rightarrow_{\mathscr{E}}$ is the corresponding adjacency relation. In particular, V can be a set of labels for the roads, intersections, and parking spaces in an urban-like environment and $\rightarrow_{\mathscr{E}}$ can show how these are connected (see Fig. 2). Assume we have a team of mobile robots $\mathscr{A}_i$, $i \in I$, whose motions are restricted by $\mathscr{E}$, where I is a set of robot labels.

Let Σ be a set of service requests, or actions to be performed at the vertices of $\mathscr{E}$. The locations of the service requests are defined as a function $a : \Sigma \rightarrow V$ (*i.e.,* different requests can occur at the same vertex but vertices do not share requests). There may be no request at some vertices of $\mathscr{E}$.

We model the capacity of the robots to service requests and cooperation among robots as a distribution Δ over Σ (*i.e.* $\cup_{i \in I} \Sigma_i = \Sigma$). Σ_i is the set of requests that can be serviced by the robot $\mathscr{A}_i$. For a given request $\sigma \in \Sigma$, we define $I_\sigma = \{i \in I \mid \sigma \in \Sigma_i\}$, *i.e.,* I_σ is the set of labels of all the agents that can service request σ. The semantics

of this distribution is defined as follows. For an arbitrary request σ, if $|I_\sigma| = 1$ (*i.e.,* there is only one agent that owns it), the agent can (and should) service the request by itself, independent of the other agents. If $|I_\sigma| > 1$, all the agents $\mathscr{A}_i$ with $i \in I_\sigma$ must service the request simultaneously. An agent is said to service a request σ if it visits the vertex $a(\sigma)$. We assume that two or more robots can communicate only when they are at vertices at which shared requests occur.

We model the motion capabilities of each agent $\mathscr{A}_i$, $i \in I$ on the environment graph $\mathscr{E}$ as a transition system T_i, defined as follows:

$$T_i = (V, v_{0_i}, \rightarrow_i, \Pi, \models_i),\ i \in I, \tag{2}$$

where $v_{0_i} \in V$ is the initial state representing the initial location of $\mathscr{A}_i$, $\rightarrow_i \subseteq V \times V$ is a reflexive transition relation satisfying $\rightarrow_i \subseteq \rightarrow_{\mathscr{E}} \cup_{v \in V} \{(v,v)\}$, $\Pi = \Sigma \cup \{\varepsilon\}$ is a finite set of observations, ε is the empty request, and $\models_i \subseteq V \times \Pi$ is a satisfaction relation where $(v,\varepsilon) \in \models_i, \forall v \in V$ and $(v,\sigma) \in \models_i$, $\sigma \in \Sigma_i$, iff $a(\sigma) = v$. A transition $(v,v') \in \rightarrow$ is also denoted by $v \rightarrow v'$. For an arbitrary state $v \in V$, we define $\Pi_v = \{\pi \in \Pi \mid (v,\pi) \in \models\} \in 2^\Pi$ as the set of all observations satisfied at v. A *trajectory* of T_i is a sequence $v(0)v(1)\ldots v(n)$ with the property that $v(0) = v_{0_i}$, $v(i) \in V$, and $(v(i) \rightarrow v(i+1))$, $\forall i \geqslant 0$. We say a trajectory $v = v(0)v(1)\ldots v(n)$ of T_i satisfies a word $w = w(0)w(1)\ldots w(n)$ if $w(i) \in \Pi_{v(i)}$, $\forall i \geqslant 0$. In other words, the motion of robot $\mathscr{A}_i$ is restricted by the transition relation $\rightarrow_i$, which captures motion constraints in addition to $\rightarrow_{\mathscr{E}}$. The locations of the requests in the environment are captured by relation $\models_i$. As it will become clear later, each vertex satisfying ε captures that a robot can pass through a vertex without servicing any request.

Definition 5. A *motion and service plan* (or MS plan for short) for robot $\mathscr{A}_i$, $i \in I$ is a word $ms_i \in (V \cup \Sigma_i)^*$ that satisfies the following conditions: (1) $ms_i(1) = v_{0_i}$, (2) if $ms_i(j) \in \Sigma_i$, then $ms_i(j-1) \in V$ and $ms_i(j) \in \Pi_{ms_i(j-1)}$ (*i.e.* $ms_i(j-1) = a(ms_i(j))$), for all $j > 1$ and (3) $ms_i \restriction_V$ is a trajectory of T_i. A *motion plan* for robot $\mathscr{A}_i$, $i \in I$, defined as $m_i = ms_i \restriction_V$, can be obtained from the MS plan by deleting all request entries $ms_i(j) \in \Sigma_i$. Similarly, a *service plan* for robot $\mathscr{A}_i$, $i \in I$, is defined as $s_i = ms_i \restriction_{\Sigma_i}$, can be obtained from the MS plan by deleting all motion entries $ms_i(j) \in V$.

The semantics of an MS plan is as follows. A vertex entry $ms_i(j) \in V$ means that the vertex $ms_i(j)$ should be visited. A request entry $ms_i(j) \in \Sigma_i$, means that robot $\mathscr{A}_i$ should service request $ms_i(j)$ at vertex $ms_i(j-1)$. A request entry $ms_i(j) \in \Sigma_i$, where $|I_{ms_i(j)}| > 1$, following a vertex entry $ms_i(j-1) \in V$, triggers a *wait-and-leave* protocol (*i.e.* synchronization across the robots that share the same request $ms_i(j)$, where $|I_{ms_i(j)}| > 1$): while at $ms_i(j-1)$, robot $\mathscr{A}_i$ broadcasts request $ms_i(j)$ and listens for broadcasts of $ms_i(j)$ from all agents $\mathscr{A}_j$, $j \in I_{ms_i(j)} \setminus \{i\}$. When they are all received, the request $ms_i(j)$ is serviced and then $\mathscr{A}_i$ moves to the next vertex.

Remark 1. Note that one robot only needs to synchronize (communicate) with other robots that share a request σ with it, where $|I_\sigma| > 1$, before servicing this shared request. The loose synchronization enables parallel executions of individual agents (*i.e.* the requests that are not shared by the same robot can be serviced in parallel).

Note that by the definition (conditions (2) and (3)) of an MS plan ms_i, the motion plan $m_i = ms_i \restriction_V$ is a trajectory of T_i satisfying a word $w_i \in (\Sigma_i \cup \varepsilon)^*$, where its corresponding service plan $s_i = ms_i \restriction_{\Sigma_i}$ is equal to $w_i \restriction_{\Sigma_i}$. We say that a word s_i can be *implemented* by the robot $\mathscr{A}_i$ if there exists a MS plan ms_i such that $ms_i \restriction_{\Sigma_i} = s_i$.

Given a set of service plans $\{s_i, i \in I\}$ for the robot team, there may exist many possible sequences of requests serviced by the team due to parallel executions of individual agents (we do not assume that we know the time it takes for each agent to service requests). For a given set of MS plans $ms_i, i \in I$, we denote

$$L_{MS}^{team}(\{ms_i, i \in I\}) := \|_{i \in I} \{s_i\}, \text{ where } s_i = ms_i \restriction_{\Sigma_i}, \tag{3}$$

(see Def. (3)) as the set of all possible sequences of requests serviced by the team of robots $\mathscr{A}_i, i \in I$ while they follow their individual MS plans ms_i. For simplicity of notations, we usually denote $L_{MS}^{team}(\{ms_i, i \in I\})$ as L_{MS}^{team} when there is no ambiguity. We say that the motion of the team with MS plans $\{ms_i, i \in I\}$ satisfies a specification given as an RE ϕ over Σ if $L_{MS}^{team} \neq \emptyset$ and all words in L_{MS}^{team} satisfy ϕ (*i.e.* $L_{MS}^{team} \subseteq \mathscr{L}(A)$, where A is an FSA accepting only the words satisfying ϕ). We are now ready to formulate the main problem:

Problem 1. Given a team of agents $\mathscr{A}_i$, $i \in I$ with motion capabilities T_i (Eqn. (2)), a set of service requests Σ, a task specification ϕ in the form of an RE over Σ, and a distribution Δ over Σ, find a set of MS plans $\{ms_i, i \in I\}$ such that the motion of the team satisfies ϕ.

Remark 2. For a set of MS plans, the corresponding L_{MS}^{team} could be an empty set by the definition of product of languages (since there may not exist a word $w \in \Sigma^*$ such that $w \restriction_{\Sigma_i} = s_i, \forall i \in I$). In practice, this case corresponds to a scenario where one (or more) agent waits indefinitely for other agents to service a request σ that is shared among these agents. For example, if σ does not appear in the service plan of one of the agent who owns σ but it appears in the service plans of some other agents, then all those agents will be stuck in a "deadlock" state and wait indefinitely. When such a deadlock scenario occurs, the motion of the team does not satisfy the specification.

Remark 3. We made some apparently restrictive assumptions in the formulation of Prob. 1: we assumed that the vertices do not share requests and that the robots can communicate only when they are in the same vertex. They are made for the simplicity of notation. To relax the first assumption, we can use a relation instead of a function to define the locations of requests. The second assumption can be relaxed by introducing a communication relation on V (*i.e.* a communication graph).

In the case that Prob. 1 has a solution, for each MS plan ms_i, a robot generates a *control and communication strategy*, which is a finite sequence of control primitives, interrupts, and communication protocols. To guarantee the uniqueness of this strategy, we assume that each robot is equipped with a set of motion primitives (feedback controllers), such that the selection of a motion primitive at a vertex uniquely determines the next vertex, given that the robot is properly initialized and the history of visited vertices is known. In other words, we assume that $\mathscr{A}_i$ can follow any trajectory of T_i (see Sec. 5).

Our approach to solve Prob. 1 can be summarized as follows. We first generate an "implementable" FSA for each robot, which captures all the possible service plans that can be implemented by the robot (Sec. 4.1). Then, if the language satisfying the global specification ϕ is trace-closed, we generate a solution to the problem. Otherwise, we attempt to construct an FSA whose language is trace-closed and satisfies the global specification. If we succeed (the language of this FSA is not empty), then we use it to generate a solution (Sec. 4.2.) Our overall approach is summarized as a provably-correct algorithm in Sec. 4.2.

In our previous work [3], we provided a solution to Prob. 1 by following the "standard" approach to distributed synthesis modulo synchronous products and language equivalence [17]. As a result, our approach was conservative, since we could only generate a solution for the particular case when the language satisfying ϕ was a product language (Def. 3). In this paper, we show that we can find a solution to Prob. 1 if the language satisfying ϕ is trace-closed. Since trace-closed languages are less restrictive than product languages (*i.e.* product languages are trace-closed but not vice versa), we significantly reduce the conservatism from our previous approach. In addition, our current approach is less expensive. Indeed, checking whether a language is trace-closed is linear in the size of the FSA accepting the language, while checking whether a language is a product language is PSPACE-complete [21].

4 Synthesis of Local MS Plans from the Global Specification

We omit all the proofs in this section due to space limitations. They are available in our detailed technical report [4].

4.1 Synthesis of the Local Implementable Specifications

We begin with the conversion of the specification ϕ over Σ to a minimal and deterministic FSA $A = (Q, q_0, \Sigma, \rightarrow, F)$, which accepts exactly the language over Σ that satisfies ϕ (using JFLAP [19]). We call A the *global* specification. Given the distribution Δ, we assign requests to each agent. Specifically, we construct a set of projected FSAs $A_i = (Q_i, q_{0_i}, \Sigma_i, \rightarrow_{A_i}, F_i)$ whose languages are the projections of $\mathscr{L}(A)$ onto the local alphabets Σ_i, $i \in I$ (see Sec. 2). The projected FSAs are used as a starting point to find a solution to Prob. 1 because of the following proposition.

Proposition 1. *If a set of MS plans $\{ms_i, i \in I\}$ is a solution to Prob. 1, then its corresponding service plans $s_i = ms_i \upharpoonright_{\Sigma_i}$ are accepted words of A_i (i.e. $s_i \in \mathscr{L}(A_i), \forall i \in I$).*

However, to provide a provably correct solution for Prob. 1, it is not sufficient to simply choose an arbitrary accepted word from the projected FSAs A_i to be a service plan s_i. We need to make sure that (1) the service plan s_i can be implemented by robot $\mathscr{A}_i$ and (2) all possible sequences of requests serviced by the team satisfy ϕ. To satisfy the first requirement, we aim to find an FSA A_i^E for each $i \in I$ such that the language of A_i^E equals all the accepted words of A_i that can be implemented

by the agent $\mathscr{A}_i$ in the environment. We address the second requirement in the next sub-section.

To obtain A_i^E, we first construct a new FSA $\widehat{A}_i$ from A_i by adding the action ε to Σ_i and self-transitions (q,ε,q) to each state $q \in Q_i$. For a robot, the action ε means that no request is serviced. We denote the set of all these self transitions by $\rightarrow_{\varepsilon_i}$. The FSA $\widehat{A}_i$, $i \in I$, can now be defined as:

$$\widehat{A}_i = (\widehat{Q}_i, \widehat{q}_{0_i}, \widehat{\Sigma}_i, \rightarrow_{\widehat{A}_i}, \widehat{F}_i), \tag{4}$$

where $\widehat{Q}_i = Q_i$, $\widehat{q}_{0_i} = q_{0_i}$, $\widehat{\Sigma}_i = \Sigma_i \cup \{\varepsilon\}$, $\rightarrow_{\widehat{A}_i} = \rightarrow_{A_i} \cup \rightarrow_{\varepsilon_i}$, and $\widehat{F}_i = F_i$.

It is important to note that these self-transitions do not affect the semantics of A_i, since they mean that if no request is served by robot $\mathscr{A}_i$, then the state of the A_i remains the same. Given a word $\widehat{w} \in \mathscr{L}(\widehat{A}_i)$, we can obtain a word $w = \widehat{w} \restriction_{\Sigma_i} \in \mathscr{L}(A_i)$. Note that the input ε corresponds to the observation ε in the transition system T_i and the set of inputs $\widehat{\Sigma}_i$ of $\widehat{A}_i$ is a subset of the observations Π of T_i.

To restrict the trajectories of a TS T_i with a set of observations Π to the language accepted by an FSA with a set of actions $\widehat{\Sigma}_i \subseteq \Pi$, we define the following product automaton, which is inspired from LTL model checking [6]:

Definition 6. (Adapted from [8]) The product automaton $P_i = T_i \otimes \widehat{A}_i$ between the transition system $T_i = (V, v_{0_i}, \rightarrow_i, \Pi, \vDash_i)$ and the FSA $\widehat{A}_i = (\widehat{Q}_i, \widehat{q}_{0_i}, \widehat{\Sigma}_i, \rightarrow_{\widehat{A}_i}, \widehat{F}_i)$, is an FSA $P_i = (Q_{P_i}, q_{0_{P_i}}, \Sigma_{P_i}, \rightarrow_{P_i}, F_{P_i})$, where $Q_{P_i} = V \times \widehat{Q}_i$, $q_{0_{P_i}} = (v_{0_i}, \widehat{q}_{0_i})$ is the set of initial states, $\Sigma_{P_i} = \widehat{\Sigma}_i \subseteq \Pi$ is the set of inputs and $F_{P_i} = V \times \widehat{F}_i$ is the set of accepting (final) states. The transition relation $\rightarrow_{P_i} \subseteq Q_{P_i} \times \Sigma_{P_i} \times Q_{P_i}$ is defined as $(v,q) \xrightarrow{\sigma_{P_i}}_{P_i} (v',q')$ iff $v \rightarrow_i v'$, $q \xrightarrow{\sigma_{P_i}}_{\widehat{A}_i} q'$ and $\sigma_{P_i} \in \Pi_v$.

A transition $(v,q) \xrightarrow{\sigma}_P (v',q')$ of P_i exists iff $(v,v') \in \rightarrow_i$ and request σ occurs at vertex v, *i.e.* $a(\sigma) = v$. Transitions with input ε mean that a robot is moving from one vertex v to vertex v' (v may be equal to v') without servicing any request. $r_{P_i} = (v_i(0), \widehat{q}_i(0)) \ldots (v_i(n), \widehat{q}_i(n))$, where $\widehat{q}_i(j) \in \widehat{Q}_i$, $v_i(j) \in V$ and $j \in \{1, \ldots, n\}$ is a run accepted by the product automaton P_i, $i \in I$. We define the projection of r_{P_i} onto T_i as $\gamma_{T_i}(r_{P_i}) = v_i(0) \ldots v_i(n)$. The following proposition shows that we can use a run of P_i to find a trajectory of T_i satisfying the local specification (a word of $\mathscr{L}(\widehat{A}_i)$).

Proposition 2. *Given any word $w_{\widehat{A}_i} \in \mathscr{L}(\widehat{A}_i)$, there exist at least one trajectory of T_i satisfying $w_{\widehat{A}_i}$ iff $w_{\widehat{A}_i} \in \mathscr{L}(P_i)$.*

Finally, we obtain A_i^E that captures $\mathscr{L}(P_i)$ by removing environment information stored in P_i. To achieve this, we collapse the states of P_i, by taking ε-closure, determinizing, and minimizing P_i. The interested readers are referred to [11] for more details about these procedures. Thus, given a word $w \in \mathscr{L}(A_i^E)$, there exists a word $w' \in \mathscr{L}(P_i)$ such that $w' \restriction_{\Sigma_i} = w$. Using this fact, the following proposition shows that A_i^E captures the largest subset of $\mathscr{L}(A_i)$ which can be implemented by the robot $\mathscr{A}_i$ in the environment.

Proposition 3. *A word $w_i^E \in \mathscr{L}(A_i)$, $i \in I$, can be used to generate a MS plan ms_i for $\mathscr{A}_i$, such that $ms_i \upharpoonright_{\Sigma_i} = w_i^E$, if and only if $w_i^E \in \mathscr{L}(A_i^E)$.*

4.2 Synthesis of Individual MS Plans

To solve Prob. 1, we need to guarantee that all possible sequences of request serviced by the team of robots following their MS plans satisfy the global specification. More specifically, we aim to find a set of service plans $\{s_i, i \in I\}$ such that $\|_{i\in I}\{s_i\} \subseteq \mathscr{L}(A)$ and $\|_{i\in I}\{s_i\} \neq \emptyset$. The following proposition shows that a trace-closed language is sufficient to satisfy this requirement and provide a solution to Prob. 1:

Proposition 4. *Given a language L and a distribution Δ of Σ, if L is a trace-closed language and $w \in L$, then $\|_{i\in I}\{w \upharpoonright_{\Sigma_i}\} \subseteq L$.*

Our approach aims to construct an FSA A_G whose language is both trace-closed and included in $\mathscr{L}(A)$. By Prop. 4, an arbitrary word accepted by A_G can be used to generate a set of service plans satisfying the desired requirement by projecting this word onto the given distribution Δ. Furthermore, we use the synchronous product (SP) of the local implementable specifications generated in the previous sub-section to ensure that the obtained service plans can be implemented by individual agents. This is achieved by taking product of automata, which produces intersection of regular languages.

Specifically, to find A_G, we first check if $\mathscr{L}(A)$ is trace-closed. An algorithm that checks this property for an arbitrary FSA can be found in [4]. If $\mathscr{L}(A)$ is trace-closed, then we define $A_G = A \times \|_{i\in I} A_i^E$. Otherwise, we define $A_G = \neg(\|_{i\in I} B_i) \times \|_{i\in I} A_i^E$, where $B_i = B \upharpoonright_{\Sigma_i}$ and $B = \|_{i\in I} A_i^E \times (\neg A)$. In this second case, A_G is constructed specifically to remove words $w \in \mathscr{L}(\|_{i\in I} A_i^E)$ that cannot be used to generate desired individual service plans for the robots (*i.e.* $\|_{i\in I}\{s_i = w \upharpoonright_{\Sigma_i}\} \nsubseteq \mathscr{L}(A)$). The following proposition shows that A_G satisfies the desired requirement.

Proposition 5. *$\mathscr{L}(A_G)$ is a trace-closed language and $\mathscr{L}(A_G) \subseteq \mathscr{L}(A)$.*

If $\mathscr{L}(A_G)$ is not empty, then a solution to Prob. 1 can be found by picking any accepted word of A_G. In this paper, we obtain this word w_g by using a backward reachability search starting from the set of accepting states and ending at the initial state. In a particular application, any optimization criterion can be used. Once obtained, w_g is projected onto the given distribution to generate a set of MS plans.

The overall approach proposed in this section is summarized in Alg. 1. In the following theorem, we show that the solution obtained by Alg. 1 is provably correct.

Theorem 1. *If $\mathscr{L}(A_G) \neq \emptyset$, then Alg.1 returns a solution to Prob. 1, i.e. , a set of MS plans $\{ms_i, i \in I\}$ such that $L_{MS}^{team} \subseteq \mathscr{L}(A)$ and $L_{MS}^{team} \neq \emptyset$.*

Remark 4 (Completeness). In the case that $\mathscr{L}(A)$ is trace-closed, our approach is complete in the sense that we find a solution to Prob. 1 if one exists. This follows

Algorithm 1. Construction of a set of MS plans from a global specification

Input: A RE ϕ, a distribution Δ, and a set of TS $\{T_i = (V, v_{0_i}, \rightarrow_i, \Pi, \models_i), i \in I\}$

1: Convert ϕ to a deterministic and minimal FSA A and construct $\{A_i, i \in I\}$ ($A_i = A \upharpoonright_{\Sigma_i}, \forall i \in I$)
2: Construct $\{\widehat{A}_i, i \in I\}$ from $\{A_i, i \in I\}$ (Eqn. 4) and $\{P_i = \widehat{A}_i \otimes T_i, i \in I\}$ (Def. 6)
3: Take ε-closure, determinize, and minimize P_i to obtain $\{A_i^E, i \in I\}$, where $\mathscr{L}(A_i^E) = \mathscr{L}(P_i)$
4: Construct $\|_{i \in I} A_i^E$, which is the synchronous product of A_i^E
5: **if** $\mathscr{L}(\|_{i \in I} A_i^E) = \emptyset$, **return** no solution exists
6: **if** $\mathscr{L}(A)$ is trace-closed, $A_G = A \times \|_{i \in I} A_i^E$ **else** $A_G = \neg(\|_{i \in I}((\|_{i \in I} A_i^E \times (\neg A)) \upharpoonright_{\Sigma_i})) \times \|_{i \in I} A_i^E$
7: **if** $\mathscr{L}(A_G) = \emptyset$, **return** no solution found
8: Find a word $w_g \in \mathscr{L}(A_G)$ and obtain a set of local words $w_i^{loc} = w_g \upharpoonright_{\Sigma_i}$
9: Construct $\{\widehat{A}_i^{loc}, i \in I\}$ (Eqn. 4) from $\{A_i^{loc}, i \in I\}$, where $\mathscr{L}(A_i^{loc}) = w_i^{loc}, \forall i \in I$
10: Construct $\{P_i^{loc} = \widehat{A}_i^{loc} \otimes T_i, i \in I\}$ and find the accepted runs $\{r_{P_i^{loc}}, i \in I\}$ and the corresponding accepted words $\{w_i = w_i(0) \ldots w_i(n), i \in I\}$.
11: Obtain $\{r_{T_i} = \gamma_{T_i}(r_{P_i^{loc}}) = v_i(0) \ldots v_i(n+1), \quad i \in I\}$ and $\{ms_i = v_i(0) w_i(0) \ldots v_i(n) w_i(n) \upharpoonright_{V \cup \Sigma_i}, i \in I\}$
12: **return** a set of words $\{ms_i, i \in I\}$

directly from Prop. 3 and the definition of product of languages. If $\mathscr{L}(A)$ is not trace-closed, a complete solution to Prob. 1 requires one to find a non-empty trace-closed subset of $\mathscr{L}(A)$ if one exists. This problem is undecidable (the proof is in [4]). Therefore, our overall approach to Prob. 1 is not complete.

Remark 5 (Complexity). Checking if a language of a DFA A is trace-closed is linear in the size of A (this can be readily seen from the algorithm checking language trace-closedness in [4]). The overall complexity of Alg. 1 also depends on the construction of $\|_{i \in I} A_i^E$ and the size of A_G. Note that the construction of $\|_{i \in I} A_i^E$ and the size of A_G are not related to the size of the transition system T_i but only with A_i, which depends on the global DFA A and the distribution Δ. This fact substantiates the statement made in the introduction that we avoid constructing the parallel composition of the individual motions (represented by T_i) to prevent state-space explosion, and therefore our method scales well with the number of agents in the team. A detailed complexity analysis can be found in our technical report [4].

5 Automatic Deployment in the RULE

In our implementation, the global specification ϕ is first converted to the minimal DFA A by using JFLAP [19]. The rest of Alg. 1 is implemented in MATLAB: (1) we take a global DFA A, a distribution Δ and a set of transition systems T_i as inputs and output a set of individual MS plans for the robotic team; (2) we use Dijkstra's algorithm (see [7]) to find a word or a run accepted by an FSA by assuming that each transition of the FSA has default cost 1; if the algorithm fails to find an accepted

run, the language of this FSA is empty; (3) we implement the standard algorithm (see [11]) for taking ε-closure, determinizing a ε-NFA and minimizing a DFA. The output of Alg. 1 is then mapped to control and communication strategies (described in Sec. 3) through the use of motion primitives.

In this section, we show how our solution to Prob. 1 can be used to deploy a team of robots using a rich specification to service requests occurring in a miniature city. Our Robotic Urban-Like Environment (RULE) shown in Fig. 2 is a collection of roads, intersections, and parking lots, which are connected following a simple set of rules (*e.g.,* a road connects two (not necessarily different) intersections, the parking lots can only be located on the side of (each bound of) a road). Each intersection has traffic lights that are synchronized in the usual way. A desktop computer is used to remotely control the traffic lights through XBee wireless boards. Each parking lot consists of several parking spaces, where each parking space can accommodate exactly one car, and each parking lot has enough parking spaces to accommodate all the robots at the same time. The city is easily reconfigurable through re-taping and re-placement of the wireless traffic lights in intersections.

The robots are Khepera III miniature cars. Each car can sense when entering an intersection from a road, when entering a road from an intersection, when passing in front of a parking lot, when it is correctly parked in a parking space, and when an obstacle is dangerously close. Each car can distinguish the color of a traffic light and different parking spaces in the same parking lot. Each car is programmed with motion and communication primitives allowing it to safely drive on a road, turn in an intersection, park, and communicate with other cars. All the cars can communicate through Wi-Fi with a desktop computer, which is used as an interface to the user (*i.e.,* to enter the global specification) and to perform all the computation necessary to generate the individual control and communication strategies. Once computed, these are sent to the cars, which execute the task autonomously by interacting with the environment and by communicating with each other, if necessary. We assume that the communication protocol is deadlock-free.

Modeling RULE using the framework described in Sec. 3 proceeds as follows. The set of vertices V of the environment graph $\mathscr{E}$ is the set of labels assigned to the roads, intersections, and parking lots (see Fig. 2). The edges in $\rightarrow_{\mathscr{E}}$ show how these regions are connected. We assume that inter-robot communication is possible

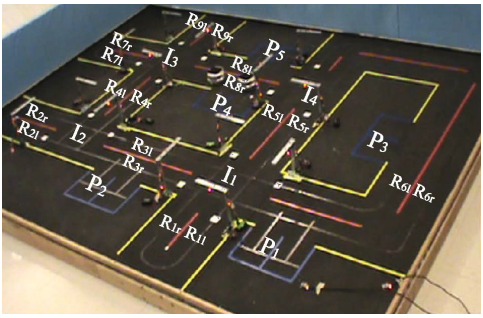


Fig. 2 The topology of the city for the case study from Sec. 5 and the labels of the roads, intersections, and parking lots

only when the robots are in the same parking lot. The motion capabilities of the (identical) robots are captured by a transition system T_i (Eqn. (2)) which has 27 vertices and 42 transitions. Note that, in reality, each vertex of T_i has associated a set of motion primitives, and each transition is triggered by a Boolean combination of interrupts. For example, at vertex R_{5l}, only one motion primitive *follow_road* is available, which allows the robot to drive on the road. There is only one possible transition from R_{5l} to I_1, which is triggered by *at_int* AND *green_light*, where *at_int* is an interrupt generated when the robot reaches the end of a road at an intersection, and *green_light* is an interrupt generated at the green color of the traffic light.

It is important to note that, by selecting a motion primitive available at a vertex, the robot can correctly execute a run of T_i, given that it is initialized on a road. Indeed, only one motion primitive (*follow_road*) is available on a road (more details about the motion primitives can be found in [4]). In other words, MS plans defined in Sec. 3 and derived as described in Sec. 4 can be immediately implemented by a robot. It is easy to see that, under some reasonable liveness assumptions about environmental events (*e.g.,* the traffic lights will eventually turn green), such a transition system captures the motion of each robot correctly.

In the rest of this section, we present a case study. Assume that two robots (cars), labeled as C_1 and C_2, are available for deployment in the city with the topology from Fig. 2. Assume the set of service requests is $\Sigma = \{H_1, H_2, L_1, L_2, L_3, L_4, L_5\}$, where L_i, $i = 1,2,3,4,5$ are "light" requests, which require only one robot, and therefore should be serviced in parallel, while H_i, $i = 1,2$ are "heavy", and require the cooperation of the two robots. Assume that C_1 can service L_1 and L_4 and C_2 can service L_2, L_3 and L_5, *i.e.,* the set of requests is distributed as $\Sigma_1 = \{L_1, L_4, H_1, H_2\}$, $\Sigma_2 = \{L_2, L_3, L_5, H_1, H_2\}$. between the two agents. Assume the requests occur at the parking lots as given by the assignment function $a(L_1) = P_1$, $a(L_2) = P_2$, $a(L_3) = P_3$, $a(L_4) = P_4$, $a(L_5) = P_1$, $a(H_1) = P_4$, and $a(H_2) = P_5$. Finally, assume that the global task specification is to service L_4 and then L_5 or first service H_1, then both L_1 and L_2 in an arbitrary order, then H_2, and finally both L_1 and L_3 in an arbitrary order. Formally, this specification translates to the following RE over Σ: $L_4L_5 + H_1\ (L_1L_2\ +\ L_2L_1)\ H_2\ (L_1L_3\ +\ L_3L_1)$.

Using Alg. 1, we generate a set of FSAs A_i^E. Since RULE is fully connected, all the words accepted by A_i can be implemented. In this example, $\mathscr{L}(A)$ is neither a product language nor a trace-closed language (*e.g.*, for $w = L_4L_5$, we have $[w]_\Delta = \{L_4L_5, L_5L_4\}$ and hence, $[w]_\Delta \not\subseteq \mathscr{L}(A)$). Therefore, the FSA A_G is obtained as described in Sec. 4.2. We choose $w_g = H_1L_1L_2H_2L_1L_3 \in \mathscr{L}(A_G)$. The corresponding service plans for C_1 and C_2 are $s_1 = H_1L_1H_2L_1$ and $s_2 = H_1L_2H_2L_3$, respectively. The FSAs generated by Alg. 1 are shown in Fig. 3. Finally, we generate the MS plans for C_1 and C_2. By assuming that C_1 and C_2 start in R_{2l} and R_{1l} respectively, the two MS plans are

$$ms_1 : R_{2l}I_2R_{4r}I_3R_{8r}P_4H_1R_{8r}I_4R_{5l}I_1R_{6r}P_1L_1R_{6r}I_4R_{8l}P_5H_2R_{8l}I_3R_{8r}I_4R_{5l}I_1R_{6r}P_1L_1$$

$$ms_2 : R_{1l}I_1R_{3l}I_2R_{4r}I_3R_{8r}P_4H_1R_{8r}I_4R_{5l}I_1R_{3l}I_2R_{3r}P_2L_2R_{3r}I_1R_{5r}I_4R_{8l}P_5H_2R_{8l}I_3R_{8r}I_4R_{6l}P_3L_3$$

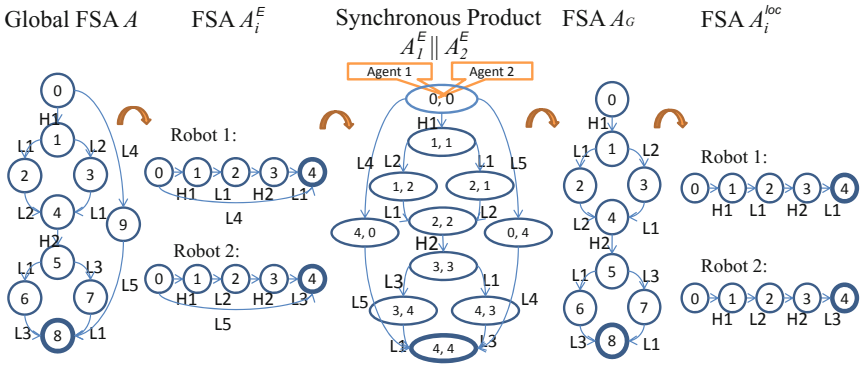


Fig. 3 The FSAs generated by applying Alg. 1

The request entry H_1 and H_2 will trigger the wait-and-leave protocol (see Sec. 3) since they are shared by both robots. The above MS plans are then mapped to control and communication strategies through the use of motion primitives and interrupts.

To demonstrate that our method scales well with respect to the number of agents in the team, we deploy 5 agents in a simulator for the RULE platform. Specifically, in this case study, the global FSA A has 9 states, the transition system T_i for each robot has 26 vertices and 41 transitions, the synchronous product $\|_{i \in I} A_i^E$ has 37 states and the determinized and minimized FSA A_G has 9 states. The generation of the MS plans for both case studies described in this section takes less than 2 seconds. The movies for the actual deployment in the RULE platform and the simulator are both available at `http://hyness.bu.edu/RULE_media.html`.

6 Conclusion

We presented a framework for automatic deployment of a robotic team from a specification given as a regular expression over a set of service requests occurring at known locations of a partitioned environment. Given the robot capabilities to service the requests, and the possible cooperation requirements for some requests, we find local control and communication strategies such that the global behavior of the system satisfies the given specification. We illustrate the proposed method with experimental results in our Robotic Urban-Like Environment (RULE).

We are currently pursuing several future directions. We are expanding the set of global specifications to formulas of temporal logics, such as Linear Temporal Logic, to enrich the expressiveness of the global specifications. We are also working on extensions to probabilistic frameworks. Specifically, we will use formulas of probabilistic temporal logics, such as probabilistic Linear Temporal Logic (pLTL). The satisfaction of the global specification will be guaranteed probabilistically and the deterministic transition systems will be replaced with Markov Decision Processes.

Acknowledgements. We are grateful to all reviewers for the thoughtful comments. This work was partially supported by ONR-MURI N00014-09-1051, ARO W911NF-09-1-0088, AFOSR YIP FA9550-09-1-020, and NSF CNS-0834260 at Boston University and by CNCSIS-UEFISCSU no. 7/05.08.2010 at the University of Pitesti.

References

1. Antoniotti, M., Mishra, B.: Discrete event models + temporal logic = supervisory controller: Automatic synthesis of locomotion controllers. In: IEEE International Conference on Robotics and Automation (1995)
2. Belta, C., Habets, L.: Control of a class of nonlinear systems on rectangles. IEEE Transactions on Automatic Control 51(11), 1749–1759 (2006)
3. Chen, Y., Birch, S., Stefanescu, A., Belta, C.: A hierarchical approach to automatic deployment of robotic teams with communication constraints. In: IEEE/RSJ International Conference on Intelligent Robots & Systems, Taipei, Taiwan, pp. 5079–5084 (2010)
4. Chen, Y., Ding, X.C., Stefanescu, A., Belta, C.: A formal approach to deployment of robotic teams in an urban-like environment. Tech. rep., Boston University (2010), `hyness.bu.edu/dars`
5. Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L., Thrun, S.: Principles of Robot Motion: Theory, Algorithms, and Implementations. MIT Press (2005)
6. Clarke, E.M., Peled, D., Grumberg, O.: Model Checking. MIT Press (1999)
7. Cormen, T.: Introduction to Algorithms. MIT press (2001)
8. Fainekos, G., Kress-Gazit, H., Pappas, G.: Hybrid controllers for path planning: A temporal logic approach. In: IEEE Conference on Decision and Control and European Control Conference, Seville, Spain, pp. 4885–4890 (2005), doi:10.1109/CDC.2005.1582935
9. Gazit, H.K., Fainekos, G., Pappas, G.J.: Where's Waldo? Sensor-based temporal logic motion planning. In: IEEE Conference on Robotics and Automation, Rome, Italy (2007)
10. Habets, L., Collins, P., van Schuppen, J.: Reachability and control synthesis for piecewise-affine hybrid systems on simplices. IEEE Trans. Aut. Control 51, 938–948 (2006)
11. Hopcroft, J., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley (2007)
12. Kloetzer, M., Belta, C.: Automatic deployment of distributed teams of robots from temporal logic motion specifications. IEEE Transactions on Robotics 26(1), 48–61 (2010)
13. Latombe, J.C.: Robot Motion Planning. Kluwer Academic Publishers (1991)
14. LaValle, S.M.: Planning algorithms. Cambridge University Press, Cambridge (2006)
15. Loizou, S.G., Kyriakopoulos, K.J.: Automatic synthesis of multiagent motion tasks based on LTL specifications. In: IEEE Conference on Decision and Control, Paradise Islands, The Bahamas, pp. 153–158 (2004)
16. Mazurkiewicz, A.: Introduction to trace theory. In: The Book of Traces, pp. 3–41. World Scientific (1995)
17. Mukund, M.: From Global Specifications to Distributed Implementations, pp. 19–34. Kluwer Academic Publishers (2002)
18. Quottrup, M.M., Bak, T., Izadi-Zamanabadi, R.: Multi-robot motion planning: A timed automata approach. In: IEEE International Conference on Robotics and Automation, Barcelona, Spain, pp. 4417–4422 (2004)
19. Rodger, S.H., Finley, T.W.: JFLAP: An Interactive Formal Languages and Automata Package. Jones and Bartlett Publishers (2006)
20. Sheng, Y.: Regular Languages. Springer, New York (1997)
21. Stefanescu, A.: Automatic synthesis of distributed transition systems. Ph.D. thesis, University of Stuttgart (2006)

22. Ştefănescu, A., Esparza, J., Muscholl, A.: Synthesis of Distributed Algorithms Using Asynchronous Automata. In: Amadio, R.M., Lugiez, D. (eds.) CONCUR 2003. LNCS, vol. 2761, pp. 27–41. Springer, Heidelberg (2003)
23. Thiagarajan, P.S., Henriksen, J.G.: Distributed Versions of Linear Time Temporal Logic: A Trace Perspective. In: Reisig, W., Rozenberg, G. (eds.) APN 1998. LNCS, vol. 1491, pp. 643–681. Springer, Heidelberg (1998)
24. Wongpiromsarn, T., Topcu, U., Murray, R.M.: Receding horizon temporal logic planning for dynamical systems. In: IEEE Conference on Decision and Control and Chinese Control Conference, Shanghai, China, pp. 5997–6004 (2009)

Heuristic Planning for Decentralized MDPs with Sparse Interactions

Francisco S. Melo and Manuela Veloso

Abstract. In this work, we explore how local interactions can simplify the process of decision-making in multiagent systems, particularly in multirobot problems. We review a recent decision-theoretic model for multiagent systems, the decentralized sparse-interaction Markov decision process (Dec-SIMDP), that explicitly distinguishes the situations in which the agents in the team must coordinate from those in which they can act independently. We situate this class of problems within different multiagent models, such as MMDPs and transition independent Dec-MDPs. We then contribute a new general approach that leverages the particular structure of Dec-SIMDPs to efficiently plan in this class of problems, and propose two algorithms based on this underlying approach. We pinpoint the main properties of our approach through illustrative examples in multirobot navigation domains with partial observability, and provide empirical comparisons between our algorithms and other existing algorithms for this class of problems. We show that our approach allows the robots to look ahead for possible interactions, planning to accommodate such interactions and thus overcome some of the limitations of previous methods.

1 Introduction

Recent years have witnessed a profusion of work on multiagent models that capture some of the fundamental features of Dec-(PO)MDPs (such as partial observability) without incurring in the associated computational cost. In this paper, we contribute to this extensive literature, and investigate a recent model for cooperative multiagent decision-making in the presence of global partial observability [17]. This model is

Francisco S. Melo
INESC-ID/Instituto Superior Técnico, UTL, Av. Prof. Dr. Cavaco Silva,
2780-990 Porto Salvo, Portugal
e-mail: fmelo@inesc-id.pt

Manuela Veloso
Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA
e-mail: veloso@cs.cmu.edu

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 329–343.
springerlink.com

motivated by the observation that, in many real-world scenarios involving multiple decision makers (*e.g.*, robots), the tasks of the different agents/robots are not coupled at every decision-step but only in relatively infrequent situations. We dub such problems as having *sparse interaction*. *Multi-robot systems* provide our primary motivation and constitute natural examples for the class of problems considered herein. In multi-robot systems, the interaction among the different robots is naturally limited by each robot's physical boundaries (workspace, communication range, etc.) and limited perception capabilities. Therefore, when dealing with multi-robot systems, one natural approach is to subdivide the overall task into smaller tasks that each robot can then execute autonomously or as part of a smaller group [5, 15, 18].

Several previous works have exploited simplified models of interaction in multi-agent settings. For example, a hierarchical learning algorithm can consider only the interaction between the different agents at a higher control level, while allowing the agents to learn lower level tasks independently [6]. Also, coordination graphs can represent compactly the dependencies between the actions of different agents, thus capturing the local interaction between them [8, 10]. Local interactions have also been exploited to minimize communication during policy execution [16] and in the game-theoretic literature to attain compact game representations [9, 20].

In this paper we consider Dec-MDPs with parse interactions (henceforth Dec-SIMDPs). Dec-SIMDPs have been proposed in [17] under the designation of *interaction-driven Markov games* and are closely related to *distributed POMDPs with coordination locales* [19] and *Dec-MDPs with event-driven interactions and complex rewards* [14]. Dec-SIMDPs leverage the independence between agents to decouple the decision process in significant portions of the joint state space, allowing the agents to base their decisions in their local perception of state and alleviating the difficulties arising from global partial observability. On those situations in which the agents interact, Dec-SIMDPs rely on communication to bring down the the computational complexity of the joint decision process. Dec-SIMDPs "balance" the independence assumptions with communication: in any given state, the agents are either independent or can communicate.[1]

The contributions in this paper are two-fold. On one hand, we build on [17], providing a precise formalization of Dec-SIMDPs and discussing in some detail the relation with well-established decision-theoretic models such as Dec-MDPs, MMDPs and MDPs. On the other hand, we contribute two new algorithms that exhibit significant computational savings when compared to existing algorithms for Dec-SIMDPs. We illustrate the application of our algorithms in several simple navigation tasks.

2 Decision Theoretic Models

We start by reviewing *decentralized partially observable Markov decision processes* (Dec-POMDPs) and related decision theoretic models. A N-agent Dec-POMDP $\mathscr{M}$

[1] We note that both independence assumptions and communication can significantly bring down the computational complexity in Dec-(PO)MDP related models [1, 7].

is specified as a tuple $\mathscr{M} = (N, \mathscr{X}, (\mathscr{A}_k), (\mathscr{Z}_k), \mathsf{P}, (\mathsf{O}_k), r, \gamma)$, where $\mathscr{X}$ is the joint state-space, $\mathscr{A} = \times_{k=1}^{N} \mathscr{A}_k$ is the set of joint actions, with each $\mathscr{A}_k$ the individual action set for agent k, each $\mathscr{Z}_k$ represents the set of possible local observation for agent k, $\mathsf{P}(x,a,y)$ represents the transition probabilities from joint state x to joint state y when the joint action a is taken, each $\mathsf{O}_k(x,a,z_k)$ represents the probability of agent k making the local observation z_k when the joint state is x and the last joint action taken was a, and $r(x,a)$ represents the expected reward received by all agents for taking the joint action a in joint state x. The scalar γ is a discount factor.

A *N-agent Decentralized Markov decision process* (Dec-MDP) is a particular class of Dec-POMDP in which the state is *jointly fully observable*. Formally this can be translated into the following condition: for every joint observation $z \in \mathscr{Z}$, with $\mathscr{Z} = \times_{k=1}^{N} \mathscr{Z}_k$, there is a state $x \in \mathscr{X}$ such that $\mathbb{P}[X(t) = x \mid Z(t) = z] = 1$, where $X(t)$ is the joint state of the process at time t and $Z(t)$ the corresponding joint observation. Similarly, a *partially observable Markov decision process* (POMDP) is a 1-agent Dec-POMDP and a *Markov decision process* (MDP) is a 1-agent Dec-MDP. Finally, a *N-agent multiagent MDP* (MMDP) is a N-agent Dec-MDP that is *fully observable*, *i.e.*, for every individual observation $z_k \in \mathscr{Z}_k$ there is a state $x \in \mathscr{X}$ such that $\mathbb{P}[X(t) = x \mid Z_k(t) = z_k] = 1$.

In the remainder of the paper paper we focus on Dec-MDPs, particularly in Dec-MDPs for which the state-space $\mathscr{X}$ can be factorized as $\mathscr{X} = \mathscr{X}_1 \times \ldots \times \mathscr{X}_N$. Although more general Dec-MDP models exist [3], we adhere to this simplified version, as this is sufficient for our purposes and makes the presentation both clearer and simpler. Indeed, since multirobot navigation scenarios constitute the main motivation behind our work, the sensible approach is, in fact, to consider a factored joint state-space, where each $\mathscr{X}_k$ denotes the individual state-space for robot k. For future reference, let $\mathscr{X}_{-k} = \mathscr{X}_0 \times \ldots \times \mathscr{X}_{k-1} \times \mathscr{X}_{k+1} \times \ldots \times \mathscr{X}_N$ and denote by x_{-k} a general element of $\mathscr{X}_{-k}$. We also write $x = (x_{-k}, x_k)$ to denote the fact that the kth component of x takes the value x_k. We use a similar notation for actions.

In this partially observable multiagent setting, an individual (non-Markov) policy for agent k is a mapping $\pi_k : \mathscr{H}_k \longrightarrow \Delta(\mathscr{A}_k)$, where $\Delta(\mathscr{A}_k)$ is the space of probability distributions over $\mathscr{A}_k$ and $\mathscr{H}_k$ is the set of all possible finite histories (finite sequences of actions and observations) for agent k.

In a Dec-MDP, the purpose of all agents is to determine a joint policy π so as to maximize the total sum of discounted rewards. In order to write this in terms of a function, we consider a distinguished initial state, $x^0 \in \mathscr{X}$, that is assumed common knowledge among all agents. The purpose of the agents is then to maximize

$$V^{\pi} = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r\big(X(t), A(t)\big) \mid X(0) = x^0\right].$$

Transition-independent Dec-MDPs [2] constitute a particular subclass of Dec-MDPs in which, for all $(x,a) \in \mathscr{X} \times \mathscr{A}$,

$$\mathbb{P}[X_k(t+1) = y_k \mid X(t) = x, A(t) = a] = \mathbb{P}[X_k(t+1) = y_k \mid X_k(t) = x_k, A_k(t) = a_k]. \tag{1}$$

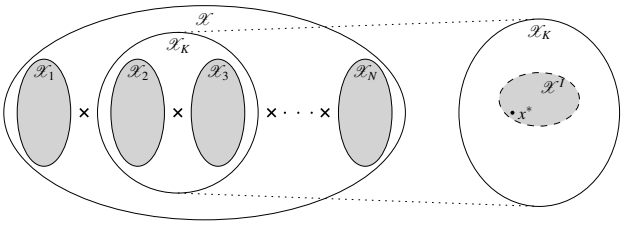


Fig. 1 Diagram representing the relation between individual state-spaces, $\mathcal{X}_k$, the joint state-space $\mathcal{X}$, and the set $\mathcal{X}_K$ for a set of agents $K = \{2,3\}$. We also represent an interaction area $\mathcal{X}^I$ associated with an interaction state $x^* \in \mathcal{X}_K$ (see main text).

The transition probabilities can thus be factorized as

$$\mathsf{P}(x,a,y) = \prod_{k=1}^{N} \mathsf{P}_k(x_k,a_k,y_k), \tag{2}$$

where $\mathsf{P}_k(x_k,a_k,y_k)$ represents the transition probabilities from local state x_k to local state y_k when the individual action a_k was taken. This particular class of Dec-MDPs has been shown to be NP-complete in finite-horizon settings, versus the NEXP-completeness of general Dec-MDPs [7].[2]

Similarly, *reward independent Dec-MDPs* correspond to a subclass of Dec-MDPs in which, for all $(x,a) \in \mathcal{X} \times \mathcal{A}$, $r(x,a) = f(r_k(x_k,a_k), k = 1,\ldots,N)$, *i.e.*, the global reward function r can be obtained from local reward functions $r_k, k = 1,\ldots,N$, and each individual reward is consistent with the global reward [7]. One typical example is

$$r(x,a) = \sum_{k=1}^{N} r_k(x_k,a_k). \tag{3}$$

Interestingly, it was recently shown that reward independent Dec-MDPs retain NEXP-complete complexity [1]. However, when associated with transition independence, reward independence implies that a Dec-MDP can be decomposed into N independent MDPs, each of which can be solved separately. The complexity of this class of problems thus reduces to that of standard MDPs (P-complete). For a summary of complexity results for Dec-POMDP related models, we refer to [1, 7].

3 Local Interactions in Dec-MDPs

In this paper we exploit sparse interactions among the different agents in a Dec-MDP. In particular, we are interested in Dec-MDPs in which there is some level of both transition and reward dependency, but this dependency is limited to specific regions of the state space. We introduce decentralized sparse-interaction MDPs

[2] In this paper we are interested in infinite horizon problems. Complexity results for infinite-horizon problems with partial observability are even more discouraging—even single-agent POMDPs have been shown undecidable in infinite-horizon settings [12].

(Dec-SIMDPs). Dec-SIMDPs essentially correspond to the model previously proposed in [17] under the designation of *interaction-driven Markov games*. However, we revisit several aspects of this model that were not properly formalized in the original work, and provide a more extensive discussion on the relation between this work and the models surveyed in the previous section. We postpone to the following section the introduction of two novel algorithms for this class of problems.

We start by introducing some auxiliary notation. Given an N-agent Dec-MDP $\mathscr{M} = (N, \mathscr{X}, (\mathscr{A}_k), \mathsf{P}, r, \gamma)$, let K be a subset of the N agents in $\mathscr{M}$. Extending the notation in Section 2, we denote by $\mathscr{X}_K = \times_{k \in K} \mathscr{X}_k$ the joint state-space of all agents in K. Similarly, we write $\mathscr{X}_{-K}$ to denote the joint state-space of the agents *not* in K. We write x_K to denote a general element of $\mathscr{X}_K$ and x_{-K} to denote a general element of $\mathscr{X}_{-K}$. We write $x = (x_{-K}, x_K)$ to distinguish the components of x corresponding to agents in K and those corresponding to agents not in K (see Fig. 1 for an illustration).

Also, for any given a Dec-MDP, we write the reward $r(x,a)$ as

$$r(x,a) = \sum_{k=1}^{N} r_k(x_k, a_k) + \sum_{i=1}^{M} r_i^I(x_{K_i}, a_{K_i}), \tag{4}$$

where each r_k corresponds to an individual component of the reward function that depends only on agent k and there are M agent sets, $K_i, i = 1, \ldots, M$, and M reward components, r_i^I (the interaction components), each depending on all the agents in K_i and only on these. We note that this decomposition can be performed at no loss of generality, since any reward r can be trivially written in that form by setting $M = 1$, $r_k \equiv 0$, $K_1 = \{1, \ldots, N\}$, and $r_1^I = r$. The scenarios that we are interested in are those in which the support of $\sum_{i=1}^{M} r_i^I$ – the subset of $\mathscr{X} \times \mathscr{A}$ in which this sum is non-zero – is small when compared with $\mathscr{X} \times \mathscr{A}$.

We say that an agent k_0 in a Dec-MDP is *independent* of an agent k_1 in a state $x \in \mathscr{X}$ if the transition probabilities for the individual state of agent k_0 at x do not depend on the state/action of agent k_1, *i.e.*,

$$\mathbb{P}\left[X_{k_0}(t+1) = y_{k_0} \mid X(t) = x, A(t) = a\right] = \mathbb{P}\left[X_{k_0}(t+1) = y_{k_0} \mid X_{-k_1}(t) = x_{-k_1}, A_{-k_1}(t) = a_{-k_1}\right].$$

and it is possible to decompose the global reward function $r(x,a)$ as in (4) in such a way that no set K_i contains both k_0 and k_1. When any of the above does not hold, we say that agent k_0 *depends* on k_1 at state x. This notion of dependence extends trivially to sets of agents by interpreting the agents in each set as a single centralized agent. Intuitively, two agents are dependent if either the rewards or the transitions of one of the agents depend on the state or action of the other.

The agents in a set K *interact* at state $x \in \mathscr{X}$ if the following conditions hold:

- If $k_0 \in K$ and agent k_0 depends on agent k_1 in state x, then $k_1 \in K$.
- If $k_1 \in K$ and there is an agent k_0 that depends on agent k_1 in state x, then $k_0 \in K$.
- There is no strict subset $K' \subset K$ such that the above conditions hold for K'.

If the agents in a set K interact in a state x, then we refer to x_K as an *interaction state* for the agents in K. Interactions capture all dependencies between the agents in K: if the agents in K interact in state x_K, no agent in K is independent of all others in x_K and no agent outside K depends on any agent in K.

In a general Dec-MDP, all agents interact in all states, since generally there are no transition or reward independences. On the other hand, transition and reward independent Dec-MDPs have no interactions at all – as expected, such problems can be decomposed into N independent single-agent models and solved in a straightforward manner. An interaction occurs whenever a group of agents is coupled in terms of either transitions or rewards and either the transition probabilities cannot be factorized as in (2) or the reward function cannot be decomposed as in (3).

In a general N-agent Dec-MDP, we define an *interaction area* $\mathscr{X}^I$ as follows:

- $\mathscr{X}^I \subset \mathscr{X}_K$ for some set of agents K;
- $\exists_{x^* \in \mathscr{X}^I}$ such that x^* is an interaction state for the agents in K;
- The set $\mathscr{X}^I$ is connected.[3]

An agent k is involved in an interaction at time t if there is one interaction area $\mathscr{X}^I$ involving a set of agents K such that $k \in K$ and $X(t) = (x_K, x_{-K})$ with $x_K \in \mathscr{X}^I$. We represent the concept of interaction area in the diagram of Fig. 1.

The purpose of defining/identifying the interaction areas in a Dec-MDP is to single out situations in which the actions of one agent depend on other agents. An agent that is not involved in any interaction should be able to choose its individual actions independently of the other agents and thus be unaffected by partial (global) state observability. In contrast, we focus on those problems for which each of the agents involved in an interaction in a particular interaction area $\mathscr{X}^I \subset \mathscr{X}_K$ at time t has full access to the state $X_K(t)$. We refer to such a Dec-MDP as having *observable interactions*. Our focus on Dec-MDPs with observable interactions, although apparently restrictive, actually translates a property often observed in real-world scenarios. For example, when interacting, robots are often able to observe/communicate relevant information for coordination. In a sense, interaction areas encapsulate the need for information sharing in a general multiagent decision problem.

We are now in position to introduce our model. A N-agent Dec-MDP $\mathscr{M}$ has *sparse interactions* if all agents are independent except in a set of M interaction areas, $\{\mathscr{X}_1^I, \ldots, \mathscr{X}_M^I\}$, with $\mathscr{X}_i^I \subset \mathscr{X}_{K_i}$ for some set of agents K_i, and such that $|\mathscr{X}_i^I| \ll |\mathscr{X}_{K_i}|$. We refer to a Dec-MDP with sparse observable interactions as a Dec-SIMDP (decentralized sparse-interaction MDP). For all agents outside interaction areas, the joint transition probabilities and reward function for a Dec-SIMDP can be factorized as in (2) and (3), and it is possible to model these agents using "individual MDPs". On the other hand, the agents involved in an interaction can be modeled using a "local" MMDP. We represent such a Dec-SIMDP as a tuple

$$\Gamma = \big(\{\mathscr{M}_k, k = 1, \ldots, N\}, \{(\mathscr{X}_i^I, \mathscr{M}_i^I), i = 1, \ldots, M\}\big),$$

where

- Each $\mathscr{M}_k$ is an MDP $\mathscr{M}_k = (\mathscr{X}_k, \mathscr{A}_k, \mathsf{P}_k, r_k, \gamma)$ that individually models agent k in the absense of other agents, where r_k is the component of the joint reward function associated with agent k in the decomposition in (3);

[3] In this context we say that a set $U \subset \mathscr{X}$ is *connected* if, for any pair of states $x, y \in U$, there is a sequence of actions that, with positive probability, yields a trajectory $\{x(0), \ldots, x(T)\}$ such that $x(t) \in U, t = 0, \ldots, T$, and either $x(0) = x$ and $x(T) = y$ or vice-versa.

- Each $\mathscr{M}_i^I$ is an MMDP that captures a *local interaction* between K_i agents in the states in $\mathscr{X}_i^I$ and is given by $\mathscr{M}_i^I = (K_i, \mathscr{X}_{K_i}, (\mathscr{A}_k), \mathsf{P}_i^I, r_i^I, \gamma)$, with $\mathscr{X}_i^I \subset \mathscr{X}_{K_i}$.

Each MMDP $\mathscr{M}_i$ describes the interaction between a subset K_i of the N agents, and the corresponding state-space, $\mathscr{X}_{K_i}$, is a superset of the respective interaction area.

A Dec-SIMDP is an alternative way of representing a Dec-MDP with observable interactions. In the states of each interaction area in a Dec-SIMDP (and only in these), the agents involved in the associated MMDP are able to observe their joint state. This can be interpreted as having the agents in this area use communication to overcome local state perception and decide jointly on their action. Outside these areas, the agents have only a local perception of the state and should, therefore, choose the actions independently of the other agents.

Note that, in the absence of any interaction areas, the Dec-SIMDP reduces to a set of independent MDPs that can be solved separately. This captures the situation in which the agents are completely independent. On the other hand, a Dec-SIMDP is a Dec-MDP model with joint state observability in the interaction areas. In those situations in which all agents interact in all states, as assumed in the general Dec-MDP model, the whole state-space is an interaction area and, as such, our assumption of full state observability in the interaction areas renders our model equivalent to an MMDP. Nevertheless, the appeal of the Dec-SIMDP model is that many practical situations do not fall in either of the two extreme cases (*i.e.*, independent MDPs vs. fully observable MMDP). It is in these situations that the Dec-SIMDP model may bring an advantage over more general (but potentially intractable) models.

4 Planning in Dec-SIMDPs

We now introduce two novel Dec-SIMDP algorithms that leverage the particular structure of this class of problems and avoid the computational complexity of more general Dec-MDP models. Our approach relies on a simple heuristic that provides interesting insights into the structure of Dec-SIMDP and on how should the interaction areas be chosen for a particular problem. As in most planning problems, the underlying Dec-MDP/Dec-SIMDP model is assumed known.

4.1 MPSI and LAPSI

Let us start by considering a Dec-SIMDP in which all except agent k have full state observability. Let us further suppose that the agents with full state observability follow some fixed known policy π_{-k}. Then, from the perspective of agent k, the environment behaves as a POMDP, since the other agents can be collectively regarded as part of the environment. In this particular situation, we can use any POMDP solution method to compute the policy for agent k.

Our heuristic departs from the simplified situation just described. For each agent $k = 1, \ldots, N$, we assume all other agents to follow some (hypothesized) policy $\hat{\pi}_{-k}$

Algorithm 1. General outline of the proposed heuristic planning algorithms.

Require: Dec-SIMDP model $\mathscr{M} = \big(\{\mathscr{M}_k, k = 1,\ldots,N\}, \{(\mathscr{X}_i^I, \mathscr{M}_i^I), i = 1,\ldots,M\}\big)$
1: **for all** $k = 1,\ldots,N$ **do**
2: Build hypothetical policy $\hat{\pi}_{-k}$ for other agents
3: From $\mathscr{M}$ and $\hat{\pi}_{-k}$ build POMDP model for agent k, $(\mathscr{X}, \mathscr{A}_k, \mathscr{Z}_k, \mathsf{P}_{\hat{\pi}_{-k}}, r_{\hat{\pi}_{-k}}, \gamma)$
4: Use preferred POMDP solution technique to compute $\pi_k : \Delta(\mathscr{X}) \to \mathscr{A}_k$
5: **end for**

that depends only on the state X_t. Given this policy $\hat{\pi}_{-k}$, we derive the POMDP model for agent k and use the corresponding solution as the policy π_k. Algorithm 1 summarizes this approach.

This heuristic rests on the assumption that the hypothesized policy, $\hat{\pi}_{-k}$, will allow agent k to approximately "track" the other agents and hence choose its actions accordingly. The closer $\hat{\pi}_{-k}$ is to the actual policy of the other agents, the better agent k will be able to track them, and the better he will decide.

The two algorithms proposed in this paper, dubbed MPSI (Myopic Planning for Sparse Interactions) and LAPSI (Look-Ahead Planning for Sparse Interactions), share this underlying structure but consider different hypothetical policies $\hat{\pi}_{-k}$ in Step 2. In MPSI, agent k models the other agents as self-centered and oblivious to the interactions. In other words, agent k acts as if each agent j, $j \neq k$, is following the optimal policy for the corresponding MDP $\mathscr{M}_j$ in the Dec-SIMDP model. In environments with almost no interaction, MPSI actually provides a good approximation to the policy of the other agents outside the interaction areas.

In contrast, in LAPSI, agent k considers that all other agents jointly adopt the optimal policy for the underlying MMDP. LAPSI is, in a sense, the counterpart to MPSI, as it provides a good approximation to the policy of the other agents in scenarios where the interactions are not so sparse.

Clearly, the idea in Algorithm 1 can be used in general Dec-POMDPs. However, the hypothetical policy $\hat{\pi}_{-k}$ will seldom correspond to the actual policy followed by the other agents, and it is only natural that this method will not allow each agent k to properly "track" the other agents and decide accordingly, this leading to poor results in general Dec-POMDPs. The particular structure of Dec-SIMDPs, however, renders this approach more appealing for two reasons: on one hand, outside interaction areas the policy of agent k (ideally) exhibits minimum dependence on the state/policy of the other agents. As such, poor tracking in these areas has little impact on the policy of agent k. In interaction areas, on the other hand, local full observability allows agent k to perfectly track the other agents involved in the interaction and choose its actions accordingly.

In the following subsection, we describe a specific instance of both MPSI and LAPSI that is closely related to the Q_{MDP} heuristic for POMDPs [11] and rests on the concept of *generalized α-vectors*. As will soon become apparent, even using such a simple POMDP solver such as Q_{MDP}, LAPSI is able to attain near-optimal performance in all test scenarios while incurring in a computational cost much lower than alternative methods.

4.2 Generalized α-Vectors

We now propose particular instances of both MPSI and LAPSI that is closely related with the Q-MDP heuristic for POMDPs [11], although exploiting the structure of the Dec-SIMDPs model.

To this purpose, we note that each agent k in a Dec-SIMDP has *full local state observability*, implying that, at each time-step t, the kth component of the state, $X_k(t)$, is always unambiguously determined. Furthermore, given our assumption of observable interactions, at each time step t only those state-components corresponding to agents *not interacting* with agent k will be unobservable. By definition, these state-components do not depend on the state/action of agent k at time t, and instead depend only on $\hat{\pi}_{-k}$. We take advantage of this fact and modify the Q-MDP heuristic as our POMDP solution method.[4] To this purpose we introduce the concept of *generalized α-vectors* for Dec-SIMDPs. Due to space limitations, we omit some of the details involved in the derivation of these vectors as well as the analysis of its properties. Instead, we refer to [13] for further details.

Let us denote by $\mathscr{X}_I$ the set of all (joint) states in interaction areas, and define a *generalized α-vector* for agent k, α_k, recursively as follows:

$$\alpha_k(x) = r_{\pi_{-k}}(x,a_k) + \gamma \sum_{y\in\mathscr{X}_I} \mathsf{P}_{\pi_{-k}}(x,a_k,y) \max_{u_k} \alpha_k(y,u_k) + \gamma \max_{u_k} \sum_{y\notin\mathscr{X}_I} \mathsf{P}_{\pi_{-k}}(x,a_k,y)\alpha_k(y,u_k), \tag{5}$$

where

$$r_{\pi_{-k}}(x,a_k) = \sum_{a_{-k}} \pi_{-k}(x_{-k},a_{-k}) r\big(x,(a_{-k},a_k)\big)$$
$$\mathsf{P}_{\pi_{-k}}(x,a_k,y) = \sum_{a_{-k}} \pi_{-k}(x_{-k},a_{-k}) \mathsf{P}\big(x,(a_{-k},a_k),y\big).$$

The generalized α-vector α_k is the fixed-point of the expression (5) and are well-defined and unique. Furthermore, they can be computed iteratively using a dynamic-programming-like approach that, essentially, iterates through the recursion in (5). It is also possible to show that α_k corresponds to the optimal Q-function of an associated MDP whose dimension grows linearly with the dimension of the original Dec-SIMDP. Recalling that the decision process for agent k can be modeled using a standard POMDP, we adopt the approximation

$$Q^*(x_k,\mathbf{b}_{-k},a_k) \approx \sum_{x_{-k}} \mathbf{b}_{x_{-k}} \alpha_k(x,a_k). \tag{6}$$

This solution can now be used to choose the actions of agent k by maximizing the above expression.

[4] In the continuation, and to avoid unnecessarily complicating the presentation, we focus on a 2-agent scenario. The development presented extends trivially to more than two agents at the cost of more cumbersome expressions.

5 Results

In this section we describe the results obtained from applying both MPSI and LAPSI to a range of problems of different dimensions, and analyze the performance of our methods in each of the test scenarios. We compare the performance of both MPSI and LAPSI to that of the optimal fully observable MMDP policy and that of the IDMG algorithm from [17]. In the IDMG algorithm, each agent k in a Dec-SIMDP $\left(\{\mathscr{M}_k, k=1,\ldots,N\}, \{(\mathscr{X}_i^I, \mathscr{M}_i^I), i=1,\ldots,M\}\right)$ follows the optimal individual policy π_k for the MDP $\mathscr{M}_k$ outside the interaction areas. In the interaction areas, the agents engage in a sequence of local matrix games in which they jointly adopt the equilibrium policy.

We used several robot navigation scenarios to test our algorithms (see Fig. 2), since the Dec-SIMDP model is particularly appealing for modeling multi-robot problems. Furthermore, in this class of problems, the results can be easily visualized and interpreted. In each of the test scenarios, each robot in a set of two/four robots must reach one specific state. In the smaller environments (Maps 1 through 4), the goal state is marked with a number, corresponding to the number of the robot. The cells with a boxed number correspond to the initial states for the robots. In the larger environments, the goal for each robot is marked with a cross, $\times$, and the robots each depart from the other robot's goal state, in an attempt to increase the possibility of interaction. Each robot has 4 possible actions that move the robot in the

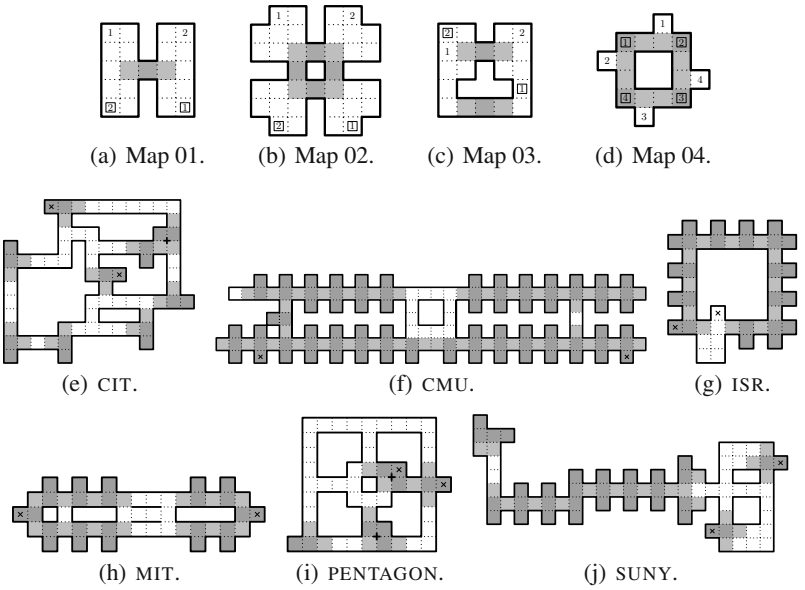


Fig. 2 Environments used in the experiments. The dark gray areas correspond to interaction states and the light gray areas to the corresponding interaction areas. We refer to the main text for details.

corresponding direction with probability 0.8 and fail with probability 0.2. The shaded regions correspond to interaction areas, inside of which the darker cells correspond to interaction states, in which the robots get a penalty of -20 if they stand in the same cell simultaneously. Also, in these interaction states, the rate of action failure is increased to 0.4.[5] Upon reaching the corresponding goal, each agent receives a reward of $+1$ and its position is reset to the initial state. The dimension of the state-space for the different Dec-MDPs is summarized in Table 1.

Table 1 Total discounted reward for each of the four different algorithms in each of the test-scenarios. The results are averaged over 1,000 independent Monte-Carlo runs. Entries in **bold** correspond to guaranteed optimal performance. Entries in *italic* in the same line are not statistically different.

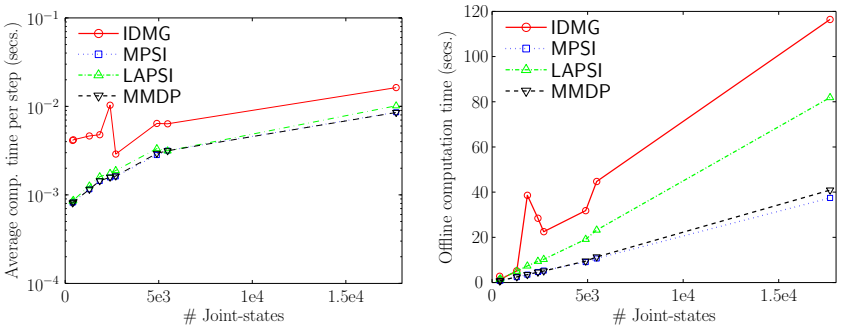
Environment	# States	IDMG	MPSI	LAPSI	MMDP
		Disc. Rew.	Disc. Rew.	Disc. Rew.	Disc. Rew.
Map 1	441	*12.035*	11.130	*11.992*	**12.588**
Map 2	1,296	10.672	10.159	**10.947**	**11.069**
Map 3	400	*13.722*	13.249	*13.701*	**14.380**
Map 4	65,536	–	15.384	15.564	**16.447**
CIT	4,900	**11.178**	**11.105**	**11.126**	**11.151**
CMU	17,689	**2.839**	2.688	**2.824**	**2.906**
ISR	1,849	14.168	*13.947*	*13.997*	**14.335**
MIT	2,401	**6.663**	**6.641**	**6.648**	**6.681**
PENTAGON	2,704	*16.031*	15.162	*15.976*	**16.312**
SUNY	5,476	**11.161**	**11.130**	**11.139**	**11.110**

For each of the different scenarios in Fig. 2, we ran the four algorithms above and then tested the computed policy for 1,000 independent trials of 100 steps each, in the smaller environments, and 250 time-steps each, in the larger environments. The obtained performance in terms of total discounted reward can be found in Table 1.

The LAPSI algorithm performed very close to the optimal MMDP policy in all environments, in spite of the significant difference in terms of state information available to both methods. Also, in most scenarios, LAPSI and IDMG performed similarly. The only exceptions are Map 2, where LAPSI outperformed IDMG, and ISR, where IDMG outperformed LAPSI. Interestingly, however, the difference in terms of time-to-goal in the ISR environment is not significant. In any case, our results agree with previous ones that showed that IDMG attained close-to-optimal performance in most such scenarios [17]. Another interesting observation is that MPSI typically performed worse than the other methods. As pointed out before, since an agent in MPSI considers the other agents to be selfish and disregard miscoordinations (each is focused only on its individual goal), it is expected that the agent following MPSI is more "cautious" and takes longer time to reach the goal.

Given the similar performance of IDMG and LAPSI, one may question the advantage of adopting the latter over the former. There are at least two clear

[5] Both the penalty and the increased action failure rate imply that there is both reward and transition dependence in the interaction areas.



(a) Online (b) Offline

Fig. 3 Computation time for the different algorithms as a function of the problem dimension

advantages. First of all, since the IDMG method requires the computation of several equilibria both during off-line planning and during on-line execution, the computational complexity of the IDMG algorithm may quickly become prohibitive, in scenarios with large action spaces and/or with many interaction areas. To assess whether this is indeed so, we compared the computational effort of our methods with that of IDMG, both in terms of the average off-line computation time and the on-line computation time (see Fig. 3). Clearly, both MPSI and LAPSI are significantly more efficient than the IDMG algorithm, according to any of the two performance metrics. It is also interesting to note how the average computation times evolve with the dimension of the problem.

The second advantage of LAPSI becomes evident by noting that the IDMG method is, by construction, unable to consider future interactions when planning for the action in a non-interaction area. In this sense, the IDMG algorithm is "myopic" to interactions and only handles these as it reaches an interaction area. This can have a negative impact on the performance of the method, as illustrated in the final test scenario (Fig. 4). In this environment, and ignoring the interaction, Robot 1 can reach its goal by using either of the narrow pathways, since both trajectories have the same length. However, Robot 2 should use the upper pathway, since it is significantly faster than using the lower pathway.

By using the IDMG algorithm, Robot 2 goes for the upper pathway while Robot 1 chooses randomly between the two. For concreteness, let's suppose that Robot 1 chooses to go for the upper pathway. In this case, according to the IDMG algorithm,

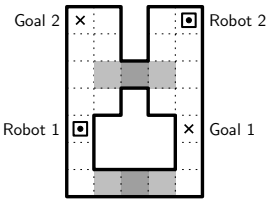


Fig. 4 Example scenario where avoiding the interaction may be beneficial

both robots reach the interaction area simultaneously and Robot 1 must move out of the way for Robot 2 to go on. This means that, in total, the two robots take a mean time of 9 steps to reach the goal. If, instead, Robot 1 takes the lower pathway, the two robots will reach their goal states in 8 steps. Since the IDMG algorithm chooses between the two randomly – or, at least, has no way to differentiate between the two – the average time to the goal is 8.5 time-steps. We ran $1,000$ independent trials using the IDMG algorithm in this scenario and, indeed, obtained an average of 8.485 steps to goal, with a standard deviation of 0.5. Clearly, it seems possible to do better in this scenario by considering more convenient to use the lower pathway.

For comparison purposes, we also ran $1,000$ independent trials using the LAPSI algorithm in this same scenario. Out of $1,000$ trials, Robot 1 *always* picked the lower pathway. As expected, the group had an average time-to-goal of 8 time-steps with a variance of 0. Notice that this difference could be made arbitrarily large by increasing the "narrow doorway" to an arbitrary number of states, thus causing an arbitrarily large delay. As such, in scenarios such as the one above, where interactions should be considered even outside interaction areas, our methods present a clear advantage over the IDMG algorithm.

6 Conclusion

As mentioned in Section 1, Dec-SIMDPs are particularly suited for modeling several multi-robot problems. On one hand, unlike models such as MMDPs, Dec-SIMDPs do not assume full joint state observability that, in a multi-robot scenario, is tantamount to having the robots perceive the state of the other robots at every step. In most settings, this would require the agents to flawlessly communicate in a continued manner, which is quite unrealistic. On the other hand, due to their physical limitations, robots are generally bound to interact locally and, when doing so, they are most likely in a position where communication is possible. Local interactions and communication are abstracted in the Dec-SIMDPs model in the notions of *interaction areas* – meaning that the interaction among robots is "local" and limited to these areas – and *observable interactions* – meaning that, when interacting, robots have access to joint state information, possible through communication. While a Dec-SIMDP is a subclass of Dec-MDPs – and hence any problem modeled as a Dec-SIMDP can be modeled as a Dec-MDP, – the form of interaction explicitly abstracted in Dec-SIMDPs is particularly suited for multi-robot scenarios and allows algorithms such as IDMG, LAPSI and MPSI to exploit them for efficient planning.

Concerning the methods, both the LAPSI and the MPSI algorithm allow each agent to track the other agents in the environment using a belief vector that is then used to choose the actions. The difference between the two algorithms lies in the assumed policy for the other agents. In MPSI and LAPSI, these "modeling strategies" are used to abstract the decision process of each agent into a single-agent decision process (a POMDP). Although we proposed a solution technique based on the generalized α-vectors, the same principle can be used with any other POMDP solver.

Also, the ability that both MPSI and LAPSI have to track the other agent allows the planning process to take into consideration the possibility of future interaction. This, as seen in the example in Fig. 4, is an important property of the method that overcomes one important limitation of the IDMG algorithm.

It is also interesting to notice that the generalized α-vectors used in MPSI and LAPSI can be interpreted in terms of an associated MDP. By comparing the optimal policy in this MMDP and the optimal policies from the individual MDPs it should be possible to pinpoint those joint-states in which the joint action significantly differs from the one prescribed by the individual MDPs and in which the actions for each agent greatly depend on the state of the other agents. This provides one recipe for choosing the interaction states as those in which individual state-information is not sufficient to determine the best action. In [16] a similar approach is used to implement decentralized execution of a jointly optimal policy.

Finally, several open questions remain to be explored. One is concerned with the worst-case complexity of Dec-SIMDP. Is a Dec-SIMDP reducible to any of the simpler Dec-MDP subclasses for which complexity results are known? Another interesting question arised from the observation that, as a particular case of a Dec-(PO)MDP, exact Dec-POMDP methods available (*e.g.*, [4]) can be applied to solve Dec-SIMDPs. It remains an open question whether it is possible to construct a more specific *optimal* solution method that actually leverages the particular structure of Dec-SIMDPs, or whether this structure actually brings a benefit in terms of computational complexity.

Acknowledgements. This research was partially sponsored by the Portuguese Fundação para a Ciência e a Tecnologia (INESC-ID multiannual funding) through the PIDDAC Program funds and under the CMU-Portugal Program, and by the Information and Communications Technologies Institute (ICTI), www.icti.cmu.edu.

References

[1] Allen, M., Zilberstein, S.: Complexity of Decentralized Control: Special Cases. In: Adv. Neural. Information Proc. Systems, pp. 19–27 (2009)
[2] Becker, R., Zilberstein, S., Lesser, V., Goldman, C.: Solving transition independent decentralized Markov decision processes. J. Artif. Intell. Res. 22, 423–455 (2004)
[3] Bernstein, D., Givan, R., Immerman, N., Zilberstein, S.: The complexity of decentralized control of Markov decision processes. Math. Oper. Res. 27(4), 819–840 (2002)
[4] Bernstein, D., Amato, C., Zilberstein, S.: Policy iteration for decentralized control of Markov decision processes. J. Artif. Intell. Res. 34, 89–132 (2009)
[5] Gerkey, B., Matarić, M.: Sold!: Auction methods for multirobot coordination. IEEE T. Robot. Autom. 18(5), 758–768 (2002)
[6] Ghavamzadeh, M., Mahadevan, S., Makar, R.: Hierarchical multiagent reinforcement learning. J. Auton. Agent Multiag. 13(2), 197–229 (2006)
[7] Goldman, C., Zilberstein, S.: Decentralized control of cooperative systems: Categorization and complexity analysis. J. Artif. Intell. Res. 22, 143–174 (2004)
[8] Guestrin, C., Koller, D., Parr, R.: Multiagent planning with factored MDPs. In: Adv. Neural Information Proc. Systems, pp. 1523–1530 (2001)

[9] Kearns, M., Littman, M., Singh, S.: Graphical models for game theory. In: Conf. Uncert. Artif. Intell., pp. 253–260 (2001)
[10] Kok, J., Hoen, P., Bakker, B., Vlassis, N.: Utile coordination: Learning interdependencies among cooperative agents. In: IEEE Symp. Comput. Intell. Games, pp. 61–68 (2005)
[11] Littman, M., Cassandra, A., Kaelbling, L.: Learning policies for partially observable environments: Scaling up. In: Int. Conf. Mach. Learn., pp. 362–370 (1995)
[12] Madani, O., Hanks, S., Condon, A.: On the undecidability of probabilistic planning in infinite-horizon partially observable Markov decision problems. In: AAAI Conf. Artif. Intell., pp. 541–548 (1999)
[13] Melo, F., Veloso, M.: Local Multiagent Coordination in Decentralized MDPs with Sparse Interactions. Tech. Report CMU-CS-10-133, CS Dep., Carnegie Mellon Univ. (2010)
[14] Mostafa, H., Lesser, V.: Offline planning for communication by exploiting structured interactions in decentralized MDPs. Tech Rep. TR 2009-020, CS Dep., Univ. Massachusetts (2009)
[15] Parker, L.: ALLIANCE: An architecture for fault-tolerant multirobot cooperation. IEEE T. Robot. Autom. 14(2), 220–240 (1998)
[16] Roth, M., Simmons, R., Veloso, M.: Exploiting factored representations for decentralized execution in multiagent teams. In: Int. Conf. Auton. Agent Multiag., pp. 469–475 (2007)
[17] Spaan, M., Melo, F.: Interaction-driven Markov games for decentralized multiagent planning under uncertainty. In: Int. Conf. Auton. Agent Multiag., pp. 525–532 (2008)
[18] Stone, P.: Layered learning in multiagent systems. PhD thesis, Carnegie Mellon Univ. (1998)
[19] Varakantham, P., Kwak, J., Taylor, M., Marecki, J., Scerri, P., Tambe, M.: Exploiting coordination locales in distributed POMDPs via social model shaping. In: Int. Conf. Autom. Plan Scheduling, pp. 313–320 (2009)
[20] Xin Jiang, A., Leyton-Brown, K., Bhat, N.: Action-graph games. Tech Rep. TR-2008-13, Univ. British Columbia (2008)

A Note on the Consensus Protocol with Some Applications to Agent Orbit Pattern Generation

Panagiotis Tsiotras and Luis Ignacio Reyes Castro

Abstract. We propose an extension to the standard feedback control for consensus problems for multi-agent systems in the plane. The proposed extension allows for a richer class of trajectories including periodic and quasi-periodic solutions, as well as agreement to consensus states outside the convex hull of the initial positions of the agents. We investigate in great detail the special case of three agents, which results in non-trivial geometric patterns described by ellipsoidal, epitrochoidal and hypotrochoidal curves.

1 A Generalized Consensus Protocol

Consensus problems have been originally used in distributed computing and management science and, most recently, have found extensive application in multi-agent, mobile network problems [12, 16]. In this paper we propose a generalization of the standard consensus algorithm which has been used extensively in the literature [13, 3, 11]. The proposed extension of the standard consensus protocol leads to the following advantages: first, it can be used to achieve consensus at points that do not necessarily belong to the convex hull of the initial conditions. This may be beneficial in case of obstacle avoidance or as part of cooperative deception strategies. Second, as shown in the second part of the paper, it can be utilized to generate intricate geometrical patterns of the agent paths. These paths can be useful for coordinated, distributed surveillance and monitoring applications.

Coordinated algorithms for network formations have appeared previously, for example, in [14, 15, 7] as well as in the work of Leonard and Sepulchre [9, 17, 18]. Therein the authors make use of geometric information to achieve specific formation patterns. The control laws are at the acceleration level (e.g., [7, 18]), often derived

Panagiotis Tsiotras · Luis I. Reyes Castro
School of Aerospace Engineering, Georgia Institute of Technology,
Atlanta, GA 30332-0105, USA
e-mail: {tsiotras,nacho_reyes}@gatech.edu

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 345–358.
springerlink.com

from potential-like functions. Typically, these works focus on a uni-directional ring communication topology, assuming identical control laws for each agent, such as in the case of cyclic pursuit. In [14], for instance, all-to-all communication and fixed ring topology is assumed for the graph resulting in a cyclic pursuit. The particular choice of the communication topology leads to a graph Laplacian which is a circulant matrix and the achievable formations are lines, circles or logarithmic spirals, similarly to the results of [9, 17, 18, 15].

The consensus control law proposed in the current paper cannot be readily derived from a scalar potential and its design is at the velocity level, similarly to the original consensus protocol. Depending on the gain matrices, the resulting paths may lead to more intricate *trochoidal* paths, as opposed to just straight lines, cycles and spirals. Using minimal assumptions we are thus able to generate geometric patterns of the agent trajectories that go beyond formation-type geometric models [20, 14, 9, 10, 15].

1.1 A New Consensus Protocol

Consider N agents in the plane, whose locations are given by the state variables $x_i \in \mathbb{R}^2$ for $i = 1,\dots,N$, satisfying the differential equations

$$\dot{x}_i = u_i, \qquad i = 1,\dots,N. \tag{1}$$

As usual, to this problem we associate a graph $\mathscr{G}$ that describes the communication topology between the agents. That is, $\mathscr{G}$ has N nodes and M edges (links), with each edge denoting knowledge of the relative position between the corresponding agents. Two nodes are neighbors in the graph $\mathscr{G}$ (hence connected by an edge) if and only if they can communicate with each other. Throughout the paper it will be assumed that the communication topology is fixed, that is, the neighbors of each node do not change as the agents move.

Define the incidence matrix $D \in \mathbb{R}^{N\times M}$ with elements

$$d_{ij} = \begin{cases} +1, & \text{if } i\text{th node is the head of } j\text{th edge,} \\ -1, & \text{if } i\text{th node is the tail of } j\text{th edge,} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

To each edge we assign the difference (error) variable

$$z_k = \sum_{\ell=1}^{N} d_{\ell k} x_\ell = \begin{cases} x_i - x_j, & \text{if } i \text{ is the head,} \\ x_j - x_i, & \text{if } j \text{ is the head,} \end{cases} \tag{3}$$

where $z_k \in \mathbb{R}^2$ for $k = 1,\dots,M$. If the columns of D are linearly independent, that is, if the graph does not contain cycles, then the error variables z_k are linearly independent vectors [1]. Note also that the graph is connected if and only if $\operatorname{rank} D = N-1$ [13, 4]. Introducing the stack vector $x = \begin{bmatrix} x_1^\mathsf{T} & \cdots & x_N^\mathsf{T} \end{bmatrix}^\mathsf{T} \in \mathbb{R}^{2N}$, the state equations (1) can be written compactly as

$$\dot{x} = u, \tag{4}$$

where $u = \begin{bmatrix} u_1^\intercal & \cdots & u_N^\intercal \end{bmatrix}^\intercal \in \mathbb{R}^{2N}$. We propose the following control law[1] for (1)

$$u_i = -\gamma_i \sum_{k=1}^{M} d_{ik} z_k + \beta_i \sum_{k=1}^{M} d_{ik} p_k, \quad i = 1, \ldots, N, \tag{5}$$

where $p_k \in \mathbb{R}^2$ such that $p_k^\intercal z_k = 0$, where $\gamma_i > 0$ and $\beta_i \in \mathbb{R}$. For instance, let $p_k \stackrel{\text{def}}{=} Sz_k$, $(k = 1, \ldots, M)$, where S is the skew symmetric matrix

$$S = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}. \tag{6}$$

Letting the stack vector $p = \begin{bmatrix} p_1^\intercal & \cdots & p_M^\intercal \end{bmatrix}^\intercal \in \mathbb{R}^{2M}$ yields, $p = (I_M \otimes S)z$, where $z = \begin{bmatrix} z_1^\intercal & \cdots & z_M^\intercal \end{bmatrix}^\intercal \in \mathbb{R}^{2M}$. The composite control law (5) then takes the form

$$u = -(\Gamma D \otimes I_2)z + (BD \otimes I_2)p = -(\Gamma D \otimes I_2)z + (BD \otimes S)z, \tag{7}$$

where $\Gamma = \operatorname{diag}(\gamma_1, \ldots, \gamma_N)$ and $B = \operatorname{diag}(\beta_1, \ldots, \beta_N)$. Note that the standard consensus algorithm results as a special case of (7) where $B = 0$.

Remark 1. The basic idea behind the control law (5) is the use of additional geometric information, inferred from the relative distance between the agent and its neighbors. Specifically, the second term in (5) is proportional to the direction which is *perpendicular* to the relative distance between the agent and its neighbor(s). In (7) this information is encoded via the multiplication of the error state with the skew-symmetric matrix S. This new skew-symmetric term provides additional flexibility in terms of the achievable final rendezvous points, as well as in terms of the resulting trajectories followed by the agents.

Remark 2. The proposed control law (7) has the same form as the one given in [1, Eq. (16)]. However, since the second term is (7) is not the gradient of a scalar function, it does not come from a potential, and hence (7) is more general than the family of control laws of [1]. The absence of a scalar potential is owing to the skew-symmetric term in (7) which introduces a circulation. In this sense, the control law (7) is akin to the gyroscopic control laws proposed in the robotics literature [6, 21, 19].

1.2 Consensus and Final Rendezvous Position

From (3) it can be easily shown that the error vector z can be written compactly as follows

[1] The alternative control law $u_i = -\sum_{k=1}^{M} \gamma_k d_{ik} z_k + \sum_{k=1}^{M} \beta_k d_{ik} p_k$ which weights each edge separately could have been used in lieu of (5). The results of the paper remain essentially the same for the latter choice as well.

$$z = (D^{\mathsf{T}} \otimes I_2)x. \tag{8}$$

Hence, equations (4) and (7) yield

$$\begin{aligned} \dot{z} = (D^{\mathsf{T}} \otimes I_2)u &= -(D^{\mathsf{T}} \otimes I_2)(\Gamma D \otimes I_2)z + (D^{\mathsf{T}} \otimes I_2)(BD \otimes S)z \\ &= -\big((D^{\mathsf{T}}\Gamma D) \otimes I_2 - (D^{\mathsf{T}}BD) \otimes S\big)z. \end{aligned} \tag{9}$$

It follows that stability is determined by the spectral properties of the matrix in (9).

Lemma 1 (Fact 5.12.3 of [2]). *Given two matrices $A, B \in \mathbb{R}^{n\times n}$, the eigenvalues of $A+B$ satisfy the inequality*

$$\frac{1}{2}\lambda_{\min}(A^{\mathsf{T}}+A) + \frac{1}{2}\lambda_{\min}(B^{\mathsf{T}}+B) \le \operatorname{Re}\lambda \le \frac{1}{2}\lambda_{\max}(A^{\mathsf{T}}+A) + \frac{1}{2}\lambda_{\max}(B^{\mathsf{T}}+B), \tag{10}$$

where λ is an eigenvalue of $A+B$.

Note that in case A is symmetric and B is skew symmetric, inequality (10) yields $\lambda_{\min}(A) \le \operatorname{Re}\lambda \le \lambda_{\max}(A)$.

Proposition 1. *Assume that $\mathscr{G}$ is a connected, and assume that Γ is a positive definite diagonal matrix. Then all solutions of (9) satisfy $\lim_{t\to\infty} z(t) = 0$.*

Proof. We first consider the case when the graph $\mathscr{G}$ has no cycles, that is, D is full column rank. It is then clear that the matrix $-(D^{\mathsf{T}}\Gamma D)\otimes I_2$ is negative definite if Γ is positive definite. Furthermore, it can be easily shown that the matrix $(D^{\mathsf{T}}BD)\otimes S$ is skew-symmetric[2]. From Lemma 1 it follows that all the eigenvalues of $-(D^{\mathsf{T}}\Gamma D)\otimes I_2 + (D^{\mathsf{T}}BD)\otimes S$ are in the open left-half of the complex plane, and the result follows.

If $\mathscr{G}$ has cycles, we can proceed similarly to the approach outlined in [11, pp. 78–82]. Specifically, the incidence matrix in this case can be factored as $D = \begin{bmatrix} D_\tau & D_c \end{bmatrix}$ where D_τ is the incidence matrix corresponding to the acyclic subgraph (spanning tree) of $\mathscr{G}$ and D_c is the incidence matrix corresponding to the remaining edges not in the tree, i.e., the cycle edges. Furthermore, D_τ is full column rank, and the columns of D_c are linearly dependent on the columns of D_τ, that is, there exist a matrix T such that $D_c = D_\tau T$. Subsequently, $D = D_\tau R$, where $R \stackrel{\text{def}}{=} \begin{bmatrix} I & T \end{bmatrix}$. The edge states (8) are partitioned conformably with D as $z_\tau = (D_\tau^{\mathsf{T}} \otimes I_2)x$ and $z_c = (D_c^{\mathsf{T}} \otimes I_2)x = (T^{\mathsf{T}} \otimes I_2)z_\tau$. Since $z = (R^{\mathsf{T}} \otimes I_2)z_\tau$, and using (8) and (7), it can be easily shown that

$$\begin{aligned} \dot{z}_\tau &= -\big((D_\tau^{\mathsf{T}}\Gamma D_\tau RR^{\mathsf{T}}) \otimes I_2\big)z_\tau + \big((D_\tau^{\mathsf{T}}BD_\tau RR^{\mathsf{T}}) \otimes S\big)z_\tau \\ &= -\big((D_\tau^{\mathsf{T}}\Gamma D_\tau) \otimes I_2 - (D_\tau^{\mathsf{T}}BD_\tau) \otimes S\big)(RR^{\mathsf{T}} \otimes I_2)z_\tau. \end{aligned} \tag{11}$$

Since $RR^{\mathsf{T}} = I + TT^{\mathsf{T}}$, it follows that $RR^{\mathsf{T}} \otimes I_2$ is positive definite. Hence, $-\big((D_\tau^{\mathsf{T}}\Gamma D_\tau) \otimes I_2 - (D_\tau^{\mathsf{T}}BD_\tau) \otimes S\big)(RR^{\mathsf{T}} \otimes I_2)$ has no zero eigenvalues if and only if the matrix $-(D_\tau^{\mathsf{T}}\Gamma D_\tau) \otimes I_2 + (D_\tau^{\mathsf{T}}BD_\tau) \otimes S$ has no zero eigenvalues. Using Lemma 1, it can be shown that all the eigenvalues of the latter matrix are in the open left-half of the complex plane (and hence are all nonzero). To complete the proof, note that all (necessarily nonzero) eigenvalues of the matrix $-\big((D_\tau^{\mathsf{T}}\Gamma D_\tau) \otimes I_2 - (D_\tau^{\mathsf{T}}BD_\tau) \otimes S\big)(RR^{\mathsf{T}} \otimes I_2)$

[2] Recall that the transpose distributes over the Kronecker product, that is, $(A\otimes B)^{\mathsf{T}} = A^{\mathsf{T}} \otimes B^{\mathsf{T}}$.

are the same as the nonzero eigenvalues of the matrix[3] $(R^{\mathsf{T}} \otimes I_2)\big(-(D_\tau^{\mathsf{T}} \Gamma D_\tau) \otimes I_2 + (D_\tau^{\mathsf{T}} B D_\tau) \otimes S\big)(R \otimes I_2) = -(R^{\mathsf{T}} D_\tau^{\mathsf{T}} \Gamma D_\tau R) \otimes I_2 + (R^{\mathsf{T}} D_\tau^{\mathsf{T}} B D_\tau R) \otimes S$. Using the same argument as in the acyclic case, the nonzero eigenvalues of the latter matrix are in the open left-half of the complex plane. Thus, $\lim_{t\to\infty} z_\tau(t) = 0$. Furthermore, $\lim_{t\to\infty} z_c(t) = \lim_{t\to\infty}(T^{\mathsf{T}} \otimes I_2) z_\tau(t) = 0$, and the result follows. □

In order to compute the final consensus value, first note that the differential equation for x is given by

$$\begin{aligned}\dot{x} &= -(\Gamma D \otimes I_2)(D^{\mathsf{T}} \otimes I_2)x + (BD \otimes S)(D^{\mathsf{T}} \otimes I_2)x \\ &= -\big((\Gamma D D^{\mathsf{T}}) \otimes I_2 - (BDD^{\mathsf{T}}) \otimes S\big)x \\ &= -\big((\Gamma L) \otimes I_2 - (BL) \otimes S\big)x, \end{aligned} \tag{12}$$

where $L \stackrel{\text{def}}{=} DD^{\mathsf{T}} \in \mathbb{R}^{N\times N}$ is the *graph Laplacian* [11]. From now on, we will assume that L always corresponds to a connected graph. We have the following lemma.

Lemma 2. *Let Γ and B be diagonal matrices as before, and assume that $\Gamma > 0$ (i.e., it is positive definite). Then* $\dim\big[\mathscr{R}^{\perp}\big((\Gamma L) \otimes I_2 - (BL) \otimes S\big)\big] = 2$.

Proof. It suffices to show[4] that $\operatorname{null}[((\Gamma L) \otimes I_2 - (BL) \otimes S)^{\mathsf{T}}] = 2$. Notice now that $\operatorname{null}[(L\Gamma) \otimes I_2 + (LB) \otimes S)] = \operatorname{null}[(L \otimes I_2)(\Gamma \otimes I_2 + B \otimes S)]$. The matrix $\Gamma \otimes I_2 + B \otimes S$ is always invertible if $\Gamma > 0$. The result now follows from the fact that, for a connected graph, the Laplacian has a single eigenvalue at the origin [4], and hence $\operatorname{null}[L \otimes I_2] = 2$. □

Let $\mathbf{1}_N \stackrel{\text{def}}{=} (1, 1, \ldots, 1)^{\mathsf{T}} \in \mathbb{R}^N$ denote the N-dimensional column vector of ones, and recall that $L\mathbf{1}_N = 0$ [11, 4]. For any $v \in \mathbb{R}^2$ we have that

$$\big((\Gamma L) \otimes I_2 - (BL) \otimes S\big)(\mathbf{1}_N \otimes v) = (\Gamma L \mathbf{1}_N) \otimes v - (BL\mathbf{1}_N) \otimes (Sv) = 0. \tag{13}$$

Since $\operatorname{null}[(\Gamma L) \otimes I_2 - (BL) \otimes S] = 2$ it follows that $\mathbf{1}_N \otimes v$ spans to the null space of the matrix in (12). It follows that the equilibrium point $\bar{x}_\infty$ of (12) satisfies the condition $\bar{x}_\infty \stackrel{\text{def}}{=} \lim_{t\to\infty} x(t) = \mathbf{1}_N \otimes x_\infty$ for some $x_\infty \in \mathbb{R}^2$, equivalently, $\lim_{t\to\infty} x_1(t) = \lim_{t\to\infty} x_2(t) = \cdots = \lim_{t\to\infty} x_N(t) = x_\infty$.

Lemma 3. *Let $\Theta = Q\Sigma$ where $\Sigma = \Sigma^{\mathsf{T}} \in \mathbb{R}^n$ and $Q \in \mathbb{R}^n$ an invertible matrix, such that $x^{\mathsf{T}} Q x \neq 0$ for all nonzero $x \in \mathbb{R}^n$. Then $\mathscr{N}(\Theta) \cap \mathscr{R}(\Theta) = \{0\}$.*

Proof. Assume, on the contrary, that there exist $x \neq 0$ such that $x \in \mathscr{N}(\Theta) \cap \mathscr{R}(\Theta)$. From $x \in \mathscr{N}(\Theta)$ it follows that $\Theta x = Q\Sigma x = 0$ and since Q is invertible, $\Sigma x = 0$. Furthermore, since $x \in \mathscr{R}(\Theta)$ it follows that there exist $y \in \mathbb{R}^n$ such that $x = \Theta y = Q\Sigma y$. Hence, $\Sigma x = \Sigma Q \Sigma y = 0$ and thus, $y^{\mathsf{T}} \Sigma^{\mathsf{T}} Q \Sigma y = 0$. It follows that $\Sigma y = 0$ or that $x = Q\Sigma y = 0$, a contradiction. □

[3] Recall that $\det(\lambda I - ABB^{\mathsf{T}}) = \det(\lambda I - B^{\mathsf{T}} A B)$ for any two matrices A and B of compatible dimensions.

[4] Here $\operatorname{null} A$ denotes the nullity of A, i.e., the dimension of the null space of the matrix A, that is, $\operatorname{null} A \stackrel{\text{def}}{=} \dim[\mathscr{N}(A)]$.

The coordinates of the final consensus point $x_\infty = [\mathsf{x}_\infty \ \mathsf{y}_\infty]^\mathsf{T} \in \mathbb{R}^2$ can be explicitly computed using the following proposition.

Proposition 2. *Let* $v_1, v_2 \in \mathbb{R}^{2N}$ *be such that* $\operatorname{span}\{v_1, v_2\} = \mathscr{R}^\perp\big((\Gamma L)\otimes I_2 - (BL)\otimes S\big)$. *The final rendezvous point is given by*

$$x_\infty = \begin{bmatrix} \mathsf{x}_\infty \\ \mathsf{y}_\infty \end{bmatrix} = \begin{bmatrix} v_1^\mathsf{T}(\mathbf{1}_N \otimes I_2) \\ v_2^\mathsf{T}(\mathbf{1}_N \otimes I_2) \end{bmatrix}^{-1} \begin{bmatrix} v_1^\mathsf{T} x(0) \\ v_2^\mathsf{T} x(0) \end{bmatrix}. \tag{14}$$

Proof. From Lemma 2 there exist linearly independent vectors $v_1, v_2 \in \mathscr{R}^\perp\big((\Gamma L)\otimes I_2 - (BL)\otimes S\big)$ such that $v_i^\mathsf{T}\dot{x} = -v_i^\mathsf{T}\big((\Gamma L)\otimes I_2 - (BL)\otimes S\big)x = 0$, $(i = 1,2)$. Consequently, $v_i^\mathsf{T} x(t) = v^\mathsf{T} x(0)$ for all $t \geq 0$. In particular, we have that $v_i^\mathsf{T}(\mathbf{1}_N \otimes x_\infty) = v_i^\mathsf{T}(\mathbf{1}_N \otimes I_2)x_\infty = v_i^\mathsf{T} x(0)$ $(i = 1,2)$. It follows that

$$\begin{bmatrix} v_1^\mathsf{T} x(0) \\ v_2^\mathsf{T} x(0) \end{bmatrix} = \begin{bmatrix} v_1^\mathsf{T}(\mathbf{1}_N \otimes I_2) \\ v_2^\mathsf{T}(\mathbf{1}_N \otimes I_2) \end{bmatrix} \begin{bmatrix} \mathsf{x}_\infty \\ \mathsf{y}_\infty \end{bmatrix} = \left(\begin{bmatrix} v_1^\mathsf{T} \\ v_2^\mathsf{T} \end{bmatrix} (\mathbf{1}_N \otimes I_2) \right) \begin{bmatrix} \mathsf{x}_\infty \\ \mathsf{y}_\infty \end{bmatrix}. \tag{15}$$

Since the vectors v_1 and v_2 are linearly independent, $\operatorname{rank}[v_1\, v_2] = 2$. Furthermore, $\operatorname{rank}(\mathbf{1}_N \otimes I_2) = (\operatorname{rank}\mathbf{1}_N)(\operatorname{rank} I_2) = 2$. Let now $\Theta \stackrel{\text{def}}{=} (\Gamma L)\otimes I_2 - (BL)\otimes S = (\Gamma \otimes I_2 - B\otimes S)(L\otimes I_2)$. From the definition of v_1 and v_2, it follows that $\mathscr{N}\big(\begin{bmatrix} v_1 \ v_2 \end{bmatrix}^\mathsf{T}\big) = \mathscr{R}(\Theta)$. An easy calculation also shows that $\mathscr{R}(\mathbf{1}_N \otimes I_2) = \mathscr{N}(\Theta)$ (refer also to equation (13)). From Lemma 3 we have that $\mathscr{R}(\Theta) \cap \mathscr{N}(\Theta) = \{0\}$, equivalently, $\mathscr{N}\big(\begin{bmatrix} v_1 \ v_2 \end{bmatrix}^\mathsf{T}\big) \cap \mathscr{R}(\mathbf{1}_N \otimes I_2) = \{0\}$. Fact 2.10.14 in [2] yields that the 2×2 matrix in (15) has rank 2 and hence it is invertible. The result now follows directly from (15). □

2 Applications to Agent Orbit Design

In this section we investigate how several choices of the gain matrices Γ and B can generate specific patterns for the agent paths. Since we are mainly interested in periodic or quasi-periodic trajectories, we assume that $\Gamma = 0$. It follows that the closed loop system is given by

$$\dot{x} = ((BL)\otimes S)x. \tag{16}$$

It can be easily shown that the eigenvalues of BL are all real, hence the eigenvalues of the closed-loop matrix in (16) all lie on the imaginary axis. The structure of the corresponding state matrix in (16) (e.g., its eigenvalues and eigenvectors) can provide a great deal of information regarding the paths followed by the agents in the Cartesian coordinate frame, as well as the relative location of the agents on these paths (i.e., their relative phasing). For instance, one can ensure that the agent trajectories either form closed paths with given phasing, or they form a dense set of

trajectories, ensuring that almost every point in a given region will be visited at least once by one or more agents.

Remark 3. In [15] Ren introduced Cartesian coupling in the consensus control law using a multiplication of the Laplacian matrix by a rotation matrix; this is similar to the skew-symmetric matrix we use in (16). Nonetheless, additional constraints on the Laplacian matrix are needed in [15] to capture the richness of trajectories we can obtain with the approach proposed in the current paper.

2.1 Case Study: Three Agents Connected in a Path Graph

In order to keep the analysis manageable, and to be able to provide closed-form expressions, henceforth we will restrict the discussion to three agents in the plane, that is, we take $N = 3$. For simplicity, we will also assume the simplest agent interconnection topology, namely a path graph. The corresponding incidence matrix is given by

$$D = \begin{bmatrix} -1 & 0 \\ 1 & -1 \\ 0 & 1 \end{bmatrix}. \tag{17}$$

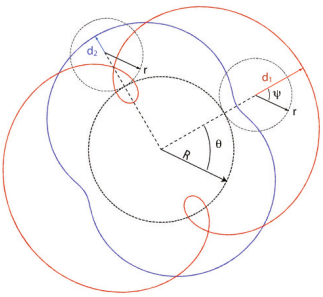
We are primarily interested in three types of closed curves: ellipses, epitrochoids, and hypotrochoids. Since the ellipses (and circles) have well-known parameterizations, next we will briefly review the main facts on epitrochoids and hypotrochoids. All these follow under the general class of *trochoid* curves, which includes cardioids, astroids, limaçons, and all polar coordinate roses [5].

An *epitrochoid* curve is generated by a point P attached at a radial distance d from the center of a circle of radius r, which is rolling without slipping around a circular track of radius R with angular velocity ω (see Fig. 1(a)). The distance d can be smaller, equal, or greater than the radius r of the rolling circle. In terms of Cartesian coordinates, an epitrochoid can be expressed as [8]

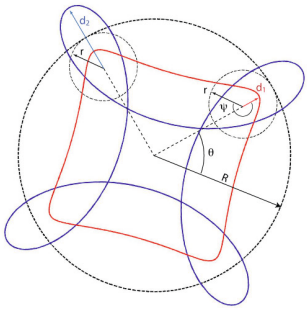
$$\mathsf{x}(\theta) = \mathsf{x}_c + (k+1)r\cos(\theta - \phi_A) - d\cos((k+1)\theta - \phi_B), \tag{18a}$$
$$\mathsf{y}(\theta) = \mathsf{y}_c + (k+1)r\sin(\theta - \phi_A) - d\sin((k+1)\theta - \phi_B), \tag{18b}$$

where ϕ_A and ϕ_B are constant angles, x_c and y_c are the coordinates of the center of the circular track of radius R and $k = R/r$. The angle θ denotes the angular position of the circle of radius r, given by $\theta = \omega t$. It can be shown that k is the number of points at which the agent is closest to the center of the circular track. For the purposes of this paper, we will henceforth refer to these points as *crests*. In the special case when $r = d$, the curve becomes an *epicycloid* with k cusps; at these points, the curve is not differentiable. Note that ellipsoidal paths correspond to the case when $k = 0$.

Another relevant curve of interest in this paper is the *hypotrochoid* [8], with parametric equations



(a) Epitrochoid curves. The blue curve has $d < r$, while the second one has $d > r$. Both epitrochoids have $R = 4$, $r = 2$ (hence $k = 2$).



(b) Hypotrochoid curves. The blue curve has $d > r$, while the second one has $d < r$. Both hypotrochoids have $R = 6$, $r = 1.5$ (hence $k = 4$).

Fig. 1 Representative examples of epitrochoids and hypotrochoid curves

$$\mathsf{x}(\theta) = \mathsf{x}_c + (k-1)r\cos(\theta - \phi_A) + d\cos((k-1)\theta - \phi_B), \tag{19a}$$
$$\mathsf{y}(\theta) = \mathsf{y}_c + (k-1)r\sin(\theta - \phi_A) - d\sin((k-1)\theta - \phi_B), \tag{19b}$$

where $k > 1$ with $k = R/r$ as before. The hypotrochoid can be reproduced by a point P attached at a distance d from the center of a circle of radius r, which rolls *inside* a circle of radius R. Again, the distance d can be smaller, equal, or greater than the radius r of the rolling circle; this radius, however, cannot exceed that of the circle R. Examples of hypotrochoids are shown in Fig. 1(b).

2.2 *Case I: Epitrochoidal Paths*

Consider the case when $\Gamma = 0$ and $B = \mathrm{diag}(\beta, \beta, \beta)$, where $\beta > 0$. Following (12), the solution of the closed loop system can be obtained easily as follows

$$x(t) = \left[\begin{array}{c} \frac{1}{2}\sqrt{c_1^2 + c_2^2}\cos(\beta t - \phi_{12}) + \frac{1}{6}\sqrt{c_3^2 + c_4^2}\cos(3\beta t - \phi_{34}) + \frac{1}{3}c_5 \\ -\frac{1}{2}\sqrt{c_1^2 + c_2^2}\sin(\beta t - \phi_{12}) - \frac{1}{6}\sqrt{c_3^2 + c_4^2}\sin(3\beta t - \phi_{34}) + \frac{1}{3}c_6 \\ \hdashline \frac{1}{3}\sqrt{c_3^2 + c_4^2}\cos(3\beta t - \phi_{34}) + \frac{1}{3}c_5 \\ -\frac{1}{3}\sqrt{c_3^2 + c_4^2}\sin(3\beta t - \phi_{34}) + \frac{1}{3}c_6 \\ \hdashline -\frac{1}{2}\sqrt{c_1^2 + c_2^2}\cos(\beta t - \phi_{12}) + \frac{1}{6}\sqrt{c_3^2 + c_4^2}\cos(3\beta t - \phi_{34}) + \frac{1}{3}c_5 \\ \frac{1}{2}\sqrt{c_1^2 + c_2^2}\sin(\beta t - \phi_{12}) - \frac{1}{6}\sqrt{c_3^2 + c_4^2}\sin(3\beta t - \phi_{34}) + \frac{1}{3}c_6 \end{array}\right], \tag{20}$$

where $x_i = [\mathsf{x}_i\ \mathsf{y}_i]^\intercal \in \mathbb{R}^2$ for $i = 1,2,3$ and where $c_1 = \mathsf{x}_1(0) - \mathsf{x}_3(0), c_2 = \mathsf{y}_1(0) - \mathsf{y}_3(0), c_3 = \mathsf{x}_1(0) - 2\mathsf{x}_2(0) + \mathsf{x}_3(0), c_4 = \mathsf{y}_1(0) - 2\mathsf{y}_2(0) + \mathsf{y}_3(0), c_5 = \mathsf{x}_1(0) + x_2(0) + \mathsf{x}_3(0), c_6 = \mathsf{y}_1(0) + \mathsf{y}_2(0) + \mathsf{y}_3(0)$ and $\phi_{12} = \arctan(c_2/c_1)$ and $\phi_{34} = \arctan(c_4/c_3)$.

Comparing the expressions for the $\mathsf{x}(\theta)$ and $\mathsf{y}(\theta)$ components of an epitrochoid in (18) to those in (20), the following observations can be made. First, for all agents, the center of their trajectories has coordinates $(\mathsf{x}_c, \mathsf{y}_c) = (\frac{1}{3}c_5, \frac{1}{3}c_6)$, which is the centroid of the initial positions of the agents. For agents no. 1 and no. 3, we have that $R_i + r_i = \frac{1}{2}(c_1^2 + c_2^2)^{\frac{1}{2}}$ and $d_i = \frac{1}{6}(c_3^2 + c_4^2)^{\frac{1}{2}}$. For the same two agents, $k = 2$, which implies that $R_i = 2r_i$ $(i = 1,3)$. In other words, for any given initial positions, the ratio of the radius of the rolling circle to the radius of the track is fixed. Moreover, from the definition of k, it becomes evident that these two agents will describe epitrochoids with only two crests. Furthermore, it can be easily shown that $r_i = \frac{1}{6}(c_1^2 + c_2^2)^{\frac{1}{2}}$ $(i = 1,3)$. The times at which agent no. 1 is closest and farthest from the center of its trajectory (its crests) can be computed from the solutions of the equation

$$\sin(2\beta t - \phi_{12} + \phi_{34}) = 0. \tag{21}$$

Also, by considering the distance of the radius of agent no. 2 from the origin, it can be easily shown that the points of closest approach for agents no. 1 and no. 3 differ by an angle $\pi/2$ with respect to the center of the circular track. This relationship, along with (21), can be employed to calculate the orientation of the curves with respect to an absolute Cartesian coordinate frame. Agent no. 2 describes a circle of radius $R_2 = \frac{1}{3}(c_3^2 + c_4^2)^{\frac{1}{2}}$ with frequency 3β.

To demonstrate these facts, consider the case $B = \mathrm{diag}(1,1,1)$ with initial positions $x_1(0) = (6,8)$, $x_2(0) = (-7,5)$, $x_3(0) = (5,-10)$. The center of the orbits is located at $(\mathsf{x}_c, \mathsf{y}_c) = (1.33, 1)$. The radii of the circular tracks for agents no. 1 and no. 3 are $R_1 = R_3 = 6.009$, and the radii of the rolling circles for agents no. 1 and no. 3 are $r_1 = r_3 = 3.005$. The radius of the circle described by agent no. 2 is $R_2 = 9.244$. After computing the phase angles ϕ_{12} and ϕ_{34}, and evaluating

$$\delta = \arctan\left(\frac{\mathsf{y}_1(\tau) - \mathsf{y}_c}{\mathsf{x}_1(\tau) - \mathsf{x}_c}\right), \tag{22}$$

where τ is the solution of (21), it is found that the angle by which the crests of the epitrochoids are inclined with respect to the x-axis is $\delta = -37.907^\circ$. The corresponding trajectories are shown in Fig. 2.

2.3 Case II: Ellipsoidal Paths

Consider now the case of the same system as before, but this time with the gains given as follows $B = \mathrm{diag}(\beta, -\beta, \beta)$, where $\beta > 0$. The solution of the closed-loop system in this case leads to

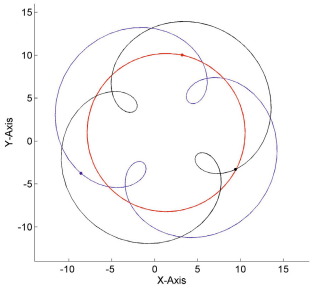


Fig. 2 Three agents displaying a circle and two epitrochoids when $B = \mathrm{diag}(1,1,1)$. The trajectory of agent no. 1 is shown in blue, the one of agent no. 2 is red, and the one of agent no. 3 in black.

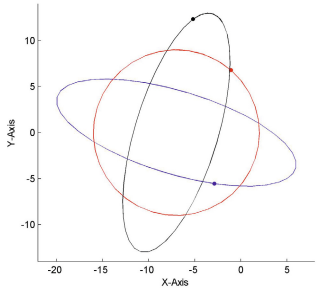


Fig. 3 Three agents displaying a circle and two ellipses when $B = \mathrm{diag}(1,-1,1)$. Again agent 1 is shown in blue, agent 2 in red, and agent 3 in black.

$$x(t) = \begin{bmatrix} c_2 \sin\beta t + c_3 \cos\beta t + c_5 \\ -c_1 \sin\beta t + c_4 \cos\beta t + c_6 \\ \hdashline (c_2 - c_4)\sin\beta t + (c_3 - c_1)\cos\beta t + c_5 \\ (c_3 - c_1)\sin\beta t + (c_4 - c_2)\cos\beta t + c_6 \\ \hdashline -c_4 \sin\beta t - c_1 \cos\beta t + c_5 \\ c_3 \sin\beta t - c_2 \cos\beta t + c_6 \end{bmatrix}, \tag{23}$$

where as before $c_1, \ldots, c_6$ are constants depending on the initial conditions, as follows $c_1 = \mathsf{x}_1(0) - \mathsf{x}_2(0), c_2 = \mathsf{y}_1(0) - \mathsf{y}_2(0), c_3 = \mathsf{x}_2(0) - \mathsf{x}_3(0), c_4 = \mathsf{y}_2(0) - \mathsf{y}_3(0), c_5 = \mathsf{x}_1(0) - \mathsf{x}_2(0) + \mathsf{x}_3(0), c_6 = \mathsf{y}_1(0) - \mathsf{y}_2(0) + \mathsf{y}_3(0)$. The trajectories of agents no. 1 and no. 3 are ellipses, while that of agent no. 2 is a circle. All three trajectories are centered around the point with coordinates $(\mathsf{x}_c, \mathsf{y}_c) = (c_5, c_6)$ and all of them have a period $T = 2\pi/\beta$. Furthermore, it is easy to show that the trajectory of agent no. 3 is an ellipse geometrically identical to that of agent no. 1, but rotated $\pi/2$ radians in the counterclockwise direction. The second observation is that whenever agent no. 1 is at the tip any of its semi-major axis, agent no. 3 is at the tip of its semi-minor axis, and vice-versa. The radius of the circle described by agent no. 2 is $R_2 = \sqrt{(c_1 - c_3)^2 + (c_2 - c_4)^2}$. Analytical expressions for the semi-major and semi-minor axes of the ellipses described by agents no. 1 and no. 3 can be computed by solving for the times at which the distance from the origin is at a maximum and at a minimum. Note that ellipses are special cases of hypotrochoids with $R = 2r$. Figure 3 shows an example for this scenario with $B = \mathrm{diag}(1,-1,1)$ and initial conditions $x_1(0) = (-12,-3)$, $x_2(0) = (-7,9)$, $x_3(0) = (-2,12)$.

2.4 *Case III: Hypotrochoidal Paths*

Consider now the case when $B = \mathrm{diag}(\beta, \beta, -\beta)$, where $\beta > 0$. The analytic calculation of the solution is cumbersome and is omitted for the sake of brevity. Instead,

insightful conclusions about the ensuing paths can be drawn by investigating directly the eigenvalues of the state matrix. A simple calculation shows that the nonzero eigenvalues of the matrix $(BL)\otimes S$ are $\lambda_{1,2}=\pm\beta(\sqrt{2}-1)\mathrm{i}$ and $\lambda_{3,4}=\pm\beta(\sqrt{2}+1)\mathrm{i}$. From the expression $\beta(\sqrt{2}+1)=(k-1)\beta(\sqrt{2}-1)$ it follows

$$\frac{R-r}{r}=k-1=\frac{1+\sqrt{2}}{\sqrt{2}-1}. \tag{24}$$

It follows that the number of crests is given by $k=4+2\sqrt{2}\cong 6.83$, which is an irrational number. A general result in analytic geometry [8] states that if k is irrational, then the number of crests described by the hypotrochoid is infinite. The curve does not close, and the trajectories form a dense subset of the space between the bounding circle R and the circle of radius $R-2r$. In other words, as $t\to\infty$, the hypotrochoids described by each of the three agents fill annular areas.

The path for agent no. 1 is given by $r_1=(5\sqrt{2}-7)(c_1^2+c_2^2)^{\frac{1}{2}}$ and $d_1=(\sqrt{2}+1)(c_3^2+c_4^2)^{\frac{1}{2}}$. For agent no. 2, we have $r_2=(10-7\sqrt{2})(c_1^2+c_2^2)^{\frac{1}{2}}$ and $d_2=(2+\sqrt{2})(c_3^2+c_4^2)^{\frac{1}{2}}$, and for agent no. 3, we have $r_3=(3-2\sqrt{2})(c_1^2+c_2^2)^{\frac{1}{2}}$, and $d_3=(c_3^2+c_4^2)^{\frac{1}{2}}$, where the constants c_i $i=1,\ldots,4$ can be easily computed in terms of the initial conditions as follows

$$c_1=\left(-\frac{1}{2}-\frac{\sqrt{2}}{4}\right)\mathsf{y}_1(0)+\left(-\frac{1}{2}-\frac{\sqrt{2}}{2}\right)\mathsf{y}_2(0)+\left(1+\frac{3\sqrt{2}}{4}\right)\mathsf{y}_3(0), \tag{25a}$$

$$c_2=\left(-\frac{1}{2}-\frac{\sqrt{2}}{4}\right)\mathsf{x}_1(0)+\left(-\frac{1}{2}-\frac{\sqrt{2}}{2}\right)\mathsf{x}_2(0)+\left(1+\frac{3\sqrt{2}}{4}\right)\mathsf{x}_3(0), \tag{25b}$$

$$c_3=\left(-\frac{1}{2}+\frac{\sqrt{2}}{4}\right)\mathsf{y}_1(0)+\left(-\frac{1}{2}+\frac{\sqrt{2}}{2}\right)\mathsf{y}_2(0)+\left(1-\frac{3\sqrt{2}}{4}\right)\mathsf{y}_3(0), \tag{25c}$$

$$c_4=\left(\frac{1}{2}-\frac{\sqrt{2}}{4}\right)\mathsf{x}_1(0)+\left(\frac{1}{2}-\frac{\sqrt{2}}{2}\right)\mathsf{x}_2(0)+\left(-1+\frac{3\sqrt{2}}{4}\right)\mathsf{x}_3(0), \tag{25d}$$

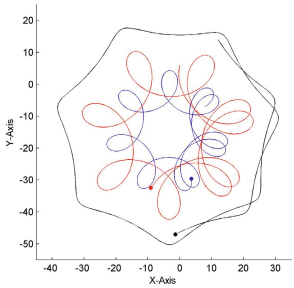
$$c_5=\mathsf{x}_1(0)+\mathsf{x}_2(0)-\mathsf{x}_3(0), \qquad c_6=\mathsf{y}_1(0)+\mathsf{y}_2(0)-\mathsf{y}_3(0). \tag{25e}$$

The center of the orbits of the three agents is located at $(\mathsf{x}_c,\mathsf{y}_c)$ where $\mathsf{x}_c=\mathsf{x}_1(0)+\mathsf{x}_2(0)-\mathsf{x}_3(0)$ and $\mathsf{y}_c=\mathsf{y}_1(0)+\mathsf{y}_2(0)-\mathsf{y}_3(0)$. The radius of the circular track for each agent is $R_i=kr_i$ $(i=1,2,3)$.

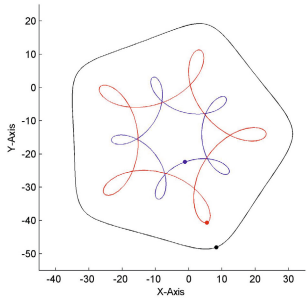
Figure 4(a) shows an example with $B=\mathrm{diag}(1,1,-1)$ and initial conditions $x_1(0)=(8,-7)$, $x_2(0)=(0,6)$, $x_3(0)=(12,14)$.

2.5 General Case

System (16) describes a rich family of geometric curves. However, the approach is still restrictive in the sense that the curves have some fixed parameters that cannot be altered by just changing the initial conditions. For instance, the epitrochoids



(a) A set of three hypotrochoids described by three agents. Although each one has a different r_i, d_i and R_i parameters, they all display non-closing curves. Notice that the value of k for this example is irrational.



(b) Same group of agents as in (a), with control redesign to describe closed hypotrochoids with five crests. This pattern necessitates a different graph topology of the intra-agent information exchange (i.e., a cycle graph).

Fig. 4 Examples of hypotrochoidal paths with three agents interconnected in a path graph and a cycle graph

described in Case I can only have two crests. Moreover, the orbits of agents no. 1 and no. 3 are identical, except for the fact that they are phased apart by an angle $\pi/2$, along with the condition that the trajectory of agent no. 2 is a circle. Similar statements can be made for Cases II and III. Also, recall that the hypotrochoids of Case III were not periodic. It would be of great interest to a mission designer to be able to employ vehicles generating (in a distributed, cooperative manner) suitable trajectories with specific geometric characteristics. For instance, it may be desirable to be able to generate epitrochoids or hypotrochoids with a certain number of crests in order to survey an area or perimeter of interest, or provide telecommunication coverage over a region, etc.

Based on the discussion in the previous sections, the shape and frequencies of the resulting paths/trajectories is determined by the eigenvalues and eigenvectors of the matrix $(BL) \otimes S$. Recall from the properties of the Kronecker product that the eigenvalues of the matrix $(BL) \otimes S$ are of the form $\lambda\mu$ where $\lambda \in \operatorname{spec}(BL)$ and $\mu \in \operatorname{spec} S$. Additionally, the corresponding eigenvectors are of the form $\mathsf{v} \otimes \mathsf{u}$ where $\mathsf{v} \in \mathbb{C}^3$ is the eigenvector of the matrix BL associated with λ and $\mathsf{u} \in \mathbb{C}^2$ is the eigenvector of the matrix S associated with μ. The task of agent trajectory design therefore reduces to the task of imposing the correct conditions on the spectral properties of the matrix BL. For example, closed paths with the correct number of crests may be ensured by selecting a suitable rational value of k, along with the eigenvalues of the matrix BL. The type of path (epitrochoid, ellipse, hypotrochoid) may be determined by the corresponding eigenvectors. It is clear that the path design depends both on the feedback gain matrix B, as well as on the imposed graph topology represented by the incident matrix D (equivalently, the graph Laplacian L).

Consider, for instance, again Case III of the hypotrochoidal paths shown in Fig. 4(a), and let us assume that we want to keep the general, overall shape of these

paths, but we want to have closed, periodic paths instead with a given number of crests. By keeping the same eigenvectors and by changing only the eigenvalues (choose for instance the smallest nonzero eigenvalue to be equal to $\lambda_{1,2} = 5/3$) and by imposing five crests (hence $k = 5$), we are led to the following control law[5]

$$u = ((B \circ D) \otimes S)z = ((B \circ D) \otimes S)(D^{\mathsf{T}} \otimes I_2)x, \tag{26}$$

which can be written, componentwise, as follows

$$u_i = \sum_{k=1}^{M} \beta_{ik} d_{ik} p_k = \sum_{k=1}^{M} \beta_{ik} d_{ik} S z_k, \quad i = 1, \ldots, N, \tag{27}$$

where $B = \begin{bmatrix} 0.6213 & 0.8431 & 0.1109 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 & -1 \end{bmatrix}^{\mathsf{T}}$ and with an incidence matrix

$$D = \begin{bmatrix} -1 & 0 & 1 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix}. \tag{28}$$

The trajectories of the agents are shown in Fig. 4(b). Note that these trajectories necessitate a different communication topology, namely, one which, for this case, corresponds to a complete graph.

3 Conclusions

We have presented an extension of the classical consensus algorithm for multi-agent systems. The main idea hinges on the use, by each agent, of additional directional information that can be readily inferred from knowledge of the relative position with respect to the other agents. The resulting control law seems to be a genuine generalization of the classical consensus design protocol since it is not induced by a scalar potential, and it can lead to agreement values that lie outside the convex hull of initial conditions. A special choice of the feedback gains leads to periodic or quasi-periodic solutions that can be used to design trajectories suitable for persistent optimized surveillance and monitoring applications by a team of agents. The resulting trajectories show intricate geometric patterns generated using only relative, local information. Future work will concentrate on developing a general theory for orbit synthesis for an arbitrary number of agents in two and three dimensions.

References

1. Arcak, M.: Passivity as a design tool for group coordination. IEEE Transactions on Automatic Control 52(8), 1380–1390 (2007), doi:10.1109/TAC.2007.902733
2. Bernstein, D.S.: Matrix Mathematics: Theory, Facts, and Formulas, 2nd edn. Princeton University Press, Princeton (2009)

[5] Recall that, given two matrices $A \in \mathbb{R}^{n \times m}$ and $B \in \mathbb{R}^{n \times m}$ the Schur product $A \circ B$ is defined by $(A \circ B)_{ij} = A_{ij} B_{ij}$.

3. Fax, J.A., Murray, R.M.: Information flow and cooperative control of vehicle formations. IEEE Transactions on Automatic Control 49(9), 1465–1476 (2004), doi:10.1109/TAC.2004.834433
4. Godsil, C., Royle, G.: Algebraic Graph Theory. Springer, New York (2001)
5. Hall, L.: Trochoids, Roses, and Thorns–Beyond the Spirograph. College Mathematics Journal 23(1), 20–35 (1992)
6. Justh, E., Krishnaprasad, P.: Equilibria and Steering Laws for Planar Formations. Systems & Control Letters 52(1), 25–38 (2004), doi:10.1016/j.sysconle.2003.10.004
7. Justh, E.W., Krishnaprasad, P.S.: Natural frames and interacting particles in three dimensions. In: Proc. 44th IEEE Conf. and 2005 European Control Conf. Decision and Control CDC-ECC 2005, pp. 2841–2846 (2005), doi:10.1109/CDC.2005.1582594
8. Lawrence, J.: A Catalog of Special Plance Curves, 1st edn. Dover Publications (1972)
9. Leonard, N.E., Fiorelli, E.: Virtual leaders, artificial potentials, and coordinated control of groups. In: Proc. of the 40th IEEE Conference on Decision and Control, pp. 2968–2973 (2001), doi:10.1109/.2001.980728
10. Marshall, J.A., Broucke, M.E., Francis, B.A.: Formations of vehicles in cyclic pursuit. IEEE Trans. on Automatic Control 49(11), 1963–1974 (2004), doi:10.1109/TAC.2004.837589
11. Mesbahi, M., Egerstedt, M.: Graph Theoretic Methods in Multiagent Networks. Princeton University Press, Princeton (2010)
12. Olfati-Saber, R., Fax, J.A., Murray, R.M.: Consensus and cooperation in multi-agent networked systems. Proceedings of the IEEE 97, 215–233 (2006), doi:10.1109/JPROC.2006.887293
13. Olfati-Saber, R., Murray, R.M.: Consensus problems in networks of agents with switching topology and time-delays. IEEE Trans. on Automatic Control 49(9), 1520–1533 (2004), doi:10.1109/TAC.2004.834113
14. Pavone, M., Frazzoli, E.: Decentralized policies for geometric pattern formation and path coverage. Journal of Dynamic Systems, Measurement, and Control 129, 633–643 (2007), doi:10.1115/1.2767658
15. Ren, W.: Collective motion from consensus with cartesian coordinate coupling - part i: Single-integrator kinematics. In: Proc. 47th IEEE Conf. Decision and Control CDC 2008, pp. 1006–1011 (2008), doi:10.1109/CDC.2008.4738708
16. Ren, W., Beard, R.W.: Consensus seeking in multi-agent systems using dynamically changing interaction topologies. IEEE Trans. on Automatic Control 50(5), 655–661 (2005), doi:10.1109/TAC.2005.846556
17. Scardovi, L., Leonard, N.E., Sepulchre, R.: Stabilization of collective motion in three dimensions: A consensus approach. In: Proc. 46th IEEE Conf. Decision and Control, pp. 2931–2936 (2007), doi:10.1109/CDC.2007.4434721
18. Sepulchre, R., Paley, D.A., Leonard, N.E.: Stabilization of planar collective motion: All-to-all communication 52(5), 811–824 (2007), doi:10.1109/TAC.2007.898077
19. Singh, L., Stephanou, H., Wen, J.: Real-time robot motion control with circulatory fields. In: Proceedings of 1996 IEEE International Conference on Robotics and Automation, pp. 2737–2742 (1996), doi:10.1109/ROBOT.1996.506576
20. Suzuki, I., Yamashita, M.: Distributed anonymous mobile robots: Formation of geometric patterns. SIAM J. on Computing 28(4), 1347–1363 (1999), doi:10.1137/S009753979628292X
21. Wang, L.S., Krishnaprasad, P.S.: Gyroscopic control and stabilization. Journal of Nonlinear Science 2, 367–415 (1992), doi:10.1007/BF01209527

Utilizing Stochastic Processes for Computing Distributions of Large-Size Robot Population Optimal Centralized Control

Dejan Milutinović

Abstract. Motivated by the success of stochastic process sampling methods in solving complex estimation problems, we explore the possibility to utilize stochastic processes for computing optimal control for a large-size robot population. We assume that the individual robot state is composed of discrete and continuous components, while the population is controlled in a probability space. The optimal control solution is based on an infinite dimensional Pontryagin-like minimum principle, which involves an evaluation of systems of partial differential equations. The paper shows that these equations can be evaluated with computations involving stochastic process samples. This is an important result because generating stochastic process multi-dimensional trajectories is much easier than solving corresponding multi-dimensional partial differential equations. The proposed evaluations are illustrated and verified by an example of the centralized optimal control for a large-size robot population.

1 Introduction

The solution of multi-robot control problems [20] can be of enormous complexity due to a large number of redundant states and robots, as well as environmental uncertainties. It has been known for many years that optimal control and optimal estimation problems are closely related [8, 23]. Motivated by the success of statistical sampling methods in solving complex estimation problems [21], we explore the possibility of utilizing stochastic processes for computing the solution of a complex optimal control problem, such as the centralized optimal control for a large-size robot population.

Dejan Milutinović
Applied Mathematics and Statistics Department, Baskin School of Engineering,
UC Santa Cruz, 1156 High Street, CA 95064, USA
e-mail: dejan@soe.ucsc.edu

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 359–372.
springerlink.com

Decentralized control has been proposed as a control strategy for large-scale dynamical systems [24] composed of coupled dynamical systems. Therefore, having in mind that an individual robot is a dynamical system, which can be coupled with dynamics of other robots to form a multi-robot system, decentralized control strategies appear to be a natural choice for multi-robot systems control. Some examples of decentralized strategies applied in control of multi-robot systems and robot swarms are provided in [3, 7, 10, 15, 16]. However, in the presence of a vantage point when commands to robots can be broadcasted, a centralized control strategy, which is in the focus of this paper is feasable and very appealing; specifically, in the case of a robot swarm composed of simple miniature robots with a limited computational capability, so that their behavior depends on the broadcasted commands. It is reasonable to expect that such miniature robot systems will be a driving force in the development of nano-robotics and that studying their control can help us in designing specifications of future nano-robots [9].

In this paper, we consider a robot model from a stochastic hybrid automata class with stochastic discrete state transitions and deterministic continuous dynamics in each discrete state. This model was previously used in modeling a large-size robot population and formulating the control problem that maximizes the robot presence in a desired region of the operating space [18, 19]. The model belongs to a class of stochastic hybrid automata [11], or, more precisely, to a class of piecewise deterministic systems [4]. However, the model allows an optimal control problem solution based on a Pontryagin-like minimum principle for partial differential equations[5], which is presented in [17, 18], and is solved numerically when the presence of robots is maximized along one dimension (1D).

The Hamiltonian from the minimum principle, which defines the optimal control, includes integral terms that depend on the solution of two systems of partial differential equations (PDEs). One system describes the hybrid state probability density function evolution and the other, the corresponding co-state distribution evolution. In general, these PDE systems are difficult to evaluate and this presents a major difficulty in computing the solution of the control problem in more dimensions. Therefore, we propose to utilize stochastic processes to compute necessary distributions, which is the major contribution of this paper. We describe a Gillespie-like numerical algorithm for generating trajectories of our discrete-continuous stochastic process and derive the relation between stochastic process samples and the co-state distribution. This relation comes in the form that is in the spirit of the so-called Feynman-Kac formula [13] from statistical physics.

Along the idea of utilizing stochastic processes for control, Kappen et. al. [1, 12] studied control of stochastic differential equations. They were able to relate the stochastic Hamilton-Jacobi-Bellman partial differential equation with samples of stochastic process trajectories and use the samples to define the stochastic optimal control of a multi-agent system. In their framework, the state is a vector of real numbers. However, in the case of our stochastic hybrid automaton, which is a more general dynamical model [22], the state is defined by continuous and discrete variables, i.e, the hybrid state.

The direction of the research we are pursuing is also different from the stochastic optimal control work presented in [14]. There, stochastic processes have been used as an analytical tool to map the stochastic process to be controlled into a finite state space in which the optimization is performed. The benefit of using a solution based on sampling, i.e, computational statistical methods, is in the opportunity to embed stochastic process generation and necessary computations into physical systems, which can provide an efficient solution of control problems in robotics and similar applications.

We discuss the modeling and control framework in Section 2, which is followed by Section 3 explaining the algorithm generating stochastic process trajectories and the state PDF evaluation. Section 4 describes the evaluation of the co-state distribution. Section 5 provides an example and Section 6 conclusions.

2 Modeling and Control Framework

In the modeling framework we consider, the state of an individual robot at time t is uniquely defined by the couple $(x(t), q(t))$, the so-called hybrid state, where $x \in X$, $X \in \mathbb{R}^n$, $q \in Q, Q = \{1, 2, ...K\}$. While in the discrete state (mode) $k \in Q$, the continuous state x of a robot obeys the differential equation $\dot{x} = f_k(x,t)$. We also assume that switching among the discrete states, say from the state $j \in Q$ to the state $k \in Q$, $(k \neq j)$, is described by time-varying stochastic transition rates $\lambda_{jk}(t)$, and that $x(t)$ is a continuous time function. The latter means that the continuous state $x(t_c^+)$ immediately following the time point t_c of the discrete state transition is equal to the state $x(t_c)$ before the state transition. This very general model of an individual robot is illustrated in Fig. 1 and, in the control framework of this paper, the stochastic transition rates $\lambda_{jk}(t)$ are functions of control variables. The modeling and control framework we are applying here is detailed in [18] and summarized in this section.

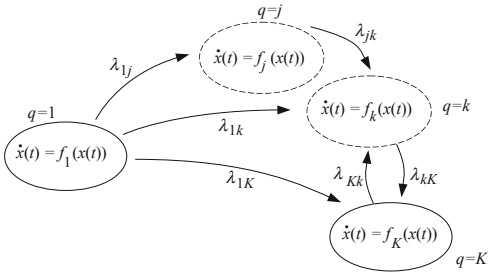


Fig. 1 Stochastic hybrid automaton model of a robot in a probabilistic framework: discrete state q; continuous state x vector field f_k, $k \in Q$ describes the change of the continuous state; stochastic transition rates λ_{jk}, $j,k \in Q$ describe the mode switching

Collective Dynamics: We describe the collective dynamics of a large-size population by the joint probability distribution evolution of the continuous variable $x(t)$ and the discrete variable $q(t)$, for example, x being a robot position and q a motion mode, or a task type. This joint distribution is a hybrid-state probability density function (PDF), or shortly a state PDF. The state PDF is a vector of functions $\rho(x,t) = [\rho_1(x,t), \rho_2(x,t), \dots \rho_K(x,t)]'$. Each component $\rho_i(x,t)$ corresponds to the discrete state $i \in Q$, and the symbol $(')$ denotes the vector transpose. Naturally, the state PDF satisfies

$$\sum_{i \in Q} \int_X \rho_i(x,t) dx = \sum_{i \in Q} P_i(t) = 1 \tag{1}$$

where $P_i(t) = \int_X \rho_i(x,t) dx$ is the probability of the discrete state i at a time point t. Let us define the vector of discrete state probabilities $P(t) = [P_1(t), P_2(t), \dots P_K(t)]'$, then the evolution of the probability vector is given by [2]

$$\dot{P}(t) = F_t(t) P(t), \quad \text{where} \quad [F_t]_{ij} = \lambda_{ij}(t) \tag{2}$$

with matrix F_t defining the transition rates among the discrete states. In general, the transition rates λ_{ij} can depend on the control vector $u(t) = [u_1(t)\ u_2(t)\ \dots u_M(t)]'$. Consequently, the control variables $u_i, i = 1,2 \dots M$, define the transition rate matrix, i.e., $F_t(t) = F_u(u(t))$ and the vector of the discrete state probabilities obeys

$$\dot{P}(t) = F_u(u(t)) P(t) \tag{3}$$

Finally, it can be proven [18] that the state PDF obeys the following system of partial differential equations (PDEs):

$$\frac{\partial \rho(x,t)}{\partial t} = F(u(t)) \rho(x,t) = (F_u(u(t)) + F_\partial) \rho(x,t) \tag{4}$$

where F_∂ is a diagonal linear differential operator. When the operator F_∂ is applied to $\rho(x,t)$, it results in:

$$[F_\partial \rho(x,t)]_{ij} = \begin{cases} -\nabla \cdot (f_i \rho_i(x,t)), i = j \\ 0, i \neq j \end{cases}, \quad i,j = 1,2 \dots K \tag{5}$$

Cost Function and Optimal Control: Taking into account that the state PDF $\rho(x,t)$ evolution depends on the vector $u(t)$, we can formulate the optimal control problem in the probability space using the vector of weighting functions $w(x)$ and the cost function:

$$J = \int_X w'(x) \rho(x,T) dx = E_{\rho(T)}\{w(x)\} \tag{6}$$

where $E_{\rho(T)}\{\cdot\}$ denotes the expectation with respect to the state PDF ρ at a terminal time point T. It is convenient to define the scalar product $\langle p, q \rangle$ of the function vectors $p(x)$ and $q(x)$ as

$$\langle p, q \rangle = \int_X p'(x) q(x) dx \tag{7}$$

Notice that in this notation we omit x inside the scalar product brackets because x is the variable of integration. We formulate the optimal control problem as the following optimization problem:

$$u^{opt} = \arg\max_{u \in U_{ad}} J(u) = \arg\max_{u \in U_{ad}} \int_X w'(x)\rho(x,T)dx = \arg\max_{u \in U_{ad}} \langle w, \rho(T) \rangle \quad (8)$$

under the PDE system constraint (4) where U_{ad} is the set of admissible control. Alternatively, to avoid singular control problems [18], we can also consider the optimal control based on a cost function including the term penalizing the control, such as:

$$u^{opt} = \arg\max_{u \in U_{ad}} \int_X w'(x)\rho(x,T)dx + \varepsilon \int_0^T u'(t)u(t)dt \quad (9)$$

where ε is a positive constant penalizing the intensity of the control. Anyway, the solution of the control problem is a sequence of the optimal control $u^{opt}(t)$ from the set of admissible control U_{ad}, such that the cost function is maximized. By a suitable choice of the weighting function $w(x)$ and a small value of ε, the cost function can be used to find the optimal control maximizing the probability of the robot presence in a desired region of the robots' operating space.

Minimum Principle: The optimal control maximizing criterion (6) is a special case of a more general optimal control problem of the evolution equation [5] in an infinite dimensional space. Under the condition that the operator $F(u(t))$ is bounded, i.e., $\|F(u(t))\| < \infty$, the minimum principle for PDEs can be applied [5]. According to the minimum principle, the optimal control $u^{opt}(t)$ satisfies:

$$u^{opt}(t) = \arg\min_{u(t) \in U_{ad}} H(\rho^{opt}(t), u(t), t) \quad (10)$$

where H is a Hamiltonian defined as

$$H(\rho(t), u, t) = \langle \pi(t), F(u(t))\rho(t) \rangle \quad (11)$$

and the function vector $\pi(x,t)$ is the so-called co-state distribution and obeys:

$$\begin{aligned} \frac{\partial \pi(x,t)}{\partial t} &= -F'(u(t))\pi(x,t) \\ \pi(x,T) &= -w(x) \end{aligned} \quad (12)$$

where $F'(u(t))$ is the adjoint operator of the operator $F(u(t))$, which means that

$$\langle \pi(t), F(u(t))\rho(t) \rangle = \langle F'(u(t))\pi(t), \rho(t) \rangle \quad (13)$$

The major difficulty in computing the optimal control is the evaluation of the state ρ and co-state π distributions resulting from the PDE system solutions (4) and (12). In the following sections, we will show that they can be evaluated utilizing stochastic process samples.

3 Stochastic Sampling Propagator

The evolution of the state PDF $\rho(x,t)$ is described by the PDE system (4). One way to obtain the evolution $\rho(x,t)$ is to solve the PDE system forward in time starting from the initial condition $\rho(x,0) = \rho^0(x)$. We propose an approach to computing the evolution $\rho(x,t)$ based on stochastic trajectories of the hybrid state $(x_s(t), q_s(t))$ resulting from the model presented in Fig. 1 where each trajectory can be associated with a single robot.

The basis for the proposed algorithm is the Gillespie's stochastic simulation algorithm [6], in which whenever the discrete state is $q_s(t) \in Q$, the evolution of the continuous state x_s obeys $\dot{x}_s = f_{q_s(t)}(x_s(t))$. To account for the fact that the transition rates can change in time, we assume that the control $u(t)$ is a piecewise constant function of time discretized with the sample time ΔT.

Initialization: The discrete state probability $P_i(t)$ of the discrete state i is:

$$P_i(t) = \int_X \rho_i(x,t)dx \tag{14}$$

and the initial discrete state $q_s(0)$ should be generated as a random number i with the probability $P_i(0)$. Simbolically, we write it as:

$$q_s(0) = i,\ i \sim P_i(0) \tag{15}$$

Once the initial discrete $q_s(0)$ state is defined, the continuous variable $x_s(0)$ can be initialized as a random number from the PDF corresponding to $\rho_{q_s(0)}(x,0)$ component of the state PDF, i.e.,

$$x_s(0) \sim \frac{1}{P_{q_s(0)}(0)} \rho_{q_s(0)}(x,0) \tag{16}$$

where normilizing coefficient $P_{q_s(0)}(0)$, which is the probability of the discrete state $q_s(0)$, provides that given $q_s(0)$ the probability density function of $x_s(0)$ is normalized to one.

Transitions: Let us assume that at time $t = t_s$, $t_s \in [(k-1)\Delta T, k\Delta T)$, the hybrid state is $(x_s(t_s), q_s(t_s))$, $q_s(t_s) = i$, and k is the integer index of the time interval; having in mind that the control vector $u(t)$ and consequently the transition rates $\lambda_{ij}(t)$, as well as the total transition rates from the state i, $\lambda_i^{out}(t) = \sum_{j\in Q, j\neq i} \lambda_{ij}(t)$ are piecewise constants, we can generate the discrete state transition points t_c based on the following two rules:

(a) $t_c = t_s + t_t$, $t_t \sim Exp(\lambda_i^{out}((k-1)\Delta T))$, under the condition that $t_c < k\Delta T$. If the condition is not satisfied, apply rule (b).
(b) $t_c = k\Delta T + t_t$, $t_t \sim Exp(\lambda_i^{out}(k\Delta T))$, under the condition that $t_c < (k+1)\Delta T$. If the condition is not satisfied, increase k by 1. Apply rule (b) until the condition is satisfied.

where $Exp(\cdot)$ denotes the exponential probability density function with the rate parameter in brackets. These two rules define the time point t_c at which the jump from the discrete state $q_s(t_c) = i$ happens, but do not specify the discrete state $q_s(t_c^+) = j$. The state $q_s(t_c^+)$ is defined as a random number j with the probability corresponding to the rates $\lambda_{ij}(t_c)$ and the discrete state $q(t_c) = i$, $j \neq i$, i.e.,

$$q(t_c^+) = j, j \sim \frac{1}{\lambda_i^{out}(t_c)} \lambda_{ij}(t_c), \tag{17}$$

and the probability of $q(t_c^+) = i$ is zero. The denominator $\lambda_i^{out}(t_c)$ provides that the probability of possible discrete state realizations of $q(t_c^+)$ sums up to one.

Continuous state evolution: The rules for the initialization of the transition points define the evolution of the discrete state $q_s(t)$. The continuous state evolution is defined by

$$\dot{x_s}(t) = f_{q_s(t)}(x_s(t)), \text{ and } x_s(t_c^+) = x_s(t_c) \tag{18}$$

The above rules define the stochastic evolution of the model shown in Fig. 1 for the piecewise constant transition rates $\lambda_{ij}(t) = \lambda_{ij}(u(t))$. In the limit of a large number of samples, the normalized density of trajectory points corresponds to the solution of the PDE system given by (4). In this respect, the stochastic simulation is a computational propagator of the evolution $\rho(x,t)$.

4 Co-state Distribution

Under the assumption that the negative co-state $-\pi(x,t)$ can be treated as the state PDF of a time-backward stochastic process corresponding to (12), we can use the same approach to evaluate $-\pi(x,t)$ as the state PDF $\rho(x,t)$. In general, this is not a valid assumption (see Appendix A). Therefore, we need to consider an alternative approach to the evaluation of $\pi(x,t)$, which is provided by the following theorem. This result is similar to the Feynman-Kac formula [13], but at the same time more general because we consider stochastic processes involving both continuous and discrete state variables.

Theorem. Assuming that the state probability density function $\rho(x,t)$ evolution obeys

$$\frac{\partial \rho(x,t)}{\partial t} = F(u(t))\rho(x,t), \ \rho(x,0) = \rho^0(x) \tag{19}$$

where $\rho^0(x)$ is a given initial condition, and there is a co-state distribution $\pi(x,t)$ satisfying

$$\frac{\partial \pi(x,t)}{\partial t} = -F'(u(t))\pi(x,t), \ \pi(x,T) = -w(x) \tag{20}$$

where $w(x) = -[w_1(x), w_2(x), \ldots w_K(x)]'$ is a given terminal condition at the terminal time T and $F'(u)$ is the adjoint operator satisfying (13). Then, the ith component of the co-state distribution $\pi_i(\hat{x},t)$ at the point $\hat{x}$ is

$$\pi_i(\hat{x},t) = E\left\{w_{q_s(T)}(x_s(T))|q_s(t) = i, x_s(t) = \hat{x}\right\} \tag{21}$$

which is the expected value of $w_{q_s(T)}(x_s(T))$, where $q_s(T)$ and $x_s(T)$ are the final discrete and continuous states, respectively, of the stochastic process corresponding to $\rho(t)$, $t \in [0,T]$ and initialized with the discrete state i and the continuous state $\hat{x}$ at the time point t.

Proof. Let us consider the evolution equation:

$$\frac{\partial \phi(x,t)}{\partial t} = F(u(t))\phi(x,t),\ x \in X \tag{22}$$

which is the same as evolution (19), but ϕ is not equal to ρ because $\rho(x,0) = \rho^0(x) \neq \phi(x,0)$. The scalar product between $\pi(x,t)$ and $\phi(x,t)$ is:

$$S(t) = \langle \pi(t), \phi(t) \rangle = \int_X \pi'(x,t)\phi(x,t)dx \tag{23}$$

and its time derivative is:

$$\dot{S}(t) = \int_X \left[(\frac{\partial}{\partial t}\pi(x,t))'\phi(x,t) + \pi'(x,t)\frac{\partial}{\partial t}\phi(x,t) \right] dx \tag{24}$$

,i.e.,

$$\dot{S}(t) = \int_X \left[-\pi(x,t)F(u)\rho(x,t) + \pi(x,t)F(u)\rho(x,t) \right] dx$$

from which we can conclude $\dot{S}(t) = 0$. In other words, the scalar product $S(t)$ does not depend on time and, consequently,

$$\int_X \pi'(x,t)\phi(x,t)dx = \int_X \pi'(x,T)\phi(x,T)dx \tag{25}$$

Let us assume that, at a given time point t, the ith component of $\phi(x,t)$ is $\phi_i(x,t) = \delta(x-\hat{x})$, where $\delta(x-\hat{x})$ denotes the Dirac pulse centered at the point $\hat{x}$, and that all other components of $\phi(x,t)$ are zero, i.e., $\phi_j(x,t) = 0,\ i \neq j,\ \forall x \in X$. Under this condition and having in mind that $\pi(x,T) = -w(x)$, we obtain:

$$\pi_i(\hat{x},t) = -\int_X w'(x)\phi(x,T)dx = -E\left\{w(x)\right\} \tag{26}$$

which is the expression derived under the condition $\phi_i(x,t) = \delta(x-\hat{x})$, $\phi(x,t) = 0$, $\forall i \neq j$ and where $E\{\cdot\}$ is the expectation operator. Introducing the hybrid state $(x_s(t), q_s(t))$ trajectory corresponding to the evolution of ϕ, we conclude

$$\pi_i(\hat{x},t) = -E\left\{w_{q_s(T)}(x_s(T))|q_s(t) = i, x_s(t) = \hat{x}\right\} \tag{27}$$

□

In order to evaluate $\pi_i(\hat{x},t)$, we should generate N trajectories starting from the discrete state $q_s(t) = i$ and the continuous state $x_s(t) = \hat{x}$. If each trajectory has index k, $k = 1,2\ldots N$, then the expected value (21) can be computed as:

$$\tilde{\pi}_i(\hat{x},t) = -\frac{1}{N}\sum_k w_{q_s^k(T)}(x_s^k(T)) \tag{28}$$

where the symbol ' ~ ' denotes the fact that the value on the right hand side of the expression is an approximate value of $\pi_i(\hat{x},t)$, $x_s^k(T)$ is the sample of trajectory k at the terminal time T and $w_{q_s^k(T)}(x_s^k(T))$ is the value of the weighting function $w(x)$ component corresponding to the discrete state of the kth trajectory $q_s^k(T)$ at the terminal time and evaluated at the point $x_s^k(T)$. In the limit of a large number of samples N, due to the central limit theorem, the distribution of the error $\tilde{\pi}_i(\hat{x},t) - \pi_i(\hat{x},t)$ is Gaussian, decreases with the rate proportional to $1/\sqrt{N}$ and for $N \to \infty$ is zero. But we should mention that the variance also depends on the shape of the co-state distribution $\pi(x,t)$ and it is expected that we need a large number of samples to achieve good precision, as it is illustrated by examples in the following section. Also, the number of samples to compute the overall distribution $\pi(x,t)$ depends on the number of points at which the distribution should be evaluated.

However, let us conclude that although we need a large number of samples to compute the co-state distribution $\pi(x,t)$, the generality of the method should not be overlooked. The method we propose is applicable in situations when the number of dimensions of the continuous state x, or nonlinearities of continuous state evolutions $\dot{x} = f_i(x,t)$ result in PDE systems that cannot be reliably evaluated . Moreover, in order to gain computational speed, sampling of stochastic trajectories and the computation of $\tilde{\pi}_i(\hat{x},t)$ can be easily parallelized and programmed for multi-processor computational hardware.

5 1D Example

The stochastic model presented in Fig. 2*b* illustrates the state PDF evolution of a large-size robot population along one dimension (Fig. 2*a*), in which u_1, u_2 and u_3 correspond to stochastic rates of the command signals: move-left (L), move-right (R) and stop (S). In this example, the velocities of moving left and right are $k_1 = -0.5$ and $k_2 = 0.25$, respectively. The PDE systems describing the state PDF and the co-state distribution evolutions are provided in the Appendix B. The control $u(t) = [u_1(t), u_2(t), u_3(t)]$ is computed as the optimal control based on the Minimum-principle and Hamiltonian presented in the previous section.

The cost function is:

$$J(u) = \int_X w'(x)\rho(x,t)dx + \varepsilon\int_0^T u_1^2(t) + u_2^2(t) + u_3^2(t)dt \tag{29}$$

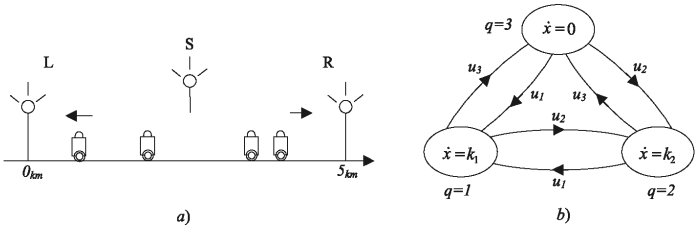


Fig. 2 1D example: a) A robotic population controlled by the three signal sources: L-left, S-stop and R-right b) The stochastic hybrid automaton model [17, 18]

where $\varepsilon = 10^{-7}$, the weighting $w(x) = [0,0,w_3(x)]'$ and the initial condition $\rho(x,0) = [0,0,\rho_3(x,0)]'$ are defined by:

$$w_3(x) = \begin{cases} \frac{1}{\sqrt{0.01}} exp(-\frac{(x-1.75)^2}{0.01}), & 1.25 < x < 2.25 \\ 0, \ otherwise \end{cases} \tag{30}$$

$$\rho_3(x,0) = \begin{cases} \frac{1}{\sqrt{0.02\pi}} exp(-\frac{(x-2.5)^2}{0.02}), & 2 < x < 3 \\ 0, \ otherwise \end{cases} \tag{31}$$

The optimal control sequence $u^{opt}(t) = [u_1^{opt}(t), u_2^{opt}(t), u_3^{opt}(t)]$ in the time interval $0 < t < 3$ is defined by:

$$u_1^{opt}(t) = \begin{cases} 2, \ 0.21 < t < 1.74 \\ 0, elsewhere \end{cases}, \ u_2^{opt}(t) = 0, \ u_3^{opt}(t) = \begin{cases} 2, \ 1.71 < t < 3 \\ 0, elsewhere \end{cases} \tag{32}$$

The evolution of the state PDF for this system under the control $u^{opt}(t)$ is presented in Fig. 3. Due to the space limitation, we present only $\rho_3(x,t)$.

The evolution of the discrete state q can be observed from the trend in x. When x decreases, the discrete state is 1, and when it remains constant, the state is $q = 3$. It is worth mentioning that, among these 10 trajectories, there is one for which $x(t)$ is constant. The small peak near the point 2.5 in the right panel of Fig.3, at $t = 3$, confirms that the probability of such trajectories is non-zero, but it is small.

To obtain the state PDF $\rho(x,t)$, i.e., its components $\rho_i(x,t)$ at a specific time point t, we need to collect points $x(t)$ and estimate components $\rho_i(x,t)$. It is obvious that 10 trajectories cannot provide a good estimate of $\rho(x,t)$. For this reason, we generated 10^5 trajectories and computed the histogram probability density function estimation. That means that we discretized the x axis into intervals of the length $\Delta x = 0.01$ and counted how many points fell into a specific region. Finally, we normalized the histogram so that for the estimated $\rho(x,t)$ we have $\langle \rho(x,t), 1 \rangle = 1$. To save space, we show the results only for $\rho_3(x,t)$ and compare the finite element PDE solution, Fig. 3 (left panel), with the stochastic simulation solution presented in Fig. 3 (right panel). As expected, the match between the finite element PDE system solution and the result obtained from stochastic trajectories is almost exact.

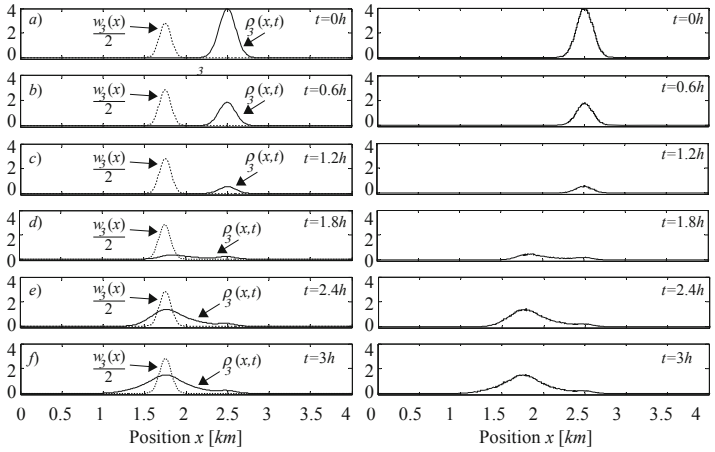


Fig. 3 Left panel: The finite element PDE solution for the state PDF $\rho_3(x,t)$ and the optimal control $u^{opt}(t)$ [17, 18]. Right panel: The stochastic process-based solution for the state PDF $\rho_3(x,t)$ and the optimal control $u^{opt}(t)$

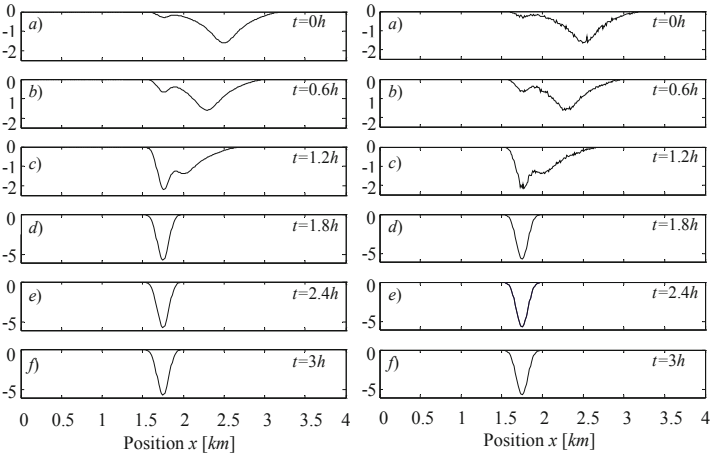


Fig. 4 Left panel: The finite element PDE solution for the co-state distribution $\pi_3(x,t)$ and the optimal control $u^{opt}(t)$. Right panel: The stochastic process-based solution for the co-state distribution $\pi_3(x,t)$ and the optimal control $u^{opt}(t)$

There are only negligible discrepancies due to data sampling from a finite number of trajectories.

Figure 4 (left panel) shows the finite element PDE solution for the co-state $\pi_3(x,t)$. We also evaluated the co-state utilizing stochastic processes (see Fig. 4, right panel). The co-state $\pi_3(x,t)$ is evaluated at 500 equally spaced points ($\Delta x = 0.01$). For each point, we generated $N = 10^4$ trajectories and applied expression (28). We can conclude that the match between the PDE solution (Fig. 4,

left panel) and the solution based on stochastic process samples (Fig. 4, right panel) is almost exact, except for the intrinsic stochastic fluctuations. The intensity of the fluctuations can be reduced with a larger number of trajectories N.

6 Conclusion and Further Work

Computing distributions defining the optimal centralized control of a large-size robot population using finite element approximations for PDEs is difficult if the continuous piece of the collective dynamics is multi-dimensional. Therefore, we explored an alternative way in which the distributions can be evaluated utilizing stochastic processes.

We established a relation between the PDE system solutions defining the optimal control and samples of corresponding stochastic processes, and showed that all the distributions can be computed based on stochastic process samples and expectation evaluations. Our result is illustrated by the given example which shows an exact match between the finite element method PDE solution and the stochastic process based solution.

Generating multi-dimensional stochastic processes and utilizing them to evaluate the distributions is beneficial because the method works in any number of dimensions. Moreover, the trajectory generation is much easier to parallelize. Once we have collected trajectory samples, the only necessary operation is to compute an average which estimates expected values corresponding to distribution values.

Although the averaging involves simple operations, we believe that the best computational efficiency can be gained if the stochastic process generation is embedded into a real physical system. By embedding stochastic processes into analog circuits and utilizing them in dedicated processors for computing robot control, multi-dimensional stochastic optimal control problems can be solved efficiently. This is highly relevant for the control of systems with many degrees of freedom, such as multi-robot systems. Our future work will consider the design of the hardware that computes the optimal control utilizing real-world stochastic processes.

Appendix A

Let us introduce the vector $P^{\pi}(t) = [P_1^{\pi}(t), P_2^{\pi}(t), ...P_K^{\pi}(t)]$, with components $P_i^{\pi}(t) = \int_X -\pi_i(x,t)dx$, which due to (12) satisfies $\dot{P}^{\pi}(t) = -F_u'(u(t))P^{\pi}(t)$. If $-\pi(x,t)$ is the state PDF of a time-backward stochastic process, then $\sum_{i\in Q} P_i^{\pi}(T) = 1$ and we also should have $\sum_{i\in Q} P_i^{\pi}(t) = 1, t < T$. The latter means that $\sum_{i\in Q} \dot{P}_i^{\pi}(t) = 0$, i.e.,

$$\sum_{i\in Q} \dot{P}_i^{\pi}(t) = -\sum_{i\in Q}\sum_{j\in Q} [F_u'(u(t))]_{ji} P_i^{\pi} = 0 \tag{33}$$

and for any vector $P^{\pi}(t)$, the above equation is satisfied if

$$\sum_{j \in Q} [F'_u(u(t))]_{ji} = 0 \tag{34}$$

On the other hand, $\rho(x,t)$ is the state PDF and the corresponding vector of discrete state probabilities $P(t)$ obeys (3), which means that

$$\sum_{j \in Q} [F_u(u(t))]_{ji} = 0 \tag{35}$$

Thus, we see that $-\pi(x,t)$ can be considered the state PDF if both conditions (34) and (35) are satisfied simultaneously. In general, this cannot be guaranteed and, therefore, we cannot assume that $-\pi(x,t)$ is the state PDF.

Appendix B

The state PDF evolution for the 1D example in Section 5 is

$$\underbrace{\begin{bmatrix} \frac{\partial \rho_1(x,t)}{\partial t} \\ \frac{\partial \rho_2(x,t)}{\partial t} \\ \frac{\partial \rho_3(x,t)}{\partial t} \end{bmatrix}}_{\frac{\partial \rho(x,t)}{\partial t}} = \underbrace{(F_u(u(t)) + F_\partial)}_{F(u(t))} \underbrace{\begin{bmatrix} \rho_1(x,t) \\ \rho_2(x,t) \\ \rho_3(x,t) \end{bmatrix}}_{\rho(x,t)}, \quad \rho(x,0) = \begin{bmatrix} 0 \\ 0 \\ \rho_3(x,0) \end{bmatrix} \tag{36}$$

with

$$F_u(u(t)) = \begin{bmatrix} -u_2(t) - u_3(t) & u_1(t) & u_1(t) \\ u_2(t) & -u_1(t) - u_3(t) & u_2(t) \\ u_3(t) & u_3(t) & -u_1(t) - u_2(t) \end{bmatrix} \tag{37}$$

$$F_\partial = diag(-k_1 \frac{\partial}{\partial x}, -k_2 \frac{\partial}{\partial x}, 0) \tag{38}$$

The co-state distribution evolution is

$$\underbrace{\begin{bmatrix} \frac{\partial \pi_1(x,t)}{\partial t} \\ \frac{\partial \pi_2(x,t)}{\partial t} \\ \frac{\partial \pi_3(x,t)}{\partial t} \end{bmatrix}}_{\frac{\partial \pi(x,t)}{\partial t}} = \underbrace{-(F'_u(u(t)) - F_\partial)}_{-F'(u(t))} \underbrace{\begin{bmatrix} \pi_1(x,t) \\ \pi_2(x,t) \\ \pi_3(x,t) \end{bmatrix}}_{\pi(x,t)}, \quad \pi(x,T) = -w(x) = - \begin{bmatrix} 0 \\ 0 \\ w_3(x) \end{bmatrix} \tag{39}$$

References

1. van den Broek, B., Wiegerinck, W., Kappen, B.: Graphical model inference in optimal control of stochastic multi-agent systems. Journal of Artificial Intelligence Research 32, 95–122 (2008)

2. Cassandras, C., Lafortune, S.: Introduction to Discrete Event Systems. Kluwer Academic Publ. (1999)
3. Correll, N., Martinoli, A.: Multi-robot inspection of industrial machinery-from distributed coverage algorithms to experiments with miniature robotic swarms. IEEE Robotics and Automation Magazine 16(1), 103–112 (2009)
4. Davis, M.H.A.: Piecewise-deterministic markov processes: A general class of non-diffusion stochastic models. Journal of the Royal Statistical Society, Series B 46(12), 353–388 (1984)
5. Fattorini, H.O.: Infinite Dimensional Optimization and Control Theory. Cambridge University Press (1999)
6. Gillespie, D.: Exact stochastic simulation of coupled chemical reactions. Journal of Physical Chemistry 81(25), 2340–2361 (1977)
7. Hamann, H.: Space-Time Continuous Models of Swarm Robotic Systems: Supporting Global-to-Local Programming. Springer (2010)
8. Ho, Y.C., Bryson, A.E.: Applied Optimal Control: Optimization Estimation and Control. John Wiley & Sons (1975)
9. Hogg, T.: Coordinating microscopic robots in viscous fluids. Autonomous Agents and Multi-Agent Systems 14(3), 271–305 (2007)
10. Hsieh, M.A., Halasz, A., Berman, S., Kumar, V.: Biologically inspired redistribution of a swarm of robots among multiple sites. Swarm Intelligence 2(2-4), 121–141 (2008)
11. Hu, J., Lygeros, J., Sastry, S.S.: Towards a Theory of Stochastic Hybrid Systems. In: Lynch, N.A., Krogh, B.H. (eds.) HSCC 2000. LNCS, vol. 1790, pp. 160–173. Springer, Heidelberg (2000)
12. Kappen, H.J.: Linear theory for control of nonlinear stochastic systems. Physical Review Letters 95(20) (2005)
13. Kloeden, P.E.: Numerical solution of stochastic differential equations. Springer (1999)
14. Kushner, H.J., Dupuis, P.: Numerical Methods for Stochastic Control Problems in Continuous Time. Springer (2001)
15. Lerman, K., Martinoli, A., Galstyan, A.: A Review of Probabilistic Macroscopic Models for Swarm Robotic Systems. In: Şahin, E., Spears, W.M. (eds.) Swarm Robotics 2004. LNCS, vol. 3342, pp. 143–152. Springer, Heidelberg (2005)
16. Martinoli, A., Easton, K., Agassounon, W.: Modeling swarm robotic systems: A case study in collaborative distributed manipulation. Int. Journal of Robotics Research 23(4), 415–436 (2004)
17. Milutinović, D., Lima, P.: Modeling and optimal centralized control of a large-size robotic population. IEEE Transactions on Robotics 22, 1280–1285 (2006)
18. Milutinović, D., Lima, P.: Cells and Robots: Modeling and Control of Large-Size Agent Populations. Springer (2007)
19. Milutinović, D., Lima, P., Athans, M.: Biologically inspired stochastic hybrid control of multi-robot systems. In: Proceedings of the 11th International Conference on Advanced Robotics (2003)
20. Parker, L.E.: Multiple mobile robot systems. Springer Handbook of Robotics, pp. 921–941 (2008)
21. Robert, C., Casella, G.: Monte Carlo Statistical Methods. Springer (2004)
22. van der Schaft, A.J., Schumacher, J.M.: An Introduction to Hybrid Dynamical Systems. Springer (2000)
23. Stengel, R.F.: Optimal control and estimation. Dover (1994)
24. Šiljak, D.D.: Large-scale dynamic systems: stability and structure. Dover (2007)

Robust Multi-robot Team Formations Using Weighted Voting Games

Prithviraj Dasgupta and Ke Cheng

Abstract. We consider the problem of distributed multi-robot team formation including the dynamic reconfiguration of robot teams after encountering obstacles. We describe a distributed robot team reconfiguration algorithm, DYN-REFORM, that uses a game-theoretic technique of team formation called weighted voting games(WVGs) along with a flocking-based formation control mechanism. DYN-REFORM works without explicit knowledge of global features such as the presence of obstacles in the environment or the number and location of all other robots in the system. It uses the locally computed metrics of each robot in the team to determine whether a team needs to split or two teams need to merge during reconfiguration. We have tested team reconfiguration using the DYN-REFORM algorithm experimentally within the Webots simulator using teams of e-puck robots of different sizes and with different obstacle geometries. We have also shown that using robots coordinated with the DYN-REFORM algorithm for a distributed area-coverage application improves the coverage performance.[1]

1 Introduction

Distributed formation control of multi-robot teams is an important research direction in robotics that is used in various applications such as convoying or escorting robots or humans for security-related applications, area coverage or clearance for demining, agricultural and other applications, cordoning off regions in hazard control situations, etc. Recently, the problem of robust robotic team formation has gained considerable research interest [6, 9, 10, 14]. Frequent occlusions during the motion of a robot team cause it to repeatedly reconfigure by changing its direction of movement and getting all the team members into a new pose so that it can continue its motion. Reconfiguring the robot team is a costly operation that expends

Prithviraj Dasgupta · Ke Cheng
Computer Science Department, University of Nebraska, Omaha, NE 68182, USA
e-mail: {pdasgupta,kcheng}@mail.unomaha.edu

[1] This research has been sponsored by the U.S. Dept. of Defense, Office of Naval Research as part of the COMRADES project, grant no. N000140911174.

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 373–387.
springerlink.com

time, consumes robots' energy to communicate with each other and culminates in reduced efficiency of the operation the robots are performing as a team. Therefore, it makes sense to investigate techniques that would reduce this overhead by avoiding inefficient team reconfigurations or by postponing reconfigurations when possible.

In this paper, we posit that robust formation maintenance can be achieved on robot teams if their configuration is dynamically adapted to combine into larger teams or split into smaller teams depending on different environmental and operational conditions. To achieve this, we use a technique based on coalition game theory called a weighted voting game(WVG). This technique provides a structured way to determine the best configuration of a robot team after encountering an obstacle by considering the recent efficiency of the robots constituting the team in performing their desired operation. Using the robots' operation efficiency ensures that the size of a reconfigured robot team after splitting or merging is proportional to the amount of free space in the vicinity of the team, and each reconfigured team can continue its operation or motion without being impeded by obstacles. Adapting the size of robot teams dynamically in this manner helps to improve their efficiency of performing operations, as illustrated in this paper through a distributed area coverage application. We have tested the operation of our proposed game theoretic team reconfiguration algorithm, called DYN-REFORM, experimentally on e-puck robots within the Webots simulator for dynamically reconfiguring teams at obstacles with different geometries. We have also measured an improvement of $5-10\%$ in a distributed area coverage application, obtained by using DYN-REFORM as compared to an algorithm where robot teams do not reconfigure dynamically.

Related Work. Much of the research on formation control with multi-robot teams [1, 8, 10] has been based on Reynolds' model for the mobility of flocks[12]. In [2], the authors describe three reactive behavior-based strategies for robot teams to move in formation, viz., unit center-referenced, neighbor-referenced, or leader-referenced. In contrast to these approaches, Fredslund and Mataric[7] describe techniques for robot team formation without using global knowledge such as robot locations, or the positions/headings of other robots, while using little communication between robots. Complementary to these approaches [6, 14] have used a combination of graph theory and control theory-based techniques to effect multi-robot formations. Our previous work on multi-robot formation [3] uses techniques where a team simply reverses its direction to avoid obstacles or other teams, without dynamically reforming, merging with other teams or splitting into smaller teams, while in [4], we have described preliminary results for team splitting only, using weighted voting games.

2 Dynamic Team Reconfiguration and Weighted Voting Games

We have used a flocking-based, leader-referenced formation control algorithm[12], to maintain a specific formation among the robots in a team while in motion, as shown in Figure 1(a) and described in [3]. Each robot in a team is given a local

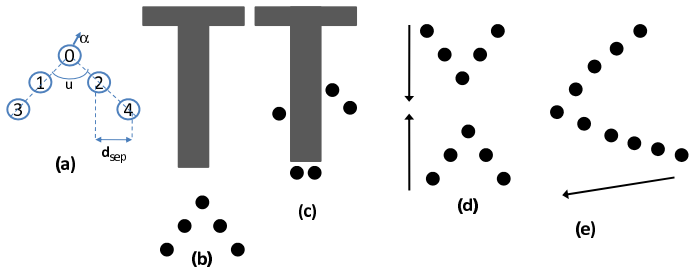


Fig. 1 (a) A robot team showing the position identifiers of each robot. The angular separation in the team is u, the separation between adjacent robots is d_{sep} and α is the heading of the team. (b)-(c) A scenario where a single team in formation encounters a T-shaped obstacle and needs to split. (d)-(e) A scenario where two teams in close proximity of each other encounter each other and need to merge.

identifier with '0' as the leader robot's identifier and odd and even numbered identifiers for the follower robots on either side of the leader. The shape of the team can be controlled by varying the angle u to transform it, for example, from a wedge shape to a line shape. Each robot has a specified separation d_{sep} that it must maintain from its neighbors. The leader robot has a predetermined direction α that it wants to move the team in. At the end of each time step, the leader robot determines the position that it should reach at the end of the next time step so that it can continue its desired motion. The leader robot also calculates the positions for each of the follower robots for the next time step relative to their current positions, so that the formation of the team is maintained. The leader robot then communicates the desired position information to each of the follower robots. Finally, the leader and follower robots start moving towards their respective designated positions.

A principal problem with flocking-based formation control is that it can fail if the leader or a follower robot encounters an obstacle that impedes its motion to the desired position for the next time step. The problem of obstacle avoidance while moving in formation is accentuated if robots are not be able to perceive the obstacle boundary before planning their motion for the next time step. For example, in the scenario illustrated in Figure 1(b), it might be inefficient or impossible to continue moving a robot team in formation when the motion of all or some of the team members gets impeded. In such inefficient scenarios, it would be beneficial to reform the members of a team into new teams by splitting the original team (Figure 1(c)). Complimentary to the team splitting scenario, there can be scenarios when two robots teams in close proximity of each other (e.g., Figure 1(d)) could improve some application-specific performance metric such as the sensor diversity, computational capability, or amount of area coverage achieved by each team, if they combined into a single team. In such scenarios, it would be beneficial to merge some or all of the members of two robot teams into a single team, as shown in Figure 1(e). We approach this problem of merging and splitting robot teams as the robot team reconfiguration problem. Reconfiguring a robot team can be viewed as a problem

of finding the partition that is the best for all the participating robots. The scenarios shown in Figures 1 (b)-(e) illustrate that a single, hard-coded splitting or merging rule cannot be guaranteed to be the best partition in all scenarios, and, therefore, the partitioning of the robot teams has to be done dynamically. The branch of micro-economics dealing with cooperative or coalitional games [11] uses concepts from utility theory and rational behavior of humans to determine rules for solving the partition problem. Coalition games are particularly attractive for our problem for two reasons. First, they can ensure that the solution is stable, or in other words, it is acceptable to all the participants. Secondly, the players, or robots in our case, do not have to be explicitly informed if a split or a merge is the best thing to do. The rules of the coalition game calculate the set of robots that are incentivized to remain together, based on the performance of each robot in the recent past.

The most important factor in a coalition game is determining the performance that a robot in a team has had because this performance determines which robots will remain together by forming a coalition. The robots participating in a coalition game share their individual performance values with each other before the game. The performance value reported by a robot should reflect its individual efficiency in performing its operation or role while participating in the team. For example, in a team formation setting, it could reflect the distance and time for which the robot has not deviated from its designated position in the formation over some finite time window in the recent past. Metrics from the application domains for the robots could also be incorporated into a robot's performance value. For example, if the robots are performing distributed area coverage, then the performance value could include the area of previously uncovered region that a robot has covered in the recent past. The performance value of a robot represents its 'power' in a coalition and is called the robot's **weight** in the game. For our problem, we have used a suitable and succinct representation of a coalition game called a weighted voting game(WVG). The main parameters of a WVG are the following:

- R Set of players or robots interested in forming a coalition
- $\mathscr{R}$ Set of possible partitions among the members of the set R, $\mathscr{R} = 2^R$. Each member of the set $\mathscr{R}$ is called a coalition of robots and denoted by $C_j : j = 1..\mid \mathscr{R} \mid$
- w_i weight of robot $r \in R$
- Q quota or threshold of the WVG. A coalition C_j of robots becomes a winning coalition if the sum of the weights of the players exceeds the quota,
- v value function that denotes whether a a coalition $C \in \mathscr{R}$ is a winning coalition or not, i.e., $v(C) = 1$ if $\sum_{i \in C} w_i >= Q$, and $v(C) = 0$, otherwise.

A few additional concepts in WVGs are useful for the formulation of our problem. A **veto player** is a player such that if the player is excluded from a coalition, that coalition cannot be a winning coalition anymore. As an example. consider a WVG with 4 players A, B, C and D with weights $4, 2, 1$, and 1 respectively. For this WVG, let quota $Q = 5$, that is any coalition must have a combined weight of at least 5 to be a winning coalition. The set of winning coalitions for this WVG are $\{A,B\}, \{A,C\}, \{A,D\}, \{A,B,C\}, \{A,B,D\}, \{A,C,D\})$ and $\{A,B,C,D\}$. This makes

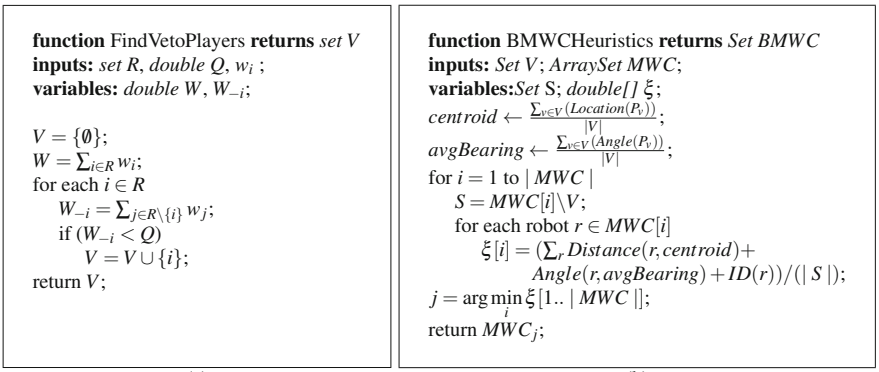


Fig. 2 (a) Algorithm to find veto players in a WVG. (b) Algorithm to find the best minimum winning coalition (BMWC) from a set of MWCs.

A a veto player because it is present in all the winning coalitions. WVGs can have more than one veto player or no veto players. For example, if we change Q from 5 to 7 both A and B become veto players, while if we change Q from 5 to 2, none of the players is a veto player anymore. We use the notation V to denote the set of veto players. The minimum set of players that can get enough combined weight among themselves to get to the quota is called the **minimum winning coalition(MWC)**. In the example above with $Q = 5$, there are three MWCs - $\{A,B\},\{A,C\},\{A,D\}$. MWCs are important because they imply that players in a MWC will not deviate from the coalition they are in because they cannot improve the benefit that they receive by forming a different coalition or a sub-coalition. This makes MWCs stable coalitions which are guaranteed not to break off after the coalition is formed.

The problem solved in a WVG is to identify a set of players that form a minimum winning coalition. This can be achieved in three steps as given below:

1. Identify all the veto players in R, because the veto players, if any, must be there in every winning coalition. The algorithm for identifying the veto players is based on the definition of veto players as players whose exclusion causes any coalition of the remaining players to lose. In other words, the combined weight of players excluding the veto player would fall below the quota. Our *FindVetoPlayers* algorithm shown in Figure 2(a), uses this concept to calculate set of veto players V. It has linear time complexity as it has to inspect each player from the set of players R for checking if it is a veto player or not.

2. Identify all the MWCs, by adding the minimum number of non-veto players to each set of veto players identified in step 1 above. Let w_v denote the combined weight of the veto players found in step 1. Then, $Q' = Q - w_v$, denotes the deficit in combined weight that should come from the non-veto players to reach the quota and form an MWC. Our objective then becomes to determine the set of players from the set $R\backslash V$ that can together reach a combined weight of Q'. This problem is a

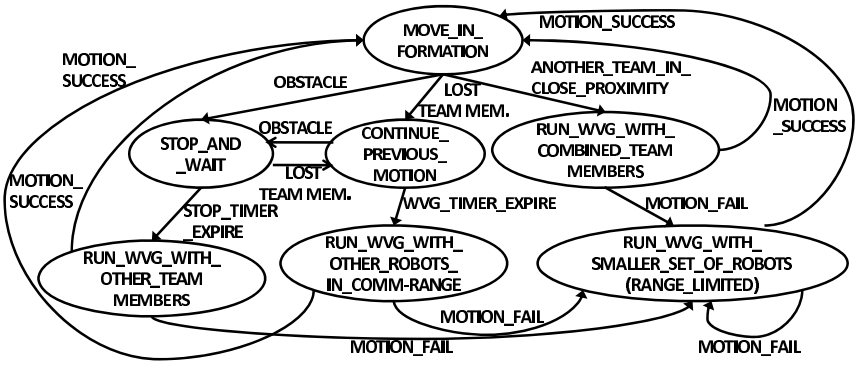


Fig. 3 State transition diagram for a robot participating in the dynamic reconfiguration algorithm

simplified version of the subset sum problem [5], with the relaxation that we need to find the smallest subset of players that is able to reach a combined weight of at least Q', (instead of exactly Q' of the subset sum problem). We have used a greedy method to solve this problem that has a quadratic time complexity in the worst case. The output of this algorithm is the set of MWCs.

3. The conventional solution to a coalition game outputs multiple MWCs, as calculated in step 2 above. However, only a single robot team needs to be selected from those MWCs as the new, reconfigured team to be formed. To make this selection, we measure the eligibility of an MWC towards forming a robot team using a heuristic-based fitness function ξ. For designing this heuristic function, we first consider the pose of the veto players because the veto players must be included in the final winning coalition. We calculate the centroid of the locations of the veto players and the average of the bearing between them. Then, for each of the non-veto players in each MWC, we calculate the distance and relative bearing with the centroid and average bearing of the veto players. If there are still any ties remaining, we use a prime number calculated from the robot id to make the value of ξ unique. The minimum of the ξ values for each MWC gives the best MWC. This algorithm is shown in Figure 2(b) and it has polynomial running time because it takes $O(|R|^2)$ steps in the worst case to calculate the value of ξ for each non-veto robot in each MWC. Integrating all the three steps, the worst case time complexity of running a WVG among R robots is $O(|R|) + O(|R|^2) + O(|R|^2) = O(|R|^2)$.

DYN-REFORM Algorithm. The DYN-REFORM algorithm realizes dynamic team reconfiguration by integrating the WVG algorithm and the flocking-based formation control algorithm. This integration is important and challenging because the WVG works only with a performance value or weight for each robot while the formation control algorithm relies on operational conditions such as presence of obstacles, proximity of robots, etc. Before running the WVG, the DYN-REFORM algorithm provides methods to determine the set or subset of robots from a team that will

participate in the WVG. After a coalition has been computed by the WVG, the DYN-REFORM algorithm provides mechanisms to handle situations where the computed coalition might not be realizable by the formation control algorithm (e.g., because of occlusions that prevent robots in a coalition from reaching their designated positions in the new team). It also specifies the operation of the robots that are excluded from the winning coalition after running the WVG.

The operation of the DYN-REFORM algorithm is summarized by the state transition diagram of a robot shown in Figure 3. First, we consider the case when a team moving in formation encounters an obstacle and could possibly have to split. When any member of the team encounters an obstacle, it enters into a STOP_AND_WAIT state for a certain time period given by the value of a timer called STOP-TIMER. When this timer expires, the robot runs a WVG. Other team members that encountered an obstacle and stopped before the STOP-TIMER expires, possibly in the vicinity of the robot that is running in the WVG, are included as participants in the WVG. In certain scenarios, only some members of a robot team might encounter an obstacle while other robots in the same team do not. The team members that do not encounter the obstacle continue their previous motion (CONTINUE_PREVIOUS_MOTION state) by moving in a straight line. Because the original team has lost some of its members due to an obstacle, it would make sense for the robots continuing their motion to try and reconfigure with other robots in the system. To achieve this, these robots schedule to run a WVG at some time in the future by starting a timer called the WVG-TIMER. When the WVG-TIMER expires on a robot, it runs a WVG including the robots that are within its communication range.

After the WVG has determined the robots comprising the best minimum winning coalition (BMWC), the robot that has the highest weight in the BMWC is selected as their leader robot. The leader robot then selects a new position and heading, usually in the direction opposite to which the obstacle that caused the reconfiguration was sensed. It then starts running the flocking-based formation algorithm to get the follower robots in their desired positions and start moving together as a team in formation. In certain cases, for example, when the vicinity of the robots forming a coalition is occupied by obstacles, the coalition of robots calculated by a WVG might not be amenable to get into formation and move together as a team. When this happens the robots that are not able to get into the desired position reattempt to get into their desired positions for NUM-FORMATION-REATTEMPT iterations. At the end of the reattempts, the robots that managed to get into their desired position for the new team, exclude the unsuccessful robots from the team and continue their motion. The unsuccessful robots attempt another WVG among themselves. If these robots are unsuccessful to form a team after NUM-WVG-REATTEMPT successive WVGs (and included formation reattempts), they continue to move individually using Braitenberg motion. Also, after running a WVG, if there are some robots that are not included as part of the best minimum winning coalition, they continue to move individually using Braitenberg motion until they encounter another team and possibly get assimilated with that team after running a WVG.

When a robot moving individually or a robot team gets within close proximity of another robot team, they run a WVG with the combined team members as the participating robots. Finally, to avoid identical yet repetitive calculation of the BMWC in a WVG by all the participating robots, we have selected the robot with the lowest local identifier to run the WVG. This robot receives the weights from all the participating robots and after running the WVG reports the outcome of the WVG to all the participants.

3 Experimental Results

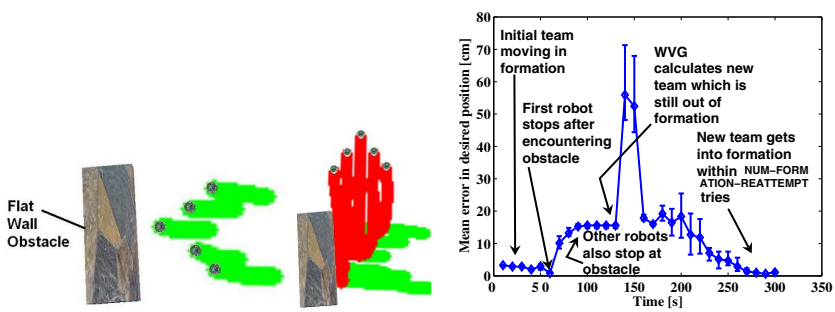
We have evaluated our multi-robot team reformation using e-puck robots on the Webots simulator. The e-puck robot has a diameter of 7 cm. Each wheel is 4.1 cm in diameter and is capable of a maximum speed of about 12 cm/sec. We have used the following sensors that are available on the e-puck robot: (1) Eight infra-red distance distance sensors to detect of obstacles in a range of 7 cm, and, (2) Bluetooth capability for wireless communication. Each e-puck robot is also provided with a local positioning system using a GPS and a compass node within Webots to enable it to determine its position in the environment within a 2-D coordinate system. For all team formations in our simulations, the robots forming the team had to get into a wedge-shaped formation with an angular separation $u = 60$ degrees. The inter-robot separation between a pair of follower robots in a team is set to 15 cm. [2] For multi-team merging scenarios, the distance between two team leaders to run the WVG-based team merging algorithm is set to 70 cm so that the team leaders are well within the communication range allowed by Bluetooth, but not too close so that the teams might collide with each other. For our experiments, one time step is defined as the time required by a robot to cover an area equal to its own footprint. For a robot speed of 2.8 cm/sec, the value of the time step is calculated as 2.5 sec. The performance function used to compute the weight value for each robot for participating in the WVG uses the mean error in the robot's desired position and the area of previously uncovered region covered by the robot, over the last 25 time steps. Based on this function, for the DYN-REFORM algorithm, the duration of the WVG-TIMER is set to 25 time steps so that the robots do not continue in inefficient configurations for long durations. The STOP-TIMER is set to 15 time steps so that robots that do not encounter an obstacle and do not need to stop get a sufficient time window to move away from the stopped robots and possibly form a new team. The number of reattempts by a newly formed team to get into formation after running a WVG (NUM-FORMATION-REATTEMPT) was set to 5 to balance between splitting teams very frequently and inefficiently trying to form a team between robots when it is physically not possible (e.g., due to an obstacle between two sets of robots trying to form a single team). To prevent the formation of excessively large teams that have a high communication and computation overhead in the

[2] The angular and inter-robot separation could be set to different values to get different shape of a robot team.

DYN-REFORM algorithm, we took two steps. First, we set the quota for a WVG at $0.9\times$ (sum of the weights of participating robots). This guarantees that robots with very poor performance, and, consequently low weights get excluded from the new team after reconfiguration. Secondly, we limited the maximum team size in our experiments to 7 robots. If the winning coalition calculated by the WVG has more than 7 robots, the excess robots that have the lowest weights in the coalition are removed from the team and move individually without forming a team, until they merge with another team at a later stage. For quantifying the efficacy of team formation, we used the deviation in positions of the team members from a perfect formation. To measure this, we calculated the mean error in the positions of the follower robots at intervals of 10 sec. over the duration of the experiment. All results were averaged over 10 simulation runs.

3.1 Team Reformation Experiments

For our first set of experiments, we verified the performance of the DYN-REFORM algorithm. We considered three types of obstacles - a flat wall obstacle, a non-uniform wall obstacle and a perpendicular, narrow wall obstacle, as shown in Figures 4(a), 5(a) and 6(a). For this set of experiments, we have traced the trail of the robots within Webots to show their motion before, during and after reconfiguration. For the flat wall obstacle, the leader robot encounters the wall first, enters into the STOP_AND_WAIT state and starts the STOP-TIMER, according to the DYN-REFORM algorithm. The follower robots successively encounter the wall and also enter the STOP_AND_WAIT state and start their individual STOP-TIMERs. The leader robot's STOP-TIMER expires first and it runs the WVG including the other robots that are stopped in its vicinity as the WVG's participants. As an example of the weight and quota values used in this WVG, one of the reported runs for this



(a) (b) (c)

Fig. 4 (a) Initial configuration of a team of 5 robots moving in a wedge-shaped formation. (b) Reformed team moving in new direction after encountering a flat wall obstacle. (c) Mean error in the desired position of the robot team during the reconfiguration.

experiment had weights of the five robots as $0.72, 1.0, 0.96, 1.16$ and 1.24 respectively and quota $Q = 0.9 \times \sum_i w_i = 4.57$. The leader robot of the team had the worst performance recently because it encountered the obstacle first and stopped, giving it a weight 0.72 in the WVG. The fringe robots that stopped last have the highest weights of 1.16 and 1.24 respectively. The BMWC contains all the five robots in this scenario. which means that all the robots should stay together in the new team. The new leader robot then selects a position in a direction opposite to the direction in which it encountered the wall, a pose or heading for the new team, and communicates the desired position of the follower robots to get in formation in the new team. After the formation succeeds, the new team starts moving as shown in Figure 4(b). Figure 4(c) shows the mean error in the position of the robots during the entire reconfiguration process. Initially, between $0 - 50$ seconds, the team is moving in formation before encountering the obstacle, and this is shown by a low mean error of about 3 cm in the position of the robots. When the robots successively start encountering the obstacle, between $50 - 100$ seconds, the error between their actual positions and their desired positions to remain in formation increases. This happens because when the robots successively stop at the obstacle they form straight line along the boundary of the wall, while their current formation, in which they had been before encountering the obstacle, requires them to form a wedge shape. While the WVG runs, the robots are stopped and their mean error in position remains unchanged, as seen between $100 - 125$ seconds. After determining the best minimum winning coalition, the robots get a new formation and the mean error in the position of the robots decreases back to the low value of around $3 - 4$ cm over $125 - 250$ seconds. We notice that the mean error suddenly spikes to about 60 cm around 150 seconds when the new team calculated by the WVG attempts reformation. This happened because all the robots stopped at the flat wall forming a horizontal line and they are in close proximity of each other. The robots themselves occlude each others paths when they try to get into a new formation. However, within 20 seconds, which was within NUM-FORMATION-REATTEMPT$= 5$ iterations, the formation control algorithm is able to resolve this problem and the robots are able to regain formation.

The scenario with robots encountering the non-uniform wall obstacle, shown in Figures 5(a) and (b), is very similar to the flat wall case. The main difference is that because of the non-uniform surface of the wall, the robots along the fringes of the former team persist longer in the CONTINUE_PREVIOUS_MOTION state than in the flat wall case. This behavior is also seen in the mean error of the robots during reconfiguration, shown in Figure 5(c). The mean error in the position of the robots w.r.t. their positions in the previous formation becomes larger than the flat wall case because the robots move farther from their erstwhile desired positions in formation into the clefts of the wall. However, in this case, we do not see any significant spikes during after the WVG when the new team is getting into its formation because the robots have dispersed further from each other because of the non-uniform surface of the wall. Therefore, they do not occlude each other's path while getting into formation.

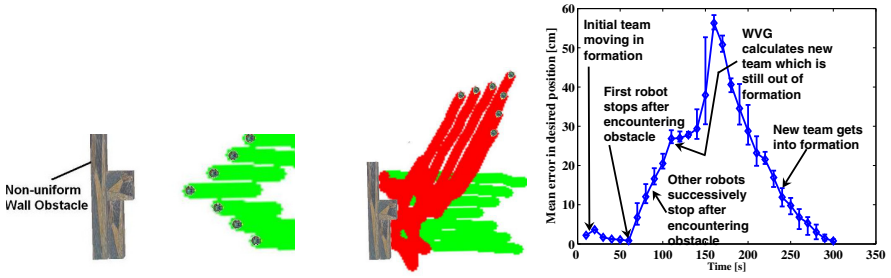


Fig. 5 (a) Initial configuration of a team of 7 robots moving in a wedge-shaped formation. (b) Reformed team moving in new direction after encountering a non-uniform wall obstacle. (c) Mean error in the desired position of a robot team during the reconfiguration.

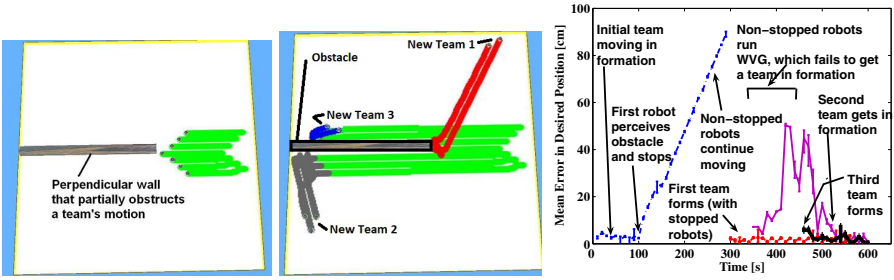
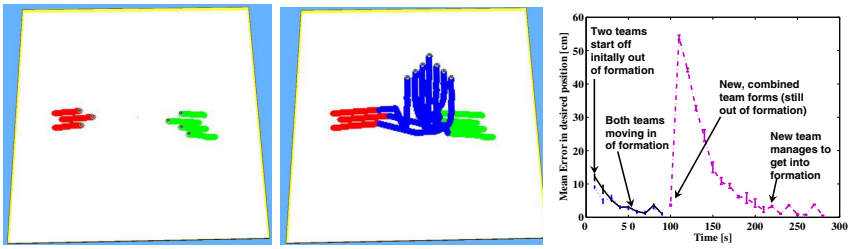


Fig. 6 (a) Initial configuration of a team of 7 robots moving in wedge-shaped formation. (b) Reformed team moving in new direction after encountering a narrow obstacle that causes the team to split. (c) Mean error in the desired position of a robot team during the reconfiguration.

Figures 6(a) and (b) show a scenario where a robot-team encounters a wall perpendicular to its heading that obstructs the team partially. In this scenario, only two robots at the center of the 7 robot team encounter the obstacle, enter into the STOP_AND_WAIT state and start their STOP-TIMERs. The rest of the team members do not encounter the obstacle and enter into the CONTINUE_PREVIOUS_MOTION state while starting the WVG-TIMER. We notice that the obstacle forces two sets of team members to continue their motion along the two sides - above and below the obstacle. When the STOP-TIMER expires for the two stopped robots, they run a WVG. Since there are no other stopped robots in their vicinity, the stopped robots form a new team among themselves and continue moving in a new direction. When the WVG-TIMER expires for the robots that continued their motion, they run a WVG. These robots are in the vicinity of each other and the BMWC contains all these robots. However, the BMWC contains two sets of robots that are located on opposite sides of the obstacle and can never get into a



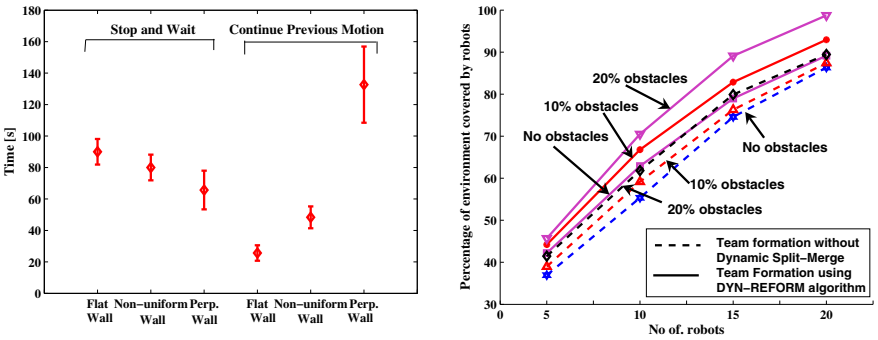
(a) (b) (c)

Fig. 7 (a) Initial configuration of two teams of 3 and 4 robots moving in wedge-shaped formations. (b) Reformed team moving in new direction after encountering each other and merging into a new team. (c) Mean error in the desired position of a robot team during the reconfiguration.

single formation. Therefore, although the WVG succeeds, the new team fails to form within NUM-FORMATION-REATTEMPT(= 5) reattempts. The DYN-REFORM algorithm then causes the robots that were unable to get in formation to to run another WVG and form a new team. Each team gets into its desired formation and continues its movement. The graphs in Figure 6(c) show that the mean error in the desired position of the robots from their erstwhile team keeps increasing because some of the robots do not encounter the obstacle and therefore, do not stop. When the WVGs run, three teams are formed at different times based on the STOP-TIMER expiry (first team) and WVG-TIMER expiry followed by WVG reattempt (second and third team), as shown in Figure 6(c).

Figure 7(a)-(c) show the scenario of two robot teams moving towards each other and merging using a WVG, and the mean error in robot positions during the reconfiguration process for this scenario. Before encountering each other, the two teams are moving in formation and consequently, the mean error in the desired position of the robots is low. When the teams encounter each other (team leaders separated by a distance of 70 cm or less) they stop and run a WVG. Examples of quota and weight values from one experiment run show that the weights of the robots in the two teams are $\{1.28, 1.2, 1.2, 1.2, \}$ and $\{1.32, 1.12, 1.16\}$ respectively, while the quota $Q = 0.9 \times \sum_i w_i = 7.6$. The WVG outputs the combined set of all robots in both teams as the BMWC, implying that the two teams have to merge into a new team. As in the case of reformation with the flat wall obstacle, we notice a large spike in the mean error of the positions of the robots around 120 seconds in Figure 7(c) because the robots from the combined teams get in each others' way while forming the new team. However, finally, they manage to get into their desired position before NUM-FORMATION-REATTEMPTS and move together as one team.

Figure 8(a) shows the mean times spent by the robots in the STOP_AND_WAIT state and the CONTINUE_PREVIOUS_MOTION state for the three different obstacle types. The time in the STOP_AND_WAIT state is the highest for the flat wall because all robots stop at the flat wall, while it is the lowest for the perpendicular wall



(a) (b)

Fig. 8 (a) Time spent by the robots in the STOP_AND_WAIT state and CONTINUE_PREVIOUS_MOTION state of the DYN-REFORM algorithm for the three different types of obstacles. (b) Percentage of the environment covered by a set of 5 robots initially configured as a team without and with DYN-REFORM algorithm. The environment is 2×2 m^2 with $0\%, 10\%$ or 20% of the area of the environment occupied by obstacles. Each experiment was run for a period of 30 mins. Error bars were omitted for legibility.

where only two robots stop and the rest of the team continues its motion. A complementary trend happens for the time spent in the CONTINUE_PREVIOUS_MOTION state with a very low value when all team members stop at the flat wall, and a higher value in the perpendicular wall case when some team members never encounter the wall and continue moving until their WVG-TIMER expires and they run a WVG.

Figure 8(b) shows the improvement in coverage achieved using the DYN-REFORM algorithm by a set of 5 robots, initially configured as a team. The robots are placed within a 2×2 m^2 walled environment with 0%, 10% or 20% of the total area of the environment occupied by obstacles. The coverage algorithm used by a robot in a team maintains only the recent coverage information (over the last 25 time steps) and passes it to the team leader. A leader robot exchanges this recent coverage history of its team with other leader robots within its communication range to avoid covering regions that have been recently covered by other robot teams. We observe that the robots using the DYN-REFORM, because of their capability to dynamically reconfigure at obstacles and avoid inefficient configurations, are able to improve coverage by about 5% when there are no obstacles, and about $7 - 10\%$ when there are obstacles in the environment.

4 Conclusions and Future Work

In this paper, we have described a novel dynamic reconfiguration technique based on weighted voting games called DYN-REFORM, that allows robust and distributed multi-robot team formations. Simulation results on Webots with e-puck robots show

that the DYN-REFORM algorithm allows robots to gracefully split and merge into new teams on encountering other teams or obstacles. Preliminary experiments on physical e-puck robots have shown that our simulation results hold fairly accurately also in hardware - area coverage using DYN-REFORM algorithm is $5-10\%$ better than coverage using fixed robot teams in a 2×2 m^2 arena with different types of obstacles. The robots in our system use absolute positioning information using a GPS and compass for the area coverage application. For a scenario that requires team reformation only without requiring to record coordinates of covered regions, the DYN-REFORM algorithm can work with only relative position information of the robots participating in a WVG, enabled, for example, with IR-based range and bearing sensors. For our experiments, the quota for forming a winning coalition in a WVG was kept constant at a fraction of the team's total weight to lean towards forming teams around $5-7$ robots. A future problem we are investigating is to dynamically adapt the value of this quota, so that the team size can be automatically changed in cluttered or open environments perceived by the robots. Yet another direction is to integrate more perceptual data from the environment, e.g., from laser scans, camera, etc., into the DYN-REFORM algorithm to improve its performance. Finally, a future direction we are investigating is to use teams of heterogeneous robots with diverse sensor capabilities and how to integrate the robot heterogeneity into the WVG framework.

References

1. Bahceci, E., Soysal, O., Sahin, E.: Review: Pattern formation and adaptation in multi-robot systems. CMU Tech. Report no. CMU-RI-TR-03-43 (2003)
2. Balch, T., Arkin, R.: Behavior-based formation control of multi-robot teams. IEEE Transactions on Robotics and Automation 14(6), 926–939 (1998)
3. Cheng, K., Dasgupta, P., Wang, Y.: Distributed Area Coverage Using Robot Flocks. In: World Congress on Nature and Biologically Inspired Computing (NaBIC 2009), pp. 678–683 (2009)
4. Cheng, K., Dasgupta, P.: Weighted voting game based Multi-robot team formation for distributed area coverage. In: 3rd Practical and Cognitive Agents and Robots (PCAR) Workshop (co-located with AAMAS 2010), Toronto, Canada, pp. 9–15 (2010)
5. Cormen, T., Leiseron, C., Rivest, R., Stein, C.: Intro. to Algorithms. McGraw Hill (2001)
6. Falconi, R., Gowal, S., Martinoli, A.: Graph Based Distributed Control of Non-Holonomic Vehicles Endowed with Local Positioning Information Engaged in Escorting Missions. In: ICRA 2010, Anchorage, AK, pp. 3207–3214 (2010)
7. Fredslund, J., Mataric, M.: A general algorithm for robot formations using local sensing and minimal comm. IEEE Trans. on Rob. and Auton. 18(5), 837–846 (2002)
8. Gokce, F., Sahin, E.: To flock or not to flock: the pros and cons of flocking in long-range migration of mobile robot swarms. In: AAMAS 2009, pp. 65–72 (2009)
9. Ji, M., Egerstedt, M.: A Graph-Theoretic Characterization of Controllability for Multi-Agent Systems. In: Proc. American Control Conference, New York, NY, pp. 4588–4593 (2007)
10. Kaminka, G., Schechter, R., Sadov, V.: Using Sensor Morphology for Multirobot Formations. IEEE Transactions on Robotics 24(2), 271–282 (2008)

11. Myerson, R.: Game Theory: Analysis of Conflict. Harvard University Press (1997)
12. Reynolds, C.: Flocks, herds and schools: A distributed behavioral model. Computer Graphics 21(4), 25–34 (1987)
13. Rutishauser, S., Correll, N., Martinoli, A.: Collaborative Coverage using a Swarm of Networked Miniature Robots. Robotics and Auton. Systems 57(5), 517–525 (2009)
14. Smith, B., Egerstedt, M., Howard, A.: Automatic Generation of Persistent Formations for Multi-Agent Networks Under Range Constraints. Mobile Networks and Applications Journal 14, 322–335 (2009)

Influence Maximization for Informed Agents in Collective Behavior*

Amir Asiaee Taheri, Mohammad Afshar, and Masoud Asadpour

Abstract. Control of collective behavior is an active topic in biology, social, and computer science. In this work we investigate how a minority of informed agents can influence and control the whole society through local interactions. The problem we specifically target is that a minority of people with a bounded budget for initiating new social relations attempt to control the collective behavior of a society and move the crowd toward a specific goal. Assuming that local interactions can only take place between friends, the minority has to initiate some new relations with the majority. The total cost of new relations is limited to a budget. The problem is then finding the optimal links in order to gain maximum impact on the society. We will model the problem as a diffusion process in a social network. The proof of NP-hardness of the problem for Local Interaction Game model of diffusion is presented. Simulations show that the proposed method surpasses the popular strategies based on degree and distance centrality in performance.

1 Introduction

Influencing society and changing the crowd behavior is one of the oldest ambitions of social science. Social and political sciences pursue strong impact on the society to change the attitude of people and prevail a desired behavior in the society. Socio-physics deals with such problems under the Opinion Formation topic [1]. In economical side this phenomena is known as Viral Marketing [2], [3].

The main problem that has been investigated extensively for attaining manipulation of crowd behavior is finding most influential persons of a society whom we

Amir Asiaee Taheri · Mohammad Afshar · Masoud Asadpour
Social Networks Lab, School of Electrical and Computer Engineering,
University of Tehran, Tehran, Iran
e-mail: {aasiaeet,mafshar,asadpour}@ut.ac.ir

Masoud Asadpour
School of Computer Science, Institute for Research in Fundamental Sciences (IPM),
P.O. Box 19395-5746, Tehran, Iran

* This research was in part supported by a grant from IPM (No. CS1388-4-14).

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 389–402.
springerlink.com

call initiators from now on. Organizational theory calls such influential persons key players [4] and in political science they are called opinion leaders [5].

The idea is that influencing initiators would lead to the greatest possible diffusion of a behavior in a society. In Viral Marketing initiators are influential customers who are selected for direct marketing. Giving free samples or discounts are examples of marketing strategies for motivating initiators which would lead to stimulation of the others for buying the new product. Two-step flow theory in social and political sciences assumes initiators are well connected opinion leaders who channel media information to the masses [6].

All solutions of "K most influential persons" (K-MIP) problem suppose that changing initiators' opinions or behaviors is possible by costing a budget [7][8]. This simplification is not the case for real world problem, since opinion and behavior of people usually cannot be influenced by paying money.

On the other hand, studies in the field of Swarm Intelligence have more robust solutions to this problem. Couzin et al. [9] showed that among a group of foraging or migrating animals only a small fraction of them have proper information about the location of food source, or about the migration route. But these informed agents can guide the whole group through simple social interactions. The bigger the group is, the smaller the fraction of the required informed agents is. Halloy et al [10] showed in real experiments that informed robots in a mixed-society of animals (cockroaches) and robots can control the aggregation behavior of the mixed-society through microscopic interactions.

The strategy followed in this paper is similar. The minority is regarded as informed agents who want to have control on the opinion of the society. They should do this through social interactions that take place between local neighbors, i.e. agents that have direct friendship ties. So the minority has to have friendship relations with the majority or initiate new ties upon necessity. But in realistic situations, the total number of links that the minority can initiate is limited to a number, due to e.g. time or geographical distance. Now, the minority should choose which links to initiate in order to gain maximum impact on the society.

The rest of this paper organized as follow: in Section 2 related works are discussed. Section 3 presents the selected model of diffusion. In Section 4 we convert our problem to an optimization and solve it in Section 5. Finally Section 6 consists of simulations that compare our method with well-known heuristics.

2 Related Work

K-MIP tries to manipulate crowd behavior by directly targeting initiators. There are many models that describe diffusion phenomena by using methods from different domains. Based on the selected diffusion model, method of finding influential persons can vary. In this section different diffusion models are described and K-MIP solution for each of them is presented.

In all models, society is modeled by a directed graph $G = (V, E)$ whose vertices V and edges E are representing individuals and social relations respectively. In some models edges are weighted. Weights are usually interpreted as node v's trust on $N(v)$ which is the set of v's neighbors.

The Linear Threshold Model (LTM) that is rooted in mathematical sociology [11] has been widely used in viral marketing [7]. At the beginning, every node v chooses a random threshold $\theta_v \in [0,1]$ that defines its general tendency to adopt a new belief. The link between nodes v and u has weight $b_{u,v}$ where $\sum_{u \in N(v)} b_{uv} \leq 1$. The process begins with a set of active nodes who has adopted the new opinion. In each time step, a node is activated if the weight of its active neighbors exceeds its threshold. The process stops when no new activation is possible.

The Independent Cascade Model (ICM), a well-known model in viral marketing [7] is originated from interacting particle systems [12]. In ICM each active individual has only one chance for activation of its neighbors. The probability of activation of a node v by its neighbor u is equal to the weight of their social connection and is independent of previous attempts of other v 's neighbors. The process starts with a set of initiators and unfolds until all active nodes have used their chance for activation of the others.

Kempe et al. [7] presents an algorithm for K-MIP when diffusions are LTM or ICM. They assume the same costs for activation of each person and showed that the problem is NP-hard. They exploited the submodularity of the problem structure and presented an $(1-1/e)$ - approximation algorithm.

Voter model is another popular model of social influence. In this model at each time step each node picks one of its neighbors at random and adopts its opinion. Even-Dar and Shapira [8] found the exact solution for K-MIP when the underlying interaction model is voter and cost of marketing each person is identical. Also they presented an FPTAS [8] for the case when each person has different costs.

All mentioned methods tackle the problem of maximizing diffusion in social networks by persuading initiators to adopt a product or accept an opinion. But what if there is no way to convince an individual about changing his behavior the way we want? This is the case especially in changing the opinion of a crowd.

This paper investigates the problem of influence maximization from a different view. The problem formulation is changed to a more realistic one. It is assumed that there exist a minority in the society with a different opinion from the majority who tries to propagate its belief by means of making new social relations. Minority has a limited budget and any new link has a cost. So, the problem is converted to finding the best links to be added by minority under the budget constraint.

3 Diffusion Model

As an underlying diffusion model, Local Interaction Game (LIG) [13] is chosen. LIG simultaneously benefits from rigorous game theoretic background and simplicity. In this model each person is under the influence of his neighbors. The person is active if he adopted the minority's opinion and inactive otherwise.

In each relation, participants benefits only if they coordinate and choose the same action. Table 1 summarizes the payoffs of each player in coordination games. For simplicity the zero payoffs is set for incoordination. Person's preferences and tendencies are distinguished by his name index in Table 1.

Table 1 Payoff table of LIG

v \ u	*active*	*inactive*
active	a_v, a_u	0, 0
inactive	0, 0	b_v, b_u

At the beginning, the society U is inactive except a minority M . The minority intends to activate initiators by adding new links. It can be shown [14] that v become active iff more than $\theta_v = b_v/(a_v + b_v)$ proportion of its neighbors is active. Set A_t is the set of active nodes at time step t .

4 Problem Formulation and Properties

Minority itself can activate a set of nodes A without any cost. Suppose that the effect function $e: 2^M \to 2^A$ finds the set A which is the union of minority and individuals that are activated by the minority up to the end of diffusion. Thus new links must be added to the members of set $U - e(M)$. Suppose that adding a new link has the constant cost, α, and minority's overall budget is B. Also suppose that for activating each node v, $c(\{v\})$ links must be created from minority to it. Then, activating initiator set S of nodes costs:

$$c(S) = \sum_{v \in S, S \subseteq U - e(M)} \alpha c(\{v\})$$

In this paper, α is considered to be one. For computing *c({v})* values, we take following steps. Each node v has neighbors in M and $U - M$ that are called I and J respectively. Assume that v is inactive, i.e. $|I|/|I \cup J| < \theta_v$. Then set $X \subseteq M$, $X \cap I = \varnothing$ should initiate links to v for its activation. These change the inequality to $\frac{|I \cup X|}{|I \cup J \cup X|} > \theta_v$. Since *c({v})* is the minimum size of set X for which the above inequality holds it can be computed as $c(\{v\}) = \left\lfloor \frac{\theta_v |J|}{1 - \theta_v} - |I| \right\rfloor$.

4.1 Optimization Problem

Previous works focused only on K-MIP which is the identical cost MIP [7]. In our problem, each individual's cost can differ from the others, so the problem can be called N-MIP (Non-identical cost Most Influential Person). We define a set

function that takes an initiator set S and maps it to the number of individuals that are going to be activated by the end of the process.

Let $f: 2^{U-e(M)} \to \mathbb{R}$ map the initiators $S \subseteq U - e(M)$ to the number of active nodes at the end of the process. Then N-MIP problem can be viewed as maximizing f subject to the limited budget. Using above definitions the problem is:

Problem: Find set S that maximize function f subject to the cost constraint:

$$S^* = \arg\max_{S \subseteq U - e(M)} f(S) \qquad s.t. \qquad c(S) \leq B$$

First we show this problem is NP-hard for LIG model. Next we claim $f(S)$ is submodular, so we can exploit maximization algorithms for submodular functions.

4.2 NP Hardness of Efficient Link Addition Problem

Kempe et al. [7] showed that finding K-MIP under LTM is NP-hard. Based on their proof, we show that NP-complete Vertex Cover problem is a special case of K-MIP for LIG model which itself is an especial case of N-MIP. For a graph $G = (V, E)$ and integer k Vertex Cover finds a set $S \subseteq V$ that every edge of G has an endpoint in it. If there is a Vertex Cover S of size k in G then $f(S) = |U - e(M)|$. On the other hand this is the only way that for all settings of thresholds one can deterministically activate all society. So Vertex Cover is an especial case of identical cost most influential person for LIG. Based on definition, K-MIP is an especial case of N-MIP. Since it is proved that Vertex Cover is an especial case of K-MIP, N-MIP is also NP-hard.

4.3 Submodularity of f

f is submodular if it satisfies a natural "diminishing returns" property: adding new element to a subset produce gain which is at least as high as adding that element to a superset [15]. Formally f is submodular iff for every $T \subseteq S$, $f(T \cup \{v\}) - f(T) \geq f(S \cup \{v\}) - f(S)$ holds.

Kempe et al. [7] showed that when diffusion model is LTM, f is a submodular function. We show that LIG is an especial case of LTM so f for LIG is submodular too. In LTM each neighbor u of node v can influence it according to the weight b_{uv} such that $\sum_{u \in N(v)} b_{uv} \leq 1$. Thus v would become active in step $t+1$ if $\sum_{u \in N(v) \wedge u \in A_t} b_{uv} \geq q_v$. If b_{uv} is set to $1/|N(v)|$ the inequality changes to $|A_t|/|N(v)|$ which is the condition of v's activation in LIG. So LIG is a special case of LTM when $b_{uv} = 1/|N(v)|$. Therefore f is submodular for LIG.

5 Optimization Algorithm

Up to this point the link addition problem has been converted to a submodular function optimization problem using game theoretic diffusion models. Submodular function optimization is an active field of research in machine learning. Since submodularity arises in many real world optimization problems many advancements has been made during recent years in this old topic [16]. Nemhauser et al. [15] proved that a simple greedy algorithm is within $(1-1/e) \approx 0.63\%$ of maximum when $c(S) = |S|$. For general cost functions [17] showed that for the special case of MAX-COVER problem $(1-1/e)/2 \approx 0.31\%$ approximation guarantee is reachable and a $(1-1/e)$ guarantee can be achieved using partial enumeration. Recently [18] extended their result to general submodular functions and [19] introduced an online boundary for any algorithm.

Since our problem is reduced to maximizing a submodular function subject to a bounded non-identical cost function, we follow [19] and call our algorithm Efficient Link Addition Strategy (ELAS). The greedy approach proposed by Leskovec et al. [19], iteratively adds nodes to the selected set S by choosing a node v that maximizes $f(S \cup \{v\}) - f(S) / c(\{v\})$. This heuristic is an extension of [15] which uses $f(S \cup \{v\}) - f(S)$ as the selection criteria in each iteration. They showed [19] that choosing best results of one of the mentioned heuristic provides a constant factor approximation. Formally, if NIC be the solution of non-identical cost algorithm that uses $f(S \cup \{v\}) - f(S) / c(\{v\})$ and IC be the solution of identical cost algorithm which uses $f(S \cup \{v\}) - f(S)$, it can be proved that:

$$\max\{f(NIC), f(IC)\} \geq \frac{1}{2}(1-\frac{1}{e}) \underset{S \subseteq U-e(M), c(S) \leq B}{\arg\max} (f(S))$$

6 Experiments and Discussion

We have used heuristics from social science that choose individuals with highest degree and betweenness [21] as the initiators, and compared our method's performance with theirs. To show the advantages of ELAS, three different network models were tested (Table 2). For each of them different parameter settings were tested. For each setting 30 networks were built and diffusion was simulated for 30 randomly chosen thresholds. So for each setting 900 simulations were done.

After building a network, every node chose a random threshold. Then some nodes were randomly selected as the minority and were given opinions opposite to the others. Then different strategies for link addition were used and their impacts were measured. The diffusion continued until no new node could be activated. Society was composed of 400 individuals and minority was 10% of them. The budget limit was 40 (i.e. each minority member could initiate one link on average). For each simulation Social Impact of Minority that is the number of active nodes at the end of the simulation, was recorded.

6.1 Budget Impact on Diffusion and Trends of Diffusion

Fig. 1.a illustrates the effect of budget on success of link addition for ER network whose structural details will be discussed later in this section. The vertical axis shows the number of active individuals at the final time-step of diffusion. As expected, increase in budget would increase the social impact of Minority. It is clear that performance of ELAS is better than degree-based and betweenness-based strategies. Further simulations showed this dominance exists for all mentioned network types and their different parameter settings.

Fig. 1.b compares the methods in different time steps for an ER network. Data is gathered for budget 40. Performances of degree and betweenness heuristics' are close to each other. As it is illustrated, ELAS outperform them in all time steps.

6.2 Effect of Network Structure on Diffusion Process

Experimental results along with analytical demonstrations show the better performance of ELAS in comparison with other link addition strategies. In this part, a closer look is taken to ELAS for finding the structural factors that affects its performance. For this goal, different syntactical networks were built and ELAS performance was tested on them.

Table 2 Network models and their parameter list

Network Type	Parameters
Erd s–Rényi (ER)	P: edge probability
Small World (SW)	A: average degree, R: rewiring probability
Scale Free (SF)	A: average degree, S: initial seed

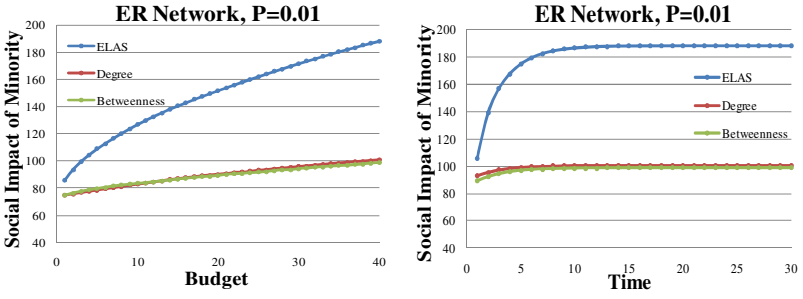


Fig. 1. Comparison of ELAS performance with degree and betweenness based strategy in an instance of ER networks a. At the final time step of simulation. b. During simulation for B = 40.

6.2.1 Erdős–Rényi Network

Erdős–Rényi network model (ER) is the most studied random network model in which the probability of relation between each individual pair is p . Fig. 2 shows the sensitivity of presented method to p . When p is very low the graph is loosely connected which hinders the cascade of influence. As p increases the giant graph component appears and facilitates the diffusion. Since $p = \ln n/n$ is a sharp threshold for the presence of giant component [20] at this point (0.01 for 400 nodes) the effectiveness of the method is maximized. Increasing p creates high degree nodes which are harder to influence. These nodes decrease ELAS influence on the whole network.

Fig 2 also illustrates the difference between ELAS and the better of the two other strategies. As it is clear, the dominance of ELAS decreases as the graph becomes more connected. It can be interpreted as when graph become denser every link addition strategy becomes ineffective.

6.2.2 Scale Free Network

It has been shown [21] that many real world networks are scale free (SF). Based on this, [22] proposed preferential attachment process for generating SF networks. This model has two parameters which are N_0 and k, initial seed of process and average degree of network respectively.

The process starts with N_0 isolated nodes and at every time step a new node is added by making k new links. The probability that a link connects j to node i is linearly proportional to the degree of i [21]: $P(i \to j) = \deg_i + 1 / \sum_l (\deg_l + 1)$ where deg_i is the degree of node i.

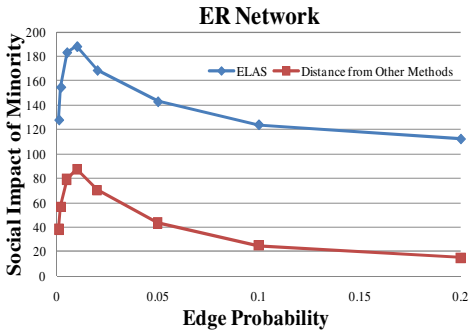


Fig. 2 Effect of network structure on the success of the method for ER network

Fig. 3.a shows the performance of ELAS for different SF networks that have been constructed using different parameters. In this set of experiences 5 value for both N_0 and k have been used. Since for making a SF network we should have

$N_0 \geq k$ no experience has been done for $N_0 < k$ which is shown by zero in Fig 3.a. This reduces the number of different settings for networks from 25 to 15.

Fig 3.b shows the same diagram as Fig 3.a except that it emphasize on the effect of network seed. In each ribbon average degree is constant. In the region of experiences where $N_0 \geq k$, ribbons are flat which shows that N_0 does not have significant impact on ELAS performance. On the other hand ribbons with identical network seed (Fig 4.c) demonstrate that average degree extremely change the number of active individuals at the end of diffusion with inverse relation.

6.2.3 Small World Network

High clustering coefficient (CC) and low average shortest path length (L) are two important characteristics of social relation networks [21]. Small world networks are networks that simultaneously exhibit high CC and low L [23]. Watts and Strogatz model [23] is the most well-known model of small world networks (SW). It has two parameters which are average degree of the network and rewiring probability of edges. The process begins with a ring lattice with *n* vertices and *k* edges per vertex. Then edges are rewired with probability *p*.

Fig. 4.a illustrates the effect of both parameters on ELAS. Fig. 4.b and Fig. 4.c are the same as Fig. 4.a diagram but they illustrate the effect of rewiring probability and average degree respectively. According to flat ribbons of Fig. 4.b, rewiring probability does not have a significant impact on the final result. But it is clear from Fig. 4.c that like previous models, average degree has inverse relation with the final outcome.

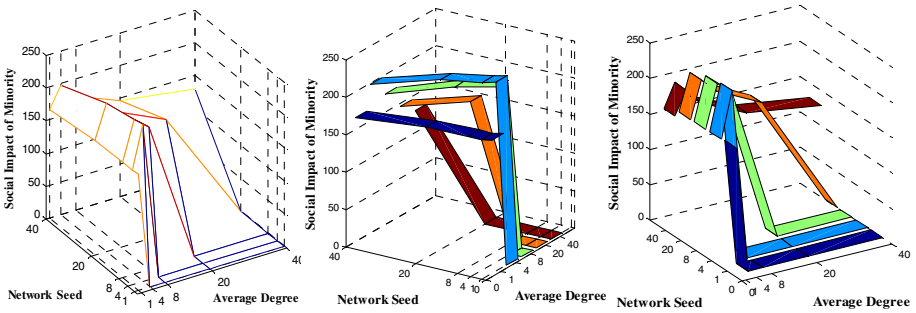


Fig. 3 Effect of network structure parameters on the success of the method for SF networks

6.2.4 Other Structural Factors

At the first glance, result of ELAS in all network structure has inverse relation with the average degree of the nodes. But Fig. 5 shows that for the same average degree, ELAS performs significantly better on scale free network than other networks. This cause to conclude that average degree is not the only structural factor

which has impact on ELAS. Further investigations on structural properties of studied networks led to interesting results. Fig. 6.a shows average degree distribution of 30 networks that have been used in Fig. 5. As expected [21], ER network has Poisson like degree distribution with the mean around 4 (Fig. 6.a) and SF network presents power law distribution (Fig. 6.b). Since the process of building SW network begins with lattice of degree 4 and rewiring probability is low (0.01) the SW network has impulse like degree distribution around 4 (Fig. 6.c).

Fig. 7 shows the degree distribution of the sets that have been activated by the minority using ELAS in different network structure of Fig. 5. Degree distribution of activated set for ER network is like Poisson distribution with mean 4 (Fig. 7.a).The activated set in SF network (Fig. 7.b) has high density in lower degrees in contrast with impulse like function of SW network (Fig. 7.c).

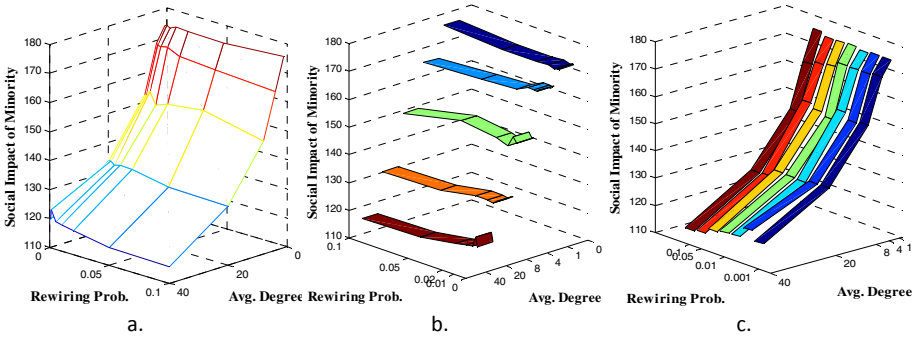


Fig. 4 Effect of network structure parameters on the success of the method for SW networks

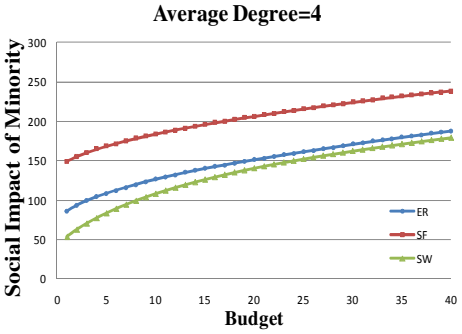


Fig. 5 Effect of network structure for same average degree

From these distributions, it can be concluded that the power of ELAS in SF networks originates from highly available individuals that can be influenced easily not the power of special persons or hubs. This is confirmed by the fact that SW network is in last place in Fig. 5, because SW mostly consists of nodes with degree of 4 which is higher than degree of available nodes in SF and ER networks.

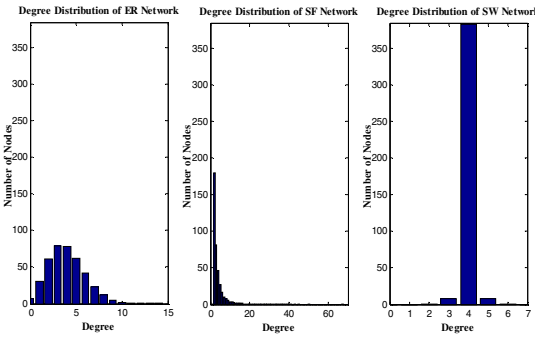


Fig. 6 Degree distribution of different network structure

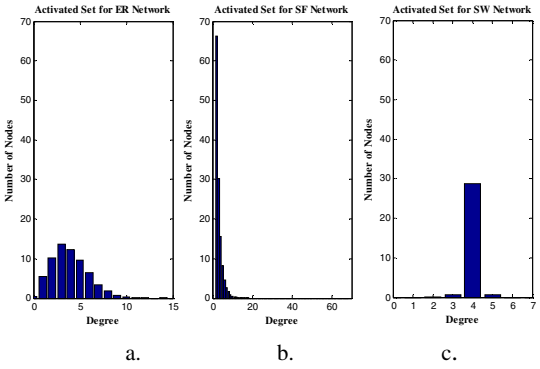


Fig. 7 Degree distribution of activated set for different network structure

7 Discussion and Future Directions

The drastic success of ELAS over other heuristics seems suspicious at the first look. So it should be mentioned that this performance is gained by spending more time for finding the initiators in ELAS. This fact becomes critical when we want to run several experiments for understanding the effect of structure on ELAS performance. In fact our experiments are infeasible for even 500 individuals.

Structure of the greedy algorithm seems neat and efficient but when it comes to the implementation part the main question is how to find $f(S)$. As we stated earlier (IV.a.) $f(S)$ is the expected value of the number of active individuals at the end of the diffusion process. So the very primitive approach to estimate $f(S)$ is to run the diffusion process for many times starting with initiator set S and take the average number of active nodes as the value of $f(S)$. This was the method that

used by Kempe et al. [7]. However they mentioned that finding $f(S)$ is the open problem for future research. After this first solution for $f(S)$ estimation, almost all efforts focused to explore the other aspects of cascading and researchers preferred to suppose that $f(S)$ is known by an oracle.

Using simple averaging as the method of computing $f(S)$ will takes hours on a modern server to select 50 seeds in a moderate sized graph (15K nodes and 31K edges) while it becomes infeasible for larger graphs [24]. Even these numbers are too large for running several experiments for exploring structural effects.

Some recent works have developed algorithms for speeding up $f(S)$ calculation with several approaches. The one that is used here is Lazy Forward Evaluation which is introduced in [19] which actually avoids $f(S)$ computing. Leskovec et al. [19] has reported the 700 times speed up in experiments. Very recent methods [24], [25] are developed separately for LTM and ICM. Both of these methods have viewed the influence propagation locally and tried to estimate $f(S)$ as the aggregation of these local cascades. Simulation results show that the final outcomes of greedy algorithm based on these methods for $f(S)$ estimation are always among best results [24], [25].

Another important extension of the naïve influence propagation is the setting in which multiple minorities exist in the society and compete with each other for adding new links and change the crowd behavior. This domain is very novel even in the context of finding K-MIP which as mentioned in IV.a is simpler than link addition problem. They are some recent works which addressed competitive setting for K-MIP problem [26], [27].

8 Conclusion

Changing belief of the majority of individuals by means of a minority was the main focus of this work. Each individual's belief is emerged from his neighbors by a simple rule that has selfishness in its nature. Based on this rule belief change propagates through the society. Minority want to change the belief of majority by making new relation with them. We leave the competitive scheme in which there exist many minorities competing on influence maximization, for future works.

A greedy algorithm was presented for finding the best new relation and its performance was compared with different relation initiation strategies. Our method, ELAS, outperformed them along with its rigorous mathematical background that shows its performance is within the specified distance of the optimal solution.

Also the effect of structure on ELAS was measured and it was found out that degree of each individual is the main parameter that impact ELAS with an inverse relation. In addition it was shown that in a population with the same average degree, number of available easily-influenced individuals is more important than influencers for the success of diffusion.

References

[1] Sobkowicz, P.: Modeling Opinion Formation with Physics Tools: Call for Closer Link with Reality. In: JASSS (2009)

[2] Jurvetson, S.: What Exactly Is Viral Marketing? Red Herring, 110–111 (2000)

[3] Leskovec, J., Adamic, L.A., Huberman, B.A.: The Dynamics of Viral Marketing. In: Proc. of the 7th ACM Conf. on Electronic Commerce, pp. 228–237. ACM, Ann Arbor (2006)

[4] Borgatti, S.P.: Identifying Sets of Key Players in a Social Network. Comput. Math. Organ. Theory 12, 21–34 (2006)

[5] Valente, T.W., Davis, R.L.: Accelerating the Diffusion of Innovations Using Opinion Leaders. The Annals of the American Academy of Political and Social Science 566, 55–67 (1999)

[6] Lazarsfeld, P.F., Berelson, B., Gaudet, H.: The People's Choice: How the Voter Makes up His Mind in a Presidential Campaign. Columbia University Press (1944)

[7] Kempe, D., Kleinberg, J., Tardos, É.: Maximizing the Spread of Influence through a Social Network. In: Proc. of the 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pp. 137–146. ACM, Washington, D.C (2003)

[8] Even-Dar, E., Shapira, A.: A Note on Maximizing the Spread of Influence in Social Networks. In: Deng, X., Graham, F.C. (eds.) WINE 2007. LNCS, vol. 4858, pp. 281–286. Springer, Heidelberg (2007)

[9] Couzin, I.D., Krause, J., Franks, N.R., Levin, S.A.: Effective Leadership and Decision-making in Animal Groups on the Move. Nature 433, 513–516 (2005)

[10] Caprari, G., Colot, A., Halloy, J., Deneubourg, J.L.: Building Mixed Societies of Animals and Robots. IEEE Robotics & Automation Magazine 12, 58–65 (2005)

[11] Granovetter, M.: Threshold Models of Collective Behavior. The American Journal of Sociology 83, 1420–1443 (1978)

[12] Durrett, R.: Lecture Notes on Particle Systems and Percolation. Brooks/Cole Pub. Co. (1988)

[13] Morris, S.: Contagion. Review of Economic Studies 67, 57–78 (2000)

[14] Easley, D., Kleinberg, J.: Networks, Crowds, and Markets: Reasoning About a Highly Connected World. Cambridge University Press (2010)

[15] Wolsey, L.A., Fisher, M.L., Nemhauser, G.: An Analysis of Approximations for Maximizing Submodular Set Functions-I. Mathematical Programming 14, 265–294 (1978)

[16] Krause, A.: Near-optimal Observation Selection Using Submodular Functions. In: AAAI NECTAR (2007)

[17] Khuller, S., Moss, A., Naor, J.: The Budgeted Maximum Coverage Problem. Inf. Process. Lett. 70, 39–45 (1999)

[18] Krause, A., Guestrin, C.: A Note on the Budgeted Maximization of Submodular Functions. pp. 05–103 (2005)

[19] Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., Glance, N.: Cost-effective Outbreak Detection in Networks. In: Proc. of the 13th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pp. 420–429. ACM, San Jose (2007)

[20] Erdos, P., Renyi, A.: On the Evolution of Random Graphs. Publ. Math. Inst. Hung. Acad. Sci. 5, 17–61 (1960)

[21] Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., Hwang, D.: Complex Networks: Structure and Dynamics. Physics Reports 424, 175–308 (2006)
[22] Barabási, A., Albert, R.: Emergence of Scaling in Random Networks. Science 286, 509–512 (1999)
[23] Watts, D.J., Strogatz, S.H.: Collective Dynamics of "Small-World" Networks. Nature 393, 440–442 (1998)
[24] Chen, W., Yuan, Y., Zhang, L.: Scalable Influence Maximization in Social Networks under the Linear Threshold Model. In: IEEE International Conference on Data Mining, ICDM (2010)
[25] Chen, W., Wang, C., Wang, Y.: Scalable Influence Maximization for Prevalent Viral Marketing in Large-scale Social Networks. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 2010, Washington, DC, USA, p. 1029 (2010)
[26] Bharathi, S., Kempe, D., Salek, M.: Competitive Influence Maximization in Social Networks. In: Deng, X., Graham, F.C. (eds.) WINE 2007. LNCS, vol. 4858, pp. 306–311. Springer, Heidelberg (2007)
[27] Pathak, N., Banerjee, A., Srivastava, J.: A Generalized Linear Threshold Model for Multiple Cascades. In: IEEE International Conference on Data Mining, ICDM (2010)

Emergence of Specialization in a Swarm of Robots

Ádám M. Halász, Yanting Liang, M. Ani Hsieh, and Hong-Jian Lai

Abstract. We investigate the emergence of specialized groups in a swarm of robots, using a simplified version of the stick-pulling problem [5], where the basic task requires the collaboration of two robots in asymmetric roles. We expand our analytical model [4] and identify conditions for optimal performance for a swarm with any number of species. We then implement a distributed adaptation algorithm based on autonomous performance evaluation and parameter adjustment of individual agents. While this algorithm reliably reaches optimal performance, it leads to unbounded parameter distributions. Results are improved by the introduction of a direct parameter exchange mechanism between selected high- and low-performing agents. The emerging parameter distributions are bounded and fluctuate between tight unimodal and bimodal profiles. Both the unbounded optimal and the bounded bimodal distributions represent partitions of the swarm into two specialized groups.

1 Introduction

In a robotic swarm, *heterogeneity* may be quantified in terms of *diversity*, or the variability of the properties of individual agents. Heterogeneity may also involve the *specialization* of individuals for certain tasks. This collective adaptation strategy is often seen in biology [6]. The design of heterogeneous swarms requires ways to quantify the degrees of heterogeneity and specialization as well as their impact on collective performance. Early work on heterogeneity and specialization in robot

Ádám Halász · Yanting Liang · Hong-Jian Lai
Department. of Mathematics, P.O. Box 6310, West Virginia University,
Morgantown, WV 26506
e-mail: {halasz,lyt814,hjlai}@math.wvu.edu

M. Ani Hsieh
Mechanical Engineering and Mechanics Department, Drexel University,
3141 Chestnut Street, Philadelphia, PA 19104
e-mail: mhsieh1@drexel.edu

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 403–416.
springerlink.com

teams established methods for the composition of group level behaviors [7, 9] and proposed a measure for heterogeneity [1].

The stick-pulling problem was originally formulated [5] to explore the swarm intelligence paradigm [3] in a context where collaboration is realized through local interactions, with limited or no global communication. The basic task, finding and pulling randomly distributed sticks, requires two robots in asymmetric roles. A robot that finds a stick must wait for another one to help pull the stick. The *gripping* or *waiting time parameter* (WTP) [4,5] of a robot is the time it will wait for help before releasing a stick. In the original study [5], Ijspeert *et al.* found that this asymmetric task could benefit from specialization. Through experimentation and a two-level modeling approach, they identified an optimal WTP for a homogeneous swarm. For a heterogeneous swarm with two subgroups (*castes* or *species*) of agents, each with a different WTP, they found a family of high-performance pairs of WTP values. This type of heterogeneity led to better performance when the number of robots was less than the number of sticks and did not make a significant difference otherwise.

Li, Martinoli and Mustafa [8] investigated how specialization could be learned by the stick-pulling team. In their system, agents changed their WTP based on local or global reinforcement signals. Learning resulted in optimal performance accompanied by increases in information-theoretic measures of diversity and specialization. This work strengthened the correlation between group performance and diversity and provided an example of global performance improvement through individual adaptation. However, it left open the question whether distinct groups with specialized behaviors could emerge through individual adaptation.

We investigated the advantages of specialization in a slightly modified version of the stick pulling problem [4], using a methodology developed for task allocation [2]. The starting point of our modeling approach was similar to the probabilistic model of [5]. Our higher level of abstraction resulted in a concise and transparent analytical model and in the possibility of scaling simulations into the range of thousands of agents and millions of updates. We identified a maximal performance level that may not be exceeded for any WTP configuration, and showed that it could be reached in many different configurations. Comparing homogeneous and two-species configurations, we showed analytically and confirmed through simulations that the two-species swarm performed better under non-ideal circumstances than the homogeneous one (in the case with more sticks than robots). Echoing the results of [5], we found that specialization was advantageous.

In this work we expand the analysis of optimal configurations and explore *collective adaptation* based on *individual adjustment* of the agents. We investigate adaptation strategies from two perspectives: (1) convergence to optimal performance; (2) emergence of subgroups with specialized behaviors. We implement a distributed adaptation algorithm where robots randomly change their WTP with a frequency based on their own performance. In the second algorithm we add an exchange mechanism where WTPs of successful agents are assigned to underperforming ones. Both algorithms converge to configurations that ensure optimal performance. The WTP exchange mechanism increases the cohesion of the WTP distribution, causing the system to converge to bounded uni- or bimodal distributions.

2 Model and Analytical Results

2.1 The Stick Pulling Problem

N are robots tasked with pulling sticks from the ground. The S_T sticks are randomly distributed in the workspace. Two robots are required to pull a stick. Robot behaviors are sketched in Figure 1. Robots initially *wander* in search of sticks and can *discover* sticks in their immediate vicinity. When a robot finds a stick held by another robot, the robots pull the stick together. If a robot finds a free stick, it *holds* it waiting for another robot to come along, but will release it after a certain time. We model the discovery of sticks as a stochastic process, characterized by a *discovery rate* k_D, the same for all sticks, whether or not they are held by another robot. This rate accounts for all physical and technological constraints, such as: the physical density of sticks, the size and accessibility of the area, the movement and detection capabilities of the robots. The numbers of sticks and robots are constant. The only element that can be chosen by design is the behavior of the robots upon discovery of a free stick. *Release* after waiting is described as a Poisson process whose characteristic time is the w*aiting time parameter* (WTP) τ_i, set individually for each agent.

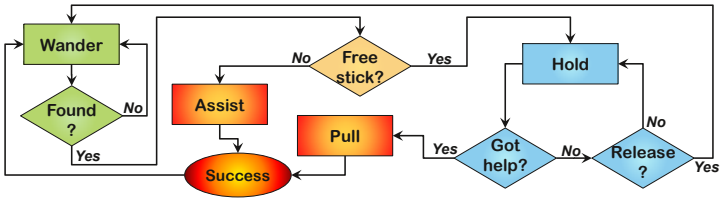


Fig. 1 Flow chart of robot behaviors in the stick pulling model

2.2 Equations of Motion

The N robots are subdivided into $p \leq N$ groups; N_i is the number of agents in group i. There are S_T sticks in total. At any time, a robot may be *free* (*wandering*), or *holding* a stick. We denote the number of free robots of type i with F_i, and by H_i the number of those holding a stick. The total number of free sticks and free and holding robots are denoted by S, F, and H. If total number robots of each type are fixed, we have:

$$\begin{aligned} N = \sum_{i=1}^{p} N_i \;\; ; \;\; H = \sum_{i=1}^{p} H_i \;\; ; \;\; F = \sum_{i=1}^{p} F_i \\ S_T = S + H \;\; ; \;\; N = F + H \;\; ; \;\; N_i = F_i + H_i, \;\; \forall i \in \{1, \cdots, p\} \end{aligned} \tag{1}$$

Robots in a group have the same WTP, τ_i, the *average* time a robot holds on to a stick before *releasing* it. The release is controlled by a Poisson process whose rate is

$\lambda_i = 1/\tau_i$. Similarly, the process of discovery of sticks by free robots is characterized by the *discovery rate* k_D. Due to (1), the state of the system is defined by the number of robots of each type holding a stick, $\{H_1, \cdots, H_p\}$. We will write our equations in terms of these variables. Since we are interested in large swarms, we will adopt a *continuum approach* in describing the dynamics of the system [2].

There are three processes that contribute to the variation of H_i, *capture* (discovery), *pull*, and *release* of sticks. The *capture rate* $r^{(i)}_{capt}$ is proportional to the number of free robots of type i and the number of free sticks. The *pulling rate* $r^{(i,j)}_{pull}$ is proportional to the number of free robots of type i and the number of robots of type j holding a stick. These two processes have the same rate constant, k_D. Note that the pulling rate does *not* impact the number of free robots of the type of the second participant. By contrast, the robot that was holding the stick changes its state from holding to free. We denote by $r^{(i)}_{pull}$ the *total* rate of successful pulls of sticks held by robots of type i. The *release rate* $r^{(i)}_{release}$ of sticks by robots of type i is proportional to the number of robots of type i holding a stick, and the rate constant $\lambda_i = 1/\tau_i$.

$$\begin{aligned} r^{(i)}_{capt} &= k_D F_i S = k_D (N_i - H_i)(S_T - H) \quad ; \quad r^{(i,j)}_{pull} = k_D F_i H_j = k_D (N_i - H_i) H_j \\ r^{(i)}_{release} &= \lambda_i H_i \qquad\qquad\qquad\quad ; \quad r^{(j)}_{pull} = k_D (N - H) H_j \ . \end{aligned} \tag{2}$$

The net rate of change in H_i is then:

$$\frac{dH_i}{dt} = k_D \left[(N_i - H_i)(S_T - H) - H_i (N - H) \right] - \lambda_i H_i \ . \tag{3}$$

2.3 Steady State Analysis

We are interested in the steady-state(s) of (3). For any such configuration, the right-hand side of the equations of motion must vanish. Setting $\frac{dH_i}{dt} = 0$, we have:

$$k_D \left[N_i (S_T - H) - H_i (S_T + N - 2H) \right] = \lambda_i H_i \ . \tag{4}$$

This equilibrium condition is more transparent in terms of dimensionless variables:

$$\frac{S_T}{N} \equiv \sigma \ ; \ \frac{H}{N} \equiv \phi \ ; \ \frac{H_i}{N_i} \equiv \varphi_i \ ; \ \frac{N_i}{N} \equiv \rho_i \quad \longrightarrow \quad \varphi_i = \frac{\sigma - \phi}{\xi_i + (1 + \sigma - 2\phi)} \ . \tag{5}$$

The *dimensionless time parameter* ξ_i is the ratio between the average time between two discoveries of the same stick by two robots, and the waiting time parameter τ_i:

$$\xi_i \equiv \frac{\lambda_i}{N k_D} = \frac{1}{N k_D \tau_i} = \frac{1/(N k_D)}{\tau_i} \ . \tag{6}$$

The *occupancy fraction* ϕ is a weighted average of the individual *occupancies* φ_i. Substituting the individual equilibrium conditions, we arrive at a global condition:

$$\phi \equiv \frac{H}{N} = \sum_i \rho_i \varphi_i \longrightarrow \phi = \sum_i \frac{(\sigma-\phi)\rho_i}{\xi_i + (1+\sigma-2\phi)} = f(\phi) \ . \tag{7}$$

It can be shown that the equation $\phi = f(\phi)$ has a unique solution, which corresponds to a stable equilibrium of the equations of motion (3).

2.4 Pulling Rates and Optimality

The global *pulling rate*, (given N robots of which H are holding sticks) is

$$R_{pull} = k_D(N-H)H \ . \tag{8}$$

Since $0 \le H \le N$, R_{pull} is always positive, vanishes for $H = 0$ and $H = N$, and is maximal for $H = H^* \equiv min(N/2, S_T)$. The *maximal pulling rate* is R^*_{pull} below:

$$R^*_{pull} = k_D(N-H^*)H^* \ ; \ R^*_{pull}(N,S_T) \le R^{max}_{pull}(N) \equiv \frac{1}{4}k_D N^2 \ . \tag{9}$$

Here, R^{max}_{pull} is the maximal pulling rate *for N robots*; it may be achieved if there are enough sticks ($S_T > N/2$). If the number of robots N is larger than $2S_T$, the maximal pulling rate is limited to $k_D(N-S_T)S_T$. *We will assume* $N < 2S_T$, so $R^*_{pull} = R^{max}_{pull}$.

2.4.1 Optimal WTP Configurations

The objective of designing our swarm is to maintain the system performance as close to the ideal situation $H = N/2$ as possible. For a given configuration of groups and WTPs, we can calculate the equlibrium state of the system and the corresponding pulling rate, by solving the equilibrium condition (7) for the global occupancy ϕ, and use it to calculate the individual occupancies. We then specify conditions for optimality by requiring $\phi = 1/2$. A configuration of waiting time parameters that results in $\phi = 1/2$ is called *optimal* or *ideal*.

One species: If all agents have the same WTP τ, the equilibrium condition (9) reads

$$2\phi^2 - (2+\sigma+\xi)\phi + \sigma = 0 \ . \tag{10}$$

Of the two solutions for ϕ, only one is in the $[0,1]$ interval. The design problem here consists of determining the waiting time parameter τ (through the dimensionless time parameter $\xi = 1/Nk_D\tau$) so that optimal performance is achieved. We can calculate the value of the ideal $\xi = 1/Nk_D\tau$ by substituting $\phi = 1/2$:

$$\xi^* = \sigma - 1 \ \leftrightarrow \ \tau^* = \frac{1}{k_D(S_T - N)} \ . \tag{11}$$

Two types of robots: We have to design three quantities, the two WTPs τ_1, τ_2, and the ratio ρ_1/ρ_2 between the sizes of the groups. Given $\{\xi_1, \xi_2, \rho_1, \rho_2\}$, the equilibrium configuration is uniquely defined. Choosing $\rho_1 = \rho_2 = 1/2$, we obtain the following constraint for the dimensionless time parameters $\{\xi_1, \xi_2\}$:

$$\frac{1}{\xi_1 + \sigma} + \frac{1}{\xi_2 + \sigma} = \frac{2}{2\sigma - 1} \quad . \tag{12}$$

Equal size groups: Consider a number of p different species, each representing $1/p$ of the population. [1] The optimality condition is

$$\frac{p}{2\sigma - 1} \;=\; \sum_{i=1}^{p} \frac{1}{\xi_i + \sigma} \quad \longleftrightarrow \quad \frac{1}{p} \sum_{i=1}^{p} \frac{1}{\xi_i + \sigma} \;=\; \frac{1}{2\sigma - 1}. \tag{13}$$

For $p = 1$ we reobtain the condition for the ideal ξ. The second version of the condition can be interpreted as a requirement that the average of the quantities $1/(\xi_i + \sigma)$ match the ideal value $1/(2\sigma - 1)$.

2.4.2 Robustness Measures

The optimality requirement (13) represents a single algebraic constraint. With p equal sized groups, all but one of the WTPs $\{\tau_1, \cdots, \tau_p\}$ may take any value over a semi-infinite interval. The corresponding configurations form a $p-1$ dimensional manifold in the p-dimensional space of WTP configurations. We are interested in *additional* performance criteria to characterize these ideal configurations.

Performance under changing conditions: Consider the pulling rate of a system that is optimal for a stick/robot ratio of σ_0, when faced with a different $\sigma \neq \sigma_0$. In Figure 2 we compare a one-group configuration with $\tau = \tau^*$ (11) and two configurations of two groups of equal size with WTP pairs $\{\tau_1, \tau_2\}$ that satisfy (12), for $\sigma = \sigma_0 = 10$. The larger τ_1, the smaller τ_2 has to be. The factor $K = \tau_1/\tau^*$ is a measure of how far the $\{\tau_1, \tau_2\}$ pair is from the one-species case ($K = 1$). Theoretical predictions for the pulling rate for $K = 1, 10, 100$ are confirmed by simulation results as indicated. The loss of performace is the strongest for the one-group configuration ($K = 1$) and becomes milder as the ratio between τ_1 and τ^* increases.

Loss of agents: As a measure of how much of the optimality would be preserved by a subset of the agents in a given configuration, it is useful to compare the pulling efficiency *per agent* for a configuration where some agents are destroyed. This measure is relevant when comparing WTP configurations that result from randomized adaptation algorithms, where no two agents would likely have the same WTP. It provides a mechanism to penalize configurations that are "too heterogeneous".

[1] This also applies to the situation when the robots are essentially independent, taking $p = N$.

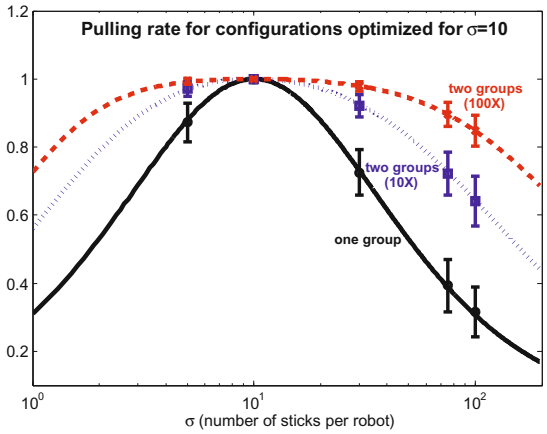


Fig. 2 Efficiency loss when conditions differ from ideal. Theoretical predictions (lines) and simulation results (points) for the pulling rate of a one-group and several two-group configurations that are optimal for $\sigma = 10$. The WTP pairs $\{\tau_1, \tau_2\}$ of the two-group configurations satisfy (12), with the stated values of the ratio $K = \tau_1/\tau^*$ (10 and 100, respectively). The simulations (one per data point) had $N = 150$ robots and numbers of sticks so that $\sigma = S_T/N = 5, 10, 30, 75, 100$.

3 Simulation Methods

3.1 Basic Simulation Algorithm

The main simulation algorithm is derived from the Gillespie algorithm used in biochemistry and adapted to multi-robot systems in previous work [4]. The state of the system is defined by that of the individual agents. Each of the N agents can be free (F) or holding a stick (H). There are three possbile transitions, corresponding to the three processes discussed above:

$$F_i + S \to H_i \ (\textsc{capture}); \ H_i + F_j \to F_i + F_j + S \ (\textsc{pull}); \ H_i \to F_i + S \ (\textsc{release})$$

In the CAPTURE(i) process, agent i goes from Free to Holding; the reverse is the RELEASE(i) process. In the PULL(i, j) process, agent i goes from holding to free, but the process requires another agent (j), whose state is not ultimately changed. Transitions are controlled by independent Poisson processes; the probability per unit time (or rate) for a specific transition is given by a time constant and the number of eligible partners, if applicable. For example, if both agents i and j are free, the probability per unit time for capturing a stick is the same for both of them, $k_D S$. The release rate for agent i while holding a stick is $\lambda_i = 1/\tau_i$.

In the Gillespie algorithm, simultaneous Poisson processes are simulated by generating *next event times* for each process, then implementing the state transition that corresponds to the smallest one of the next event times. When there are many

possible transitions, one may calculate the cumulative transition rate for each *type* of transition, then choose a specific (pair of) agent(s) for the transition. This approach is correct for simultaneous Poisson processes and we do this for events that involve encounters between free robots and sticks. The additional computational cost due to updating the states of *individual agents* is almost negligible. Thus, the Poisson model for transitions allows us to have the equivalent of an agent-based simulation for the cost of a centralized one.

3.2 Individual Adaptation

We construct a self-evaluation measure or *satisfaction level* χ_i for agent i, as follows. Every time agent i participates in a successful pull (in either role), χ_i is incremented by 1. At every update, χ_i decreases exponentially with a characteristic time τ_{forget}: $\chi_i(t+\Delta t) = \chi_i(t) exp(-\Delta t/\tau_{forget})$. Thus, agents have a memory of past successes, but their satisfaction level decreases as they go through a dry spell.

The satisfaction level defined here does not provide an absolute measure of an individual agent's effectiveness. There is no reference value for it, unless the agent knows what pulling rate it should expect. The maximal pulling rate can be computed from the number of sticks and agents; however, we are interested in an adaptive strategy that can find the optimum *without* relying on global knowledge.

In this algorithm, each agent changes their WTP *randomly*, at a rate proportional to *the inverse* of the agent's satisfaction level (lower satisfaction increases the rate of change). Adaptation is implemented as a Poisson process with time constant τ_{learn}/χ_i, that runs in parallel with the other transitions (but much slower). Every time this process fires, the respective agent changes its WTP with a small random quantity: $\tau_i' = \tau_i exp((r-1/2)\Delta))$ where r is a uniformly distributed random number between $[0,1]$ and Δ is the *Monte-Carlo (MC) step size* (typically a small number). This algorithm results in a random walk in the space of $log(\tau_i)$ biased by the satisfaction function. Our approach is simpler than the one used by Li *et al.*, but it also relies on a proper self-assessment of performance.

3.3 Swapping

As we discuss next, the individual adaptive strategy succeeds in optimizing the pulling rate, but generates configurations where the individual WTPs are spread over many orders of magnitude. In order to increase the coherence of the resulting WTP distributions, we introduced a *collective mechanism* to supplement individual adaptation. It consists of an additional WTP change, performed with a small (fixed) probability ν, during normal WTP updates. This corresponds to an additional Poisson process, with propensity ν/τ_{learn}. When this process fires, we select a pair of agents, a *donor* (with a high satisfaction level), and an *acceptor* (with a low satisfaction level), and change the WTP of the acceptor to that of the donor. This procedure is reminiscent of biologically inspired algorithms. While it requires some degree of

collective communication, it can be implemented in a way that ensures reasonable scaling as the number of agents increases. The key is in that the donor and acceptor agents can *self-select* and communicate using a pre-determined procedure of asynchronous communication to upload or download their WTPs.

4 Results and Discussion

4.1 Model Validation - Equilibration

The standard system in our simulations consists of $N = 150$ robots and $S_T = 2000$ sticks. We use time units where the discovery rate $k_D = 1$. Thus, the maximal pulling rate is $\frac{1}{4}N^2 k_D = 5625$ pulls per unit time. The corresponding optimal waiting time is $\tau^* = 1/k_D(S_T - N) = 5.4 \times 10^{-4}$. The average time between two robot-stick encounters is $\tau_E = 1/(k_D S_T N) = 3.33 \times 10^{-6}$. The time between two consecutive updates in a simulation is on the order of 10^5 iterations per time unit. We performed simulations with various WTP configurations and verified that the system converges to the average occupancy fractions and pulling rates predicted by the continuum equations in Sec.2 . The value of the equilibration time is comparable to the average time of 5×10^{-4} units it takes for one robot to find one of the 2000 sticks. Figure 2 shows theoretical and simulation results for the equilibrium pulling rate for configurations with one or two WTP groups, for the same number of robots ($N = 150$), but different numbers of sticks (simulations with $\sigma \equiv S_T/N = 5, 10, 30, 75, 100$). All configurations are ideal for $\sigma = 10$ ($S_T = 1500$ sticks). The simulation results confirm the analytical predictions given in Sec.2.4.1.

4.2 Individual Adaptation Algorithm

We implemented the individual adaptation algorithm described in Sec.3.2 on the $N = 150$, $S_T = 2000$ system, exploring parameter values around $\tau_{learn} = 1.0 \times 10^{-4}$, $\tau_{forget} = 0.1$ and a Monte-Carlo step size of $\Delta = 1.0 \times 10^{-2}$.

The evolution of the system with these parameter values is shown in Figure 3. The waiting time parameters are initially set to 50% of the optimal value τ^*. As the individual τ values change, the number of free robots evolves, reaching the optimum of 75 in approximately 400 time units. This τ-*convergence time* is significantly longer than the *equilibration time* of 5×10^{-4} it takes the number of free robots to reach the equilibrium value corresponding to a *fixed* WTP configuration. It is useful to visualize the time evolution of WTPs using the distribution of the $log(\tau)$ values, as in Figure 3. The results are qualitatively different from the one- or the two-group configurations described previously. As the simulation starts, the $\log(\tau)$ distribution spreads out and continues to do so over time, expending into extremely large and small (positive) values, reaching widths of 10 orders of magnitude and higher after 10^8 steps. The $log(\tau)$ distribution is close to a normal, whose standard deviation increases like the square root of the simulation time.

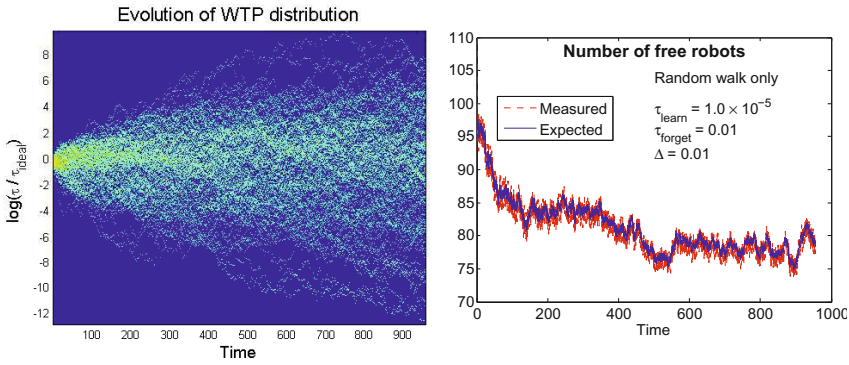


Fig. 3 (Left) Evolution of the distribution of WTPs in an adaptive simulation. Notice the similarity to a diffusion process. (Right) The number of free robots in the same simulation (expected = based on the current configuration of WTPs). Based on a single simulation with $N = 150$ robots, $S_T = 2000$ sticks, and 3×10^8 updates.

We performed a number of simulations to investigate the effect of changing the adaptation parameters on convergence. The results are presented in Figure 4. We define convergence for the purposes of these simulations as the state (after the initial equilibration) where the moving average over 10,000 iterations of the number of free robots is within 1 of the ideal value of 75. We limited the simulations to 10^7 iterations, and we plot both the time to convergence and the number of held sticks after the maximum number of iterations. For the converged simulations, the final number of sticks is very close to 75, and the convergence time varies. For the unconverged simulations, better adaptation corresponds to final sticks held counts that are closer to the ideal value of 75.

The dependence on the averaging time τ_{forget} is relatively weak. None of the simulations using this algorithm converged (within the iteration limit), due to the values for Δ and τ_{learn}. However, the final state approaches 75 as τ_{forget} is reduced by a factor of 10, and moves further away as τ_{forget} is increased. Increase in the frequency of WTP changes (decreased τ_{learn}) leads to marginal improvement. A 10-fold increase in the Monte-Carlo step size Δ improves the adaptation performance to the point where the system converges within the 10^7 iteration cutoff. Further increase of the step size leads to additional improvement in the convergence time. However, the configurations reached in this manner are increasingly incoherent, with a very wide WTP distribution.

4.3 Swapping Algorithm

The introduction of swapping leads to dramatically improved convergence, over all parameter values investigated. It is remarkable that WTP swapping, performed at a frequency corresponding to one swap per every 100 individual WTP changes, improves convergence this much. The parameter sensitivity results for this algorithm

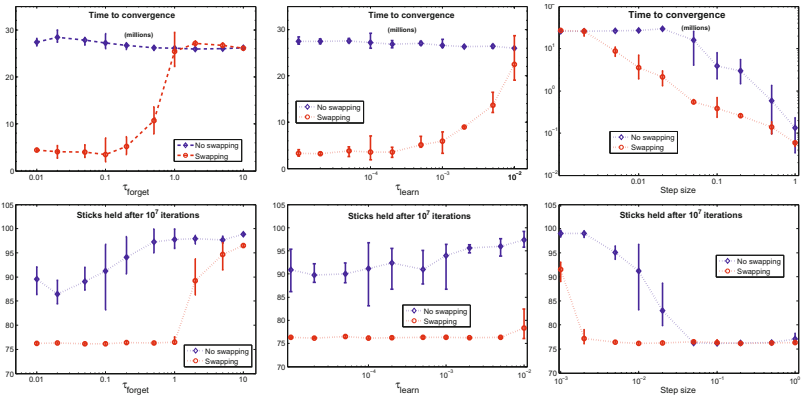


Fig. 4 Convergence time (*top*, in millions of iterations) and sticks held after 10^7 iterations (*bottom*) versus τ_{forget}, τ_{learn}, and MC step size Δ, in the individual adaptive algorithm (blue) and with swapping (red). Each point represents 5-20 simulations with $N = 150$ robots and $S_T = 2000$ sticks.

are also plotted in Figure 4. The effect of parameter changes is qualitatively similar in the two algorithms. The swapping result converges for all but the highest values of the averaging time τ_{forget}. The performance of the algorithm deteriorates as this parameter is increased, and convergence is lost as τ_{forget} goes from 0.1 to 1.0. Increased τ_{learn} also reduces the performance of the swapping algorithm.

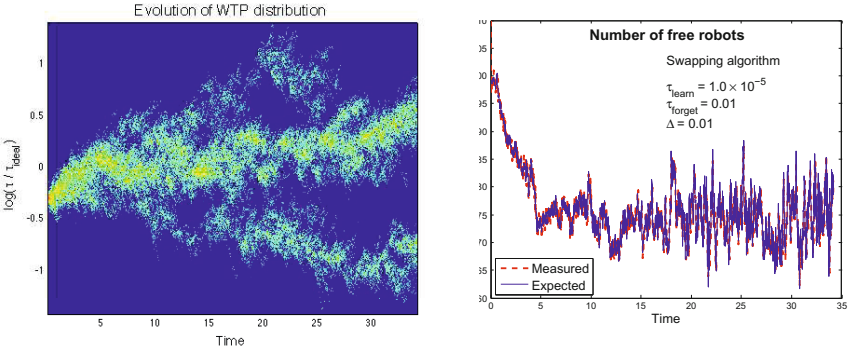


Fig. 5 Evolution of the distribution of waiting time parameters (left) and the number of free robots (right) for an adaptive simulation with swapping (expected = based on the current configuration of WTPs). Note the different time scale of convergence compared to the non-swapping simulation shown in Figure 3; the two simulations have the same adaptation parameters. Based on a single simulation with $N = 150$ robots, $S_T = 2000$ sticks, and 10^7 update steps.

The first algorithm was the most sensitive to the Monte-Carlo step size Δ. Increased Δ improved the convergence of both algorithms, and their performance became similar as $\Delta \approx 1$. A value of $\Delta = 1$ means that the random change of a WTP is comparable to the value of the WTP. With such large variation steps, the newly selected parameters have little to do with the previous ones. In this limit, the algorithm tends to become purely random selection of parameters rather than a search process (on the level of *individual* agents).

Swapping dramatically limits the expansion of the WTP distribution, as shown in Figures 5 and 6. For most parameter values (except very high Δ) the simulations resulted in bounded, unimodal distributions with a spread of little more than one order of magnitude, much less than in the individual adaptation case (compare Fig.6 and 3). In the longer term, some of the simulations exhibit transitions to bimodal distributions. The bimodal distributions we observed had narrow modes, with maxima separated by 1-2 orders of magnitude. While the bimodal distributions extended over almost three orders of magnitude, they remained bounded, and the system eventually transitioned back to the unimodal regime.

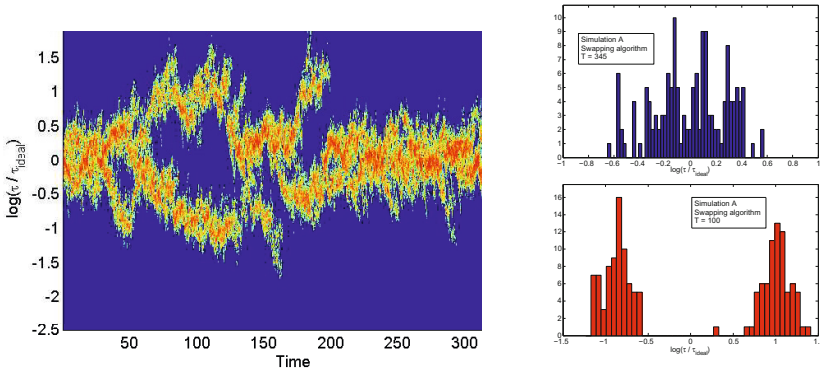


Fig. 6 Evolution of the distribution of waiting time parameters during a long simulation with swapping. Based on a single simulation with $N = 150$ robots, $S_T = 2000$ sticks, and 10^8 update steps.

4.4 Waiting Time Parameter Distributions

From a design and analysis perspective, configurations with one, two or a small number of distinct waiting time parameters seem more straightforward. By contrast, both adaptation algorithms result in configurations that can only be characterized by a continuous distribution of waiting time parameters, rather than one or a few distinct WTPs shared by groups of agents.

The evolution of the $log(\tau)$ in the individual adaptation algorithm (Figure 3) is similar to pure diffusion, consistent with a random walk. This makes sense because each individual WTP change is a small variation taken from a distribution that is

symmetric in terms of $log(\tau)$. This random walk is influenced through the satisfaction function and is practically confined to globally optimal configurations. The global optimality condition (13) corresponds to a single constraint on the N waiting time parameters. If the individual $log(\tau)$ are allowed to increase or decrease indefinitely, most agents will have waiting times that are either much larger or much smaller than the ideal value. In this case, the $1/(\xi_i + \sigma)$ terms in Eq.(13) would approach $1/\sigma$ and 0, respectively. The optimality condition for a configuration with N_{high} agents with very high WTP ($\tau_i >> 1 \ \rightarrow \ \xi_i << 1$) and the rest with very small WTP is simply

$$\frac{N_{high}}{N} = \frac{\sigma}{2\sigma - 1} = \frac{S}{2S - N} \ . \tag{14}$$

This simple constraint on the values of the WTPs of the agents in the high and low groups ensures optimality for the late WTP configurations obtained in the individual adaptation algorithm. We will call these *divergent-optimal* or *DO* configurations. Presumably, these could also be obtained more easily, by a simple random search on the level of individual agents. A DO configuration can be interpreted as an example of specialization (castes with $\tau = \{0, \infty\}$).

5 Summary and Conclusions

We have expanded our analysis [4] of the stick-pulling problem and established that each WTP configuration corresponds to a unique equilibrium pulling rate which can be estimated analytically. We showed that there is a maximum possible or *optimal* pulling rate for a given number of sticks and robots (9). The optimality requirement can be formulated as a single algebraic condition (13) for the N parameters.

We designed and implemented two adaptive optimization strategies and showed that both converge to optimal configurations. The *individual adaptation algorithm* relies exclusively on the agents' own record of their performance, in the form of a *satisfaction function*. Robots change their WTP based on this function (low satisfaction $\rightarrow$ higher change rate). Each change is a Monte-Carlo step in a random direction. The evolution of the WTP distribution in this algorithm is consistent with diffusion. The distribution of $log(\tau)$ approaches a normal whose witdh increases indefinitely, while maintaining optimal performance. The long-term limit for this type of distribution, called *divergent-optimal* (DO), has WTPs that approach either zero or infinity. Optimality can be ensured by the appropriate ratio between the two groups (14). DO configurations can be regarded as extreme examples of emerging specialization. The $\tau \rightarrow \infty$ species specializes in discovering and holding sticks, and the $\tau \rightarrow 0$ specializes in assisting stick holders.

In the *swapping algorithm* we supplement individual adaptation with a mechanism that assigns the WTP of well performing agents to under-performing ones. While requiring a limited amount of global communication, this algorithm leads to dramatic improvement of the rate of convergence. It also limits the width of the

WTP distributions. Increased Monte-Carlo step size in the swapping algorithm leads to faster converence but eventually results in the emergence of DO configurations.

Emergence of specialization can also be observed in the swapping algorithm, where long-term simulations fluctuate between *bounded* uni- and bimodal distributions with narrow modes. The bounded bimodal configurations are closer to the idea of specialized groups, each with a narrowly defined set of features (similar to biological phenotypes).

In conclusion, our results provide two mechanisms by which specialized groups of agents can emerge from an agent-based adaptation strategy. The more easily obtained DO configurations may not be satisfactory for a given application. Further refinements are necessary to stabilize the bounded bimodal configurations. This will require more sophisticated measures of performance, which can enforce our preference for one or another type of WTP distribution. We gave two possible examples of such measures that may be implemented in future applications. Finally, future work in this direction should also integrate results from machine learning and information theory.

References

1. Balch, T.: Hierarchic social entropy: An information theoretic measure of robot team diversity. Autonomous Robots 8(3), 209–238 (2000)
2. Berman, S., Halasz, A., Hsieh, M.A., Kumar, V.: Optimized stochastic policies for task allocation in swarms of robots. IEEE Transactions on Robotics 25(4), 927–937 (2009)
3. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From natural to artificial systems. Oxford University Press, New Yok (1999)
4. Hsieh, M.A., Halasz, A.M., Cubuk, E.D., Schoenholz, S., Martinoli, A.: Specialization as an optimal strategy under varying external conditions. In: Proceedings of the International Conference on Robotics and Automation (ICRA), Kobe, Japan (2009)
5. Ijspeert, A., Martinoli, A., Billard, A., Gambardella, L.M.: Collaboration through the Exploitation of Local Interactions in Autonomous Collective Robotics: The Stick Pulling Experiment. Autonomous Robots 11(2), 149–171 (2001), doi:10.1023/A:1011227210047
6. Kussell, E., Leibler, S.: Phenotypic diversity, population growth, and information in fluctuating environments. Science 309(5743), 2075–2078 (2005)
7. Lerman, K., Galstyan, A., Martinoli, A., Ijspeert, A.J.: A Macroscopic Analytical Model of Collaboration in Distributed Robotic Systems. Artificial Life 7(4), 375–393 (2001), doi:10.1162/106454601317297013
8. Li, L., Martinoli, A., Abu-Mostafa, Y.: Learning and Measuring Specialization in Collaborative Swarm Systems. Adaptive Behavior 12(3-4), 199–212 (2004); Special issue on Mathematics and Algorithms of Social Interactions, Anderson, C., Balch, T. (eds.), doi:10.1177/105971230401200306
9. Matari'c, M.J.: Designing and understanding adaptive group behavior. Adaptive Behavior 4, 50–81 (1995)

Distributed Colony-Level Algorithm Switching for Robot Swarm Foraging

Nicholas Hoff, Robert Wood, and Radhika Nagpal

Abstract. Swarm robotics utilizes a large number of simple robots to accomplish a task, instead of a single complex robot. Communications constraints often force these systems to be distributed and leaderless, placing restrictions on the types of algorithms which can be executed by the swarm. The performance of a swarm algorithm is affected by the environment in which the swarm operates. Different environments may call for different algorithms to be chosen, but often no single robot has enough information to make this decision. In this paper, we focus on foraging as a multi-robot task and present two distributed foraging algorithms, each of which performs best for different food locations. We then present a third adaptive algorithm in which the swarm as a whole is able to choose the best algorithm for the given situation by combining individual-level and distributed colony-level algorithm switching. We show that this adaptive method combines the benefits of the other methods, and yields the best overall performance.

1 Introduction

The performance of a robot swarm algorithm is affected by the environment in which it operates. In a search task for example, the location and number of search targets and presence or absence of obstacles could affect the efficiency of the swarm. One algorithm may be fast but fail in the presence of obstacles, and a slower one may be more resilient. Because the specifics of the environment are generally not known before hand, we would like the swarm itself to be able to intelligently change its own algorithm based on the environment. An interesting challenge is whether a robot swarm, as a whole, can assess the success or failure of an algorithm and, as a whole, switch algorithms to increase its success.

Nicholas Hoff · Robert Wood · Radhika Nagpal
Harvard University, Boston, MA 02163, USA
e-mail: {nhoff,rjwood,rad}@eecs.harvard.edu

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 417–430.
springerlink.com

Colony-level algorithm switching is difficult for two reasons. First, the information on which the colony will base its decision is distributed throughout the environment and not detectable by any one robot. Therefore, the environment detection must be distributed. Second, if a swarm algorithm relies on strong coordination, then switches must be nearly unanimous and synchronized. Changing algorithms will not help unless all individuals change to the same algorithm at the same time. Both of these actions—environment sensing and algorithm switching—must truly take place on the colony level, as opposed to the individual level.

In this work, we focus on the problem of foraging for robot swarms. Foraging algorithms enable a collection of robots to search a space for a goal (the 'food'), then return it incrementally to the nest. We assume robots with simple sensing and communication capabilities that can exchange simple messages (a few bits) directionally with other robots within a short range. Since the robots do not have global localization or odometry, coordination within the swarm is essential for performing the task efficiently.

We present three distributed foraging algorithms. All of the algorithms are based on a few core "sub-algorithms" which the swarm intelligently switches between in various combinations and at various times. The first is a gradient–based method in which robots form a stationary beacon field around the nest to create gradients to the nest and food. The second is an area–sweeping algorithm (called "sweeper") in which robots use virtual forces to coordinate and form a line which sweeps the world. Finally, in the third algorithm ("adaptive"), the colony cooperates to detect when one algorithms is failing and switches to another, thus increasing performance.

To evaluate the algorithms, we place food at varying distances from the nest. We find that the gradient algorithm works fast but only in a short range, and the sweeper algorithms is slower but has a larger range. We show that the adaptive algorithm is capable of high-level switching, and that the swarm is able to choose the algorithm best suited to the current food location.

1.1 Related Work

Gradient-based Foraging. Gradient-based foraging methods create a gradient leading to the goal, using the sensing capabilities of the robot, such as chemical sensing or communication. Algorithms using many different types of sensing capabilities have been studied. Algorithms exist for robots with global positioning and global communication [18], robots that use physical marks to leave a trail [9, 11], robots that use a pre-deployed sensor network [10], and robots that use deployable beacons [6]. Our work focuses on robots with directional communication. Payton et.al. have developed an algorithm in which each robot can receive messages in a small radius, and use this to create a virtual pheromone. Our gradient algorithm is similar to [5], [19], [8], and [21] in that directional communication is used to transmit relative position information to establish the gradient. Networking researchers would also recognize the gradient algorithm as being very similar to "hop-count" routing [22].

Area-Sweeping Foraging. In this algorithm, inter-robot virtual forces are used to make movement decisions which cause the swarm to adapt a line shape. Spears studied physics-based control of vehicle swarms, with attractive and repulsive forces forming a lattice of vehicles [20]. In the field of sensor networks, Howard et.al. have used potential fields to achieve dispersion of nodes [7]. The closest work to our application is Balch's notion of social potentials [3]. Social potentials involve robots navigating to a goal while remaining in a formation, feeling virtual forces based on the position of the goal and the relative positions of other robots. These algorithms are focused on maintaining a formation. We use a similar concept in which robots feel virtual forces, and we show how to use the formation to search the region. We use this to develop a virtual forces-based foraging method.

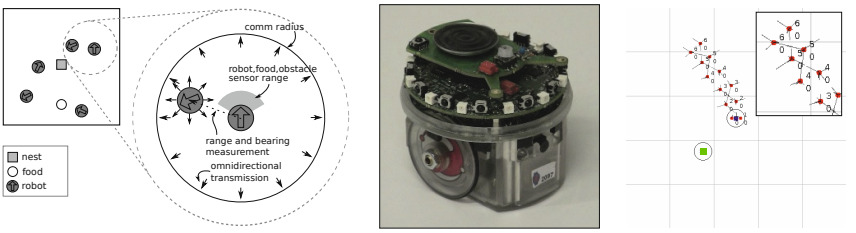
Algorithm Switching. Multiple behaviors within a single algorithm are well-known inside the swarm algorithm community [16]. Matarić and Arkin have worked extensively in behavior-based robotics [2, 4]. It is common for individual robots to switch between the behaviors of food collecting, obstacle avoidance, and resting, for example. Parker has studied distributed consensus in a swarm setting, using it to enable the swarm to move between subtasks in an overall task [17]. We focus on foraging, and do not use formal distributed consensus (or assume that our robots are '*well-stirred*'). McLurkin has developed a large range of robot swarm behaviors[12] as well as dynamic task assignment methods for individual robots within a swarm [14]. These methods focus on individuals, whereas we need a method for the swarm as a whole to switch algorithms.

The central contribution of this work is an adaptive foraging method in which the swarm makes colony-level decisions based on distributed information, choosing the algorithm best suited to the given food location.

2 Robot and Task Model

For our robots, we use a simple model inspired by recent swarm robot hardware, such as the E-Puck [15] (shown in figure 1b), the RBZ communication board (an E-Puck extension described in [1]), McLurkin's SwarmBots [13], and Payton's pherobots [5]. We assume a simple non-holonomic robot that moves and turns in continuous space. Each robot has sensors for nest, food, and obstacles in direct proximity to the robot. The sweeper algorithm also requires two of the robots to have compasses. Each robot can communicate with nearby robots and measure the range and bearing from which each transmission came. Robots do not have global position measurement or global communication. See figure 1a.

For the foraging task, we assume a world with a nest in the middle and one unlimited food source placed randomly. The swarm must find the food, then begin returning food units to the nest. Robots can pick up / drop food units when in direct



(a) (b) (c)

Fig. 1 The communication and sensing capabilities of the robots (1a), the E-Puck robot with communication ring (1b), and a simulation snapshot with a blowup for detail (1c).

proximity to the food / nest. Because the robots have no direct position measurement system, they must coordinate in order to maintain and share information about their own position and the food position.

To test the algorithms, we developed a continuous-world multi-robot simulator. The simulator models robots, food, and the nest, along with proper movement and interactions (collisions, communication). We chose a continuous world model over a gridded world environment so that the algorithms would face real problems such as collisions and congestion. A snapshot of the simulator is shown in figure 1c.

3 Algorithm Description

In this section, we will describe the three algorithms and the means by which algorithm switches are made at the individual- and colony-level. Figure 2 diagrams the relationship between the algorithms and their parts.

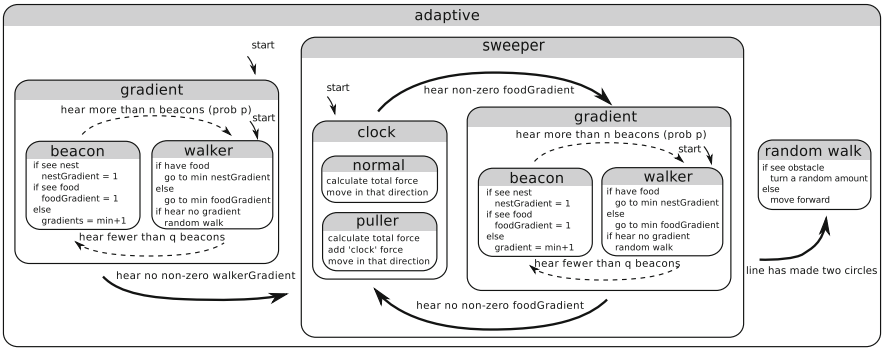


Fig. 2 This figure describes the relationships between the gradient, sweeper, and adaptive algorithms, and their switches. The "start" arrows indicate how each algorithm begins. Bold arrows indicate colony-level switches and dotted arrows indicate individual-level switches.

3.1 Gradient Algorithm

Robots need a way to navigate to the food and the nest. The gradient algorithm provides this information using two gradients—one leading to the nest, and a second leading to the food once it is found. To implement these gradients, some robots decide to stop their normal food searching and become fixed beacons. These beacons transmit two numbers (one for each gradient), which the remaining robots can use to navigate. As the swarm expands outward from the nest, the beacon network expands and creates the nest gradient. Once the food is found, the food gradient is developed. At that point, robots can navigate to either location to efficiently return food.

3.1.1 Local Description

Beacon: Robots acting as beacons are stationary, and broadcast two numbers, called `nestGradient` and `foodGradient`. They listen for all other beacons in their communication range and record the `nestGradient` and `foodGradient` of each one. Beacons find the minimum of all `nestGradient` values they have received, increment that by one, and take that as their own `nestGradient`. An analogous procedure is used to calculate `foodGradient`. These new values are then broadcast by the beacon. Any beacon directly next to the nest/food broadcasts a 1 for its `nestGradient`/`foodGradient`.

If a beacon has no information about its distance from the food (as happens early in the run, before the food has been found), it broadcasts 0. The value of 0 is treated specially—when a beacon hears a 0, it does not include it in its normal 'minimum plus one' calculation.

Walker: Walker robots always attempt to navigate either to the food or the nest, depending on whether they have food. In either case, a walker measures the bearing to the minimum gradient value toward the target of interest, and moves in that direction. If a walker has no information about where it should go (it can only hear 0), it does a random walk.

Beacon to Walker transition: If a beacon robot can detect more than 4 other beacons, it will become a walker robot with a 20% chance. This probabilistic effect is required to prevent several beacons, all of whom can collectively hear each other, from becoming walkers at exactly the same time and leaving a hole in the beacon field.

Walker to beacon transition: A walker robot will decide to become a beacon if it can only detect 1 or 2 other beacons.

3.1.2 Global Behavior

All robots start as walkers clustered around the nest. Some of them will decide to become beacons immediately because initially, there are no beacons. The remaining walkers will begin searching for the food. There will be no `foodGradient` (it will

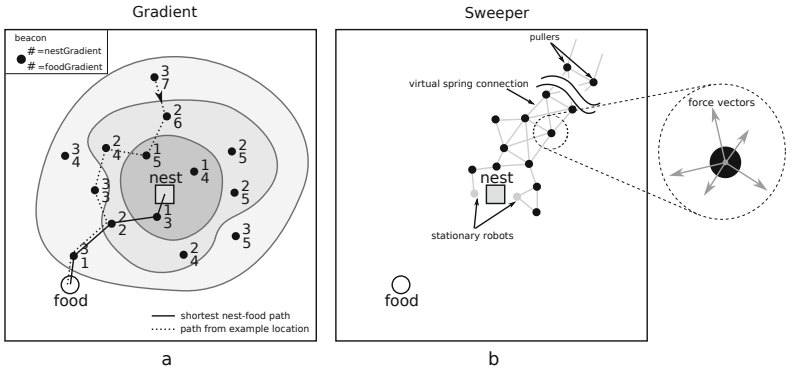


Fig. 3 Example situations in the gradient algorithm (a) and the sweeper algorithm (b). Walkers in (a) are not shown to reduce clutter. The gray contours roughly depict the gradient to the nest.

all be 0), so they will random walk. As they wander away from the nest, some will decide to become beacons, and the beacon field will expand away from the nest. Eventually one of the robots could stumble across the food, and would then begin transmitting 1 for its `foodGradient`, causing the food gradient to form. At this point, any walker can listen to the beacons near it and know how far it is from the food and how to get there. All the walkers immediately start moving directly toward the food. As walkers pick up food, they use the gradient field to bring it to the nest. Figure 3a illustrates an example snapshot of the gradient algorithm.

3.2 Sweeper Algorithm

A different strategy for search or foraging involves individuals forming a "search front" and systematically sweeping an area to find an object. Here we describe an algorithm that uses virtual forces to form a line of robots extending from the nest that sweeps the world like the hand of a clock. When the line finds food, some fraction of the robots remain as beacons while others act as walkers to return the food.

Fundamentally, this strategy creates a 1D structure of robots (roughly a line), as opposed to the gradient strategy which creates a 2D structure of robots (roughly a circle). The 1D structure is expected to be able to sweep a larger area than could be 'filled in' with the same number of robots

3.2.1 Local Description

Normal: In the sweeper algorithm, all robots are always transmitting. Each robot measures the range and bearing to all the other robots in its communication range. Based on the position of each other robot, it calculates a virtual force on itself. For each robot detected at relative position $\overrightarrow{r}$, this force is

$$\overrightarrow{F} = a\frac{\overrightarrow{r}}{r_c} - b\hat{r}$$

where r_c is the communication range of the robot and a and b are empirically chosen constants. In other words, the force is similar to what would be experienced if there were a virtual spring between the robots. The robot sums the virtual forces from each other robot in its communication range, and moves in that direction by an amount proportional to the magnitude of the force. There is one special case: two robots directly next to the nest never move regardless of the virtual forces on them.

While the robots are calculating forces and moving, they are simultaneously using communication to establish a gradient field similar to the one described in the gradient algorithm. This does not require extra communication. The data content of the signal encodes the two gradient values, and range and bearing to the transmitter are used to calculate the virtual forces.

The robot treats the `nestGradient` exactly as before, updating it using the $min+1$ algorithm. `foodGradient` is treated slightly differently. Any time a robot sees a non-zero `foodGradient`, it temporarily stops executing the sweeper algorithm and switches to gradient. If the `foodGradient` returns to zero, the robot returns to executing the sweeper algorithm.

Puller: Two robots are pre-determined to be "puller" robots. These participate in the virtual forces system described above, but they also feel one additional force. These two robots must use their compasses to measure the relative bearing to north (the unit vector $\hat{N}$), then put a virtual 'clock' force on themselves equal to

$$\overrightarrow{F_c} = c\hat{N}R\left(\frac{2\pi t}{T}\right)$$

where R is simply the 2D rotation matrix,

$$R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

$\overrightarrow{F_c}$ is a force which simply rotates around like the hand of a clock as time t increases. The parameter c is an empirically chosen magnitude and T sets the period of the rotation. T is determined by the puller, and can be changed at any time based on the puller's `nestGradient` value.

3.2.2 Global Behavior

When the sweeper algorithm begins, all robots calculate forces and begin moving appropriately. Initially, repulsive forces cause the swarm to expand into a tight clump around the nest. The pullers will be forced to the edge of the pack and a line of robots will form extending from the nest to the pullers. This line of robots will rotate as the pullers pull it around, sweeping around the world like the hand of a clock. When the line encounters food, it stops moving and the swarm returns the food using the gradient algorithm, with walkers moving along the line of robots

already established. When the food source is exhausted, the forces resume and the line keeps sweeping.

One can imagine that longer lines (with more robots) would need to rotate slower than shorter lines. Hard-coding the period T would require knowing the number of robots in the swarm, which is not a scalable solution. Instead, the pullers set T based on their `nestGradient`. A higher value for `nestGradient` indicates a long line, so the puller will choose a larger T.

Figure 3b illustrates an example snapshot of the sweeper algorithm.

3.3 Adaptive Algorithm

The first two algorithms, gradient and sweeper, each have strengths and weaknesses. Gradient operates in a short range but is fast, while sweeper has a longer range but is much slower. (This is quantified in section 4.) The adaptive algorithm combines the benefits of these two algorithms, along with random walk, by trying each one in sequence and choosing the best one for a given situation. It first tries gradient, which would work well if the food is near enough to use it. If not, it switches to sweeper to get food further away. If it still doesn't find the food, it switches to the last resort – random walk. Switches between these three algorithms are made in a distributed manner at the colony level. To accomplish this, a third gradient is included.

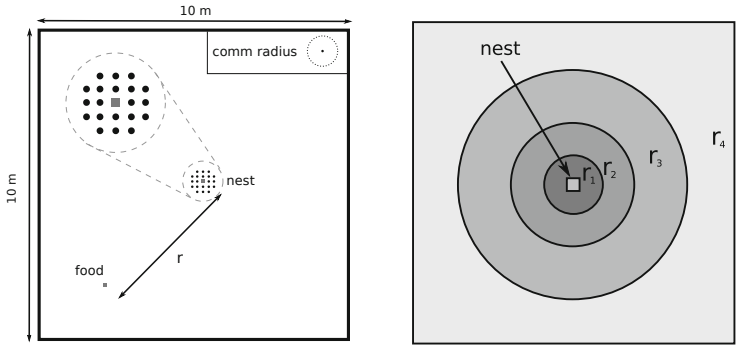
3.3.1 Local Description

Robots begin with an algorithm very similar to the gradient algorithm above. They are split between walker and beacon robots as before, but they maintain three gradients as opposed to two. The third one measures how far each beacon is from any walker robot. This requires all walker robots to transmit a single bit of information indicating their presence and identity as a walker. The beacons then transmit a 1 for the `walkerGradient` if they can see a walker, and $min + 1$ if they can not see a walker. Other than this, they execute the gradient algorithm exactly as described above.

To implement adaptive foraging, robots need to explicitly detect when to change algorithms. If a robot does not see any (non-zero) `walkerGradient` for several time steps in a row, it will completely switch algorithms from gradient to sweeper. Thereafter, if the line of robots has swept the world twice and still has not found food (as evaluated by the pullers), the pullers send a signal through the beacon network causing every robot to again switch algorithms to random walk.

3.3.2 Global Behavior

The swarm will begin executing the gradient algorithm. There are many walkers at the beginning of the execution, so the values for `walkerGradient` are all fairly low. If the swarm finds the food, it returns it as usual, and the `walkerGradient` remains irrelevant. If, however, the swarm expands to the point that all robots have



(a) (b)

Fig. 4 Test setup is shown in figure 4a, drawn to scale. The parameter r ranges from 1m to 4m. Figure 4b shows the regions described in section 4.

become beacons and the swarm has still not found the food, then there will be no walkers left. When the last walker becomes a beacon, suddenly no beacon anywhere in the swarm has information on which to broadcast a `walkerGradient`, so all `walkerGradient` values revert to 0. A short time later, they all decide nearly-simultaneously to switch algorithms and begin the sweeper algorithm. From this point on, the swarm proceeds as normal as if it had just begun the sweeper algorithm, pulling out a line of robots and sweeping the world. Once this line has swept around the world twice without finding food, the swarm switches to random walk, which is the only option left. Random walk does not involve coordination, which relieves the robots of the requirement of staying near each other. This is the only way to get food so far away.

4 Performance

These algorithms were tested in a continuous-world multi-agent simulator (screenshot in figure 1c). An unlimited food source was placed at varying distances from the nest, with a swarm of 20 robots trying to find and retrieve it (as diagrammed in figure 4a). Since we are focusing on algorithm switching and the environmental impact, we chose to experiment with a fixed number of robots. In the future, we will study scalability of the individual algorithms more closely.

We assessed the performance of the algorithms using three simple metrics: (1) whether or not the swarm found the food, (2) how quickly it found the food, and (3) the rate at which it returned the food to the nest. Each data point represents an average of 100 runs.

Region-Based Analysis. Based on performance (figure 5), we can see that the world can be divided into four distinct regions, diagrammed in figure 4b.

region	description
r_1	any algorithm works
r_2	coordination needed, gradient works well
r_3	too far for gradient, sweeper works well
r_4	too far for sweeper, only random walk works

If the food is inside r_1, it is so close to the nest that any algorithm will find it (figure 5a), find it quickly (figure 5b), and return it quickly (figure 5c).

r_2 is the boundary inside which the gradient method works well. It finds the food, finds it quickly, and returns it quickly. Outside of r_2, gradient works poorly. As the robots expand and form a beacon field, eventually the swarm will expand to its maximum size and there will be no walkers left to continue the expansion. If the food is beyond this critical radius (r_2), there is no way for the gradient method to get it. The sweeper algorithm is also capable of finding the food inside r_2, but as seen in figure 5b, takes much more time to do it. The adaptive algorithm is able to choose the gradient method in this region, finding food quickly with a high success rate.

In r_3, the gradient algorithm is useless, and the sweeper algorithm performs well, forming a line and sweeping the world out to approximately r_3, although this boundary is less well defined. It finds the food but takes a long time to do it. In this region, the adaptive algorithm correctly selects sweeper.

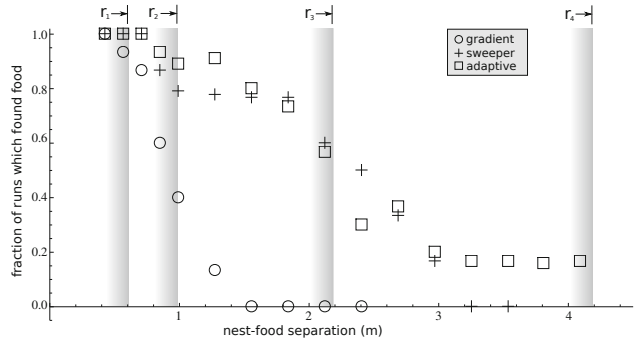
Outside r_3, even the sweeper algorithm fails because the line of robots can not reach that far. In r_4, the adaptive algorithm switches to random walk. This works poorly ($\sim 20\%$ success rate, slow to locate and return food), but beyond about 3m, it is the only method capable of finding any food at all.

In every region, the adaptive algorithm is able to choose the most appropriate foraging method. In r_2 it runs the gradient method, in r_3 it runs the sweeper method, and in r_4 it runs random walk.

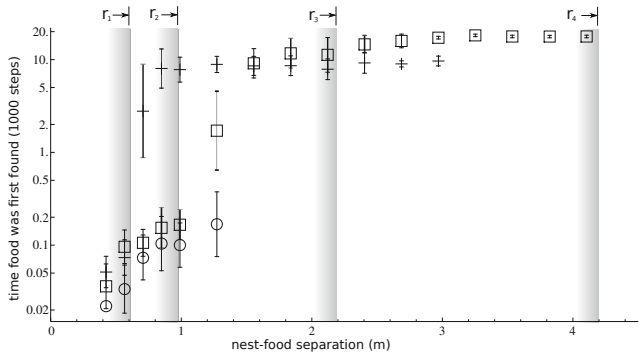
Overall Assessment. As an overall assessment of each algorithm, we can place food in an unknown random location, and measure the performance. In the table below, "r_1, r_2, or r_3" indicates that the food is placed randomly anywhere in those three regions, and "whole world" indicates that the food is randomly placed anywhere. The numbers are averages over all placements. For example, if the food is randomly placed anywhere and the sweeper algorithm is running, the swarm can be expected to find it 32% of the time, after an average of 7300 time steps with a standard deviation of 3500 time steps. (One time step roughly corresponds to one second.)

Table 1 Overall performance assesment

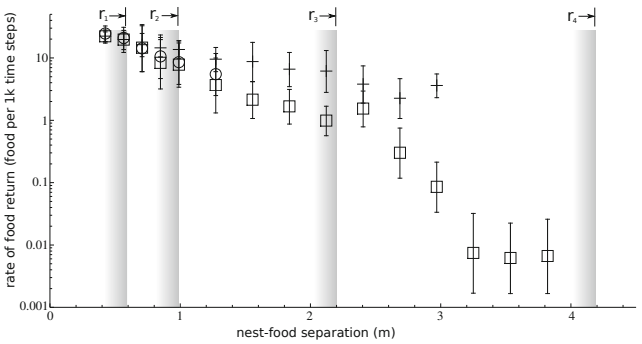
	r_1, r_2, or r_3		whole world	
algorithm	success rate	time food found	success rate	time food found
gradient	31%	69 ± 34	20%	69 ± 34
sweeper	82%	6500 ± 3900	32%	7300 ± 3500
adaptive	86%	5500 ± 3700	46%	9200 ± 6000



(a)



(b)



(c)

Fig. 5 Figure 5a shows the success rate of each algorithm. Figure 5b shows the time at which food was first found. Figure 5c shows the rate at which food is returned to the nest once it is found. Each point represents an average of 100 runs, and the error bars indicate one standard deviation. Note the log scales in the second two plots.

In r_1, r_2, or r_3, the adaptive algorithm finds the food almost as often as the sweeper algorithm, but does so faster. This is because it is able to take advantage of the speed of the gradient method when the food is nearby. When very distant food locations are included (whole world), all algorithms suffer lower success, but the adaptive algorithm is able to use random walk to at least achieve some success in a very long time.

5 Algorithm Switching Generalizations

Algorithms other than those presented in this paper could potentially be combined into a single adaptive algorithm using the same method. The critical requirement is the connectedness of the communication network. This section will discuss several generalizations we can draw about colony-level algorithm switching, beyond the specific cases discussed in this paper.

There are two kinds of algorithm switches: individual switches and colony switches (see figure 2), with two main differences between the types. First, individual switches are made by a particular robot based on the information it can perceive, whereas colony-level switches require information which is distributed throughout the environment and not directly perceivable by every robot making the decision. Second, colony-level switches must be nearly synchronized, whereas individual switches need not be explicitly coordinated.

To achieve a colony-level algorithm switch, information must be shared throughout the swarm, because each robot requires global-level knowledge in order to decide to switch algorithms. For example, when the adaptive algorithm switches from gradient to sweeper, each robot must be aware that no walkers are left, but no single robot is capable of perceiving this. This global information is detected and shared through the beacon network. Because global information is required for colony-level switches, maintaining the connectedness of the beacon network is critical for these switches.

We can distinguish three types of information:

information type	example from this paper
directly perceivable	sense a beacon ahead
directly perceivable by another robot	someone found food
only perceivable by swarm as a whole	swarm has expanded to max size

Individual switches can be made solely based on information of the first type. Colony switches require the second and third types, which requires a beacon network. There is no robot in the swarm with a sensor capable of detecting the third type of information; it can only be detected by the swarm as a whole through cooperation. A connected network is critical for detecting and transmitting this information.

6 Conclusion

We have presented two distributed foraging algorithms which perform best under different food locations, and a third method in which the swarm as a whole can choose the best algorithm for the given situation. The gradient algorithm can return nearby food quickly, and the sweeper algorithm can find food further away but is much slower. The adaptive algorithm uses the gradient, sweeper, and random walk methods, detecting in a distributed manner if one has failed and switching to the next. For food in any region of the world, the adaptive method is able to choose the most appropriate algorithm. Colony–level algorithm switching requires communication, but can combine benefits of multiple algorithms and improve overall performance.

There are several possible improvements and expansions planed for the future, both in hardware and software. Although these algorithms are designed to be scalable, scalability will be tested experimentally. Second, we will consider methods by which swarms could switch algorithms in a more general manner, including dynamic environments (which could require them to switch back to previously tried algorithms). Finally, we will conduct a hardware study on the effect of sensor / communication capability on algorithm possibilities and performance. This study will use the E-Puck robots and IR communication rings described earlier.

References

1. Gutierrez, A., et al.: Open E-Puck Range & Bearing Miniaturized Board for Local Communication in Swarm Robotics. In: ICRA (2009)
2. Arkin, R.: Behavior-Based Robotics. MIT Press (1998)
3. Balch, T., Hybinette, M.: Social potentials for scalable multi-robot formations, pp. 73–80 (2000)
4. Jones, C., Mataric, M.: Behavior-Based Coordination in Multi-Robot Systems. Autonomous Mobile Robots: Sensing, Control, Decision-Making, Applications
5. Payton, D., Daily, M., Estkowski, R., Howard, M., Lee, C.: Pheromone Robotics. Autonomous Robots 11(3), 319–324 (2001)
6. Barth, E.: A dynamic programming approach to robotic swarm navigation using relay markers. In: Proceedings of the 2003 American Control Conference, vol. 6, pp. 5264–5269 (2003)
7. Howard, A., Mataric, M., Sukhatme, G.: Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In: Sixth Int. Symposium on Distributed Autonomous Robotics Systems, pp. 299–308 (2002)
8. Ichikawa, S., Hara, F.: Experimental characteristics of multiple-robots behaviors in communication network expansion and object-fetching. In: Distributed Autonomous Robotic Systems, pp. 183–194 (1996)
9. Svennebring, J., Koenig, S.: Building Terrain-Covering Ant Robots. Autonomous Robots 16(3), 313–332 (2004)
10. O'Hara, K., Walker, D., Balch, T.: The GNATs Low-cost Embedded Networks for Supporting Mobile Robots, pp. 277–282 (2005)
11. Mamei, et al.: Spreading Pheromones in Everyday Environments via RFID Technologies. In: 2nd IEEE Symposium on Swarm Intelligence (2005)

12. McLurkin, J.: Stupid robot tricks: A Behavior-Based distributed algorithm library for programming swarms of robots. S.M. thesis, MIT (2004)
13. McLurkin, J.: Measuring the accuracy of distributed algorithms on Multi-Robot systems with dynamic network topologies. In: 9th International Symposium on Distributed Autonomous Robotic Systems, DARS (2008)
14. McLurkin, J., Yamins, D.: Dynamic task assignment in robot swarms. In: Proceedings of Robotics: Science and Systems, June 8 (2005)
15. Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.-C., Floreano, D., Martinoli, A.: The e-puck, a robot designed for education in engineering. In: Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions, vol. 1(1), pp. 59–65 (2009)
16. Nouyan, S., Groß, R., Bonani, M., Mondada, F., Dorigo, M.: Teamwork in self–organized robot colonies. IEEE Transactions on Evolutionary Computation 13(4), 695–711 (2009)
17. Parker, C., Zhang, H.: Collective unary decision-making by decentralized multiple-robot systems applied to the task-sequencing problem. Swarm Intelligence
18. Vaughan, R., Stoy, K., Sukhatme, G., Mataric, M.: LOST: Localization-space trails for robot teams. IEEE Trans. on Robotics and Automation 18(5), 796–812 (2002)
19. Schmickl, T., Crailsheim, K.: Trophallaxis within a robot swarm: Bio-inspired communication among robots in a swarm. Autonomous Robots 25, 171–188 (2008)
20. Spears, W., Spears, D., Hamann, J., Heil, R.: Distributed, physics-based control of swarms of vehicles. Autonomous Robots 17(2-3), 137–162 (2004)
21. Sugawara, K., Kazama, T., Watanabe, T.: Foraging behavior of interacting robots with virtual pheromone. In: IROS 2004, vol. 3, pp. 3074–3079 (2004)
22. Yoon, S., Soysal, O., Demirbas, M., Qiao, C.: Coordinated locomotion of mobile sensor networks. In: 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, pp. 126–134 (2008)

On Fault Tolerance and Scalability of Swarm Robotic Systems

Jan Dyre Bjerknes and Alan F.T. Winfield

Abstract. This paper challenges the common assumption that swarm robotic systems are robust and scalable by default. We present an analysis based on both reliability modelling and experimental trials of a case study swarm performing team work, in which failures are deliberately induced. Our case study has been carefully chosen to represent a swarm task in which the overall desired system behaviour is an emergent property of the interactions between robots, in order that we can assess the fault tolerance of a self-organising system. Our findings show that in the presence of worst-case partially failed robots the overall system reliability quickly falls with increasing swarm size. We conclude that future large scale swarm systems will need a new approach to achieving high levels of fault tolerance.

1 Introduction

Research papers in Swarm Robotics frequently assert that swarm robotic systems are both scalable and robust. The fact that individual robots in the swarm make decisions based only on local sensing and communication is assumed to lead naturally to swarms that will scale to very large numbers of robots; the high degree of parallelism in robot swarms, which typically consist of homogeneous robots, is assumed to lead to a high level of robustness and dependability. While it may be true that robot swarms can exhibit an unusual level of tolerance to failure of individual robots, or external threats, when compared with conventionally engineered distributed systems, it is not safe to assume that scalability and robustness are automatically properties of all (or any) swarm systems. It is surprising therefore that, in the field of swarm

Jan Dyre Bjerknes · Alan F.T. Winfield
Bristol Robotics Laboratory, University of the West of England, Bristol, UK
e-mail: jandyre@tankeogteknikk.no,Alan.Winfield@uwe.ac.uk

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 431–444.
springerlink.com

robotics, there has been relatively little systematic study of dependability and fault tolerance. In previous papers we have argued for a systematic approach to engineering dependable swarms [15], and started to consider fault tolerance in robot swarms [16]. A recent paper by Christensen et al notably proposed a swarm algorithm, inspired by synchronised flashing seen in fireflies, in which failed robots can be detected and physically removed by operational robots [7]. In [10], Marino et al analyse, in simulation and real robot experiments, the tolerance to failures of a multi-robot team of border patrol robots.

In this paper we develop a reliability model for a case study swarm of robots that exhibit emergent, or self-organised, swarm taxis. After describing the swarm algorithm in Section 2, we outline the key failure modes for the case study swarm, and our experimental setup. In Section 3 we show that we can model this swarm – from a reliability perspective – as a k-out-of-N system. We then extend the k-out-of-N reliability model to take account of worst-case partial robot failures and swarm scaling properties in Section 4, introducing the new concept of swarm self-repair. Section 5 concludes the analysis with a model of reliability as a function of swarm size and hence addresses the question of scalability.

2 Case Study: Emergent Swarm Taxis

For our experimental case study we make use of a swarm of e-puck robots [11] with two swarm behaviours: flocking and swarm taxis toward a beacon. The combination means that the swarm maintains itself as a single coherent group while moving toward an infra-red (IR) beacon. The algorithm is a modified version of the wireless connected swarming algorithm (the α-algorithm) developed by Nembrini et al [12, 14].

Our modified algorithm, which we refer to as the ω-algorithm, works as follows. Flocking is achieved with the well-known combination of short-range repulsion and longer-range attraction. Short-range repulsion is implemented with obstacle avoidance behaviour using the e-puck's IR proximity sensors. Longer-range attraction (coherence) is achieved as follows. Each robot times the duration since it last made an avoidance manoeuvre and if that value exceeds a given threshold ω, the robot turns towards its estimate of the centre of the swarm; an estimate based on readings from the ring of infrared proximity sensors around the e-puck's body. To increase the distance at which robots can sense each other, and also to enable robots to distinguish between robots and ambient infra-red, each of the robots are equipped with infra-red emitters that flash at 80 Hz. By sampling the sensors at 400 Hz and passing the data through a bandpass filter the 80 Hz flashing is reliably detected. Each robot can then estimate the direction of the local centre of the swarm based on which of its sensors detect a flashing signal from other robots. For

the results obtained from hardware trials reported here we set $\omega = 2.5$ s; ω (like α) controls the overall swarm density.

For beacon-taxis, we implement an additional 'beacon' sensor on each robot. The beacon sensor is deliberately minimal, in that it is unable to detect the range and bearing of the remote beacon and has only a two-state output: *on = illuminated* or *off = not-illuminated*. An important requirement of the beacon sensor is that it can be occluded by other robots, thus those robots that have a direct line-of-sight to the beacon will have beacon sensors illuminated, and those robots that are in the shadow of other robots will have beacon sensors not-illuminated. This means that for a typical swarm only the robots on or close to the leading edge of the swarm (with respect to the beacon) will have illuminated beacon sensors. Our experimental trials make use of the same IR sensors, that are used for short-range collision avoidance and longer-range coherence, for beacon sensing.

We then introduce a simple symmetry breaking mechanism. We set the short-range avoid sensor radius for those robots that are illuminated by the beacon to be slightly larger than the avoid sensor radius for those robots in the shadow of other robots. This simple mechanism results in a net swarm movement (taxis) toward the beacon. Note that the swarm taxis is an emergent property of the swarm: with a simple two-state beacon sensor a single robot cannot sense the direction of the beacon, and even with the symmetry breaking mechanism two or three robots are not enough to give rise to emergent swarm taxis; experimentally we find that swarm taxis requires at least five robots. This is important to our case study as we are interested in determining the reliability of a swarm with emergent swarm behaviours. For a detailed analysis of the swarm taxis behaviour see [5], and for implementation details and code listings see [4].

2.1 Failure Modes and Effects

This paper is concerned with analysis and modelling of reliability in swarm robotic systems and so we need to understand which faults, in individual robots, might seriously affect the operation of the overall swarm. We can summarise the failure modes and effects for our case study swarm as follows:

- *Case 1: complete failures of individual robots.* These are relatively benign, in the sense that 'dead' robots simply become obstacles in the environment to be avoided by the other robots of the swarm. Completely failed robots (due, for instance, to a power failure) might have the effect of slowing down the swarm taxis toward the beacon, but – as shown later in Section 4 – this effect is marginal. The only situation in which complete failures could be critical is if they reduce the number of working robots in the swarm below the minimum number for the self-organising team work to function. This eventuality is modelled by the k-out-of-N approach in Section 3.

- *Case 2: failure of a robot's IR sensors.* While highly unlikely given that there are 8 IR sensors fitted to the e-puck, this could conceivably result in the robot leaving the swarm and becoming lost. Such a robot would become a moving obstacle to the rest of the swarm and might – as in Case 1 above – reduce the number of robots required for team work. This situation is thus also modelled by the k-out-of-N approach in Section 3.
- *Case 3: failure of a robot's motors only.* Motor failure only leaving all other functions operational, including IR sensing and signalling, will have the potentially serious effect of causing the partially-failed robot to 'anchor' the swarm, impeding its taxis toward the beacon. If both motors fail the robot will be 'live' but stationary; if only one motor fails the robot will turn on the spot, which amounts to the same thing. Of the 3 cases this is by far the most serious, and will be analysed, and modelled, in Section 4.

2.2 Experimental Trials

Experimental trials have been conducted with a swarm of 10 e-puck robots. Fig. 1(a) shows a trial of emergent swarm taxis, with no failures, in progress. Videos of typical experimental trials, with a speed-up of 25x, have been uploaded to YouTube for (a) no failures [3], (b) two simultaneous Case 1 (complete) robot failures [1], and (c) 2 simultaneous Case 3 (partial) robot failures [2]. Note that in this particular Case 3 trial (c), 2 healthy robots become trapped by the 2 partially failed robots, and only 6 robots reach the beacon.

(a) (b)

Fig. 1 (a) Hardware trial of emergent swarm taxis using 10 e-puck robots. The swarm is moving toward the IR beacon located on the RHS of the arena. (b) An e-puck fitted with an opaque 'skirt' required to block IR light from passing through the transparent e-puck body. Also note the yellow 'hat' which provides a matrix of pins for the reflective spheres which allow the position tracking system to identify and track each robot.

3 The k-Out-of-N Reliability Model

The purpose of a reliability model is to enable the estimation of overall system reliability, given the (known) reliability of individual components of the system, see [8]. Reliability R is defined as the probability that the system will operate without failure, thus the unreliability (probability of failure) of the system, $P_f = 1 - R$. In our case the overall system is the robot swarm and its components are the individual robots of the swarm.

From a reliability modelling perspective a swarm of robots is clearly a parallel system of N components (robots). If the robots are independent, with equal probability of failure p, then the system probability of failure is clearly the product of robot probabilities of failure. Thus, for identical robots, $R = 1 - p^N$. p can be estimated using a classical reliability block diagram approach on the individual sub-systems of the robot. Since the individual robot does not internally employ parallelism or redundancy then its reliability will be modelled as a series system, giving p less than the worst sub-system in the robot, which is most likely to be its motor drive system. However, this simplistic modelling approach makes a serious and incorrect assumption, which is that the overall system remains fully operational if as few as one of its components remains operational. This is certainly not true of our case study swarm. The desired emergent swarm behaviours require the interaction of multiple robots and our swarm beacon taxis behaviour is a dramatic example: with one robot only the behaviour simply cannot emerge. It is a frequent characteristic of swarm robotic systems that the desired overall swarm behaviours are not manifest with just one or a very small number of robots. However, the question of how many (or few) robots are needed in order to guarantee a required emergent behaviour in a particular swarm and for a particular behaviour is often not straightforward.

Thus, from a reliability perspective, we propose that the swarm must be be modelled as a k-out-of-N:G system. That is, a system of N parallel elements which requires that at least k of these elements are operational (Good) for the overall system to function correctly. In a swarm of N robots, if more than $N - k$ fail, the self-organised functionality of the overall swarm will be compromised.

In a k-out-of-N:G system, the probability that *at least* k out of N robots are working at a given time t is given, from [9], by:

$$P(k, N, t) = \sum_{i=k}^{N} \binom{N}{i} (e^{-t\lambda})^i (1 - e^{-t\lambda})^{N-i} \tag{1}$$

where $\lambda = \frac{1}{MTBF}$. MTBF is the mean time before failure of an individual robot.

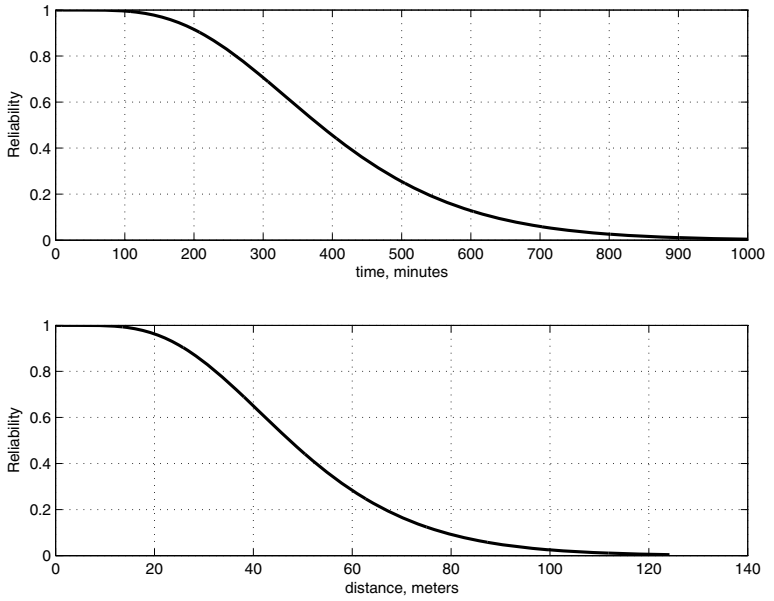


Fig. 2 Top: The reliability of a robot swarm modelled as a k-out-of-N system, with $k = 5$, swarm size $N = 10$ robots and MTBF = 480 m. Bottom: Reliability of the same swarm as a function of distance travelled, based on a measured mean swarm velocity of 12.4 cm. per min. for a swarm of 10 robots.

Based on Eq. 1 we can now plot swarm reliability against time for our case study swarm. Experimental trials indicate that at least five robots have to be working in order for the emergent swarm taxis behaviour to work properly. Thus, we can model our swarm as a 5-out-of-N system. Consider now the individual robots' MTBF. Carlson et al. tracked failure data for 13 robots by three different manufacturers over a period of two years. They found the MTBF to be eight hours [6]. Experiments with the e-pucks used in our experimental trials might suggest that their failure rate might be higher (because of the design of the e-puck battery connector). However, as no systematic data is available, the value reported by Carlson et al. will be used here. Fig. 2 (top) plots Eq. 1 for a swarm of ten robots, and shows that the swarm reliability starts to decline rapidly after 100 minutes of operation.

Fig. 2 (bottom) plots the reliability of the same swarm of ten robots, with the same values for k and MTBF, against the distance the swarm will travel (the emergent swarm taxis behaviour) based on a measured mean swarm velocity of 12.4 cm per minute for a swarm of 10 robots.

Although providing some insight, the reliability assessments based on the k-out-of-N model here fail to take into account two important factors. Firstly, each robot that fails is likely – depending on the exact nature of that failure – to slow down the swarm; if the failed robot(s) are immobile then the swarm

will slow down until it 'escapes' from the failed robots, leaving them behind. Secondly, the swarm velocity might then change after the failed robot(s) have been left behind, typically a smaller swarm (of at least 5 robots) will have a higher swarm taxis velocity. We now analyse these factors in more detail in order to improve the swarm reliability model.

4 Swarm Self-repair

We now introduce the concept of *swarm self-repair*. Consider the case-study swarm and its failure modes and effects analysis outlined above in Sect. 2.1. Our experimental trials confirm the failure modes and effects analysis of Sect. 2.1 and demonstrate that, while all failure modes have the effect of slowing down swarm progress toward the beacon, the swarm is tolerant to the simultaneous (i.e. worst case) failure of more than one robot. Furthermore, we notice two different categories of effect on the overall swarm: (i) sensor failures (Case 2) which slow down progress of the swarm, but the whole swarm reaches the beacon and (ii) motor failures (Cases 1 and 3) which hold back progress of the swarm until the swarm breaks free of the failed robots; for a detailed analysis of these results see [4]. Consider the second, and more serious category (ii), which gives rise to the notion of swarm self-repair.

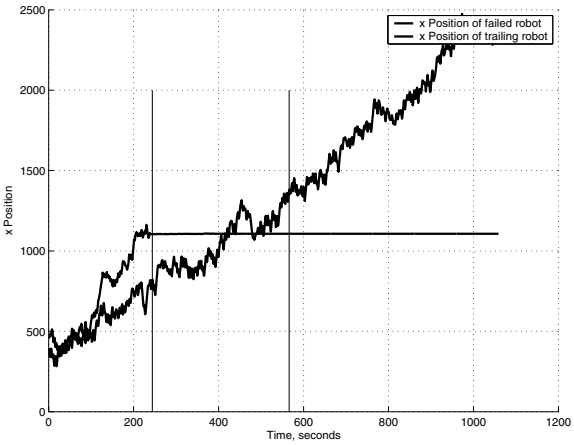


Fig. 3 Hardware trials using 10 e-puck robots: single robot complete failure Case 1, swarm self-repair time. Two robots are tracked: the failed robot and the trailing robot from the rest of the swarm. At about 250 s. a single robot on the leading edge of the swarm experiences Case 1 failure; at about 580 s. the trailing robot leaves the failed robot. In this case the failed robot is simply a static obstacle to the swarm, to be avoided as the swarm moves toward the beacon.

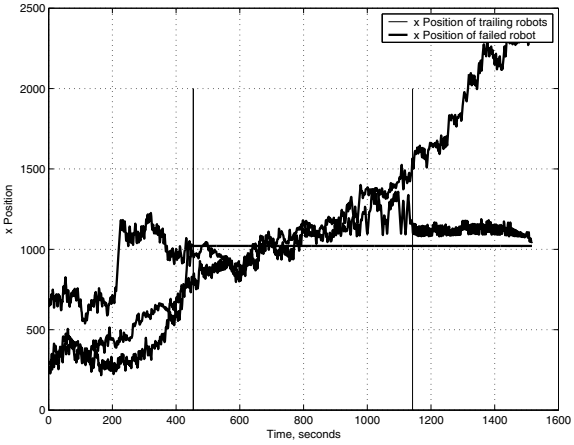


Fig. 4 Hardware trials using 10 e-puck robots: single robot partial failure Case 3, swarm self-repair time. Three robots are tracked: the failed robot, the trailing robot from the rest of the swarm and a third healthy robot left behind with the failed robot. At about 450 s. a single robot on the leading edge experiences Case 3 failure; at about 1150 s. the trailing robot 'escapes' the failed robot. Here the partially failed robot actively holds back the swarm - as outlined in Sect. 2.1 Case 3. To escape, the 'pull' of the swarm taxis needs to overcome the anchoring force of the failed robot. The healthy robot that is, by chance, left behind remains attracted by - and in the orbit of - the partially failed robot.

Refer to Figs. 3 and 4. We define swarm self-repair time as the time between (simultaneous) motor failure of one (or more) robots and the point at which the trailing robot in the rest of the swarm escapes the influence of the failed robot(s). This is a useful metric because it varies with both the type of robot motor failure (Cases 1 or 3) and the number of robots. Table 1 lists the measured swarm self-repair times for one and two simultaneous failures for Cases 1 and 3. For comparison the table also shows a baseline notional self-repair time: the time the swarm would take to leave behind a failed robot if that robot failure did not slow down the swarm.

Table 1 Mean swarm self-repair times for the case study swarm of $N = 10$ e-puck robots. Ten runs for each case. *Here the swarm reached the beacon in only 6 of 10 runs.

Case	Mean (s)	Std. Dev. (s)
Baseline (no penalty)	328	174
One failed robot Case 1	387	132
Two failed robots Case 1	453	172
One failed robot Case 3	879	417
Two failed robots Case 3	1279	see note*

5 Swarm Scaling and Reliability

We have argued that in the k-out-of-N reliability model above, the minimum value of $k = 5$ because the swarm taxis property is present even with as few as 5 robots. For $N = 10$ robots and an MTBF of 8 hours, this reliability model suggests that the swarm will become unreliable after approximately 100 minutes. While it is clear that we can increase the swarm reliability by increasing the individual robots' MTBF, can we also make the swarm more reliable by increasing swarm size? At first it might seem plausible to suggest that the increased redundancy in a larger swarm would maintain reliability for a longer period. One may even be led to believe that the swarm could be made reliable for an arbitrarily long time, given a sufficiently large number of robots. This is not correct, and we now combine a model of swarm self-repair with the k-out-of-N model to determine the maximum upper size for our case-study swarm.

Consider the argument informally. When a swarm is larger it will take longer to self-repair than a smaller swarm. There are two reasons for this. Firstly, it is a property of our case study swarm that the swarm taxis velocity reduces with increasing swarm size. Secondly, the swarm is physically larger and must move a longer distance before it is fully self-repaired. Thus the self-repair rate will *remain constant* with increased swarm size. However, for a given robot MTBF, the swarm failure-rate will *increase* for larger swarms. It is unavoidable that at some point the failure rate will overtake the self-repair rate of the swarm, and the swarm will come to a complete halt - the desired emergent swarm-taxis property will fail. In fact a swarm of sufficient size would die under it own weight, so to speak, before it has even started to move.

We now estimate the values of k and self-repair time t_s as a function of N. We will then use these values, together with the k-out-of-N model Eq. 1, to estimate swarm reliability as a function of swarm size.

5.1 The Value of k

In experimental tests it is clear that, for complete failures Case 1, two out of ten robots could fail without permanently damaging the swarm. The swarm would always self-repair. The cases with partial failure Case 3 fared less well. When one out of ten robots failed, the swarm did always self repair, even though a functioning robot might occasionally become stuck with the failed robot. But when two out of ten robots failed, the swarm would suffer a complete breakdown in four out of ten cases, and in the remaining six cases, as many as three healthy robots stayed behind with the failed robots.

Based on this the value of k will be conservatively estimated as 90% of N for a k-out-of-N:G system. In other words, when the swarm has ten percent failed robots or less it will be assumed that it can self repair. Arguably, this

may not hold true for larger swarms - the empirical evidence is limited to swarms with ten robots. But this is our best estimate from the evidence available.

5.2 The Value of t_s

We know from an analysis of the scaling properties of our case study swarm [4], that swarm-taxis velocity v as a function of N follows this relationship:

$$v(N) = CN^{-\frac{1}{2}} \tag{2}$$

Where C is a scaling constant. Thus larger swarms move more slowly. Note, as stated already, that the minimum value of swarm size N for the swarm to exhibit swarm taxis is 5, thus Eqn. 2 is not valid for $N < 5$.

Clearly, the diameter d, of the swarm will increase with swarm size.

$$d(N) = D\sqrt{N} \tag{3}$$

Where D is the density constant for the swarm.

Since a robot can fail anywhere within the swarm: on the leading edge, in the middle of the swarm or at the trailing edge, the average distance that the swarm needs to move before it has moved away from the failed robot will be half the diameter, $\frac{d}{2}$. Thus the self-repair time becomes $t_s = \frac{d}{2v}$.

Thus,

$$t_s(N) = \frac{D\sqrt{N}}{2C\frac{1}{\sqrt{N}}} \tag{4}$$

Which simplifies to

$$t_s(N) = \frac{D}{2C}N \tag{5}$$

Eq. 5 is important as it demonstrates that the self-repair time increases linearly with N. Based on this equation it is now possible to introduce a new constant for a given swarm, namely the self-repair-time-constant. Let this constant have the symbol S for Self-repair, where $S = \frac{D}{2C}$. Now we have established that S is linear with N, we can determine its value experimentally. For a swarm with ten robots with one partially failed robot the mean self-repair time was found to be 879 s (see table 1). This was for a case with ten robots, so the self-repair constant for our case study swarm, for Case 3 partial failures, then becomes $S = \frac{879}{10} = 87.9$.

5.3 Swarm Scalability

Using the estimated values for k and t_s and the k-out-of-N reliability model we can now plot swarm reliability against swarm size N.

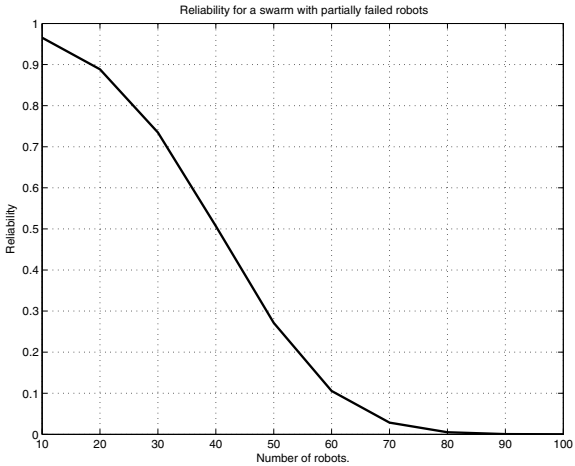


Fig. 5 Reliability of the case study swarm as a function of swarm size, based on a k-out-of-N reliability model and assuming Case 3 partially failed robots; $k = 0.9N$, self-repair-time-constant $S = 87.9$ and robot MTBF 8 h

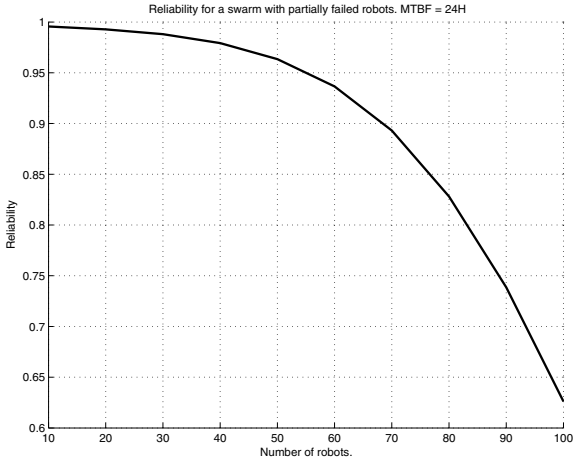


Fig. 6 Reliability of the case study swarm as a function of swarm size, based on a k-out-of-N reliability model and assuming Case 3 partially failed robots; $k = 0.9N$, self-repair-time-constant $S = 87.9$ and robot MTBF 24 h

Fig. 5 shows that with an MTBF of 8 hours, a swarm with as few as 40 robots will have a reliability of only 0.5. This reliability model is based on a number of assumptions (including, for instance, a circular swarm morphology that remains constant with increasing swarm size), together with experimentally estimated constants. Notwithstanding these assumptions and estimates,

the main idea that the self-repair-time increases with larger swarms is well argued based on the experiments presented here. Even though the actual reliability for a given swarm size may be a somewhat higher or lower than the k-out-of-N model suggests, it is undoubtedly true that our case study swarm will eventually become non-functioning with increasing size, and that this occurs at a much lower swarm size than one might intuitively expect. Clearly we can significantly improve swarm reliability by increasing robot MTBF, as shown in Fig. 6. A four-fold increase in MTBF from 8 h to 24 h increases the swarm size with 0.9 reliability from 20 robots (Fig. 5) to 70 robots.

6 Concluding Discussion

The analysis of this paper raises two questions: firstly, to what extent can our conclusions be generalised, and secondly, what measures might be needed to improve the fault tolerance of swarm robotic systems. Addressing these questions in turn:

(1) We would argue two general conclusions from this work. Firstly, that our k-out-of-N approach to reliability modelling holds true for any swarm robotic system which depends on team work, i.e. the interaction of multiple robots giving rise to the desired overall swarm behaviour(s). Team work contrasts with parallel work in which any single robot can complete the task on its own, but multiple robots speed up task completion (subject to the constraint of interference between robots). Secondly, it follows that scaling to larger swarm sizes requires either more reliable individual robots, or active measures to improve fault tolerance (or both). Our analysis of what we call swarm self-repair and how it impacts swarm scaling and reliability is, of course, specific to the algorithm of our case study but, we contend, should apply to any swarm system in which swarm failure rate can overtake the swarm self-repair rate, with increasing swarm size. But even if that contention is wrong, invoking the Popperian criterion of scientific falsification, we only need to show that the assumption of swarm robustness and scalability is false once in order to cast its general validity into doubt.

(2) What active measures might be needed to improve fault tolerance and hence scalability? Since a swarm is a completely decentralised system we need to introduce new behaviours into individual robots that allow robots to be able to detect and respond to failures in co-workers. The problem breaks down into two parts: first, how can one robot reliably detect that another has failed, and second, what can it do about it. [7] provides a good example in which failed robots once detected can be physically grabbed and removed. But if the failed robot is only partially failed, as in failure Cases 2 and 3 in this paper, it may be that detecting that they have failed is very difficult, and isolating them from harmfully influencing the swarm within their locale, even

more so. We propose that an appropriate systematic approach to this problem is that of distributed artificial immune systems, and Timmis et al have begun initial work in this direction [13]. We believe that this is an important new direction in swarm robotics. Indeed we would argue that the conclusions of this paper might reflect a general truth about large-scale self-organising systems (including swarms of robots, swarm of insects, and assemblages of cells into multi-celled organisms), which is that such systems cannot function without an active approach to dealing with failed or rogue units, i.e. an immune response. What is perhaps surprising is that such an approach will be needed in swarm systems with relatively few individuals, i.e. less than a hundred.

References

1. Bjerknes, J.D.: Video of 10 e-puck swarm taxis trial with 2 simultaneous complete failures (2008), http://www.youtube.com/watch?v=zXrXCbag2iM
2. Bjerknes, J.D.: Video of 10 e-puck swarm taxis trial with 2 simultaneous partial failures (2008), http://www.youtube.com/watch?v=e2Zsga1pUIo
3. Bjerknes, J.D.: Video of 10 e-puck swarm taxis trial with no failures (2008), http://www.youtube.com/watch?v=DCyCaHePTd8
4. Bjerknes, J.D.: Scaling and fault tolerance in self-organised swarms of mobile robots. Ph.D. thesis, University of the West of England, Bristol (2010)
5. Bjerknes, J.D., Winfield, A.F.T., Melhuish, C.R.: An analysis of emergent taxis in a wireless connected swarm of mobile robots. In: Proc. IEEE Swarm Intelligence Symposium (SIS 2007), pp. 45–52 (2007)
6. Carlson, J., Murphy, R.R.: Reliability analysis of mobile robots. In: Proc. IEEE Int. Conf. on Robotics and Automation (ICRA 2003), pp. 274–281 (2003)
7. Christensen, A., O'Grady, R., Dorigo, M.: From fireflies to fault tolerant swarms of robots. IEEE Transactions on Evolutionary Computation 13(4), 754–766 (2009)
8. Elsayed, E.: Reliability Engineering. Addison Wesley Longman (1996)
9. Kuo, W., Zuo, M.J.: Optimal Reliability Modeling: Principles and Applications. Wiley (2002)
10. Marino, A., Parker, L.E., Antonelli, G., Caccavale, F., Chiaverini, S.: A modular and fault-tolerant approach to multi-robot perimeter patrol. In: Proceedings of IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 735–740 (2009)
11. Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.C., Floreano, D., Martinoli, A.: The e-puck, a Robot Designed for Education in Engineering. In: 9th Conference on Autonomous Robot Systems and Competitions, Portugal, vol. 1, pp. 59–65 (2009)
12. Nembrini, J.: Minimalist Coherent Swarming of Wireless Networked Autonomous Mobile Robots. PhD Thesis, University of the West of England, Bristol, UK (2005)
13. Timmis, J., Tyrrell, A., Mokhtar, M., Ismail, A., Owens, N., Bi., R.: An artificial immune system for robot organisms. In: Levi, P., Kernback, S. (eds.) Symbiotic Multi-Robot Organisms: Reliability, Adaptability and Evolution, pp. 268–288. Springer (2010)

14. Winfield, A., Liu, W., Nembrini, J., Martinoli, A.: Modelling a wireless connected swarm of mobile robots. Swarm Intelligence 2(2-4), 241–266 (2008)
15. Winfield, A.F.T., Harper, C.J., Nembrini, J.: Towards Dependable Swarms and a New Discipline of Swarm Engineering. In: Şahin, E., Spears, W.M. (eds.) Swarm Robotics 2004. LNCS, vol. 3342, pp. 126–142. Springer, Heidelberg (2005)
16. Winfield, A.F.T., Nembrini, J.: Safety in numbers: fault-tolerance in robot swarms. International Journal of Modelling, Identification and Control 1(1), 30–37 (2006)

Hierarchical Distributed Task Allocation for Multi-robot Exploration

John Hawley and Zack Butler

Abstract. In order to more effectively explore a large unknown area, multiple robots may be employed to work cooperatively. When properly done, the group allocates specific portions of the overall exploration task to different robots such that the entire environment is explored with minimal excess effort. In this work, we present a new hierarchical market-based approach to this allocation problem. Our approach builds on standard auction approaches to provide agents with a mechanism to independently form coalitions and to divide a coalition into smaller coalitions in response to the progress of their cooperative exploration process. These coalitions allow a subset of the team to move together efficiently, especially in constrained environments when there are few avenues for exploration. We also present implementation and simulated experiments which show how this natural hierarchy forms and can lead to more efficient exploration than using a greedy allocation technique or without the use of coalitions.

1 Introduction

Exploration of unmapped terrain is a task well-studied in robotics, and is well suited to multi-robot systems. Teams of robots can fan out and visit locations in parallel to make the overall discovery process more efficient, and a variety of approaches have been proposed to coordinate this process. One common approach to coordinating multiple robots is through the use of market-based schemes for task allocation [5, 6, 13, 16]. When a new task is given to the team (or discovered by a team member, in the case of exploration), the robots bid on the right to take on that task. Bids are computed based on the difficulty of the robot to accomplish the task, creating an essentially greedy assignment of tasks. In the case of exploration, tasks will generally take the form of a location or region to be visited.

John Hawley · Zack Butler
Computer Science Dept., Rochester Institute of Technology, Rochester, NY 14623, USA
e-mail: jph1996@gmail.com, zjb@cs.rit.edu

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 445–458.
springerlink.com

Depending on the particular type of mission, there may be a surplus of tasks or a surplus of robots (or both at different times during the mission). Exploration of open terrain will generally have a surplus of tasks, but in indoor environments there may be few tasks, such as one for each hallway currently being explored. Most market-based systems for task allocation are designed for the former case (a surplus of tasks), and simply try to assign each task or set of tasks to the best available robot. When there is a surplus of robots, those without a task to accomplish simply remain idle. In the context of exploration, this may not be the best choice, as new tasks will be generated on the frontier of explored area. For example, when one robot is exploring a hallway and discovers a four-way intersection, it will generate three new tasks and we would like to have three robots close at hand if possible to make the process more efficient.

To address these issues, we propose a method through which robots can autonomously form coalitions during the exploration process via a market-based mechanism. That is, each robot decides for itself whether it is more profitable to take on a task for itself or join up with a group that already has one or more tasks. The coalition formation and dissolution is performed in addition to a standard market-based task allocation technique to handle the assignment of the exploration tasks.

1.1 Related Work

As mentioned, there have been many different approaches to multi-robot exploration. In general, they consist of determining locations to visit and assigning those locations to robots. For the locations, a common approach is to use frontiers [1, 7, 11, 15], identified as contiguous groups of map cells that represent explored open space adjacent to unexplored space[1]. A goal point is created for each such group, and is located at the arithmetic mean of all points in the group. Other approaches to goal identification that have been implemented include random point selection, greedy exploration, map segmentation [14] and quad-tree subdivision [16]. Once determined, a variety of approaches exist to assign these to robots. Some rudimentary but successful approaches simply direct an agent towards the nearest goal point [15]. In the case where multiple agents are participating in the exploration, however, this task becomes far more complex. In such cases, more advanced techniques are often employed, including greedy mechanisms [1, 11], optimal centralized approaches [14], genetic algorithms [7], Voronoi-based approaches that can implicitly keep robots in different areas [3] and market-based mechanisms [6, 10, 16].

Among these techniques, market-based allocation strategies are quite popular. In these strategies, agents negotiate with each other and treat goal points as a commodity that they exchange. Such exchanges are determined by auction mechanisms, though the specific auction mechanism used varies between implementations. In some implementations, single-round single-item sealed-bid auctions that closely

[1] Our implementation uses frontiers to generate goal points, but the method is intended to function equivalently for other methods of goal point generation.

resemble greedy allocation strategies are preferred [16]. In other more distinctly market-based implementations, multi-round auctions may be used so that bids can account for the effects of previous allocations [4, 6], particularly in the calculation of goal point utility. Other issues addressed include handling constraints of communication across a dispersed robot team within the context of bidding [8, 10], which is important from a practical standpoint but is not considered in this work. Occasionally, combinatorial bids are used, in which agents bid on multiple goals in batches rather than individually [5], as it can be advantageous for an agent to pursue groups of nearby goals rather than treat each goal independently. Our work is similar in that an auction mechanism is used to assign goals to robots. However, we use a second auction process in parallel that assigns robots to coalitions. In this way, the coalitions will form naturally and in a purely decentralized way depending on whether the robot finds it more advantageous to pursue its own goal or join a nearby team. The work in [2] also involves groups of robots in a market-based allocation, but in that case leader robots can reassign tasks among ad-hoc groups for a more optimal assignment, whereas we are considering longer-term coalitions with a common goal.

Coalition formation has been addressed in different contexts as well. Often, these works consider tasks which can or must be completed by a team of agents instead of a single agent. A foundational work in this area is that of Shehory and Kraus [9], which includes distributed algorithms for coalition formation with provable bounds on task completion efficiency. In a more closely related context, the AsyMTRe-D algorithm [12] allows robots to create small coalitions based on their capabilities to solve complex tasks. As such, it makes decisions on a discrete basis to form necessary groupings rather than the real-valued numeric bidding used here to form opportunistic groups.

2 Hierarchical Exploration

Our exploration technique includes both goal assignment and formation as well as maintenance and dissolution of coalitions through different auction mechanisms. The first type of auction is a *Goal Auction*, in which agents offer and bid on goals, similar to existing mechanisms for task allocation. The second type is an *Agent Auction*, in which an agent auctions its services in the event that it does not have its own goals to pursue, potentially forming a coalition with other agent(s). These two auction mechanisms take place asynchronously, but care must be taken so that an agent does not transfer a goal in a *Goal Auction* that it has used to make a bid in an *Agent Auction*.

2.1 Goal Auctions

In order for an exploration strategy to be truly distributed, there must be a sharing of responsibility for goals among agents. The agent initially responsible for a goal

is trivially the agent that discovered the open space to which the frontier is adjacent. During the course of exploration, however, the agent that generated a goal point may become no longer the most optimal agent for exploring that goal point. In this work, we use a market architecture as a simple and effective way for agents to transfer responsibility for goal points. As agents traverse the environment, they build a local map and periodically share that information with the other agents in the team. When an agent generates new goal points, usually by reaching a current goal point, it can hold an auction so that goal points may be transferred to more optimal agents. Depending on the structure of the environment, a frontier may be discovered or enlarged on the way to another goal, and will be put up for auction if so. An agent may also periodically hold auctions even when no new goals are discovered so that goals can still be transferred between agents during long travels through explored regions. While many complex auction strategies exist, here we use simple single-round highest-bidder closed auctions.

In order for agents to appropriately bid on goals, measures of *cost* and *utility* of goals are required, as in much previous work in market-based allocation. Here, cost is calculated as the distance an agent must travel to reach a goal point. Since the environment is only partially known, the cost is optimistically calculated by treating unexplored space as open space within a standard A* search. Utility is defined in a way dependent on the type of goal used, but generally describes the expected increase in explored area from visiting that goal. The value of a goal is then calculated as $value = utility - \beta \cdot cost$ where β represents a coefficient representing the relative values of *cost* and *utility*. To compute utility when frontiers are used as goal locations, we estimate how much unexplored space would be revealed by that agent (based on its sensing radius) were it to be at that particular goal point, resulting in the *expected information gain* [11] of that goal point. We note however that the general form of the hierarchical task allocation that we present does not rely on any particular definition of utility.

2.2 Coalitions

In order to hierarchically distribute goals to agents, agents can form coalitions. Here, we define a *coalition* as a set of agents simultaneously and intentionally moving to explore the same goal point. A coalition is comprised of exactly one *supervisor* and zero or more *workers*. While an agent is deciding what to do next, it is in a third state, *retasking*. An agent is always in exactly one of these three states. See Fig. 1 for a representation of the transitions between these states.

The *supervisor* of a coalition is the agent responsible for that coalition's goal. When an agent does not have any goals that it is responsible for, it can obtain a goal from another agent. In some cases, this entails joining another agent in a coalition. Coalitions necessarily form when there are more agents than available goals, which is often the case in highly structured environments. When a coalition has been formed to pursue a goal and the resulting exploration of that goal reveals two or more new goals, that coalition will divide into smaller coalitions so that the newly

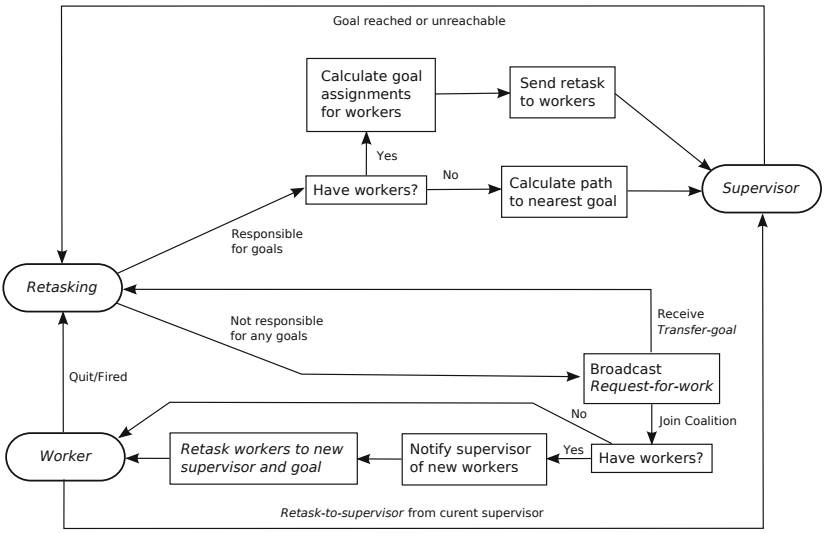


Fig. 1 State diagram describing the transitions between *supervisor*, *worker*, and *retasking* states

generated goals can be effectively explored. In this way, coalitions hierarchically divide and allocate tasks accordingly.

Workers are agents who have joined the supervisor because they do not have any goals of their own to pursue. Each other agent is therefore trivially the supervisor of a coalition of size one – the coalition containing only that agent. Agents that are workers do not have any goals for which they are responsible and therefore do not hold auctions to transfer goals to other agents and cannot be supervisors. In addition, an agent will belong to exactly one coalition at any time. The supervisor of a coalition is responsible for notifying the workers of that coalition of any changes to the current goal of the coalition.

2.3 Coalition Formation

The mechanism used to form coalitions is similar to the market used to allocate goals to agents. However, instead of holding a goal auction, an agent holds an agent auction, which allows it to discover the most profitable goal for it to pursue. The agent auction is initiated by broadcasting a *Request-for-work* message. See Fig. 2 for more details regarding the interaction between an auctioneer and bidders that takes place in response to a *Request-for-work* message.

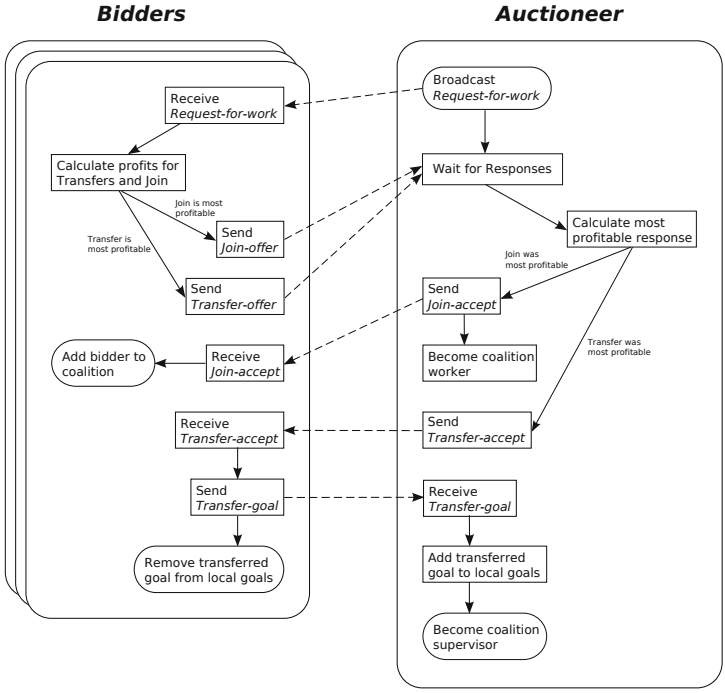


Fig. 2 *Request-for-work* mechanism, initiated by an agent that does not have any of its own goals to pursue

In order to compare joining a coalition to alternative courses of action, an expected profit must be calculated for a potential coalition. The profit of a coalition can be calculated by

$$Profit = \frac{\max_{i \in A}(Utility(i,g)) - \beta \cdot \sum_{i \in A} Cost(i,g)}{|A|}$$

where A is the set of agents belonging to the coalition and g is the coalition's goal. In the case where this results in a negative value for *Profit*, the formula

$$Profit = \left(\max_{i \in A}(Utility(i,g)) - \beta \cdot \sum_{i \in A} Cost(i,g) \right) \cdot |A|$$

must be used so that larger coalition sizes are penalized (i.e. given more negative profit values) rather than rewarded. Since there is no way to know how many avenues of exploration a goal will produce, it is assumed that smaller coalitions provide a more even distribution of workers among available goal points and are therefore

desirable. This is relevant as an agent may find it best to join some coalition if it has no goal of its own, and so it is possible the smallest negative profit will be chosen.

2.4 Coalition Maintenance

In many cases, particularly in highly structured environments such as hallways, exploration of a goal point produces only a single new goal point. In these cases, it is sensible for a coalition to continue on to the new goal. This is accomplished by the supervisor sending a retask message to coalition workers informing them of the new goal to pursue. A worker may leave a coalition at any time, but this retasking provides a particularly opportune time for workers to consider alternative courses of action based on the utility of the new goal.

The case in which a coalition explores a goal that results in multiple new goals requires particular attention. If a worker discovers a new goal point, that worker will quit the coalition and become its own supervisor. It will then respond to future *Request-for-work* broadcasts to obtain its own workers. If the supervisor of a coalition discovers multiple goal points, it is the supervisor's responsibility to decide which workers will pursue which goals. This can be done in different ways, but in our implementation, we have chosen a greedy approach, as follows. First, agents are ordered by topological distance to the supervisor, including the supervisor, which trivially has a distance of zero. Each agent is then assigned to the most profitable goal for that agent, beginning with the supervisor. The first agent assigned to a goal will become a supervisor responsible for that goal, and any further agents assigned to the same goal will be transferred to the new supervisor as workers. Once an agent has become its own supervisor, it is no longer affiliated with the agent that was previously its supervisor.

In order to accommodate all these interactions, three types of retask messages are required. A *Retask-simple* message simply instructs a worker to calculate a path to and pursue a new goal point. A *Retask-become-supervisor* message instructs a worker to become a supervisor that is responsible for the included goal point. Implicitly, the new supervisor is to calculate a path to and pursue the new goal. A *Retask-change-supervisor* message instructs an agent to join a new supervisor's coalition. Upon doing so, the worker will be given a new goal point to pursue.

2.5 Coalition Dissolution

Coalitions may be dissolved for a number of reasons. A worker may choose to quit its current coalition and reevaluate a new task at any time. This is particularly useful when the worker is far away from the coalition's goal, since the state of the exploration changes over time, and it is possible, if not likely, that a better alternative will arise for the worker. A worker may also receive its own goal, either by revealing newly explored open territory, or by bidding in goal auctions (workers do not hold

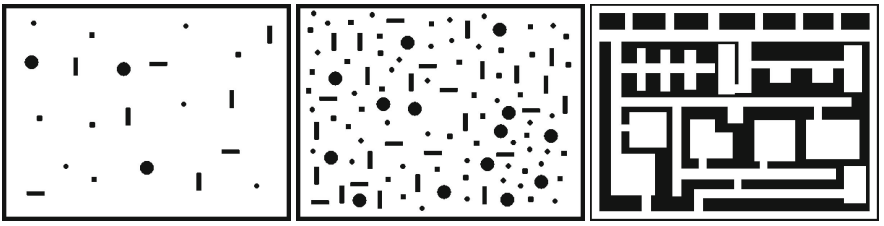


Fig. 3 Three maps used for experiments: sparse, dense, structured

goal auctions because they have no goals of their own to auction, but they still bid in goal auctions held by other agents).

It is also possible for the supervisor to dissolve a coalition. If the exploration of the coalition goal results in no new goals, the supervisor will make use of the *Request-for-work* mechanism (see Fig. 2) to obtain a task. If this results in the supervisor joining another coalition, it will become a worker itself and will transfer the workers of the old coalition to its new supervisor by sending them *Retask-change-supervisor* messages. It is also necessary to notify the new supervisor of the addition and the workers of the change in supervisor, by sending it a *Transfer-workers* message.

3 Experiments

To evaluate the utility of the coalition-based algorithm, experiments were performed on a variety of maps with different numbers of participating agents. A simulator was written in Java that communicates with clients over TCP/IP. The simulator notifies clients of newly explored area and provides messaging between clients in both point-to-point and broadcast manners. Robots are assumed to have accurate localization. Communication between clients and the server is asynchronous. Clients are not provided with any means of contacting other clients directly.

In order to make comparisons in a proof-of-concept sense, we tested our algorithm against two other basic techniques. One of these techniques is simply using our algorithm without the agent auctions; this will mimic traditional auction-based task allocation. In this case, any agent without a goal assigned will simply be idle and remain at its present location. The other point of comparison is a greedy algorithm in which each agent is responsible for the goals to which it is closer than any other agent. An agent pursues whatever goal it has that is the closest topologically. If an agent is not responsible for any goals, it will broadcast a request to other agents and will pursue the goal it receives that is the closest topologically. Note that this mechanism does not transfer responsibility for the goal, it merely provides the agent with an interim goal to pursue until it is responsible for its own goal rather than remaining stationary. This algorithm should eliminate some inefficiency due to idling but without using explicit coalitions.

Experiments were performed on a set of four maps, the three shown in Fig. 3 as well as an open map with no obstacles. Each map is represented as an 800 by 600 pixel bitmap, and agents have a radius of vision of 40 pixels. The first map is devoid of obstacles, except for a boundary preventing agents from reaching the edge of the map. The second and third maps contain increasingly many variously sized, shaped, and positioned obstacles. The fourth map is a highly structured office-building-like map specifically designed to test the performance of exploration methods in an inherently hierarchical environment. Team sizes of 2, 4, 8, 16 and 32 homogeneous agents were tested, and in all cases, all agents started in the center of the map for all tests, simulating a standard group deployment.

3.1 Results/Discussion

As expected, results varied between map types. In general, the greatest benefit of coalitions was seen on the structured map, while hierarchical allocation methods consistently outperformed the no-coalition algorithm and did not perform noticeably worse than the greedy control algorithm in any of the tests.

Perhaps the most intuitive measure of the performance of an exploration algorithm is the amount of area explored versus time. Since the simulator calculates its

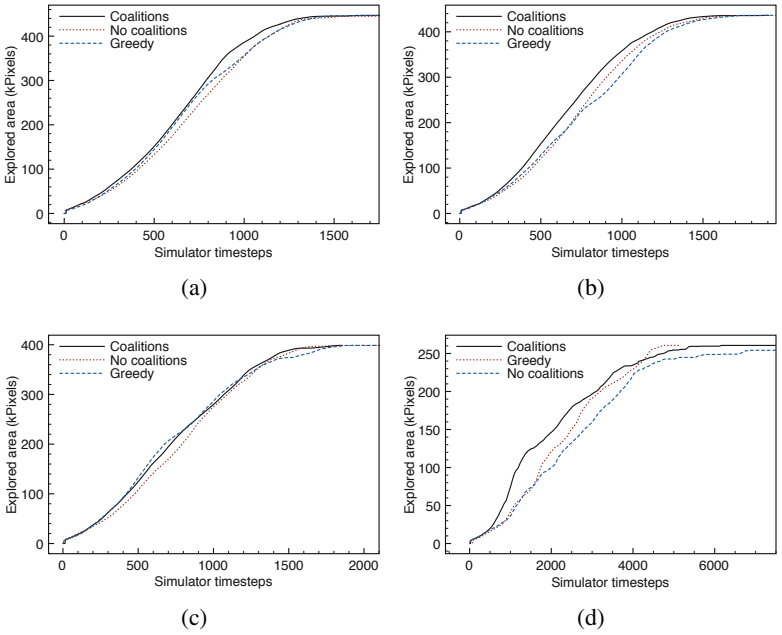


Fig. 4 Area Explored Versus Time for 16 agents exploring the (a) open, (b) sparse, (c) dense and (d) structured environments

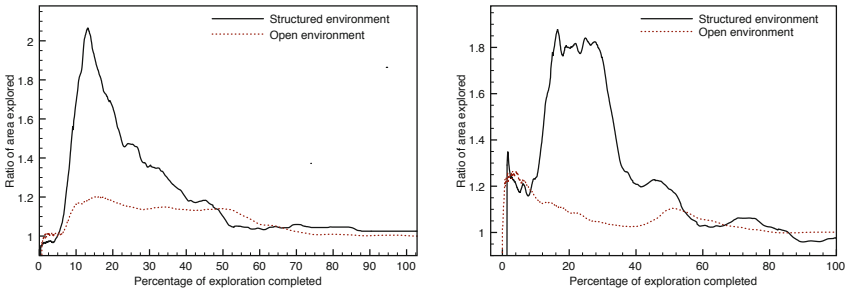


Fig. 5 Ratio of Area Explored by 16 Agents in the open and structured environments — (a) Hierarchical algorithm vs no-coalition algorithm; (b) Hierarchical algorithm vs greedy algorithm

state in time increments, henceforth referred to as ticks, the units of time used in the graph are arbitrary and correspond to simulator ticks rather than any wall time unit. Area explored is simply measured in pixels, since the exploration environment maps are loaded as bitmaps, providing pixels as a convenient measure of area. In the simpler, more regular environments, there was little difference between the different algorithms, whereas in the structured map, the hierarchical approach was able to explore more quickly than either of the comparison approaches. Plots for the 16 robot case are shown in Fig. 4.

We can also look at the relative progress of the different algorithms with respect to time across different map types. In this case, area explored and exploration time must be expressed as percentages, since the maps vary in amount of free space and thus time required for exploration. In particular, we compute the ratio A_h/A_{nc} where A_h and A_{nc} are the area explored by the hierarchical and no-coalition methods respectively at a given time. The value of this ratio over time is shown in Fig. 5a for both the open and structured environments. From this graph, it can be seen that the hierarchical allocation method performed better in the very early stages of exploration in both environments, but that it performed much better throughout in the structured environment. This is as expected, because the hierarchical method allocates agents more effectively when there are more agents than goals. Even the greedy algorithm, which allocates the extra agents by assigning them to their respective nearest goals instead of idling them, suffers in comparison during this initial phase in both environments, though it does catch up effectively by the end. This comparison is shown in Fig. 5b.

Other team sizes showed similar trends with the benefit of coalitions generally larger as the team size increases, as expected. However, these environments do suffer from diminishing returns. Table 1 shows the time required to explore 80% of the free

Table 1 Time (in simulator ticks) required to explore 80% of the free space in different environments with different team sizes and algorithms

	8 robots			16 robots		
	No coalition	Greedy	Coalitions	No coalition	Greedy	Coalitions
Open environment	1393	1442	1478	972	943	934
Structured environment	4006	3929	3887	3753	3370	2827

space of the environment for 8 and 16 robots using the three different algorithms.[2] The more open environment showed greater improvement when going to the larger team, while the structured environment showed less improvement (presumably since there are fewer avenues of exploration) but more effect of coalitions, especially when the larger team is employed.

In addition to the time required for exploration, we also considered the total distance traveled by the team. In general, since both the greedy approach and the coalition-based approach do not allow robots to idle, we expect these methods to produce more total travel even when the exploration time is less. As can be seen in Fig. 6, this is borne out in the experiments. Distance traveled for these techniques goes up almost perfectly linearly with time. When using coalitions, especially in large teams, some robots do pause while determining their next course of action, but this does not have a major effect on total distance traveled. Without coalitions, the team initially has lower distance traveled since several robots will be idle at the outset. During the bulk of the exploration, some robots may remain idle in the larger teams, but not in the smaller teams.

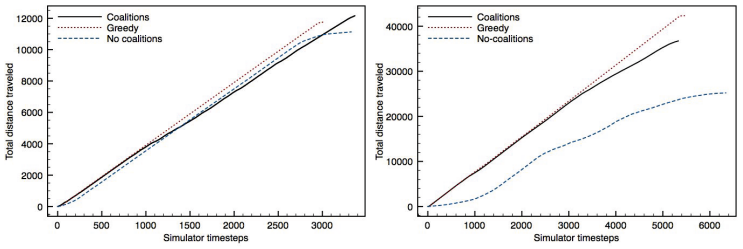


Fig. 6 Total distance traveled by the robot team over time. (Left) 8 robots in the open environment (Right) 16 robots in the structured environment.

Qualitative Observations: Qualitative observations do not provide concrete support for the validity of hierarchical task allocation, but they can help explain the quantitative observations made and provide some insight into future ideas worth pursuing. In particular, we can identify different stages of exploration under which the hierarchical allocation method performs more or less effectively.

[2] We use the time to explore 80% instead of 100% since the last portion of exploration, though very important, is highly dependent on the locations of the robots near the end of exploration and is not as directly comparable across algorithms.

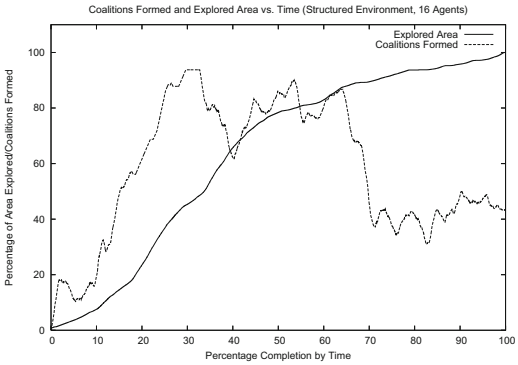


Fig. 7 Percentage of Area Explored and Coalitions Formed Versus Time for 16 agents exploring the dense environment

As mentioned earlier, our motivation for using coalitions is largely to handle the situation when agents outnumber goals. In many exploration processes, there will eventually come a point at which there are more goals to be explored than there are agents. Once agents are no longer forming coalitions, the hierarchical allocation method essentially becomes a standard market-based allocation algorithm. In some environments, however, this may never occur. For example, in the structured environment and with 16 agents exploring, there was always at least one coalition of more than one agent. This can be seen in Fig. 7. In this figure, the area explored and number of coalitions formed are represented as percentages. Coalitions Formed is calculated as the number of coalitions that exist at any given time divided by the number of agents, since the number of agents is the maximum number of coalitions that may form. This figure also demonstrates the decrease in rate of area being explored as the number of coalitions drops significantly around 70% of the way through the exploration. Even in those instances when there are more goals than agents, there will eventually be fewer goals than agents again near the end of the exploration. The hierarchical allocation method does not appear to perform particularly well once there are more goals than agents, even after the number of goals decreases back below the number of agents. It is not immediately clear why this is, but observations of the experiments indicate that building the coalitions from physically dispersed robots seems to be unhelpful when there is little area left to explore.

4 Conclusions / Future Work

Overall, these results indicate that hierarchical coalition-forming task allocation techniques for robotic exploration can perform better than greedy or coalition-free approaches. This is particularly the case in very dense or structured environments,

but even in open space, hierarchical task allocation performed no worse than simple traditional auctioning or greedy allocation. The improvements apply to teams of various sizes, depending somewhat on the environment. This improvement in time does come at the cost of greater energy expenditure in terms of total distance traveled by the team.

We have also considered several potential enhancements to the basic distributed coalition formation technique. For example, existing allocation methods for robot teams often use combinatorial auctions in which several nearby goals can be bid on and assigned as a group, exploiting their collocation. With coalitions, a group of nearby goals may be bid on by a coalition, such that the size of the goal set is equal (or close) to the size of the coalition. More generally, the utility of a coalition to achieve a goal (or set of goals) may be dependent on the capabilities of the members of the coalition, especially if the system is heterogeneous. This could allow for such systems to effectively use their varied abilities in similar fashion to other task allocation strategies in addition to the advantages of the coalitions.

Finally, even though the environment is assumed to be unknown, the robots could use their experiences from the initial exploration to inform future decisions. For example, it seems from our experiments that the number of coalitions follows a pattern of increase and decrease throughout exploration. Being able to detect this on the fly may allow us to idle robots toward the end of the exploration process to conserve energy with minimal loss of exploration efficiency. Also, while it is impossible to know for sure which goals will branch into multiple new goals to explore, it may be possible to employ pattern matching techniques to predict a likelihood that a goal branches. The ability to predict branching with any accuracy could be conveniently incorporated into a coalition forming exploration strategy. The coalition profit calculation could be easily modified to account for the optimal coalition size for a goal, based on estimated branching. In heterogeneous systems, even the membership of coalitions could be informed by the expectation of the needs of the exploration process. Together, we hope to show in the future that these improvements can lead to a cooperative exploration system that is even more efficient and effective.

References

1. Burgard, W., Moors, M., Fox, D., Simmons, R., Thrun, S.: Collaborative multi-robot exploration. In: Proceedings of ICRA, pp. 476–481 (2000)
2. Dias, M.B., Stentz, A.: Opportunistic optimization for market-based multirobot control. In: Proceedings of IROS, pp. 2714–2719 (2002)
3. Fu, J.G.M., Bandyopadhyay, T., Ang Jr., M.: Local voronoi decomposition for multi-agent task allocation. In: Proceedings of ICRA, pp. 1935–1940 (2009)
4. Lagoudakis, M., Berhault, M., Koenig, S., Keskinocak, P., Kleywegt, A.: Simple auctions with performance guarantees for multi-robot task allocation. In: Proceedings of IROS, pp. 698–705 (2004)
5. Lin, L., Zheng, Z.: Combinatorial bids based multi-robot task allocation method. In: Proceedings of ICRA, April 18-22, pp. 1145–1150 (2005)
6. Ma, X., Meng, F., Li, Y., Chen, W., Xi, Y.: Multi-agent-based auctions for multi-robot exploration. In: The Sixth World Congress on Intelligent Control and Automation, pp.

9262–9266 (2006)
7. Ma, X., Zhang, Q., Li, Y.: Genetic algorithm-based multi-robot cooperative exploration. In: International Conference on Control and Automation, pp. 1018–1023 (2007)
8. Pei, Y., Mutka, M., Xi, N.: Coordinated multi-robot real-time exploration with connectivity and bandwidth awareness. In: Proceedings of ICRA, pp. 5460–5465 (2010)
9. Shehory, O., Kraus, S.: Methods for task allocation via agent coalition formation. Artificial Intelligence 101(1-2), 165–200 (1998)
10. Sheng, W., Yang, Q., Ci, S., Xi, N.: Multi-robot area exploration with limited-range communications. In: Proceedings of IROS, pp. 1414–1419 (2004)
11. Simmons, R., Apfelbaum, D., Burgard, W., Fox, D., Thrun, S., Younes, H.: Coordination for multi-robot exploration and mapping. In: Proceedings of the National Conference on Artificial Intelligence, AAAI 2000 (2000)
12. Tang, F., Parker, L.: Distributed multi-robot coalitions through ASyMTRe-D. In: Proceedings of IROS, pp. 2606–2613 (2005)
13. Walsh, W., Wellman, M.: A market protocol for decentralized task allocation. In: Proceedings of International Conference on Multi Agent Systems, July 3-7, pp. 325–332 (1998)
14. Wurm, K.M., Stachniss, C., Burgard, W.: Coordinated multi-robot exploration using a segmentation of the environment. In: Proceedings of IROS, pp. 1160–1165 (2008)
15. Yamauchi, B.: A frontier-based approach for autonomous exploration. In: Proceedings of Computational Intelligence in Robotics and Automation (CIRA 1997), pp. 146–151 (1997)
16. Zlot, R., Stentz, A., Dias, M.B., Thayer, S.: Multi-robot exploration controlled by a market economy. In: Proceedings of ICRA, pp. 3016–3023 (2002)

Endocrine Control for Task Distribution among Heterogeneous Robots

Joanne H. Walker and Myra S. Wilson

Abstract. This paper details an endocrine based system which automatically reassigns tasks among heterogeneous robots dependent on the ability of the robot to do the task. This ability (or sensitivity) to a task is initialised for each individual robot after an evolutionary training stage, then constantly adapts as the robots perform the various tasks. The system does not require a centralised controller, and relies on little communication between the robots.

1 Introduction

A key aim of robotics research is to design robots that can perform autonomously in the real world; a world that is complex and dynamic. An important challenge for researchers is therefore to produce robots that can continually adapt to their surroundings and tasks, so that they can improve their performance when the environment is stable, and adapt when the environment changes. As with all systems, failure is possible and so adaptation methods should include recognition and appropriate handling of failure.

This paper describes a multiple, heterogeneous robot system that is able to operate without external assistance for an extended period of time. This highlights the importance of adaptability, usefulness in diverse environments, and the advantage of not having to perform complex reprogramming. The system identifies when a robot has degraded and is unable to complete a task. The task can then be reallocated to another robot with a similar competence. Where the environment has changed and the robot is no longer appropriate for the task, this is recognised and the task reassigned to another robot and the original robot assigned elsewhere.

Joanne H. Walker · Myra S. Wilson
Department of Computer Science, Aberystwyth University,
Penglais, Aberystwyth, SY23 3DB, UK
e-mail: mxw@aber.ac.uk

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 459–472.
springerlink.com

This paper reports a method for autonomous task assignment within a group of heterogeneous robots. The method should assign tasks based upon the different robots' abilities and continuously adapt as the robots carry out their task.

The work described here builds on previous developments which have:

- Used evolutionary methods to continuously adapt a single robot to a dynamic environment as it carried out a task [11].
- Demonstrated an endocrine based system which assigns tasks between a group of homogenous robots in simulation [13].
- Demonstrated that the endocrine based system successfully re-assigns tasks between simulated robots when a robot breaks down [14].

The work described here expands upon previous by:

- Extending the system to a group of heterogeneous robots, distinguished by their sensor complements.

In formulating the design, inspiration from biological systems was considered because they represent extant methods which are proven to co-ordinate a range of functions. More specifically, the human endocrine system was chosen as it co-ordinates the execution of various tasks within the body by the sending of signals which are received by cells, changing their behaviour. Within a group, individual robots can be seen as representing parts of an overall physiology, or body, and radio broadcasts as chemical signals between them. It was anticipated that by the release and decay of hormone signals and in sharing these signals between robots, task assignment, and re-assignment upon change would be emergent properties of the system.

2 Background

In previous work, evolutionary methods have been used to train a robot in simulation, followed by lifelong adaptation in the real world [10]. In this method, a genetic algorithm (GA) was used to evolve a Khepera robot controller in simulation (based on a training scheme reported in [8]). Training was followed by lifelong adaptation using an evolution strategy (ES) which was designed to allow the robot to continue to adapt to a dynamic environment for the lifetime of the robot. The work reported here was motivated by a desire to expand the method to groups of heterogeneous robots, which could autonomously assign tasks, responding to changing performance.

Endocrine systems for robot control have rarely been investigated for co-ordination of behaviors on a single robot. Brooks [3] incorporated a hormone control model into the subsumption architecture to switch between behavior sets. In Arkin's Schema Architecture [2], a 'homeostatic' control mechanism was added in order to maintain the internal environment of the robot [1]. Internal sensors such as a fuel gauge and thermistor were monitored and their

output used to alter the robots' behavior. A hormone-inspired system has been used in the design of adaptive communication and control for modular self-reconfigurable robots [9]. The authors suggest endocrine-inspired systems are ideal for distributed control of multi-robot systems.

In [6] an endocrine model was used to control 'emotive' behavior in a robot. The robot's task was to wander safely whilst hormone was released in proportion to the proximity of perceived obstacles. The input weights of an artificial neural network (ANN) increased in response to the hormone levels, which increased the speed at which the robot moved away from obstacles.

The work reported in [5] is particularly relevant as it has been used as a starting point for the multi-robot system work reported here. In [5] hormone signals were used to coordinate target seeking and wall avoidance behaviors on a single robot. The hormones were released in response to the proximity of targets and walls into a pool, once the pool's threshold was reached the hormone was released to become 'free hormone' whereupon it would interact with an ANN. The free hormone then decayed exponentially over time. Each behavior was active to a level dependent on the amount of free hormone for that behavior. In this way, when the robot was close to a target it would tend to move towards it, but when near a wall it would move away. The system reported here is made distinct from that in [5], by expanding to a group of robots; it also does not use an ANN, and the behaviours switch completely rather than being partially active according to hormone level.

Hormone-inspired control also bears resemblance to Maes' Action Selection Architecture [4] in which levels of activation control which behaviors were active at a given time. Activation was stimulated by the environment, by the overall system goals and from other behaviors within the system; activation could also be suppressed. The system reported here distributes tasks based on the robots' performances alone.

Previous work in evolution for groups of non-cooperating robots has focussed on the robots sharing genetic information in order to improve the group's performance. In [15] a group of mobile robots became the population on which a GA was implemented. Each robot's behavior was defined by a single chromosome, and when two robots met they could "mate" by probabilistically replacing their own chromosome with a mutated version of the other robot's chromosome. Unlike some GA implementations, this can be practically implemented on physical robots, and it is especially appropriate for multi-agent tasks which will naturally bring the robots into contact with each other.

In terms of emergent task assignment for groups of robots the ALLIANCE architecture [7] is especially relevant. ALLIANCE switches between behavior sets according to 'impatience' and 'aquiesence' signals. In the work reported here, hormone signals are analogous to the impatience and aquiesence signals, but different mechanisms are used. However, unlike in Alliance, our hormone-inspired system does not assume the robots can autonomously monitor the performance of other group members – a very difficult task with current

sensor capabilities – instead they only monitor their own performance, and the active behaviors of other robots. This is then enough to cause a single robot to change its task, which results in behavior reallocation across the group. In addition, the hormone inspired system has been designed to be computationally less complex.

3 Robot Training Phase

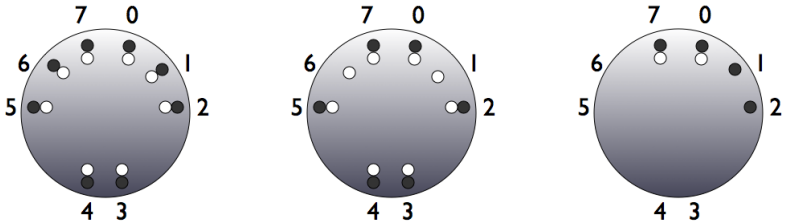
3.1 Evolutionary Training Methodology

Three simulated ePuck robots, each with a different sensor complement, were trained in a set of three possible tasks using a GA in the Webots[1] simulator. The aim of the training phase was to give the individual robots competence at each task before they went on to perform the tasks post-training.

The Webots world was $1m^2$ with walls around the edge, walls jutting into the centre and a number of cuboid obstacles. A Khepera robot carrying a light, and controlled by a simple obstacle avoidance algorithm, began in the middle of the arena and wandered randomly. This light bearing robot formed the goal for a light-seeking task.

At the beginning of each trial, the ePuck robots were placed back to their original corner positions, the Khepera back to the centre, and the cuboid obstacles were randomly placed, each having a 50% chance of falling inside the arena. In this way the robots faced a different environment for each trial, reducing the likelihood of over-fitting to a particular world configuration.

The sensor complements of the three robots were designed so that the three robots would have differing abilities to perform the three tasks, with each having the potential to work well at one particular task (Figure 1). Each of the robots was given a colour to make them easily identifiable.



(a) Blue Robot (b) Red Robot (c) Green Robot

Fig. 1 The sensor locations on each of the three robots, where a filled circle indicates a reflected IR (or proximity) sensor, and an open circle indicates an ambient IR sensor

[1] From Cyberbotics, `http://www.cyberbotics.com/`

3.1.1 Robot Behaviours

Three tasks were evolved: find-light, wall-follow, patrol-dark. Each task used a number of individual behaviours working together based on Arkin's Schema Architecture [2], whereby individual behaviours, or 'schema', are combined to produce an emergent behaviour.

Find Light Task. The aim of this task is to find, and remain close to, a light source whilst avoiding obstacles; it uses three schema: Find-light, Avoid-obstacle and Noise.

The Find-light schema simply turns the robot in the direction of the brightest light. One schema is instantiated for the sensor which has the brightest ambient light reading, and the direction the wheels spin in (forwards or backwards) is determined by that sensor's position.

One Avoid-obstacle schema is instantiated for each sensor which detects an obstacle. The schema use information about the distance to an obstacle (measured as reflected IR) to determine the speed of the wheels, and the direction of the wheels is determined by the sensor position.

The Noise schema gives a random direction which remains the same for a given amount of time.

Wall Follow Task. The aim of the wall-follow task is for the robot to find a surface on its right hand side, and follow it; it uses one schema.

The Follow schema looks for a surface and then follows it on its right side.

Patrol Dark Task. The aim of this task was to wander, whilst avoiding obstacles and areas of brightness; it uses three schema.

Find-dark schema where the robot will find the direction with the brightest ambient light, and move away from it.

The Avoid-obstacle schema and Noise schema are identical to those for Find Light.

3.1.2 Genetic Algorithm for Training

Evolution was used to train the robots in the tasks. The methods used here have been described previously (and explained fully) in [11]. This section briefly explains the details of the methods, required as a context for the rest of the paper.

A genetic algorithm (GA) was used during the training phase. The GA had a population size of 30, reproduction rate of 0.6, crossover rate of 0.6 and mutation rate of 0.05. The selection method was the roulette wheel. The fitness (or cost) function for Find-light was designed to maximize going forwards in a straight line towards brightly lit are as of the arena (the goal) and away from darker areas and obstacles. Cost for Follow-wall rewards moving quickly and in a straight line, and away from obstacles; it also rewards keeping close

to a surface on the left of the robot. Patrol-dark rewards moving quickly, in a straight line, away from obstacles, and in darker areas of the environment.

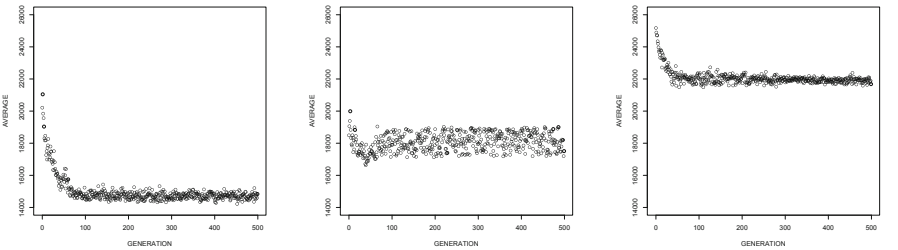
In each of the three tasks, the fitness was calculated for each time step, and accumulated over a run; each chromosome had three runs and the final score was the sum of the three accumulated values. A run lasted 3000 timesteps. Low 'fitness' was good because here fitness is analogous to the inverse of the cost, so to avoid confusion it will be referred to as cost for the remainder of the paper.

3.2 Task Training Results

The robots' performance over 500 generations of training for each of the three tasks can be seen in Figures 2, 3 and 4.

Find Light Evolution Results. In adapting for the Find Light task (Figure 2), the Blue robot most quickly reached the best measured levels of performance. The Red robot, with its reduced number of IR sensors, had a significantly more varied performance. The Green robot, with its much reduced ambient light and proximity sensors did not reach the same level of performance as the other two (as might be expected), although it did perform more consistently. Of course, this apparently more consistent performance could be because its reduced number of sensors did not record as much of its environment, and so it was unable to detect as many obstacles in its path.

The evolved controllers (i.e. the best from the 499th generation) were also tested in simulation and compared to the best controllers from the zeroth generation. Each chromosome was run 20 times under the same conditions as during training (the same simulated world, and 3 trials for each chromosome). There were big improvements in performance during training, and in testing for significant difference between performance in generations zero and 499 (Kruskal Wallis at 95%), for each robot, there was a statistically significant difference in each case.

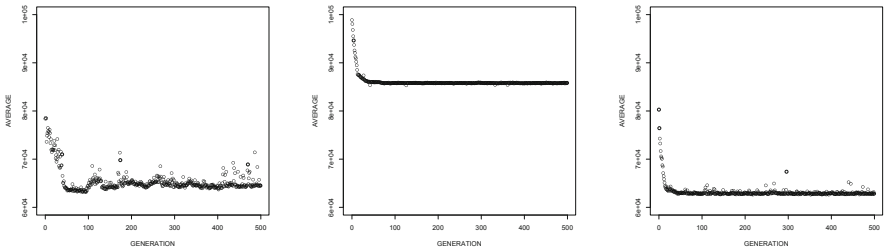


(a) Blue Robot (b) Red Robot (c) Green Robot

Fig. 2 Performance at the Find Light task over 500 generations of the three robots as their behaviours evolved

Wall Following Evolution Results. The results were quite different during the wall following task evolution (Figure 3). In this case the Blue robot, with the full complement of sensors had the most varied performance during evolution. The Green robot, with its smaller sensor number gave the best performance as, although it was more varied than the Red robot, it did have the best cost results. The Green robot's sensor configuration, with only two light sensors and distance sensors on the front and down the left side was designed with the wall follow task in mind, and perhaps with the less input from other sensors, it was able to record better cost values, though not as consistently, compared to the other two robots.

Again, for each robot the best chromosomes from the zeroth and 499th generations were tested 20 times. As for the previous tasks, a statistically significant difference was found between the performance of the chromosomes from the beginning and end of training.



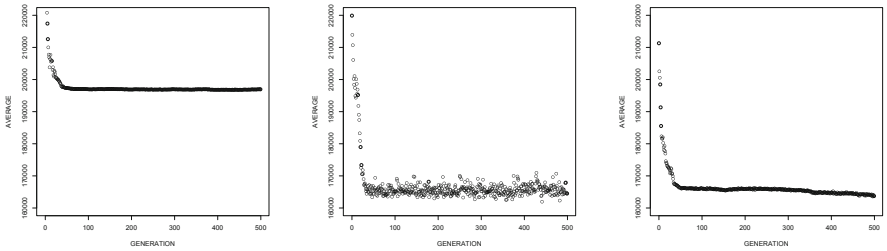
(a) Blue Robot (b) Red Robot (c) Green Robot

Fig. 3 Performance at the Wall Follow task over 500 generations of the three robots as their behaviours evolved

Patrol Dark Evolution Results. In adapting for the patrol dark task (Figure 4), the Blue robot did not reach the same performance level as the other two robots. The Red robot with its small number of sensors had the most variable performance during evolution, but produced the best average cost of the three robots. The Green robot also performed consistently well, but was slightly worse than the Red robot overall.

Again, each of the best chromosomes from the zeroth and 499th generations was run 20 times in the training environment in order to test for significant improvement in performance during the 500 generations of training. Again, there was a statistically significant difference (Kruskal Wallis at 95%) in each case.

At the end of evolution each robot had a set of schema parameters and set of cost values indicating its performance during evolution. This information was then used to initialise task performance during the execution stage.



(a) Blue Robot (b) Red Robot (c) Green Robot

Fig. 4 Performance at the Patrol Dark task over 500 generations of the three robots as their behaviours evolved

4 Lifelong Adaptation and Task Distribution

During the training phase, three robots, each with different sensor configurations, and therefore differing abilities trained to perform three tasks. At the end of training the average cost values of the final five generations of training were calculated. Then for each task, each robot calculated a sensitivity value which was used to determine how well that robot was proceeding with the task. The three robots' sensitivity values for each of the three tasks were calculated as follows: $sensitivity = \frac{1}{\frac{\bar{a}}{L}}$ where the $\bar{a}$ is the average cost value of the final five generations and L is the lowest average cost of the three robots. In this way the initial sensitivity values were scaled between zero and one, and the best (lowest cost) robot had a sensitivity of one. These values were then used to initialise the sensitivity values for the hormone-based task distribution system described in section 4.1.

The simulated world used during the lifelong experiments was the same as that used during training, but without the randomly appearing obstacles. The behaviour schema were the same as for training, with the same parameters continuing to be adapted during lifelong evolution. An evolution strategy (ES)-based method was used for adaptation during the lifelong phase of the experiments. This is described in detail in [12].

The first generation of chromosomes for lifelong evolution was made up from the best performing chromosome from the final generation of training, and two further chromosomes which were mutations of it.

4.1 Basic Hormone-Based System

Each robot has one endocrine system which has one gland for each task. Each gland releases 'hormones' at a continuous rate into a pool. Once the amount of hormone in the pool reaches a threshold it is released as 'free

hormone', and begins to decay geometrically. The free hormone with the highest concentration controls which task is executed; during which time its gland stops hormone release.

The robots continuously broadcast the task they are currently doing; when a robot receives such a signal from another robot it blocks hormone release from the gland associated with that task, and decays the amount of hormone already in the pool. The endocrine parameters were as follows: gland release rate: 1; pool threshold: 5000; pool decay rate: 0.75; free decay rate: 0.3.

In order to take into account changes in performance as the robots carry out their tasks, updated "sensitivity" values for each robot were calculated which reflect how well each robot is performing, initialised from the training stage as detailed in section 4. Lifelong evolution already calculates a performance measure (from the cost function), and it is from this value that sensitivity is calculated: $sensitivity = \frac{lastCost}{thisCost} \times lastSensitivity$
where $lastCost$ is the cost at the previous time-step, $thisCost$ is the new cost value, and $lastSensitivity$ is the sensitivity calculated at the last time-step.

The sensitivity values of each robot are used to affect the pool release rates within the robot itself, the aim being that it can begin other tasks when it performs badly at its current one. Each robot transmits the task it is currently executing to the team at each time-step. Each robot then changes the pool hormone levels for each task in one of three ways:

1. If the robot is currently doing this task, then the pool level remains at 0.
2. If there are no robots currently doing this task: the robot releases hormone at a rate of $releaseRate * sensitivity$, where $releaseRate = 1$, and $sensitivity$ is the sensitivity calculated when the robot last did this task. This meant that robots which had previously performed better at this task would fill their pool more quickly, and therefore be more likely to take on the task than robots which had previously performed poorly.
3. If other robots in the group are currently performing this task: the amount of hormone in the pool is altered by $pool - (releaseRate * (sensitivity - 1))$ where $releaseRate = 1$, and $sensitivity$ is the sensitivity of the robot for its current task. The effect is to reduce the amount of hormone in the pools when the robot is doing well at its task (sensitivity > 1), and increase it when the robot is performing poorly (sensitivity < 1), making it more likely the robot will switch tasks.

Whenever a robot's pool threshold is reached, the pool for that task is emptied as free hormone, and the pools of the other robots for that task are also zeroed. This prevents the robots from all starting the same task shortly after one another.

4.2 Simulation Results during Lifelong Adaptation

Two experiments were undertaken in simulation to test the functionality of the new endocrine system: three robots with three tasks to perform, and two

robots with three tasks to perform. The aim of these experiments was to see how the tasks were distributed between differently performing robots in these two scenarios. In each case the experiments were run for 25,000 time steps which was enough time to see how well tasks allocation was reacting to the current situation.

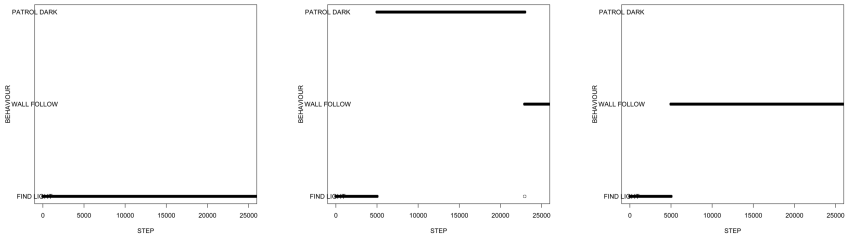
4.2.1 Simulation - Three Robots: Three Tasks

The behaviours the robots undertook during the experiment are shown in Figure 5. The robots all began doing the Find Light task until one of their other pools reached its threshold when other tasks took over. If a robot is not performing a task, its sensitivity to that task remains the last calculated value.

During the training experiments detailed in section 3.2, it was seen that the Blue robot was best at performing the Find Light task and thus had an initial sensitivity value of 1. As can be seen from the sensitivity graph in Figure 6(a), the Blue robot continued to have a high sensitivity to the Find Light task, thus showing good performance, throughout the experiment. There was no incentive for the Blue robot to change from the Find Light task as it continued performing well. Its sensitivity for Wall Follow was initially the lowest of the three robots (see Figure 6(b)). Its pool value increased the slowest when the task was not being done, and thus was always beaten to the change task threshold. Similarly with the Patrol Dark task. The Blue robot initially had the lowest sensitivity value for the task (see Figure 6(c)), although the Red robot's value became lower as it performed the task and did badly, but it was always beaten to the threshold by another robot.

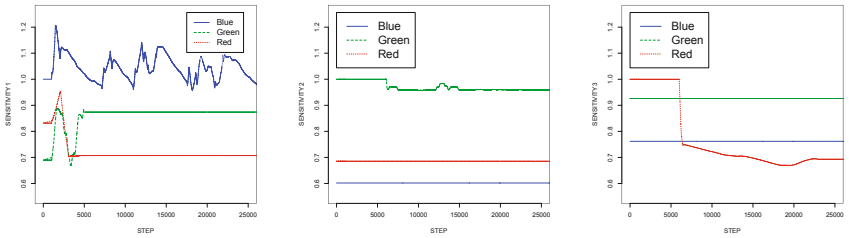
The Red robot was best at the Patrol Dark task during training and thus the sensitivity for this task was set as the highest of the three (again to 1). Thus the Red robot's pool value for Patrol Dark filled up quickest at the start when the task was not being performed by any of the robots (see Figure 7(c)) and thus the Red robot took over the Patrol Dark task. The change in behaviour of the Red robot is shown to be due to its low sensitivity to the Patrol Dark hormone of a little less than zero, that is, the robot was not performing very well at the task of Patrol Dark. This caused a steady increase in the amount of both its Find Light and Wall Follow hormones. As its sensitivity to Find Light was slightly higher than to Wall Follow the Find Light pool filled, very shortly followed by the Wall Follow which then became the task the robot did. Once no robots were performing the Patrol Dark task, the Patrol Dark hormone of all three robots began to increase in the pool. The Green robot, with its higher sensitivity approached the pool threshold fastest although the experiment ended before the threshold was reached to change tasks (see Figure 7(c)).

The Green robot was given an initialisation value of 1 for the Wall Follow task at which it was best in the training stage. This allowed the Green robot



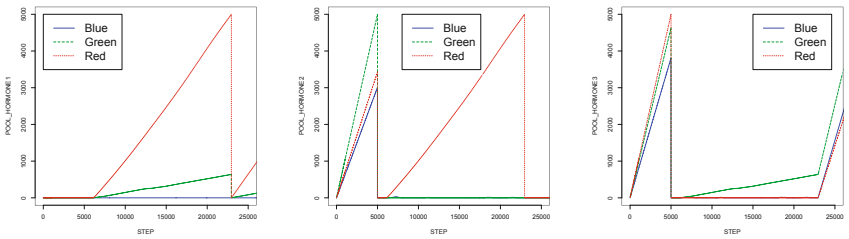
(a) Blue Robot (b) Red Robot (c) Green Robot

Fig. 5 The behaviours executed by the simulated robots over time when there were three robots and three tasks



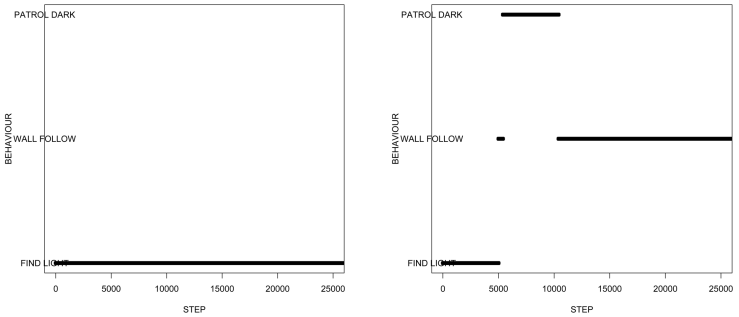
(a) Find Light Task (b) Wall Follow Task (c) Patrol Dark Task

Fig. 6 Sensitivity values over time for each of the three tasks for three simulated robots



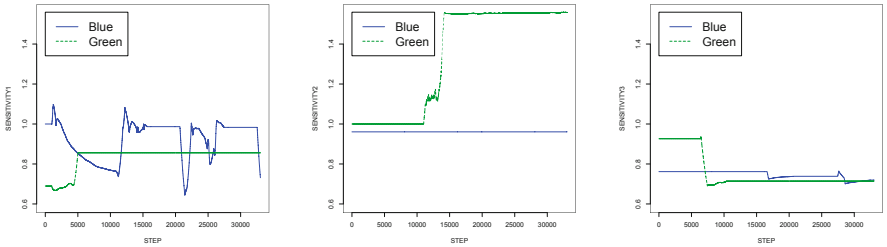
(a) Find Light Task (b) Wall Follow Task (c) Patrol Dark Task

Fig. 7 The pool values over time for each of the three tasks for all three simulated robots



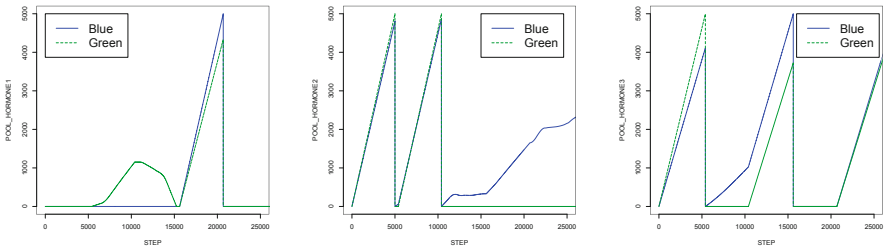
(a) Blue Robot (b) Green Robot

Fig. 8 The behaviours executed by the simulated robots over time when there were two robots and three tasks



(a) Find Light Task (b) Wall Follow Task (c) Patrol Dark Task

Fig. 9 The sensitivity values over time for each of the three tasks for both simulated robots



(a) Find Light Task (b) Wall Follow Task (c) Patrol Dark Task

Fig. 10 The pool values over time for each of the three tasks for both simulated robots

initially to fill up its Wall Follow pool the fastest and commence the Wall Follow task (see Figure 7(b)).

4.2.2 Simulation - Two Robots: Three Tasks

When there were more tasks than robots it was expected that there would be more task switching as non-executed tasks' pools filled up quickly. The behaviours executed can be seen in Figure 8, and the robots' sensitivities and pool values in Figures 9 and 10.

In this experiment (which was typical, though of course different experimental runs did produce slightly differing results), the Blue robot almost exclusively did the Find Light task, while the Green robot mostly did the Wall Follow task, each only switching to Patrol Dark infrequently and not for long. The reason for this is that both robots had a low (less than one) sensitivity to the Patrol Dark hormone. At the beginning of the experiment, when the robots are using their inherited sensitivities from the training phase, the Green robot's Patrol Dark pool fills first (Figure 9(c)), so it begins the task. However, as it performs Patrol Dark, its sensitivity to it drops as its performance drops, and so next time the Blue robot's Patrol Dark pool fills first and it begins the task (see Figure 10(c)). Both robots do not perform the Patrol Dark task for long as their sensitivity to it remains very low, that is, their ability to perform the task is very low. The task is given a chance to be done, but neither robot is capable of performing it well, so they go back to tasks they are good at.

5 Summary

This paper provides details of an endocrine-inspired task distribution system which reassigns tasks among a group of heterogeneous robots with little communication and no centralised controller. The robots were trained in simulation which provided a starting chromosome in the experimental world, and also a sensitivity value which gave the system information about how well a particular robot coped with a specific task. Some robots were better at particular tasks due to their sensor configuration. During lifelong adaptation (using an ES) where the robots autonomously tackled tasks in the experimental environment, they constantly updated this sensitivity value depending on how well they currently were performing the task. This value, through a hormonal system, allowed task changing. Depending on the values of the hormones in the pool and specifically the level of free hormone, the robot would continue with a task or change. The maximum valued free hormone determined which task the robot would choose to do. Therefore, the robot which performs best at a task is the most likely to fill up the pool quickest due to its high sensitivity value, and therefore the task gets done. However, if there are more tasks

than robots, the robots change between tasks, again, the sensitivity value determining the best choice for that selection. The experiments showed the successful distribution of tasks between heterogeneous robots ensuring that all the tasks were attempted, with the robots which were best at particular tasks most often chosen to continue with, or change to, that task. All the tasks are attempted at some point giving the possibility that they will be continued by a robot which is good at that task.

Further work will look at transferring these results to real robots.

References

1. Arkin, R.: Survivable Robotic Systems: Reactive and Homeostatic Control. Prentice-Hall (1993)
2. Arkin, R.C.: Motor Schema–Based Mobile Robot Navigation. International Journal of Robotics Research 8(4), 9–12 (1989)
3. Brooks, R.: Integrated systems based on behaviours. Sigart Bulletin 2, 46–50 (1991)
4. Maes, P.: The Dynamics of Action Selection. In: 11th International Joint Conference on Artificial Intelligence, vol. 2, pp. 991–997 (1989)
5. Mendao, M.: Neuro-Endocrine Control Architectures applied to Mobile Robotics. Ph.D Thesis, University of Kent, Canterbury, UK (2008)
6. Neal, M., Timmis, J.: Timidity: A Useful Emotional Mechanism for Robot Control? Informatica 27, 197–203 (2003)
7. Parker, L.E.: ALLIANCE: An Architecture for Fault Tolerant Multi-Robot Cooperation. IEEE Transactions on Robotics and Automation 14(2), 220–240 (1998)
8. Ram, A., Arkin, R., Boone, G., Pearce, M.: Using Genetic Algorithms to Learn Reactive Control Parameters for Autonomous Robotic Navigation. Adaptive Behavior 2(3), 277–304 (1994)
9. Shen, W.M., Salemi, B., Will, P.: Hormone-Inspired Adaptive Communication and Distributed Control for CONRO Self-Reconfigurable Robots. IEEE Transactions on Robotics and Automation 18(5) (2002)
10. Walker, J.: Experiments in Evolutionary Robotics: Investigating the Importance of Training and Lifelong Adaptation by Evolution. Ph.D. Thesis, Department of Computer Science, University of Wales, Aberystwyth, UK (2003)
11. Walker, J., Garrett, S., Wilson, M.: The balance between initial training and lifelong adaptation in evolving robot controllers. IEEE Transactions on Systems, Man and Cybernetics: Part B 36, 423–432 (2006)
12. Walker, J., Wilson, M.S.: Lifelong Evolution for Adaptive Robots. In: International Conference on Intelligent Robots and Systems (IROS), pp. 984–989 (2002)
13. Walker, J., Wilson, M.S.: Hormone-Inspired Control for Group Task-Distribution. In: Towards Autonomous Robotic Systems, TAROS (2007)
14. Walker, J., Wilson, M.S.: A Performance Sensitive Hormone-Inspired System for Task Distribution Amongst Evolving Robots. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS (2008)
15. Watson, R., Ficici, S., Pollack, J.: Embodied Evolution: Embodying an Evolutionary Algorithm in a Population of Robots. In: Congress on Evolutionary Computation (CEC), pp. 335–342 (1999)

Part IV
Modularity, Distributed Manipulation, and Platforms

Part IV: Modularity, Distributed Manipulation, and Platforms

Kasper Stoy

In distributed autonomous robotic systems, robots often consider other robots to be little more than moving obstacles. This is acceptable for many areas of application and research, however, as soon as the task or the robot system itself requires robots to come within close proximity of each other, the physicality of the robots, their physical interactions, and the coordination of these interactions become important.

The physicality of modular robots directly causes the self-reconfiguration planning problem, which is to find an action sequence that transforms an initial configuration into a desired configuration. A problem that in general is considered to be NP-hard due in part to the complex motion constraints of modular robots. Fitch and McAllister address this problem by using a hierarchical planning strategy where changes in connector topology are planned at one level and the feasibility of this plan with regards to motion constraints and collisions is checked at another level. Kernbach considers the related self-assembly problem, but view it as a constraint satisfaction problem. This gives a larger solution space since the problem is underspecified and hence many assemblies may satisfy the given constraints. Finally, Golestan, Asadpour, and Moradi propose a new graph signature calculation method for collapsing isomorphic module configurations into one state in the planning space and thereby significantly reduce the size of the planning space. Combined, these approaches to self-reconfiguration planning provide three different ways to reduce the complexity of the self-reconfiguration problem and hence make it possible to find useful plans despite the NP-hard nature of the problem.

The self-reconfiguration problem is, as described in the examples above, typically solved using a top-down approach where the goal is specified at the global level and solutions are found at the local level. This is necessary because there is little room for randomness in the self-reconfiguration process, e.g., a bad action can lead to modules dropping from the structure. Hence there is a strict order of actions that leads to success and many more that lead to failure. However, bottom-up approaches, where the local interactions produce a globally desired behavior, are also often used in modular robotics in cases where there is more room for error. An example of this is the work by Zahadat, Christensen, Katebi and Stoy where the

Kasper Stoy
The Maersk Mc-Kinney Moller Institute, University of Southern Denmark,
Campusvej 55, 5230 Odense M, Denmark
e-mail: kaspers@mmmi.sdu.dk

local actions of modules are designed to make a desired global locomotion behavior emerge. In this particular work, evolved fractal gene regulatory networks are used to control locomotion of the ATRON modular robot. Similarly, An, Ikemoto, Asama, and Arai use a bottom-up perspective to find synergies between distributed muscle actuation during a human standing-up motion.

While handling physicality is very important in modular robots, it also comes into play in multi-robot systems when robots manipulate objects collectively. An example of this is the work of Mellinger, Shomin, Michael, and Kumar on cooperative grasping and transport using multiple quadrotors where great care has to be taken to ensure precise coordination. In systems where the physical interaction is indirect, emphasis can be put on robustness and scalability as is the case for the work of Fujisawa, Imamura, and Matsuno on using pheromone communication for cooperative transportation of many small objects. Robustness also comes into play in the work by Ferrante, Brambilla, Birattari, and Dorigo where the coordination for the purpose of navigation and obstacle avoidance is socially mediated. In the work by Bonani, Rétornaz, Magnenat, Bleuler, and Mondada the physical interaction between robots and between the assembled robot and the environment is improved by considering a heterogeneous multi-robot system where each robot is morphologically optimized for its role and task within the swarm.

Regarding the physical robot hardware itself, McLurkin, Lynch, Rixner, Barr, Chou, Foster, and Bilstein open the doors to the exciting field of distributed autonomous robot systems by describing a low-cost multi-robot system for research, teaching, and outreach.

Overall, the work described here gives a nice overview of what to do when robots of a distributed autonomous robotic system get into physical contact directly as is the case for modular robots or indirectly through manipulated objects as is the case for multi-robot systems.

Hierarchical Planning for Self-reconfiguring Robots Using Module Kinematics

Robert Fitch and Rowan McAllister

Abstract. Reconfiguration allows a self-reconfiguring modular robot to adapt to its environment. The *reconfiguration planning* problem is one of the key algorithmic challenges in realizing self-reconfiguration. Many existing successful approaches rely on grouping modules together to act as meta-modules. However, we are interested in reconfiguration planning that does not impose fixed meta-module relationships but instead forms cooperative relationships between modules dynamically. This approach avoids the need to hand-code meta-module motions and potentially allows reconfiguration with fewer modules. In this paper we present a general two-level reconfiguration framework. The top level plans in module-connector space using distributed dynamic programming. The lower level accepts a transition function for the kinematic model of the chosen module type as input. As an example, we implement such a transition function for the 3*R*, SuperBot-style module. Although not explored in this paper, this general approach is naturally extended to consider power use, clock time, or other quantities of interest.

1 Introduction

Self-reconfiguring modular robots use module disconnections and reconnections to change their overall shape. In so doing, these robots can adapt to the environment or task at hand. Performing such adaptation requires solving the algorithmic problem of computing a sequence of module moves that transforms an initial shape into a goal shape. This problem, known as the *reconfiguration problem*, remains one of the key algorithmic challenges in self-reconfiguring robotics.

There are several dimensions by which to categorize specific instances of the reconfiguration problem. Algorithms have been proposed for specific module types, such as unit-compressible modules [4, 25], and abstract cube modules with simple motion primitives [7]. The idea in planning for an abstract module is to compile down an abstract move into a sequence of native moves. A possible technique to accomplish this is to simplify the problem by treating a group of modules as a

Robert Fitch · Rowan McAllister
Australian Centre for Field Robotics (ACFR), ARC Centre of Excellence for Autonomous Systems, The University of Sydney, Sydney NSW Australia
e-mail: {rfitch,r.mcallister}@acfr.usyd.edu.au

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 477–490.
springerlink.com

single meta-module with fewer kinematic constraints. Two other important issues are parallelism – how many modules can move at one time – and decentralized versus centralized control. We are interested in the question of autonomous reconfiguration planning that acts directly in the native kinematic action space of the module, and does not use meta-modules. In this paper, we study the problem of general parallel decentralized reconfiguration planning for given module kinematics.

There are several reasons to address this specific variation of the reconfiguration problem. It is useful to consider pairs or small groups of modules working together, but planning for individual modules allows these groupings to be dynamic. Avoiding static meta-modules reduces the minimum number of modules required for reconfiguration. This is especially useful for hardware prototypes with few modules. Furthermore, planning in the native kinematic space of the module opens the possibility of optimizing reconfiguration for various quantities of interest. We are interested in a planner that can consider power use, time cost, and (heterogeneous) modules with differing capabilities. A general planning framework that easily admits changes to its underlying kinematic model also opens the possibility of using reconfiguration simulation as a tool for design optimization. The effects of simplifying a given module design by removing a degree of freedom, for example, could be readily evaluated.

The fundamental challenge in solving the reconfiguration problem is that the number of degrees of freedom in a self-reconfiguring robot increases with the number of modules. The number of possible configurations thus increases exponentially. These combinatorial issues have been understood for many years [18]. Searching this huge space directly is not possible; some structure must be imposed on the problem. The success of meta-module and cube-module planners relies on such a structuring.

Our approach is to build on our earlier planner for abstract cube-shaped modules [9] hierarchically by adding a lower level. The low-level planner computes a sequence of moves, in the joint space of the module, that results connection/ disconnection. This approach decomposes the full problem into many local subproblems. Each subproblem is a kinematic motion planning problem small enough to be solved quickly. Chained together, these solutions move a single module from one point in the robot to another along a sequence of intermediate connections. Point-to-point paths are then computed, as in our abstract cube planner, by formulating a *Markov-decision problem* (MDP) and solving it using distributed dynamic programming. The value function acts as a navigation function over all connectors that indicates the next step towards an open goal position. Many modules share this navigation function. As modules move, the navigation function is updated.

We present our reconfiguration algorithm as a general framework that accepts a module's kinematic model in the form of a transition function. We present a specific transition function for SuperBot-style modules [21] as an example, and illustrate its behavior with simple examples in simulation. Our intention is for this example to provide sufficient information such that other researchers can implement this algorithm on various module types.

The paper is organized as follows. We discuss related work in Sect. 2. In Sect. 3, we present details of our cube-style planner as background information and then present our general reconfiguration algorithm. We define a sample transition function for SuperBot-style modules in Sect. 4 along with implementation examples in simulation. Sect. 5 concludes the paper with discussion and future work.

2 Related Work

Reconfiguration planning is a well-studied problem. A survey of accomplishments can be found in [26]. Another survey paper is [20]. We briefly discuss a selection of relevant results in this section.

The root of this paper lies in cellular automata-based locomotion [3]. The MILLION MODULE MARCH algorithm [9] can be viewed as a generalization of this idea. The present paper can be viewed as a further generalization along two fronts: goal shape representation and module kinematics.

A number of planners leverage the concept of meta-modules. Important examples include planners for MTRAN [27], ATRON [6] and I-Cubes [19]. The key difference between this work and ours is that we are interested in the question of how to reconfigure without meta-modules.

Complete planners have been developed for unit-compressible modules [4, 25]. Other early work in reconfiguration planning includes [5, 14]. The idea of gradient-based planning is explored in [22] and [23]. A graph-signature method is presented in [1].

A planner for SuperBot modules viewed in a chain-based manner is presented in [11]. Optimal reconfiguration for chain-based robots was recently proven to be NP-complete [12].

3 Hierarchical MDP Planning with Dynamic Programming

The reconfiguration algorithm we propose in this paper builds on our earlier MILLION MODULE MARCH algorithm for scalable locomotion through reconfiguration [9]. In this section we summarize MILLION MODULE MARCH for convenience, focusing on the MDP formulation and dynamic programming solution method. We then present a new MDP formulation that, unlike MILLION MODULE MARCH, models native module kinematics. We define a general reconfiguration algorithm based on this new MDP formulation. Like MILLION MODULE MARCH, this new algorithm is fully decentralized and scalable.

3.1 Background: MDP Planning with Abstract Modules

The MILLION MODULE MARCH algorithm was originally presented as a scalable algorithm for locomotion-through-reconfiguration for the Sliding Cube [7] module

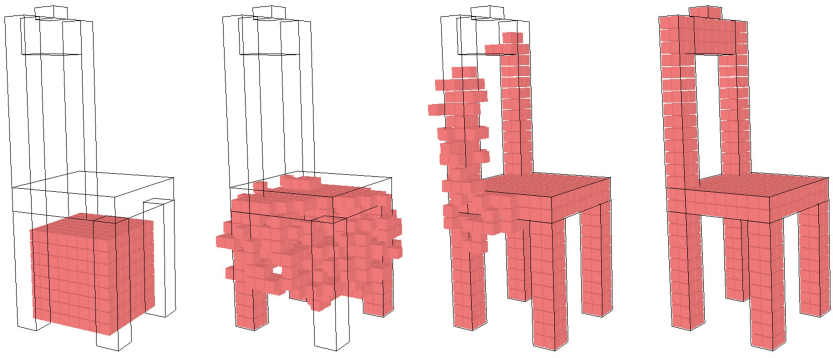


Fig. 1 Reconfiguration example using Algo. 1 and the Sliding Cube module abstraction. Simultaneously executing single module paths results in global reconfiguration. Subfigs. (a) through (d) show four stages of a reconfiguration sequence that assembles a chair shape from an initial cube shape. Simple assembly order heuristics are used that guide modules to the bottom center of the goal shape as it is formed.

abstraction. The algorithm produces locomotion by first specifying a goal bounding box at an offset to the current location. Modules move to fill the box, the box is shifted in a receding-horizon fashion, and locomotion results. Providing a different shape for the goal results in reconfiguration into that shape, for convex goal shapes. Non-convex goal shapes are also possible with the addition of local assembly rules that prevent internal holes from forming [13]. Fig. 1 shows an example of reconfiguration into a chair shape. The algorithm is fully decentralized and has been implemented in simulation with million-module systems [9]. It has also been implemented on embedded processors with wireless radio communication in hardware-in-the-loop simulation [10, 15], and extended to control a team of nine mobile robots [8].

The algorithm is composed of two main components: (1) planning via a global navigation function; and (2) control of parallel module movements (connectivity checking) via local graph search and shared locks. The essence of the second component is that each module, in parallel, searches for a local module substructure sufficient to guarantee that it is a non-articulation point in the module connectivity graph. This search is performed using message-passing. If successful, modules in the substructure are temporarily locked (prevented from moving) until the locking module has completed its move. Locks can be shared by multiple moving modules. Many modules can thus safely move in parallel while preserving global connectivity. This component of the algorithm is used unmodified in the present work. Full implementation details are provided in [10].

The planning component of MILLION MODULE MARCH computes a value function that acts as a global navigation function. Modules use this function as a one-step planner to choose the next move. By sequentially choosing such moves, each

module is guided towards an available destination in the goal shape. As many modules move in parallel, the topology of the robot structure also changes. The value function is updated online to reflect these topology changes (continuous replanning).

The planning problem is formulated as a distributed MDP. An MDP is a sequential decision-making problem defined by a $4-tuple < S,A,T,R >$, where S is the set of states, A is the set of actions, T is the transition function that maps state-action pairs to resulting states, and R is a one-step reward function [24]. A decision-making agent repeatedly takes actions and earns rewards. Its objective (commonly) is to maximize the sum of future rewards. If the transition function is known, dynamic programming can be used to solve the MDP. A solution is a policy mapping states to actions. This policy can be encoded as a value function over states. The transition function can be either deterministic or stochastic.

The set of states in MILLION MODULE MARCH is the set of module faces. In the Sliding Cube abstraction, a module is a cube that lives in a cubic lattice. Therefore the set of allowable states can be thought of as open lattice positions adjacent to at least one other module. The Sliding Cube model provides two motion primitives - a sliding move and a convex transition. These primitives define the action set. A module can either make an axis-aligned (lateral) move, or move "diagonally" around another module. The transition function is also defined by these two motion primitives. The reward function is -1 per move.

The value function is stored in a distributed fashion. Each module stores the value of states corresponding to its connectors. The MDP is solved using asynchronous distributed dynamic programming implemented with message passing. An update is performed when a module receives a message with a value for a nearby state. Using the transition function, the module updates its local value function and sends these new values to its neighbors. This process is guaranteed to converge in polynomial time in the number of states [17]. A moving module queries the value function by

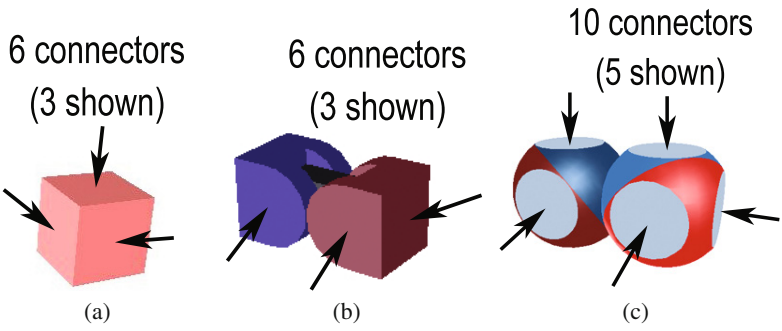


Fig. 2 $M-C$ space is a generalization of the Sliding Cube representation to any module type. The Sliding Cube abstraction, (a), has a connection on every face. Other module types, such as SuperBot-style modules, (b), and Roombot-style modules [1], (c), do not. $M-C$ space is simply defined as the set of all module-connector pairs.

sending a request to its connected neighbors. After a move, changed values are again sent to neighbors and the value function is updated.

3.2 MDP Planning with Native Module Kinematics

Building on the Sliding Cube MDP formulation, we now introduce a new MDP formulation that replaces the Sliding Cube and instead assumes the availability of a kinematic model for a physical module. Instead of abstract motion primitives, motion primitives now correspond to changes in module joint angle and connector state. Because actions are no longer unit-time, this is technically a semi-markov decision problem (SMDP) [2]. However, for the purposes of this paper we assume unit-time actions. The SMDP formulation allows more sophisticated optimization (time, power, etc.) but we will leave this for future work.

To define the state space, we first define the set of module-connector pairs, or *M-C space*. Fig. 2 illustrates sample $M-C$ states for three different module types. The entire set is not necessarily reachable. One obvious example of a non-reachable state is a connector that is occupied (connected to another module). In general, reachability is determined by the transition function. The transition function, in turn, is partially determined by the robot configuration topology. Therefore, connectivity of $M-C$ space changes with reconfiguration. A module with k connectors can potentially occupy k M-C states simultaneously. Our state space S is therefore defined as the set of all k-tuples of $M-C$ states, including a null $M-C$ state that models a free connector. Fig. 3 shows an example of a single module making state transitions in $M-C$ space and the corresponding module movements in a sample configuration.

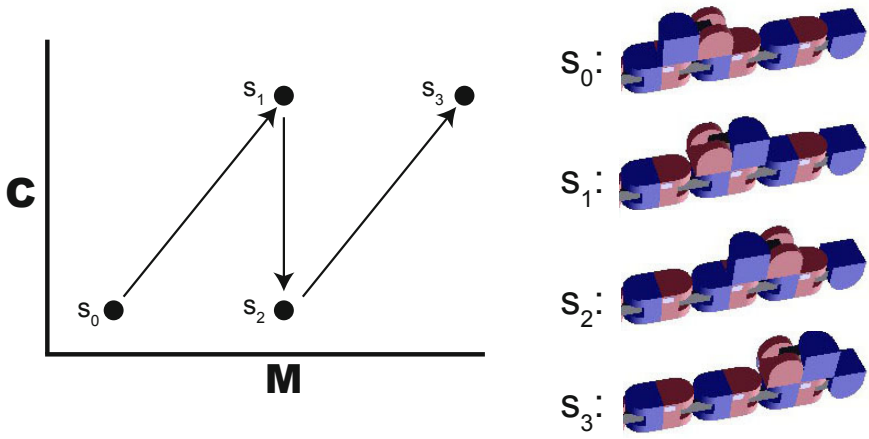


Fig. 3 Path of a single $3R$ module moving through $M-C$ space. States s_i are $M-C$ states. A state transition in $M-C$ space corresponds to a module attaching to a new connector in the robot workspace. Corresponding module movements are illustrated in the right half of the figure.

The set of actions is defined by the kinematic model. We assume that an action consists of a set of joint angle increments and connection/disconnection actions. An action can involve a single module, or a module plus one or more helper modules.

Actions result in changing state. This means that the set of occupied $M-C$ states will change following a successful action. An action that fails or otherwise does not result in a state change is a null action. The transition model defines this change, mapping a state-action pair to a resulting state: $T(s,a) = s'$, where $s \in S$, $s' \in S$, and $a \in A$. The transition function must take into account surrounding modules. The potential for collision means that not all actions are available at all times. The transition function can be stochastic.

The reward function is -1 for every action. This attempts to minimize the total number of actions. A more sophisticated reward function can be used to minimize other quantities, such as time, power use, heterogeneous modules, etc. Further, the reward function can be modified during reconfiguration to allow the robot to adapt to changes. However, we do not consider these possibilities in this paper.

3.3 Hierarchical Reconfiguration Algorithm

Having formulated the MDP, we solve using dynamic programming. To allow modules to move in parallel, we integrate the parallel movement control approach from MILLION MODULE MARCH. To prevent collisions, we lock all modules within the workspace of a moving module. The algorithm is listed in pseudocode as Algo. 1. Transition function T, goal configuration G, and start configuration c are assumed as input. In parallel, modules follow a path to the goal by chaining together a sequence of state transitions. Within the goal, modules are guided by local assembly order heuristics as above. The algorithm terminates when all modules are in the goal.

The value function is recomputed as the robot configuration changes, as described in Sect. 3.1. The MDP will converge in polynomial time [17]. Convergence of the robot to the goal shape depends on the transition function supplied.

Algorithm 1. General framework for reconfiguration.

T: a transition function
G: a goal shape
c: the current robot configuration

Generate value function V for c given T and G using dynamic programming
repeat
  Find mobile modules
  Move mobile modules one step according to V
  Recompute V using new configuration c'
until all modules in goal

4 A Local Kinematic Planner for $3R$ Modules

In this section, we flesh out Algo. 1 by defining a transition function based on module kinematics. There are many ways to do this in general. We have chosen to view the problem as motion planning for an $n-$link kinematic chain among regular orthohedral obstacles. Motion planning in high-dimensional spaces is computationally intensive. By imposing this strict structure, we can use a simple grid search method for motion planning. We illustrate this technique with the $3R$ (SuperBot-style) module.

$M-C-O-\Theta$ space is $M-C$ space augmented by adding two extra dimensions, $O \in \{e,s\}$ and $\Theta \in \{0,90,180,270\}$, that represent how a module is connected to the $M-C$ pair. Due to connector symmetry, we can encode which connector is connected by specifying and end (e) or a side (s). The Θ dimension encodes rotation represented discretely in 90-degree increments.

We consider two cases for planning. The first is single module motion. Given a starting (m,c,o,θ) state, lattice (workspace) position, and vector of joint angles x, we use the forward kinematics of the $3R$ module to determine the position of its connectors in the workspace. We then consider the set of actions formed by all permutations of discrete 90-degree increments/decrements of joint angles. We iterate through this set of actions. At each iteration, we add the joint angle increments to

Algorithm 2. A local kinematic planner. This planner dynamically computes the transition function for the reconfiguration MDP.

s_{start}: starting configuration
N: local neighborhood of modules around s_{start}
M: list of moves for output, initially empty
A: set of actions (joint angle increments)
T: search tree, initially empty
S: search queue, initialized with s_{start}

while S not empty **do**
  pop search node s from S
  if s not in T **then**
    add s to T
    if s is a goal configuration in N **then**
      add new move to M
    end if
    for all actions $a \in A$ **do**
      generate new state s' by integrating forward from s
      if path from s to s' is collision-free **then**
        add s' to S
      end if
    end for
  end if
end while

output M

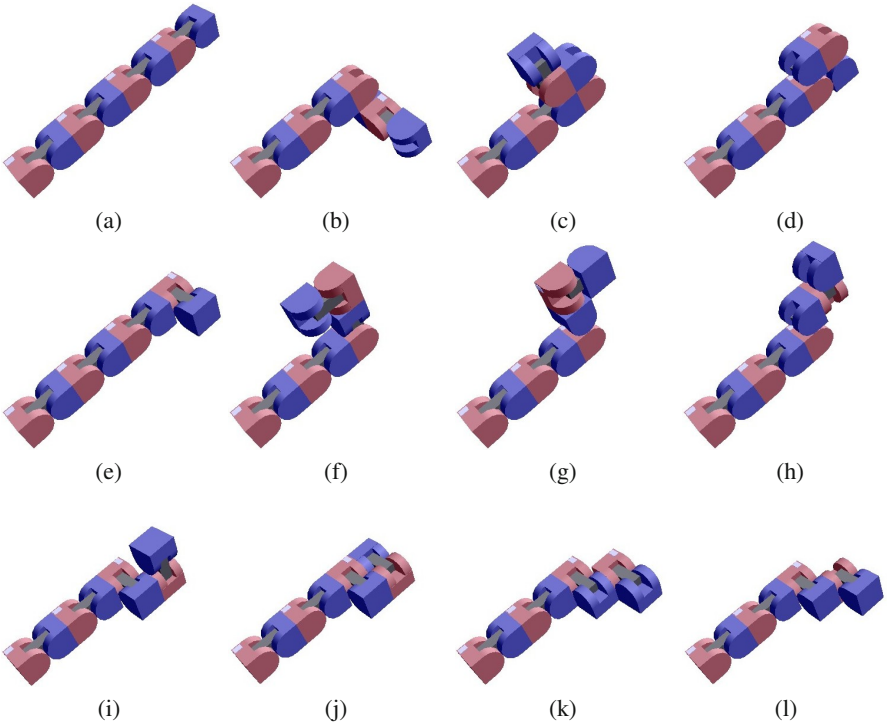


Fig. 4 Workspace reachability. Subfigs. (a) through (d) show sample configurations reachable with a helper module attached in an end-to-end configuration. Likewise, Subfigs. (e) through (h) show samples reachable from an end-to-side configuration. Subfigs. (i) through (l) correspond to a side-to-side configuration.

the initial position, resulting in a new joint angle vector x'. We again use the forward kinematics to compute the new position of connectors in workspace. If no connectors are in a position to connect to some other connector in the neighborhood, this configuration is discarded. Otherwise we perform collision checking in the workspace. We check intermediate configurations between x and x' in small increments, as described in [16]. If there is a collision, this configuration is discarded. Else we place x' on a queue and continue. We then pop the queue and repeat. When the queue is empty, the algorithm terminates.

The second case involves a helper module. A helper module is a (connected) neighbor. In this case, the joint angle vector includes joints of both modules. We search as described above.

The algorithm is listed in pseudocode as Algo. 2. Fig. 4 shows examples of different $O - \Theta$ configurations. In the helper case, a module can reach positions up to a radius of manhattan distance four from its end connector. Fig. 5 shows examples

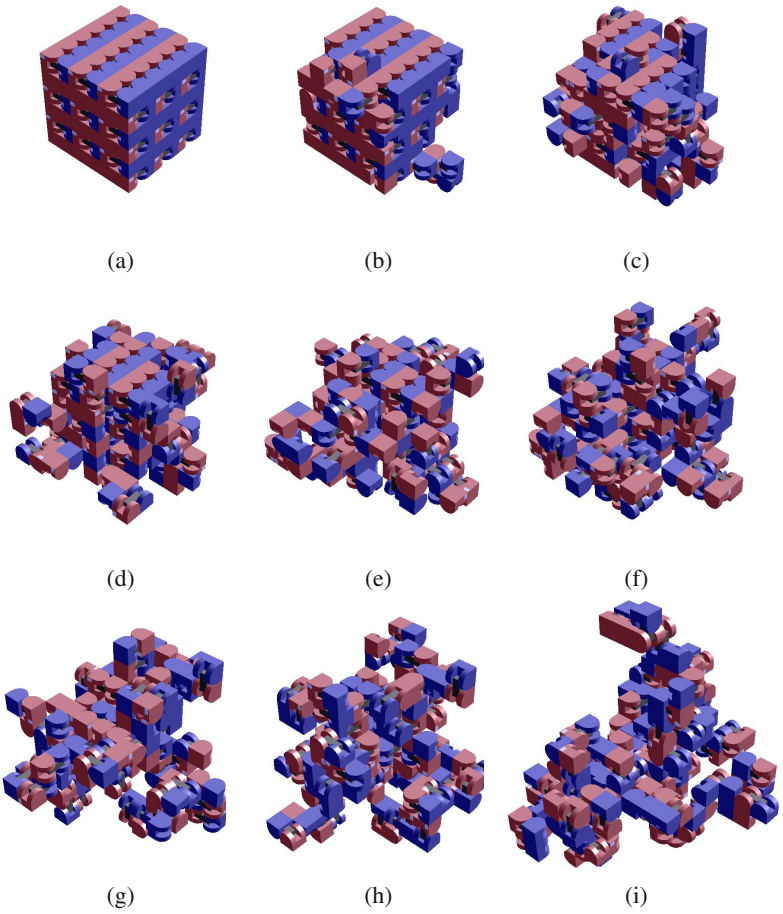


Fig. 5 Sample configurations generated by successive random module movements in a robot with 108 modules

drawn from a sequence of configurations generated by successive random module movements.

Because we search all joint angle positions, the running time of this algorithm is exponential in the number of joint angles. For constant-length chains of helper modules, time is constant (albeit with a potentially large constant factor). For two $3R$ modules, the size of the search space is $3*4*3*3*4*3=1296$. This is reasonable to implement with modest embedded computational resources, even considering that in computing the value function, each module must perform this computation for each possible (o,θ) pair $(2*4=8)$ and each of its open connectors.

4.1 Implementation

We implemented the reconfiguration algorithm in the SRSim simulation environment [7], with SuperBot graphics rendered by a simulation developed at ISI. Collision checking is implemented by testing for intersections between the bounding box surrounding each module part and those surrounding modules in its neighborhood. Configuration space is represented as a 6D grid corresponding to module joint angles. The grid can represent one helper module in addition to the main module. Fig. 6 shows an example of nine modules reconfiguring from a line shape into a box. Fig. 7 shows an example reconfiguration between two cuboid configurations.

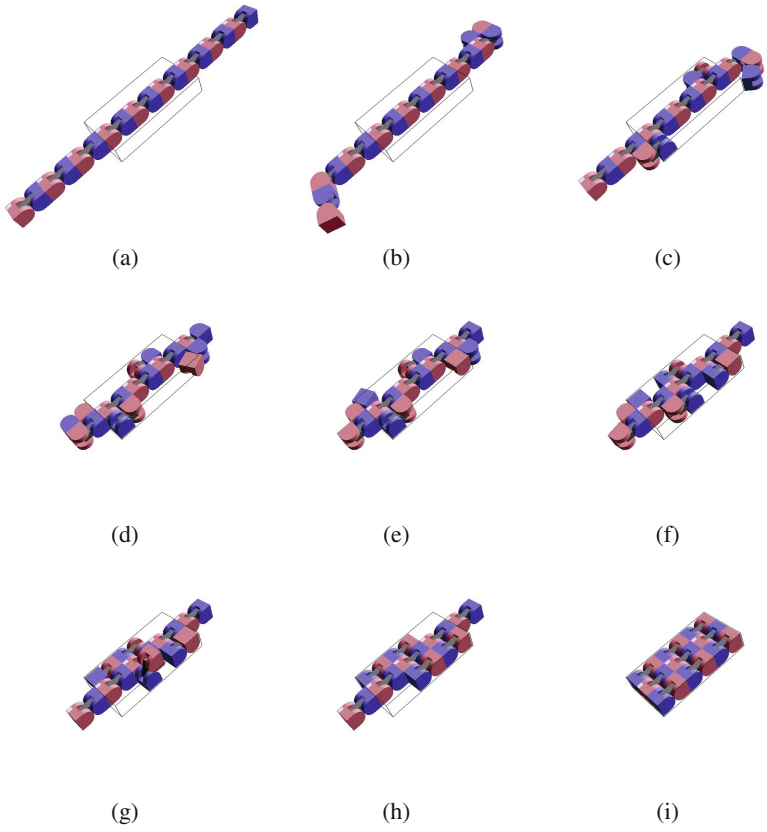


Fig. 6 Nine modules reconfiguring from a line shape into a box shape

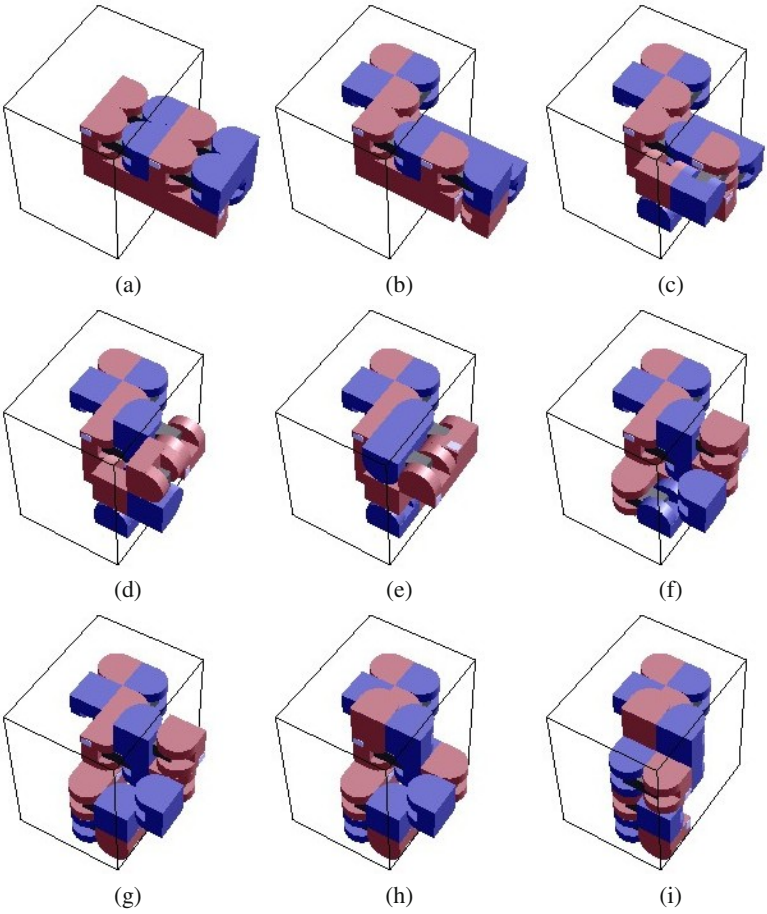


Fig. 7 Eight modules reconfiguring from an initial cuboid configuration into a goal configuration specified by the wire-frame bounding box shown

5 Discussion and Future Work

We have presented a general framework for reconfiguration and an example implementation for SuperBot-style modules. Because this simple implementation is exponential in the degrees of freedom of the kinematic chain, this planner is suited mainly to lattice-based and hybrid robot types. A planner for chain-based robots (with short chains) could possibly be developed. We have not yet explored the potential in optimizing for quantities other than number of connection/disconnection cycles, but this should be a promising avenue. So far we have been concerned only with finding a feasible reconfiguration plan, but another interesting problem would be to attempt to prove an approximation to optimal reconfiguration. One idea is to

build on the lower-bound construction for reconfiguration [18] and attempt to prove an upper-bound on the maximum deviation from shortest path taken by any module in travelling to the goal.

We are currently implementing our algorithm in a decentralized fashion in hardware-in-the-loop simulation [15]. Computation and communication run on embedded processors but actuation is simulated on a desktop computer. It is also our intention to test the algorithm on real robots. One possible platform is a new module we are currently constructing. This module has SuperBot-style kinematics combined with a novel connection mechanism based on grippers or pincers. We would also like to implement and test our algorithm on other module types.

Acknowledgements. This work is supported by the ARC Centre of Excellence programme, funded by the Australian Research Council (ARC) and the New South Wales (NSW) State Government. Many thanks to Surya Singh for lending expertise in robot kinematics. The SuperBot simulator was written by David Brandt under the auspices of ISI.

References

1. Asadpour, M., Sproewitz, A., Billard, A., Dillenbourg, P., Ijspeert, A.J.: Graph signature for self-reconfiguration planning. In: Proc. of the IEEE/RSJ IROS, pp. 863–869 (2008)
2. Barto, A., Mahadevan, S.: Recent advances in hiearchical reinforcement learning. Special Issue on Reinforcement Learning, Discrete Event Systems Journal 13, 41–77 (2003)
3. Butler, Z., Kotay, K., Rus, D., Tomita, K.: Generic decentralized locomotion control for lattice-based self-reconfigurable robots. Int. J. Rob. Res. 23(9) (2004)
4. Butler, Z., Rus, D.: Distributed motion planning for modular robots with unit-compressible modules. Int. J. Rob. Res. 22(9), 699–716 (2003)
5. Chiang, C.H., Chirikjian, G.: Modular robot motion planning using similarity metrics. Autonomous Robots 10(1), 91–106 (2001)
6. Christensen, D., Stoy, K.: Selecting a meta-module to shape-change the atron self-reconfigurable robot. In: Proc. of IEEE ICRA, pp. 2532–2538 (2006)
7. Fitch, R.: Heterogeneous self-reconfiguring robotics. Ph.D. thesis, Dartmouth College (2004)
8. Fitch, R., Alempijevic, A., Lal, R.: A self-reconfiguring team of mobile robots. In: Proc. of IEEE ICRA, Workshop on Network Science and Systems in Multi-Robot Autonomy (2010)
9. Fitch, R., Butler, Z.: Million module march: Scalable locomotion for large self-reconfiguring robots. Int. J. Rob. Res. 27(3-4), 331–343 (2008)
10. Fitch, R., Lal, R.: Experiments with a ZigBee wireless communication system for self-reconfiguring modular robots. In: Proc. of IEEE ICRA, pp. 1947–1952 (2009)
11. Hou, F., Shen, W.M.: Distributed, dynamic, and autonomous reconfiguration planning for chain-type self-reconfigurable robots. In: Proc. of IEEE ICRA (2008)
12. Hou, F., Shen, W.M.: On the complexity of optimal reconfiguration planning for modular reconfigurable robots. In: Proc. of IEEE ICRA (2010)
13. Itzstein, B.: Assembly order planning for stable self-reconfiguration of modular robots. Undergraduate thesis, The University of Sydney (2009)
14. Kotay, K., Rus, D.: Algorithms for self-reconfiguring molecule motion planning. In: Proc. of the IEEE/RSJ IROS (2000)
15. Lal, R., Fitch, R.: A hardware-in-the-loop simulator for distributed robotics. In: Proc. of ARAA Australasian Conference on Robotics and Automation, ACRA (2009)
16. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge (2006)

17. Littman, M.L., Dean, T.L., Kaelbling, L.P.: On the complexity of solving markov decision problems. In: Proc. of UAI, pp. 394–402 (1995)
18. Pamecha, A., Ebert-Uphoff, I., Chirikjian, G.: Useful metrics for modular robot motion planning. IEEE Trans. on Robotics and Automation 13(4), 531–545 (1997)
19. Prevas, K.C., Unsal, C., Efe, M.O., Khosla, P.K.: A hierarchical motion planning strategy for a uniform self-reconfigurable modular robotic system. In: Proc. of IEEE ICRA (2002)
20. Fitch, R., Rus, D.: Self-reconfiguring robots in the USA. Journal of the Robotics Society of Japan 21(8), 4–10 (2003)
21. Salemi, B., Moll, M., Shen, W.M.: SUPERBOT: A deployable, multi-functional, and modular self-reconfigurable robotic system. In: Proc. of IEEE/RSJ IROS (2006)
22. Stoy, K.: Controlling self-reconfiguration using cellular automata and gradients. In: Proceedings of IAS-8 (2004)
23. Stoy, K., Nagpal, R.: Self-reconfiguration using directed growth. In: 7th International Symposium on Distributed Autonomous Robotic Systems, DARS 2004 (2004)
24. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
25. Vassilvitskii, S., Yim, M., Suh, J.: A complete, local and parallel reconfiguration algorithm for cube style modular robots. In: Proc. of IEEE ICRA, pp. 117–122 (2002)
26. Yim, M., Shen, W.M., Salemi, B., Rus, D., Moll, M., Lipson, H., Klavins, E., Chirikjian, G.: Modular self-reconfigurable robot systems. IEEE Robot. Automat. Mag. 14(1), 43–52 (2007)
27. Yoshida, E., Matura, S., Kamimura, A., Tomita, K., Kurokawa, H., Kokaji, S.: A Self-Reconfigurable Modular Robot: Reconfiguration Planning and Experiments. Int. J. Rob. Res., 903–915 (2002)

Heterogeneous Self-assembling Based on Constraint Satisfaction Problem

Serge Kernbach

Abstract. This paper is devoted to a self-assembling of heterogeneous robot modules into specific topological configurations with desired kinematic properties. The approach utilizes a constrained nature of self-assembling and involves constraint satisfaction and constraint optimization techniques for finding optimal connections between modules. Scalability, locality and noise of sensor information as well as environmental dependability are addressed. This approach is implemented in real reconfigurable robots and in simulation.

1 Introduction

Reconfigurable robotics is a well-established research field, which involves such areas as evolutionary computation, bio-inspired and developmental systems as well as topology or non-linear dynamics [3]. This field is characterized by multiple challenges related to a platform development, complex kinematic calculations, finding optimal morphology and functionality for heterogeneous modules, distributed self-assembling and other problems [10].

State of the art solutions for morphological problems refer to evolutionary algorithms for evolving structures and functionality in the off-line and off-board mode (i.e. in simulation on external computer) [11]. The task for on-line and on-board mode is rather to select and to adapt one of pre-evolved (or pre-developed) solutions instead of evolving the required topology and functionality anew. Combination of off-line pre-development and on-line selection/adaptation of structural solutions has several advantages, such as on-demand availability of different kinematic, controlling, homeostasis, energetic and other mechanisms, safe and fast adaptation in real environments. Using on-line and off-line approaches for self-assembling has been

Serge Kernbach
University of Stuttgart, Germany
e-mail: Serge.Kernbach@ipvs.uni-stuttgart.de

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 491–503.
springerlink.com

already considered in [7]. This paper extends that idea and introduces the constraint-based approach for topological problems.

The self-assembling structures are limited by multiple constraints, e.g. useful kinematics, specific connectivities, required degrees of freedom, scalability properties and other constraints. It is natural to formulate distributed self-assembling of reconfigurable robot modules in the form of Constraint Satisfaction Problem (CSP) and Constraint Optimization Problem (COP). Due to connectivity and functional constraints, this approach is very useful for modules with different geometry and functionality, i.e. for heterogeneous reconfigurable robots. It allows addressing challenges of noisy and incompetence sensor information and optimality of topologies based on the selected cost function. Since linear optimization is very fast, this approach can be run on-board and on-line. Moreover, optimization can be considered as a mean of synchronization between different modules (i.e. two independent optimizers receive the same results when they use the same initial data). This allows using self-organizing mechanisms for a structural regulation.

The rest of the paper is organized in the following way. Sec. 2 introduces a connectivity-based description of topologies and integration of kinematic constraints into self-assembling. Sec. 3 formulates CSP/COP, cost function and scalability approaches. Sec. 4 describes implementation and performed experiments, whereas Sec. 5 concludes this work.

2 Description of Topologies for Self-assembling

Example of heterogeneous reconfigurable modules is shown in Fig. 1. All these modules have the same docking mechanism and can dock to each other. Modules differ in a number of docking elements, in a provided functionality (degree of freedom of individual modules) and geometries. Since assembling and disassembling are performed on a 2D plane, most topologies of artificial organisms

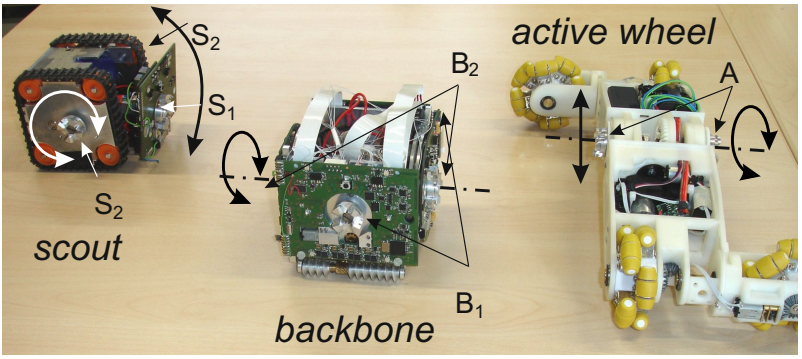
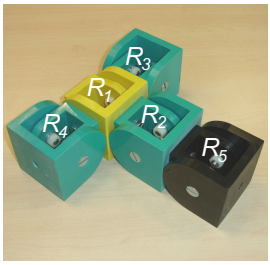


Fig. 1 Example of heterogeneous robot modules (prototypes) from the SYMBRION/REPLICATOR projects. Individual degree of freedom are shown, letters denote corresponding docking elements, see Table 2.



(a)

(b)

j \ i	R_1	R_2	R_3	R_4	R_5
R_1	D_1	Z_5	Z_9	Z_{13}	0
R_2	Z_5^T	B_1	0	0	0
R_3	Z_9^T	0	C_5	0	Z_9
R_4	Z_{13}^T	0	0	B_1	0
R_5	0	0	Z_9^T	0	B_1

(c)

Fig. 2 **(a)** Model of a simple topology; **(b)** High-dimensional configuration matrix based on the docking connections, see more in [7]; **(c)** Low-dimensional configuration matrix based on the connections between modules. Type of connection is coded based on the docking connections from (b).

generally belong to 2D grid-based reconfigurable systems. The matrix-based (and correspondingly a graph-based) representation of such topologies is common in reconfigurable robotics, see e.g. [12] or [9]. Such a representation for the model of a simple topology is shown in Fig. 2. Here several high- and low- dimensional representations [7],[2] are distinguished.

The matrix-based description of topologies has several disadvantages: it requires a large memory for on-board storage and processing; it introduces IDs of placeholders (descriptors of robots in the configuration matrix), and it restricts topologies only to those, which are described by this matrix. There are also several proposals to improve this description, e.g. [4], most of them utilize symbolic, operational and topological generators. The symbolic generators use production rules: each symbol a_i means specific connection $a_i : x_i \rightarrow x_j$, L-systems [1] are well-known examples. Operational generators are based on a structural decomposition into standard topologies and operation on them (e.g. topology from Fig. 2 can be decomposed into "T" shape with $R_1 - R_4$ and extension R_5), each of them is described by its own operator, see more in [7]. Topological generators are based on properties of symmetric and circulant matrices [5], which allows a compact analytical generation of corresponding matrices, see more in [7], [6].

As mentioned, there are multiple constraints, imposed on connectivity, kinematic properties, heterogeneity and others. Therefore it makes sense to describe a topology also in the form of constraints. Let us consider the Fig. 3, which shows 2x segmented cross (2x centipede or "dog"). It can be remarked that such a topology: **(1)** can be split on a combination of several so-called "core" elements ($R_1 - R_5$ and $R_6 - R_{10}$), the cores have a low number of elements. Decomposition on cores enables us to reduce the dimensionality of self-assembling and to consider large topologies as a scalability/deviation problem; **(2)** all elements within/between cores are connected to each other in a specific way, i.e. each connection has a defined DoF/functionality; **(3)** core elements have a specific connectivity of all components, such as 4x cross-like, 3x triangle-like and others.

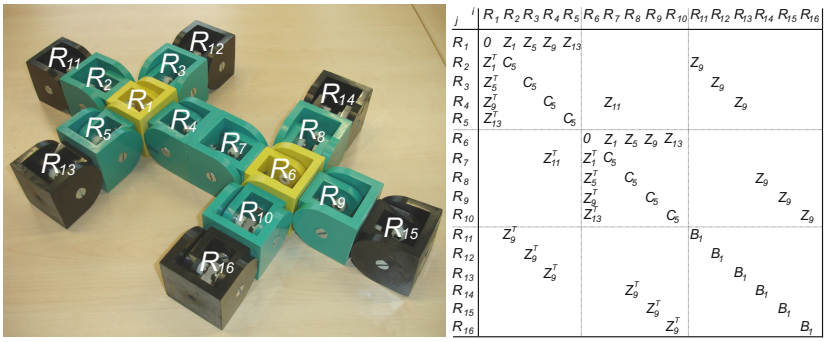


Fig. 3 2x-Centipede ("dog") and its symbolic description, obtained as a combination of two extended crosses (from [7])

Generally, the connectivity means the number of elements, connected to each of modules. For example, the central element of the cross has the connectivity 4 (modules connected from each side). Connectivity constrains the number of connections and can be effectively utilized in a description of topologies. When c_i is the connectivity of the i-element, where i goes from 1 to n (n is the number of robots in the topology; in contrast N is a total number of robots), the topology Φ can be described as $n+1$ set $(c_1, c_2, ..., c_n, c_t)$, c_t is a total number of connections in the topology with n robots. Each of c_i varies between 1 and 2 for active wheel and between 1 and 4 for scout and backbone robots from Fig. 1. In general case, max. of c_i is equal to the maximal connectivity of the platform. All c_i are re-ordered from c_{max} to c_{min} so that the first element c is always that one, which has a maximal degree of connectivity. The topology Φ can be described as

$$\Phi = \{c_{max}, c_{max-1} ..., c_{min+1}, c_{min}, c_t\}, c_i \in \{1,2,3,4\}. \quad (1)$$

Several examples of Φ for $n = 5-7$ are shown in Table 1.

The description, defined by (1) has different topological properties, whose analysis oversteps boundaries of this work. Generally, there are basis topologies, which are unique, provided the topology is coherent (coherent topology = no disconnected nodes). For example, the first row in Table 1 demonstrates disconnected topologies. To eliminate disconnected topologies, a coherency constraint has to be integrated into CSP/COP solver. Basic topologies can be perturbed by one or several modules, this increases n and c_t. Such perturbed topologies are not unique. One of possible ways to deal with perturbed topologies is indicated in [7], in this work we limit ourselves only to basic (non-perturbed) topologies.

Integration of Kinematic Constraints into Self-assembling. Topology Φ defined by (1) creates connections, which are invariant to robot's IDs. To integrate kinematics into topology, Φ should be supplemented with a functional description: it means to involve the desired degrees of freedom φ_i for a particular connection. The degrees of freedom between robots $R_k : R_p$ depends on both R_k and R_p, i.e. we can encounter the situation when both are relevant, one of them is relevant and none of

Table 1 Examples of different topologies for $n = 5, 6, 7$, described through connectivity constraints, n is the number of robots, c_i is the connectivity and c_t is a total comber of connections

n	c_i	c_t	Example	n	c_i	c_t	Example	n	c_i	c_t	Example
5	2,1,1,1,1	3		6	2,2,1,1,1,1	4		7	2,2,2,1,1,1,1	5	
5	4,1,1,1,1	4		6	4,2,1,1,1,1	5		7	4,2,2,1,1,1,1	6	
	——-			6	3,3,1,1,1,1	5		7	3,3,2,1,1,1,1	6	
	——-				——-			7	4,3,1,1,1,1,1	6	
5	3,2,1,1,1	4		6	3,2,2,1,1,1	5		7	3,2,2,2,1,1,1	6	
	——-			6	4,3,2,1,1,1	6		7	4,3,2,2,1,1,1	7	
	——-				——-			7	4,4,2,1,1,1,1	7	
5	3,3,2,1,1	5		6	3,3,2,2,1,1	6		7	3,3,2,2,2,1,1	7	
	——-			6	4,4,2,2,1,1	7		7	4,4,2,2,2,1,1	8	
	——-				——-			7	4,4,3,2,1,1,1	8	
5	4,2,2,1,1	5		6	4,2,2,2,1,1	6		7	4,2,2,2,2,1,1	7	
5	3,2,2,2,1	5		6	3,2,2,2,2,1	6		7	3,2,2,2,2,2,1	7	
5	2,2,2,1,1	4		6	2,2,2,2,1,1	5		7	2,2,2,2,2,1,1	6	
5	2,2,2,2,2	5		6	2,2,2,2,2,2	6		7	2,2,2,2,2,2,2	7	

Table 2 Combination of different types of connections between $R_k : R_p$, x means "*any type of connection*", A - active wheel, S - scout, B - backbone robots, indexes point to corresponding DoF

Number (φ)	Type	Number (φ)	Type	Number (φ)	Type	Number (φ)	Type
0	x:x	5	$B_2 : x$	10	$A : x$	15	$S_1 : S_1$
1	$B_1 : B_1$	6	$B_1 : S_1$	11	$A : B_1$	16	$S_1 : S_2$
2	$B_1 : B_2$	7	$B_1 : S_2$	12	$A : B_2$	17	$S_2 : S_2$
3	$B_2 : B_2$	8	$B_2 : S_1$	13	$A : S_1$	18	$S_1 : x$
4	$B_1 : x$	9	$B_2 : S_2$	14	$A : S_2$	19	$S_2 : x$

them are relevant. For example, in the configuration shown in Fig. 4, the functional requirement imposed on all connections is "$A_x : x$", where x means "*any*". Table 2 introduces φ_i for connections, shown in Fig. 1. Since each node has max. four connections (i.e. in general case different φ_i), the functional topology should include all of them. We use the agreement, that when only one φ is specified for a connectivity, it means $\varphi_i = \varphi$. Now we can generalize $\boldsymbol{\Phi}$ from (1):

$$\boldsymbol{\Phi} = ((c_{max} : \{\varphi\}_{max}), (c_{max-1} : \{\varphi\}_{max-1}), ..., (c_{min} : \{\varphi\}_{min}), c_t) \qquad (2)$$

To give an example of this description, we consider the simple organism from Fig. 4. It has three robots $n = 3$, the maximal connectivity is $c_{max} = 2$ (two modules are connected to the active wheel), all other connectivities are 1 (one mode from each side), the total number of connections $c_t = 2$, i.e. $\Phi = (2,1,1,2)$. Functionality is described as $A : x$ (10 from the Table 2) for the maximal connectivity (active wheel connected from each side to *any* module) and "x:x", "x:x" (i.e. 0 from the Table 2) for other connectivities (any module can connect to the active wheel), i.e. $\Phi = ((2 : 10),(1 : 0),(1 : 0),2)$. This description is unique for each topology and kinematics, taking into account other constraints, mentioned in the previous section. Kinematic constraints are involved into calculation of the cost function (4), i.e. there are penalties, when a robot does not satisfy the functional requirements.

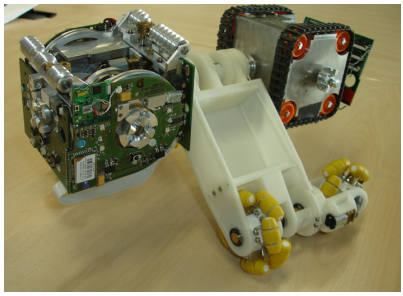


Fig. 4 Simple organism, defined by topology $\Phi = ((2 : 10),(1 : 0),(1 : 0),2)$, see explanation in text

3 Formulation of CSP/COP, Cost Functions and Scalability Issues

The constraint-based approach assumes, that basic topologies with corresponding kinematics are evolved or designed off-board/off-line. All of them, as well as corresponding control procedures are stored on-board. Robots during self-assembling decide which of these topologies is the most optimal one to the given environmental conditions and self-assemble into scalable versions of this configuration. There are two challenges here. Firstly, the decision process is distributed and based only on local sensor data, i.e. it should be stable to noisy and incompetence sensor information. Secondly, only the optimal topology and one scalability approach should be selected (which optimizes a cost function), i.e. distributed optimization and decision making processes should be integrated. As mentioned, these challenges are approached in the CSP/COP way.

CSP is a useful way of solving combinatorial problems, when constraints can essentially limit the search space, see e.g. [8]. There are several CSP solvers, one of them is based on a linear programming (LP). LP is formulated to optimize the linear objective function $\Theta = \underline{s}^T \underline{x}$, where $\underline{s}$ is the vector of costs and $\underline{x} = (x_1, x_2, ...x_m)^T$ is a vector of variables, which are bounded by 0 and 1. LP is constrained as follows

$$\underline{\underline{A}}\underline{x} = \underline{b}, \quad x_i \in \{0, ..., 1\}, \tag{3}$$

where $\underline{\underline{A}}$ is a matrix and $\underline{b}$ is a vector of numerical coefficients, which form m linear equations (in general case inequalities). In this form it is known as the integer program. Finally, by solving (3), all variables x_i take "0" or "1" so that to optimize $\underline{s}^T \underline{x}$.

First of all, we need to defined the objective function Θ, which is specified by s_i. When n robots are involved into some topology Φ, the variables $\underline{x}$ represent all possible bilateral connections between there robots. The vector of variables has m components $m = \frac{n!}{(n-2)!2!}$.There are several different Θ, in the experiments we used

$$s_i = f_i(R_k : R_p) = D(R_k : R_p) + F(R_k : R_p), k, p = 1, ..., n; k \neq p, i = 1, ..., m \quad (4)$$

where $D(R_k : R_p)$ is a distance between neighbor R_k and R_p, $F(R_k : R_p)$ satisfaction of functional constraints ("0" when satisfied or "$> maxD(R_k : R_p)$" when not satisfied); all of them are estimated only between locally visible robots R_k and R_p.

Now $\underline{\underline{A}}$ and $\underline{b}$ in (3) have to be defined; they reflect the connectivity constraints of the corresponding topology. As mentioned, all c_i are disconnected from robots, i.e. we have to map the set of c_i to all possible combinations between these robots

$$(c_{max}, c_{max-1}, ..., c_{min}) \rightarrow Permutation(R_1, R_2, ..., R_n). \quad (5)$$

Since the number of permutations is equal to $n!$, computational power of the most of microprocessors allows computation for n below 10 closely to real time. This is more than enough for a large diversity of cores (see Table 1), complex topologies are created through scalability. Since the variable x_i points to connections between robots, defined by (5), the vector $\underline{b}$ is equal to the set of c_i in the order from c_{max} to c_{min} and the matrix $\underline{\underline{A}}$ creates corresponding placeholders (see the example below).

To exemplify the LP solver of CSP, we assume that $N = 5$ robots $(R_1, R_2 ... R_5)$ are positioned on the surface. The costs of connections between robots are written into the vector $\underline{s}$

$$\underline{s} = (R_1 : R_2, R_1 : R_3, R_1 : R_4, R_1 : R_5, R_2 : R_3, R_2 : R_4, R_2 : R_5, R_3 : R_4, R_3 : R_5, R_4 : R_5)^T$$

where $R_1 : R_2$ means a placeholder for the corresponding function $f_i(R_p, R_k)$ defined by (4), for $n = 5, m = 10$.

In a particular example, we set $\underline{c}$=(35,40,80,36,41,42,31,32,55, 60). Thus, m variables x_i correspond to costs of connections $R_k : R_p$, where $k, p = 1, ..., n$ and $k \neq p$. We consider the topology, defined by the connectivity $\underline{C} = (3, 2, 1, 1, 1, 4)$. Linear constraints for the mentioned case are defined as shown in Fig. 5. Here $c_{max} - c_{min}$ define the connectivity of the $R_1 - R_5$ and c_t defines the total number of connections in this group. The defined $\underline{\underline{A}}$, $\underline{b}$ and $\underline{c}$ allow us to find a minimal cost for connections between $R_1 - R_5$ only for one case, namely when the connectivity vector $(c_{max}, c_{max-1}, ..., c_{min})$ is assigned to the vector of robots in this order $(R_1, R_2, R_3, R_4, R_5)$, i.e. the first robot has a maximal connectivity. We have to assume that all robots from R_1 to R_5 can have c_{mac} and all other connectivities. In other words, the connectivity vector $\underline{C}$ should be assigned to each of the

$$\underline{\underline{A}} = \begin{pmatrix} 1&1&1&1&0&0&0&0&0&0 \\ 1&0&0&0&1&1&1&0&0&0 \\ 0&1&0&0&1&0&0&1&1&0 \\ 0&0&1&0&0&1&0&0&1&0&1 \\ 0&0&0&1&0&0&1&0&1&1 \\ 1&1&1&1&1&1&1&1&1&1 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0 \end{pmatrix}, \underline{b} = \begin{pmatrix} c_{max} \\ c_{max-1} \\ c_{max-2} \\ c_{min+1} \\ c_{min} \\ c_t \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad or \quad \underline{b} = \begin{pmatrix} 3 \\ 2 \\ 1 \\ 1 \\ 1 \\ 4 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

Fig. 5 $\underline{\underline{A}}$ and $\underline{b}$ for the introduced example

permutation sets $R_1, R_2, ..., R_n$ (for $n = 5$ there are 120 permutations of $R_1, ..., R_5$). For the mentioned example with 5 robots, the minimal cost $\underline{s}^T \underline{x} = 139$ is achieved for the connections $(R_2 : R_3, R_2 : R_1, R_2 : R_5, R_3 : R_4)$, i.e. $\underline{x} = (1,0,0,0,1,0,1,1,0,0)$.

COP. The CSP solver delivers m solutions, which satisfy connectivity constraints and are optimal for the cost function $\underline{s}^T \underline{x}$ for each set from (5). However, not all of them satisfy the set of constraints (e.g. coherency constraints). COP solver goes through all m solutions and eliminates those which do not satisfy the rest of constraints. Finally, the solution with a minimal cost is delivered as an output.

Scalability. Scalability addresses the relation between n and N (n is the number of robots in the topology, N is a common number of robot). There are different possibilities when N is increasing:

(1) for $N = xn, x = 1, 2, 3, ...$, the topology with n robots can be replicated x times. Each of these new topologies is an independent structure. This is the simplest form of scalability, which can be denoted as the behavioral scalability.

(2) x topologies from the previous case can joint into one common structure. This is typically segmented body construction, where n robots within one segment are repeated x times. This is the structural scalability.

(3) the robots from $N \; mod \; n > 0$ cannot create a new topology. These robots are still useful for the already existing topology, as e.g. energy reserve, so these robots can perturb the topology Φ, this is the perturbational scalability.

(4) finally, $N \; mod \; n > 0$ robots are not aggregating with any other structures, they build a "reserve" for e.g. self-repairing.

For each topology, a corresponding scalability class has to be defined. For this work we use the scalability class (1) and (4), i.e. there are (int)N/n cores, remaining robots are not connected. Algorithm for CSP/COP solver is shown in Fig. 6. The core of

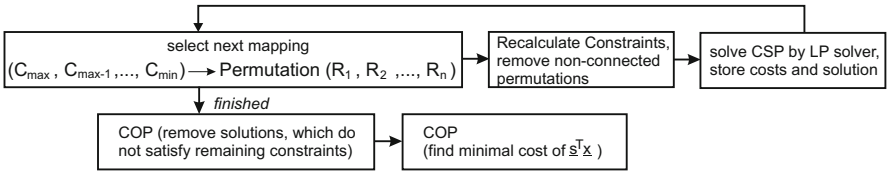


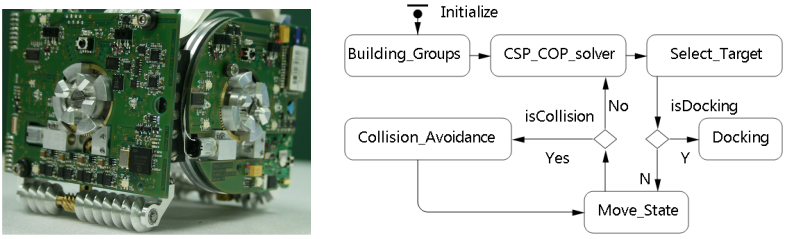
Fig. 6 Algorithm for CSP/COP solver

this algorithm is the LP solver, which cyclically takes one permutation from the set and delivers optimal connections for the given connectivity. All these solutions are stored and later used by COP solves to eliminate non-consisted solutions and to find the minimal one.

4 Implementation and Experiments

For implementation of LP solver for CSP, we used lp_solve 5.5 routine (see lp-solve.sourceforge.net) of Mixed Integer Linear Programming solver, which is under

the GNU lesser general public license and is available in several programming languages (C++ version is used for real robots, Java version is used for simulation). Real robots use Blackfin double core as the main CPU (in each module) with 64 Mb SDRAM on board. The implementation on the real platform (see Fig. 7(a)) was intended to test computational properties as well as to estimate the level of distortion in creating the objective function Θ. Since currently there are not enough robots for testing scalability, several experiments are performed in simulation, which is done in AnyLogic (with Java version of lp_solve and the same algorithm). Tests are performed with two topologies: Φ_1=((3:4,5),(2:4),(1:4),(1:4),(1:4),4), which is shown in Fig. 2 and Φ_2=((2:4),(2:4),(2:4),(1:4),(1:4),4) (a snake of 5 robots).



(a) (b)

Fig. 7 **(a)** Prototype of the reconfigurable module used for testing the objective function Θ; **(b)** Sketch of behavioral algorithm for self-assembling (autonomy cycle)

The behavioral algorithm is sketched in Fig. 7(b). First of all, a robot collects data about availability of other robots and their functionality. This is done through ZigBee communication channel and allows defining N and functional constraints φ_i. For a temporal identification of robots, ZigBee identification code is used. The ZigBee channel does not provide distances and orientation; this is achieved through sensor-fusion level of local IR-based proximity sensors with color sensor and vision-based data. Collision avoidance uses 8-directional force-based model with a global gradient, docking is performed when a robot has a corresponding position and angle (i.e. specific routines control docking approach).

Synchronous and asynchronous updates. All robots start their CSP/COP solvers independently from each other. Since the original cost vector is the same in all robots, all initial solutions are consistent. Each 10 iterations of the autonomy cycle, a robot updates the cost vector and starts the CSP/COP solver again. This new solution can deliver a new partner for docking (when such a solution is more efficient than the original one). Fig. 8 demonstrates the number of partner's changes during one self-assembling run. Since a duration of one

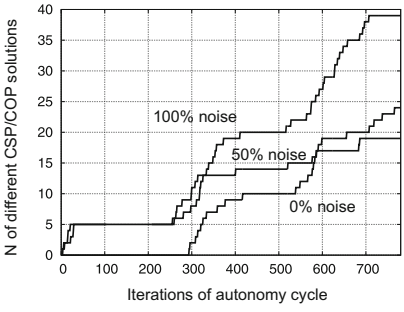


Fig. 8 Number of different CSP/COP solutions in relation to the level of sensor noise

autonomy cycle varies from robot to robot, all next solutions use cost vectors with different time stamps. Such an asynchronous update CSP/COP data can lead to loss of consistency in solutions. There are several methods to keep data consistent even for asynchronous updates, for example, when one robot receive a new assigned partner for docking, this triggers all robots in the group to update CSP/COP solutions.

Noise of sensor data. Non-accuracy of reflective IR sensors, out-of-focus images from cameras, wrong identification of robots are sources of sensor noise. Overview of different sensors and their properties is given in [10]; normally the level of noise increases towards boundaries of perception range. To test a stability of this approach, noise was added to sensor data (as $c_i \pm max.(c_i/2)$ for 100% of noise). Fig. 8 shows the number of different solutions delivered by CSP/COP solver at 100%, 50% and 0% of noise. Generally, noise in sensor data does not change the self-assembling behavior, however triggers more frequent solutions by the solver.

Self-assembling with a large perception radius. The Fig. 9 plots the sum of elements in the cost vector $\sum_i s_i$, when a half of the whole arena is visible to robots, i.e. a robot in the middle of arena can perceive all other robots. The common cost function as well as particular cost functions in each group are monotonically decreased

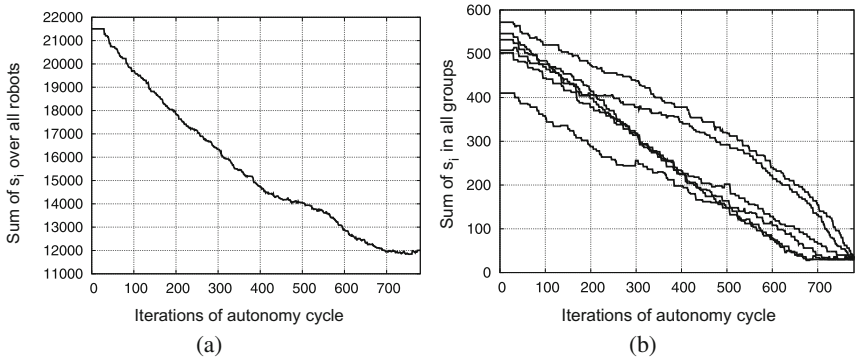
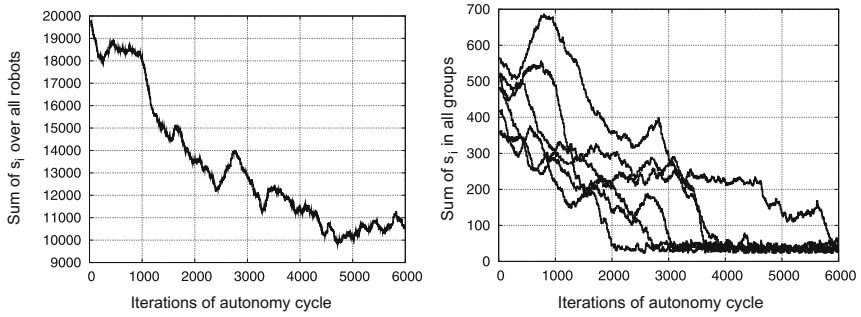


Fig. 9 Assembling of Φ_1 at $N = 30, n = 5$, behavioral scalability is utilized (i.e. 6 groups of 5 robots). Shown are $\sum_i s_i$, calculated for **(a)** all robots in the arena; **(b)** each group of robots.

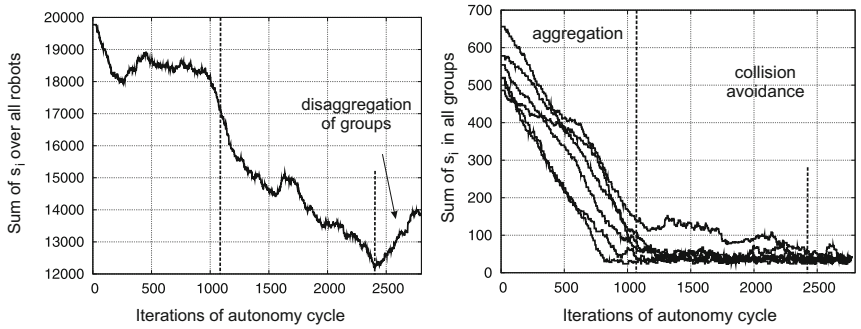
during the self-assembling process. Fluctuation of the function can be explained by collision avoidance behavior and by finding a final alignment during the docking.

Self-assembling with reduced perception radius and noisy sensor data. The perception radius was set to 8-10 body lengths of a robot (what approximately corresponds to the data from camera). Robots outside the visibility radius receive a large constant value in the cost vector and move randomly in the arena. As soon as a robot became within the perception radius, CSP/COS solver starts anew and recalculates the solution. Fig. 10 shows the common and particular objective functions. Comparing to a large visibility radius, the self-assembling here takes almost 10 times longer. Such a long convergence time can be explained by the random



(a) (b)

Fig. 10 Self-assembling with the same parameters as in Fig. 9, perception radius is limited to 10 body length of a robot. Assembling is finished within 6000 iterations of the autonomy cycle (calculated as a sum over all robots).



(a) (b)

Fig. 11 The same case as in Fig. 10, simple two steps aggregation strategy is used. Assembling is finished within 2700 iterations of the autonomy cycle.

motion of those robots, which are outside of the perception radius and so not involved into self-assembling. When these robots increase compactness of the group, this will essentially improve the efficiency of the approach without making it more complex. Fig. 11 demonstrates the objective functions for the strategy, when robots outside of the perception radius move first to the middle of robot arena. All robots get quickly visible to each other, however this creates more stronger collision avoidance in the groups and robots need more time to resolve collision problems. Despite simplicity and drawback of this strategy, it allows improving the efficiency more than twice.

5 Conclusions

This paper describes the constraint-based self-assembling strategy, which used CSP/COP solver with LP core. Due to connectivity and functional constraints, this

approach is very useful for modules with different geometry and functionality, i.e. for heterogeneous reconfigurable robots. Since kinematic chains are directly involved into self-assembled structures, self-assembled organisms immediately after aggregation are ready for performing locomotive tasks.

There are several observations for this approach. First of all, the constraint-based topological description is efficient for basic and symmetric topologies. To define perturbations and scalability, additional specifications are necessary. This can be done by using a generator-based approach [7] or by introducing compact explicit descriptors. Secondly, in practical situations the CSP/COP solver can run only once, when all components of the objective function Θ are known. Possible small non-optimality of solutions can be ignored by the reason of saving computational power. Moreover, very restrictive formulation of a heterogeneous topology (e.g. only with specific modules) leads to deadlocks when such modules are not available. It is generally recommended to use "A:x", "S:x" or "B:x" kind of functional descriptions. Finally, a combination of low-dimensional assembling cores and scalability management enables an efficient management of high-dimensional topologies; in the demonstrated example the problem of 30 robots was efficiently solved within a few seconds by on-board microprocessors.

Limited perception radius of robots has an essential impact on the performance of this approach, drop of efficiency lies between 4 and 10 times. However, nether noise nor a small perception radius stops the self-assembling. By using dedicated algorithms for increasing compactness, the performance can be improved; this as well as performing experiments with 30 real heterogeneous robots represents future works.

References

1. Brener, N., Ben Amar, F., Bidaud, P.: Designing modular lattice systems with chiral space groups. Int. J. Rob. Res. 27(3-4), 279–297 (2008), doi:http://dx.doi.org/10.1177/0278364908089349
2. Castano, A., Will, P.: Representing and discovering the configuration of CONRO robots. In: Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA 2001), vol. 4, pp. 3503–3509. IEEE (2001)
3. Chiang, C.J., Chirikjian, G.: Modular robot motion planning using similarity metrics. Auton. Robots 10(1), 91–106 (2001), doi:http://dx.doi.org/10.1023/A:1026552720914
4. Christensen, A., O'Grady, R., Dorigo, M.: Swarmorph-script: A language for arbitrary morphology generation in self-assembling robots. Swarm Intelligence (2), 143–165 (2008)
5. Davis, P.: Circulant matrices. John Willey & Sons (1979)
6. Kernbach, S.: Structural Self-organization in Multi-Agents and Multi-Robotic Systems. Logos Verlag, Berlin (2008)
7. Kernbach, S.: From robot swarm to artificial organisms: Self-organization of structures, adaptivity and self-development. In: Levi, P., Kernbach, S. (eds.) Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution, pp. 5–25. Springer, Heidelberg (2010)
8. Kornienko, S., Kornienko, O., Priese, J.: Application of multi-agent planning to the assignment problem. Computers in Industry 54(3), 273–290 (2004)

9. Lau, H.Y.K., Ko, A.W.Y., Lau, T.L.: The design of a representation and analysis method for modular self-reconfigurable robots. Robot. Comput.-Integr. Manuf. 24(2), 258–269 (2008), doi:http://dx.doi.org/10.1016/j.rcim.2006.11.003
10. Levi, P., Kernbach, S. (eds.): Symbiotic Multi-Robot Organisms. Cognitive Systems Monographs, vol. 7. Springer, Heidelberg (2010)
11. Nolfi, S., Floreano, D.: Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines. The MIT Press, Cambridge (2000)
12. Salemi, B., Shen, W.M.: Distributed behavior collaboration for self-reconfigurable robots. In: Proc. of the IEEE International Conference on Robotics and Automation (ICRA 2004), New Orleans, USA, pp. 4178–4183 (2004)

A New Graph Signature Calculation Method Based on Power Centrality for Modular Robots

Keyvan Golestan, Masoud Asadpour, and Hadi Moradi

Abstract. Graph signature is a fast isomorphism test that is used in self-reconfiguration planning of modular robots. In case of dealing with homomorphic modules, the required time to calculate the signature grows exponentially with the number of symmetry lines. We tackle this problem by introducing an isomorphism-invariant signature calculation method, which is based on the power centrality of nodes. We also introduce a new sample-based search method. Simulation results show the new method finds better solutions in a significantly shorter time.

1 Introduction

Modular robots are composed of some relatively simplified and usually small-sized robotic parts called *modules*. The modularity comes with properties such as versatility, robustness and low cost. The modules can be connected in different ways (either manually or automatically) thus creating different configurations.

Based on the connection structure and the movement of modules, they are classified into two main categories. *Lattice-type* modules use cluster-flow to move and reconfigure. Crystalline [1], ATRON [2], Telecube [3] and Molecule [4] are examples of this kind. *Chain-type* modules form chain structures and have joints that help them move without necessarily doing reconfiguration. M-TRAN [5], CONRO [6], Roombot [7], PolyBot [8], YaMoR [9] and SuperBot [10] are examples of this type. Our work is based on chain-type modular robots.

Self-Reconfiguration Planning (SRP) is a task in which an optimal or sub-optimal solution is found for reshaping a modular robot from an initial configuration to a final one. Solution to this task is hard to achieve as the time complexity of planning problem grows exponentially when the number of modules or their degrees of freedom (DOF) increases. In this paper we propose a general framework for SRP to reduce the time complexity specially when dealing with modules

Keyvan Golestan ; Masoud Asadpour ; Hadi Moradi
Faculty of Electrical and Computer Engineering, University of Tehran, Tehran, Iran
e-mail: {kgolestan,asadpour,moradih}@ut.ac.ir

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 505–516.
springerlink.com

with high DOF. This method uses the idea of graph signature introduced by Asadpour et al. [11][12] and improves it by using the concept of power centrality in social networks [13] and establishing a hierarchical graph signature calculation algorithm. A sample-based method for investigating feasible "attach" actions, has also been incorporated with our previous general search to make it faster.

The rest of this paper is organized as follows: In the next section, some previous works on SRP are explained. The third section explains our proposed method. The paper is then finalized by simulation results and conclusions.

2 Background

SRP is one of the most challenging tasks in modular robots. The task is even more difficult in chain-type robots where mechanical limitations come into play. SRP becomes practically intractable when dealing with large number of modules, or modules with high DOF.

Reconfiguration planners are usually based on a guided search strategy and a distance function. Casal and Yim [14], [15] introduced a divide-and-conquer strategy to solve SRP for chain-type robots. The configuration was decomposed into a hierarchy of small sub-structures belonging to a finite set. These sub-structures were non-homomorphic and reconfiguration between them was simple. So the reconfiguration steps could be specified and stored in a look-up table in advance. The reconfiguration was then consisted of an ordered set of pre-defined actions among sub-structures which happen locally.

Hou & Shen [16] presented a distributed reconfiguration method on the unlabeled graph representation of configurations. They did the reconfiguration by first utilizing a distributed comparison to detect substructures in two configurations. Then the reconfiguration was limited only to the modules that indicated difference in topology. Reconfiguration took place by first converting the initial configuration to an intermediate structure and then transforming it to final configuration.

Asadpour et al. [11] proposed a method based on graph theory. They encoded the 3D structure of a configuration by an isomorphism-invariant code, called *signature,* and used *edit distance* based similarity metric as an upper-bound for isomorphism test. The time complexity of their algorithm grows exponentially as the number of modules increases. They improved the method in [12] in order to deal with modules with symmetry. Again the time complexity grows exponentially as the number of modules increased. Another disadvantage of their method is where they are looking for feasible attach actions; they had to search all possible permutations of joint angles of each module in a configuration. Here, an increase in the DOF of the modules would cause the time complexity of finding feasible attach actions grow exponentially.

3 Our Proposed Method

We represent a configuration by a graph with modules as *nodes* and connection between modules as *edges* (directed for male/female connectors and undirected for genderless ones). Reconfiguration takes place by performing feasible edit actions,

i.e. attach or detach, on initial configuration hoping to make it closer to the final configuration. Among the new unexplored configurations, the *closest* configuration (based on a distance function) to the final one is chosen for further exploration. This process continues until a path of *transitions* from the initial graph to the final graph is found. The network of transitions between configurations in which, nodes are configurations and edges are edit actions, is called a *transition graph*.

The isomorphism checking of graphs is not yet proved to be NP-Complete or not [17] except some special cases like graphs with bounded degrees [18] or ordered graphs [19]. Asadpour et al. [11] compute a unique identifier, called *graph signature*, for a configuration using properties of ordered graphs. An ordered graph is a graph whose edges have a specific order. So they adopted an edge labeling method to assign a unique identifier to each edge. The labeled graph is trivially transformable to an ordered graph by sorting the out-edges of the vertices in lexicographic order. It is shown in [19] that Isomorphism test on such a graph takes quadratic time in worst-case in terms of the number of nodes.

3.1 Labeling the Graph

Edge labels include information about how modules are connected to each other. A labeling strategy may be as follows: First, the connectors of a module are indexed. The indexing order is arbitrary, but should be the same for all modules. Fig.1.A and B show an indexing order for SuperBot modules.

Relative rotation of two modules around their connection point should also be encoded. This is done by assigning an index to each relative rotation. For instance, if only multiples of 90 is allowed, the 90° angle between the modules in Fig.1-C would have index 1 (index 0 for 0°, index 1 for 90°, … , index 3 for 270°).

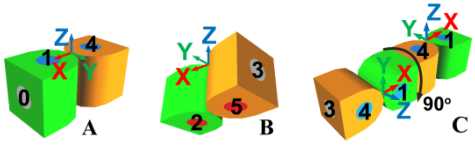


Fig. 1 Connector indexing of a SuberBot module. (A) Top view (B) Bottom view (C) Relative rotation at the connection point is 90°, so rotation index is 1.

Putting connector and relative rotation indices together, connection of two modules can be labeled as [11]:

$$l_{ij} = |C| \, |R| \, c_{ij} + |R| \, c_{ji} + r_{ij} \tag{1}$$

where c_{ij} is the index of the connector of module i which is connected to module j, $|C|$ is the total number of connectors, $|R|$ is the number of possible relative rotations, and r_{ij} is the relative rotation of module i with respect to module j. For SuperBot, $|C|$ is 6 and $|R|$ is 4. This way, each connection gets a unique label and an ordering can be imposed on the edges. Fig. 2 shows two different configurations and their graph representations.

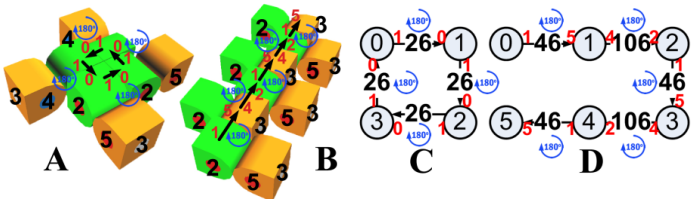


Fig. 2 SuperBots configured as (A) 4-Module quadruped (B) 6-Module climber. (C) and (D) Their corresponding labeled graphs. Red indices belong to the used connectors.

3.2 Graph Signature

The term *Graph Signature* proposed by Asadpour et al. [11] is an isomorphism invariant property of the configuration graph that encodes the 3D structure of a modular robot. It is inspired by the way we help somebody who is searching for an address (e.g. go straight, turn left at the junction). A configuration is like a city whose entire map should be encoded in a string. The code specifies for a tourist (here a planner), what would he/she see when walking along the streets? A big problem is to specify (in an isomorphism invariant way) the place he should start walking. That is why Asadpour et al. [11] had to try all possible start places (i.e. all nodes) and select in some way a unique code among them (e.g. by sorting).

The graph signature is created by performing a modified DFS on an ordered graph and recording what is seen meanwhile, in this way: 1) start from a node (i.e. a module) and record its index; start indexing from one, and increment it upon visiting new unvisited node; 2) If possible traverse the out-edges in the lexicographic order of their labels and record their labels, after that 3) traverse the in-edges (i.e. opposed to their direction) and record the negation of their labels; and finally 4) if no move is possible, back track to the previous node(s). In case of dealing with undirected (i.e. hermaphrodite) edges, the traversal direction is decided once it is first encountered, from the current node towards an unvisited node.

The procedure is repeated for all nodes as start position. Each time a signature is created whom we call a *node signature*. Among all node signatures, the one with maximum lexicographical order is selected as the graph signature. The worst

Table 1 Signature of nodes of the graphs in Fig.2. Graph signature is shown in bold face.

4-Module Quadruped Configuration		**6-Module Climber Configuration**	
Node	Node Signature	Node	Node Signature
0	**(0 26 1) (1 26 2) (2 26 3) (3 26 0)**	0	(0 46 1)(1 106 2)(2 46 3)(3 106 4)(4 46 5)
1	**(0 26 1) (1 26 2) (2 26 3) (3 26 0)**	**1**	**(0 106 1)(1 46 2)(2 106 3)(3 46 4)(0 -46 5)**
2	**(0 26 1) (1 26 2) (2 26 3) (3 26 0)**	2	(0 46 1)(1 106 2)(2 46 3)(0 -106 4)(4 -46 5)
3	**(0 26 1) (1 26 2) (2 26 3) (3 26 0)**	3	(0 106 1)(1 46 2)(0 -46 3)(3 -106 4)(4 -46 5)
		4	(0 46 1)(1 -106 2)(2 -46 3)(3 -106 4)(4 -46 5)
		5	(0 -46 1)(1 -106 2)(2 -46 3)(3 -106 4)(4 -46 5)

time complexity of signature calculation is $O(|V||E|)$ for $|V|$ and $|E|$ as the number of nodes and edges, respectively [11]. Table 1 shows all possible graph signatures for the configurations of Fig.2.

3.2.1 Symmetric Modules

A module is called *homomorphic* or *symmetric* if there exist at least one symmetry line that rotating the module around it produces the same (matchable) 3D shape. Fig. 3 shows some SuperBots that have the same 3D shapes. The number of matchable shapes, whom we call, *symmetry factor*, can be calculated once for each module type through exhaustive search. For instance, M-TRAN modules, if all connectors are genderless, have 3 symmetry lines, and their symmetry factor is 4. The symmetry factor is 8 for SuperBots [10] and 36 for Roombots [7]. This is like *re-indexing* the connectors and acquiring the same shape. Thus, a mapping between connector indices of symmetric shapes could be achieved and saved in a lookup table. Since re-indexing the connectors changes the edge labels, a new graph signature might be achieved.

Fig. 3 (A)-(F) Examples of SuperBots with identical 3D shapes. In B, E and, F, the middle servo rotates 180° (G) Two isomorphic configurations with a module rotated around a symmetry line.

The case of symmetric modules is tackled in [12] by putting some order on the connections. They compute the node signatures by testing all permutations of symmetric positions of each module and choose the one with maximum lexicographical order. The time complexity of signature calculation is $O(|V|^2 + |V| \times S^{|V|})$ for S being the symmetry factor.

3.2.2 The Improved Signature Calculation Algorithm

The inefficiency of signature calculation in symmetric modules returns back to the calculation of multiple node signatures. If we find an isomorphism invariant way to fix the starting node, we could calculate the signature in one run. Here we use a centrality measure from social networks domain called, *power* centrality [13] based on which the most powerful node is selected as the starting node for signature calculation. This measure can also specify the priority of visit for nodes in case of tallies. Following, the steps of the proposed method are explained:

Step 1: Isomorphism-invariant node prioritization based on power centrality

Based on the power centrality [13], (social) power of a node recursively depends on the sum of the power of its friends with attenuation factor $0 \leq \beta \leq 1$:

$$C_p(n_i) = \sum_j a_{ij}(\beta . C_p(n_j)) \tag{2}$$

where $C_p(n_k)$ is the power of node k and a_{xy} is 1 if x and y are friends (or neighbors) and 0 otherwise. If we start from a positive initial value for $C_p(n_k)$, $k=1…n$ and recursively calculate (2) for all nodes, power centralities finally converge if $|\beta|<1/\lambda_{max}$ where λ_{max} is the highest eigenvector of the adjacency matrix of the graph[13]. The number of required iterations can go up to the diameter of the graph that is at most $|V|-1$.

It is evident that for isomorphic configurations the same power centralities are gained for corresponding nodes. In case of generating different power centralities for some nodes, we can surely say the configurations are different. However, the reverse is not always true i.e. if power centralities are the same we cannot surely say the configurations are isomorphic.

We hope the most powerful node is unique such that it could be selected as the starting node, otherwise we have to run the signature calculation once for each of the most powerful nodes. Table 2 shows the converged power centralities of graphs presented in Fig. 2.

Table 2 Normalized initial and converged node power centralities of the graphs in Fig. 2C-D

4-Module quadruped configuration			**6-Module climber configuration**		
Node	Initial power	Converged power	Node	Initial power	Converged power
0	**0.25**	**0.25**	0	0.1073	0.1038
1	**0.25**	**0.25**	1	0.1964	0.1804
2	**0.25**	**0.25**	**2**	**0.1964**	**0.2157**
3	**0.25**	**0.25**	**3**	**0.1964**	**0.2157**
			4	0.1964	0.1804
			5	0.1073	0.1038

Choosing the initial values for power centralities is very important. A popular initial value is the nodal degree [13], which is not appropriate here because the degree keeps only the information about the number of neighbors, but not how they are connected. Instead, we assign a value to each node that is calculated by summing the labels of the edges connected to a specific node. We call it the *Vicinity Value*. The *maximum* possible vicinity values of nodes are used as their initial powers. If we assume the modules are genderless, the vicinity value of node i is:

$$v(i) = \sum_{j \in V} a_{ij} l_{ij} \tag{3}$$

where l_{ij} is the label of the edge between nodes i and j. Based on (1) and (3) the maximum vicinity value is:

$$v^*(i) = \sum_{j \in V} a_{ij} l_{ij} = \sum_{j \in V} a_{ij} (|C| \, |R| \, c_{ij}^* + |R| \, c_{ji}^* + r_{ij}^*) \tag{1}$$

where c_{ij}^* and c_{ji}^* are the connector indices that maximizes the vicinity value, and are obtained by re-indexing the modules i and then j; and r_{ij}^* is the maximum rotation index between i and j. Equation (4) can be written as:

$$v^*(i) = \sum_{j \in V} a_{ij} (|C| \, |R| \, c_{ij}^* + r_{ij}^*) + |R| \sum_{j \in V} a_{ij} (c_{ji}^*) \tag{2}$$

Maximizing both terms would maximize the whole expression. The first term, which we call *Node Value,* only depends on the module i and can be maximized using the equation below:

$$n^*(i) = \max_{s \in S} \{ \sum_{j \in V} a_{ij} (|C| \, |R| \, s(c_{ij}) + s(r_{ij})) \} \tag{3}$$

where S is the set of possible connector mappings (|S| being the symmetry factor), and $s(c_{ij})$ and $s(r_{ij})$ are the connector and rotation indices, respectively, provided by the mapping s. We call this process, *Node Value Maximization Procedure (NVMP)*. The neighboring nodes can independently run NVMP and maximize the second term of (5) and finally the sum of two terms. The whole process is called *Vicinity Value Maximization Procedure (VVMP)*. Fig.4 shows an example of how NVMP and VVMP are performed using equations 3, 5 and 6. If two nodes are connected in more than one way, VVMP selects the mapping that leads to maximum vicinity value.

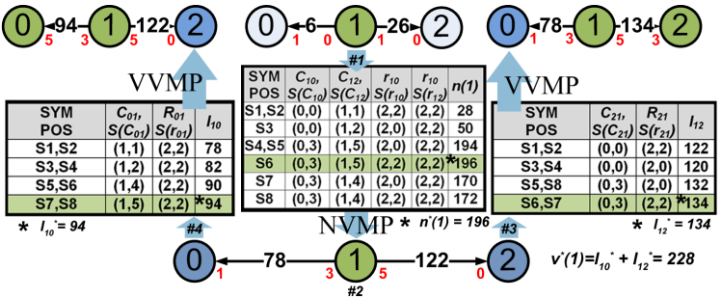


Fig. 4 Finding the maximum vicinity value of node 1 for 4-Module quadruped graph shown in Fig. 2C. (#1 & #2) NVMP is done by node 1 using the Eq. 6 for each symmetric position S1-S8. (#2 & #3) VVMP is done independently by each neighbor using Eq. 6 for each symmetric position S1-S8 concerning the rule that common edges with node 1 can only get increased.

Step 2: Hierarchical graph re-indexing

This procedure is done once for the node with maximum power centrality, which we call the *master node*. The master node performs VVMP followed by re-indexing according to the mapping provided by VVMP. Then, the edges incident to the master node with their new labels are frozen so that no other node could modify their labels. This process is repeated for a neighbor of the master node and each time some edges are frozen. Neighbors are prioritized according to their vicinity value. When all edges are frozen, the re-indexed graph is ready for signature calculation. Fig. 5 shows an example of re-indexing.

It should also be noted that during NVMP some cases happen where for two different symmetric positions, calculated node values of a specific module are equal and it cannot decide which symmetric position *s* to choose for re-indexing. In such cases we simply look at the power centralities of the nodes that the connector tends to connect, and choose the symmetric position in which the connectors with higher indices are going to connect to nodes with higher power centralities. We think in cases of equal power centralities choosing either one of the nodes would not change the overall outcome (not proved yet, left for future work).

Fig. 5 Hierarchical re-indexing of 6-Module climber configuration graph. Module 3, the master starts VVMP and freezes its edges, the process is continued by 2, 4, 1, 5, and 0.

Step 3: Graph signature calculation

The graph signature is generated by performing a DFS starting from the master node on the frozen graph of step two. If more than one master node (and consequently more than one re-indexed graph exists), graph signature is generated for each case by starting from the corresponding nodes and the one with maximum lexicographical order is chosen. It is easy to verify that the worst time complexity of this new method is $O(S \times |V|^2)$ which is much better than [12].

3.2.3 Improvement of Searching for Feasible Attach Actions

As mentioned earlier, reconfiguration takes place by performing feasible edit actions, i.e. attach/detach, on initial configuration hoping to make it closer to final one. Finding feasible detach actions is easy; it can be done by searching for modules that form loop (so they can be disconnected from either side). This can at worst be done in $O(|E|^2|V|)$. However, computing all possible attach actions, especially in case of modules with high DOF, is very difficult. Asadpour et al. [11] [12] used a brute force approach and checked all permutations of discretized servo movements for possible attach action i.e. $O(p^{|V|\times|M|})$, for p being the number of discretized servo movements and $|M|$ being the DOF of modules (so the dimension of the *joint state space* is $|V|\times|M|$).

We tackle this problem by applying a sample based method called Rapidly-exploring Random Trees (RRT) [20]. The main property of RRT is its tendency to search unexplored regions of the space, while insuring that the whole space will be explored if sampling runs for a long time. Here, instead of finding all feasible attach actions which takes a lot of time, the sampling is continued until either "enough" attach actions are found or no part of space remains unexplored.

4 Simulation Results

Our method is tested on simulated M-TRANs and SuperBots, each having 6 genderless connectors, and 2 and 3 rotational servos respectively. Fig.2A, B and Fig.6 depict the configurations we study. The SRP tasks for MTRAN are Quadruped-to-snake with 4 and 8 modules. Then the scalability of our method is verified on stool-to-snake reconfiguration (with 12 M-TRAN). The SRP tasks on SuperBots are line-to-climber configuration with 4, 6, and 8 modules. Simulations are repeated on 30 random seeds and are continued until 30 solutions are found.

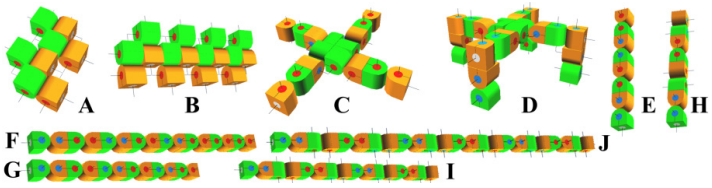


Fig. 6 (A, B) 4 and 8-module climber (C) 8-module quadruped (D) 12-module stool (E, F, G) 4, 6 and 8-module line (H, I, J) 4, 8 and 12-module snake

4.1 Reconfiguration with M-TRAN

Fig.7 (left) shows the number of graphs examined before finding the first and the best solutions of 4-module quadruped-to-snake reconfiguration. In about 70% of simulations the first solution is found before less than 500 graphs are examined, in less than 5 seconds. Moreover the first solution is always the best solution with 9 attach/detach actions. This is much better than Asadpour et al. [12] where the first solution was among 4,000 visited graphs and only about 17% of best solutions were within the first 2,500 examined graphs.

Fig.7 (right) shows the results for 8-module quadruped-to-snake task. The best solution of this task has 7 actions which is equal to results of [12]. The first solution is found by visiting 7000 or fewer graphs in 50% of simulations in less than 300 seconds. The best solution is found in 7 simulations after visiting 15000 or fewer graphs in less than 450 seconds. This is almost similar to [12], but is gained in significantly shorter time.

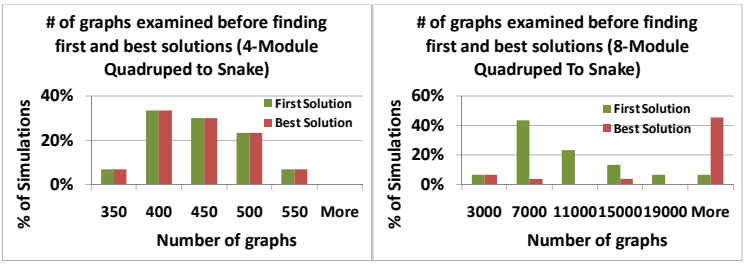


Fig. 7 Reconfiguration result for (Left) 4-Module quadruped to snake and (Right) 8-Module quadruped to snake

To test the scalability of our framework, we solved the stool-to-snake reconfiguration with 12 M-TRAN modules. The best solution that has 27 actions is found by examining around 120000 graphs in about 85 minutes.

4.2 Reconfiguration with SuperBot

Fig.8 shows the number of graphs examined before finding the first and the best solutions of line-to-climber reconfiguration with SuperBots. Fig.8 (top-left) shows the result for 4-module reconfiguration. It is seen that more than 70% of the first solutions and about 20% of the best solutions are found after visiting 2800 or fewer graphs. The first and best solutions are found in less than 10 seconds.

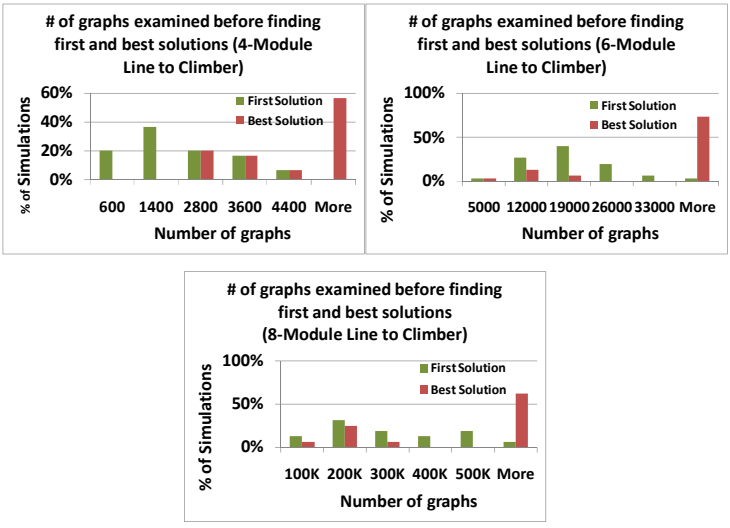


Fig. 8 Result for line-to-climber reconfiguration with (top-left) 4, (top-right) 6 and (bottom) 8 modules

Fig.8 (top-right) shows the results for 6-module reconfiguration. It is seen that more than 50% of the first solutions are found by examining 19000 or fewer graphs in less than 30 seconds. The best solution to this task has 12 actions and is found in about 23% of our simulations.

Fig.8 (bottom) shows the results for 8-module reconfiguration. This is the hardest reconfiguration task in our simulations (the configuration has totally 24 DOFs). About 63% of the times, the first solution is found after examining 300000 or fewer graphs in less than 120 minutes. The best solution that has 24 actions is found only in 34% of simulations always before examining 300000 or fewer graphs in less than 200 minutes.

5 Conclusion

We proposed an isomorphism-invariant graph signature calculation based on power centrality. We could enhance the time complexity of signature calculation to polynomial time even for symmetric modules. We also tackled the problem of finding feasible attach actions by using the sample based RRT method. The results showed an impressive drop in reconfiguration time for both M-TRAN and Super-Bot modules.

As future works, we think finding the feasible attach action by sampling can be improved through parameter tuning. Physical restrictions during reconfiguration should be mentioned and finally the cases where power centralities or vicinity values of some nodes are equal need more investigation.

References

[1] Rus, D., Vona, M.: Crystalline robots: Self-reconfiguration with compressible unit modules. Autonomous Robots 10, 107–124 (2001)

[2] Jørgensen, M., Østergaard, E., Lund, H.: Modular ATRON: Modules for a self-reconfigurable robot. In: 2004 IEEE/RSJ Int. Conf. on Intelligent. Robots & Systems (IROS), pp. 2068–2073 (2004)

[3] Vassilvitskii, S., Kubica, J., Rieffel, E., Suh, J., Yim, M.: On the general reconfiguration problem for expanding cube style modular robots. In: Proceedings - IEEE International Conference on Robotics and Automation, pp. 801–808 (2002)

[4] Kotay, K., Rus, D., Vona, M., McGray, C.: Self-reconfiguring robotic molecule. In: Proceedings - IEEE International Conference on Robotics and Automation, pp. 424–431 (1998)

[5] Murata, S., Yoshida, E., Kamimura, A., Kurokawa, H., Tomita, K., Kokaji, S.: M-TRAN: Self-reconfigurable modular robotic system. IEEE/ASME Trans. on Mechatronics 7, 431–441 (2002)

[6] Castano, A., Shen, W., Will, P.: CONRO: towards deployable robots with inter-robot metamorphic capabilities. Autonomous Robots 8, 309–324 (2000)

[7] Sproewitz, E., Billard, A., Dillenbourg, P., Ijspeert, A.J.: Roombots—Mechanical Design of Self-Reconfiguring Modular Robots for Adaptive Furniture. In: IEEE International Conference on Robotics and Automation, pp. 4259–4264 (2009)

[8] Yim, M., Duff, D.G., Roufas, K.D.: PolyBot: a modular reconfigurable robot. In: Proceedings - IEEE International Conference on Robotics and Automation, pp. 514–520 (2000)

[9] Moeckel, R., Jaquier, C., Drapel, K., Dittrich, E., Upegui, A., Ijspeert, A.: Exploring adaptive locomotion with YaMoR, a novel autonomous modular robot with Bluetooth interface. Industrial Robot 33, 285–290 (2006)

[10] Salemi, B., Moll, M., Shen, W.: Superbot: A deployable, multi-functional, and modular self-reconfigurable robotic system. In: IEEE International Conference on Intelligent Robots and Systems, pp. 3636–3641 (2006)

[11] Asadpour, M., Sproewitz, A., Billard, A., Dillenbourg, P., Ijspeert, A.: Graph signature for self-reconfiguration planning. In: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, pp. 863–869 (2008)

[12] Asadpour, M., Ashtiani, M., Sproewitz, A., Ijspeert, A.: Graph signature for self-reconfiguration planning of modules with symmetry. In: 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, pp. 5295–5300 (2009)

[13] Bonacich, P.: Power and Centrality: A Family of Measures. The American Journal of Sociology 92, 1170–1182 (1987)

[14] Yim, M., Goldberg, D., Casal, A.: Connectivity planning for closed-chain reconfiguration. In: Proceedings of SPIE - The Int. Society for Optical Engineering, pp. 402–412 (2000)

[15] Casal, A., Yim, M.: Self-reconfiguration planning for a class of modular robots. In: Proceedings of SPIE - The International Society for Optical Engineering, pp. 246–257 (1999)

[16] Hou, F., Shen, W.: Distributed, dynamic, and autonomous reconfiguration planning for chain-type self-reconfigurable robots. In: Proceedings - IEEE International Conference on Robotics and Automation, pp. 3135–3140 (2008)

[17] Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences). W.H. Freeman (1979)

[18] Luks, E.: Isomorphism of graphs of bounded valence can be tested in polynomial time. Journal of Computer and System Sciences 25, 42–65 (1982)

[19] Jiang, X., Bunke, H.: Optimal quadratic-time isomorphism of ordered graphs. Pattern Recognition 32, 1273–1283 (1999)

[20] Lavalle, S.M.: Rapidly-Exploring Random Trees: A New Tool for Path Planning (1998)

Sensor-Coupled Fractal Gene Regulatory Networks for Locomotion Control of a Modular Snake Robot

Payam Zahadat, David Johan Christensen, Serajeddin Katebi, and Kasper Stoy

Abstract. In this paper we study fractal gene regulatory network (FGRN) controllers based on sensory information. The FGRN controllers are evolved to control a snake robot consisting of seven simulated ATRON modules. Each module contains three tilt sensors which represent the direction of gravity in the coordination system of the module. The modules are controlled locally and there is no explicit communication between them. So, they can synchronize implicitly using their sensors, and coordination of their behavior takes place through the environment. In one of our experiments, all the three tilt sensors are available for the FGRNs and a simple controller is evolved. The controller is a linear mapping of one input sensor to the output. It is only based on one sensor input and ignores the other sensors as well as the regulatory part of the network. In another experiment, the controller's input uses one of the other sensors that carries less information. In this case, the evolved controller blends sensory information with the regulatory network capabilities to come up with a proper distributed controller.

1 Introduction and Related Work

Modular robots are distributed robots made up from a number of mechanically coupled modules where each module is typically controlled by its own local controller. These robots are distributed and dynamic by nature and they have limited inter-modular communication and processing capabilities. In this paper we evolve FGRNs as distributed controller for modular robots. The purpose of the paper is to study the FGRN controllers based on sensor information. The FGRNs are evolved as local controllers of modules. Each FGRN controller receives inputs provided by

Payam Zahadat · David Johan Christensen · Kasper Stoy
Modular Robotics Lab, The Maersk Mc-Kinney Moller Institute,
University of Southern Denmark, Campusvej 55, DK-5230 Odense M, Denmark

Payam Zahadat · Serajeddin Katebi
Department of Computer Science and Engineering, School of Engineering,
Shiraz University, Shiraz, Iran

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 517–530.
springerlink.com

the local sensors of the module containing it. The usefulness of the sensor information and the FGRN capability to make proper output patterns are investigated in this paper.

Gene regulatory network (GRN) is a network of genes which interact with each other and regulate each other's activation behavior. Instead of direct mapping of genotypes to phenotypes, nature implements an indirect development of a phenotype using GRNs. In biological cells, genome consists of a number of genes which encode proteins. Proteins play different roles in a cell. They can represent input signals; operate as intermediate substrates to drive the interaction between genes, and shape structure or behavior of the cell which can change during time. Proteins interact with each other and with the genes and this is an ongoing process in the whole life time of a cell. Complex behavior of a cell is the result of this interaction. Production of a protein can be initiated by signals coming from the environment of the cell. The environment might be either the outside world or even the neighboring cells. In this way, the local environment can influence the cell's inner dynamics and changes the behavior of the cell. Differentiation of cells in a multicellular creature takes place through similar processes. In a multi-cellular creature, all cells contain the same genome, but based on the local environment of the call they differentiate during development and may behave differently.

In the field of computation systems different GRN models [2, 12, 14, 18] have been defined to indirectly map genotype to phenotype in order to make more complex phenotypes and behaviors. In some works, models of GRNs are evolved for making mathematical output functions [19], developing neural networks for controlling robots [9, 11, 16] or specifying the morphology of 3D organisms [10]. Also, GRN models have been used to develop the morphology of robots as well as their neural network controllers [6]. A special type of GRNs, which utilizes fractal proteins as the intermediate substrate of gene interaction, is called FGRN [3]. The recursive and self-similar nature of fractal proteins make the fractal genetic space evolvable, complex, and redundant [3, 4, 5]. In a number of previous works, FGRNs are evolved to do different tasks such as producing desired patterns, controlling conventional robots and motion planning [4, 26]. They have also been used [26] as local controllers of modular robots in a simpler version than the current paper such that each FGRN controller selects between different possible commands that can be executed by every module and without any dynamic influence from the outside environment.

The main contribution of this paper is further investigation of the usefulness of FRGN for control of modular robots; in particular, we extend on previous work [26] by looking at how sensor-inputs can be integrated with FGRN. The controllers we develop in this paper are tied to the physics of the ATRON self-reconfigurable robot and are thus not directly applicable to control of other modular robots such as M-TRAN [20], SuperBot [22], CKBot [25] due to their differences in weight, actuator strength, placement of sensors, etc. However, it is a general problem of all embodied controllers that they rely on the specific physical properties of the robot on which they run. For the same reason, the controllers cannot be directly applied to control of non-reconfigurable snake robots either (see [23] for an overview). However, the idea of a model-free approach relying on

tilt-sensors for both local control and implicit synchronization between segments of the snake may be transferable to other robots as well. More importantly, we expect our development method based on evolution of FGRN can be applied to these systems. In modular and multi-segment robotics, controllers for snake robots have been extensively studied based on gait control tables [24], Central Pattern Generators (CPGs) [15, 17], artificial hormones [13], and role-based control [22]. However, opposed to our controllers these controllers except [17] are open-loop and in the case of the latter two rely on explicit communication between modules for synchronization.

The paper is organized as follows. The next section reviews the biological inspiration and computational implementation of FGRN. Then, the application of FGRN as a sensor-based controller is described. Consequently, the controllers which respectively are evolved with unrestricted and restricted access to sensor information are investigated and the achieved behaviors are compared.

2 Gene Regulatory Networks

2.1 Biological Inspiration

Development of phenotypes can be thought of as a product of interaction between genes and proteins in their environment. Proteins drive development and functioning of a cell and are used for communication between a cell and its environment that might include other cells.

A cell contains a genome and a cytoplasm which are surrounded by a membrane (Fig. 1) [1]. The membrane separates the interior of a cell from the outside environment. Receptor proteins are embedded in the membrane and control the movement of environmental proteins into the cell. The cytoplasm contains a compound of proteins inside the cell. The genome consists of a set of genes. Every gene contains a sequence that encodes a protein (coding region) and a sequence that determines the conditions for activation or suppression of that gene (promoter region) (Fig. 1).

An active gene expresses and produces its appropriate protein as encoded in its coding region. For a gene to be activated, the similarity between the cytoplasm content and the promoter region of the gene has to reach a threshold.

Fig. 1 An example cell (left) and a gene (right)

The cytoplasm content is altered by proteins produced by genes inside the cell or the environmental proteins which have entered the cell passing through receptors.

During the development of a cell, the protein content of the cytoplasm might match against the promoter of some genes and get them to suppress or express proteins. Every produced protein will enter to the cytoplasm and alter its content. The new content, in turn, affects the expression of genes in the next step. In this way, every protein inside a cell either produced by the genes or from environment might influence the expression of the genes directly or indirectly. On the other hand, the functional behavior of a cell is determined by special proteins in the cell and is controlled by the cytoplasm content.

The ongoing interaction between proteins and genes continues for the whole lifetime of a cell and is considered a network of genes which regulate the expression of each other and is called a Gene Regulatory Network (GRN).

2.2 Fractals and Gene Regulatory Networks

In a series of works reported by Bentley [3, 4, 5] a protein model called fractal protein is developed as an abstraction of the protein substance of gene regulatory networks in an evolutionary system.

Fractal proteins are square windows on the Mandelbrot fractal set with a fixed resolution (Fig. 2). Each fractal protein is represented by a square matrix of integer values, but it is encoded by only three values (x, y, z). (x, y) is the coordination of the center of the window on the fractal set and z is the length of the sides. Therefore, by changing these three values we can reach different locations and different scales of the fractal set which benefit the evolvability due to the self-similarity found in fractals. This property makes a desirable redundancy which means the same potential solution can be found in indefinite number of points in genotype space and it facilitates the evolutionary process. Fig. 2 represents an example fractal protein.

In addition to a square matrix of integer values, a single integer value relates to each fractal protein as its concentration level. The concentration level represents the current amount of the protein. The value increases when more of the protein is produced and decreases slowly over time to resemble normal degradation that happens in biological cells. The value is constant for the receptor proteins.

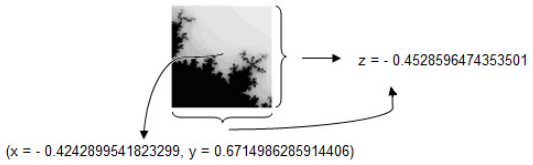


Fig. 2 An example fractal protein and the three values which specify it

Fractal proteins can merge together and make protein compounds. A fractal protein compound is represented by a square matrix of integer values in the same way as fractal proteins. Merging is a pixel-wise max operation between the corresponding matrices. See Fig. 3(a) for an example.

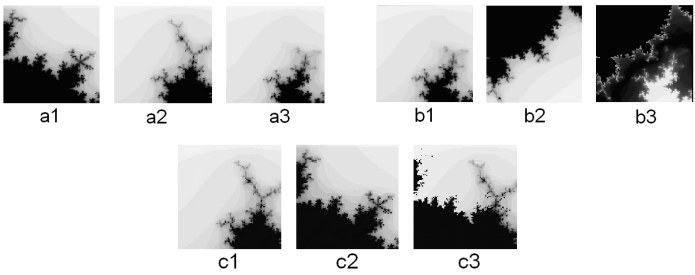


Fig. 3 Fractal Protein Operators: a) Merge: Two proteins a1 and a2 are merged as a3. b) Match: The cytoplasm protein compound b1 matches against the promoter of a gene (b2) and b3 is resulted as the calculated absolute difference. c) Mask: Environmental protein c1 passes through the receptor protein c2 and some portions of it (c3) which are corresponding to non-black pixels of c2 are allowed to enter the cytoplasm.

The cytoplasm of an FGRN cell is a compound of all the proteins inside the cell. Every protein that is produced in the cell or enters the cell from outside will be merged into the content of the cytoplasm.

A genome in an FGRN cell consists of a set of genes. Genes consist of a sequence of values representing promoter region, coding region, threshold parameters, and type of the gene.

The coding region contains the three real values which encode a fractal protein. In the same way as the coding region, the promoter region consists of three real values that encode a square matrix of fractal values as well. This matrix works as a window that will be put on the cytoplasm protein compound matrix and is used to calculate the matching degree between the promoter of the gene and cytoplasm content (See Fig. 3(b) for an example). The matching degree along with the total concentration of matched proteins on promoter region, determine the degree of activation (or suppression) of the gene and might specify its protein production rate. Threshold parameters are used to calculate the matching degree and protein production rate of each gene. To assimilate different types of genes in a cell, every gene belongs to one of the types represented in Table 1. Each gene contains an integer value that represents its type. The lifetime of an FGRN cell consists of a number of developmental cycles which can be summarized as the steps represented in Fig. 4. For more detailed descriptions of FGRN systems and the corresponding formulas see [3, 4, 26].

Table 1 Different gene types

Gene Type	Description
Regulatory	Includes both promoter and coding region. Its encoded protein will be produced and merged into cytoplasm and participate in regulation of gene expression.
Environmental	Determines the proteins which might be present in the environment of the cell.
Cell receptor	Contains a coding region and produces a receptor protein. Receptor proteins merge together and act as a mask to permit variable portions of environmental proteins to the cytoplasm (See Fig. 3(c)).
Behavioral	Comprises a promoter region and a coding region. The values in the coding region can directly participate to determine the outputs of the cell.

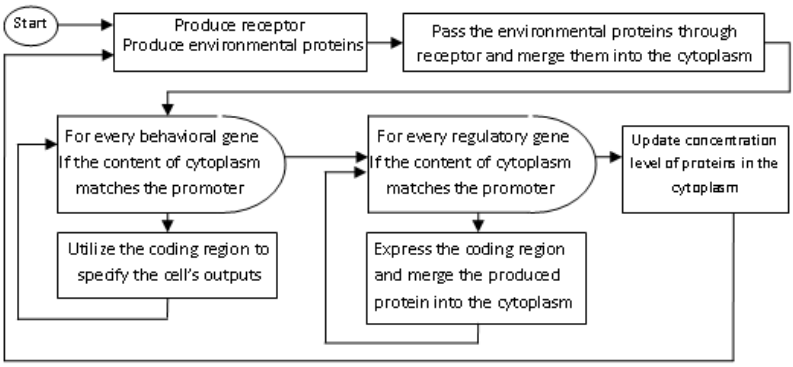


Fig. 4 A developmental cycle of an FGRN cell

3 Evolving FGRN Local Controllers with Tilt-Sensor Input

In this work, FGRN controllers are evolved for the ATRON robot [21] which is a homogenous, lattice-based self-reconfigurable modular robot. An ATRON module weighs 0.850kg and has a diameter of 110mm. A module consists of two hemispheres which can rotate infinitely relative to each other with a speed of 60 degrees per second. Each hemisphere contains two passive (bars) and two active connectors (hooks), see Fig. 5.

Fig. 5 From left to right: An ATRON module, a seven-segment snake robot

Simulation experiments are performed in an open-source simulator named Unified Simulator for Self-Reconfigurable Robots (USSR) [8]. The simulator is based on Open Dynamics Engine which provides simulation of collisions and rigid body dynamics. Physical forces like gravity and friction are implemented and the parameters, e.g. strength, speed, weight, etc., has been calibrated with the existing hardware. The physical validity of the mechanical simulation has been demonstrated in the previous works [7] where the controllers were successfully transferred from simulation to the real modules. The implemented sensors are ideal tilt sensors and not still verified.

FGRN local controllers with access to tilt-sensor inputs are evolved. The controller is evolved for a snake-shaped robot consists of seven ATRON modules (See Fig. 5) and there is no explicit communication or synchronization between the modules. Every module contains three tilt sensors as (TiltX, TiltY, TiltZ). The sensors specify the direction of gravity related to the coordination system of the module (Fig. 5). The initial tilt sensor values of a module are different for the neighbor modules because of the positioning of the connectors in ATRON. The initial values are (0, -90, 0) for the modules in the odd positions of the snake and (-90, 0, 0) for the ones in the even positions.

Evolution searches for FGRN genomes which are used in the local FGRN controllers to solve a locomotion task. To evaluate a genome, an identical version of genome is copied to all the FGRN cells which are situated in the modules. Each cell receives tilt sensor values from the module's local sensors. Initially, one input gene is related to each sensor. The level of protein expression of each input gene is determined by the value received from the related sensor. The development cycle in Fig. 4 is performed and the new concentration level of each protein in the cytoplasm is specified. In order to make an actuator command for each module in every step, each module independently run its own FGRN cell for one developmental cycle and receives an output from the cell. The FGRN output is calculated on the basis of activation level of behavioral genes and the real values of the coding region [4]. The output value received from the cell is scaled and used as the

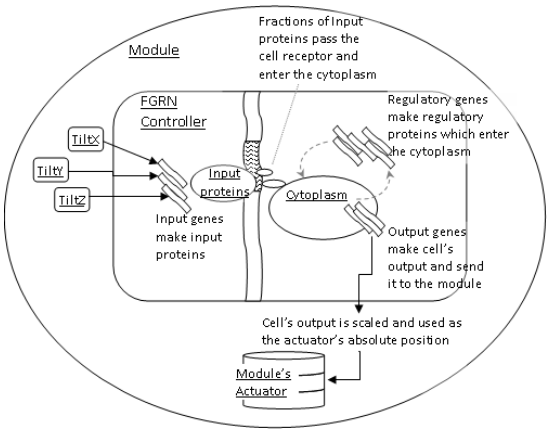


Fig. 6 Each module contains an FGRN controller that specifies the actuator's absolute position

absolute position of the module's actuator which is between -180 to 180 degrees (See Fig. 6). Modules use the nearest rotation angle to reach the desired absolute position. Robot runs for a specific time period (50 sec.) and fitness is simply evaluated as the average speed of locomotion of the robot. For each evolutionary run, a population of 50 FGRN genomes is evolved for 250 generations using a version of steady-state genetic algorithm with lifespan limits [2]. Each genome is initialized with randomly generated regulatory, receptor, environmental, and behavioral genes. Evolution is allowed to regulate the number of each type of genes (See [3, 26] for more details).

4 Experimental Results and Discussion

In order to investigate the usefulness and properties of integrating sensors with FGRN controllers for the snake robot, we performed two experiments. First we studied the FGRNs that are evolved when the input is available from all of the local tilt sensors and observed the usefulness of the sensor-inputs. Then we examined the ability of FGRN to produce proper output patterns when the input is limited to a sensor with less information.

4.1 Evolving Controllers with Unrestricted Access Sensors

In order to study if FGRN controller can gain any benefit from the tilt sensor inputs, we evolved the controllers with access to all the three local sensors. Evolution was free to use all or some of the sensor inputs for the controllers. The evolved controllers were evaluated in the locomotion task and the speed of locomotion was measured as the distance between the initial position and the end position of the center of mass of the robot and used as the fitness value. We repeated the experiment for 10 independent runs. The average speed of the best controllers from the ten runs was 0.0334 m/s (with standard deviation of 0.0032) and all the runs evolved controllers that generated rolling locomotion.

In order to investigate the effects of different sensor values in producing robot behavior, we limited access of the evolved controller to different combinations of the sensors and set the others to zero. The achieved results demonstrated that for 9 runs out of 10, there is no detectable effect for the TiltY and TiltZ sensors. In the only other run, output was produced based on both TiltY and TiltZ sensor values and no use of TiltX detected. This controller had the speed of 0.027 m/s.

In the same way as the sensor values, we removed regulatory genes of the evolved FGRN controllers in order to investigate their influence on the controllers' behavior. The investigation demonstrated that only in one of the evolved solutions, regulatory genes were participating in producing the controllers' output. No significant difference was observed between the speed of this controller and the rest.

Based on the above investigations, for the eight runs out of the 10 runs, the evolved controllers produced output merely from TiltX sensor value. This means the controller directly maps one input to the output which is a simple controller for this robot.

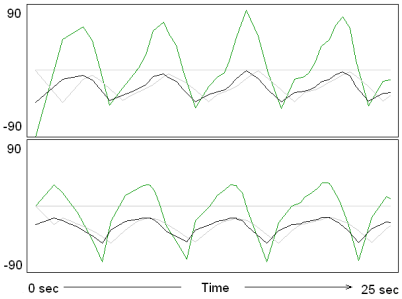


Fig. 7 Internal dynamics of the controllers of the first experiment for the two first modules of the snake. Green lines represent sensor value, black lines represent the output for the actuator absolute position, and the gray lines represent the actuator's real position.

For a typical evolved FGRN controller, the internal dynamics of the modules are represented in Fig. 7. As it is demonstrated in the figure, the output values can be simply calculated using a linear equation. We derived the equation from the related input and output data as:

$$\text{Output} = \text{TiltX} * 0.33 - 60.8$$

4.2 FGRN with Restricted Sensor Information

In the second experiment, we investigated whether the regulating dynamics of FGRN can make proper output patterns when the instant values of input sensors doesn't carry enough information. As the results of the last experiment demonstrated, TiltZ sensor has no detectable effect in producing the control outputs. It made us suspect that this sensor doesn't have enough information for this control task. Therefore, we first tried to evolve a linear equation solely based on TiltZ sensor. We implemented a real-valued genetic algorithm to evolve a population of 50 individuals for 250 generations. The experiment was repeated ten times and we observed that evolution failed to find a proper controller. Then, we evolved FGRN controllers which have only access to TiltZ sensor value to investigate if FGRN can exploit this restricted sensor information.

We repeated the evolutionary process for 10 independent runs and observed different locomotion-types for the best controllers of the different runs. The locomotion-types are discussed in three groups. The first group consists of the controllers which generate rolling-type locomotion for the robot. In order to study which parts of the network are involved in the control process, we disabled the sensor and each of the regulatory genes one by one. In all cases the controller failed to make proper locomotion. It demonstrates that both regulatory genes and sensor input are used by the controller. The Internal dynamics of one of the best controllers we achieved in this group is represented in Fig. 8. In order to get an informal impression of the robustness of the controllers in case a module breaks which lead to restarting controller, we randomly chose a module and restarted its controller to the initial state during the robot's run. We repeated the experiment several times and observed that the robot continues its normal locomotion after a short while.

The second group includes the controllers that make crawling-type locomotion. Investigation of the internal dynamics of the controller demonstrates that these solutions are mainly based on the regulatory genes and doesn't really exploit the input information. We observed that these robots are not robust against randomly restarting of the controllers.

The third group consists of the controllers which make efficient locomotion once in a while. Benefiting from the robot's body accidental flips over, these controllers sometimes make fast locomotion, and otherwise they do not produce locomotion. Since evolution only searches for fast controllers and there was no selection pressure towards the robustness and reproducability of the locomotion, the large fitness that these controllers gain by the accidental success is enough to pick them up among the other controllers in the evolutionary process. These controllers are not robust even during normal locomotion. Average and standard deviation of speed reached by the different controller groups are shown in Table. 2.

Table 2 Speeds reached by different types of locomotion

	All	Rolling	Crawling	Others
Average speed	0.0209	0.0248	0.0168	0.0212
Standard deviation	0.0076	0.0047	0.0015	0.0112

The inner dynamics of a typical controller is represented in Fig. 8. The controller is selected from the rolling-type group which demonstrates an efficient and robust behavior. As it is represented in the figure, the TiltZ value is zero for all the modules on the start of the execution. Therefore, there is no difference between the cells of a robot at the beginning and all of them make the same output for their module actuators. Rotating actuators as a result of command execution, changes module's orientation. This might lead to different TiltZ sensor values for different

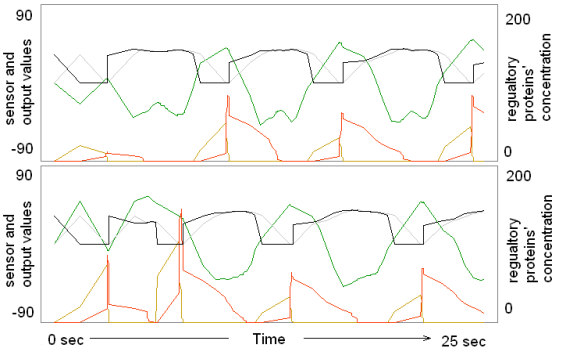


Fig. 8 Internal dynamics of the selected controller of the second experiment for the two first modules. Green lines represent the sensor values, black lines represent the output for the actuator absolute position; and the red and brown lines represent the concentration level of the two regulatory proteins.

modules. After a short while of chaotic behaviors, modules start to synchronize and coordinate their behaviors through environmental feedback which is received in the form of the sensor values.

4.3 Comparison of Behaviors from the Best Evolved Controllers of the Two Experiments

In order to have an impression of the rolling behavior produced by the best controllers of each of the two experiments, we studied the actuator's absolute positions for one typical controller evolved in the first experiment and one typical controller from the second one.

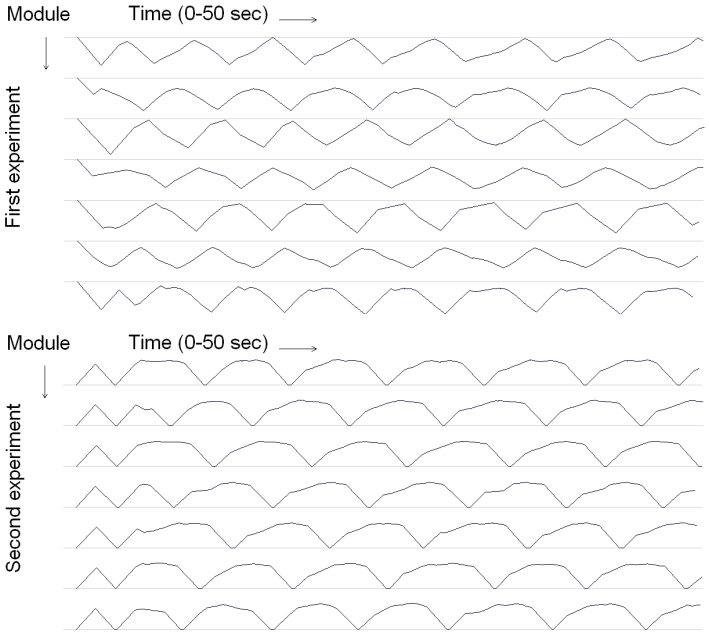


Fig. 9 Actuators' absolute positions of all the modules for a typical controller of the first and second experiments respectively

As it is demonstrated in Fig. 9, the module actuators have oscillatory behaviors. For the first experiment, the average period of estimated oscillation of the actuator absolute positions is 6.46 sec (with standard deviation of 0.63). The estimated phase shifts between the actuator signals of the consecutive modules is represented in Table 3.

For the second experiment, the average period of oscillation of the actuator absolute positions is 7.6 sec (with standard deviation of 0.2). The estimated phase shifts of the consecutive modules are represented in Table 3.

Table 3 Phase shift (per period) between the neighbor modules for the typical controller of each experiment

Module number	#1	#2	#3	#4	#5	#6	#7
first experiment	-	0.21	0.52	0.34	0.34	0.41	0.48
second experiment	-	0.56	0.37	0.41	0.40	0.36	0.38

It is interesting to note that the phase difference between neighbor modules is not constant. We suspect this is because modules are subject to different forces and dynamics depending on their position in the snake.

5 Conclusions

In this paper we explored fractal gene regulatory network controllers for a snake-shaped modular robot where only the tilt sensor inputs are available for the controllers. First, we provided the controllers with all the three tilt sensor inputs. The evolved controllers were simple linear equation which exploits only one of the three sensor inputs and ignores the regulatory abilities. In the next step, we restricted the controller's access to one of the other sensors and tried to evolve new linear equation controllers based on this information. Since evolution couldn't find the proper controllers, we suspect that the information provided by that sensor is not enough to be used by such a simple controller.

Then we evolved FGRN controllers with access to this sensor information. The resulting controllers made appropriate oscillatory output patterns to control the modules. Investigating the different parts of the FGRN genome demonstrated that the system exploits both sensor values and regulatory network capabilities to make the proper controller commands. As it might be expected, the generated outputs of the controllers were oscillatory patterns shifted for each module. Furthermore, we performed some preliminary tests towards robustness of the controllers in both cases and observed that the controllers can drive the robot properly in the case of random restarting of the controllers during locomotion.

All in all, as an early step to use FGRN as a modular robot controller, it is demonstrated that FGRN can be evolved to both simple and relatively complex controllers depending on the problem. Furthermore, when the capability of FGRN to make oscillatory patterns is coupled with the sensor information, the controllers show some degree of adaptability. In this way, the identical controllers generate different oscillatory outputs when situated in different modules and may provide some levels of robustness for the whole system. While it has not been verified we think that the idea of using sensor-coupled FGRN controllers for local control and synchronization between segments can be transferable to other modular robots as well.

Acknowledgments. The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 - Future Emerging Technologies, Embodied Intelligence, under grant agreement no. 231688.

References

1. Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., Walter, P.: Molecular Biology of the Cell, 4th edn., Garland (2002)
2. Banzhaf, W.: On evolutionary design, embodiment and artificial regulatory networks,". In: Iida, F., Pfeifer, R., Steels, L., Kuniyoshi, Y. (eds.) Embodied Artificial Intelligence, pp. 284–292. Springer (2004)
3. Bentley, P.J.: Fractal proteins. J. Genetic Programming and Evolvable Machines 5(1), 71–101 (2004)
4. Bentley, P.J.: Adaptive Fractal Gene Regulatory Networks for Robot Control. In: Genetic and Evolutionary Computation Conference, Seattle, USA (2004)
5. Bentley, P.J.: Methods for Improving Simulations of Biological Systems: Systemic Computation and Fractal Proteins. J. R Soc Interface (2009)
6. Bongard, J.C., Pfeifer, R.: Evolving Complete Agents Using Artificial Ontogeny. In: Hara, F., Pfeifer, R. (eds.) Morpho-functional Machines: The New Species (Designing Embodied Intelligence), pp. 237–258. Springer (2003)
7. Christensen, D.J., Bordignon, M., Schultz, U.P., Shaikh, D., Stoy, K.: Morphology Independent Learning in Modular Robots. In: Int. Symposium on Distributed Autonomous Robotic Systems, pp. 379–391 (2008)
8. Christensen, D.J., Schultz, U.P., Brandt, D., Stoy, K.: A Unified Simulator for Self-reconfigurable Robots. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (2008)
9. Dellaert, F., Beer, R.: A Developmental Model for the Evolution of Complete Autonomous Agents. In: 4th Int. Conf. on Simulation of Adaptive Behavior, pp. 393–401. MIT Press, Cambridge (1996)
10. Eggenberger, P.: Evolving Morphologies of Simulated 3D Organisms Based on Differential Gene Expression. In: Husbands, P., Harvey, I. (eds.) 4th European Conf. on Artificial Life (ECAL), pp. 205–213. MIT Press, Cambridge (1997)
11. Federici, D.: Evolving a Neurocontroller through a Process of Embryogeny. In: Schaal, S., et al. (eds.) 8th Int. Conf. of Simulation and Adaptive Behavior, pp. 373–384. MIT Press (2004)
12. Federici, D., Downing, K.: Evolution and Development of a Multi-Cellular Organism: Scalability, Resilience and Neutral Complexification. J. Artificial Life 12(3), 381–409 (2006)
13. Hamann, H., Stradner, J., Schmickl, T., Crailsheim, K.: Artificial Hormone Reaction Networks: Towards Higher Evolvability in Evolutionary Multi-Modular Robotics. In: The 12th Int. Conf. on Artificial Life (2010)
14. Hornby, G.S., Pollak, B.: The Advantages of Generative Grammatical Encodings for Physical Design. In: Congress on Evolutionary Computation, pp. 600–607. IEEE Press (2001)
15. Ijspeert, A.J., Crespi, A.: Online trajectory generation in an amphibious snake robot using a lamprey-like central pattern generator model. In: IEEE Int. Conf. on Robotics and Automation, pp. 262–268 (2007)
16. Jakobi, N.: Harnessing Morphogenesis. In: Paton, R. (ed.) Int. Conf. on Information Processing in Cells and Tissues, Liverpool, UK, pp. 29–41 (1995)
17. Kamimura, A., Kurokawa, H., Yoshida, E., Murata, S., Tomita, K., Kokaji, S.: Automatic Locomotion Design and Experiments for a Modular Robotic System. IEEE/ASME Transactions on Mechatronics 10(3), 314–325 (2005)
18. Kennedy, P.J., Osborn, T.R.: A Model of Gene Expression and Regulation in an Artificial Cellular Organism. J. Complex Systems 13(1), 1–28 (2001)

19. Kuo, P.D., Leier, A., Banzhaf, W.: Evolving Dynamics in an Artificial Regulatory Network Model. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 571–580. Springer, Heidelberg (2004)
20. Murata, S., Tomita, K., Yoshida, E., Kurokawa, H., Kokaji, S.: Self-reconfigurable robot-module design and simulation. In: Proc. 6th Int. Conf. on Intelligent Autonomous Systems, Venice, Italy, pp. 911–917 (2000)
21. Ostergaard, E.H., Kassow, K., Beck, R., Lund, H.H.: Design of the Atron Lattice-Based Self-Reconfigurable Robot. J. Auton. Robots 21(2), 165–183 (2006)
22. Stoy, K., Shen, W.M., Will, P.: How to make a self-reconfigurable robot run. In: Proc. First Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2002), Bologna, Italy, pp. 813–820 (2002)
23. Transeth, A.A., Pettersen, K.Y., Liljeb, P.: A survey on snake robot modeling and locomotion. J. Robotica, 999–1015 (2009)
24. Yim, M.: Locomotion with a unit-modular reconfigurable robot. PhD thesis, Department of Mechanical Engineering, Stanford University, Stanford, CA (1994)
25. Yim, M., Shirmohammadi, B., Sastra, J., Park, M., Dugan, M., Taylor, C.J.: Towards robotic selfreassembly after explosion. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, San Diego, CA, pp. 2767–2772 (2007)
26. Zahadat, P., Katebi, S.D.: Tartarus and Fractal Gene Regulatory Networks with Input. J. Adv. Complex Sys. 11(6), 803–829 (2008)
27. Zahadat, P., Christensen, D.J., Schultz, U.P., Katebi, S.D., Stoy, K.: Fractal gene regulatory networks for robust locomotion control of modular robots, In: The 11th Int. Conf. on Simulation of Adaptive Behavior (2010)

Analysis of Human Standing-Up Motion Based on Distributed Muscle Control

Qi An, Yusuke Ikemoto, Hajime Asama, and Tamio Arai

Abstract. In developed countries, an aging society has become a serious issue; many activities of daily living (ADL) are impaired in the elderly. In order to improve this situation, it is necessary to develop an assisting method for the human standing-up motion because it is considered to be an important factor to ADL. It is unclear, however, how humans coordinate their multiple distributed actuators, muscles, due to the ill-posed problem of redundant their body system. In this paper, we analyze the human standing-up motion based on muscle coordinations, called synergies. A simulation method was developed to make mappings between muscle activations, joint torque, and the human body trajectory; thus, it can be predicted how modular muscle coordinations contribute to the motion. As a result, two primary synergies were extracted and how they coordinate to achieve the motion was elucidated; one synergy strongly affected joint movements and speed of the motion while bending the back and lifting the body up, and the other synergy controls their posture after they lift up their body. These findings could be useful for development of an assisting robotic system for rehabilitative training based on extracted distributed synergies from complex redundant human motion.

1 Introduction

In developed countries, a serious issue in healthcare is the aging society. As life expectancy increases, the ratio of the elderly to younger people has been increasing

Qi An · Hajime Asama · Tamio Arai
Dept. of Precision Engineering, Faculty of Engineering, The University of Tokyo,
7-3-1, Hongo, Bunkyou-ku, Tokyo, Japan
e-mail: {anqi,asama,arai-tamio}@robot.t.u-tokyo.ac.jp

Yusuke Ikemoto
Dept. of Mechanical and Intellectual Systems Engineering, Faculty of Engineering,
University of Toyama, 930-8555, Gofuku 3190, Toyama City, Japan
e-mail: yikemoto@eng.u-toyama.ac.jp

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 531–543.
springerlink.com

rapidly [1]. This situation has brought problems both to the elderly people and to caregivers. For the elderly, many activities of daily living (ADL) decline with age: walking, transferring themselves from the bed or chairs, dressing, and using the toilet [2]. Subsequently, many informal family caregivers suffer from physical and mental stress [3]. Therefore, in order to solve those problems, preventive medicine has become more and more important, with the suggestion that people should train themselves to stay healthy to avoid the necessity of being taken care of by others. For preventive medicine, human standing-up motion is considered to be an important factor; it is reported that elderly people without ability to perform this basic action have difficulty in mobility necessary for their ADL [4][5].

There have been studies to analyze human standing-up motion based on each joint angle or joint torque. For instance, according to changes of three joint angles (ankle, knee, and hip), Shenkman et al. divided human sit-to-stand motion into 4 phases: flexion momentum, momentum transfer, extension, and stabilization phases. They evaluated each phase in terms of momentum and stability [6]. On the other hand, Kotake et al. divided human sit-to-stand motion into six stages based on the angles of the ankle, knee, and hip, and computed the minimum torque of the hip and knee required to complete the motion [7]. It is unclear, however, how humans actually coordinate multiple distributed actuators, muscles, to achieve the standing-up motion. For assisting human daily motions, a robot suite has been proposed [8] to assist people using biological signals from their body, but suit-type machines need complex methods for controlling high D.O.F. state values due to redundant body systems. To avoid this complicated control, and to develop effective assisting machines, it is absolutely required to consider how human solves the ill-posed problem and to discover the dominant motion component of this human behavior.

We have focused on control of distributed muscle coordinations, called synergy, to analyze human standing-up motion [9]. Several researches suggest that training methods corresponding to muscles coordination are effective to improve motor control [10][11]; thus, it is important to analyze human standing-up motion based on synergies. If the standing-up motion can be divided into individual muscles movements that play different roles toward the motion, it will be useful to develop training or assisting methods. Our objective is, therefore, to extract essential synergies to control variant muscles in human standing-up motion. Moreover, a model of the motion is developed to simulate body trajectory from muscle activations in order to elucidate how each modular muscle synergy coordinates to achieve the motion.

2 Methods

2.1 Experimental Setup

2.1.1 Experiment Overview

In order to extract muscle coordination from human standing-up motion and to develop a simulation model, we performed an experiment to obtain three types of data

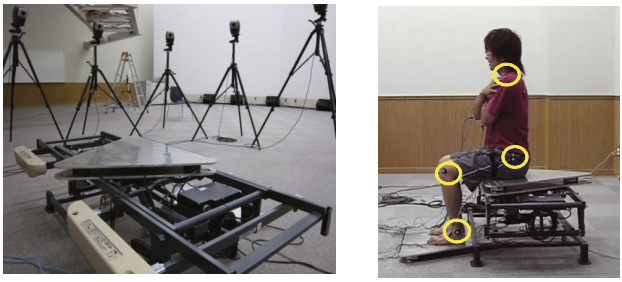
during the standing-up motion. In this experiment, one healthy 22 years old healthy man participated and 12 trials were obtained.

- Body motion trajectory
- Floor reaction force
- Surface electromyography (sEMG)

The experiment consisted of several trials of the standing-up motion, and at the beginning of trials, there were some initial conditions for body state of the subject: angle of his ankle was kept at 80 deg, his arms were crossed in front of the chest, and his back was straight. Also, the height of the chair used in the experiment was 0.425 m. Data recording of each trial continued for 7 secs and subject would start the motion approximately 2 secs after the start of recording by receiving a prompt from us.

2.1.2 Data Measurement

We recorded motion trajectory data at four points of the body using motion capture machines [HMK-200RT; MotionAnalysis]: ankle, knee, hip, and shoulder (Figure 1). The sampling rate for this data was 64 Hz and three joint angles, $\theta_{i\{i=foot,knee,hip\}}$, were obtained.



(a) (b)

Fig. 1 (a) A motion capture machine with eight cameras [HMK-200RT; MotionAnalysis] was used in our experiment to record body position. (b) Four points were recorded during the experiments: ankle, knee, hip, and shoulder.

Reaction forces from both feet and hip were recorded at 64 Hz by two specially made force plates (Figure 2). There were three force sensors in each corner of the force plates, and the three vertical forces from one plate were summed up to calculate the reaction force.

Personal-EMG [Oisaka Electroni Device Ldt] was used to record sEMG from sixteen muscles at 11200 Hz (illustrated in Figure 3-b). Two monopole electrodes (Figure 3-a) were attached along the axis of the muscle fibers, and distance between each electrodes were approximately 0.02 m. Muscle activation was recorded with

Fig. 2 Two specially designed force plates placed at the positions of the feet and hip of subjects

single differential between two electrodes. The data were filtered with a 10 Hz hi-pass filter and 50-60 Hz hum noise filter. Moreover sEMG data were filtered by the smoothing filter calculated by Equation 1 and downsampled to 64 Hz.

$$EMG_i(t) = \frac{\sum_{t'=0}^{24} EMG_i(t-t')}{25} \tag{1}$$

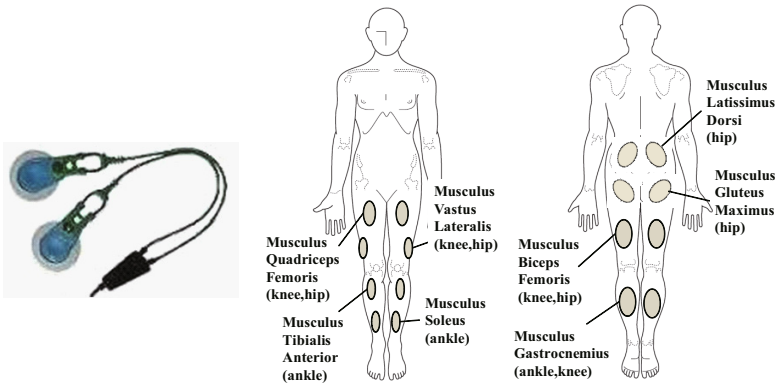


Fig. 3 (a) Two sEMG monopolar electrode sensors were used to measure each muscle. (b) Sixteen muscles were measured (eight muscles for the each half body). Above figure illustrates positions of measured muscles and joints of muscle attachement.

2.2 Synergy Analysis

2.2.1 Movement Generation

Figure 4 illustrates relationship of inputs and outputs of human body systems to generate motion. When humans move, the brain sends motor commands into several muscles to exert forces of flexion or extension. Next, muscles generate torques

related to differential of paired antagonist muscles attached to joints, and finally the human body moves according to its dynamics. In order to discover how distributed muscle coordination affects human body movement, we developed a simulation method was based on the model described in Figure 4.

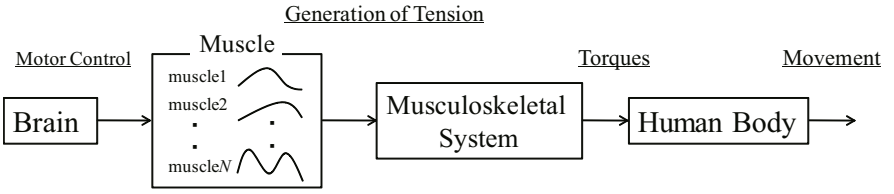


Fig. 4 The figure indicates model of human body movement. Firstly, brain sends motor control signals into muscles to exert forces by flexion or extension. Next, each muscle generates torque to human joints and human body moves according to its dynamics.

2.2.2 Synergy Hypothesis

A synergy hypothes proposed by Bernstein suggests that human complex motion with redundant active degrees of freedom could be controlled by a relatively smaller number of degrees of freedom via coordinated activation of several muscles called synergy [12]. Furthermore, d'Avella et al. developed a synergy model that regards muscle patterns as a linear combination of several smaller patterns of muscles [13]. In this paper, we adopted this model to analyze the data. In the model, let d be the number of measured muscles, t_{max} be a maximum time steps of the obtained sEMG data and $\mathbf{m}(t)(d \times t_{max})$ be a matrix indicating activation of d muscles during the motion at time $t(0 < t < t_{max})$. This $\mathbf{m}(t)$ was approximated by the linear-summation of time-varying synergies $\mathbf{w}_i(t)_{i=1,2...N}$ (N is the total number of extracted synergies and the duration time of each synergy is not always the same as t_{max}) with non-negative coefficient c_i and onset time delay t_i as in Equation 2. Although one pair of specific patterns is extracted from individual person, different motion can be achieved by changing values of c_i and t_i for every synergy. When changing c_i, strength of synergies activation can be controlled and the time of starting each synergy is adjusted by value of t_i.

$$\mathbf{m}(t) = \sum_{i=1}^{N} c_i \mathbf{w}_i(t - t_i) \tag{2}$$

2.2.3 Extraction of Synergies

We applied the decomposition algorithm [14] in order to extract synergies from observed sEMG patterns. This algorithm uses the multiplicative update rule to optimize elements of synergies, $\mathbf{w}_i(t)$, non-negative amplitude, c_i , and onset time delay

t_i. Squared error E^2 was used to evaluate error between observed muscle patterns and generated patterns by the model.

$$E^2 = trace\left((\mathbf{m}(t) - \sum_{i=1}^{N} c_i \mathbf{w}_i(t - t_i))^T (\mathbf{m}(t) - \sum_{i=1}^{N} c_i \mathbf{w}_i(t - t_i))\right) \tag{3}$$

Also, the cross-validation method was used to determine the number of synergies to be extracted. The procedure is described as below.

1. The twelve trial of data were randomly divided into four groups (each group has three trials).
2. The number of extracted synergies was set.
3. Three groups (training group) out of four were used to extract synergies and the remaining group (testing group) was used to calculate E^2. This process was conducted four times to compute E^2 for all four data sets and to calculate R^2 by Equation 4 where S_M^2 is variance of observed patterns.

$$R^2 = 1 - \frac{E^2}{S_M^2} \tag{4}$$

In the procedure, we calculated R^2 from obtained trials for the number of synergies, 1–5, in order to determine the minimum number of synergies to express human standing-up motion.

2.3 Simulation of Body Movement during Standing-Up Motion

2.3.1 Link Model

The model with four links and three joints (described in Figure 5-a) was used to indicate human body. This model focused on planar movement of body; thus, body movements of right and left side were averaged together. From the experiment explained above, three joint angles, $\theta_{i\{i=foot,knee,hip\}}$, were obtained and every joint torque, $\tau_{i(i=\text{ankle,knee,hip})}$, was computed by applying inverse dynamics calculation (Equation 5-7). In equations, m is mass of i-th link, g is gravity acceleration, $(x_n, y_n)_{n=1,2,3,4}$ is the position of center of gravity of each link, $(f_{xj}, f_{yj})_{j=2,3,4}$ is the horizontal force and vertical force between link i and link $i-1$, I is the inertial moment, and M is the moment from the center of gravity.

$$m\ddot{x}_n = f_{xj} - f_{xi} \tag{5}$$

$$m\ddot{y}_n = f_{yj} - f_{yi} - mg \tag{6}$$

$$I\ddot{\theta}_i = M - \tau_i - \tau_j \tag{7}$$

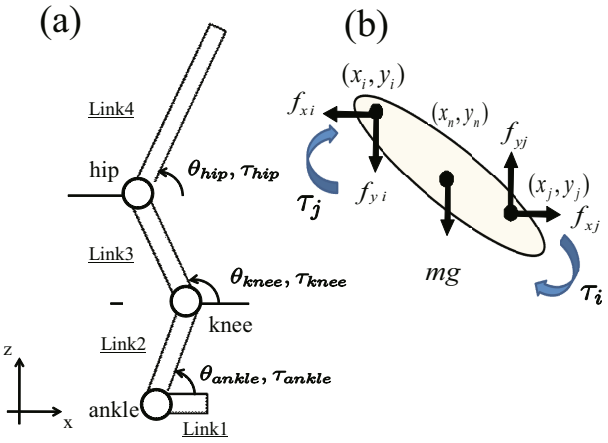


Fig. 5 (a) The link model used in our research to express human body. Each link expresses feet (Link1), lower legs (Link2), upper legs (Link3), and upper body (Link4) and joints between links are ankle, knee, and hip. (b) indicates variables used in computation of torques.

2.3.2 Estimation of Joint Torque and Angles

In order to understand how human body motion is generated from distributed muscle movements, neural networks were used to create mappings between sEMG patterns, and joint torques and between joint torques and angles (Figure 6) [15]. Both neural networks consist of three layers: input, hidden, and output. Among 7 secs recorded data, 3 secs were used for the analysis; the time when the shoulders of the subject reached the highest position was found in all trials, and 1 sec after that point and 2 secs before the point were used. All sEMG, torque, and angle data were normalized between 0.0-1.0 for the inputs and outputs of the neural networks.

One neural network was used to estimate one joint torque from sEMG patterns regarded as motor commands send by the brain. For the inputs of each neural network, only muscles attached to the joint were used; Figure 3-b illustrates which muscles are attached to each joint. There are two kinds of muscles: one-articular muscles and two-articular muscles; for example, musculus quadrcieps femoris was used for both knee and hip estimation. In addition to the strength of motor commands, both the length and the speed of muscle expansion and contraction are related to the tension generated by muscles [16]. Thus, in order to estimate joint torques, not only EMG patterns but also angle and angular velocity were included as inputs to the neural network, and a joint torque was obtained as an output signal. There were thirty-five nodes in the hidden layer of each network, and the back-propagating rule was adopted for the learning rule of neural networks. When learning phase of the neural network, angle data obtained from experiments and torque data calculated from previous session were used. In order to test the accuracy of the estimation, R^2 (Equation 4) was used, where E^2 is squared error between obtained data and estimated data and S_M^2 is variance of the obtained data. When testing the accuracy,

cross-validation method was used; observed trials of data were divided into nine sets of a training data, which were only used for teaching the network and the other three testing data which were used for calculating accuracy of the model. This was conducted four times to calculate R^2 for all trials.

Human joint angles were also estimated by a neural network. For input signals, three joint torques $\tau_i(t)$, joint angles $\theta_i(t)$, and joint angular velocity $\dot{\theta}_i(t)$ at time t were used (where i=foot, knee, and hip). Throughout forty nodes of the hidden layer, $\Delta\theta_i(t+1)$ and $\Delta\dot{\theta}_i(t+1)$ were obtained. For next inputs at time t+1, angle $\theta_i(t+1)$ and angular velocity $\dot{\theta}_i(t+1)$ were added by outputs $\Delta\theta_i(t+1)$ and $\Delta\dot{\theta}_i(t+1)$. Back-propagating rule was used for the learning rule, and in order to test the accuracy of the model, the same method for torque estimation was used here.

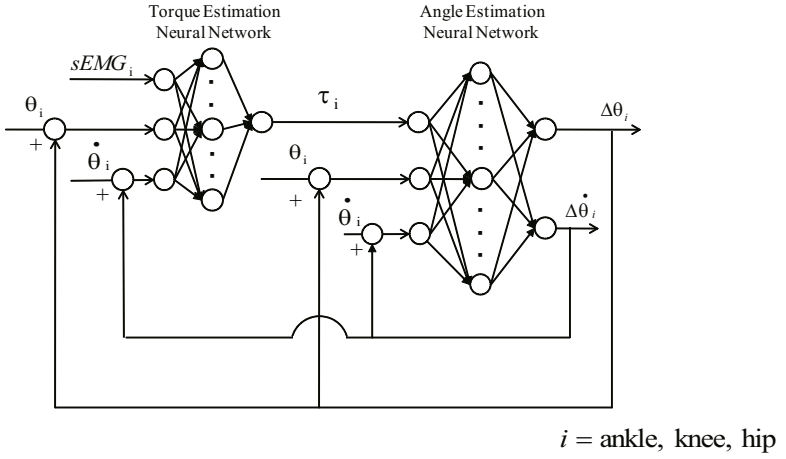


Fig. 6 The simulation method that estimates human body trajectory from muscle activations. Neural networks were used for both torque and angle estimations.

2.3.3 Detection of Synergy Contribution to Body Motion

In order to elucidate how extracted synergies affect human motion, the following procedure was repeatedly performed to simulate how the body trajectory changes corresponding to weakened sEMG patterns. For the simulation, both the torque and angle estimation neural networks previously described were used.

1. Weakened sEMG patterns were computed by decreasing c_i for the particular synergy in Equation 2.
2. The torque estimation neural networks output changed torques from the originally learned data.
3. The angle estimation neural network received changed torques, and it outputted altered joint angles and angular velocities.
4. $\Delta\theta_i$ and $\Delta\dot{\theta}_i$ were recurrently added into inputs of both neural networks in order to obtain θ and $\dot{\theta}$ at the next time step.

3 Results

3.1 Results of Torque and Angle Estimation

Table.1 and Table.2 show the average value and standard deviation of R^2 of the proposed simulation model. Figure 7 indicates examples of both joint torque and angle estimations from one trial; blue solid line is observed data and red dotted line is estimated data. They indicates that neural networks can adequately construct generation of human body movement.

Table 1 Results of Torque Estimation

	Average R^2	Standard Deviation
Foot Torque	0.60	0.14
Knee Torque	0.80	0.12
Hip Torque	0.73	0.11

Table 2 Results of Angle Estimation

	Average R^2	Standard Deviation
Foot Angle	0.76	0.32
Knee Angle	0.94	0.15
Hip Angle	0.85	0.19

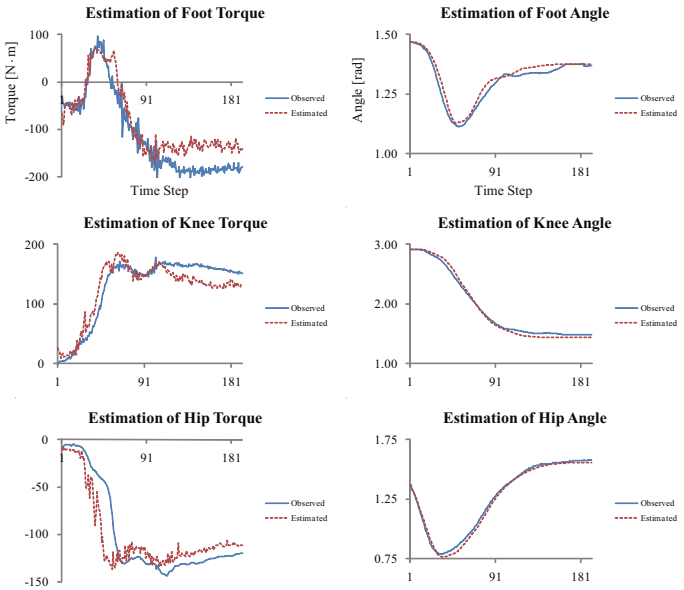


Fig. 7 Example of estimation of three joint torques and joint angles from the same trial. Left three graphs are results of torque extimation and right three graphs are ones of angle estimation. Blue solid line is observed data and red dotted line is estimated data.

3.2 Results of Synergy Analysis

The synergy analysis and simulation method were applied to the data measured from one healthy 22 year old man. The number of synergies to be extracted from

Table 3 Results of the time delay for two extracted synergies

	average time delay	standard deviation
Synergy 1	2.9	4.1
Synergy 2	88.2	1.8

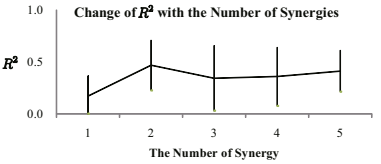


Fig. 8 Results of cross-validation method

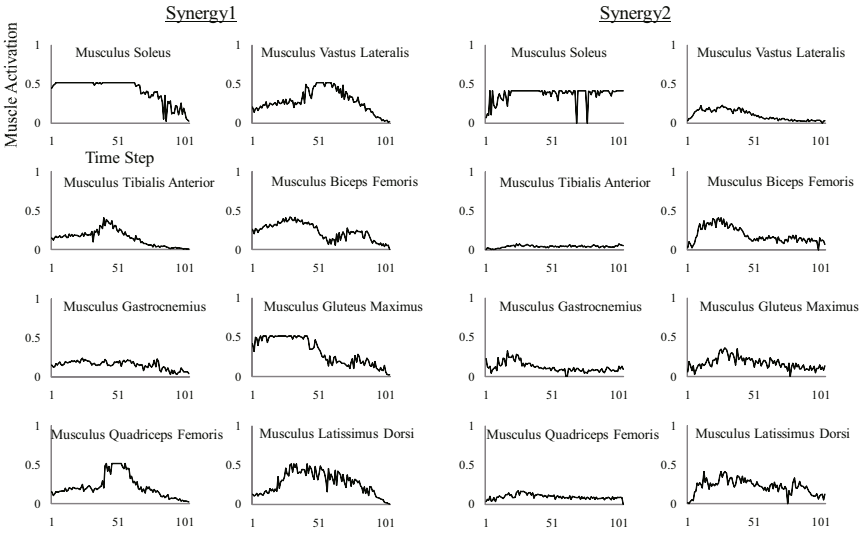
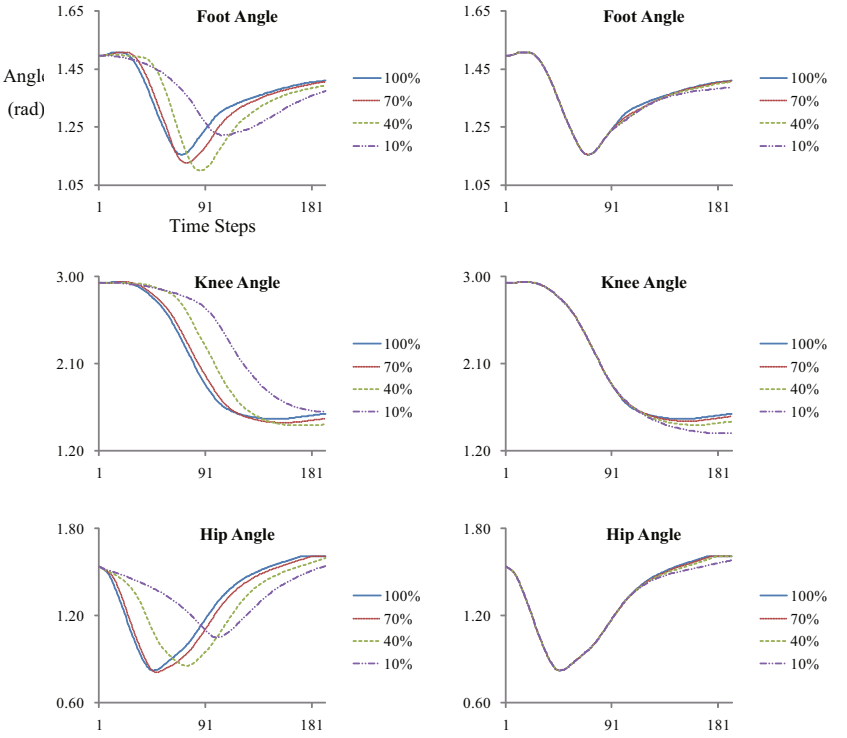


Fig. 9 shows extracted two synergies. From the decomposition algorithm, synergy1 started at the beginning of the motion and synergy2 started at the middle.

the observed EMG patterns is clarified by cross-validation. The relationship between mean value of R2 and the synergy number is depicted in Figure 7. It shows that two synergies are optimal to be extracted; before that number, the slope of the graph increases rapidly and the slope does not change sharply after that point.

As a result, two synergies were extracted (Figure 9) and Table.3 shows that synergy1 started at the beginning of the motion, and synergy2 started in the middle of the motion. In synergy1, all muscles except musculus gastrocnemius were activated, and in synergy2, musculus soleus, musculus gastrocnemius, musculus biceps femoris, musculus gluteus maximus, and musculus latissimus dorsi were activated.

Changed joint angles corresponding to weakened synergies are shown in Figure 10. Each graph shows every joint angle when c_i of each synergy is changed to 100%, 70%, 40% and 10%. According to the Figure 10, when synergy1 was weakened, trajectory patterns of every angle were distorted and the timing of each angle



(a) Angle Change when Synergy1 is Weakened (b) Angle Change when Synergy2 is Weakened

Fig. 10 Graphs indicate how each joint angle change with weakened synergies. Left three graphs show values when synergy1 was weakened, and right ones show values when synergy2 was weakened. X-axis represents a time step (1/64 sec) and y-axis represents angle (rad).

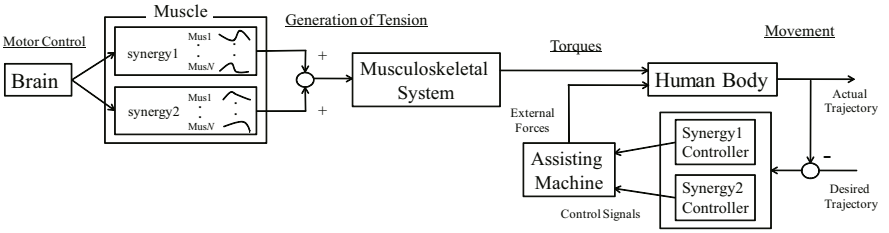


Fig. 11 Block diagram for controlling an assistive training equipment based on synergies

changes was shifted backward; this means synergy1 works to lift the body upward. On the other hand, when synergy2 was weakened, angle change occurred only at the end of the motion; it means that synergy2 affected stabilization of their posture after lifting the body.

4 Discussion and Conclusion

Integrated simulation method to estimate human body trajectory from muscle activations was developed. This method consisted of two neural networks; one estimated a joint torque from sEMG patterns, joint angle, and angular velocity, and the other estimated changes of angle and angular velocity from joint torques, angles, and angular velocities.

From the results of synergy analysis, it was implied that human standing-up motion was dominated by two behaviors based on different muscle coordination corresponding to bending the back, and lifting up the body and to controlling posture after returning his back straight. Results of the angle change with weakened synergies show that when synergy1 was weakened, every angle changes rapidly and the timing of lifting up was shifted backward. This implies that when synergy1 was weakened, the subject would take more time to complete lifting up their body compared to when synergy1 was at the maximum level. When examining angles with weakened synergy2, only changes occurred at the end of the motion. Those differences show that synergy2 mainly affected keeping the posture stable.

Those findings from the analysis of the human standing-up motion can be used for controlling an assistive training machine. Figure 11 shows an example of controlling an assisting machine. Since the complex human standing-up motion can be divided into only two dominant synergies, ”a synergy controller” can compute necessary muscle activations from difference between desired and actual body trajectories. Then, the controller sends control signals to an assistive training machine to exert external forces to help people stand-up. However, the developed method was applied to only one subject in this research and it will be required to test the efficacy of the suggested method or synergy analysis although the model did not have a particular assumption for the subject.

Efficient simulation method for human standing-up motion was developed. Joint torques were estimated from sEMG data, joint angle, and angular velocity based on the mechanism of torque generation from muscles. Also, he human body trajectory was estimated by the output of neural network using joint torques, angles, and angular velocities.

To analyze the human standing-up motion in terms of distributed muscle coordination, two synergies were extracted. While one synergy started at the beginning of the standing-up motion, which mainly activated seven muscles, the other synergy started at the middle of the motion and five muscles were activated. Synergy1 controls the speed of joint angles or timing of the motion when bending and lifting up their body and, the other synergy mainly affected the posture of the body after

lifting up the body. This finding would be helpful for development of an assistive training machine based on human distributed muscle coordinations.

References

1. Department of Economic United Nation and Social Affairs: World population, pp. 1–2 (2006)
2. Anderson-Ranberg, K., et al.: Declining physical abilities with age: a cross-sectional study of older twins and centenarians in Denmark. Age and Aging 28, 373–377 (1999)
3. Huston, P.G.: Family care of the elderly and care stress. American Family Physician 42(3), 671–676 (1990)
4. Guralnik, J.M., et al.: A short physical performance battery assessing lower extremity function: association with self-reported disability and prediction of mortality and nursing home admission. J. Gerontology 49, 85–94 (1990)
5. Guralnik, J.M., et al.: Lower-extremity function in persons over the age of 70 years as predictor of subsequent disability. N. Engl. J. Med. 332, 556–561 (1995)
6. Shenkman, M., et al.: Whole-body movements during rising to standing from sitting. Physical Therapy 70(10), 638–651 (1990)
7. Kotake, T., et al.: An analysis of sit-to-stand movements. Arch. Phys. Med. Rehabil. 74, 1095–1099 (1993)
8. Suzuki, K., et al.: Intention-based walking supp.ort for paraplegia patients with Robot Suit HAL. Advanced Robotics 21(12), 1441–1469 (2007)
9. An, Q., et al.: Extraction of Behavior Primitives for Understanding Human Standing-up Motion. In: Proc. of IEEE ICMA, pp. 1800–1805 (2009)
10. Rutherford, O.M.: Muscular coordination and strength training. Implications for injury rehabilitation. Sports Medicine 5(3), 196–202 (1988)
11. Ross, S.E., Guskiewicz, K.M.: Effect of Coordination Training With and Without Stochastic Resonance Stimulation on Dynamic Postural Stability of Subjects With Functional Ankle Instability and Subjects With Stable Ankles. Clinical Journal of Sport Medicine 16(4), 323–328 (2006)
12. Bernstein, N.A.: The Co-ordination and Regulation of Movement. Pergamon, Oxford (1967)
13. d'Avella, A., et al.: Combinations of muscle synergies in the construction of a natural motor behavior. Nature Neuroscience 6(3), 300–308 (2003)
14. d'Avella, A., et al.: Decomposition of emg patterns as combinations of time-varying muscle synergies. In: IEEE EMBS Conference on Neural Engineering, pp. 44–48 (2003)
15. Koike, Y., et al.: Estimation of dynamic joint torques and trajectory formation from surface electromyography signals using a neural network model. Biol. Cybern. 73, 291–300 (2005)
16. Partidege, L.D.: Muscle Properties: A Problem for the Motor Controller Physiologist. Posture and Movement, 189–241 (1979)

Cooperative Grasping and Transport Using Multiple Quadrotors

Daniel Mellinger, Michael Shomin, Nathan Michael, and Vijay Kumar

Abstract. In this paper, we consider the problem of controlling multiple quadrotor robots that cooperatively grasp and transport a payload in three dimensions. We model the quadrotors both individually and as a group rigidly attached to a payload. We propose individual robot control laws defined with respect to the payload that stabilize the payload along three-dimensional trajectories. We detail the design of a gripping mechanism attached to each quadrotor that permits autonomous grasping of the payload. An experimental study with teams of quadrotors cooperatively grasping, stabilizing, and transporting payloads along desired three-dimensional trajectories is presented with performance analysis over many trials for different payload configurations.

1 Introduction

Autonomous grasping, manipulation, and transportation of objects is a fundamental area of robotics research important to applications which require robots to interact and effect change in their environment. With recent advancements in relevant technologies and commercially available micro aerial vehicles (MAVs), the problem of autonomous grasping, manipulation, and transportation is advancing to the aerial domain in both theory and experiments. However, individual MAVs are fundamentally limited in their ability to manipulate and transport objects of any significant size. We address this limitation in this paper and consider the problem of controlling multiple quadrotor robots that cooperatively grasp and transport a payload in three dimensions.

Daniel Mellinger · Michael Shomin · Nathan Michael · Vijay Kumar
GRASP Laboratory,
University of Pennsylvania,
Philadelphia, PA 19104, USA
e-mail: {dmel,shomin,nmichael,kumar}@seas.upenn.edu

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 545–558.
springerlink.com

We approach the problem by first developing a model for a single quadrotor and a team of quadrotors rigidly attached to a payload (Sect. 3). In Sect. 4, we propose individual robot control laws defined with respect to the payload that stabilize the payload along three-dimensional trajectories. We detail the design of a gripping mechanism attached to each quadrotor that permits autonomous grasping of the payload (Sect. 5). An experimental study with teams of quadrotors cooperatively grasping, stabilizing, and transporting payloads of different configurations to desired positions and along three-dimensional trajectories is presented in Sect. 6.

2 Related Literature

The problem of aerial manipulation using cables is analyzed in [1, 2] with the focus on finding robot configurations that ensure static equilibrium of the payload at a desired pose while respecting constraints on the tension. We address a different problem as the robots use "grippers" that grasp the payload via rigid connections at multiple locations. The modeling of contact constraints is considerably simpler as issues of form or force closure are not relevant. Additionally, contact conditions do not change in our case (e.g., rolling to sliding, or contact to no contact). However, the system is statically indeterminate and the coordination of multiple robots is significantly more complex than in the case when the payload is suspended from aerial robots. In particular, as the problem is over-constrained the robots must control to move in directions that are consistent with kinematic constraints.

There is extensive literature on multi-fingered grasping and legged locomotion that discusses the problem of coordinating robot actuators with kinematic constraints [3, 4, 5]. However, our work is different in many ways. First, unlike legs or fingers, we have less control over the wrenches that can be exerted at each contact. Each robot is capable of controlling propellers to exert wrenches of a fixed pitch, (i.e., a thrust and a moment proportional to the thrust, both perpendicular to the plane of the rotor). Second, the robot system can be underactuated if the planes associated with each rotor are all parallel. In fact, this is generally the case in formation flight and it may be desirable to grasp the payload at multiple points, allowing the quadrotors to be in parallel horizontal planes. Third, the control of quadrotors necessitate dynamic models that reconcile the aerodynamics of flight with the mechanics of cooperative manipulation.

In this work we take advantage of the fact that we have access to many rotors to generate the thrust necessary to manipulate payloads. A similar concept is presented in [6], where the authors propose control laws that drive a distributed flight array consisting of many rotors along a desired trajectory. However, our control methods differ considerably as we are working with quadrotor robots and must derive feedback control laws based on the control inputs required by these robots. Similar to the concept of using multiple rotors in a flight array is the development of an aerial robot with more than four rotors (as in quadrotors), such as the commercially available *Falcon* with eight rotors from Ascending Technologies, GmBH [7].

A gripping mechanism is presented in this work that enables autonomous grasping of the payload by the quadrotors. Toward this design, we build upon considerable research in the area of climbing robots which generally rely on clinging to surface asperities via microspine arrays [8]. Similar designs with microspine arrays enable aerial vehicles to perch on vertical walls [9]. These robots do not require penetration to cling to the wall. However, in our work, the normal forces required to grasp objects are much higher compared to the shear forces that are exerted on the surfaces interfacing with the spines. Using similar microspine technology, we utilize the advantages of penetration in softer material such as wood and cardboard to attach to horizontal planar surfaces.

3 Dynamic Model

3.1 Coordinate Systems

The coordinate systems are shown in Fig. 1. The world frame, $\mathbf{W}$, is defined by axes x_W, y_W, and z_W with z_W pointing upward. We consider n quadrotors rigidly attached to a body frame, $\mathbf{B}$. It is assumed that the body frame axes are chosen as the principal axes of the entire system. Each quadrotor has an individual body frame, $\mathbf{Q}_i$, attached to its center of mass with z_{Qi} perpendicular to the plane of the rotors and pointing vertically up. Let (x_i, y_i, z_i) be the coordinates of the center of mass of the i^{th} quadrotor in $\mathbf{B}$ coordinates and ψ_i be the relative yaw angle. For this quadrotor, rotor 1 is on the positive x_{Qi}-axis, 2 on the positive y_{Qi}-axis, 3 on the negative x_{Qi}-axis, 4 on the negative y_{Qi}-axis. We require the z_{Qi} axes and z_B to be parallel. We use ZXY Euler angles to model the rotation of the body (and the quadrotors) in the world frame. To get from $\mathbf{W}$ to $\mathbf{B}$, we first rotate about z_W by the yaw angle, ψ, then rotate about the intermediate x-axis by the roll angle, ϕ,

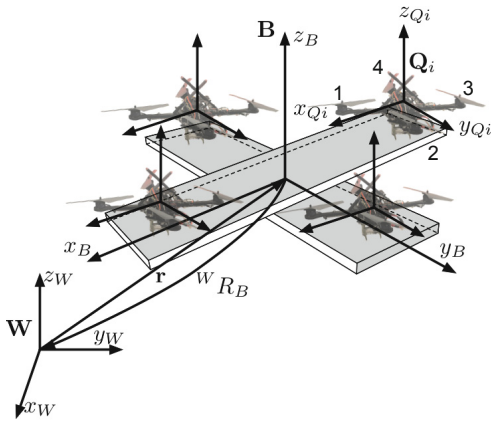


Fig. 1 The coordinate systems

and finally rotate about the y_B axis by the pitch angle, θ. The rotation matrix for transforming coordinates from $\mathbf{B}$ to $\mathbf{W}$ is given by

$$ {}^{W}R_B = \begin{bmatrix} c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + c\theta s\phi s\psi \\ c\theta s\psi + c\psi s\phi s\theta & c\phi c\psi & s\psi s\theta - c\psi c\theta s\phi \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix}, $$

where $c\theta$ and $s\theta$ denote $\cos\theta$ and $\sin\theta$, respectively, and similarly for ϕ and ψ. The components of angular velocity of the body in the body frame are p, q, and r. These values are related to the derivatives of the roll, pitch, and yaw angles according to:

$$ \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} c\theta & 0 & -c\phi s\theta \\ 0 & 1 & s\phi \\ s\theta & 0 & c\phi c\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}. $$

The position vector of the center of mass of the body in the world frame is denoted by $\mathbf{r}$. The rotation matrix, Euler angles, angular velocities, and position vector to the center of mass of the i^{th} quadrotor are denoted as ${}^{W}R_{Qi}$, $(\phi_i, \theta_i, \psi_i)$, (p_i, q_i, r_i), and $\mathbf{r}_i$, respectively.

3.2 Motor Model

Each rotor has an angular speed ω and produces a vertical force F according to

$$ F = k_F \omega^2. \tag{1} $$

Experimentation with a fixed rotor at steady-state shows that $k_F \approx 6.11 \times 10^{-8} N/rpm^2$. The rotors also produce a moment according to

$$ M = k_M \omega^2. $$

The constant, k_M, is determined to be about $1.5 \times 10^{-9} Nm/rpm^2$ by matching the performance of the simulation to the real system.

The results of a system identification exercise suggest that the rotor speed is related to the commanded speed by a first-order differential equation:

$$ \dot{\omega} = k_m(\omega^{des} - \omega). $$

This motor gain, k_m, is found to be about $20\,\text{s}^{-1}$ by matching the performance of the simulation to the real system. The desired angular velocities, ω^{des}, are limited to a minimum and maximum value determined through experimentation to be approximately 1200 rpm and 7800 rpm.

3.3 Equations of Motion

Each of the j rotors on each of the the i quadrotors produces a force, $F_{i,j}$, and moment, $M_{i,j}$, in the z_{Qi} direction. These rotor forces can be rewritten as a total force from each quadrotor $F_{q,i}$ as well as moments about each of the quadrotor's body frame axes:

$$\begin{bmatrix} F_{q,i} \\ M_{xq,i} \\ M_{yq,i} \\ M_{zq,i} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & L & 0 & -L \\ -L & 0 & L & 0 \\ \frac{k_M}{k_F} & -\frac{k_M}{k_F} & \frac{k_M}{k_F} & -\frac{k_M}{k_F} \end{bmatrix} \begin{bmatrix} F_{i,1} \\ F_{i,2} \\ F_{i,3} \\ F_{i,4} \end{bmatrix}, \tag{2}$$

where L is the distance from the axis of rotation of the rotors to the center of the quadrotor. The total force and moments on the system from the quadrotors in the body frame coordinates, $\mathbf{B}$, are:

$$\begin{bmatrix} F_B \\ M_{xB} \\ M_{yB} \\ M_{zB} \end{bmatrix} = \sum_i \begin{bmatrix} 1 & 0 & 0 & 0 \\ y_i & cos\psi_i & -sin\psi_i & 0 \\ -x_i & sin\psi_i & cos\psi_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} F_{q,i} \\ M_{xq.i} \\ M_{yq.i} \\ M_{zq.i} \end{bmatrix}. \tag{3}$$

Note that z_i is not present in (3) so this formulation allows for quadrotors in different planes. If we let m be the mass of the entire system and ignore air drag, then the equations governing the acceleration of the center of mass are simply:

$$m\ddot{\mathbf{r}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + {}^W R_B \begin{bmatrix} 0 \\ 0 \\ F_B \end{bmatrix}. \tag{4}$$

The moment of inertia matrix for the entire system referenced to the center of mass along the $x_B - y_B - z_B$ axes is denoted by I. We assume $x_B - y_B - z_B$ are chosen such that I is diagonal. The angular accelerations determined by the Euler equations are:

$$I \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} M_{xB} \\ M_{yB} \\ M_{zB} \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix}.$$

4 Control

4.1 Control Basis Vectors

The linear system in (3) defines four equations with $4n$ unknowns and can be rewritten as:

$$\left[F_B,\, M_{xB},\, M_{yB},\, M_{zB}\right]^{\mathrm{T}} = A\mathbf{u},$$

where $A \in \mathbb{R}^{4\times 4n}$ is fixed and determined by the relative positions and orientations of the n quadrotors. Here $\mathbf{u} \in \mathbb{R}^{4n}$ contains the four control inputs for each of the quadrotors:

$$\mathbf{u} = [F_{q,1}, M_{xq,1}, M_{yq,1}, M_{zq,1}, ..., F_{q,n}, M_{xq,n}, M_{yq,n}, M_{zq,n}]^T.$$

For a system with more than one quadrotor the linear system is underdetermined so we have a choice on how to achieve net forces and moments on the entire system. Here we choose an optimal control input $\mathbf{u}^*$ which achieves the desired net force and moments (F_B^{des}, M_{xB}^{des}, M_{yB}^{des}, and M_{zB}^{des}) while minimizing the cost function, J:

$$\mathbf{u}^* = \underset{\mathbf{u}}{\operatorname{argmin}}\{J | [F_B^{des}, M_{xB}^{des}, M_{yB}^{des}, M_{zB}^{des}]^{\mathrm{T}} = A\mathbf{u}\} \tag{5}$$

where

$$J = \sum_i w_{Fi} F_{q,i}^2 + w_{Mxi} M_{xq,i}^2 + w_{Myi} M_{yq,i}^2 + w_{Mzi} M_{zq.i}^2.$$

A natural way to treat the point-wise minimization of the function J is by choosing control inputs using the Moore-Penrose inverse. First we define $H \in \mathbb{R}^{4n\times 4n}$ so that $J = \|H\mathbf{u}\|_2^2$:

$$H = \operatorname{diag}\left(\sqrt{w_{F1}}, \sqrt{w_{Mx1}}, \sqrt{w_{My1}}, \sqrt{w_{Mz1}}, ..., \sqrt{w_{Fn}}, \sqrt{w_{Mxn}}, \sqrt{w_{Myn}}, \sqrt{w_{Mzn}}\right).$$

After algebraic manipulation we get:

$$\begin{aligned}\mathbf{u}^* &= H^{-1}(AH^{-1})^{+}[F_B^{des}, M_x^{des}, M_y^{des}, M_z^{des}]^{\mathrm{T}} \\ &= H^{-2}A^{\mathrm{T}}(AH^{-2}A^{\mathrm{T}})^{-1}[F_B^{des}, M_x^{des}, M_y^{des}, M_z^{des}]^{\mathrm{T}},\end{aligned} \tag{6}$$

where $^+$ denotes the Moore-Penrose inverse. It is instructive to the think of the columns of the matrix $H^{-1}(AH^{-1})^+$ as control basis vectors $\mathbf{u}_F$, $\mathbf{u}_{Mx}$, $\mathbf{u}_{My}$, and $\mathbf{u}_{Mz}$. Then the optimal control input can be written as:

$$\mathbf{u}^* = [\mathbf{u}_F, \mathbf{u}_{Mx}, \mathbf{u}_{My}, \mathbf{u}_{Mz}][F_B^{des}, M_x^{des}, M_y^{des}, M_z^{des}]^{\mathrm{T}}. \tag{7}$$

We now consider the special case in which all quadrotors are identical and axially symmetric meaning roll and pitch can be treated the same way. Indeed this is the case in our experimental testbed. In this case $w_{Fi} = w_F$, $w_{Mxi} = w_{Myi} = w_{Mxy}$, and $w_{Mzi} = w_{Mz}$. Consider the following term from (6) for this case:

$$AH^{-2}A^T = \begin{bmatrix} \frac{n}{w_F} & \frac{\sum y_i}{w_F} & -\frac{\sum x_i}{w_F} & 0 \\ \frac{\sum y_i}{w_F} & \frac{\sum y_i^2}{w_F} + \frac{n}{w_{Mxy}} & \frac{-\sum x_i y_i}{w_F} & 0 \\ -\frac{\sum x_i}{w_F} & \frac{-\sum x_i y_i}{w_F} & \frac{\sum x_i^2}{w_F} + \frac{n}{w_{Mxy}} & 0 \\ 0 & 0 & 0 & \frac{n}{w_{Mz}} \end{bmatrix}. \tag{8}$$

Here we can assume that the positions of the quadrotors dominate the mass properties of the entire structure since the quadrotors are heavier than what they can carry. The x and y locations of the center of mass of the payload and quadrotors together are close to that of just the quadrotors so $\sum x_i = \sum y_i = 0$. Additionally, $\sum x_i y_i = 0$, as the principle axes of the quadrotors are aligned with the principal axes of the structure. Therefore, all quadrotors contribute an equal force and yaw moment to produce a net body force or yaw moment:

$$\mathbf{u}_F = \frac{1}{n}\left[1, 0, 0, 0, ..., 1, 0, 0, 0\right]^{\mathrm{T}}$$
$$\mathbf{u}_{Mz} = \frac{1}{n}\left[0, 0, 0, 1, ..., 0, 0, 0, 1\right]^{\mathrm{T}}.$$

The control basis vectors for moments in pitch and roll reflect the tradeoff between the weighting factors:

$$\mathbf{u}_{Mx} = \frac{1}{\frac{w_{Mxy}}{w_F}\sum y_i^2 + n}\left[\frac{w_{Mxy}}{w_F}y_1, c\psi_1, s\psi_1, 0, ..., \frac{w_{Mxy}}{w_F}y_n, c\psi_n, s\psi_n, 0\right]^{\mathrm{T}}$$
$$\mathbf{u}_{My} = \frac{1}{\frac{w_{Mxy}}{w_F}\sum x_i^2 + n}\left[-\frac{w_{Mxy}}{w_F}x_1, -s\psi_1, c\psi_1, 0, ..., -\frac{w_{Mxy}}{w_F}x_n, -s\psi_n, c\psi_n, 0\right]^{\mathrm{T}}.$$

Here, an increase in the cost of individual quadrotor moments relative to the forces, w_{Mxy}/w_F, causes the individual body forces used to create a net body moment to increase and the individual body moments from each quadrotor to decrease. This ratio allows a user to tradeoff between the individual quadrotor force and moments used to create body moments in pitch and roll.

4.2 Attitude Control

To control the attitude of the body we use proportional derivative control laws that take the form:

$$\begin{aligned} M_{xB}^{\text{des}} &= k_{p,\phi}(\phi^{\text{des}} - \phi) + k_{d,\phi}(p^{\text{des}} - p) \\ M_{yB}^{\text{des}} &= k_{p,\theta}(\theta^{\text{des}} - \theta) + k_{d,\theta}(q^{\text{des}} - q) \\ M_{zB}^{\text{des}} &= k_{p,\psi}(\psi^{\text{des}} - \psi) + k_{d,\psi}(r^{\text{des}} - r). \end{aligned} \tag{9}$$

4.3 Hover Controller

Here we use pitch and roll angle to control position in the x_W and y_W plane, M_{zB}^{des} to control yaw angle, and F_B^{des} to control position along z_W. This approach is similar to that used for individual quadrotors in [10, 11, 12]. We let $\mathbf{r}_T(t)$ and $\psi_T(t)$ be the trajectory and yaw angle we are trying to track. Note that $\psi_T(t) = \psi_0$ for the hover controller. The command accelerations, $\ddot{\mathbf{r}}^{\text{des}}$, are calculated from PID feedback of the position error, $\mathbf{e} = (\mathbf{r}_T - \mathbf{r})$, as:

$$(\ddot{\mathbf{r}}_T - \ddot{\mathbf{r}}^{\text{des}}) + K_d(\dot{\mathbf{r}}_T - \dot{\mathbf{r}}) + K_p(\mathbf{r}_T - \mathbf{r}) + K_i \int (\mathbf{r}_T - \mathbf{r}) = \mathbf{0},$$

where $\dot{\mathbf{r}}_T = \ddot{\mathbf{r}}_T = \mathbf{0}$ for hover. We linearize (4) to get the relationship between the desired accelerations and roll and pitch angles:

$$\begin{aligned}
\ddot{r}_1^{\text{des}} &= g(\theta^{\text{des}} \cos\psi_T + \phi^{\text{des}} \sin\psi_T) \\
\ddot{r}_2^{\text{des}} &= g(\theta^{\text{des}} \sin\psi_T - \phi^{\text{des}} \cos\psi_T) \\
\ddot{r}_3^{\text{des}} &= \frac{1}{m} F_B^{\text{des}} - g.
\end{aligned}$$

These relationships are inverted to compute the desired roll and pitch angles for the attitude controller, from the desired accelerations, as well as F_B^{des}:

$$\begin{aligned}
\phi^{\text{des}} &= \frac{1}{g}(\ddot{r}_1^{\text{des}} \sin\psi_T - \ddot{r}_2^{\text{des}} \cos\psi_T) \\
\theta^{\text{des}} &= \frac{1}{g}(\ddot{r}_1^{\text{des}} \cos\psi_T + \ddot{r}_2^{\text{des}} \sin\psi_T) \\
F_B^{\text{des}} &= m(\ddot{r}_3^{\text{des}} + g).
\end{aligned}$$

We substitute these into (9) to yield the desired net body force and moments. From these quantities the control inputs for individual quadrotors are computed using the control basis vectors developed in Sec. 4.1:

$$\mathbf{u} = F_B^{\text{des}} \mathbf{u}_F + M_{xB}^{\text{des}} \mathbf{u}_{Mx} + M_{yB}^{\text{des}} \mathbf{u}_{My} + M_{zB}^{\text{des}} \mathbf{u}_{Mz}.$$

We then calculate the desired angular velocities for each of the $4n$ rotors from a linearization of (1,2) about the nominal hovering operating point.

4.4 3D Trajectory Control

The trajectory controller is used to follow 3D trajectories with modest accelerations so the near-hover assumptions hold. We use an approach similar to those described in [13, 12]. We have a method for calculating the closest point on the trajectory, $\mathbf{r}_T$, to the the current position, $\mathbf{r}$. Let the unit tangent vector of the trajectory associated with that point be $\hat{\mathbf{t}}$ and the desired velocity vector be $\dot{\mathbf{r}}_T$. We define the position and velocity errors as:

$$\mathbf{e}_p = ((\mathbf{r}_T - \mathbf{r}) \cdot \hat{\mathbf{n}})\hat{\mathbf{n}} + ((\mathbf{r}_T - \mathbf{r}) \cdot \hat{\mathbf{b}})\hat{\mathbf{b}}$$

and

$$\mathbf{e}_v = \dot{\mathbf{r}}_T - \dot{\mathbf{r}}.$$

Note that here we ignore position error in the tangent direction by only considering position error in the normal, $\hat{\mathbf{n}}$, and binormal, $\hat{\mathbf{b}}$, directions. This is done because

we are more concerned about reducing the cross-track error rather than error in the tangent direction of the trajectory.

We calculate the commanded acceleration, $\ddot{\mathbf{r}}^{\text{des}}$, from PD feedback of the position and velocity errors:

$$\ddot{\mathbf{r}}^{\text{des}} = K_p \mathbf{e}_p + K_d \mathbf{e}_v + \ddot{\mathbf{r}}_T.$$

Note that $\ddot{\mathbf{r}}_T$ represents feedforward terms on the desired accelerations. At low accelerations these terms can be ignored but at larger accelerations they can significantly improve controller performance. Finally, we use the process described in Sec. 4.3 to compute the desired angular velocities for each rotor.

4.5 Decentralized Control Law

We assume the quadrotors are attached rigidly to the body. As long as each quadrotor knows its fixed relative position and orientation with respect to the body and the goal of the body controller (hover location or desired trajectory) then this controller can be decentralized. If each quadrotor senses its own orientation and angular velocity then the orientation and angular velocity of the body are calculated as follows:

$$^{W}R_B = {}^{W}R_{Q_i}\,{}^{Q_i}R_B \quad \text{and} \quad [p, q, r]^{\mathrm{T}} = {}^{B}R_{Q_i}[p_i, q_i, r_i]^{\mathrm{T}}.$$

From the position and velocity of the i^{th} quadrotor, the position and velocity of the center of mass of the body are calculated as:

$$\begin{aligned}\mathbf{r} &= \mathbf{r}_i - {}^{W}R_B[x_i, y_i, z_i]^{\mathrm{T}} \\ \dot{\mathbf{r}} &= \dot{\mathbf{r}}_i - \omega_{\mathbf{B}} \times \left({}^{W}R_B[x_i, y_i, z_i]^{T}\right).\end{aligned}$$

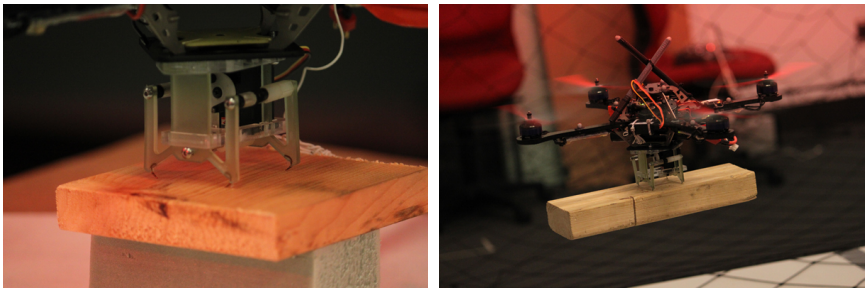
Each quadrotor then runs a local hover or velocity controller along with the attitude controller (9).

For completely centralized control, the state estimates of the n quadrotors are combined to create a single estimate of the state of the entire body from which the control inputs are computed. This averaging reduces the noise on the state estimate of the entire body and thus results in a cleaner control signal.

For the results presented in this work, we use a combination of the decentralized and centralized formulations. The position, velocity, and orientation estimates all come from a single source so the terms in the control input that depend on these values are calculated in a centralized fashion. The angular velocity is measured directly onboard each quadrotor so the terms in the control law that depend on the angular velocity are calculated using the decentralized method.

5 Gripping Mechanism

The gripping mechanism shown in Fig. 2 enables the quadrotors to attach to and release from the payload. The design is specialized to allow gripping of



(a) (b)

Fig. 2 Gripping mechanism engaged in wood. The assembly consists of a compliant polymer with embedded opposed microspines, linkages, and a servo mechanism for engaging and releasing (Fig. 2(a)). The gripping mechanism is attached to each quadrotor (Fig. 2(b)).

horizontal planar surfaces. This design choice is motivated by the problem definition, as all quadrotors are expected to attach to parallel horizontal planes with respect to the payload. It is also desirable for the gripper to be able to engage at any point on suitable materials. This gripper is designed to penetrate surfaces via opposed microspines actuated by a servo motor. Opposed spines allow large shear forces, which in turn allow a large normal force.

The gripper penetrates the surface of an object by driving four hooks into the plane with a servo motor. The hooks are standard fishing hooks which are accessible, cheap, sharp, and sufficiently strong for use in this application. Locating the hooks precisely with respect to the quadrotor is important as the the top face of the object must remain parallel to the quadrotor. To this end, we employ shape deposition manufacturing (SDM) to manufacture the full spine. The resulting compliant polymer spine introduces assistive compliance into the gripper. When the servo opens the gripper, the polymer acts as a spring that when released, aids the servo in penetrating the surface. After some penetration is achieved, the piece passes through its natural state and is stretched in the opposite direction. The piece responds with a restoring force that assists the servo on releasing the surface. We tested the gripping mechanism with a number of materials and found it to be effective in grasping soft to medium hardness woods, cardboard, high density foam, and carpet.

6 Results

In this section we describe results from two experimental trials designed to evaluate the performance of the controllers described in Sect. 4.3 and demonstrate cooperative grasping, manipulation, and transportation of payloads in 3D.

The hardware, software, and implementation details of the experiments follows. The position and orientation of the quadrotor is observed using a VICON motion capture system operating at 100 Hz [14]. The position is numerically differentiated

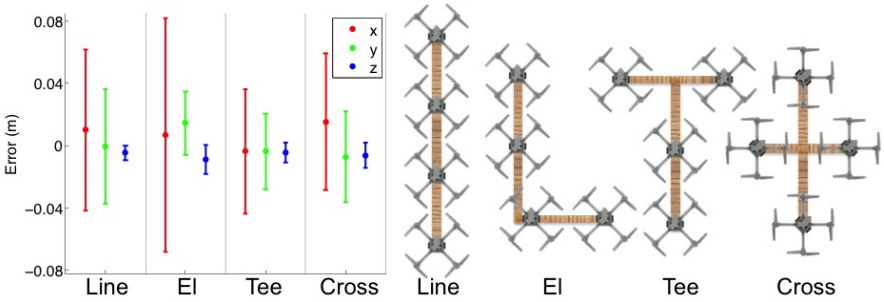


Fig. 3 Hover performance data for 40 seconds for four configurations. The center dot represents the mean error and the error bars represent one standard deviation. Graphical depictions of the four configurations.

Table 1 Mass and Inertia Properties

Configuration	I_{xx} (kg m^2)	I_{yy} (kg m^2)	m (kg)
Line	0.0095	0.73	3.33
El	0.079	0.50	3.33
Tee	0.082	0.43	3.33
Cross	0.11	0.19	3.23

to compute the linear velocity of the robot while the angular velocity is sensed onboard the quadrotor with a 3-axis rate gyro. The position, linear velocity, and orientation are available to MATLAB via ROS [15] and a ROS-MATLAB bridge [16]. All commands are computed in MATLAB using the latest state estimate at the rate of the VICON. The commands in MATLAB are bridged to ROS and the most recent command is sent to the robot via ZIGBEE at a fixed rate of 100 Hz. This fixed rate is due to the limited bandwidth of ZIGBEE (57.6 kbps). Commands sent to the robot consist of the gains, desired attitude, and thrust values described in Sect. 4.

The first experimental trial consists of a team of four quadrotors rigidly attached to different payload configurations (see Figures 3 and 5). For this test, we wish to focus on cooperative manipulation and transportation and as such use a payload structure built of wood with quadrotors attachments made via Velcro for easy rearranging. The total mass and x and y principal moments of inertia for each configuration (payload and quadrotors) are shown in Table 1. Note that the mass of a single quadrotor with a battery is about 500 g, so in each of these configurations the total payload is greater than 1.2 kg.

For each configuration, the control basis vectors are computed as described in Sect. 4.1 with $w_{Mxy}/w_F = 2$. We chose this ratio as our connection to the payload is stronger in resisting a force pulling it away from a surface than a moment in pitch or roll. Data for each configuration is shown in Fig. 3. Note that for each configuration, control along the x axis is intentionally performed with the body

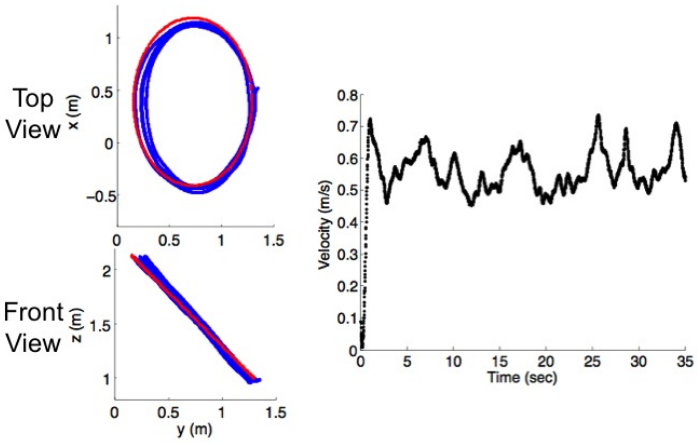


Fig. 4 Trajectory tracking data for the *Cross* configuration along a $0.8,$ m radius circle tilted at 45° from horizontal at 0.6 m/s

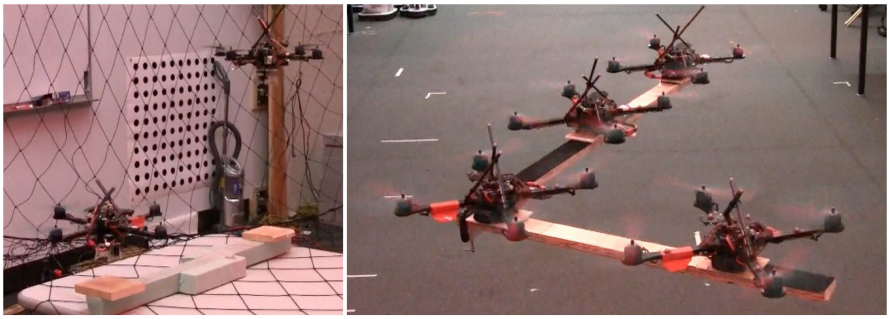


Fig. 5 Left: Image from an experiment with the gripping mechanism enabling cooperative grasping, manipulation, and transportation. Right: Four quadrotors carrying a payload in the *El* configuration. Videos of the experiments are available at http://tinyurl.com/penndars.

angle corresponding to the larger principal moment of inertia, I_{yy}. The performance along the x axis is worse than the y axis as expected. A large moment of inertia limits the bandwidth of the control on that angle. This decrease in attitude control performance leads to decreased position control performance along that axis. Here we note that position control for a single quadrotor is much better than for any of the multi-robot structures because their moments of inertia are much larger.

The trajectory tracking controller in Sect. 4.4 is implemented on the *Cross* configuration for which data is shown in Fig. 4. We see that the system performs well and controls to the desired trajectory in three-dimensions.

The gripping mechanism described in Sect. 5 is used on two quadrotors to pick up and transport an 0.8 m, 320 g structure as shown in Fig. 5. The quadrotors first descend to the structure and engage the gripping mechanism. The quadrotors ascend with the structure and fly twice along the same circular trajectory as in Fig. 4 at

$0.5\,\mathrm{m/s}$. Finally, the quadrotors descend to structures initial location, disengage the gripping mechanism, and depart.

7 Conclusions and Future Work

We addressed the problem of controlling multiple quadrotor robots that cooperatively grasp, manipulate, and transport a payload in three dimensions. We approach the problem by first developing a model for a single quadrotor and a team of quadrotors rigidly attached to a payload. We propose individual robot control laws defined with respect to the payload that stabilize the payload along three-dimensional trajectories. We detail the design of a gripping mechanism attached to each quadrotor that permits autonomous grasping of the payload. We conclude with an experimental study with teams of quadrotors cooperatively grasping, stabilizing, and transporting payloads of different configurations to desired positions and along three-dimensional trajectories.

We are currently working on autonomous system identification methods for multiple quadrotors picking up payloads with unknown masses and moments of inertia. We also plan to modify the gripping mechanism design to enable passive engagement so that a quadrotor can simply land on a surface to attach to it.

Acknowledgements. We would like to acknowledge Mark Cutkosky, Alexis Desbiens, Alan Asbeck, and Ben Kallman for their input in the choice of gripping strategies.

References

1. Michael, N., Fink, J., Kumar, V.: Cooperative manipulation and transportation with aerial robots. In: Proc. of Robotics: Science and Systems, Seattle, WA (June 2009)
2. Fink, J., Michael, N., Kim, S., Kumar, V.: Planning and control for cooperative manipulation and transportation with aerial robots. In: Proc. of the Int. Symposium of Robotics Research, Luzern, Switzerland (August 2009)
3. Salisbury, K., Roth, B.: Kinematics and force analysis of articulated mechanical hands. J. Mechanisms, Transmissions, and Automation in Design 105, 35–41 (1983)
4. Bicchi, A., Kumar, V.: Robotic grasping and contact. In: Proc. of the IEEE Int. Conf. on Robotics and Automation, San Francisco, CA, pp. 348–353 (April 2000)
5. Kumar, V., Waldron, K.J.: Analysis of omnidirectional gaits for walking machines for operation on uneven terrain. In: Proc. of Sym. on Theory and Practice of Robots and Manipulators, Udine, Italy, pp. 37–62 (September 1988)
6. Oung, R., Bourgault, F., Donovan, M., D'Andrea, R.: The distributed flight array. In: Proc. of the IEEE Int. Conf. on Robotics and Automation, Anchorage, AK, pp. 601–607 (May 2010)
7. Ascending Technologies, GmbH, `http://www.asctec.de`
8. Asbeck, A.T., Kim, S., Cutkosky, M.R.: Scaling hard vertical surfaces with compliant microspine arrays. In: Proc. of Robotics: Science and Systems, Cambridge, MA (June 2005)
9. Desbiens, A.L., Cutkosky, M.R.: Landing and perching on vertical surfaces with microspines for small unmanned air vehicles. J. Intell. Robotic Syst. 57, 313–327 (2010)

10. Bouabdallah, S.: Design and control of quadrotors with applications to autonomous flying. Ph.D. dissertation, Ecole Polytechnique Federale de Lausanne, Lausanne, Switzerland (February 2007)
11. Gurdan, D., Stumpf, J., Achtelik, M., Doth, K., Hirzinger, G., Rus, D.: Energy-efficient autonomous four-rotor flying robot controlled at 1 khz. In: Proc. of the IEEE Int. Conf. on Robotics and Automation, Roma, Italy (April 2007)
12. Michael, N., Mellinger, D., Lindsey, Q., Kumar, V.: The GRASP multiple micro UAV testbed. In: IEEE Robotics and Automation Magazine (September 2010)
13. Hoffmann, G., Waslander, S., Tomlin, C.: Quadrotor helicopter trajectory tracking control. In: AIAA Guidance, Navigation and Control Conference and Exhibit, Honolulu, Hawaii (April 2008)
14. Vicon Motion Systems, Inc., `http://www.vicon.com`
15. Robot Operating System (ROS), `http://www.ros.org`
16. ROS-Matlab Bridge, `http://github.com/nmichael/ipc-bridge`

Cooperative Transportation by Swarm Robots Using Pheromone Communication

Ryusuke Fujisawa, Hikaru Imamura, and Fumitoshi Matsuno

Abstract. Ants communicate with each other using pheromones, and their society is highly sophisticated. When foraging, they transport cooperatively with interplay of forces. The swarm is robust against changes in internal state, and shows flexibility in dealing with external problems. In this brief paper, we focus on the robot swarm that achieves cooperative transportation making use of ethanol as a substantial artificial pheromone. We also propose a swarm system with a newly developed algorithm that enables cooperative transportation of real robots. They will transport food to the nest analogous to the behaviour of a swarm of ants. Emphasis will be placed on the systematic task solution process. We present a number of experiments demonstrating the robustness and flexibility of the system and also confirming the effectiveness of the algorithm.

1 Introduction

1.1 Basic Characteristics of a Swarm

Generally, a swarm is a distributed autonomous system. It acts only according to local information in the given environment without any global information. An individual acts autonomously in the swarm, according to the circumstances [1]. Global behaviour emerges by interactions among individuals. Thus, these interactions are

Ryusuke Fujisawa
Hachinohe Institute of Technology, Hachinohe, Japan
e-mail: rfujisawa@hi-tech.ac.jp

Hikaru Imamura
DENSO, Japan

Fumitoshi Matsuno
Kyoto University, Kyoto, Japan
e-mail: matsuno@me.kyoto-u.ac.jp

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 559–570.
springerlink.com

fundamental term for formation of the swarm. Living organisms that form swarms communicate with each other to interact frequently. Therefore, swarms have robustness -a property for adapting to changes in the internal state- and also flexibility -a property for adapting to changes in the external state (e.g., the environment)[2].

1.2 Pheromone Communication in Ants

The social insects, such as ants and termites, are known to communicate with each other and form swarms using pheromones [3]. Ants form especially complex societies [3, 4, 5].

A pheromone is any chemical or set of chemicals produced by a living organism that transmits a message to other members of the same species [6]. In this paper, we focus on foraging behaviour of ants using a pheromone. When an ant finds and brings food back to the nest, it secretes a pheromone that forms a trail. The other ants trace the pheromone trail and reach the food. An ant stops to lay down the pheromone trail when it cannot find the food. The pheromone trail accordingly volatilises and/or diffuses into the environment, and thus information meaningless to the ants disappears. This is a simple but advanced communication method.

Itou *et al.* reported that the merits of this method are: 1) local and decentralised information management, and 2) self-propagation effect for information exchange [7]. Thus, pheromone communication is a suitable method for a distributed autonomous robot system. A number of individuals can converge at the food, communicating with each other. As each individual shares the purpose of action, i.e., collecting food, there is an interplay of forces that drive cooperative transportation. Each individual acts in accordance with a very simple corrective model, but the swarm shows advanced behaviours. Thus, "pheromone communication in the ant colony" and "emergence of cooperative transportation by interplay of forces" are useful to apply to robotics. Once we provide swarm robots with functions of task solution, the robots spontaneously find a method by interaction with each other, which is of remarkable significance.

1.3 Related Studies and Issues

Several studies, such as those of Sugawara *et al.* [8] and Garnier *et al.* [9], have demonstrated swarms of robots achieving foraging behaviour of ants with a virtual pheromone. In these studies, however, the swarm is inevitably non-autonomous in that it requires an external measurement system composed of a projector and a camera. Using substantial pheromones to control the robots is required to make the swarm autonomous.

Shimoyama *et al.* [10] achieved pheromone tracking behaviour using real insect antenna and substantial pheromone, but biomaterials cannot be handled easily by swarm robots. In addition, Shimoyama *et al.* did not pay particular attention to the swarm behaviour. A number of problems remain to be resolved in pheromone

communication robotics. Diffusion, an important factor for pheromone communication, is just one of these problems. To adjust the duration of the pheromone signal, it is necessary to change the concentration of a pheromone and/or mix with some other substance(s). In addition, only a few advantageous chemical sensors are available at present. Purnamadjaja *et al.* [11] studied swarm robots that communicate using two chemical substances, regulating a gas sensor in a sophisticated manner. However, as only one robot secretes the pheromone, this system achieves only one-sided communication.

There have been some studies on cooperative transportation by swarm robots. Dorigo *et al.* [12] developed a swarm robot system, "Swarm-bots", in which each robot has a grasping mechanism connecting the individual robots, which enables the robots to run through a gap or a trench. They accomplish cooperative transportation with the interplay of forces [13]. The robot itself and/or packages to be transported emit light, and the robots recognise what they should transport. Kube *et al.* also accomplish cooperative transportation of swarm robots [14]. The robots recognise the target to be transported with the installed light receiving element. These studies focused on the interplay of forces, but did not pay a great deal of attention to the indirect communication to advance emergence.

In our previous study, we achieved pheromone communication of swarm robots for recruitment behaviour [1]. In the present study, we propose cooperative transportation using pheromone communication as seen in ants. Experiments with the newly developed swarm robots indicated the effectiveness of pheromone communication in cooperative transportation, and suggested the robustness and flexibility of the swarm.

2 Swarm Behavior Algorithm

2.1 Swarm Behavior as Deterministic Finite Automaton

We have also been investigating a transportation algorithm. The result of previous studies [1, 15, 16, 17] will be applied to the new algorithm logic presented below. In previous studies, we focused on pheromone communication. Here, we concentrate on cooperative transportation with pheromone communication as seen in ants. The algorithm is described by the deterministic finite automaton shown in Fig. 1. The robots act in a completely autonomous manner by this algorithm. This algorithm premises that swarm robots search for food in a given field, and transport the food to the nest.

To design this algorithm, we defined 6 internal states, $S_i (i = 1, \cdots, 6)$; 10 perceptual cues (stimuli), $P_i (i = 1, \cdots, 10)$; and 6 effector cues (actions), $E_i (i = 1, \cdots, 6)$. We also assumed that there are many robots in the field, and that all agents can detect the direction of the nest as in the case of ants. As shown in Fig. 1, the agent whose state is S_i selects the action E_i. If the agent in state S_i detects the perceptual

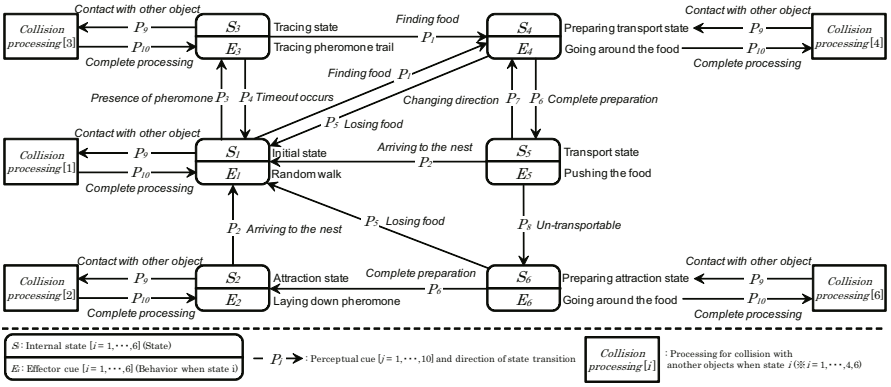


Fig. 1 Algorithm for cooperative transportation using pheromone communication

cue P_j, the state of the agent is transited to S_k. The details of the internal states S_i of the robot, perceptual cues P_i and effector cues E_i are as follows.

S_i: S_1, Search: the agent does not have any information on the food; S_2, Attraction: the agent has the location information on the food; S_3, Tracing: the agent has only the direction information on the location of the food; S_4, Pre-transportation: the agent has the location relationship of the nest and the food. As each robot can only push (not pull), it needs to run around the food so that the latter is on the line between the former and the nest; S_5, Transportation: the robot pushes the food; S_6, Pre-attraction: when the food will not move, the robot runs around it to find the way and returns to the nest.

P_i: P_1, Contact with food; P_2, Nest arrival; P_3, Presence of pheromone; P_4, Time-out occurrence; P_5, Losing pheromone trail; P_6, Completion of running around the food; P_7, Necessity of direction adjustment during transportation; P_8, Impossibility of transportation; P_9, Contact with other object (a wall or other robots); P_{10}, Completion of collision processing.

E_i: E_1, Random walk; E_2, Pheromone secretion; E_3, Following the pheromone path; E_4, Running around the food (to get behind it and on the line between the food and the nest); E_5, Pushing the food; E_6, Running around the food (to go to the nest).

2.2 Collision Processing

The robot perceives its external environment by contact with object(s), and acts in accordance with the collision algorithm shown in Table 1. The improvement of the previously developed collision algorithm [1] enabled us to avoid traffic jams on the pheromone trail. In addition, as the robots always make contact with each other during cooperative transportation, collision processing is not executed at the state S_5.

Table 1 Behaviour selection of algorithm for cooperative transportation after collision

Collision Processing	Contact position	Robot's behaviour after making contact with other object
Collision Processing [1]	front	rotation on-site after disengaging from contact point by reversing
	back	rotation on-site after disengaging from contact point by proceeding
Collision Processing [2]	front	stop on-site
	back	disengaging from contact point by proceeding
Collision Processing [3]	entire circumference	disengaging from contact point by reversing
Collision Processing [4]	front	disengaging from contact point by reversing
	back	disengaging from contact point by proceeding
Collision Processing [6]	front	disengaging from contact point by reversing
	back	disengaging from contact point by proceeding

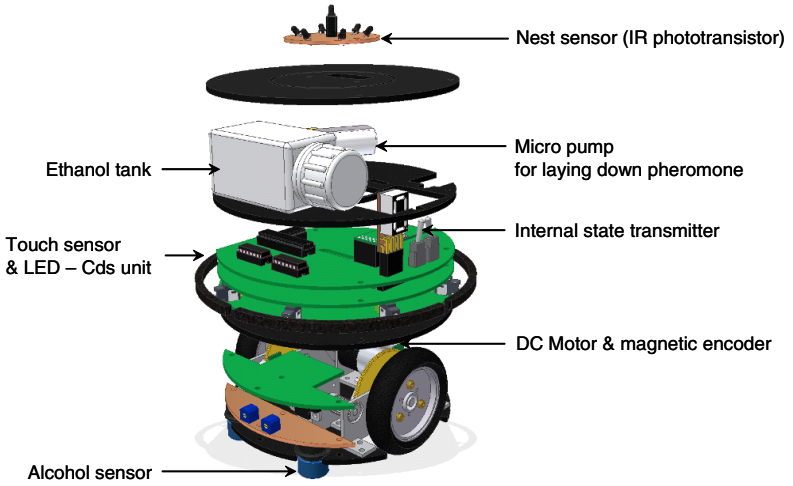


Fig. 2 Construction of the robot developed here (ARGOS01)

3 Construction of Robot

Figure 2 shows our newly developed swarm robot, ARGOS01. The robot has two active wheels and four castors. Its active wheels can be controlled independently so that the robot can move on a flat plane. The specifications of ARGOS01 are as follows: body diameter, 150 [mm]; height, 195 [mm]; weight, 1.26 [kg]; and maximum speed, 0.1 [m/s]. It has a Ni-MH battery (7.2 [V], 3900 [mAh]) at its centre. Four microcontrollers (Cypress Semiconductor) are installed in the robot. Its master controller is connected to 2 slave controllers by I^2C, and these three process the data from the sensors and control the motors. The other microcontroller sends the internal state to an external computer, which is used only to observe the internal state and swarm behaviour of the robots.

To implement the algorithm described in 2.1, a robot needs sensors to detect 10 perceptual cues (P_i $(i = 1, \cdots, 10)$), and actuators or mechanisms to carry out 6

effector cues (E_i $(i = 1, \cdots, 6)$). The following are the sensors and actuators installed on the robot.

- Nest sensor: detects the direction of the nest with infrared lamps.
- Touch sensor: detects contact with other objects.
- Photoreceptor unit: detects food (the food emits light by blue LED).
- Rotary encoder: detects success/failure of transportation.
- Alcohol sensor: detects pheromone.
- Pheromone secretion mechanism: lays down a pheromone trail.

The robot detects a collision with push switches at the side of its body. The food has blue LEDs at all around the body. With its photoreceptor unit, the robot looks for an object with LED light, and identifies it as food at contact. Based on the results of the previous experiments, we set the perceptible distance of the robot as around 300 [mm]. Rotary encoders at motors are used to detect the transport state. In this study, instead of a biological pheromone, we used ethanol, which is a volatile substance similar to the trail pheromone. The robot has alcohol sensors to detect the pheromone trail on the experimental field. A micropump on the robot secretes ethanol from the installed 50 [ml] tank.

The robot is composed of four layers supported by spacers. The first (bottom) layer has DC motors with a rotary encoder, alcohol sensors and an ethanol vent. The system substrate, the sensor substrate for the photoreceptor units and touch sensors, and the power and wireless communication substrate are installed at the second layer. The third layer has a micropump and a tank to secrete ethanol. The fourth (top) layer has the nest sensor to determine the direction of the nest.

4 Experiments and Results

Applying the designed swarm behaviour algorithm to the robot system, we have verified the effectiveness of pheromone communication in cooperative transportation. Figure 3 shows the experimental field: a 3,600 [mm]$\times$1,800 [mm] 2D flat plane surrounded by walls. The diameter of the food is 300 [mm] and that of the nest is 900 [mm]. Considering the perceptible distance of the robot, we set as the perceivable area of the robot a circle of radius 450 [mm] with the food at its centre. If the robot goes into the perceivable area, it can detect the direction of the food and make contact. As this study focused on cooperative transportation by the swarm robots, the weight of the food should be heavy enough so that one robot cannot move it by itself; we set the weight as 3.58 [kg]. This weight requires the cooperation of at least three robots. The transportation distance is 2,000 [mm]. We define a task solution when the robots transport the food to the nest. To trace the pheromone (100% ethanol) trail, we set a threshold on a value detected by the alcohol sensor. We set the signal duration time to trace the pheromone trail as 3 minutes. As a result of these settings, the robot continues to trace the pheromone trail for 3 minutes. The robot can lays down 5 trails in an experiment. When tank is empty, we change the robot with reserve robots. If the robot depletes the pheromone (ethanol), we change

the robot to continue the experiment. We also define that the robot recognises the P_8 (impossibility to transport) 20 [s] after starting pushing the food.

Figure 4 shows the results of an experiment with 10 robots. A, B and C are normal camera images; and A', B' and C' are thermographic images, which allow us to distinguish the pheromone trail with its lower temperature caused by heat evaporation of ethanol. The robots had laid down the pheromone trail at 10 [min]. At 20 [min], they had finished transporting the food to the nest in a concerted manner. The results of the experiment clearly showed both that the robots achieved cooperative transportation using the pheromone trail and that our new swarm behaviour algorithm works effectively. In the next chapter, setting the task solution time as the evaluation index, we will consider the robustness and flexibility of the swarm.

4.1 Effect of Pheromone Communication

We performed an experiment to determine the effectiveness of the pheromone communication on cooperative transportation. Figure 5 shows the relationship between task solution time [with or without pheromone communication] and the number of individuals; the vertical axis represents the average task solution time of 10 trials, and the horizontal axis represents the number of robots used. Error bars show the maximum and minimum in the experiments. With four or seven robots, the task solution time with pheromone communication was shorter than that without pheromone. The pheromone communication was effective when the density of the robots was low in the field. With a high density (10 robots) in the field, pheromone communication did not strongly influence the task solution time. This indicated that pheromone communication is not always necessary for cooperative transportation

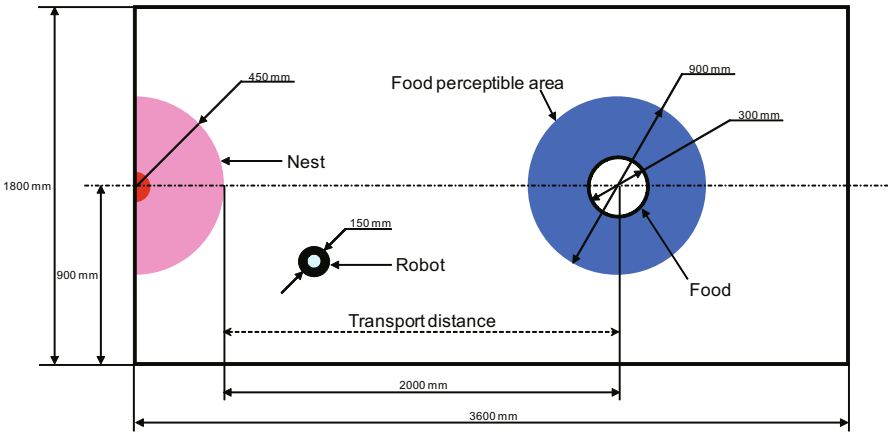


Fig. 3 Plane view of experimental field

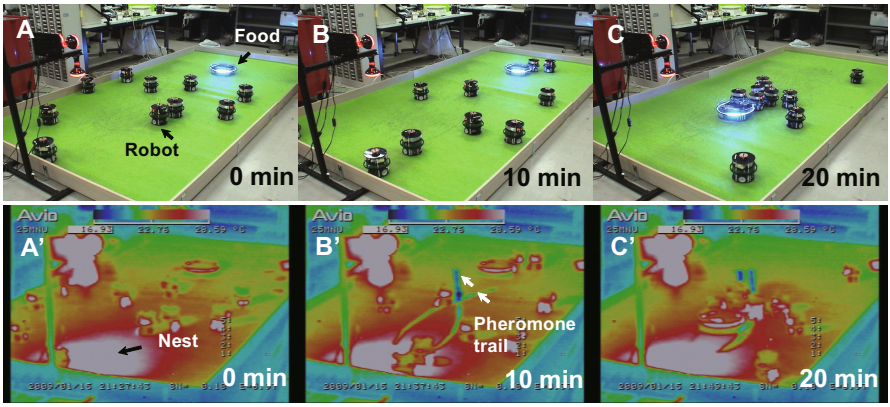


Fig. 4 Snapshots of experimental result by 10 robots using pheromone communication (A,B,C : Shot by digital cameraCA',B',C' : Shot by thermography)

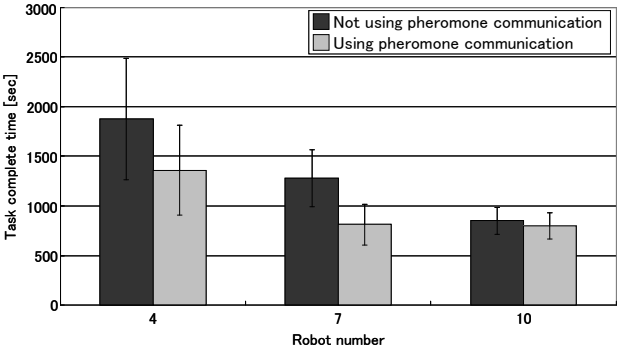


Fig. 5 Experimental result comparing cases with and without pheromone communication

when a sufficient number of robots are in the field as many robots find the food and begin cooperative transportation before laying down the pheromone trail.

Table 2 shows the average time of each event of 10 trials. When only a few robots (4 robots) are in the field; we found a marked difference between pheromone and non-pheromone communications. Without the pheromone communication, as the robots depend completely on the random walk to congregate at the food, they take a great deal of time to start pushing the food. With pheromone communication, it attracts the robots and they can begin to transport the food quickly. However, there were no clear effects of pheromone communication with ten robots in the field for the same reason as described above for the case of the task solution.

Table 2 Comparing average times of transportation starting in cases with and without pheromone communication

pheromone	number of robots 4 robots	7 robots	10 robots
With	625 [s]	335 [s]	361 [s]
Without	1057 [s]	612 [s]	355 [s]

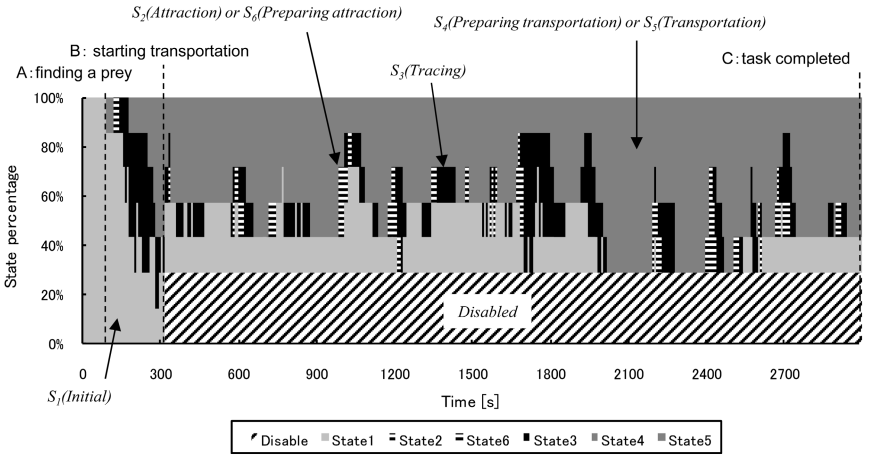


Fig. 6 Experimental result for robustness (percentage of disable robots : 2/7)

4.2 Robustness of the Swarm

To demonstrate the robustness of the swarm robots, we performed an experiment similar to that described above in 4.1 with seven robots using pheromone communication. During cooperative transport, we stopped two robots (S_5) in the swarm to determine whether the robots keep functioning as a swarm even when the swarm loses its soundness.

Figure 6 shows the typical experimental result. The vertical axis represents the percentage of internal states of the robots in the swarm, and the horizontal axis represents the experiment time. Light grey indicates that the robot is in state S_1 (search), horizontal stripes indicate S_2 (attraction) and S_6 (pre-attraction), black indicates S_3 (Tracing), dark grey indicates S_4 (transportation) and S_5 (pre-transportation). The disabled robots are indicated by diagonal stripes.

The elapsed time until the robots begin each action is shown in Table 3. Eighty-nine seconds after the start of the trial, a robot found the food and tried to transport it to the nest (Fig. 6-A). Four robots were attracted at the same time by the pheromone trail laid down by the first robot. As a result, four ¡five?¿ robots transported the food, using pheromone communication. When the food began to move (Fig. 6-B), we stopped two of the robots from functioning so as to impede cooperative transportation. The five robots still performed repeated actions of attraction, tracing and

Table 3 Events of robustness experiment

event	detail	time
A	finding food	89 [s]
B	starting transportation	309 [s]
C	task completed	2,990 [s]

transportation. At around 2,100 [s], all of the robots took part in transportation, and they successfully brought the food to the nest at 2,990 [s] (Fig. 6-C). Our intervention did not affect the systematic function of the swarm.

The results of this experiment indicated that this swarm robot system is robust against internal variation. However, it should be noted that this robustness is derived from system redundancy, i.e., there needs to be more robots than required to complete the task.

4.3 Flexibility of the Swarm

To determine the flexibility of the swarm, we performed an experiment with seven swarm robots implementing cooperative transportation using pheromone communication. At 60 [s] after starting cooperative transportation, we changed the weight of the food from 3.58 [kg] to 5.28 [kg], transportation of which requires at least four robots. Figure 7, which has the same axis representations and colour patterns as Fig. 6, shows the experimental results. Table 4 shows the elapsed time in the same way as in Table 3. A robot found the food and began trying to transport it at 233 [s] (Fig. 7-A). The robots laid down pheromone trails four times before the first cooperative transportation by three robots at around 600 [s]; soon after this, four robots were attracted at the same time (Fig. 7-B). Sixty seconds after smooth transportation started, we increased the weight of the food at 762 [s] (Fig. 7-C). Transportation stopped due to a lack of sufficient number of participants. However, the robots soon laid down the pheromone trail again at around 800 [s]. As a result of this, five robots aggregated, and they began to move the food again at 923 [s] (Fig. 7-D). At 1,509 [s], having transported the food to the nest, they had completed the task (Fig. 7-E).

Table 4 Events of flexibility experiment

event	detail	time
A	finding food	233 [s]
B	1st transportation start	702 [s]
C	changing food mass	762 [s]
D	2nd transportation start	923 [s]
E	task completed	1,509 [s]

Changes in the external state did not cause any systematic problems for the swarm robots, and they dealt with the changes in a concerted manner. Thus, our newly developed robots possess both robustness and flexibility as a swarm.

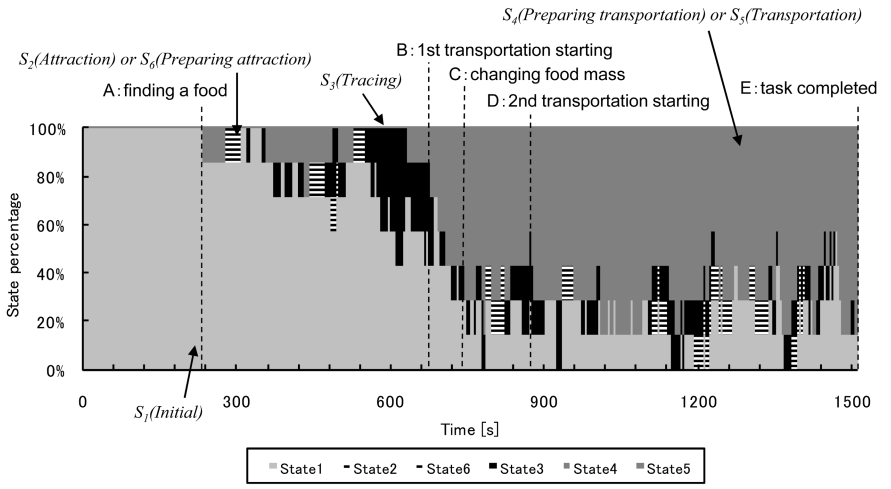


Fig. 7 Experimental results of flexibility experiment

5 Conclusion

As mentioned in 4.1, pheromone communication contributes to a reduction in task solution time, especially when the density of the robots is relatively low as shown in Fig. 5. This suggests the effectiveness of pheromone communication. Even when there are only a limited number of swarm robots in a given environment, they can solve the cooperative transportation task by making use of pheromone communication. This means that the effectiveness of pheromone communication is dependent on the density of individuals in the environment.

As shown in Fig. 6, the robustness is likely mainly due to redundancy, which also provides the swarm with flexibility (Fig. Fig. 7). When a swarm is not redundant, the swarm robots could solve a task, but they would still hardly shift smoothly to a new task, such as approaching another food. Our future studies on swarm robots will aim to clarify two crucial interrelationships: i.e., those among simultaneous multitask processing, the swarm behaviour and its redundancy, and those among robustness, flexibility and redundancy.

References

1. Fujisawa, R., Imamura, H., Hashimoto, T., Matsuno, F.: Development of multi-robots communicating by pheromone trail. Information Processing Society of Japan, Transactions on Mathematical Modeling and its Applications 2(2), 81–90 (2009) (in Japanese)
2. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence. Oxford University Press (1999)
3. Wilson, E.O.: Sociobiology: The New Synthesis. Belknap Press of Harvard University Press, Cambridge (1975)

4. Matsuka, M., Kitano, H., Matsumoto, T., Oono, M., Gokan, N.: Biology of Insect. Tamagawa University Publishing (1992) (in Japanese)
5. Katsumata, A., Ozaki, M.: Chemical communications in ants. Journal of Comparative Physiology and Biochemistry 24(1), 3–17 (2007)
6. Karlson, P., Butenandt, A.: Pheromones (ectohormones) in insects. Annual Review of Entomology 4, 39–58 (1959)
7. Ito, K. (ed.): Emergence of Intelligence. NTT Publishing (2000) (in Japanese)
8. Sugawara, K., Kazama, T., Watanabe, T.: Foraging behavior of interacting robots with virtual pheromone. In: Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS 2004), vol. 3, pp. 3074–3079 (2004)
9. Garnier, S., Tache, F., Combe, M., Grimal, A., Theraulaz, G.: Alice in pheromone land: An experimental setup for the study of ant-like robots. In: Proc. IEEE Swarm Intelligence Symposium (SIS 2007), April 1-5, pp. 37–44 (2007)
10. Shimoyama, I., Kanzaki, R.: Biological type micromachine. Journal of Society of Biomechanisms 22(4), 152–157 (1998) (in Japanese)
11. Purnamadjaja, A.H., Russell, R.A.: Guiding robots behaviors using pheromone communication. Autonomous Robots 23(2), 113–130 (2007)
12. Dorigo, M., Tuci, E., Groß, R., Trianni, V., Labella, T.H., Nouyan, S., Ampatzis, C., Deneubourg, J.-L., Baldassarre, G., Nolfi, S., Mondada, F., Floreano, D., Gambardella, L.M.: The SWARM-BOTS Project. In: Şahin, E., Spears, W.M. (eds.) Swarm Robotics 2004. LNCS, vol. 3342, pp. 31–44. Springer, Heidelberg (2005)
13. Mondada, F., Bonani, M., Guignard, A., Magnenat, S., Studer, C., Floreano, D.: Superlinear Physical Performances in a SWARM-BOT. In: Capcarrère, M.S., Freitas, A.A., Bentley, P.J., Johnson, C.G., Timmis, J. (eds.) ECAL 2005. LNCS (LNAI), vol. 3630, pp. 282–291. Springer, Heidelberg (2005)
14. Kube, C.R., Bonabeau, E.: Cooperative transport by ants and robots. Robotics and Autonomous Systems 30(1-2), 85–101 (2000)
15. Fujisawa, R., Imamura, H., Hashimoto, T., Matsuno, F.: Communication Using Pheromone Field for Multiple Robots. In: Proc. IEEE/RSJ 2008 International Conference on Intelligent Robots and Systems (2008)
16. Fujisawa, R., Dobata, S., Kubota, D., Imamura, H., Matsuno, F.: Dependency by Concentration of Pheromone Trail for Multiple Robots. In: Dorigo, M., Birattari, M., Blum, C., Clerc, M., Stützle, T., Winfield, A.F.T. (eds.) ANTS 2008. LNCS, vol. 5217, pp. 283–290. Springer, Heidelberg (2008)
17. Fujisawa, R., Imamura, H., Hashimoto, T., Matsuno, F.: Swarm Intelligence of Multi-Robot Using Pheromone Trail. In: Proc. The 2nd Internal Symposium on Mobiligence, pp. 203–206 (2007)

Socially-Mediated Negotiation for Obstacle Avoidance in Collective Transport

Eliseo Ferrante, Manuele Brambilla, Mauro Birattari, and Marco Dorigo

Abstract. In this paper, we present a novel method for performing collective transport in the presence of obstacles. Three robots are physically connected to an object to be transported from a start to a goal location. The task is particularly challenging because the robots have a heterogeneous perception of the environment. In fact, the goal and the obstacles can be perceived only by some of the robots. Hence, the task requires appropriate negotiation of the direction among the robots. We developed a novel negotiation strategy in order to tackle this challenge. We perform experiments in simulation. In the experiments, we analyze efficiency in an environment with only one obstacle, and robustness in an environment with several obstacles.

1 Introduction

The ability of robots to move in a coordinated fashion is of central importance for the multi-robot research community. Research in coordinated motion can be divided in two categories. In the first category we find works in *multi-robot formation*, were no physical connection between robots is assumed. In the second category, we find works in *collective transport* and *coordinated motion* where there is a physical connection between the robots or between the robots and the object to be transported.

Works in multi-robot formation have been documented in some surveys [1, 2], where the authors compare centralized vs. decentralized approaches. The most studied decentralized method in this area are social potentials [3] and artificial physics [4].

In collective transport, a group of robots has to cooperate in order to transport an object that, because of its weight, cannot be transported by a single robot. The task we are interested in is particularly challenging because communication between the

Eliseo Ferrante · Manuele Brambilla · Mauro Birattari · Marco Dorigo
IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
e-mail: {eferrant,mbrambil,mbiro,mdorigo}@ulb.ac.be

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 571–583.
springerlink.com

robots is only local, robots have no access to global information and they coordinate using a decentralized approach.

Several works on collective transport were developed using centralized approaches like leader-following behaviors. In these works [5, 6, 7], a group of robots is able to collectively push/pull an object. In order to coordinate their movements, the robots follow a leader that has the knowledge of the goal area or of the path.

Balch [8] was one of the first to study the impact of communication in multi-robot systems. Later, Donald et al. [9] and Yamada et al. [10], studied collective transport with limited communication. In the first work [9], robots had to transport an object without a goal location, whereas in the second work [10] robots had to carry an heavy object towards a common goal determined by a light emitter (photo-taxis).

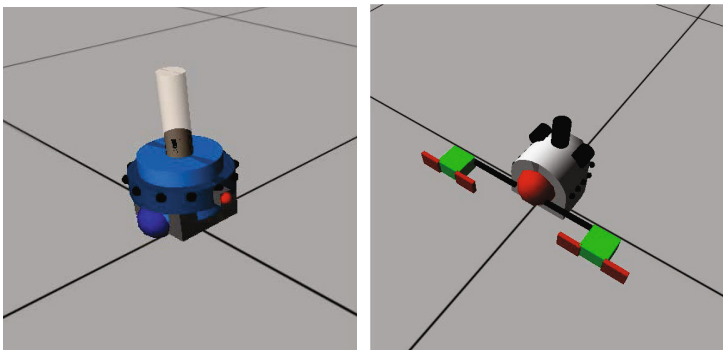
Campo et al. [11] investigated the use of goal negotiation strategies for performing collective transport to a given goal location. The robots used by the authors had only a noisy perception of the goal, or they were not able to perceive the goal at all. Furthermore, each of the robots used LEDs and an on-board camera to perceive the orientation of the other robots, and used this information to compute an average direction of motion.

Groß and Dorigo [12] used artificial evolution to synthesize a neural network to achieve collective transport. Their robots were able to cope with objects of different size and weight as well as with groups of different size (from 4 to 16). The authors were able to obtain three different transport strategies. In the first one, the robots directly connect to the object and pull it. In the second one, the robots connect to each other (self-assembly) and to the object in order to pull it. In the third strategy, the robots create a physical loop around the object. This last strategy involves a high number of robots and a small (but heavy) object.

Trianni et al. [13] studied a task similar to obstacle avoidance in collective transport. They call it *collective hole-avoidance*. In their task, robots are physically connected to each other, and they have to navigate in an environment with holes. The authors used artificial evolution for the synthesis of robots' neural network controllers, and studied different communication strategies among the robots: no direct communication, handcrafted signaling and communication induced by artificial evolution. Differently from the work described in this paper, in Trianni et al. [13] no object had to be transported. Furthermore, the robots did not have a specific goal direction on where to go but they were rather exploring the environment while avoiding holes.

Baldassarre et al. [14] studied a task similar to the one studied by in Trianni et al. [13]. In their study, physically connected robots collectively navigate in an environment with obstacles, furrows and holes and a light source to be found. The authors used artificial evolution to synthetize a behavior able to integrate these three sub-behaviors in a coherent fashion: collective motion, collective obstacle avoidance and collective light approaching. However, the synthetized behavior heavily exploited the traction sensor, a specialized sensor that is able to detect forces exerted by the connected robots and that might not be available on all robotics platforms.

In this paper, a group of three simulated robots have to transport an object from a start to a goal location in an environment with obstacles. Almost all tasks studied so far in the literature consider collective transport in an obstacle-free environment



(a) (b)

Fig. 1 Picture of the simulated foot-bot (a) and of the hand-bot, which is the irregularly shaped object to be carried (b)

where a goal location is given, with two notable exceptions. In Trianni et al. [13], the environment is cluttered but a goal direction is not given. In Baldassarre et al. [14], both elements can be present at the same time but the synthesized solution, rather than exploiting direct local communication, uses instead indirect communication via specialized hardware. In this paper, we propose a novel negotiation strategy for collective transport in presence of both obstacles and of a goal. The proposed negotiation strategy is based on local direct communication.

The remaining of the paper is organized as follows. In Section 2, we describe the task and the simulated robots. In Section 3, we describe the method we propose to design the controller. In Section 4, we present experimental results, whereas in Section 5 we conclude and sketch possible future works.

2 Task Definition

A group of three identical simulated mobile robots (like the one depicted in Figure 1a) attach to an irregularly shaped object (b). The task is to collectively transport the object from an initial to a goal location. The robots we used are modeled after the foot-bot robot [15], developed within the Swarmanoid project[1]. The irregularly shaped object is an object which cannot be grasped through its entire perimeter but only in certain regions. In our case, it is another simulated robot of the Swarmanoid project, the hand-bot (Figure 1b) [16]. This robot is a manipulator that does not have locomotion capabilities and thus needs to be carried by the foot-bots. In this task, the hand-bot is passive during the entire process.

[1] `http://www.swarmanoid.org`

The environment is an arena where a number of cuboid-shaped obstacles are present, each with an arbitrary position and orientation. Each of the three simulated mobile robots is equipped with a number of sensors and actuators. We considered and used only the following sensors and actuators: i) a light sensor, that is able to perceive the intensity of the light coming from different directions around the robot; ii) a distance scanner, that is used to obtain distance and angular values from the robot to other objects in the environment [17]; iii) a range and bearing communication system, with which a robot can send a message to other nearby robots in line of sight [18]; iv) a gripper, that is used to physically connect to the transported robot considered in the experiment; v) a turret actuator which, when set to active mode, can be used to rotate the gripper installed on a rotating ring or, when set to passive mode, can freely rotate in accordance with the speed of the wheels when the gripper is gripping an heavy object; vi) a wheels actuator, that is used to control independently the speed of the left and right wheels of the robot. The light sensor and the distance scanner sensor are not perfect but subject to a certain degree of noise. The range and bearing communication device can perceive messages coming from up to 4 meters away, more than enough to guarantee communication between the robots when connected. The distance scanner has a range of 1.5 meters.

In the experiments, we also place a light source in a fixed position in the environment behind the goal area. The light source has a high intensity such that it can be perceived by all the robots. The aim of the light source is to act as a common environmental cue, which is used as an implicit and shared reference frame by the robots.

For the sake of simplicity, the robots use the direction of the light source as the goal direction, that is they perform photo-taxis. Since the proposed methodology is not restricted to this case, in Section 3 we consider the goal direction and the environmental cue (or light) direction as two separated concepts. In the case where the goal direction is different from the light direction, the robot might need to be equipped with a separate sensor to detect the goal direction.

The presence of obstacles and the need to move to a given goal location create the need of handling conflicting individual decisions, which can be produced due to the non uniform perception of the environment.

For each individual robot, information of the following nature can be available at a given time:

No information: The goal is not perceived, for example because occluded by obstacles, and no obstacles are perceived as well.

Goal only: Only the goal is perceived, hence the robot moves towards it.

Obstacle only: The robot does not perceive the goal. However, it perceives an obstacle, hence it has to avoid it. At the same time, it has to inform other robots about the obstacle avoidance direction.

Goal and obstacle: The robot perceives both the goal and an obstacle. The direction of movement, considered by the robot and communicated to the other robots, has to take into account both these elements.

Table 1 Explanation of the notation used to describe the two behaviors

Notation	Meaning	Behavior
θ_P	Preferred direction when in $S_{stubborn}$ state	Social mediation
θ_S	Socially mediated angle $\theta_S \leftarrow \angle \sum_{i=0}^{k} e^{j\theta_i}$	Social mediation, collective transport
θ_0	Direction sent by social mediation behavior: θ_S in S_{social} state or θ_P in $S_{stubborn}$ state	Social mediation
$\theta_1 \dots \theta_k$	Direction received from the k neighbors	Social mediation
θ_G	Goal direction	Collective transport
θ_{CO}	Obstacle direction	Collective transport
θ_{OA}	Obstacle avoidance direction. It has to take into account also θ_G if the goal is perceived.	Collective transport
θ_F	Direction of the shared environmental cue. All other directions are always relative to this	Collective transport
$\overline{\theta}_S$	Weighted time average of θ_S	Collective transport

We now have all the elements to introduce the method we propose for tackling this task.

3 Method

In this section we first introduce the main idea behind the proposed method. Subsequently, we present the *collective transport* behavior, which we decomposed into three sub-behaviors: *go to goal*, *obstacle avoidance* and *social mediation*.

In the following, we will use a certain notation to denote directional information used in the behaviors. This is explained and summarized in Table 1.

The low level behaviors *go to goal* and *obstacle avoidance* are used as follows. The go to goal behavior is used to query sensors and to obtain a goal direction, denoted as θ_G; the obstacle avoidance behavior is used to detect the presence of obstacles and the angle θ_{CO} of the closest one. The *social mediation* and *collective transport* behaviors are the core focus of the proposed method.

The social mediation behavior, explained in Section 3.1, is used to negotiate the direction to be followed in collective transport. This is needed since, as explained in Section 2, different robots in the group can have access to conflicting information, for example one might perceive the goal as well as an obstacle while the others might perceive just the goal. Furthermore, when two or more robots perceive an obstacle, they can perceive it from different angles.

Once a collective decision has been made on the direction to be followed, this is used by the *collective transport* behavior, explained more in details in Section 3.2.

3.1 Social Mediation

The social mediation behavior is responsible for the negotiation of the direction of motion. The behavior uses the directional information given by θ_S and θ_P: θ_S

represents a socially mediated heading direction and θ_P the robot desired heading direction. The main idea behind the algorithm is the following. When a robot in the group has no information (i.e., it does not have any information on the goal or on the obstacles), it has an internal *state* set to S_{social}. In this state, the robot acts as a repeater, that is, it computes θ_S, the average of the direction information available to its neighbors, and it sends this value around. However, when information (such as on the obstacle) is available to the robot, its internal *state* is set to $S_{stubborn}$. In this state, it will relay its own preferred direction θ_P (for example the obstacle avoidance direction) instead of θ_S. When all other robots are still sending θ_S, the opinion of the stubborn robot will soon diffuse in the entire group, that is θ_S through the group will converge to θ_P. The internal state of this behavior can be changed only by the overall collective transport behavior, as explained in Section 3.2. Algorithm 1 depicts the steps executed at every control step.

Algorithm 1. Social mediation control loop

1: $Receive(\theta_1, \theta_2, \ldots, \theta_k)$
2: $\theta_S \leftarrow \angle \sum_{i=0}^{k} e^{j\theta_i}$
3: **if** $state = S_{social}$ **then**
4: $\quad \theta_0 = \theta_S$
5: **else**
6: $\quad \theta_0 = \theta_P$
7: **end if**
8: $Send(\theta_0)$

At the beginning of the control loop, the robot receives the heading direction information $\theta_1, \theta_2, \ldots, \theta_k$ of its neighbors, where k is the number of neighbors. Communication is restricted to all neighboring robots in line of sight [18], as we are using the range and bearing communication mechanism. Due to this restriction, the

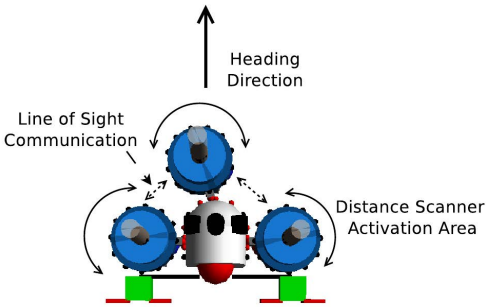


Fig. 2 The carried robot and the carrying robots. The circular arrows show the area of the distance scanner which is active for sensing, whereas dashed straight arrow show the line of sight communication relationships.

robot attached at the center has $k = 2$ neighbors, whereas the other two have $k = 1$ neighbor (see Figure 2). The socially mediated heading θ_S is computed by averaging the directional information ($\angle$ means "the angle of") received by the neighbors (line 2), with the robot's own information θ_0.

By using the mechanism depicted above, we are solving the issue of how to diffuse a heading direction information, perceived only by one robot, through the entire group, without the need of special signaling. This allows all robots in a group to be aware of the avoidance direction of an obstacle, even if only one member of the group can perceive the obstacle.

In the following section, we describe how this mechanism is used to achieve effective collective transport with obstacle avoidance.

3.2 Collective Transport and Obstacle Avoidance

In this section we present the behavior responsible for collective transport with obstacle avoidance. This behavior uses the directional information computed in the social mediation behavior. In this behavior, θ_O denotes the direction of the obstacle (if perceived), θ_G denotes the goal direction (if the goal is perceived) and θ_{OA} denotes the obstacle avoidance direction (see table 1 for a summary). This directional information is always considered as relative to the direction of the shared environmental cue, denoted with θ_F and represented in our case by the light source.

Algorithm 2. Collective transport control loop

1: $[\theta_G, goalPerceived] \leftarrow PerceiveGoal()$
2: $[\theta_{CO}, d, obstaclePerceived] \leftarrow PerceiveObstacle()$
3: **if** *goalPerceived* **or** *obstaclePerceived* **then**
4: $\quad SocialMediation :: state \leftarrow S_{stubborn}$
5: **else**
6: $\quad SocialMediation :: state \leftarrow S_{social}$
7: **end if**
8: **if** *goalPerceived* **then**
9: $\quad SocialMediation :: \theta_P \leftarrow \theta_G$
10: **end if**
11: **if** *obstaclePerceived* **then**
12: $\quad$ **if** *goalPerceived* **then**
13: $\quad\quad w \leftarrow -\frac{d}{\min(d, d_{max})} + 1$
14: $\quad$ **else**
15: $\quad\quad w \leftarrow 1$
16: $\quad$ **end if**
17: $\quad \theta_{OA} \leftarrow \angle w \cdot e^{j\theta_O + \pi} + (1 - w) \cdot e^{j\theta_G}$
18: $\quad SocialMediation :: \theta_P \leftarrow \theta_{OA}$
19: **end if**
20: $SocialMediation :: ControlStep()$
21: $\overline{\theta}_S \leftarrow \angle(1 - \alpha) \cdot e^{j\overline{\theta}_S} + \alpha \cdot e^{j\theta_S}$
22: $MotionControl(\overline{\theta}_S)$

At the beginning of Algorithm 2, sensors are queried to detect whether the goal and/or obstacles are perceived (lines 1-2). The corresponding directions θ_G, corresponding to the goal direction, and θ_{CO}, corresponding to the angle of the closest obstacle, are also queried.

According to the information available to the robot (see Section 2) the internal state of the social mediation behavior is set (lines 3-7). If the robot perceives an obstacle with its distance scanner its state is set to $S_{stubborn}$. The same happens when the robot perceives the goal. In all other cases, that is when both the goal and the obstacles are not perceived, the state is set to S_{social}.

If the goal is perceived, the robot simply informs the others about the goal by setting its desired direction θ_P to the goal direction θ_G (line 9).

In case an obstacle is perceived two things can happen. If no goal direction θ_G is available, the robot simply tries to avoid the obstacle using the angle $\theta_{OA} = \theta_{CO} + \pi$ and by setting $w = 1$ (line 15). If, however, both the obstacle and the goal are perceived, the robot needs to compute the desired direction according to this two pieces of information: θ_O and θ_G are thus averaged using a weighted average and the result is assigned to θ_{OA} (lines 17). The weighted average uses a weight $w \in [0,1]$ dependent on the distance between the robot and the obstacle (line 13) which represents how urgent it is to avoid obstacles: it is 1 when the obstacle is very close ($d = 0$) and 0 when it is far away ($d = d_{max}$, the maximal perception range of the obstacle avoidance behavior). We set $d_{max} = 0.75$ meters, half of the maximal range of the distance scanner, and we use the min operator to avoid negative values for w. The angle θ_{OA} is then assigned to the desired direction θ_P of the social mediation behavior (line 18).

Once θ_P is computed, the control step of the social mediation behavior is executed (line 20). As a result, the angle θ_S is computed by the social mediation behavior. This angle is then filtered by computing a time average (line 21) to filter out the effect of noise.

Finally, the motion control logic uses the filtered socially mediated direction $\overline{\theta}_S$ as a reference direction to be followed. The robot first converts the socially mediated direction to its local frame of reference using the common environmental cue direction θ_F. All robots then compute the left and right wheels speed in the following way:

$$N_L = u + \omega b \, , N_R = u - \omega b \, , \omega = K_p \overline{\theta}_S,$$

where N_L, N_R are the wheels rotation speed of the left/right wheel speed respectively, b is the distance between the center of the robot and each of the wheels, u and ω are the forward and angular velocities respectively. The forward velocity u is kept constant, whereas we vary the angular velocity ω proportionally to the socially mediated direction $\overline{\theta}_S$ to be followed, where K_p is a proportional factor (we assume a clockwise convention for the angles). Furthermore, the motion control rule considers the robot attached to the left as the left wheel of the compound system and the robot attached to the right as the right wheel. This assumes that the two robots have always the direction of the wheels axis parallel to each other and it is ensured by the fact that we set the turret to active mode. Hence, the robot attached to the left of the

compound will set both wheels speed to N_L, whereas the robot to the right will set them to N_R. The robot at the center can instead independently control its own left and right wheels depending on value computed by the motion control logic. The turret of the central robot, which is set to passive mode, freely rotates passively and follows the dynamics of the compound and the one imposed by the wheels.

To summarize the idea, the collective transport behavior interacts with the social mediation behavior to obtain a socially mediated direction $\overline{\theta}_S$ which is consistent in the group and allows a coherent motion. The social mediation behavior needs to be set in the appropriate state ($S_{stubborn}$ or S_{social}), according to which information is available to the robot. It also needs the direction θ_P to be sent to the neighbors in case it is in $S_{stubborn}$ state. θ_P can be the direction to the goal, the obstacle avoidance direction or the direction which takes into account both the goal and the obstacles. The behavior achieves coherent collective motion even in case of conflicting opinions, since the motion control logic uses the socially mediated direction, that is the direction negotiated through the entire group, as the target direction to be followed.

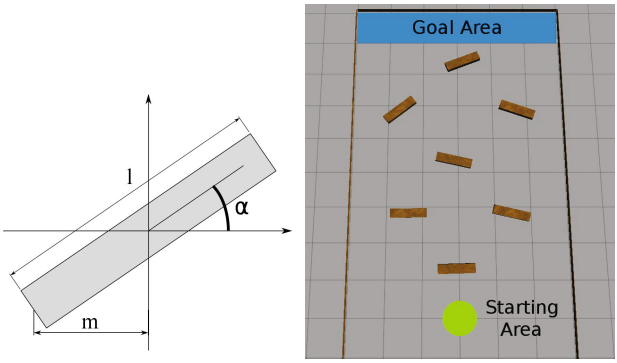
4 Experiments and Results

We performed three sets of experiments. The first two sets consider a simple environment, where we position an obstacle at the center of the arena with varying angle α (see Figure 3a). For each setting, we executed 100 runs. Our prior expectation is that the more α tends to 0, the longer it takes to avoid the obstacle in collective transport. We also expect that the proposed behavior is robust enough to always accomplish the task (move from an initial to a goal location, see Figure 3b) in this simplified setting. We hence report the completion times as a function of α. The difference between the first and the second set of experiments is that in the first set we just analyze the impact of the angle α by keeping the projected size of the obstacle m fixed (Figure 3a), whereas in the second set we also analyze the impact of the varying projected size, keeping l fixed. Execution times are reported in time-steps. Each simulated second corresponds to 10 time-steps.

In the third and last set of experiments, we generate at random some more complex environments, of the type depicted in Figure 3b. We report the success rate of the behavior. We executed a total of 1000 runs, where in each run the angle and an offset of the position of each obstacle is generated at random.

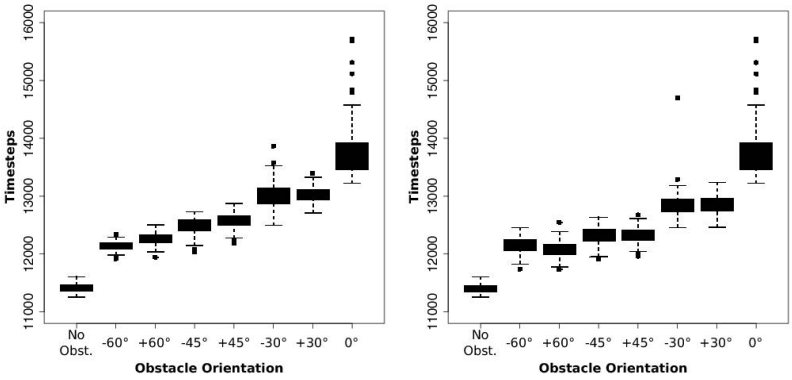
Figure 4 shows the results for the first two sets of experiments performed in the simple environments. As we can see, the initial hypothesis can be accepted, as the execution times solely depends on α and not on the projected length m of the obstacle. In fact, execution times increase with increasing values for α. The more the obstacle is perpendicular to the direction of motion, the longer it takes for the robots to to perform obstacle avoidance.

The case $\alpha = 0$ is particularly problematic. Average times are much higher, and many more outliers are present (not fully shown due to scale differences). This is explained by the fact that, when the obstacle is perpendicular to the direction of



(a) (b)

Fig. 3 (a) The controlled obstacle's parameter in the first two sets of experiments and (b) an example of complex environment. S denotes the starting area, G the goal area.



(a) (b)

Fig. 4 Box plot of completion time for the experiment set with fixed m (a) and for fixed l (b)

motion, i.e. $\alpha = 0$, the avoidance direction θ_{OA} takes some time to converge to one of the two possible obstacle avoidance sides. All the runs were successful and no collision was registered.

In the third set of experiments, results showed a remarkable success rate of 96%. In the remaining 4% of the cases, robots hit an obstacle and hence the corresponding run was terminated. After analyzing failures cases separately, we found out that they were all due to slow turning rate achieved by the compound robot structure in the goal direction after avoiding an obstacle. This slow turning rate made the robot hit the next obstacle with the blind side of the carried structure, corresponding to the region of the object where the robots cannot attach and which is blind with respect

to the distance scanner. A video showing one typical run for this set of experiments can be found in a supplementary page [19].

5 Conclusion and Future Work

In this paper, we presented a novel method to tackle a task that has received limited attention in the literature: obstacle avoidance in collective transport. The task involves collective transport of an object by a group of three robots. In this task, robots assemble to the object and have to navigate to a given goal location while avoiding obstacles.

The proposed method consists of two interacting behaviors. The first behavior is called social mediation and is used to perform negotiation of an heading direction which takes into account possibly conflicting perceptions of the members of the group. The second behavior achieves collective transport, using this mediated heading direction.

Experiments were performed in a simple arena with one obstacle placed at different angles and in a more complex arena with several obstacles. Results in the simple arena show that the efficiency (inversely linked to execution times) of the behavior solely depends on the angle at which obstacles are placed, and that the more the obstacle is placed perpendicularly to the direction of motion the more time it takes to avoid it. In a more complex environment, we measured the success rate of the proposed approach, obtaining 96% of success.

This work can be extended in a number of directions. As a first step, the proposed methodology can be validated on real robots. We speculate that the social mediation method, being a very high level behavior, will need few adaptations for the real robots experiments, whereas the collective transport might need some adjustments, especially for the motion control rule that has to minimize wheel slippage. Second, some of the assumptions made in this work could be relaxed. For example, it can be interesting to investigate how to solve the task by assuming that the irregular shape of the object is not known in advance. In this case, we speculate that the motion control logic will need to be extended. Third and more ambitiously, a long term goal would be to understand how to control a group of an arbitrary number of robots, connected between each other and/or to an irregular object at different positions. In this case, we speculate that the social mediation methodology can be extended to tackle dynamic negotiation of heading direction with an arbitrary number of robots. Finally, a theoretical model of the system can be developed and used to prove some properties of the algorithm, such as that no cyclic situations (i.e. no “deadlocks”) can arise.

Acknowledgements. This work was supported by the *SWARMANOID* project funded by the Future and Emerging Technologies programme (IST-FET) of the European Commission (grant IST-022888). This work was partially supported by the European Union through the ERC Advance Grant “E-SWARM: Engineering Swarm Intelligence Systems” (contract

246939). M. Birattari and M. Dorigo acknowledge support from the F.R.S.-FNRS of the French Community of Belgium.

References

1. Bahçeci, E., Soysal, O., Şahin, E.: A review: pattern formation and adaptation in multi-robot systems. Technical Report CMU-RI-TR-03-43, Robotics Institute, Pittsburgh, PA (2003)
2. Varghese, B., McKee, G.: A review and implementation of swarm pattern formation and transformation models. International Journal of Intelligent Computing and Cybernetics 2(4), 786–817 (2009)
3. Balch, T., Hybinette, M.: Social potentials for scalable multi-robot formations. In: Carlisle, B.R., Khatib, O. (eds.) Proceedings of the 2000 IEEE International Conference on Robotics and Automation (ICRA 2000), pp. 73–80. IEEE Press, Piscataway (2000)
4. Spears, W.M., Spears, D.F., Hamann, J.C., Heil, R.: Distributed, physics-based control of swarms of vehicles. Autonomous Robots 17(2-3), 137–162 (2004)
5. Stilwell, D., Bay, J.: Toward the development of a material transport system using swarms of ant-like robots. In: Proceedings IEEE International Conference on Robotics and Automation, pp. 766–771. IEEE Comput. Soc. Press, Piscataway (1993)
6. Kosuge, K., Oosumi, T., Satou, M., Chiba, K., Takeo, K.: Transportation of a single object by two decentralized-controlled nonholonomic mobile robots. In: Giralt, G., Dario, P. (eds.) IEEE International Conference on Robotics and Automation, vol. 4, pp. 2989–2994. IEEE Press, Piscataway (1998)
7. Wang, Z., Takano, Y., Hirata, Y., Kosuge, K.: A pushing leader based decentralized control method for cooperative object transportation. In: Asama, H. (ed.) IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 1, pp. 1035–1040. IEEE Press, Piscataway (2004)
8. Balch, T., Ronald, Arkin, R.C.: Communication in reactive multiagent robotic systems. Autonomous Robots 1, 27–52 (1994)
9. Donald, B.R., Jennings, J., Rus, D.: Information invariants for distributed manipulation. The International Journal of Robotics Research 16(5), 673 (1997)
10. Yamada, S., Saito, J.: Adaptive action selection without explicit communication for multi-robot box-pushing. In: Harashima, F., Fukuda, T. (eds.) IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3, pp. 1444–1449. IEEE Press, Piscataway (1999)
11. Campo, A., Nouyan, S., Birattari, M., Groß, R., Dorigo, M.: Negotiation of Goal Direction for Cooperative Transport. In: Dorigo, M., Gambardella, L.M., Birattari, M., Martinoli, A., Poli, R., Stützle, T. (eds.) ANTS 2006. LNCS, vol. 4150, pp. 191–202. Springer, Heidelberg (2006)
12. Groß, R., Dorigo, M.: Towards group transport by swarms of robots. International Journal of Bio-Inspired Computation 1(1-2), 1–13 (2009)
13. Trianni, V., Dorigo, M.: Self-organisation and communication in groups of simulated and physical robots. Biological Cybernetics 95, 213–231 (2006)
14. Baldassarre, G., Parisi, D., Nolfi, S.: Distributed coordination of simulated robots based on self-organization. Artificial Life 12(3), 289–743 (2006)
15. Bonani, M., Longchamp, V., Magnenat, S., Rétornaz, P., Burnier, D., Roulet, G., Vaussard, F., Bleuler, H., Mondada, F.: The MarXbot, a miniature mobile robot opening new perspectives for the collective-robotic research. In: Huo, R., Asama, H. (eds.) 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010). IEEE Press, Piscataway (2010)

16. Bonani, M., Magnenat, S., Rétornaz, P., Mondada, F.: The Hand-Bot, a Robot Design for Simultaneous Climbing and Manipulation. In: Xie, M., Xiong, Y., Xiong, C., Liu, H., Hu, Z. (eds.) ICIRA 2009. LNCS (LNAI), vol. 5928, pp. 11–22. Springer, Heidelberg (2009)
17. Magnenat, S., Longchamp, V., Bonani, M., Rétornaz, P., Germano, P., Bleuler, H., Mondada, F.: Affordable slam through the co-design of hardware and methodology. In: Snyder, W., Kumar, V. (eds.) IEEE International Conference on Robotics and Automation. IEEE Press, Piscataway (2010)
18. Roberts, J., Stirling, T., Zufferey, J., Floreano, D.: 2.5d infrared range and bearing system for collective robotics. In: Hamel, W.R. (ed.) IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE Press, Piscataway (2009)
19. Ferrante, E., Brambilla, M., Birattari, M., Dorigo, M.: "Look-out!": Socially mediated obstacle avoidance in collective transport: Complete data (2010), Supplementary information page at `http://iridia.ulb.ac.be/supp/IridiaSupp2010-005/`

Physical Interactions in Swarm Robotics: The Hand-Bot Case Study

Michael Bonani, Philippe Rétornaz, Stéphane Magnenat,
Hannes Bleuler, and Francesco Mondada

Abstract. This paper presents a case-study on the performance achieved by the mechanical interactions of self-assembling mobile robots. This study is based on the hand-bot robot, designed to operate within heterogeneous swarms of robots. The hand-bot is specialized in object manipulation and can improve its performance by exploiting physical collaborations by self-assembling with other hand-bots or with foot-bots (ground robots). The paper analyzes the achieved performance and demonstrates the highly super-linear properties of the accessible volume in respect to the number of robots. These extremely interesting performances are strongly linked to the self-assembling mechanisms and the physical nature of the interaction, and do not scale to a large number of robots. Finally, this study suggests that such interesting properties are more accessible for heterogeneous systems or devices achieving complex tasks.

1 Introduction

Self-assembling is a feature in collective robotics which allows us to drastically improve the performances of single individuals by exploiting mechanical interactions [10, 5]. Applications can be found in space robotics [14], all-terrain mobility [13], underwater robotics [7], and simulation of living systems [4]. The main advantages of this approach to robotics are robustness to failure, because of the redundancy provided by multiple physical units, and flexibility. This last property is achieved by the large number of configurations these robots can form. They can, for

Michael Bonani · Philippe Rétornaz · Hannes Bleuler · Francesco Mondada
Laboratoire de Sytèmes Robotiques, Ecole Polytechnique Fédérale de Lausanne,
Station 9, CH-1015 Lausanne, Switzerland
e-mail: firstname.lastname@epfl.ch

Stéphane Magnenat
Autonomous System Lab, ETH Zurich, Switzerland

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 585–595.
springerlink.com

instance, form structures to navigate in specific shapes [1], to pass obstacles [11], or to pull heavy objects [12, 11]. An extended overview of the field is given in [5].

Self-assembling is widely studied in homogeneous groups of robots forming 2D structures [6, 5]. Few studies address self-assembly in 3D and none, to our knowledge, perform manipulation tasks in this space. This paper presents the case-study of the hand-bot, a robot specialized in object manipulation and capable of self-assembling to access 3D space. This example shows how self-assembly can enhance performance by mechanical interaction between assembled units. This work is an extension into the third dimension and to heterogeneous swarms of the work presented in [10]. While some of the conclusions are similar, the application of the same principles into heterogeneous and 3D systems capable of complex tasks allows a much deeper understanding of the phenomena.

2 The Hand-Bot Robot

The hand-bot is a small-size robot specialized in manipulation of small objects positioned much higher than its size (details are given in [3]). The robot is about 30 cm high, weights 2.8 Kg, and can manipulate an object placed in a vertical structure, for instance a shelf, between the floor and a ceiling located at 2.5 m above the floor. The ceiling above the robot has to be ferromagnetic, which is the case in many offices at the Ecole Polytechnique Fédérale de Lausanne. To perform this task, we equipped the hand-bot with three main groups of actuators, presented in Fig. 1:

1. A launcher allowing to shoot to the ceiling a switchable magnet pulling a rope. Once attached, the robot can lift itself. When the operation is finished, the robot can detach the magnet and wind the rope, making it ready for a new launch.
2. Two fans to stabilize and control the yaw of the robot when suspended. These actuators also allow the robot to move forward and backward.
3. Two arms equipped with grippers to allow the robot to attach to existing structures, to grasp object, and to self-assemble with other hand-bots.

Using the attachment to the ceiling, the fans and the arms, a single hand-bot can operate following two main strategies:

1. Lift up in an empty area. This approach uses the rope for the vertical displacement and the fans for stabilizing yaw and for making small forward and backward movements. This leaves both grippers available for manipulation but the positioning accuracy is poor.
2. Lift up by grabbing parts of a structure, like a shelf. In this type of operation (see Fig. 3 and the corresponding video on YouTube[1]), the stabilization of the movement is made using the grippers by attaching to the structure [3]. The hand-bot uses the two grippers in an alternate way to keep a contact with the structure. This improves stability, provides precise positioning, and provides access to a

[1] http://www.youtube.com/watch?v=92bLgE6DO2g

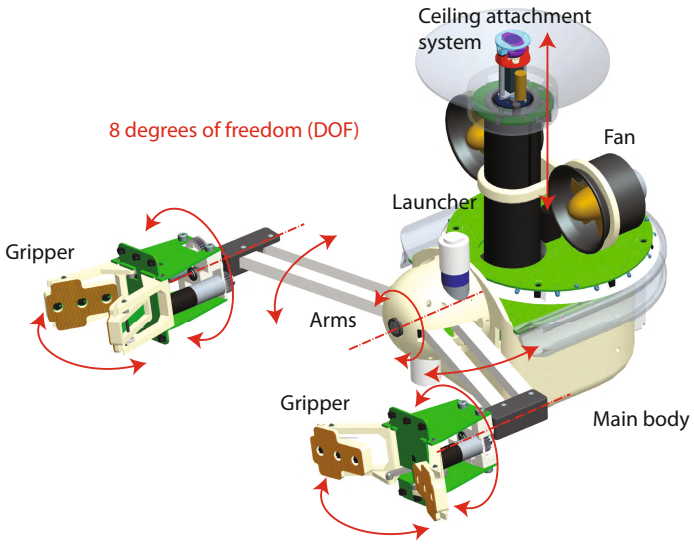


Fig. 1 Structure of the hand-bot and its degrees of freedom

large area because of lateral movements; but this limits the manipulation to one free gripper. Thus, the robot cannot climb and transport an object concurrently.

In both cases, the lifting principle is to use the attachment to the ceiling to generate most of the vertical force (F_w in Fig. 2). In addition, in strategy 1, the hand-bot can move a little on the horizontal axis, thanks to the fans which provide an horizontal force (F_f in Fig. 2). In strategy 2, the robot can push or pull on the structure using its attached gripper. The resultant (F_r in Fig. 2) is aligned with the arm.

As Fig. 2 shows, the maximal attachment force F_w depends on the orientation of the rope, given by the angle α. Indeed, as the attachment point of the rope on the magnetic device is located at six centimeters of distance from the ceiling (see real device on top right of Fig. 2), the force F_w generates a peeling moment which tends to detach the magnet when the rope is not vertical ($\alpha \neq 0$). Because the peeling moment is extremely hard to compute analytically, we measured the maximal value of F_w as function of the angle α on the real device, obtaining the values illustrated in Fig. 2.

It appears clearly that the main limitation of the first strategy (based on the fan) is the weak propulsion force of the fans, resulting in a limited accessible volume. When blowing at full power, we measured that the fans can only generate an angle α of 0.026 rad.

The main limitation of the second approach is in the use of one arm and one gripper for lateral displacement, which renders them unavailable for manipulation. The accessible area is much larger than in the first approach and is limited by the morphology. Moreover, near to the ceiling the angle of the rope increases and the

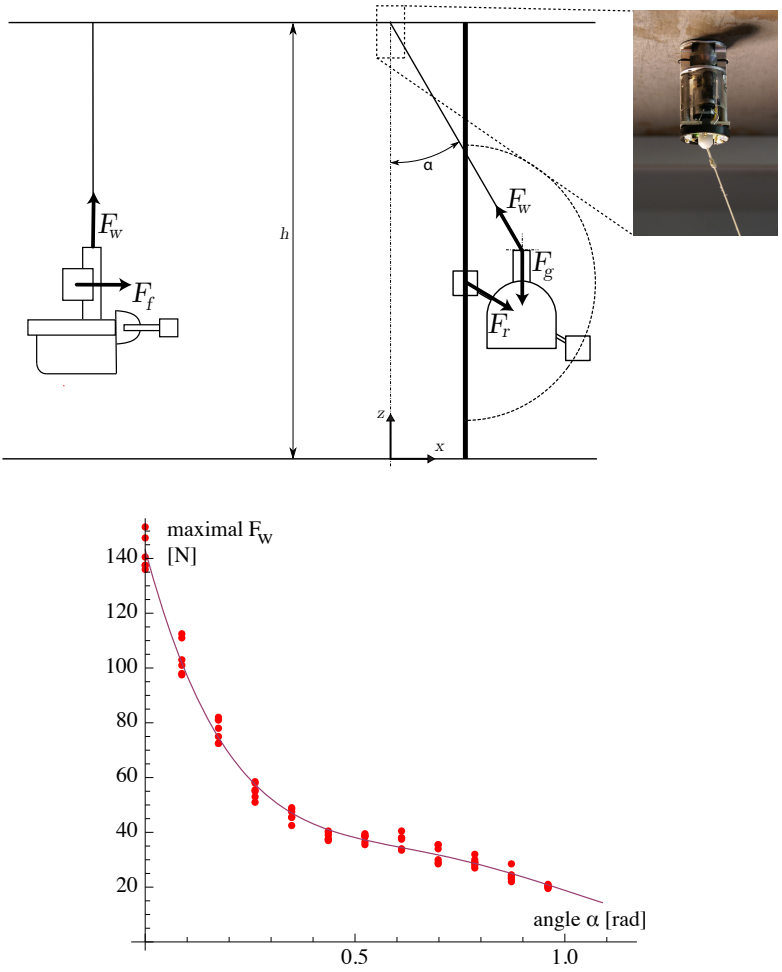


Fig. 2 Top: Scheme of forces and the detail of the magnetic attachment system. Bottom: Maximal attachment force of the magnetic system as function of the angle α of the rope; the measurements are represented as red dots, the line shows a possible interpolation.

magnetic system cannot support anymore the robot. Fig. 3 (right) compares the access zones of these two approaches.

3 Collective Strategies Based on Self-assembling

To overcome the limitations described in section 2, the hand-bot can self-assemble with other hand-bots or with foot-bots, a type of robots which we designed for

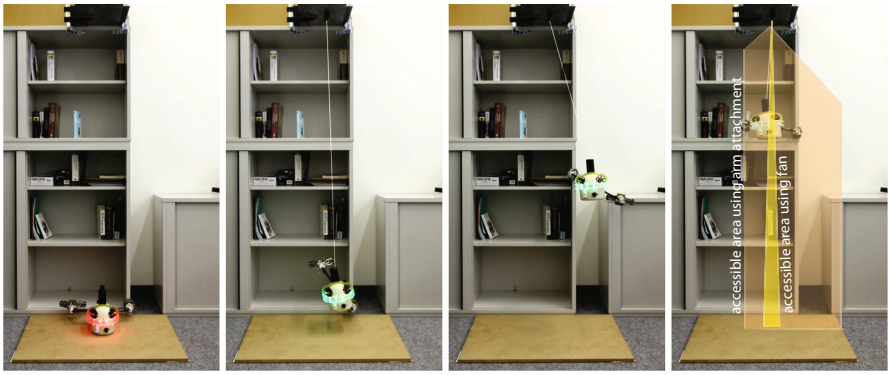


Fig. 3 Sequence of climbing and grasping: positioning, climbing and grasping of a book

displacement and navigation on the ground. The goal is to increase the accessible volume by a self-assembling collective approach.

3.1 Self-assembling with Foot-Bots

The hand-bot is strictly specialized in climbing and manipulation. This means for instance that the hand-bot is not equipped with wheels or any other actuator to move on the ground. The hand-bot achieves displacement on the ground by self-assembling with another type of robot, the foot-bot.

The foot-bot is a modified version of the marXbot, a robot designed for research in collective robotics [2, 8] (see Fig. 4, left). The foot-bot has tracks and wheels (called together "treels") and can move in all-terrain conditions. For this experiment, the foot-bot is specifically equipped with a self-assembling module allowing it to physically connect to the hand-bot and to form a common rigid body. Details on the marXbot robot and on the self-assembling mechanism are described in [2]. The main characteristics of this self-assembling mechanism of the foot-bot is that it can rotate all around the robot body. This allows a connected robot to move in any direction. Foot-bots can attach to the sides and to the back of the hand-bot (see Fig. 4, right).

This assembly of robots can provide the hand-bot with the necessary mobility on the ground. Its performance in displacement depends strongly on the number of foot-bots involved. One foot-bot alone can displace a hand-bot but can hardly position it correctly. A foot-bot connected to a hand-bot can only pull or push the hand-bot. It cannot move in other directions than those two, because this would make the hand-bot nearly rotate on the spot. If the foot-bot pulls the hand-bot, this makes it very hard to position, for instance, the hand-bot facing a shelf. If the foot-bot pushes the hand-bot, the control is highly unstable and requires very complex maneuvers. If foot-bots are available, they can achieve a much better configuration by attaching laterally to the hand-bot. This allows the foot-bots to move in any

Fig. 4 Left: Foot-bot robot. Right: Possible positioning of foot-bot robots around the hand-bot to provide movement on the ground.

direction. While this two-points system does not control all degrees of freedom, the positioning in the horizontal plane is much more precise and easier to control. The ideal situation occurs when the hand-bot is connected to three foot-bots, two placed laterally and one behind the hand-bot. This configuration allows to control all degrees of freedom. The hand-bot is therefore well stabilized and can be positioned in the best way. Self-assembly with more than three robots does not make sense from a stability and mobility point of view, and is difficult because of the limited area of attachment.

3.2 Self-assembling with Hand-Bots

In the previous sections we have seen the ability of one hand-bot to move vertically and the possibility to use self-assembly to position it on the ground. Probably the most interesting possibility is to self-assemble several hand-bots to extend the volume accessible by the robots. The hand-bot can self-assemble with other hand-bots using its gripper, as illustrated in Fig. 5.

The complete scenario using self-assembling of foot-bots and hand-bots is the following: Foot-bots place the hand-bots in several locations at the limits of the working area. In the following examples we will consider two or three hand-bots placed at distance d to each other. When the hand-bots are placed, they attach to the ceiling, shooting their attachment system. Then foot-bots bring all hand-bots together in the center of the working area. This allows them to self-assemble using their grippers. When assembled, the hand-bots can move within the 3D space by concurrently using their attachments to the ceiling.

As illustrated in section 2, the hand-bot can use two main actuators for lateral displacement: fans and arms. By self-assembling we can add a third actuator, which

is the ceiling attachment system of another hand-bot. If we consider the situation of elevating the hand-bot in the air keeping both gripper available for manipulation, we can distinguish three main situations:

1. When the hand-bot is alone, it can control its lateral movements and yaw (three degrees of freedom "DOF") using only the fans. The accessible volume is limited, as we discussed in section 2.
2. When two hand-bots are assembled, two DOF are controlled by this additional connection and only one DOF needs to be controlled by the fans. The volume is bigger and follows a vertical plane crossing the two attachment points.
3. When three hand-bots are assembled, all DOF are controlled by the ropes connected to the ceiling. The volume becomes more important and is enclosed within the three vertical planes crossing the attachment points.

In the second and third situations, the upper limitation of the volume is given by the attachment force with respect to the angle as illustrated in Fig. 2. The resulting volumes for a distance between robots d of 100 cm are illustrated in Fig. 6.

In this self-assembling collaboration, it is interesting to observe the effect of cooperation on normalized system performances. An interesting measurement of collective performance is the collective speedup factor [9] of a group of n robots, given by equation 1.

Fig. 5 Three hand-bots self-assembled and suspended by the ropes

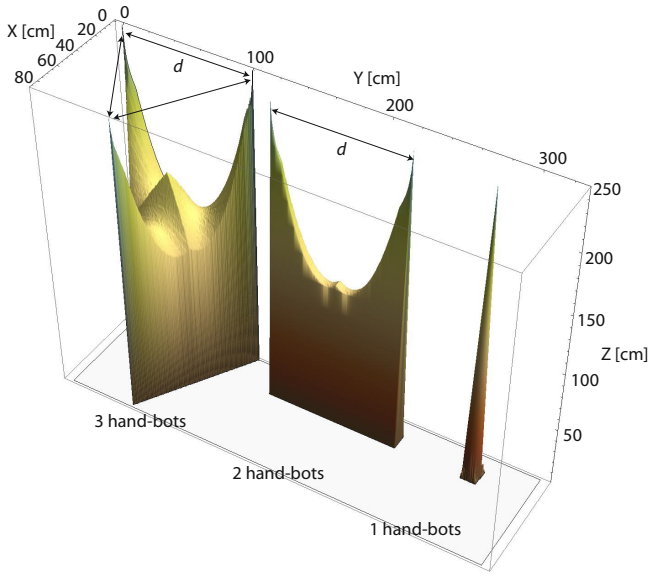


Fig. 6 Accessible volume for one, two and three hand-bots cooperating by self-assembly. This situation considers a distance between robots of $d = 100$cm

$$CS(n) = \frac{mP(n)}{nP(m)} \tag{1}$$

where $P(n)$ is the performance of a group of n robots and m is the minimal number of robots needed to perform the task. We can distinguish between superlinear performances when $CS(n) > 1$, linear performances when $CS(n) = 1$ and sublinear performances when $CS(n) < 1$. A simple combination of n robots having no influence on each-other should generate a linear performance by performing the task n times better or faster than one robot or module.

In our case we can look to the accessible volume, for a task feasible with one robot:

$$CS_v(n,d) = \frac{V(n,d)}{nV(1)} \tag{2}$$

where $V(n,d)$ is the volume accessed by n hand-bots with attachments placed at a distance d to each other, and $V(1)$ the volume accessed by one hand-bot using the fans for lateral displacement. Fig. 7 shows on the left the plot of the accessible volume $V(n,d)$ as function of d for n =1,2 and 3. On the right of the same figure, the resulting $CS_v(n,d)$ shows that the system exhibits highly superlinear performances.

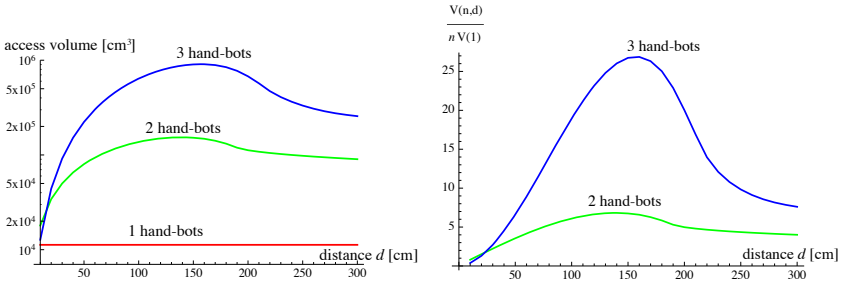


Fig. 7 Left: Volume accessible by one, two and three hand-bots exploiting self-assembling. Right: Ratio between access volume using n hand-bots and volume using one hand-bot, normalized per number of robots.

4 Discussion

The system presented in this paper shows a case study of physical collaboration among robots based on self-assembling. The example shows clearly that physical collaboration can generate a very strong multiplication of performance. This was already observed in [10] but the performance factor was much smaller than the one observed here. In the positioning task, the absolute performance is hard to define and measure — we should measure the involved engineering work — but the improvement in performance is clear when moving from one to two foot-bots. Three foot-bots can generate the best performance, even if the performance factor between two and three could be sublinear. Indeed the gain in performance between two and three robots appears to be minor considering that the number of robots has been increased by fifty percent. Additional foot-bots can improve the robustness, the payload and the speed of the system, but do not improve quality in positioning or in control efficiency.

Our system shows superlinear performance when considering 3D access. Self-assembling two hand-bots increases the collective-volume access by a factor of 6.8 in the best case (d =140 cm). Adding another hand-bot allows us to achieve a collective factor of 26.8, which represents an additional increase of a factor four with respect to a combination of two robots. These performances are achieved through the geometry of the system, where adding a robot means adding dimensionality to the system. This is very specific to physical systems. Also very specific to physical systems is the fact that those systems do not scale. For more that three robots (both foot-bots and hand-bots) the evolution of the volume depends on the shape of its base, evolving from a equilateral triangle of side d, to a n-sided polygon inscribed in a circle of diameter d, n being the number of hand-bots. This area saturates (at $d^2\pi/4$) and the corresponding volume too. Therefore the system performance stay stable with n increasing and therefore CS_v drops.

The collective-performance factors listed above do not consider the total number of robots, which should include both the foot-bots and the hand-bots involved in

the task. When taking into consideration both types of robots, the ratio m/n would increase and the performance factor as well. This shows a very interesting effect of heterogeneity. Having different robots for ground mobility and for vertical operations allows us to optimize each subsystem independently and to combine functionalities in an orthogonal manner. For instance, in this case-study, once three foot-bots are available, the number of hand-bots can be chosen freely, the foot-bots being capable to position each hand-bots in a sequential way. This would increase the *CS*.

Finally, we can make an observation about the hardware requirements for self-assembling. Foot-bots, which are simple mobile robots that have to move around — a trivial task — require a specific hardware module to ensure the self-assembling capability. The added hardware is even one of the most complex hardware component in the foot-bot. On the contrary, the hand-bot embeds grippers in its basic configuration because its basic task is more complex (manipulation). Using the same grippers for self-assembling, the hand-bot does not require additional hardware to perform physical cooperation. This suggests that in more complex tasks, requiring more complex robots, cooperation based on self-assembling could be more accessible and require less extra specific hardware than for simple robots.

5 Conclusion

We presented a case study of physical interactions in a heterogeneous group of robots. This type of collaboration allows very high increase in performance, but is not scalable. The heterogeneity improves performances and allows to optimise different robots for different sub-tasks. The complexity of the tasks, requiring complex robotic hardware, improves accessibility to self-assembling operation. These conclusions shows very singular properties of heterogeneous self-assembling systems. Additional research work is necessary to increase the understanding and to develop the exploitation of this particular but promising type of systems.

Acknowledgements. The hand-bot and foot-bot robots have been designed and produced with the precious help of Tarek Baaboura, Daniel Burnier. Valentin Longchamps has provided substantial contributions for software development. This work was supported by the Swarmanoid and Perplexus projects, both funded by the Future and Emerging Technologies programme (IST-FET) of the European Community, respectively under grants 022888 and 34632. The information provided is the sole responsibility of the authors and does not reflect the Community's opinion. The Community is not responsible for any use that might be made of data appearing in this publication.

References

1. Baldassarre, G., Trianni, V., Bonani, M., Mondada, F., Dorigo, M., Nolfi, S.: Self-Organised Coordinated Motion in Groups of Physically Connected Robots. IEEE Transactions on Systems, Man and Cybernetics Part B: Cybernetics 37(1), 224–239 (2007)

2. Bonani, M., Longchamp, V., Magnenat, S., Rétornaz, P., Burnier, D., Roulet, G., Bleuler, H., Mondada, F.: The marxbot, a miniature mobile robot opening new perspectives for the collective-robotic research. In: Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems IROS 2010, IEEE Press, Piscataway (2010)
3. Bonani, M., Magnenat, S., Rétornaz, P., Mondada, F.: The Hand-Bot, a Robot Design for Simultaneous Climbing and Manipulation. In: Xie, M., Xiong, Y., Xiong, C., Liu, H., Hu, Z. (eds.) ICIRA 2009. LNCS, vol. 5928, pp. 11–22. Springer, Heidelberg (2009)
4. Breivik, J.: Self-organization of template-replicating polymers and the spontaneous rise of genetic information. Entropy 3(4), 273–279 (2001)
5. Gross, R., Dorigo, M.: Self-assembly at the macroscopic scale. Proceedings of the IEEE 96(9), 1490–1508 (2008)
6. Groß, R., Dorigo, M., Mondada, F., Dorigo, M.: Autonomous Self-assembly in Swarm-Bots. IEEE Transactions on Robotics 22(6), 1115–1130 (2006)
7. von Haller, B., Ijspeert, A.J., Floreano, D.: Co-evolution of Structures and Controllers for Neubot Underwater Modular Robots. In: Capcarrère, M.S., Freitas, A.A., Bentley, P.J., Johnson, C.G., Timmis, J. (eds.) ECAL 2005. LNCS (LNAI), vol. 3630, pp. 189–199. Springer, Heidelberg (2005)
8. Magnenat, S., Longchamp, V., Bonani, M., Rétornaz, P., Germano, P., Bleuler, H., Mondada, F.: Affordable SLAM through the Co-Design of Hardware and Methodology. In: Proceedings of the 2010 IEEE International Conference on Robotics and Automation. IEEE Press (2010)
9. Martinoli, A.: Swarm intelligence in autonomous collective robotics: From tools to the analysis and synthesis of distributed collective strategies. Ph.D. thesis, Swiss Federal Institute of Technology in Lausanne (EPFL), Lausanne, Switzerland (1999)
10. Mondada, F., Bonani, M., Guignard, A., Magnenat, S., Studer, C., Floreano, D.: Superlinear Physical Performances in a SWARM-BOT. In: Capcarrère, M.S., Freitas, A.A., Bentley, P.J., Johnson, C.G., Timmis, J. (eds.) ECAL 2005. LNCS (LNAI), vol. 3630, pp. 282–291. Springer, Heidelberg (2005)
11. Mondada, F., Pettinaro, G.C., Guignard, A., Kwee, I., Floreano, D., Deneubourg, J.L., Nolfi, S., Gambardella, L., Dorigo, M.: SWARM-BOT: a New Distributed Robotic Concept. Autonomous Robots, Special Issue on Swarm Robotics 17(2-3), 193–221 (2004)
12. Nouyan, S., Groß, R., Bonani, M., Mondada, F., Dorigo, M.: Teamwork in Self-Organized Robot Colonies. IEEE Transactions on Evolutionary Computation 13(4), 695–711 (2009)
13. O'Grady, R., Groß, R., Christensen, A., Dorigo, M.: Self-assembly strategies in a group of autonomous mobile robots. Autonomous Robots 28(4), 439–455 (2010)
14. Toglia, C., Kettler, D., Kennedy, F., Dubowsky, S.: A study of cooperative control of self-assembling robots in space with experimental validation. In: ICRA 2009: Proceedings of the 2009 IEEE International Conference on Robotics and Automation, pp. 3847–3852. IEEE Press, Piscataway (2009)

A Low-Cost Multi-robot System for Research, Teaching, and Outreach

James McLurkin*, Andrew J. Lynch, Scott Rixner, Thomas W. Barr, Alvin Chou, Kathleen Foster, and Siegfried Bilstein

Abstract. We describe a new low-cost robot design that enables large-scale multi-robot research, innovative new curriculum, and multi-robotics outreach to younger students. There are four main parts to the system: the r-one robot, a Python development environment, a camera tracking system for ground-truth localization, and server software to connect all the pieces together. This paper presents our preliminary work on the robot design and our experience using it to teach an introductory engineering class. The hardware can support classes in computer science, electrical engineering, and mechanical engineering. The low-cost and small size will enable more research groups to perform multi-robot experiments on physical hardware. The Python development environment greatly simplifies programming and will make robotics more accessible to a larger group of educators, students, and researchers.

1 Introduction

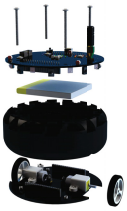
Multi-robot research is enjoying a boost in popularity. However, there is a gap in the spectrum of available solutions of feature-rich, affordable hardware platforms. Existing platforms are too expensive, too big to test large populations (25 to 100) indoors, or lack key sensors for multi-robot research. This paper presents our work on building a low-cost robot with sufficient sensors and processing power for basic multi-robot research. Our ultimate goal is to produce a small, capable, integrated platform with a parts cost of around $100 per robot. A robotic system like this will enable research with large populations of robots, multi-robot curricula at the undergraduate level, and outreach into K-12 schools.

James McLurkin · Andrew J. Lynch · Scott Rixner · Thomas W. Barr · Alvin Chou · Kathleen Foster · Siegfried Bilstein
Rice University, Houston, TX, USA
e-mail: jmclurkin@rice.edu

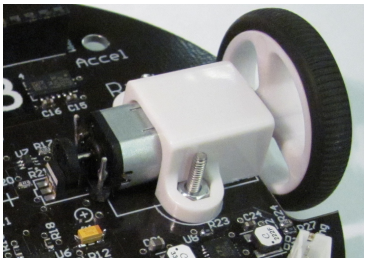
* Corresponding author.

A. Martinoli et al. (Eds.): Distributed Autonomous Robotic Systems, STAR 83, pp. 597–609.
springerlink.com

(a) The r-one robot.



(b) Exploded CAD view.



(c) The r-one encoder.

Fig. 1 **a:** The r-one robot. **b:** Exploded CAD view of the robot assembly. The robot is composed of two circuit boards bound together with a circular shell and four screws. **c:** The motors and encoders mount directly to the circuit board.

Our design, the r-one robot shown in Figure 1(a), attempts to fill this gap in the design space. It has a cost of around $220 per robot, a full sensor suite including a gyro, accelerometer, wheel encoders, and light sensors. It includes a radio for global control, an infrared beacon for ground-truth localization, and an infrared inter-robot communication and localization system. The robot runs an embedded Python interpreter to make programming more accessible to younger, less experienced students. We have started building a server infrastructure to support centralized command and control and data logging. We are currently using the platform in an introductory first-year engineering course, with good success to date.

This paper is organized as follows. Section 2 describes currently available robots, and makes the case for our new hardware platform. Section 3 describes the details of the robot's hardware and software. Section 4 discusses the ground-truth global positioning system, and Section 5 describes the inter-robot communication. Section 6 mentions our preliminary work on a server infrastructure for the system. Finally, Section 7 describes the three main applications, research, teaching, and outreach, with our current experiences in this semester's course.

2 Existing Robotic Platforms

Our goal is to produce a platform capable of basic research, teaching, and outreach. Low hardware cost is a requirement for these goals; it enables large populations for research, use in teaching labs on tight budgets, and use in outreach activities. In tension with cost is functionality. In particular, the sensor suite needs to be selected for the intended usage of the system, but carefully limited in scope. Our effort is not the first to address this concern, there are several existing robot platforms designed for research and teaching. Table 1 compares the differences between these platforms.

The Pololu 3pi and Scribbler are inexpensive, but lack basic sensors, such as wheel encoders, and do not have the communication systems required for

Table 1 A comparison of available low-cost robots suitable for multi-robot research

Robot	Source	Wheel Encoders	Radio	Neighbor Position	Ground Truth Position	Other Sensors	Retail Price ($)	Parts Cost ($)
Khepera III	K-Team	yes	yes	no	no	ultrasonic(x5), IR range (x8), cliff(x2)	2000	-
Create	iRobot	yes	-	no	no	bump (x2), cliff (x2)	220	-
Scribbler	Parallax	no	yes	no	no	visible light(x3), IR range (x1)	198	-
Mindstorms	LEGO	yes	yes	no	no	bump(x2), light(x1), ultrasonic ranger(x1)	249	-
3pi	Pololu	no	no	no	no	IR line(x5)	99	-
e-puck	EPFL	yes	yes	no	no	mic(x3), 3d accel, IR range (x8), camera	979	-
e-puck	EPFL	yes	yes	yes(see text)	no	mic(x3), 3d accel, IR range (x8), camera	1388	-
CostBots	Berkeley	no	yes	no	no	accelerometer, NEST sensor boards	-	200
robomote	USC	yes	yes	no	no	compass, bump, IR range	-	150
r-one	Rice	yes	yes	yes	yes	visible light(x3), 3d accel, 2d gyro	-	220

multi-robot coordination. LEGO Mindstorms is the leader in educational robotics, but there is no available sensor for detecting local network geometry, *i.e.*, the positions of neighboring robots. The iRobot Create is a popular platform for medium-sized robots, but the size, cost, and limited sensor suite require many add-on components for multi-robot work. The robomote and the costbots do not have the sensors needed to determine local network geometry. Most of these platforms need to be programmed in C, which makes them difficult to use with younger students. The platforms that are closest in size and sensor suite is the Khepera robot and EPFL e-puck, but their cost makes it difficult to field large numbers for research and teaching. The e-puck robot is designed to be expandable. This allows the addition of an inter-robot communication turret, but this adds cost to the system. None of the existing platforms are uniquely identifiable from a global localization system. The r-one's main contribution to the multi-robotics community is an integrated, low-cost platform with inter-robot communications, a sensor for network geometry, a system for ground truth position and a flexible embedded python development environment. This integration comes with a downside, as the r-one design is not expandable like some of the others, notably the Khepera and EPFL e-puck. This simplification to the r-one design was a good trade-off for this initial version, but it may limit future use of the platform and need to be revisited when other capabilities are needed.

Our current r-one robot costs $220 in quantities of 30. The component costs total $130 and PCB fabrication is $40. The assembly of the top board cost $30. The assembly for the bottom circuit board was completed in-house. The final mechanical assembly of the robot was also done in-house, and takes about 15 minutes per robot. In future revisions, we will use an assembly company for the entire process, at an estimated rate of $50 per robot. Each robot would cost $220 assembled. We expect this cost to decrease for larger quantities. A $20 cost reduction can also be achieved by removing the more expensive sensors such as the 3D accelerometer and gyro. We are not at our goal of $100 in parts and assembly, but the current price is still low-cost.

3 Hardware Design and Python Interpreter

The r-one hardware design tries to strike a balance between features and cost. We have also developed a rich Python infrastructure for r-one that greatly simplifies its use, making it accessible to younger students.

3.1 r-one Hardware

Figure 1(a) shows a fully assembled robot, and Figure 1(b) shows the exploded diagram. The sensor suite consists of a 2-axis gyro, 3-axis accelerometer, and 3 visible-light photo resistors. The robot has two motors with quadrature encoders. The robot includes 8 IR transmitters, 8 IR receivers, a 2.4 GHz radio with 2Mbps data rate, and a USB port. To interact with the user, the robot has 3 push buttons and 3 arrays of five LEDs each in red, green, and blue. Each of the 15 total LED elements has individual brightness adjustment.

The robot is controlled by a Texas Instruments Stellaris LM3S8962 microcontroller. The CPU core is an ARM Cortex-M3 running at 50 MHz with 256 KB of Flash memory and 64 KB of SRAM. Total system power is 140 mA without activating motors or LEDs. Under stall torque with an active LED group, the robot can peak at 650 mA. In normal movement operations with LEDs active, the robot consumes 510 mA. With a 3.7V 2000 mAh lithium-polymer battery, the robot has been tested to last for 4 hours. The battery is charged from the USB port.

The exploded CAD diagram is shown in Figure 1(b). The robot is a 10 cm circle and weighs 230 grams. The robot is composed of two circuit boards bound together with a circular shell and four screws. The shell also serves as a protective shield to channel IR sensor measurements. The top circuit board contains the user interface and the microcontroller. The two boards connect together with two 16 pin 2.54mm (0.100”) headers. Standard headers are low-cost and make it easy to attach oscilloscope probes to demonstrate PWM and encoder signals to students.

The motors and encoders mount directly to the bottom circuit board, shown in Figure 1(c). These low-cost quadrature encoders are a new design, and use an optical interruption sensor to detect gaps in a custom encoder wheel attached to the rear motor shaft. The wheels are made from plastic on a laser cutter, and manufacturing tolerances limit the design to four slots, producing a 0.0625 mm/tick linear resolution at the wheel. The 32 mm wheels coupled with a 100:1 gearbox give the robots a maximum speed of 300 mm/sec, while internal friction limits the minimum controllable speed to around 40 mm/sec.

The 2.4 GHz radio on the robot can be used for inter-robot communication, but is designed for centralized command and control. The primary means of inter-robot communication is the local IR communication system described in Section 5. Robots can broadcast 32 byte packets and the chipset handles packet acknowledgments and retransmissions at the hardware level. Our Python API allows for packets to be defined as strings up to 32 bytes which significantly simplifies radio messaging. To test the power of our system, we constructed a simple distance-vector

mesh network router (in 200 lines of Python) that auto-discovers nodes and properly routes packets over many hops to their destination.

3.2 Python on the r-one

The r-one runs an embedded Python virtual machine based on the open-source Python-on-a-Chip project (http://code.google.com/p/python-on-a-chip/). We have ported the Python interpreter to the ARM Cortex M3 and implemented a full set of libraries providing Python programs access to all of the robot hardware. Others have ported Python-on-a-Chip to a Lego NXT robot, but they have not yet shown that you can effectively control a robot with Python [17]. In contrast, our Python environment provides full control over all of the sensors and actuators of the robot and we have shown that one can write non-trivial control programs in Python that work effectively, such as velocity control loops, simple light sensor behaviors, and network protocols. Our environment implements a significant fraction of the Python language and a comprehensive set of robot control libraries using 140KB of flash. If the interpreter is compiled with size optimizations, this shrinks to 89KB, leaving 167KB available for user programs. The running interpreter requires 1.6KB for its C stack, and 2.3KB for BSS and C data. The remainder of SRAM (60KB) can be used for the Python heap. We configured a 49KB Python heap to provide a margin of safety for the C stack and have found it to be more than adequate for our purposes.

Python code is developed on a desktop which compiles it into bytecode to be loaded onto the robot via USB. There are two methods of transferring compiled Python bytecode onto the microcontroller. The first method compiles an entire Python program into a single image to be executed by the robot, similar to the C-style compile-link-load workflow used in typical embedded development. This image can be programmed into the robot's Flash memory or SRAM. A second, more powerful mode of development uses the standard Python read-eval-print loop. A desktop program establishes a connection to the virtual machine on the robot. Python code typed into an interactive prompt running on the desktop is then compiled and sent to the robot where it is executed. Finally, the results of that statement are displayed back on the desktop. This allows for interactive use of the robot, providing immediate feedback on the effects of Python commands. This greatly simplifies debugging and development.

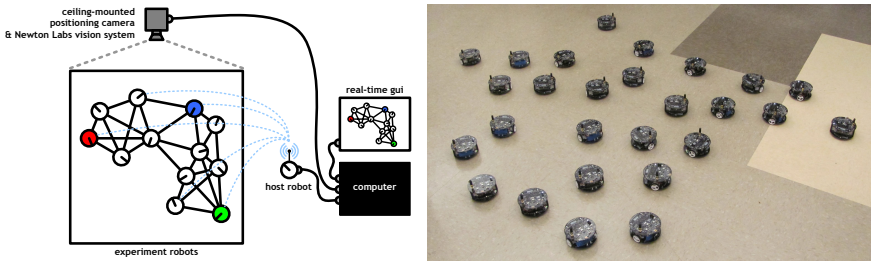
4 Global Robot Localization

Data collection on multi robot systems requires the user to know the ground truth positions of the robots. There are many means of determining a robot's global position: GPS [4], a Vicon-like tracking system [14], radio-acoustic ranging [9, 18], or camera-based tracking [22, 8, 15, 12]. However, GPS is unavailable indoors, a

Vicon system is expensive, and radio acoustic systems work well, but increase the complexity of each robot.

Camera-based tracking systems are currently the most common low-cost method for determining a robot's location in an indoor environment. These systems must have the ability to uniquely identify individual robots in the camera image. Fiducial tracking can find multiple markers, and with initialization, identify unique robots [22]. In a uniform environment, robots can be tracked by color alone as with SwisTrack [8]. Bar code tags such as AprilTags [15] provide unique IDs without initialization, and 6-DOF pose estimation. However, bar codes can large, and initialization is time-consuming so an alternative is to track IR beacons on each robot [14, 12]. The beacons transmit a pattern unique to each robot. One beacon per robot and one camera allow 2-DOF position to be measured directly. Multiple cameras or beacons can be used for full 6-DOF pose estimation [2].

Figure 2(a) shows a diagram of the complete global localization system that we are developing for the r-one. The system will have four main components: the robots running the experiment, a single host robot, a ceiling-mounted infrared (IR) localization camera, and a server computer with data logging software. Ground truth positions of the robots are measured by a vision-based localization system. The system uses an IR beacon LED on the top circuit board of each robot, which transmits an encoded pattern. The vision system detects the beacon LEDs, and tracks the $\{x, y\}$ positions of all of the robots simultaneously, while decoding 10 bits of unique ID data per second per robot. The server will collect and display all camera estimates of individual robot positions based on unique IDs. This system has been tested with the r-one robot, but not characterized. In our previous generation of robots, this system had a mean position error of 15.4mm [12]. This accuracy will vary with each setup, as it is a function of the camera placement. Our previous setup had the camera mounted on an 18 foot ceiling, pointing straight down. This covered a test area



(a) Data collection block diagram. (b) Twenty-five r-one robots.

Fig. 2 a: Diagram of the planned r-one data collection system. Each robot has a single top-mounted infrared beacon that flashes with an unique pattern. The ceiling-mounted camera identifies each robot with this pattern and tracks the {x, y} positions of each robot simultaneously, reporting the results to the computer at 1 Hz. **b:** A collection of 25 robots. We demonstrated this group on the first day of class. This kind of outreach activity is only possible with an inexpensive and portable robot.

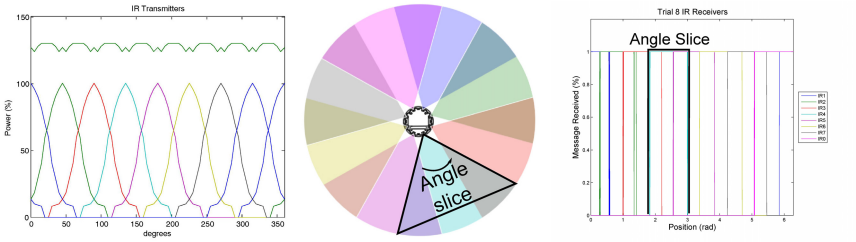
approximately 3 meters on a side. Most labs have lower ceilings, so the camera will need to be mounted at an angle to cover a large area, and perspective will make the accuracy non-uniform across the workspace.

5 Inter-robot Communication and Localization

The ability of individual robots to measure relative positions to neighboring robots is a critical feature in a multi-robot research system. Without this ability the robots have no way to control their physical configuration. There are many approaches to measure the geometry of a robot's neighbors, including vision-based systems [7, 6], and infrared light [10, 19, 5, 16]. The r-one robot uses an IR communication system to communicate and determine the local network geometry of neighboring robots.

Each robot has a set of eight IR transmitters and eight IR receivers. The transmitters transmit in unison, and were designed into the shell to provide a nearly radially uniform energy emissions pattern. Figure 3(a) shows the predicted angular output based on the IR emitter data sheet and the CAD model of the shell. This shows a variation of 4%, but we have not verified this performance on physical hardware yet. Because the communications bandwidth is very limited (see below), we selected a maximum range of around 1.5 meters in order to limit the number of neighbors and messages that are received by each robot.

Each robot has eight IR receivers, arranged so that their reception regions overlap as shown in Figure 3(b). The shell is designed to limit the detection arc of each receiver to 68°. By noting which receiver(s) detect a neighboring robot, the bearing can be estimated to within ≈22.5°. We characterized the arc and overlap of each



(a) Predicted IR transmitter power output. (b) Top CAD view of IR receiver overlap. (c) Measured IR overlap.

Fig. 3 a: The transmitter and shell are designed to produce a radially uniform power output. **b:** This is a top view of a CAD model of each IR receiver's detection region. Each receiver detects signals in a 68°arc. These regions overlap to form 16 distinct sectors. A message from a neighboring robot will be received on one or two receivers, and can be processed to determine the direction to within 2.5°. **c:** Experimental verification of the overlap of the receiver regions. The plot is showing the angle each receiver can detect an incoming message. The average width is 68°, which matches the CAD model. The corresponding arc from the top view is highlighted in black.

IR region by transmitting messages to a robot rotating on a turntable. Figure 3(c) shows the measured reception arc from each receiver with color corresponding to Figure 3(b). The reception arc varies from 63°to 74°with a mean of 68°over 10 trials.

The receivers are standard Sharp IR remote control devices, with 38khz modulation and a maximum bit rate of 1250bps. Our current protocol uses Manchester encoding to provide DC balance and has an 8-bit preamble and 8-bit CRC. We currently use 4-byte messages, which produces a 96-bit packet. We plan to use a simple TDMA scheme: the robots to transmit at periodic intervals, but with a random offset, similar to the ALOHA protocol [1]. This will limit our effective bandwidth before network congestion causes saturation. We have tested a protocol similar to RS232 8N1 that reduces message size to 60 bits and should increase the number of possible neighbors. These bandwidth constraints place a limit on robot density and algorithm complexity, but increasing bandwidth would require using more expensive or custom receivers.

A proof-of-concept follow-the-leader implementation was used to test this system on a prototype robot with no shell. Written in Python in less than 140 lines, robots are divided by the user into a transmitter, a leader and a group of followers. The user remote controls the leader via the radio. The leader constantly transmits an IR message which the followers use to flock towards it. If a follower sees the leader on one of the front two transmitters, it moves forwards. Otherwise, a follower rotates according to the direction of the leader's signal. Interestingly, this program takes advantage of the lack of a shell to produce a rough estimate of range. If the transmitter is more than a few feet from the receiver, a message is generally seen on only one or two receivers. As the transmitter gets closer, messages are observed on more receivers. This can be exploited to prevent robots from running into the leader.

6 Server

The final hardware component needed is a centralized server to provide hands-free operation, data logging, and integrate the ground-truth localization with telemetry data logged from the robots. We are developing a system that will be similar to our previous generation SwarmBot server [11]. The system will use a USB tethered host robot to communicate with the active robots over radio. Data for individual robots will be tagged with their positions from the localization system, and logged into a single file. The user will be able to observe the positions and current state of each robot in real time. Radio bandwidth must be shared with all the robots in the system, limiting the amount of information that each robot can transmit. Also, our current system uses an expensive camera running propitiatory software. The next generation will use an inexpensive camera, and be readily available to the community.

7 Applications

The three main applications of this system are research, teaching, and outreach. This section discusses an example and results of the r-one system in these applications.

7.1 Research

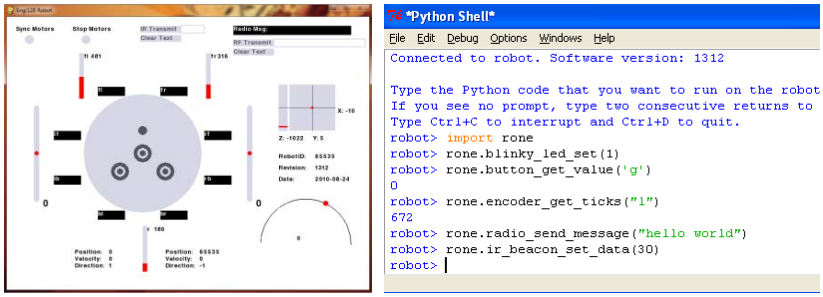
The primary use of this system is for multi-robot research on large populations of robots. To support this, the system will support a "hands-free" philosophy [13]; robots will be able to be programmed and administered remotely from a centralized location. The ground truth measurements of robot positions are vital for experimental validation of performance claims [12]. The biggest drawback of the r-one robot is the limited precision of the inter-robot localization system. To address this, we have started two research projects; one to improve the estimates with odometry, and one to improve the estimates by passing messages around local network neighborhoods.

7.2 Education

Current robot software stacks are challenging to use in introductory courses. Low-level software stacks are generally very difficult to use. However, higher-level software stacks are often too watered down to build complex systems. Computer science departments have tackled similar problems for introductory courses on software development and many have moved to languages such as Python that are both easy-to-use as well as powerful [3, 20, 21]. Python combines a simple syntax with a garbage-collected environment that prevents a programmer from having to deal with pointers and memory management.

Additionally, the interactive prompt turns development into an exploration-based activity. Users familiarize themselves with the environment by programming interactively, one line at a time, as shown in Figure 4(b). If a statement does not accomplish the desired behavior, or causes an error, the user can immediately type a different statement and observe the result. Users can explore the behavior of both built-in functions and the ones they have defined themselves. This low turnaround time encourages experimentation in programming. Additionally, programs running on the desktop can send Python commands to be run on the robot. This facility was used to easily write a GUI, shown in Figure 4(a), that allows motors and sensors to be used interactively.

We have implemented a high-level API which allows the students to use a single function call to adjust a motor or get an encoder value. The simplicity of Python coupled with the API allows the students to accomplish fairly complex tasks with a relatively small amount of code. This increases students' ability to focus on logically solving the problem rather than on programming complexities. The memory game Simon uses long light patterns the player must repeat to continue playing. In our



(a) GUI for interacting with the Robot. (b) Interpreter prompt on the robot.

Fig. 4 a: The GUI allows a new user to quickly visualize sensor readings and test different motor settings. The GUI also allows for sending and receiving radio and IR messages to other robots. **b:** The interactive prompt serves as a debugging interface. Any errors in robot operation or programming are returned to the user here.

class at Rice, students were able to implement Simon after one week of instruction on basic Python syntax. In less than 80 lines of code, students were able to create a fully functioning game that runs on the robot and uses the LEDs and buttons. This assignment enabled new students to learn how to program the robot with an interesting program that utilized the high level robot API.

Creating robot behaviors in Python is also straight forward and simple. The second lab activity was to create a Braitenberg vehicle that either moves toward or away from light after practicing some basic programming. It requires even less code than Simon to create the third vehicle, Love, in Valentino Braitenberg's book *Vehicles: Experiments in Synthetic Psychology*. This vehicle uses two sensors to drive towards a light source. Figure 5 shows the Python code to implement this behavior. Seeking the light first requires reading the values from the two light sensors. The robot then needs to determine which side collected the higher value. With this information, the robot can drive the motors in proportion to the difference between the two readings. While not shown in the figure, in practice, the duty cycle of the PWM signals that drive the motors need to be clamped to ensure that no values are too high or too low.

In a lab exercise , students created a velocity feedback control loop with less than 150 lines of code, completely composed of basic Python constructs. The low cost of the r-one robot allowed us to give a robot to each student, and the python programming language significantly lowers the barriers to programming for young students.

7.3 Outreach

A low-cost robot makes the cost of deploying a medium-scale (around 20) multi-robot system within the reach of many middle and high schools. This is a novel and unique way to motivate the next generation of engineers and computer scien-

```
def seek(k):
   ## seeks the light using front sensors (fl, fr) and
   ## adjusts motor PWM values based off of the readings
   while True:
      # get light sensor values
      fr = rone.light_sensor_get_value('fr')
      fl = rone.light_sensor_get_value('fl')

      # calculate duty cycles
      difference = fr - fl
      duty_cycle_r = 60 - (difference * k)
      duty_cycle_l = 60 + (difference * k)

      # run motors
      rone.motor_set_pwm('r',int(duty_cycle_r))
      rone.motor_set_pwm('l',int(duty_cycle_l))
```

Fig. 5 Source listing for Braitenberg vehicle "love"

tists. This type of system allows robot demonstrations to become more engaging to students and spectators.

Equipped with a box of 20 robots, the visiting educator can give each student their own robot, pre-loaded with software for the day's activity. As a group, the class can use the entire population of robots to develop and test basic multi-robot algorithms, like ad-hoc network formation, communications relaying, navigation, and leader election. The Python programming language makes it possible for younger students to participate in programming while simultaneously allowing advanced students to experiment at their pace without significant documentation or training.

8 Conclusions and Future Work

Using a robot platform in an introductory classroom setting creates a unique challenge. To be interesting for college students, the available sensor suite must be sufficient to build complex behaviors. However, to be accessible to first year students, the software stack must be easy to program. Finally, to allow each student access to their own robot, the entire system must be cheap. These constraints eliminated existing systems, so we have built a novel hardware/software robot platform, the r-one. Our robot combines an advanced sensor suite that allows multi-robot system behavior with an embedded Python interpreter, easily allowing rapid development of interesting systems. We have deployed a robot to each of the twenty-three people in our freshman class, all of whom have been able to program interesting behaviors in a few weeks.

In the future, we expect the r-one to be a powerful platform for more advanced multi-robot system research. The small size and low cost of our robot will allow experimentation of much larger swarms than was previously possible, uncovering new challenges in the field. The simplicity of programming will help this research, since new algorithms can be developed and explored more easily. Finally, the r-one

robot can be used in settings where traditional robot platforms are impractically expensive or complex, such as in elementary and high-school settings.

For future versions of the hardware, we are planing a full-coverage bump skirt, an audio circuit and speaker, and line trackers/cliff detectors.

Acknowledgements. Several exceptionally talented undergraduate students have contributed to this project: Brandon Heath, Brian Shields, Linge Dai, James Hill, Joan Chao, Lan Li, Chris Licato, Nelson Chen and Joshua Bryant. Dr. McLurkin would like to thank his University of Washington CSE491 class for their many comments and suggestions on earlier multi-robot curriculum.

References

1. Abramson, N.: The aloha system: Another alternative for computer communications. Technical Report B70-1. University of Hawaii, Honolulu (1970)
2. Das, A., Fierro, R., Kumar, V., Ostrowski, J., Taylor, C.J.: A vision-based formation control framework. IEEE Transactions on Robotics and Automation 18(5), 813–826 (2002)
3. Dodds, Z., Libeskind-Hadas, R., Alvarado, C., Kuenning, G.: Evaluating a breadth-first cs 1 for scientists. In: SIGCSE 2008: Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education, pp. 266–270. ACM, New York (2008), doi:http://doi.acm.org/10.1145/1352135.1352229
4. Feddema, J., Lewis, C., Schoenwald, D.: Decentralized control of cooperative robotic vehicles: theory and application. IEEE Transactions on Robotics and Automation 18(5), 852–864 (2002)
5. Gutierrez, A., Campo, A., Dorigo, M., Donate, J., Monasterio-Huelin, F., Magdalena, L.: Open e-puck range and bearing miniaturized board for local communication in swarm robotics. In: Proceedings of the 2009 IEEE International Conference on Robotics and Automation, pp. 1745–1750. IEEE Press, Kobe (2009)
6. Howard, A., Parker, L.E., Sukhatme, G.S.: The SDR experience: Experiments with a Large-Scale heterogenous mobile robot team. In: 9th International Symposium on Experimental Robotics, Singapore (2004)
7. Howard, A., Parker, L.E., Sukhatme, G.S.: Experiments with large heterogeneous mobile robot team: Exploration, mapping, deployment and detection. International Journal of Robotics Research 25(5), 431–447 (2006)
8. Lochmatter, T., Roduit, P., Cianci, C., Correll, N., Jacot, J., Martinoli, A.: SwisTrack - A Flexible Open Source Tracking Software for Multi-Agent Systems. In: Proceedings of the IEEE/RSJ 2008 International Conference on Intelligent Robots and Systems (IROS 2008), pp. 4004–4010. IEEE (2008), doi:http://iros2008.inria.fr/
9. Mataric, M.J.: Interaction and intelligent behavior. Ph.D. thesis, Massachusetts Institute of Technology (1994)
10. McLurkin, J.: Stupid robot tricks: A Behavior-Based distributed algorithm library for programming swarms of robots. S.M. thesis, Massachusetts Institute of Technology (2004)
11. McLurkin, J.: Analysis and implementation of distributed algorithms for Multi-Robot systems. Ph.D. thesis, Massachusetts Institute of Technology (2008)
12. McLurkin, J.: Experiment design for large Multi-Robot systems. In: Robotics: Science and Systems, Workshop on Good Experimental Methodology in Robotics, Seattle, WA, USA (2009)
13. McLurkin, J., Smith, J., Frankel, J., Sotkowitz, D., Blau, D., Schmidt, B.: Speaking swarmish: Human-Robot interface design for large swarms of autonomous mobile robots. In: Proceedings of AAAI Spring Symposium (2006)

14. Michael, N., Fink, J., Kumar, V.: Experimental testbed for large multirobot teams. IEEE Robotics & Automation Magazine 15(1), 53–61 (2008)
15. Olson, E.: Apriltag: A robust and flexible multi-purpose fiducial system. Tech. rep., University of Michigan APRIL Laboratory (2010)
16. Payton, D., Estkowski, R., Howard, M.: Compound behaviors in pheromone robotics. Robotics and Autonomous Systems 44(3-4), 229–240 (2003)
17. Pedersen, R.U., Nørbjerg, J., Scholz, M.P.: Embedded programming education with lego mindstorms nxt using java (lejos), eclipse (xpairtise), and python (pymite). In: WESS 2009: Proceedings of the 2009 Workshop on Embedded Systems Education, pp. 50–55. ACM, New York (2009)
18. Priyantha, N.B., Chakraborty, A., Balakrishnan, H.: The cricket location-support system. In: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, pp. 32–43. ACM, Boston (2000)
19. Pugh, J., Martinoli, A.: Relative localization and communication module for small-scale multi-robot systems. In: Proceedings 2006 IEEE International Conference on Robotics and Automation, ICRA 2006, pp. 188–193 (2006)
20. Radenski, A.: "python first": a lab-based digital introduction to computer science. In: ITICSE 2006: Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, pp. 197–201. ACM, New York (2006), doi:http://doi.acm.org/10.1145/1140124.1140177
21. Shannon, C.: Another breadth-first approach to cs i using python. In: SIGCSE 2003: Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education, pp. 248–251. ACM, New York (2003), doi:http://doi.acm.org/10.1145/611892.611980
22. Veloso, M.M., Bowling, M., Achin, S., Han, K., Stone, P.: The CMUnited-98 Champion Small-Robot Team. In: Asada, M., Kitano, H. (eds.) RoboCup 1998. LNCS (LNAI), vol. 1604, pp. 77–92. Springer, Heidelberg (1999)

Author Index