Springer Tracts in Advanced Robotics

Volume 55

Cyrill Stachniss

Robotic Mapping and Exploration

Springer

Professor Bruno Siciliano, Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II, Via Claudio 21, 80125 Napoli, Italy, E-mail: siciliano@unina.it

Professor Oussama Khatib, Artificial Intelligence Laboratory, Department of Computer Science, Stanford University, Stanford, CA 94305-9010, USA, E-mail: khatib@cs.stanford.edu

Professor Frans Groen, Department of Computer Science, Universiteit van Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, The Netherlands, E-mail: groen@science.uva.nl

Author

Dr. Cyrill Stachniss
Albert-Ludwigs-University of Freiburg
Institute of Computer Science
Autonomous Intelligent Systems
Georges-Koehler-Allee 079
79110 Freiburg
Germany
E-mail: stachnis@informatik.uni-freiburg.de

ISBN 978-3-642-01096-5 e-ISBN 978-3-642-01097-2

DOI 10.1007/978-3-642-01097-2

Springer Tracts in Advanced Robotics ISSN 1610-7438

Library of Congress Control Number: Applied for

Typeset & Cover Design: Scientific Publishing Services Pvt. Ltd., Chennai, India.

Printed in acid-free paper

5 4 3 2 1 0

springer.com

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

For Janna and Maren

Series Editor's Foreword

By the dawn of the new millennium, robotics has undergone a major transformation in scope and dimensions. This expansion has been brought about by the maturity of the field and the advances in its related technologies. From a largely dominant industrial focus, robotics has been rapidly expanding into the challenges of the human world. The new generation of robots is expected to safely and dependably co-habitat with humans in homes, workplaces, and communities, providing support in services, entertainment, education, healthcare, manufacturing, and assistance.

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 goal of the series of Springer Tracts in Advanced Robotics (STAR) is to bring, in a timely fashion, the latest advances and developments in robotics on the basis of their significance and quality. It is our hope that the wider dissemination of research developments will stimulate more exchanges and collaborations among the research community and contribute to further advancement of this rapidly growing field.

The monograph written by Cyrill Stachniss is a contribution in the area of self-localization and mapping (SLAM) for autonomous robots, which has been receiving a great deal of attention by the research community in the latest few years. The contents expand the authors doctoral dissertation and are focused on the autonomous mapping learning problem. Solutions include uncertainty-driven exploration, active loop closing, coordination of multiple robots, learning and incorporating background knowledge, and dealing with dynamic environments. Results are accompanied by a rich set of experiments, revealing a promising outlook toward the application to a wide range of

mobile robots and field settings, such as search and rescue, transportation tasks, or automated vacuum cleaning.

Yet another STAR volume on SLAM, a very fine addition to the series!

Naples, Italy
February 2009

Bruno Siciliano
STAR Editor

Foreword

Simultaneous localization and mapping is a highly important and active area in mobile robotics. The ability to autonomously build maps is widely regarded as one of the fundamental preconditions for truly autonomous mobile robots. In the past, the SLAM has mostly been addressed as a state estimation problem and the incorporation of control into the map learning and localization process is a highly interesting research question. In this book by Cyrill Stachniss, the reader will find interesting and innovative solutions to the problem of incorporating control into the SLAM problem. I know Cyrill since over eight years and I still appreciate his enthusiasm in developing new ideas and getting things done. He has been working with a large number of different robots, participating in several public demonstrations, and has gained a lot of experience which can also be seen from his large number of papers presented at all major robotic conferences and in journals. His work covers a variety of different topics. He has acquired several project grants and received several awards. He furthermore is an associate editor of the IEEE Transactions on Robotics. It's safe to say that he is an expert in his field.

This book is a comprehensive introduction to state-of-the-art technology in robotic exploration and map building. The reader will find a series of solutions to challenging problems robots are faced with in the real world when they need to acquire a model of their surroundings. The book focuses on autonomy and thus the robot is not supposed to be joysticked though the world but should be able to decide about his actions on its own. I regard the ability to learn maps by making own decisions as a key competence for autonomous robots. Cyrill rigorously applies probabilistic and decision-theoretic concepts to systematically reducing the uncertainty in the belief of a robot about its environment and its pose in the environment.

The book contains impressively demonstrates the capabilities of the described solutions by showing results obtained from real robotic datasets. A further strength lies in the sound and thorough evaluation of all presented techniques going beyond the world of simulation. At this point, I would like to encourage the reader to follow Cyrill's example to take real

robots and data obtained with real robots to demonstrate that novel approaches work in reality. For readers not in possession of particular sensors or for comparison purposes, Cyrill and colleagues have created a Web site (http://www.openslam.org/) in which the community can share implementations of SLAM approaches and where the reader will find links to datasets to support future research.

Freiburg, Germany
February 2009

Wolfram Burgard

Preface

Models of the environment are needed for a wide range of robotic applications including search and rescue, transportation tasks, or automated vacuum cleaning. Learning maps has therefore been a major research topic in the robotics community over the last decades. Robots that are able to reliably acquire an accurate model of their environment on their own are regarded as fulfilling a major precondition of truly autonomous agents. To autonomously solve the map learning problem, a robot has to address mapping, localization, and path planning at the same time. In general, these three tasks cannot be decoupled and solved independently. Map learning is thus referred to as the simultaneous planning, localization, and mapping problem. Because of the coupling between these tasks, this is a complex problem. It can become even more complex when there are dynamic changes in the environment or several robots are being used together to solve the problem.

This book presents solutions to various aspects of the autonomous map learning problem. The book is separated into two parts. In the first part, we assume the position of the robot to be known. This assumption does not hold in the real world, however, it makes life easier and allows us to better concentrate on certain aspects of the exploration problem such as coordinating a team of robots. We describe how to achieve appropriate collaboration among exploring robots so that they efficiently solve their joint task. We furthermore provide a technique to learn and make use of background knowledge about typical spatial structures when exploring an environment as a team.

In the second part, we relax the assumption that the pose of the robot is known. To deal with the uncertainty in the pose of a robot, we present an efficient solution to the simultaneous localization and mapping problem. The difficulty in this context is to build a map while at the same time localizing the robot in this map. The presented approach maintains a joint posterior about the trajectory of the robot and the model of the environment. It produces accurate maps in an efficient and robust way. After addressing step-by-step the different problems in the context of active map learning, we integrate the main techniques into a single system. We present an integrated approach

that simultaneously deals with mapping, localization, and path planning. It seeks to minimize the uncertainty in the map and in the trajectory estimate based on the expected information gain of future actions. It takes into account potential observation sequences to estimate the uncertainty reduction in the world model when carrying out a specific action. Additionally, we focus on mapping and localization in non-static environments. The approach allows a robot to consider different spatial configurations of the environment and in this way makes the pose estimate more robust and accurate in non-static worlds.

In sum, the contributions of this book are solutions to various problems of the autonomous map learning problem including uncertainty-driven exploration, SLAM, active loop closing, coordination of multiple robots, learning and incorporating background knowledge, and dealing with dynamic environments.

A lot of the work presented in this book has been done in collaboration with other researchers. It was a pleasure for me to work with all the wonderful people in the AIS lab in Freiburg. First of all, I thank Wolfram Burgard for his tremendous support, his inspiration, and for providing a creative atmosphere. My thanks to my friends and colleagues for the great time in the lab, especially to Maren Bennewitz, Giorgio Grisetti, Dirk Hähnel, Óscar Martínez Mozos, Patrick Pfaff, Christian Plagemann, and Axel Rottmann for the great collaboration on the topics addressed in this book. It was a pleasure to work with all these people and to benefit from their knowledge. My thanks also to Mark Moors and Frank Schneider for the collaboration on multi-robot exploration. Special thanks to Nick Roy and Mike Montemerlo who did a great job in developing and maintaining the Carnegie Mellon Robot Navigation Toolkit. It was a pleasure for me to work together with all of them.

Additionally, I thank several people, who published robot datasets and in this way helped to make mapping approaches more robust and more easily comparable. In this context, I would like to thank Patrick Beeson, Mike Bosse, Udo Frese, Steffen Gutmann, Dirk Hähnel, Andrew Howard, and Nick Roy.

Freiburg, Germany
December 2008

Cyrill Stachniss

Contents

1 Introduction 3

2 Basic Techniques 7
2.1 Introduction to Particle Filters 7
2.1.1 Mobile Robot Localization Using Particle Filters 11
2.2 Grid Maps 13
2.2.1 Occupancy Probability Mapping 14
2.2.2 Sensor Model for a Laser Range Finder 16
2.2.3 Sensor Model for a Sonar Sensor 16
2.2.4 Reflection Probability Mapping 18

Part I: Exploration with Known Poses

3 Decision-Theoretic Exploration Using Coverage Maps . . . 23
3.1 Introduction 23
3.2 Definition of Coverage Maps 24
3.3 Updating Coverage Maps Upon Sensory Input 25
3.4 Decision-Theoretic Exploration with Coverage Maps 29
3.4.1 Choosing the Closest Target Location 29
3.4.2 Exploration Using the Information Gain 30
3.4.3 Using IG in a Local Window 32
3.4.4 Combination of IG and CL 32
3.5 Exploration Using Occupancy Grid Maps 32
3.6 Experimental Results 33
3.6.1 Mapping with Noisy Sensors 33
3.6.2 Comparing the Viewpoint Selection Strategies 34
3.6.3 Advantage over Scan Counting 37
3.7 Related Work 38
3.8 Conclusion 40

4 Coordinated Multi-Robot Exploration 43
4.1 Introduction 43
4.2 Coordinating a Team of Robots during Exploration 45
4.2.1 Cost of Reaching a Target Location 46
4.2.2 Computing Utilities of Frontier Cells 47
4.2.3 Target Point Selection 48
4.2.4 Coordination with Limited Communication Range . . . 49
4.3 Collaborative Mapping with Teams of Mobile Robots 50
4.4 Experimental Results 51
4.4.1 Exploration with a Team of Mobile Robots 51
4.4.2 Comparison between Uncoordinated and Coordinated Exploration 52
4.4.3 Simulation Experiments 54
4.4.4 Exploration with Limited Communication 58
4.5 Comparisons to Other Coordination Techniques 59
4.5.1 Target Assignment Using the Hungarian Method 59
4.5.2 Using a Priorization Scheme to Coordinate a Team of Robots 64
4.5.3 Coordination of a Team of Robots by Solving a TSP 66
4.6 Related Work 66
4.7 Conclusion 71

5 Multi-Robot Exploration Using Semantic Place Labels 73
5.1 Introduction 73
5.2 Semantic Place Labeling 74
5.3 Estimating the Label of a Goal Location 77
5.4 Using Semantic Place Information for Efficient Multi-Robot Exploration 80
5.5 Experimental Results 81
5.5.1 Performance Improvement Using Semantic Place Information 81
5.5.2 Influence of Noise in the Semantic Place Information 84
5.5.3 Applying a Trained Classifier in New Environments 84
5.5.4 Improvements of the HMM Filtering and Error Analysis of the Classifier 85
5.6 Related Work 88
5.7 Conclusion 89

Part II: Mapping and Exploration under Pose Uncertainty

6 Efficient Techniques for Rao-Blackwellized Mapping 93
6.1 Introduction 93
6.2 The Concept of Rao-Blackwellized Mapping 94
6.3 Improved Proposals and Selective Resampling 96
6.3.1 Using Laser Range Data to Compute an Improved Proposal Distribution 97
6.3.2 Selective Resampling 100
6.4 Complexity 101
6.5 Experimental Results 102
6.5.1 Mapping Results 103
6.5.2 Quantitative Results 105
6.5.3 Effects of Improved Proposals and Adaptive Resampling 107
6.5.4 Situations in Which the Scan-Matcher Fails 109
6.5.5 Computational Cost 112
6.6 Related Work 112
6.7 Conclusion 115

7 Actively Closing Loops During Exploration 117
7.1 Introduction 117
7.2 Active Loop-Closing 118
7.2.1 Detecting Opportunities to Close Loops 119
7.2.2 Representing Actions under Pose Uncertainty 121
7.2.3 Stopping the Loop-Closing Process 122
7.2.4 Reducing the Exploration Time 124
7.2.5 Handling Multiple Nested Loops 126
7.3 Experimental Results 126
7.3.1 Real World Exploration 127
7.3.2 Active Loop-Closing vs. Frontier-Based Exploration 127
7.3.3 A Quantitative Analysis 128
7.3.4 Importance of the Termination Criterion 129
7.3.5 Evolution of N_{eff} 130
7.3.6 Multiple Nested Loops 132
7.3.7 Computational Resources 132
7.4 Related Work 133
7.5 Conclusion 134

8 Recovering Particle Diversity 135
8.1 Introduction 135
8.2 Recovering Particle Diversity after Loop Closure 137
8.3 Experimental Results 138

8.4 Related Work . . . 141
8.5 Conclusion . . . 142

9 Information Gain-based Exploration . . . 143
9.1 Introduction . . . 143
9.2 The Uncertainty of a Rao-Blackwellized Mapper . . . 145
9.3 The Expected Information Gain . . . 147
9.4 Computing the Set of Actions . . . 151
9.5 Experimental Results . . . 152
9.5.1 Real World Application . . . 152
9.5.2 Decision Process . . . 154
9.5.3 Comparison to Previous Approaches . . . 154
9.5.4 Corridor Exploration . . . 157
9.6 Related Work . . . 157
9.7 Conclusion . . . 160

10 Mapping and Localization in Non-Static Environments . . . 161
10.1 Introduction . . . 161
10.2 Learning Maps of Low-Dynamic Environments . . . 163
10.2.1 Map Segmentation . . . 163
10.2.2 Learning Configurations of the Environment . . . 163
10.2.3 Map Clustering . . . 164
10.3 Monte-Carlo Localization Using Patch-Maps . . . 166
10.4 Experimental Results . . . 168
10.4.1 Application in an Office Environment . . . 168
10.4.2 Localizing the Robot and Estimating the State of the Environment . . . 171
10.4.3 Global Localization . . . 173
10.5 Related Work . . . 173
10.6 Conclusion . . . 175

11 Conclusion . . . 177

A Appendix . . . 181
A.1 Probability Theory . . . 181
A.1.1 Product Rule . . . 181
A.1.2 Independence . . . 181
A.1.3 Bayes' Rule . . . 181
A.1.4 Marginalization . . . 182
A.1.5 Law of Total Probability . . . 182
A.1.6 Markov Assumption . . . 182

References . . . 183

Index . . . 195

Notation

Throughout this book, we make use of the following notation:

variable	description
x_t	pose of the robot at time step t. This pose is a three dimensional vector containing the x, y-position and the orientation θ of the vehicle
$x_{1:t}$	sequence of poses of the robot from time step 1 to time step t
z_t	sensor observation obtained at time step t
u_t	odometry information describing the movement from x_t to x_{t+1}
a	action or motion command
w	importance weight
$w_t^{[i]}$	importance weight of the i-th particle at time step t
m	grid map
c	grid cell
r	resolution of a grid map. Each cell covers an area of r by r.
$\mathcal{G}$	topological map
$E[]$	expectation
$\mathcal{N}(\mu, \Sigma)$	Gaussian with mean μ and covariance Σ
H	entropy
I	information gain
U	utility function
V	cost function
η	normalizer, typically resulting from Bayes' rule
N_{eff}	effective number of particles

1

Introduction

Models of the environment are needed for a wide range of robotic applications, from search and rescue to automated vacuum cleaning. Learning maps has therefore been a major research focus in the robotics community over the last decades.

In general, learning maps with single-robot systems requires the solution of three tasks, which are *mapping*, *localization*, and *path planning*. Mapping is the problem of integrating the information gathered with the robot's sensors into a given representation. It can be described by the question "What does the world look like?" Central aspects in mapping are the representation of the environment and the interpretation of sensor data. In contrast to this, localization is the problem of estimating the pose of the robot relative to a map. In other words, the robot has to answer the question, "Where am I?" Typically, one distinguishes between pose tracking, where the initial pose of the vehicle is known, and global localization, in which no a priori knowledge about the starting position is given. Finally, the path planning or motion control problem involves the question of how to efficiently guide a vehicle to a desired location or along a trajectory. Expressed as a simple question, this problem can be described as, "How can I reach a given location?"

Unfortunately, these three tasks cannot be solved independently of each other. Before a robot can answer the question of what the environment looks like given a set of observations, it needs to know from which locations these observations have been made. At the same time, it is hard to estimate the current position of a vehicle without a map. Planning a path to a goal location is also tightly coupled with the knowledge of what the environment looks like as well as with the information about the current pose of the robot.

The diagram in Figure 1.1 depicts the mapping, localization, and path planning tasks as well as the combined problems in the overlapping areas. *Simultaneous localization and mapping* (SLAM) is the problem of building a map while at the same time localizing the robot within that map. One cannot decouple both tasks and solve them independently. Therefore, SLAM is often referred to as a chicken or egg problem: A good map is needed for localization

C. Stachniss: Robotic Mapping and Exploration, STAR 55, pp. 3–6.
springerlink.com

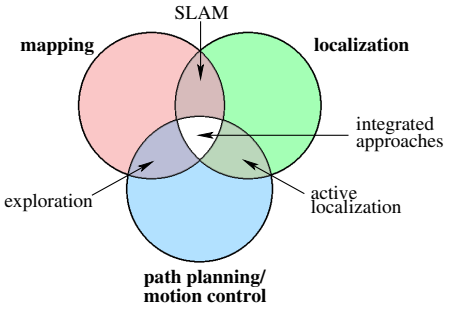


Fig. 1.1. Tasks that need to be solved by a robot in order to acquire accurate models of the environment. The overlapping areas represent combinations of the mapping, localization, and path planning tasks [94].

while an accurate pose estimate is needed to build a map. *Active localization* seeks to guide the robot to locations within the map to improve the pose estimate. In contrast to this, *exploration* approaches assume accurate pose information and focus on guiding the robot efficiently through the environment in order to build a map. The center area of the diagram represents the so-called *integrated approaches* which address mapping, localization, and path planning simultaneously. The integrated approaches are also called solutions to the *simultaneous planning, localization, and mapping* (SPLAM) problem. A solution to the SPLAM problem enables a mobile robot to acquire sensor data by autonomously moving through its environment while at the same time building a map. Whenever the robot is moving, it considers actions to improve its localization, to acquire information about unknown terrain, and to improve its map model by revisiting areas it is uncertain about. In the end, the robot is assumed to have learned an accurate model of the whole environment as well as determined its own pose relative to this model.

Several researchers focus on different aspects of these problems. This is done using single robot systems as well as teams of robots. The use of multiple robots has several advantages over single robot systems. Cooperating robots have the potential to accomplish a task faster than a single one. Furthermore, teams of robots can be expected to be more fault-tolerant than a single robot. However, when robots operate in teams, there is the risk of possible interference between them. The more robots that are used in the same environment, the more time each robot may spend on detours in order to avoid collisions with other members of the team. In most approaches, the performance of the team is measured in terms of the overall time needed to learn a map. This means that the robots need to be distributed over the environment in order to avoid redundant work and to reduce the risk of interference. A team of robots makes finding efficient solutions to problems like exploration more complex, since more agents are involved and so more decisions need to be made.

It is worth mentioning that all these problems become even more complex in the case where the environment changes over time. Most mapping techniques assume that the environment is static and does not change over time. This, however, is an unrealistic assumption, since most places where robots are used are populated by humans. Changes are often caused by people walking through the environment, by open and closed doors, or even by moved furniture. One possibility to deal with dynamic aspects is to filter them out and to map the static objects only. More challenging, however, is the problem of integrating the information about changes into the map and utilizing such knowledge in other robotic applications. This can enable a mobile robot to more efficiently execute its tasks. For example, one can expect a robot to more robustly localize itself in case where it knows about the typical configurations of the non-static aspects in its surroundings.

In summary, the key problems in the context of map learning are the questions of

- where to guide a robot during autonomous exploration,
- how to deal with noise in the pose estimate and in the observations,
- how to deal with the uncertainty in the robot's world model and how to interprete the sensor data,
- how to model changes in the environment over time, and
- how to efficiently coordinate a team of mobile robots.

The contributions presented in this book are solutions to different aspects of the map learning problem which explicitely consider these five aspects. We present approaches to autonomous exploration that take into account the uncertainty in the world model of the robot. We minimize this uncertainty by reasoning about possible actions to be carried out and their expected reward. We furthermore describe how to achieve good collaboration among a team of robots so that they efficiently solve an exploration task. Our approach effectively distributes the robots over the environment and in this way avoids redundant work and reduces the risk of interference between vehicles. As a result, the overall time needed to complete the exploration mission is reduced. To deal with the uncertainty in the pose of a robot, we present a highly accurate technique to solve the SLAM problem. Our approach maintains a joint posterior about the trajectory of the robot and the map model. It produces highly accurate maps in an efficient and robust way. In this book, we address step-by-step the problems in the context of map learning and integrate different solutions into a single system. We provide an integrated approach that simultaneously deals with mapping, localization, and path planning. It seeks to minimize the uncertainty in the map and trajectory estimate based on the expected information gain of future actions. It takes into account potential observation sequences to estimate the uncertainty reduction in the world model when carrying out a specific action. Additionally, we focus on mapping and localization in non-static environments. Our approach allows the robot

to consider different spatial configurations of the environment and in this way makes the pose estimate more robust and accurate in non-static worlds.

This book is organized as follows. First, we introduce the particle filtering technique and the ideas of grid maps. The first part of this book concentrates on single- and multi-robot exploration given the poses of the robots are known while they move through the environment.

Chapter 3 addresses the problem of decision-theoretic, autonomous exploration with a single vehicle. We consider a sensor which is affected by noise and investigate a technique to steer a robot through the environment in order to reduce the uncertainty in the map model.

In Chapter 4, we explore how to coordinate a team of robots in order to achieve effective collaboration and to avoid redundant work. The presented approach is extended in Chapter 5 so that background information about the structure of the environment is integrated into the coordination procedure. The knowledge about different structures is learned by the mobile robots from sensor data.

In the second part of this book, we relax the assumption of known poses and consider the uncertainty in the pose the a mobile robot. We present in Chapter 6 an efficient solution to the SLAM problem. It allows us to learn highly accurate grid maps while the pose information of the robot is affected by noise. Our technique maintains the joint posterior about the map and the trajectory of the robot using a particle filter. Chapter 7 describes a system to detect and to actively close loops during exploration. With this technique, we are not optimizing the pose estimation procedure but are planning appropriate trajectories for the mobile robot. The revisiting of known locations from time to time allows the robot to reduce the uncertainty in its pose. As a result, the obtained map is better aligned and shows less inconsistencies.

Actively revisiting known areas during SLAM offers not only the possibility to relocalize a vehicle, it also introduces the risk of becoming overly confident especially in the context of nested loops. To cope with this limitation, we present in Chapter 8 an approach for recovering the particle diversity after closing loops. This allows the robot to stay an arbitrary period of time within a loop without depleting important state hypotheses.

In Chapter 9, we present a decision-theoretic approach to exploration with respect to the uncertainty in the map and the pose estimate of the robot. The presented algorithm integrates different techniques introduced in the preceding chapters. It simultaneously addresses mapping, localization, and planning. As a result, our approach enables a real mobile robot to autonomously learn a model of the environment with low uncertainty even if its pose estimates are affected by noise.

Finally, Chapter 10 addresses the problem of mapping and localization in non-static environments. By explicitly modeling the different states the environment is observed in, the robot is able to more robustly localize itself in a non-static world.

2

Basic Techniques

This chapter explains two techniques which are frequently used throughout this book. First, we will introduce the concept of particle filters. A particle filter is a recursive Bayesian technique for estimating the state of a dynamic system. We then explain the ideas of grid maps and "mapping with known poses". Note that elementary laws in the context of probability theory can be found in the Appendix.

2.1 Introduction to Particle Filters

A particle filter is a nonparametric implementation of the Bayes filter and is frequently used to estimate the state of a dynamic system. The key idea is to represent a posterior by a set of hypotheses. Each hypothesis represents one potential state the system might be in. The state hypotheses are represented by a set S of N weighted random samples

$$S = \left\{ \left\langle s^{[i]}, w^{[i]} \right\rangle \mid i = 1, \ldots, N \right\}, \tag{2.1}$$

where $s^{[i]}$ is the state vector of the i-th sample and $w^{[i]}$ the corresponding importance weight. The weight is a non-zero value and the sum over all weights is 1. The sample set represents the distribution

$$p(x) = \sum_{i=1}^{N} w_i \delta_{s^{[i]}}(x), \tag{2.2}$$

where $\delta_{s^{[i]}}$ is the Dirac function in the state $s^{[i]}$ of the i-th sample. Such set S of samples can be used to approximate arbitrary distributions. The samples are drawn from the distribution they should approximate. To illustrate such an approximation, Figure 2.1 depicts two distributions and their corresponding sample sets. In general, the more samples that are used, the better the

C. Stachniss: Robotic Mapping and Exploration, STAR 55, pp. 7–20.
springerlink.com

approximation. The ability to model multi-modal distributions by the set of samples is an advantage compared to a series of other filters. The Kalman filter [73], for example, is restricted to Gaussian distributions.

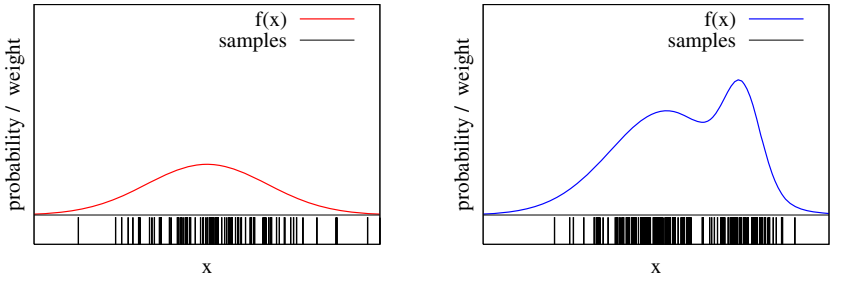


Fig. 2.1. Two functions and their approximations by samples with uniform weights. The samples are illustrated by the vertical bars below the two functions.

Whenever we are interested in estimating the state of a dynamic system over time, we can use the particle filter algorithm. The idea of this technique is to represent the distribution at each point in time by a set of samples, also called particles. The particle filter algorithm allows us to recursive estimate the particle set S_t based on the estimate S_{t-1} of the previous time step. The *sampling importance resampling* (SIR) particle filter can be summarized with the following three steps:

1. **Sampling:** Create the next generation S'_t of particles based on the previous set S_{t-1} of samples. This step is also called sampling or drawing from the proposal distribution.
2. **Importance Weighting:** Compute an importance weight for each sample in the set S'_t.
3. **Resampling:** Draw N samples form the set S'_t. Thereby, the likelihood to draw a particle is proportional to its weight. The new set S_t is given by the drawn particles.

In the following, we explain these three steps in more detail. In the first step, we draw samples in order to obtain the next generation of particles for the next time step. In general, the true probability distribution to sample particles from is not known or not in a suitable form for sampling. We show that it is possible to draw samples from a different distribution than the one we want to approximate. This technique is known as *importance sampling*.

We are faced with the problem of computing the expectation that $x \in A$, where A is a region. In general, the expectation $E_p[f(x)]$ of a function f is defined as

$$E_p[f(x)] = \int p(x) f(x)\, dx. \tag{2.3}$$

Let B be a function which returns 1 if its argument is true and 0 otherwise. We can express the expectation that $x \in A$ by

$$E_p[B(x \in A)] = \int p(x)B(x \in A)\,dx \tag{2.4}$$

$$= \int \frac{p(x)}{\pi(x)}\pi(x)B(x \in A)\,dx, \tag{2.5}$$

where π is a distribution for which we require that

$$p(x) > 0 \Rightarrow \pi(x) > 0. \tag{2.6}$$

Thus, we can define a weight $w(x)$ as

$$w(x) = \frac{p(x)}{\pi(x)}. \tag{2.7}$$

This weight w is used to account for the differences between p and the π. This leads to

$$E_p[B(x \in A)] = \int \pi(x)w(x)B(x \in A)\,dx \tag{2.8}$$

$$= E_\pi[w(x)B(x \in A)]. \tag{2.9}$$

Let us consider again the sample-based representations and suppose the sample are drawn from π. By counting all the particles that fall into the region A, we can compute the integral of π over A by the sum over samples

$$\int_A \pi(x)\;dx \approx \frac{1}{N}\sum_{i=1}^{N} B(s^{[i]} \in A). \tag{2.10}$$

If we consider the weights in this computation, we can compute the integral over p as

$$\int_A p(x)\;dx \approx \sum_{i=1}^{N} w^{[i]}B(s^{[i]} \in A). \tag{2.11}$$

It can be shown, that the quality of the approximation improves the more samples that are used. For an infinite set of samples, the sum over the samples converges to the integral

$$\lim_{N\to\infty}\sum_{i=1}^{N} w^{[i]}B(s^{[i]} \in A) = \int_A p(x)\;dx. \tag{2.12}$$

Let p be the probability distribution which is not in a suitable form for sampling and π the one we actually sample from. In the context of importance

sampling, p is typically called the *target distribution* and π the *proposal distribution*.

This derivation tells us that we can sample from an arbitrary distribution π which fulfills (2.6) to approximate the distribution p by assigning an importance weight to each sample according to (2.7). This condition is needed to ensure that a state which might be sampled from p does not have zero probability under π. An example that depicts a weighted set of samples in case the proposal is different from the target distribution is shown in Figure 2.2. Note that the importance sampling principle requires that we can point-wise evaluate the target distribution. Otherwise, the computation of the weights would be impossible.

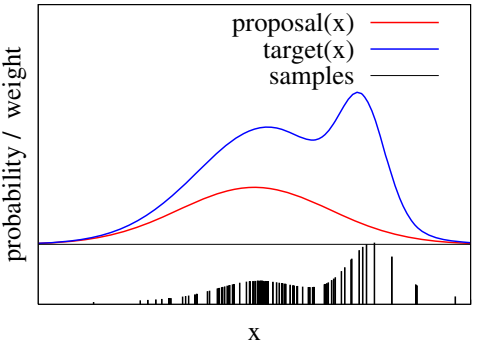


Fig. 2.2. The goal is to approximate the target distribution by samples. The samples are drawn from the proposal distribution and weighted according to (2.13). After weighting, the resulting sample set is an approximation of the target distribution.

Let $p(s_{1:t} \mid d)$ be the posterior to estimate, where d stands for all the data or background information. The importance weighting performed in Step 2 of the particle filter implementation (see Page 8) accounts for the fact one draws from the proposal π by setting the weight of each particle to

$$w_t^{[i]} = \eta \frac{p(s_{1:t}^{[i]} \mid d)}{\pi(s_{1:t}^{[i]} \mid d)}, \tag{2.13}$$

where η is the normalizer that ensures that the sum over all weights is 1.

The resampling step within a particle filter removes particles with a low importance weight and replaces them by particles with a high weight. After resampling, the weights are set to $1/N$ because by drawing according to the importance weight, one replaces "likelihoods" by "frequencies".

Resampling is needed since we use only a finite number of samples to approximate the target distribution. Without resampling, typically most particles would represent states with a low likelihood after some time and the

filter would loose track of the "good" hypotheses. On the one hand, this fact makes resampling important, on the other hand removing samples from the filter can also be problematic. In practice, it can happen that samples are replaced even if they are close to the correct state. This can lead to the so-called particle depletion or particle deprivation problem [29, 31, 155].

To reduce the risk of particle depletion, one can apply low-variance resampling. This technique does not draw the particles independently of each other in the resampling step. Instead of generating N random numbers to select N samples, the approach uses only a single random number to choose the first particle. The others are drawn depended on the first draw but still with a probability proportional to the individual weights. As a result, the particle set does not change during a resampling in case the weights are uniformly distributed. A detailed explanation on low-variance resampling as well as on particle filters in general can be found in [148]. The complete particle filter algorithm is listed in Algorithm 2.1.

Algorithm 2.1 The particle filter algorithm

Input: Sample set S_{t-1} and the data d.

1: $S'_t = \emptyset$
2: **for** i=1 to N **do**
3: draw $\hat{s} \sim \pi(s_t \mid s_{t-1}^{[i]}, d)$
4: $\hat{w} = \eta \left[p(\hat{s} \mid s_{t-1}^{[i]}, d) \right] \left[\pi(\hat{s} \mid s_{t-1}^{[i]}, d) \right]^{-1}$ // where η is a normalizer
5: $S'_t = S'_t + \langle \hat{s}, \hat{w} \rangle$
6: **end**
7: $S_t = \emptyset$
8: **for** j=1 to N **do**
9: draw a sample $s_t^{[i]}$ from S'_t. Thereby, $s_t^{[i]}$ is drawn with probability $w_t^{[i]}$
10: $S_t = S_t + \left\langle s_t^{[i]}, 1/N \right\rangle$
11: **end**
12: **return** S_t

2.1.1 Mobile Robot Localization Using Particle Filters

In the context of mobile robotics, particle filters are often used to track the position of the robot and we briefly illustrate the most important facts of Monte-Carlo localization [25]. In this scenario, the state vector s is the pose of the vehicle. Mostly, the motion estimate of the robot resulting from odometry is used to compute the proposal distribution in Step 1. The so-called motion model $p(x_t \mid x_{t-1}, u_{t-1})$ is used to draw the next generation of particles. In this case, the importance weight $w_t^{[i]}$ of the i-th sample has to be computed based on the observation likelihood $p(z_t \mid m, x_t^{[i]})$ of the most recent sensor observation z_t given a map m of the environment and the corresponding pose of the particle. This becomes clear by considering the following

derivations. We can transform the full posterior $p(x_{1:t} \mid m, z_{1:t}, u_{1:t-1})$ and obtain a recursive formula

$$\begin{aligned}
p(x_{1:t} \mid m, z_{1:t}, u_{1:t-1}) &\overset{\text{Bayes' rule}}{=} \eta p(z_t \mid m, x_{1:t}, z_{1:t-1}, u_{1:t-1}) \\
&\quad p(x_{1:t} \mid m, z_{1:t-1}, u_{1:t-1}) && (2.14)\\
&\overset{\text{Markov}}{=} \eta p(z_t \mid m, x_t) \\
&\quad p(x_{1:t} \mid m, z_{1:t-1}, u_{1:t-1}) && (2.15)\\
&\overset{\text{product rule}}{=} \eta p(z_t \mid m, x_t) \\
&\quad p(x_t \mid m, x_{1:t-1}, z_{1:t-1}, u_{1:t-1}) \\
&\quad p(x_{1:t-1} \mid m, z_{1:t-1}, u_{1:t-1}) && (2.16)\\
&\overset{\text{Markov}}{=} \eta p(z_t \mid m, x_t) p(x_t \mid x_{t-1}, u_{t-1}) \\
&\quad p(x_{1:t-1} \mid m, z_{1:t-1}, u_{1:t-2}), && (2.17)
\end{aligned}$$

where η is the normalizer resulting from Bayes' rule. Under the Markov assumption, we can transform the proposal as

$$\begin{aligned}
\pi(x_{1:t} \mid m, z_{1:t}, u_{1:t}) = \pi(x_t \mid m, x_{t-1}, z_t, u_{t-1}) \\
\pi(x_{1:t-1} \mid m, z_{1:t-1}, u_{1:t-2}). && (2.18)
\end{aligned}$$

The computation of the weights needs to be done according to (2.13). In the sequel, we drop the normalizer that ensures that all weights sum up to 1. This leads to

$$\begin{aligned}
w_t &= \frac{p(x_{1:t} \mid m, z_{1:t}, u_{1:t-1})}{\pi(x_{1:t} \mid m, z_{1:t}, u_{1:t-1})} && (2.19)\\
&= \frac{\eta p(z_t \mid m, x_t) p(x_t \mid x_{t-1}, u_{t-1})}{\pi(x_{1:t} \mid m, z_{1:t}, u_{1:t-1})} p(x_{1:t-1} \mid m, z_{1:t-1}, u_{1:t-2}) && (2.20)\\
&= \frac{\eta p(z_t \mid m, x_t) p(x_t \mid x_{t-1}, u_{t-1})}{\pi(x_t \mid m, x_{t-1}, z_t, u_{t-1})} \underbrace{\frac{p(x_{1:t-1} \mid m, z_{1:t-1}, u_{1:t-2})}{\pi(x_{1:t-1} \mid m, z_{1:t-1}, u_{1:t-2})}}_{w_{t-1}} && (2.21)\\
&= \frac{\eta p(z_t \mid m, x_t) p(x_t \mid x_{t-1}, u_{t-1})}{\pi(x_t \mid m, x_{t-1}, z_t, u_{t-1})} w_{t-1}. && (2.22)
\end{aligned}$$

If we choose the motion model as the proposal, we obtain for the i-the sample

$$\begin{aligned}
w_t^{[i]} &= \frac{\eta p(z_t \mid m, x_t^{[i]}) p(x_t \mid x_{t-1}^{[i]}, u_{t-1})}{p(x_t \mid x_{t-1}^{[i]}, u_{t-1})} w_{t-1}^{[i]} && (2.23)\\
&= \eta p(z_t \mid m, x_t^{[i]}) w_{t-1}^{[i]} && (2.24)\\
&\propto p(z_t \mid m, x_t^{[i]}) w_{t-1}^{[i]}. && (2.25)
\end{aligned}$$

Since the resampling step resets the weights of the whole set by $1/N$, we can ignore the weight of the previous time step and obtain

$$w_t^{[i]} \propto p(z_t \mid m, x_t^{[i]}). \tag{2.26}$$

This derivation shows that by choosing the motion model to draw the next generation of particles, we have to use the observation likelihood $p(z_t \mid m, x_t)$ to compute the individual weights.

To summarize this section, particle filters are a nonparametric implementations of the recursive Bayes filter. They use a set of weighted samples and can represent arbitrary distributions. The samples are drawn from a proposal distribution. After determining the importance weights which account for the fact that the target distribution is different from the proposal distribution, the resampling step replaces particles with a low weight by particles with a high importance weight.

In the techniques described in this book, we apply particle filters to solve the simultaneous localization and mapping problem. Furthermore, we apply them in the context of information gain-based exploration and to localize a mobile robot in dynamically changing environments.

2.2 Grid Maps

There exist different types of models for representing the environment which are frequently used in mobile robotics. The most common ones are feature maps, geometric maps, and grid maps. A feature map stores a set of features detected in the environment. Typical features are lines and corners when proximity sensors are used. Other possibilities are visual features based on the scale invariant feature transform (SIFT) [91] whenever a camera is used to perceive the environment. For each feature, these maps store the feature information together with a coordinate and eventually an uncertainty measure. This can be realized by a list of features or by using more efficient data structures like KD-trees [49, 11].

Geometric maps represent all obstacles detected by the robot as geometric objects, like circles or polygons. This kind of representation is comparably compact and needs only few memory resources.

The approaches described in this book use grid maps to model the environment. Grid maps discretize the environment into so-called grid cells. Each cell stores information about the area it covers. Most frequently used are occupancy grid maps that store for each cell a single value representing the probability that this cell is occupied by an obstacle. The advantage of grids is that they do not rely on predefined features which need to be extracted from sensor data. Furthermore, they offer a constant time access to grid cells and provide the ability to model unknown (unobserved) areas, which is an important feature in the context of exploration. However, they have the disadvantages of discretization errors and of requiring a lot of memory resources.

In this section, we first introduce the occupancy mapping algorithm, developed by Moravec and Elfes [108]. Afterwards, we briefly describe a variant called reflection probability maps. Both approaches are also referred to as "mapping with known poses."

2.2.1 Occupancy Probability Mapping

Grid maps discretize the environment into equally sized cells. Each cell represents the area of the environment it covers. It is assumed that each cell is either free or occupied by an obstacle. Occupancy grids store for each cell c a probability $p(c)$ of being occupied by an obstacle. In the following, we will derive the map update algorithm introduced by Moravec and Elfes which computes the occupancy probability $p(m)$ for the grid map m.

The algorithm takes into account a sequence of sensor observations $z_{1:t}$ obtained by the robot at the positions $x_{1:t}$ and seeks to maximize the occupancy probability for the grid map. One assumption in the algorithm of Moravec and Elfes is that the different cells are independent. Therefore, the probability of a map m is given by the product over the probabilities of the individual cells

$$p(m) = \prod_{c \in m} p(c). \tag{2.27}$$

In the following, we concentrate on the estimation of the occupancy probability of the individual cells $c \in m$. By applying Bayes' rule using $x_{1:t}$ and $z_{1:t-1}$ as background knowledge, we obtain

$$p(c \mid x_{1:t}, z_{1:t}) = \frac{p(z_t \mid c, x_{1:t}, z_{1:t-1}) p(c \mid x_{1:t}, z_{1:t-1})}{p(z_t \mid x_{1:t}, z_{1:t-1})}. \tag{2.28}$$

We assume that z_t is independent from $x_{1:t-1}$ and $z_{1:t-1}$. This leads to

$$p(c \mid x_{1:t}, z_{1:t}) = \frac{p(z_t \mid c, x_t) p(c \mid x_{1:t}, z_{1:t-1})}{p(z_t \mid x_{1:t}, z_{1:t-1})}. \tag{2.29}$$

We apply Bayes' rule for the term $p(z_t \mid c, x_t)$ in (2.29) and obtain

$$p(z_t \mid c, x_t) = \frac{p(c \mid x_t, z_t) p(z_t \mid x_t)}{p(c \mid x_t)}. \tag{2.30}$$

We can now combine (2.30) and (2.29). Let us furthermore assume that x_t does not carry any information about c if there is no observation z_t. This leads to

$$p(c \mid x_{1:t}, z_{1:t}) = \frac{p(c \mid x_t, z_t) p(z_t \mid x_t) p(c \mid x_{1:t-1}, z_{1:t-1})}{p(c) p(z_t \mid x_{1:t}, z_{1:t-1})}. \tag{2.31}$$

If we exploit the fact that each cell is a binary variable, we can derive the following equation in an analogous way

$$p(\neg c \mid x_{1:t}, z_{1:t}) = \frac{p(\neg c \mid x_t, z_t)p(z_t \mid x_t)p(\neg c \mid x_{1:t-1}, z_{1:t-1})}{p(\neg c)p(z_t \mid x_{1:t}, z_{1:t-1})}. \quad (2.32)$$

By dividing (2.31) by (2.32), we obtain

$$\frac{p(c \mid x_{1:t}, z_{1:t})}{p(\neg c \mid x_{1:t}, z_{1:t})} = \frac{p(c \mid x_t, z_t)p(\neg c)p(c \mid x_{1:t-1}, z_{1:t-1})}{p(\neg c \mid x_t, z_t)p(c)p(\neg c \mid x_{1:t-1}, z_{1:t-1})}. \quad (2.33)$$

Finally, we use the fact that $p(\neg c) = 1 - p(c)$ which yields

$$\frac{p(c \mid x_{1:t}, z_{1:t})}{1 - p(c \mid x_{1:t}, z_{1:t})} = \frac{p(c \mid x_t, z_t)}{1 - p(c \mid x_t, z_t)} \frac{1 - p(c)}{p(c)} \frac{p(c \mid x_{1:t-1}, z_{1:t-1})}{1 - p(c \mid x_{1:t-1}, z_{1:t-1})}. \quad (2.34)$$

If we define

$$\text{Odds}(x) = \frac{p(x)}{1 - p(x)}, \quad (2.35)$$

Equation (2.34) turns into

$$\text{Odds}(c \mid x_{1:t}, z_{1:t}) = \text{Odds}(c \mid x_t, z_t)\, \text{Odds}(c)^{-1}\, \text{Odds}(c \mid x_{1:t-1}, z_{1:t-1}). \quad (2.36)$$

This equation has a recursive structure similar to that of a recursive Bayesian update scheme. The corresponding log Odds representation of (2.36) is given by

$$\begin{aligned} \log \text{Odds}(c \mid x_{1:t}, z_{1:t}) = {} & \log \text{Odds}(c \mid z_t, x_t) \\ & - \log \text{Odds}(c) \\ & + \log \text{Odds}(c \mid x_{1:t-1}, z_{1:t-1}). \end{aligned} \quad (2.37)$$

The usage of the log Odds notation has advantage that it can be computed efficiently. It is only necessary to compute a sum in order to update a cell based on sensory input. To recover the occupancy probability from the Odds representation given in (2.36), we use the following formula which can easily be derived from (2.35):

$$p(x) = \frac{\text{Odds}(x)}{1 + \text{Odds}(x)} \quad (2.38)$$

This leads to the following *occupancy update formula*

$$p(c \mid x_{1:t}, z_{1:t}) = \left[1 + \frac{(1 - p(c \mid x_t, z_t))}{p(c \mid x_t, z_t)} \frac{p(c)}{(1 - p(c))} \frac{1 - p(c \mid x_{1:t-1}, z_{1:t-1})}{p(c \mid x_{1:t-1}, z_{1:t-1})}\right]^{-1}. \quad (2.39)$$

Equation (2.39) tells us how to update our belief $p(c \mid x_{1:t}, z_{1:t})$ about the occupancy probability of a grid cell given sensory input. In practice, one often assumes that the occupancy prior is 0.5 for all cells so that $\frac{p(c)}{(1-p(c))}$ can be removed from the equation.

It remains to describe how to compute the occupancy probability $p(c \mid x_t, z_t)$ of a grid cell given a *single* observation z_t and the corresponding pose x_t of the robot. This quantity strongly depends on the sensor of the robot and has to be defined manually for each type of sensor.

2.2.2 Sensor Model for a Laser Range Finder

In case a laser range finder is used, a quite simplistic model can be applied. Each cell c that is covered by the n-th beam $z_{t,n}$ of the observation z_t and whose distance to the sensor is shorter than the measured one, is supposed to be unoccupied. The cell in which the beam ends is supposed to be occupied. The function $dist(x_t, c)$ refers to the distance between the sensor and the center of the cell c. This can be formulated

$$p(c \mid z_{t,n}, x_t) = \begin{cases} p_{prior}, & z_{t,n} \text{ is a maximum range reading} \\ p_{prior}, & c \text{ is not covered by } z_{t,n} \\ p_{occ}, & |z_{t,n} - dist(x_t, c)| < r \\ p_{free}, & z_{t,n} \geq dist(x_t, c), \end{cases} \tag{2.40}$$

where r is the resolution of the grid map. Furthermore, it must hold $0 \leq p_{free} < p_{prior} < p_{occ} \leq 1$. Figure 2.3 depicts an example for such a sensor model for laser range finder data.

2.2.3 Sensor Model for a Sonar Sensor

In case a sonar sensor is used, the sensor model is slightly more complicated, since the sensor is not a beam sensor and the observations are more noisy than the ones of a laser range finder. In practice, one typically uses a mixture of three functions to express the model. First, the influence of an observation (which is represented by the difference between p_{prior} and p_{occ} as well as between p_{prior} and p_{free}) decreases with the measured distance.

Second, the proximity information of a sonar is substantially affected by noise. Therefore, one typically uses a piecewise linear function to model a smooth transition from p_{free} to p_{occ} as illustrated in Figure 2.4.

Finally, the sonar sensor should not be modeled as a beam sensor, since it sends out a conic signal. The accuracy of an observation decreases with the angular distance between the cell under consideration and the optical axis of the observation. This is expressed by the derivation from the prior and is typically modeled using a Gaussian with zero mean. Therefore, it is maximal along the optical axis and decreases the bigger the angular distance form the optical axis is.

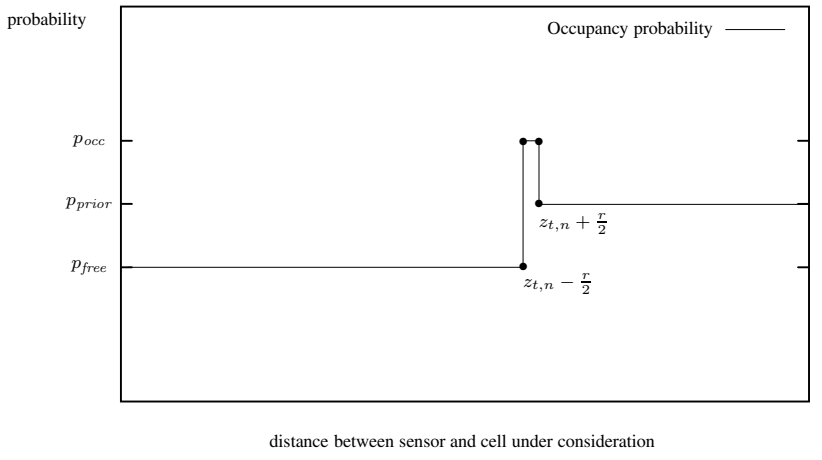


Fig. 2.3. Sensor model for a laser range finder. It depicts the probability that a cell is occupied depending on the distance of that cell from the laser sensor.

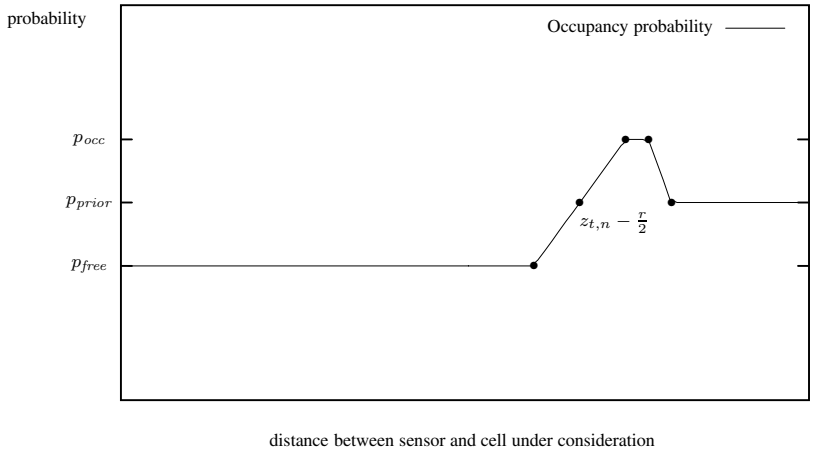


Fig. 2.4. Probability that a cell on the optical axis of the sensor is occupied depending on the distance of that cell from the sensor.

Two examples for a resulting model are depicted in Figure 2.5. It shows two three-dimensional plots of the resulting occupancy probabilities for a measurement of 2 m (left image) and 2.5 m (right image). In this figure, the optical axis of the sensor cone was identical with the x-axis and the sensor was placed in the origin of the coordinate frame. As can be seen, the occupancy probability is high for cells whose distance to x_t is close to $z_{t,n}$. It decreases for cells with shorter distance than $z_{t,n}$ as well as with increasing values of the angular distance.

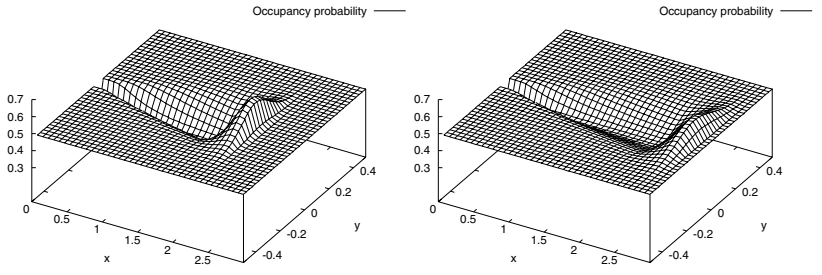


Fig. 2.5. Occupancy probability introduced by a single ultrasound measurement of $z_{t,n} = 2.0m$ (left image) and $z_{t,n} = 2.5m$ (right image).

Figure 2.6 depicts the mapping process for a sequence of observations recorded with an iRobot B21r robot. The first row shows a map was built from a sequence of previous ultrasound scans. Afterwards the robot perceived a series of 18 ultrasound scans each consisting of 24 measurements. The occupancy probabilities for these 18 scans are depicted in the rows from 2 to 7. The occupancy probability grid obtained by integrating the individual observations into the map is shown in the last row of this figure. As can be seen, the belief converges to a representation of the corridor structure in which the scans where recorded.

2.2.4 Reflection Probability Mapping

Beside occupancy probability grids, there exist alternative realization of grid maps. A frequently used model is the so-called reflection probability map or counting model. In contrast to occupancy grid maps, they store for each cell a reflection probability value. This value provides the probability that a measurement covering the cell is reflected. Note that the occupancy model and the counting model are similar but not identical.

In this model, we are interested in computing the most likely reflection probability map m^* given the observations and poses of the robot.

$$m^* = \operatorname*{argmax}_m p(m \mid x_{1:t}, z_{1:t}) \tag{2.41}$$

By series of mathematical transformations (see [16] for the details), one can derive that the most likely map m^* is the map for which each grid cell c has the value

$$p(c \mid x_{1:t}, z_{1:t}) = \frac{\#hits(c, x_{1:t}, z_{1:t})}{\#hits(c, x_{1:t}, z_{1:t}) + \#misses(c, x_{1:t}, z_{1:t})}, \tag{2.42}$$

where $\#misses(c, x_{1:t}, z_{1:t})$ is the number of times a beam $z_{t,n}$ taken from x_t passed through the grid cell c and $\#hits(c, x_{1:t}, z_{1:t})$ is the number of times

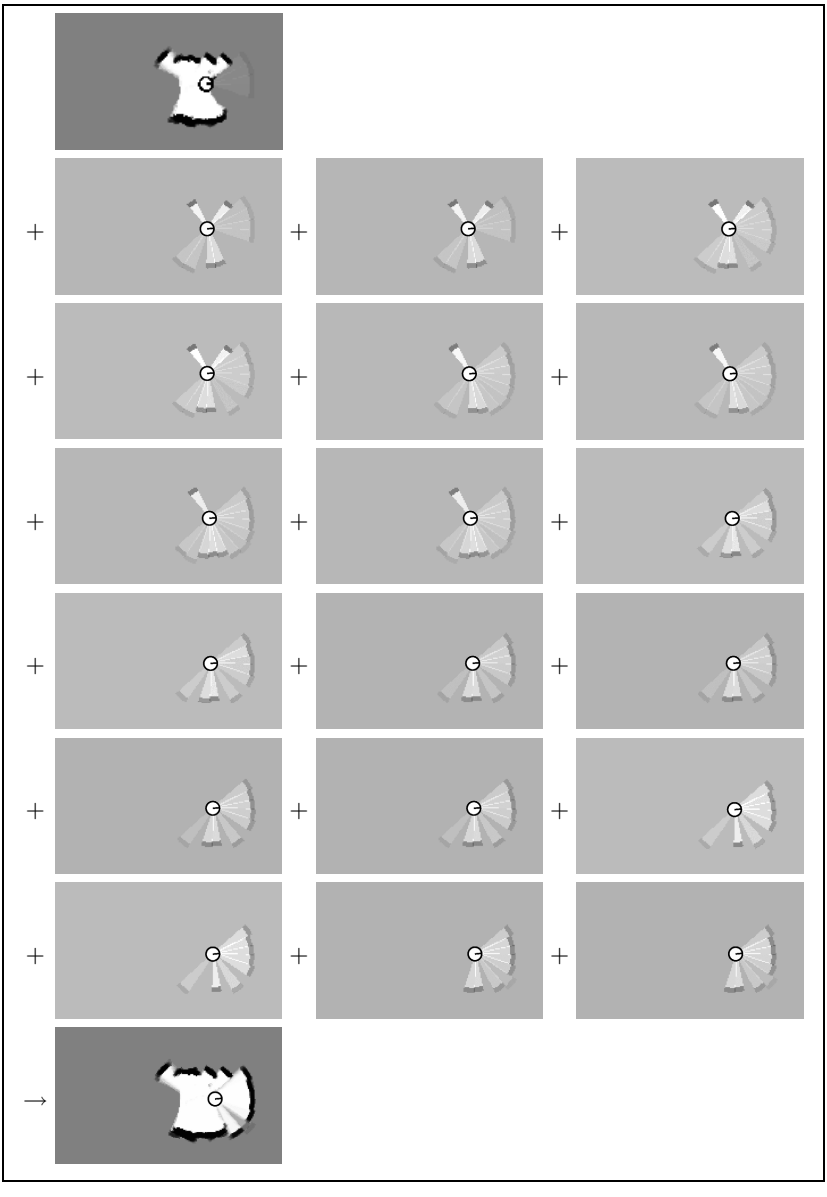


Fig. 2.6. Incremental mapping in a corridor environment. The upper left image shows the initial map and the lower one contains the resulting map. The maps in between are the local maps built from the individual ultrasound scans perceived by the robot.

a beam ended in that cell. Since the value of each cell can be determined by counting, this technique is also called counting model.

The differences between occupancy probability and reflection probability maps is that the occupancy probability typically converges to 0 or 1 for each cell which is frequently observed. In contrast to that, reflection probability values converge to values *between* 0 and 1. Values significantly different from 0 or 1 often occur when mapping objects much smaller than the grid discretization or, for example, for glass panes which are repeatedly observed with a laser range finder.

3

Decision-Theoretic Exploration Using Coverage Maps

3.1 Introduction

There exist several applications in which the exploration task is an integral part of the robotic mission. The complete and efficient coverage of terrain is one of the elementary problems in planetary exploration [6], reconnaissance [63], rescue [110, 149], mowing [67], or cleaning [68, 38, 137].

Throughout this chapter, we focus on the problem of how to efficiently explore an environment with a single mobile robot. We describe a decision-theoretic approach to exploration of unknown terrain with noisy sensors. The goal is to come up with an accurate model of the environment without steering the robot manually. Our approach seeks to minimize the uncertainty in the map over time. Therefore, the next viewpoint of the robot is chosen in a way that its action provides the highest expected uncertainty reduction. In the first part of this book, we assume that the movement of the vehicle is not affected by noise. Later on, we relax this assumption and present a technique to deal with the pose uncertainty of a mobile robot.

In addition to the exploration aspect, we consider the problem of how to model the environment of a mobile robot and how to update the map upon new sensory input. In particular, we introduce coverage maps as a probabilistic way to represent the belief of the robot about the state of the environment. In contrast to occupancy grids [108], in which each cell is considered as either occupied or free, coverage maps represent for each cell of a given discretization a posterior about the percentage this cell is covered by an object. As an example consider the situation depicted in the left images of Figure 3.1 in which a cell is partly covered by an obstacle. Using occupancy grid maps the probability that this cell is occupied converges to 1 if the sensors of the robot repeatedly detect the obstacle (as illustrated in the left image of this figure). Since the object covers only 20% of the area of this cell, a *coverage value* of 0.2 (as shown in the right image of Figure 3.1) would be a better representation of the given situation. Additionally, we present a sensor model that allows us to appropriately update a coverage map upon sensory input affected by noise.

C. Stachniss: Robotic Mapping and Exploration, STAR 55, pp. 23–41.
springerlink.com

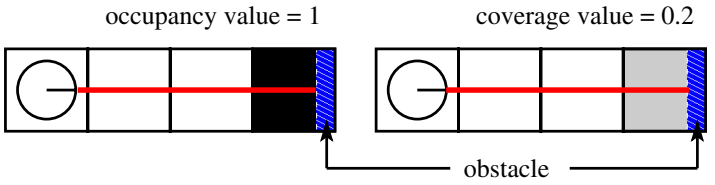


Fig. 3.1. Typical occupancy map obtained in situations in which cells are only partly occupied (left) and a coverage map containing the corresponding coverage values (right). Black represents high occupancy probability respectively coverage value.

This chapter is organized as follows. In the next section, we introduce the idea of coverage maps. In Section 3.3, we present a sensor model that allows us to update a given coverage map upon sensory input. In Section 3.4, we describe a decision-theoretic approach to exploration based on coverage maps. After this, the experiments illustrate the various properties of our approach. We present accurate maps learned by a real robot and discuss the advantages of our technique over existing approaches. Finally, we discuss related work in Section 3.7.

3.2 Definition of Coverage Maps

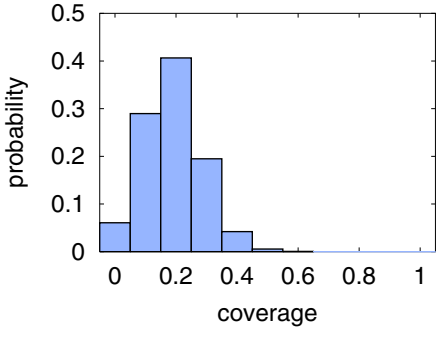


Fig. 3.2. The coverage posterior for the cell containing the obstacle in Figure 3.1.

As already mentioned above, occupancy grids rest on the assumption that the environment has binary structure, i.e., that each grid cell is either occupied or free. This assumption, however, is not always justified. For example, if the environment contains a wall that is not parallel to the x- or y-axis of the grid

there must be grid cells which are only partly covered. In occupancy grids, the probability that such cells are occupied will inevitably converge to 1 (see Figure 3.1). Coverage maps overcome this limitation by storing for each cell a posterior about its coverage. Coverage values range from 0 to 1. A coverage of 1 means that the cell is fully occupied and an empty cell has a coverage of 0. Since the robot usually does not know the true coverage of a grid cell c it maintains a probabilistic belief $p(c)$ about the coverage of c. In principle, there are different ways of representing $p(c)$. They range from parametric distributions such as (mixtures of) Gaussians or non-parametric variants such as histograms. Throughout this work, we assume that each coverage posterior is modeled by a histogram over possible coverage values. More precisely, we store a histogram for each grid cell, where each bin contains the probability that the corresponding grid cell has the particular coverage.

A coverage map cell is typically initialized using a uniform distribution in order to represent the maximum uncertainty about the actual state of the cell. In contrast to this, Figure 3.2 shows a typical coverage posterior we frequently obtain for partly occupied cells. The depicted posterior was generated based on observations perceived in a simulated environment like the one shown in Figure 3.1.

So far, we only explained the idea of coverage maps but left open how to actually determine the posterior based on observations. In the next section, we describe how we can update coverage maps based on sensory input.

3.3 Updating Coverage Maps Upon Sensory Input

To update a coverage map whenever sensor data arrives, we apply a Bayesian update scheme. Throughout this chapter, we assume that our sensor provides distance information. Thus, we need a formalism to convert the distance information to coverage values. What we need to determine is the coverage map m that has the highest likelihood under all distance measurements $z_{1:t} = z_1, \ldots, z_t$. If we use Bayes' rule and then assume that consecutive measurements are independent given that we know the map m, we obtain

$$p(m \mid x_{1:t}, z_{1:t}) \overset{\text{Bayes' rule}}{=} \frac{p(z_{1:t} \mid m, x_{1:t}) p(m \mid x_{1:t})}{p(z_{1:t} \mid x_{1:t})} \tag{3.1}$$

$$= \frac{p(m \mid x_{1:t})}{p(z_{1:t} \mid x_{1:t})} p(z_{1:t} \mid m, x_{1:t}) \tag{3.2}$$

$$\overset{\text{independence}}{=} \frac{p(m \mid x_{1:t})}{\prod_{t'=1}^{t} p(z_{t'} \mid x_{t'})} \prod_{t'=1}^{t} p(z_{t'} \mid m, x_{t'}). \tag{3.3}$$

Next we need to know how to determine the likelihood $p(z_t \mid m, x_t)$ of measuring z_t given the map m and the pose x_t of the vehicle. Again, we apply Bayes' rule and obtain

$$p(m \mid x_{1:t}, z_{1:t}) \overset{\text{Bayes' rule}}{=} \frac{p(m \mid x_{1:t})}{\prod_{t'=1}^{t} p(z_{t'} \mid x_{t'})} \prod_{t'=1}^{t} \frac{p(m \mid x_{t'}, z_{t'}) p(z_{t'} \mid x_{t'})}{p(m \mid x_{t'})} \tag{3.4}$$

$$= \frac{p(m \mid x_{1:t})}{\prod_{t'=1}^{t} p(z_{t'} \mid x_{t'})} \frac{\prod_{t'=1}^{t} p(z_{t'} \mid x_{t'})}{\prod_{t'=1}^{t} p(m \mid x_{t'})} \prod_{t'=1}^{t} p(m \mid x_{t'}, z_{t'}) \tag{3.5}$$

$$= \frac{p(m \mid x_{1:t})}{\prod_{t'=1}^{t} p(m \mid x_{t'})} \prod_{t'=1}^{t} p(m \mid x_{t'}, z_{t'}) \tag{3.6}$$

$$\overset{m \text{ ind. of } x}{\simeq} \underbrace{\frac{1}{p(m)^{t-1}}}_{\eta'} \prod_{t'=1}^{t} p(m \mid x_{t'}, z_{t'}) \tag{3.7}$$

$$= \eta' \prod_{t'=1}^{t} p(m \mid x_{t'}, z_{t'}). \tag{3.8}$$

Equation (3.7) is obtained from (3.6) by assuming that m is independent of x_t given we have no observations. The variable η' represents a normalization constants ensuring that the left-hand side sums up to one over all m. We assume that the individual cells of a coverage map are independent. This is not true in general, but is frequently used in the context of grid maps. We would like to refer to a work by Thrun [147] on how to better deal with the dependency between cells. Finally, we obtain

$$p(m \mid x_{1:t}, z_{1:t}) = \eta' \prod_{t'=1}^{t} \prod_{c \in m} p(c \mid x_{t'}, z_{t'}) \tag{3.9}$$

$$= \eta' \prod_{c \in m} \prod_{t'=1}^{t} p(c \mid x_{t'}, z_{t'}). \tag{3.10}$$

Thus, to update a map given a measurement z_t we simply have to multiply the current belief about the coverage of each cell c by the belief about the coverage resulting from z_t. The maximum likelihood coverage map is obtained by choosing the mode of the coverage histogram for each cell c.

It remains to describe how we actually compute $p(c \mid x_t, z_t)$, i.e. how we determine the distribution about the potential coverage values of a cell c with distance $dist(c, x_t)$ to the sensor given a measurement z_t. In our current system, we use a mixture of a Gaussian $\mathcal{N}(\mu, \sigma)$ and a uniform distribution γ to describe the probability distribution $p(c \mid x_t, z_t)$ about the coverage of c

$$p(c \mid x_t, z_t) = \gamma(dist(c, x_t), z_t) + \xi(z_t)\mathcal{N}(\mu(dist(c, x_t) - z_t), \sigma(dist(c, x_t), z_t)), \tag{3.11}$$

where $\xi(z_t)$ is an indicator variable about the validity of the observation z_t. In case z_t is a maximum range reading, $\xi(z_t)$ equals zero otherwise it is one. $dist(c, x_t)$ is the euclidian distance between the center of the cell c and the position of the robot (the sensor) at time t.

The value of the uniform distribution $\gamma(dist(c, x_t), z_t)$ increases with $dist(c, x_t)$ and the measured distance z_t. This reflects a typical property of proximity sensors like sonars, because the accuracy of a measurement decreases with the distance to the obstacle. The mean $\mu(x)$ of the Gaussian is computed in the following way:

$$\mu(x) \;=\; \begin{cases} 0, & x < -\frac{r}{2} \\ \frac{1}{2} + \frac{x}{r}, & |x| \leq \frac{r}{2} \\ 1, & x > \frac{r}{2} \end{cases} \tag{3.12}$$

Here r is the resolution of the grid discretization. We distinguish three situations, depending on whether the measurement z_t ends in c or not. Suppose that the measurement does not end in c and the distance $dist(c, x_t)$ is shorter than z_t. In this case, we have $dist(c, x_t) - z_t < -\frac{r}{2}$. In such a situation, the mean of the Gaussian is zero. In this way, we assume that a cell which is covered by a measurement that does not end within this cell is most likely empty. The second line of (3.12) represents the situation in which z ends within c. In this case, the mean is inverse proportional to the area the cell is covered by z_t. Finally, cells lying a small distance behind a cell, in which the measurement ends, are most likely completely occupied so that the mean is 1. This value which is set to 20 cm in our implementation models the thickness of the walls and objects in the environment.

The value of the standard deviation $\sigma(dist(c, x_t), z_t)$ of the Gaussian is also a function that is monotonously increasing in $dist(c, x_t)$ and z_t except when $|dist(c, x_t) - z_t| < \frac{r}{2}$. In this interval, $\sigma(dist(c, x_t), z_t)$ has a constant value that exceeds all values outside of this interval.

To obtain the optimal parameters for the various functions in our sensor model shown in (3.11), we apply the maximum likelihood principle. We used data sets recorded with a B21r robot in our department building using sonar and laser observations. We then compared the resulting maps build with the sonar sensors to the ground truth map obtained by applying a highly accurate scan-alignment procedure [61] on the laser range information. We can easily compute the exact coverage of each cell of a given discretization by straightforward geometric operations. We evaluate a particular set of parameters by computing the likelihood of the ground truth map given the corresponding coverage map and by applying a local search techniques to determine a parameter setting that maximizes the likelihood of the ground truth map.

Figure 3.3 depicts a fraction of the resulting sensor model $p(c \mid z_t, x_t)$ for the ultrasound sensors. As the plot illustrates, for a measured distance of 1 m, cells close to the robot are with high likelihood unoccupied. Cells close the measured distance are covered with a high likelihood.

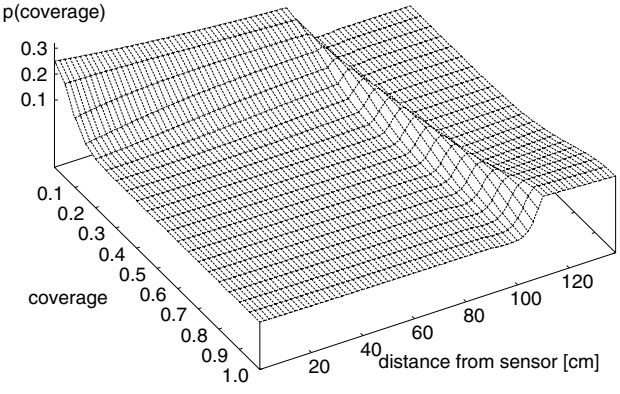


Fig. 3.3. This picture shows our sensor model $p(c \mid x_t, z_t)$ for a proximity measurement (here for a sonar reading with a measured distance $z_t = 1\text{m}$).

The maximum likelihood coverage map obtained with this model is shown in the top image of Figure 3.4. The size of the environment depicted in this figure is 17 m by 2.6 m. The lower image of this figure shows the ground truth map. As can be seen from the figure, the similarity between the learned map and the ground truth is quite high.

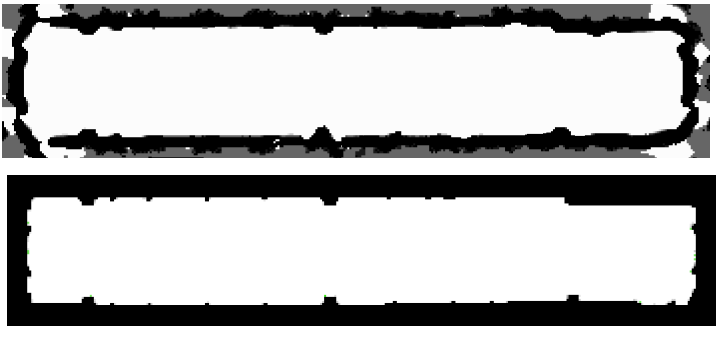


Fig. 3.4. The top image depicts a coverage map learned from ultrasound data using our sensor model and map update technique. In this experiment, most of the obstacles fit into the grid discretization and therefore only a few cells show partly occupied cells. The lower image illustrates the corresponding ground truth map learned from laser range data.

3.4 Decision-Theoretic Exploration with Coverage Maps

One of the key problems during exploration is to choose appropriate viewpoints. In general, there are different aspects that are relevant. On the one hand, the uncertainty of the robot about the current state of the map should be as small as possible and on the other hand, the number of measurements to be incorporated to achieve this uncertainty reduction as well as the traveled distance should be minimized.

Coverage maps are well-suited to support a decision-theoretic approach to exploration. To determine the uncertainty in the state of a particular cell, we consider the entropy of the posterior for that cell. Entropy is a general measure for the uncertainty of a belief and is defined as

$$H(p(x)) \;=\; -\int_x p(x)\log p(x)\,dx. \tag{3.13}$$

In case a histogram is used to represent $p(x)$, the integral turns into a sum over the bins of the histogram. H is maximal in case of a uniform distribution. The minimal value of zero (in the case of a discrete posterior) is obtained if the system is absolutely certain about the state of the corresponding cell. To minimize the uncertainty in the current map, all we need to do is to reduce the entropy of the individual histograms in the coverage map since the cells are assumed to be independent.

The entropy also allows us to define when the exploration task has been completed. We regard the exploration task as completed as soon as the robot reaches a defined level of certainty about the map. This is a more appropriate choice than regarding an environment as explored as soon as all (reachable) cells have been covered with the robots' sensors. Suppose the environment is of limited size. Then, we define the goal of the exploration process for a coverage map m as $H(p(c)) < \epsilon$ for all cells $c \in m$ that can be reached by the robot. The value of ϵ describes the desired level of certainty about the state of all cells. Additionally, the system has to detect a situation in which the robot is unable to reduce the entropy of a cell below ϵ to ensure the termination of the exploration task. In our system, this is achieved by monitoring the change of entropy. If this change is below a threshold value for consecutive measurements, the cell is regarded as explored.

In this section, we specified the termination criterion for our exploration task based on the entropy in the map model. In the following, we explain how we actually guide the robot through the environment.

3.4.1 Choosing the Closest Target Location (CL)

A popular exploration strategy is to drive to the closest location at which the robot can gather information about a cell that has not been sufficiently explored. This strategy has been shown to provide short trajectories for single

robot exploration tasks [77]. As mentioned above, our approach uses the entropy to measure the uncertainty about a grid cell. A cell is regarded as been sufficiently observed if the entropy of the coverage belief does not exceed ϵ or if it does not change any longer. The first strategy, called CL, does not take into account how much information will be obtained at a particular viewpoint. It rather seeks to minimize the distance to the next point by selecting

$$c^* = \operatorname*{argmin}_{c \in L(m)} dist_m(x, c). \tag{3.14}$$

Here $L(m)$ is the set of reachable cells which have a grid cell with high entropy in its visibility range. $dist_m(x, c)$ is the length of the shortest path between the current pose x of the robot and the location c given the current map m of the environment.

3.4.2 Exploration Using the Information Gain (IG)

The second strategy, called IG, is solely governed by the information gain that can be obtained about the environment at a specific viewpoint. The information gain is used to take into account the accuracy of the information provided by the sensor. We compute the expected information gain which is the expected change of entropy given that the robot obtains a measurement at a certain location in the map.

For a given cell c and measurement z taken from x, the information gain is defined as

$$I(c, x, z) = H(p(c)) - H(p(c \mid x, z)). \tag{3.15}$$

Here $p(c)$ is the coverage histogram of cell c and $p(c \mid x, z)$ the same histogram after integrating the measurement z taken from the pose x according to our sensor model. The information gain of a measurement is then computed as the sum of the information gains for all cells covered by that measurement. Since we do not know which measurement we will receive if the robot measures at a certain position x, we have to integrate over all possible measurements to compute the expected information gain for that viewpoint

$$E[I(x)] = \int_z p(z \mid m, x) \sum_{c \in Cov(x,z)} I(c, x, z)\, dz. \tag{3.16}$$

Here $Cov(x, z)$ is the set of cells covered by measurement z taken from location x. In order to estimate $Cov(x, z)$, we apply a ray-casting technique based on the current maximum likelihood map. Considering only the maximum likelihood map to compute the observation $p(z \mid m, x)$ is an approximation but it allows us to compute this quantity in an efficient way. In our approach, we take into account a discretized set of proximity measurements. In this way, the integral turns into a sum

$$E[I(x)] \simeq \sum_{z} p(z \mid m, x) \sum_{c \in Cov(x,z)} I(c, x, z). \tag{3.17}$$

Since the complexity of (3.17) depends exponentially on the number of dimensions of the measurement, we consider all dimensions independently. Otherwise the computation would be intractable. For example, for our B21r robot Albert (see Figure 3.5) equipped with 24 ultrasound sensors, we compute the average information gain over all 24 sensors independently.

To evaluate a potential viewpoint, we generate a set of potential proximity observations. This set is given by all possible distances the sensor can return up to a given resolution. In our current implementation, we simulate all proximity observations between 20 cm and 2.5 m with a resolution of 2 cm. We then determine the likelihood for each observation and its effect on the entropy of the map. This simulation process is computationally intensive, but it provides the expected reduction of entropy for each grid cell in the map. This information is required when seeking for exploration strategies that minimize the uncertainty in the map model. In our approach, we consider each grid cell c as a potential next viewpoint and select the one which provides the highest expected entropy reduction

$$c^* = \operatorname*{argmax}_{c \in L(m)} E[I(c)]. \tag{3.18}$$

In extensive experiments, we figured out that an approach that purely relies on the information gained at particular viewpoints usually minimizes the number of measurements needed to learn a map. However, it has the major disadvantage that it does not take into account the overall path length of the resulting trajectory.

Fig. 3.5. The left image depicts the B21r robot Albert used to carry out the experiments. Albert is equipped with ring of 24 ultrasound sensors. The other images show photographs taken within the corridor of our office environment.

3.4.3 Using IG in a Local Window (IG_WIN)

To overcome the disadvantage that the strategy IG does not take into account the overall path length of the resulting trajectory, we defined the strategy IG_WIN. This technique restricts the search for potential viewpoints to a local window. This window defines an area in the environment the robot has to explore completely before focusing on a different area. The next viewpoint can be determined by

$$c^* = \underset{c \in L_{win}(m)}{\operatorname{argmax}} E[I(c)]. \tag{3.19}$$

Here, $L_{win}(m)$ refers to the potential goal locations which are located in the local window. Once the window has been explored, there is no need for the robot to return to this area again. As we point out in the experiments, the distance to be traveled can be significantly reduced using this strategy.

3.4.4 Combination of IG and CL (IG_CL)

The final strategy discussed in this chapter tries to combine the properties of the strategies CL and IG. The goal is to find the best trade-off between the expected information gain $E[I(c)]$ of possible viewpoints $c \in L(m)$ and the costs $dist_m(x, c)$ of reaching them. This is achieved by combining (3.14) and (3.18)

$$c^* = \underset{c \in L(m)}{\operatorname{argmax}} \left[\alpha \frac{E[I(c)]}{\max_{c' \in L(m)} E[I(c')]} - (1-\alpha) \frac{dist_m(x, c)}{\max_{c' \in L(m)} dist_m(x, c')} \right]. \tag{3.20}$$

The normalization in (3.20) is performed to account for the fact that it is unclear how to subtract a distance from an information gain. Therefore, we subtract the relative cost from the relative information gain. As we show in the experiments of this chapter, this leads to a well balanced exploration behavior.

Equation (3.20) combines the advantages of the strategies IG and CL. It reduces the distance to be traveled by the robot and the number of measurements necessary to achieve the desired level of certainty. By adapting the weight α the user can easily influence the behavior of the robot and optimize its performance for a special task. A value close to zero results in the strategy CL, whereas a value close to 1, in contrast leads to a strategy that only considers the information gain.

3.5 Exploration Using Occupancy Grid Maps

In general, the decision-theoretic exploration technique presented in this chapter is not restricted to coverage maps. As long as the underlying map representation allows the robot to compute the uncertainty of a local area

like a grid cell. One can also compute the entropy of an occupancy grid cell c as

$$H(p(c)) = -p(c)\log p(c) - (1-p(c))\log(1-p(c)). \tag{3.21}$$

An occupancy grid map can also be seen as a coverage map using coverage histograms with two bins. Since each cell is represented by a binary variable, the amount of information stored per cell is small compared to coverage maps. This is due to the fact that coverage maps allow to model partly occupied cells and use a full histogram instead of a binary variable. We therefore chose this representation for our approach presented in this chapter.

3.6 Experimental Results

Our techniques described above have been implemented and evaluated using a real robot as well as in simulation runs. In our experiments, the use of coverage maps with our decision-theoretic viewpoint selection strategy has shown an advantage over standard exploration strategies often used in combination with occupancy grids. We figured out that whenever a robot has to actively control its motions in order to acquire all information necessary to generate an accurate map, the uncertainty-driven approach is of utmost importance.

The experiments described in this section are designed to illustrate that coverage maps in combination with our sensor model can be used to learn high quality maps even if noisy sensor are used. We demonstrate that they allow us a decision-theoretic control of the robot during exploration. We furthermore compare our method to the scan counting technique for exploration. Scan counting stores for each cell the number of times it has been observed and in this way decides if a cell has been sufficiently explored. As we show, the use of scan counting leads to either longer trajectories than our approach or to less accurate maps.

Please note that the simulation of potential observation sequences is computationally expensive. In our experiments, the robot had to stop after it reached its viewpoint in order to evaluate future actions. Therefore, we do not consider measures like average speed of the robots in this chapter.

3.6.1 Mapping with Noisy Sensors

The first experiment is designed to illustrate that we obtain highly accurate coverage maps using our sensor model. In this real world experiment, the mobile robot Albert explored parts of our office environment using our decision-theoretic viewpoint selection technique. Albert traveled along the corridor and entered three rooms. The middle and the right image of Figure 3.5 show pictures of this environment. As can be seen, there are lots of glass panes which are hard to map with ultrasounds. The resulting coverage map is shown in Figure 3.6. We would like to emphasize that even smaller

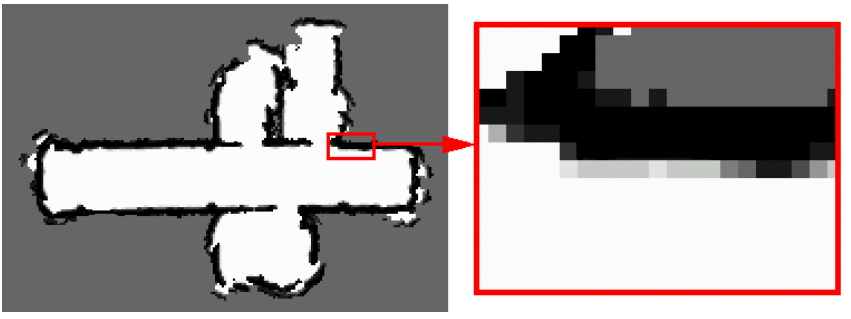


Fig. 3.6. This figure depicts a coverage map learned by Albert using its ultrasound sensors in the environment depicted in Figure 3.5. In the magnified view, partly occupied cells are visible (grayish cells). The size of each grid cell within this map is 10 cm by 10 cm.

details such as the narrow pillars at the walls are visible in the resulting map. The left image of this figure shows a magnified view on partly occupied cells.

Figure 3.7 depicts snapshots of an exploration experiment in the same environment using the exploration strategy IG_CL. The individual images depicts the map constructed so far as well as the entropy values of the individual map cells. The darker the value, the higher the entropy. The red points represent the set $L(m)$, which is the set of potential target locations the robots considers to approach. As can be seen, the robot explores the environment until the set $L(m)$ is empty, which means that all reachable cells and their direct surroundings have a low entropy value.

Another example for a coverage map build from real sonar data is depicted in the top image of Figure 3.8. The sonar data (see lower image of the same figure) has been recorded while the robot was controlled manually using a joystick. Since the robot was not performing an exploration task it did not enter any of the rooms close to the corridor.

3.6.2 Comparing the Viewpoint Selection Strategies

Robots performing 2D exploration tasks with sonars or laser range scanners typically integrate every sensor measurement because the amount of data is reasonably small and easy to integrate. In this section, we also consider the situation that analyzing a measurement produces high costs. This might by the case if, for example, the distance information needs to be extracted from stereo images. In such a situation, the number of measurements needed for the exploration task is a value of interest.

As mentioned above, one of the major advantages of our decision-theoretic exploration technique is that they allow us to integrate the uncertainty in the map model into the selection process of the next viewpoint. The experiments

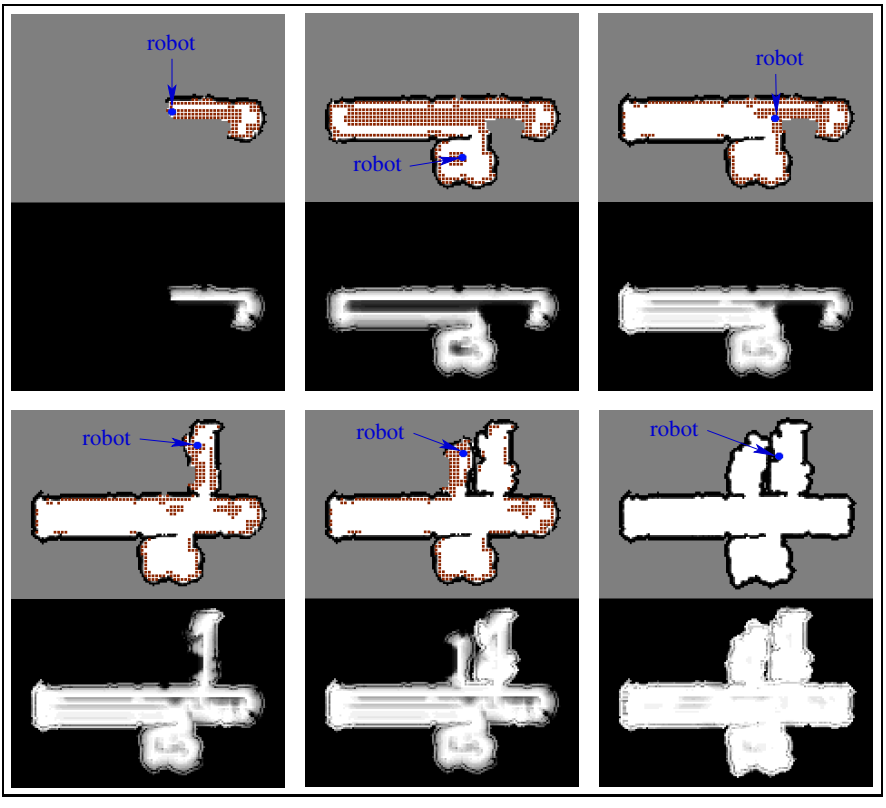


Fig. 3.7. The images show snapshots during an exploration experiment. The upper parts of each image shows the current map and the lower one corresponding entropy. Darker values in the lower image indicate higher entropy. The poses of the robot is indicated by the blue circle in the upper parts. The red points indicate potential viewpoints.

in this section are designed to compare the performance of the different strategies using the traveled distance and the number of required observations as measure. To carry out the experiments, we varied the size of the local window when using IG_WIN and the weight α in the evaluation function shown in (3.20). In Figure 3.9, the numbers behind IG_CL show the value of the weight α and the numbers behind IG_WIN indicate the radius of a circle which defines the window. The results have been obtained using 20 runs per strategy. Please note that further experiments carried out in alternative environments showed similar results. The maximum allowed entropy during all experiments described in this section was set to 0.6 (using 11 histogram bins).

The left graph in Figure 3.9 shows the average number of measurements necessary to complete the exploration task for each strategy. As can be seen

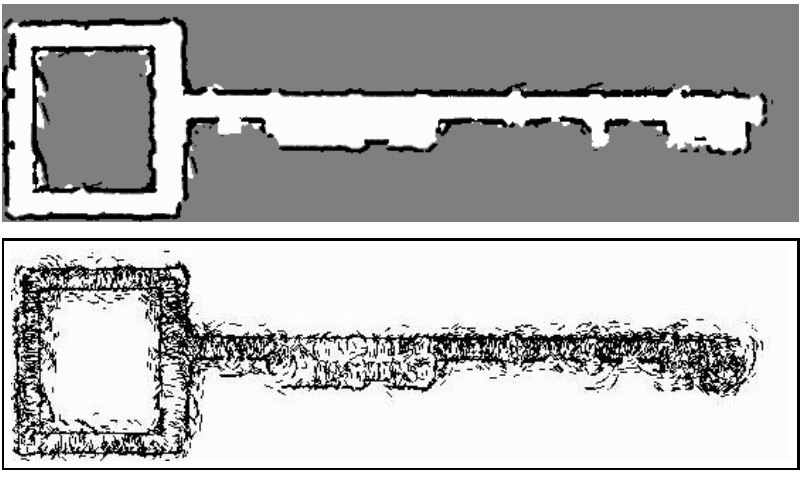


Fig. 3.8. The top image depicts a coverage map build from recorded sonar data at the University of Washington. The lower image shows 2% of all sonar scans used to build the map above and illustrates the high noise in the measurement data.

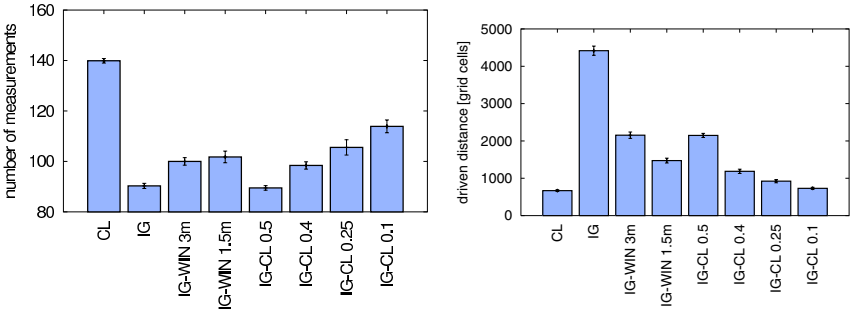


Fig. 3.9. The left graph shows the average number of measurements needed by different strategies. The right one depicts the average path length of the exploration for each strategy. The value behind IG_WIN shows the size of the local window and behind IG_CL the value of the parameter α. The errorbars show the 5% significance interval.

from the figure, the strategy IG needs the minimum number of measurements. The strategy IG_CL with $\alpha = 0.5$ needs approximately the same number of measurements as IG. The strategy CL requires the maximum number of measurements compared to all other strategies considered here. The reason is that it only seeks to minimize the path length without considering the information gained at particular locations.

In our experiments, we found that a nearest neighbor viewpoint selection strategy like CL outperforms an approach considering the information gain if

the robot is allowed to integrate measurements while it is moving (assuming that the acquisition and integration of measurements can be done fast). This can be seen in right image of Figure 3.9 which plots the average path length driven by the robot during the exploration task for all different strategies. With respect to the path length the strategy CL shows the best behavior as the resulting trajectories are shorter than those of all other techniques. In contrast to that, the IG strategy ignores the distance to be driven and therefore produces an extremely long path which results in the worst behavior of all strategies. The IG_CL strategy with $\alpha = 0.4$ appears to yield a good trade-off between the number of measurements and the overall path length. According to our experiments, it slightly outperforms the IG_WIN strategy.

3.6.3 Advantage over Scan Counting

The next experiment is designed to illustrate that an approach which considers the uncertainty in the belief of a cell to select viewpoints yields more accurate maps than techniques relying on scan counting approaches. Scan counting techniques store for each cell the number of times this cell has been intercepted by a measurement. Several exploration techniques [18, 36, 164] use scan counting and assume that a place is explored if it has been scanned once. This can be problematic especially when the underlying sensors are noisy. Figure 3.10 shows a typical occupancy grid map of the corridor at our laboratory obtained from real sonar data and using this approach. Although this map reveals the structure of the environment, it lacks several details that are contained in the corresponding coverage map obtained with our uncertainty-driven exploration technique. Since the exploration process stops as soon as all reachable locations were intercepted by a measurement at least once, typically many cells of the resulting map have a high uncertainty. Especially, if noisy sensors are used the robot has to scan cells multiple times. This leads to an extension of scan counting in which one assumes that each cell has to be covered n times and not only once. In practice, it is unclear how to choose the value for n. A candidate value could be the maximum number of measurements necessary for obtaining a coverage map that fulfills the entropy threshold criterion.

technique	**path length**	$\lvert\{c \mid H(h(c)) > \epsilon\}\rvert$
coverage maps	89.1m	0%
counting (n=1)	26.6m	21%
counting (n=50)	90.6m	1.5%

Table 3.1. This table shows the path length and number of cells with high entropy for different exploration strategies. The values are obtained by a series of real world explorations runs performed in our department.

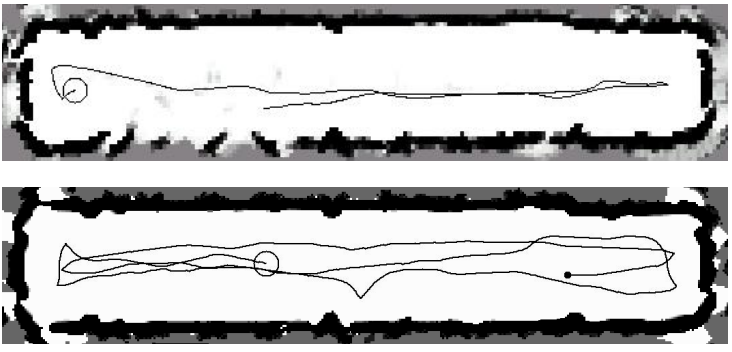


Fig. 3.10. The top image depicts an occupancy grid map obtained with scan counting ($n = 1$). The bottom image shows the corresponding coverage map.

To analyze the quality of occupancy grid maps obtained for different values of n, we performed several experiments. The results for $n = 1$ and $n = 50$ are summarized in Table 3.1 (in practical experiments we found that $n = 50$ yields coverage maps that typically fulfill the entropy criterion for the majority of grid cells using ultrasound sensors). The right column of this table contains the percentage of cells in m for which the entropy exceeds the given threshold. As can be seen from the figure, more than 20% of the cells do not fulfill the entropy criterion if $n = 1$. In the case of $n = 50$, still 1.5% of the cells are above this threshold. In contrast to this, our approach considers the uncertainty in the coverage of the individual cells so that the resulting maps are more accurate. As this experiment demonstrates, even extended scan counting does not guarantee that in the end every cell is explored sufficiently. Typically, some cells will be measured too often, others not often enough.

To analyze the relationship between the overall distance traveled and the percentage of sufficiently explored cells, we performed a series of 50 simulation experiments. In these experiments, we forced the robot to reach a scan count of n where n varied between 1 and 130. We counted the number of cells that were sufficiently explored given the entropy criterion for coverage maps and plotted it against the length of the overall path. The resulting graph is shown in Figure 3.11. The cross on the right side indicates the path length obtained when using our exploration strategy IG_CL for coverage maps. If one requires that 85% or more of the cells c of the map should satisfy $H(h(c)) < \epsilon$ (here $\epsilon = 0.6$), a decision-theoretic exploration strategy yields shorter trajectories than extended scan counting.

3.7 Related Work

Exploration is the task of guiding a vehicle during mapping such that it covers the environment with its sensors. In addition to the mapping task, efficient

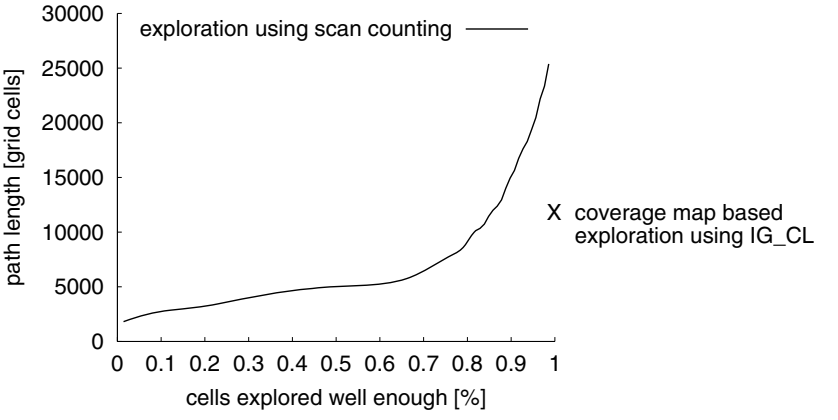


Fig. 3.11. This image shows the resulting path length for scan counting obtained using a simulator. The cross shows the average path length when using coverage maps.

exploration strategies are also relevant for surface inspection, mine sweeping, or surveillance [21, 96]. In the past, several strategies for exploration have been developed. A popular technique is to extract frontiers between known and unknown areas [18, 78, 163, 164] and to visit the nearest unexplored place. Koenig and Tovey [77] have shown that such a strategy which guides the vehicle to the closest unexplored point keeps the traveled distance reasonably small compared to the shortest trajectory which covers the whole environment. Most approaches applying such a technique solely distinguish between scanned and unscanned areas and do not take into account the actual information gathered at each viewpoint. To overcome this limitation, González-Baños and Latombe [52] determine the amount of unseen area that might be visible to the robot from possible viewpoints. To incorporate the uncertainty of the robot about the state of the environment, Moorehead *et al.* [105] as well as Bourgault *et al.* [15] use occupancy grids and compute the entropy of each cell in the grid to determine the utility of scanning from a certain location. Whaite and Ferrie [161] have presented an approach that also uses the entropy to measure the uncertainty in the geometric structure of objects that are scanned with a laser range sensor. In contrast to our work, they use a parametric representation of the objects to be scanned.

Grabowski *et al.* [54] present an exploration technique based on occupancy grids which is optimized for sonar sensors. In their approach, the robot is forced to observe obstacles from different angles. In this way, they obtain sharper boundaries between obstacles and free space area. To select the next viewpoint, they choose the closest one.

Edlinger and von Puttkamer [36] developed a hierarchical exploration strategy for office environments. Their approach first explores rooms and

then traverses through doorways to explore other parts of the environment. Tailor and Kriegman [144] describe a system for visiting all landmarks in the environment of the robot. Their robot maintains a list of unvisited landmarks that need to be approached and mapped by the robot. Dudek *et al.* [34] propose a strategy for exploring an unknown graph-like environment. Their algorithm does not consider distance metrics and is designed for robots with very limited perceptual capabilities.

Additionally, several researchers focus on the problem of simultaneous localization and mapping during exploration [15, 22, 94], an aspect that we do not address in this chapter. We assume that the relative pose information of the robot is accurate enough to integrate a sequence of measurements correctly into a grid map.

Our representation of the environment can be seen as an extension of occupancy grid maps introduced by Moravec and Elfes [108] (see also Chapter 2). Coverage maps are able to model partly occupied cells and provide tools to reason about the uncertainty of the system about the state of grid cells. Compared to occupancy grids, our approach is a richer representation because it can store more information about cells. As a result, it has the disadvantage of high memory requirements since it stores histograms instead of a single probability values. Furthermore, the update of the cells upon sensory input is computationally more expansive.

Very recently, our exploration framework including coverage maps have been reimplemented by Rocha *et al.* [123] and applied to a stereo camera sensor instead of sonars. In their system, they use the concept of coverage maps to build up a three-dimensional grid instead of a two-dimensional one. A further difference to the work presented here is that they compute a gradient field based on the entropy to generate smoother trajectories for the robot. Their experiments confirmed the results reported here: "*Experimental results obtained with a real robot and stereo-vision successfully validated the proposed framework*" [123].

3.8 Conclusion

In this chapter, we introduced coverage maps as a probabilistic representation scheme for grid-based maps built by mobile robots from sensor data. Coverage maps store in each cell a posterior about the coverage of that cell. In this way, they offer the opportunity to model partly occupied cells. We also developed a sensor model designed to update coverage maps upon sensory input. We then presented a decision-theoretic approach to guide a vehicle during exploration. This technique uses the coverage posterior in the map to reason about the uncertainty of the robot about each location in the environment. It simulates potential observations to be obtained at the different target locations and takes into account their effect on the map model. The goal is to choose the viewpoint that minimized the overall uncertainty in the map model.

Our technique has been implemented and evaluated in extensive simulation runs and in real world experiments. The experiments illustrate that by using coverage maps it is possible to build accurate maps even with noisy sensors. Additionally, they demonstrate that our decision-theoretic exploration approach can be used to control a robot in order to obtain maps not exceeding a given level of uncertainty, which is useful especially if the robot uses noisy sensors such as ultrasounds. Experiments analyzing different exploration strategies indicate that a technique combining the maximum uncertainty reduction and the distance to be traveled yields the best tradeoff between the number of necessary measurements and the length of the resulting paths.

4

Coordinated Multi-Robot Exploration

4.1 Introduction

In the previous chapter, we introduced a framework for mobile robot exploration. The goal of our approach was to select appropriate viewpoints for a *single robot* in order to build a map with low uncertainty. In contrast to that, we consider in this chapter the problem of exploring unknown environments with a *team of cooperating robots*. The use of multiple robots is often suggested to have advantages over single robot systems [20, 35]. First, cooperating robots have the potential to accomplish a task faster than a single robot [58]. By using several robots, redundancy can be explicitely introduced so that such a team can be expected to be more fault-tolerant than a single robot. Another advantage of robot teams arises from merging overlapping sensor information, which can help to compensate for sensor uncertainty. As a result, the map can be expected to be more accurate. Multiple robots have also been shown to localize themselves more efficiently, especially when they have different sensor capabilities [42, 122].

However, when robots operate in teams there is the risk of interference between them [132, 50]. For example, if the robots have the same type of active sensors such as ultrasound sensors, the overall performance can be reduced due to cross-talk. The more robots that are used, the more time each robot may spend on detours in order to avoid collisions with other members of the team. Efficient exploration techniques that seek to minimize the overall time to complete the task should consider techniques to distribute the robots over the environment and to reduce the number of redundantly explored areas.

Most approaches to multi-robot exploration proceed in the following way. First, a set of potential target locations or target areas is determined. Secondly, target locations are assigned to the individual members of the team. The robots then approach those target locations and include their observations obtained along the paths into a map. This process is repeated, until the environment has completely been explored. A stopping criterion can be a threshold on the entropy as applied in the previous chapter or a scan counting

C. Stachniss: Robotic Mapping and Exploration, STAR 55, pp. 43–71.
springerlink.com

technique (see Section 3.6.3) which requires that each cell is covered at least once by the sensor of one robot.

In this chapter, we present an algorithm for coordinating a group of robots that enables them to efficiently explore their environment. The goal is to complete the task as fast as possible. Our technique assigns a utility to each target location and follows a decision-theoretic approach to explicitly coordinate the robots. It does so by maximizing the overall utility and by minimizing the potential for overlap in information gain amongst the various robots. The algorithm simultaneously considers the utility of observing unexplored areas and the cost for reaching these areas. By trading off the utilities and the cost and by reducing the utilities according to the number of robots that are already heading towards this area, coordination is achieved in an elegant way. The basic idea of discounting the utility of target locations that might be visible by a different robot has originally been presented in [106] and has been integrated into two different systems [17, 136].

In practice, one also has to deal with problems like limited communication ranges of the network that limit the ability of the vehicles to exchange data. Naturally, the task of exploring a terrain with limited communication range is harder than without this constraint. If the distance between the robots becomes too large to be bridged by the wireless network or if a temporal network error occurs, robots may explore an area another robot has already explored before, which can lead to a suboptimal behavior. We describe how to use our algorithm with robot teams that provide only a limited communication range.

Our technique has been implemented on teams of heterogeneous robots and has been proven to work effectively in real-world scenarios. Additionally, we have carried out a variety of simulation experiments to explore the properties of our approach and to compare the coordination mechanism to other approaches developed so far. As the experiments demonstrate, our technique significantly reduces the time required to completely cover an unknown environment with a team of robots compared to an approach which lacks our centralized coordination. We also consider other coordination techniques and provide comparisons to our approach. Furthermore, we describe experiments in which we analyze our algorithm in the context of teams of mobile robots with a limited communication range.

The rest of this chapter on multi-robot exploration is organized as follows. In the next section, we present the decision-theoretic approach to coordinated exploration with mobile robots. In Section 4.3, we briefly describe the technique used by our system to acquire and communicate maps of the environment. Section 4.4 presents a series of experiments carried out with real robot systems and in simulation and Section 4.5 provides comparisons to exiting coordination techniques. Finally, we discuss related work in Section 4.6.

4.2 Coordinating a Team of Robots during Exploration

The goal of the exploration process is to cover the whole environment in a minimum period of time. Therefore, it is essential that the robots keep track of which areas of the environment have already been explored. Furthermore, the robots have to construct a global map online in order to plan their paths and to coordinate their actions. We first assume that at every point in time both, the map of the area explored so far and the positions of the robots in this map, can be communicated between the robots. We focus on the question of how to coordinate the robots in order to efficiently cover the environment. We then consider the situation in which the robots have a limited communication range.

In this chapter, we focus on the coordination aspect of the exploration problem. Since we deal with large teams of robots, we are interested in keeping the memory requirements small. We therefore use standard occupancy grids instead of coverage maps to model the environment since they store only a binary random variable for each cell instead of a histogram. In case we had enough memory resources available, coverage maps would have been a better choice. However, the coordination aspect can be regarded as independent from the underlying representation. We furthermore assume throughout this chapter that "exploredness" is a binary concept, since we focus on the coordination aspect. We regard a cell as explored as soon as it has been intercepted by a sensor beam. This concept is also known as scan counting.

To guide the exploration process, we adopt the notation of frontiers which has originally been introduced by Yamauchi [163]. As a frontier cell we denote each already explored cell that is an immediate neighbor of an unknown, unexplored cell. If we direct a robot to such a cell, we can expect that it gains information about the unexplored area when it arrives at its target location. The fact that a map generally contains several unexplored areas raises the problem of how to assign exploration tasks represented by frontier cells to the individual robots. If multiple robots are involved, we want to avoid that several of them move to the same location. To deal with these problems and to determine appropriate target locations for the individual robots, our system uses a decision-theoretic approach. We simultaneously consider the cost of reaching a frontier cell and the utility of that cell. For each robot, the cost of a cell is proportional to the distance between the robot and that cell. The utility of a frontier cell instead depends on the number of robots that are moving to that cell or to a place close to that cell.

In the following sections, we describe how we compute the cost of reaching a frontier cell for the individual robots, how we determine the utility of a frontier cell, and how we choose appropriate assignments of frontier cells to robots.

4.2.1 Cost of Reaching a Target Location

To determine the cost of reaching a frontier cell, we compute the optimal path from the current position of the robot to all frontier cells based on a deterministic variant of the *value iteration*, a popular dynamic programming algorithm [8, 66]. In the following, $c_{x,y}$ corresponds to the x-th cell in the direction of the x-axis and the y-th cell in direction of the y-axis of the two-dimensional occupancy grid map. In our approach, the cost for traversing a grid cell $c_{x,y}$ is proportional to its occupancy value $p(c_{x,y})$. The minimum cost path is computed using the following two steps:

1. **Initialization.** The grid cell that contains the robot location is initialized with 0, all others with ∞:

$$V_{x,y} \longleftarrow \begin{cases} 0, & \text{if } (x,y) \text{ is the position of the robot} \\ \infty, & \text{otherwise} \end{cases}$$

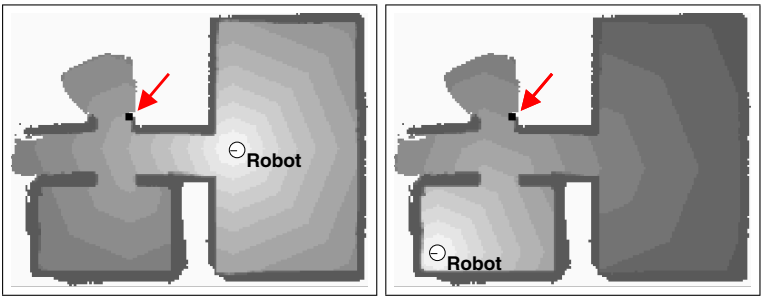
2. **Update loop.** For all grid cells $c_{x,y}$ do:

$$V_{x,y} \longleftarrow \min \Big\{ V_{x+\Delta x, y+\Delta y} + \sqrt{\Delta x^2 + \Delta y^2}\, p(c_{x+\Delta x, y+\Delta y}) \mid \Delta x, \Delta y \in \{-1, 0, 1\} \wedge p(c_{x+\Delta x, y+\Delta y}) \in [0, occ_{max}] \Big\},$$

where occ_{max} is the maximum occupancy probability value of a grid cell the robot is allowed to traverse. This technique updates the value of all grid cells by the value of their best neighbors, plus the cost of moving to this neighbor. Here, cost is equivalent to the probability $p(c_{x,y})$ that a grid cell $c_{x,y}$ is occupied times the distance to the cell. The update rule is repeated until convergence. Then each value $V_{x,y}$ corresponds to the *cumulative cost* of moving from the current position of the robot to $c_{x,y}$. The convergence of the algorithm is guaranteed as long as the cost for traversing a cell is not negative and the environment is bounded. Both criteria are fulfilled in our approach. The resulting cost function V can also be used to efficiently derive the minimum cost path from the current location of the robot to arbitrary goal positions $c_{x,y}$. This is done by steepest descent in V, starting at $c_{x,y}$.

The computation of V is done independently for each robot. This allows us to coordinate also heterogenous teams of robots. For example, a robot traveling faster than its team mates can be modeled by assigning lower travel cost to this vehicle. As a result, this robot will be send to more distant target locations compared to its team mates. Additionally, it is possible to model robots of different size. This can be achieved by expanding the size of the obstacles in the maps of the robots individually.

Figure 4.1 shows the resulting cost functions for two different robot positions. The black rectangle indicates the target point in the unknown area with minimum travel cost. Note that the same target point is chosen in both situations. Accordingly, if the robots are not coordinated during exploration, they would move to the same position which obviously is not optimal.



(a) (b)

Fig. 4.1. Typical cost functions obtained for two different robot positions. The black rectangle indicates the target points in the unknown area with minimum cost.

Our algorithm differs from standard value iteration in that it regards all actions of the robots as deterministic, which seriously speeds up the computation. To incorporate the uncertainty in the motion of the robots into the process and to benefit from the efficiency of the deterministic variant, we smooth the input maps by a convolution with a Gaussian kernel. This has a similar effect as generally observed when using the non-deterministic approach: It introduces a penalty for staying close to obstacles so that the robots generally prefer to move in open spaces.

4.2.2 Computing Utilities of Frontier Cells

Estimating the utility of frontier cells is more difficult. In fact, the actual information that can be gathered by moving to a particular location is hard to predict, since it strongly depends on the structure of the corresponding area. However, if there already is a robot that moves to a particular frontier cell, the utility of that cell can be expected to be lower for other robots. But not only the designated target location has a reduced utility. Since the sensors of a robot typically cover a certain region around a particular frontier cell as soon as the robot arrives there, even the expected utility of frontier cells in the vicinity of the robot's target point is reduced.

In this section, we present a technique that estimates the expected utility of a frontier cell based on the distance and visibility to cells that are assigned to other robots. Suppose that in the beginning each frontier cell f has the utility U_f which is equal for all frontier cells if no additional information about the usefulness of certain positions in the environment is available. Whenever a target point f is selected for a robot, we reduce the utility of all frontier cells f' close to f. This is done according to the probability $p_{vis}(f, f')$ that the robot's sensors will cover f' given the robot moves to f.

One can estimate $p_{vis}(f, f')$ by learning a posterior about the estimated distances to be measured. The longer the average proximity measurements

are in an environment, the more likely the target f' can be observed from f. While the robot moves through the environment, this posterior can be updated and in this way adapt to the spacial structure.

Accordingly, we compute the utility $U(f_n \mid f_1, \ldots, f_{n-1})$ of a frontier cell f_n given that the cells $f_1, \ldots, f_{n-1}$ have already been assigned to the robots $1, \ldots, n-1$ as

$$U(f_n \mid f_1, \ldots, f_{n-1}) = U_{f_n} - \sum_{i=1}^{n-1} p_{vis}(f_n, f_i). \tag{4.1}$$

The more robots move to a location from where f_n is likely to be visible, the lower is the utility of f_n. Note that we also take into account whether there is an obstacle between two frontier cells f and f'. This is achieved by a ray-casting operation on the grid map. If there is an obstacle between f and f', we set $p_{vis}(f, f')$ to zero.

The obtained function for p_{vis} typically has the shape of a decreasing, more or less linear function. The gradient of that function was quite similar for different environments. We observed only small differences in the resulting exploration time when applying the learned posterior p_{vis} in a different environment. We therefore use a linear function to represent p_{vis} and use the same parameters for all environments according to

$$p_{vis}(f, f') = \begin{cases} 1.0 - \frac{||f - f'||}{max\_range}, & \text{if } ||f - f'|| < max\_range \\ 0, & \text{otherwise,} \end{cases} \tag{4.2}$$

where *max_range* is the maximum range of the used proximity sensor.

4.2.3 Target Point Selection

To compute appropriate target points for the individual robots, we need to consider for each robot the cost of moving to a location and the utility of that location. In particular, for each robot i we trade off the cost V_f^i to move to the location f and the utility U_f of f.

To determine the assignment of target points to robots, we use an iterative approach. In each round, we compute that tuple (i, f) where i is the number of a robot and f is a frontier cell which has the best overall evaluation $U_f - \beta V_f^i$ (where β is a constant as explained below). We then recompute the utilities of all frontier cells given the new and all previous assignments according to (4.1). Finally, we repeat this process for the remaining robots. This results in Algorithm 4.1. The complexity of this algorithm is $O(n^2 F)$ where n is the number of robots and F is the number of frontier cells.

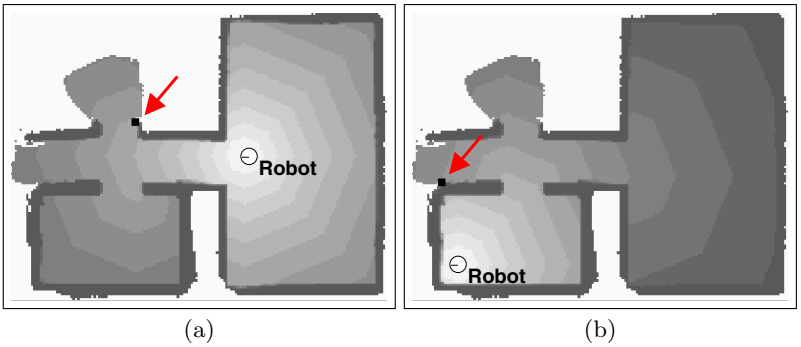
The quantity $\beta \geq 0$ determines the relative importance of utility versus cost. Experiments showed that the exploration time stays nearly constant if $\beta \in [0.1, 50]$. For bigger values of β the exploration time increases because the impact of the coordination is decreased too much. If β is close to 0, the

Algorithm 4.1 Goal assignment for coordinated multi-robot exploration.

1: Determine the set of frontier cells.
2: Compute for each robot i the cost V_f^i for reaching each frontier cell f.
3: Set the utility U_f of all frontier cells to 1.
4: **while** there is one robot left without a target point **do**
5: Determine a robot i and a frontier cell f which satisfy:
$(i, f) = \text{argmax}_{(i',f')} \left(U_{f'} - \beta V_{f'}^{i'} \right)$.
6: Reduce the utility of each target point f' in the visibility area according to $U_{f'} \leftarrow U_{f'} - p_{vis}(f, f')$.
7: **end while**

robots ignore the distance to be traveled which also leads to an increased exploration time. As a result of our experiments, β is set to 1 in our current implementation.

Figure 4.2 illustrates the effect of our coordination technique. Whereas uncoordinated robots would choose the same target position (see Figure 4.1), the coordinated robots select different frontier cells as the next exploration targets. When coordinating a team of robots, one question is when to recompute the target locations. In the case of unlimited communication, we compute new assignments whenever one robot has reached its designated target location or whenever the distance traveled by the robots or the time elapsed after computing the latest assignment exceeds a given threshold.



(a) (b)

Fig. 4.2. Target positions obtained using the coordination approach. In this case, the target point for the second robot is to the left in the corridor.

4.2.4 Coordination with Limited Communication Range

In practice, one cannot assume that the robots are able to exchange information at every point in time. For example, the limited range of nowadays

wireless networks can prevent robots from being able to communicate with other robots at a certain point in time. If the distances between the robots become too large so that not all robots can communicate with each other, a centralized approach as described above can no longer be applied. However, our algorithm can easily be adapted to cope with a limited communication range. The key idea is to apply our centralized approach to each sub-team of robots which are able to communicate with each other. Obviously, this can, at least in the worst case, lead to a situation in which all robots individually explore the whole environment. In practical experiments, however, we found that this approach still results in a quite efficient exploration process, since the robots can quickly exchange the necessary information and coordinate with each other again as soon a connection between them has been reestablished.

In our experiments, we furthermore found that the risk of redundant work is increased if the robots forget about the assignments of other robots as soon as the communication breaks down. Instead, if each robot stores the latest target locations assigned to all other robots the overall performance is increased in situations in which the communication range has been exceeded, since the robots avoid going to places already explored by other robots. This approach turned out to be useful especially in the context of small robot teams.

4.3 Collaborative Mapping with Teams of Mobile Robots

As mentioned before, the robots must be able to build maps online, while they are in motion. The online characteristic is especially important in the context of the exploration task since mapping is constantly interleaved with decision making as to where to move next.

Additionally, to map an environment a real robot has to cope with noise. Our system applies the statistical framework presented in detail in [146] to compute consistent maps while the robots are exploring the environment. Each robot starts with a blank grid map in which each cell is marked as unseen. During exploration, each robot simultaneously performs two tasks: It determines a maximum likelihood estimate for its own position and a maximum likelihood estimate for the map. To recover from possible localization errors, each robot maintains a posterior density characterizing its "true" location. The current version of the multi-robot mapping system relies on the following two assumptions:

1. The robots must begin their operation in nearby locations, so that their range scans show substantial overlap.
2. The software must be told the approximate relative initial pose of the robots. Errors up to 50 cm and 20 degree in orientation are admissible.

To achieve the coordination, the team must be able to communicate the maps of the individual robots during exploration. In our current system,

we assume that the robots set up an ad-hoc network which forms clusters. The messages sent by a robot are forwarded to all team-mates within the corresponding cluster.

Whenever two clusters are merged, care has to be taken to avoid that robots do not become overly confident in the state of the environment. Suppose that each cluster maintains an occupancy grid map built from all observations made by the robots of that team and that two robots which currently share a map m leave their communication range. As long as they explore the environment individually, each robot needs to maintain its own map and update it. As a result, they obtain two different maps m_1 and m_2. Now suppose the robots can communicate again and exchange their maps. If they use the recursive update rule for occupancy grids to combine m_1 and m_2 the information originally contained in m is integrated twice in the resulting map. Integrating the same information several times leads to overly confident map of the environment.

There are several ways to avoid the multiple use of sensor information. One solution is to prevent the robots from exchanging information more than once [43], which reduces the benefit of a multi-robot system. An alternative solution is that each robot maintains an individual map for each other robot. These maps can be combined to a joint map and at the same time be updated separately. In our current system, we apply a different approach that we found to be less memory intensive and requiring less communication bandwidth. In this approach, each robot stores for each other robot a log of sensor measurements perceived by this robot and integrates this information into its own map. A robot only transfers those measurements that have not been transmitted to the corresponding robot so far. Additionally, the robots maintain a small data structure containing the time stamp of the most recent sensor measurement of a robot that was transmitted to all other robots. This allows the robots to discard those measurements which have been received by all other robots already. In this scenario, one of the robots of each sub-team is randomly selected as the leader. This leader performs all the necessary computations to solve the assignment of target locations to robots.

4.4 Experimental Results

The approach described has been implemented on real robots and in different environments. Additionally, we performed extensive simulation experiments.

4.4.1 Exploration with a Team of Mobile Robots

The first experiment is designed to demonstrate the capability of our approach to efficiently cover an unknown environment with a team of mobile robots. To evaluate our approach, we installed three robots (two Pioneer 1 and one iRobot B21) in an empty laboratory environment. Figure 4.3 shows

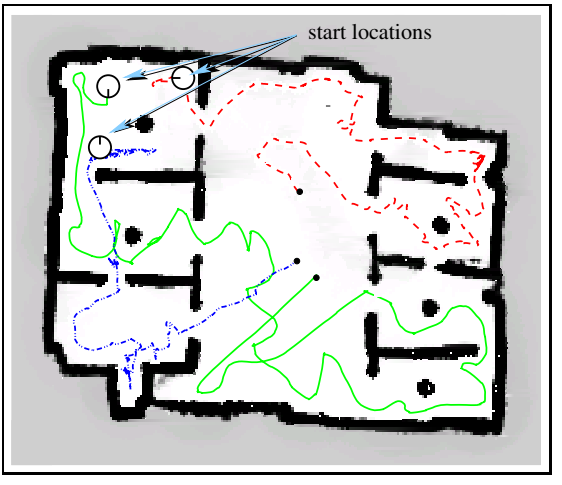


Fig. 4.3. Coordinated exploration by a team of three robots with unlimited communication abilities in a real world experiment. This experiment has been carried out by Mark Moors at the Forschungsgesellschaft für Angewandte Naturwissenschaften (FGAN), Wachtberg, Germany.

the map of this environment. The size of the environment is 18 m by 14 m. Also shown are the paths of the robots which started in the upper left room. As can be seen from the figure, the robots were effectively distributed over the environment by our algorithm. In this experiment, the robots could communicate all the time.

4.4.2 Comparison between Uncoordinated and Coordinated Exploration

The goal of the second experiment described here is to illustrate the advantage of our coordination technique over an approach in which the robots share a map but in which there is no arbitration about target locations so that each robot approaches the closest frontier cell [163]. This technique is called *implicit coordination*. For this experiment, we used two different robots: An iRobot B21 robot equipped with two laser range scanners covering a 360 degree field of view (robot 1) and a Pioneer 1 robot equipped with a single laser scanner covering a 180 degree field of view (robot 2). The size of the environment to be explored in this experiment was 14 m by 8 m and the range of the laser sensors was limited to 5 m.

Figure 4.4 shows the behavior of the two robots when they explore their environment without coordination, i.e., when each robot moves to the closest unexplored location. The white arrows indicate the positions and directions of the two robots. Both robots decide first to explore the corridor. After

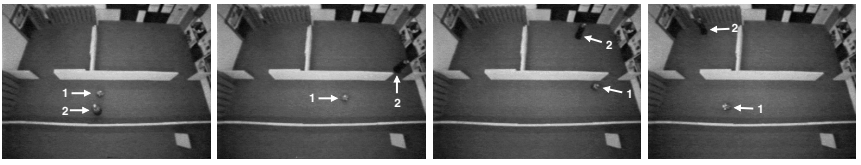


Fig. 4.4. Uncoordinated exploration with two robots, namely an ActivMedia Pioneer robot and an iRobot B21. In the images (a) and (b) both robots drive along the corridor, but robot 1 is slower than robot 2. In image (c), robot 1 reached the end of the corridor, but robot 2 already has explored the right room. Therefore, robot 1 turns around and follows the corridor. In image (d) robot 2 has entered the left room from the right hand side and explored it. This experiment has been carried out by Mark Moors.

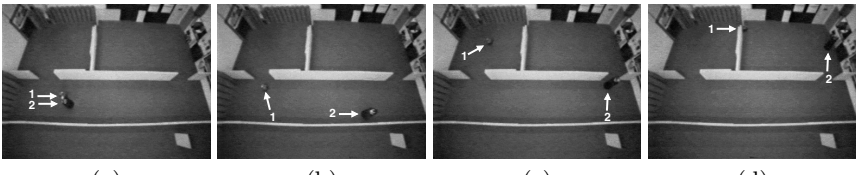


Fig. 4.5. Coordinated exploration using two heterogenous robots. In image (b), both robots focus on different frontiers due to the coordination strategy. Therefore, robot 1 explores the left room and robot 2 the right one. This leads to a better performance compared to the uncoordinated behavior. This experiment has been carried out by Mark Moors.

reaching the end of the corridor robot 2 enters the upper right room. At that point, robot 1 assigns the highest utility to the upper left room and therefore turns back. Before robot 1 reaches the upper left room, robot 2 has already entered it and has completed the exploration mission. As a result, robot 2 explores the whole environment almost on its own while robot 1 does not contribute much. The overall time needed to complete the exploration was 49 seconds in this case.

However, if both robots are coordinated, they perform much better as illustrated in Figure 4.5. Like in the previous example, robot 2 moves to the end of the corridor. Since the utilities of the frontier cells in the corridor are reduced, robot 1 directly enters the upper left room. As soon as both robots have entered the rooms, the exploration mission is completed. This run lasted 35 seconds.

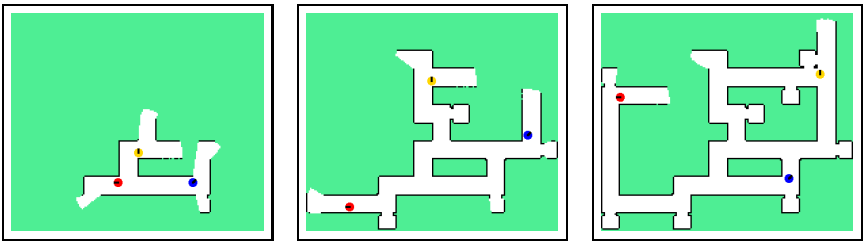


Fig. 4.6. Simulated exploration with three robots.

4.4.3 Simulation Experiments

The previous experiments demonstrate that our approach can effectively guide robots to collaboratively explore an unknown environment. To get a more quantitative assessment, we performed a series of simulation experiments in different environments.

To carry out these experiments, we developed a simulation system, that allows us to consider the effects of various parameters on the exploration performance. The simulator can handle an arbitrary number of robots and can models interferences between the robots. Whenever robots are close to each other, the system performs the planned movement with a probability of 0.7. Thus, robots that stay close to each other move slower than robots that are isolated. This approach is designed to model time delays introduced by necessary collision avoidance maneuvers.

Screenshots of this simulation system during a run in which three robots explore the environment are shown in Figure 4.6. The simulator also allows the specification of different properties of the robot systems and sensors. To carry out these experiments, we used sensors with a 360 degree field of view as is the case, for example, for robots equipped with two laser range sensors or with a ring of ultrasound sensors. Note that our approach does not require a 360 degree field of view. We successfully applied our approach even to robots with a limited field of view, equipped only with a single laser scanner.

Throughout the experiments presented in this section, we compare three different strategies. The first approach is the *implicit coordination* technique used by Yamauchi [163] as well as Singh and Fujimura [138], in which each robot always approaches the closest unexplored area of a joint map. In the sequel, this approach is denoted as *uncoordinated exploration* since it lacks a component that arbitrates between the robots whenever they choose the same frontier cells. The second approach is our coordination approach specified by Algorithm 4.1. Additionally, we evaluated an alternative approach that seeks to optimize the assignments computed in lines 4–7 of our algorithm. Figure 4.7 illustrates a situation in which the assignment computed by our approach is suboptimal. Here, two robots are exploring a corridor with two rooms. The already explored area is depicted in yellow. Suppose both target

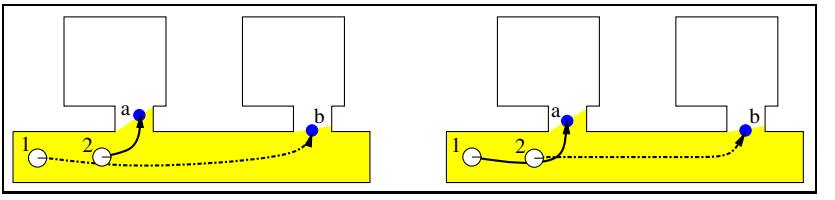


Fig. 4.7. The left image illustrates an assignment of frontiers to robots computed by Algorithm 4.1. Here, yellow corresponds to known areas and white to unknown. The assignment is suboptimal, when considering the overall time to complete the exploration task. The situation shown in the right image provides a better assignment because it leads to a shorter exploration time. In both depicted situation, the sum of the travel cost of both robots are equal but the right one is more balanced.

points a and b have the same utility. In the first round of the iteration, our algorithm assigns robot 2 to target a since this assignment has the least cost of all other possible assignments. Accordingly, in the second round, robot 1 is assigned to target b. The resulting assignment is depicted in the left image of Figure 4.7. If we assume that both robots require the same period of time to explore a room, this assignment is suboptimal. A better assignment is shown in the right image of Figure 4.7. By directing robot 1 to the left room and robot 2 to the right room, the whole team can finish the task faster, because the overall time required to reach and than explore the rooms is reduced. The sum of the travel cost, however, is the same for both assignments.

One approach to solve this problem is to consider all possible combinations of target points and robots. As before, we want to minimize the trade-off between the utility of frontier cells and the distance to be traveled. However, just adding the distances to be traveled by the two robots does not make a difference in situations like that depicted in Figure 4.7. Since the robots execute their actions in parallel the time to complete the whole task depends on the longest trajectory. To minimize the completion time (by choosing more balanced trajectories for the individual robots), we therefore modify the evaluation function so that it considers squared distances to choose target locations $f_1, \ldots, f_n$:

$$\underset{(f_1,\ldots,f_n)}{\operatorname{argmax}} \sum_{i=1}^{n} \left[U(f_i \mid f_1, \ldots, f_{i-1}, f_{i+1}, \ldots, f_n) - \beta (V_{f_i}^i)^2 \right].$$

The resulting algorithm that determines in every round the optimal assignment of robots to target locations according to this evaluation function is given in Algorithm 4.2. Compared to the selection scheme of our previous algorithm, the major problem of this approach lies in the fact that one has to figure out $\frac{F!}{(F-n)!}$ possible assignments where F is the number of target

Algorithm 4.2 Goal assignment over all permutations.

1: Determine the set of frontier cells.
2: Compute for each robot i the cost V_f^i for reaching each frontier cell.
3: Determine a set of target locations $f_1, \ldots, f_n$ for the robots $i = 1, \ldots, n$ that maximizes the following evaluation function: $\sum_{i=1}^{n} U(f_i \mid f_1, \ldots, f_{i-1}, f_{i+1}, \ldots, f_n) - \beta(V_{f_i}^i)^2$.

locations, n is the number of robots, and $n \leq F$[1]. This number can be handled for small teams of robots but it becomes intractable for larger teams, because the number of possible assignments grows exponentially in the number of robots. In practice, one therefore needs appropriate search techniques to find good assignments in a reasonable period of time. In the experiments described here, we applied a randomized search technique combined with hill-climbing to search for optimal assignments of frontiers to robots. The approach starts with the assignment provided by Algorithm 4.1 and tries to optimize the assignment by exchanging target locations between the robots. It also uses several restarts based on the solution provided by Algorithm 4.1 in order to reduce the risk of getting stuck in a local maxima. This technique is in the following called *randomized* strategy.

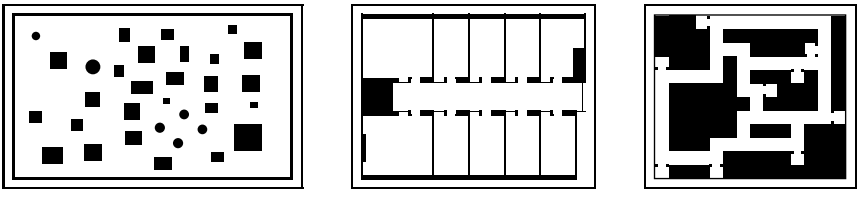


Fig. 4.8. Maps used for the simulation experiments: unstructured (left), office (middle), and corridor environment (right).

To compare the different exploration strategies, we chose three different environments which are depicted in Figure 4.8. For each environment and each number of robots, we performed 50 different simulation experiments for each strategy. In each comparison of the three strategies, the robot team was started at the same randomly chosen location. We then evaluated the average number of time steps the robots needed to complete the job. The resulting plots are shown in Figure 4.9. The error bars indicate the 5% confidence level. As can be seen from the figure, the team using our algorithm significantly outperforms the uncoordinated system with respect to the exploration time. This is mainly due to the fact that Algorithm 4.1 provides a better distribution of the robots over the environment.

[1] In case $n > F$, we allow each frontier to be assigned $\lceil \frac{n}{F} \rceil$ times to a robot.

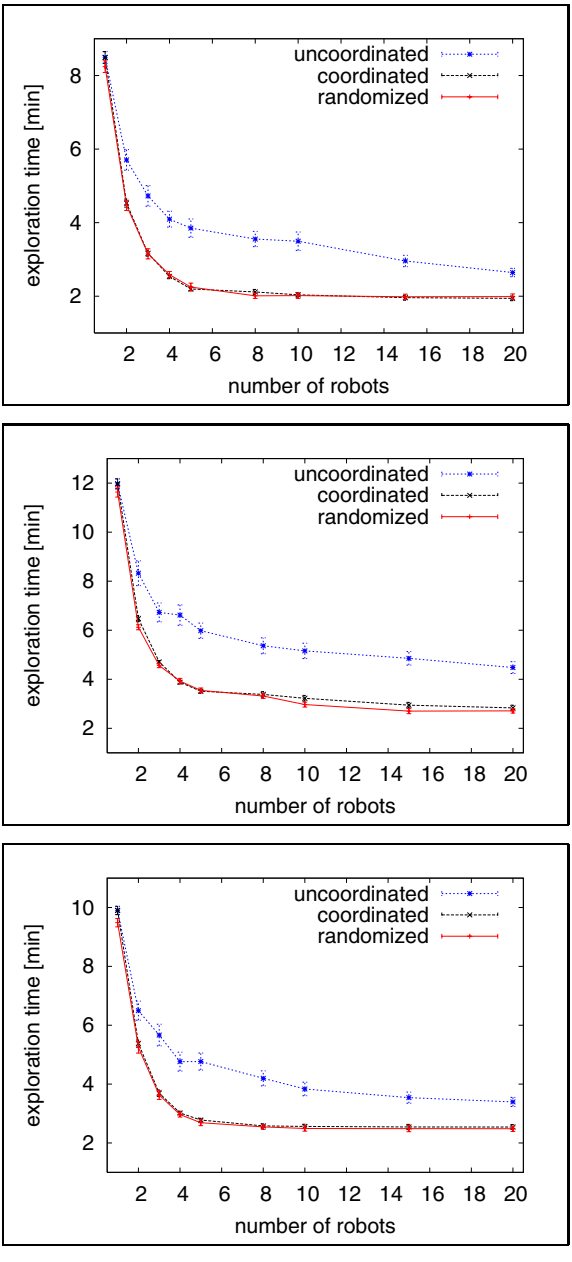


Fig. 4.9. Performance of the different coordination strategies for the environments shown in Figure 4.8: unstructured environment (top), office environment (middle), and corridor environment (bottom).

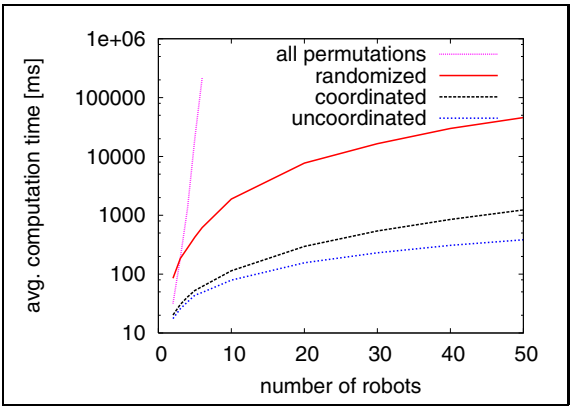


Fig. 4.10. Time required on a Pentium-4, 2.8 GHz machine to compute the assignment of target locations to robots for three different strategies.

The randomized optimization strategy usually yields slightly better results than our coordination technique although the improvement is not significant. Thus, the usage of the complex search technique that seeks to determine the optimal assignment from all $\frac{F!}{(F-n)!}$ permutations appears to yield only slight improvements compared to our original algorithm which has complexity $O(n^2F)$. Given the computational overhead introduced by the randomized search in the space of all permutations (see Figure 4.10) especially for large teams of robots, Algorithm 4.1 appears to be preferable over Algorithm 4.2.

4.4.4 Exploration with Limited Communication

The next experiments are designed to analyze the performance of our coordination strategy in case the robots have only a limited communication range. As explained above, if the communication range is limited the robots cannot globally coordinate their actions anymore. As a result, different robots may explore the same regions which reduces the overall efficiency.

The next real world experiment was carried out with three robots. Throughout this experiment, we limited the communication range to 5 m. Figure 4.11 depicts the exploration process. Each row shows the maps of the individual robots at different points in time. The initial situation is depicted in the first row. The communication ranges of the robots are highlighted by colored disks around each robot. As can be seen from the second row, the robots quickly split up in this experiment and had to plan their trajectories individually. In row three, the robots R_1 and R_3 are able to communicate again and therefore can exchange their maps and coordinate their behavior again. Robot R_2, however, still acts independently of the other two robots. In row five, R_1 and R_3 again leave their communication range, whereas R_2 and R_3

are able to merge their maps and approach the last unexplored area in the top left corner. In the last row, the robots R_2 and R_3 have covered the whole environment and in this way have completed the exploration task.

To analyze the influence of the communication range, we performed a series of simulation experiments. We carried out 50 simulation runs for each configuration using a different number of robots and different communication ranges. In each run, we chose a random starting point for the robot team. We regard the exploration task as completed as soon as the known area in the map of one robot covers the whole environment. The results are depicted in Figure 4.12. The x-axis shows the communication range of the robots in relation to the maximum distance in the map and the y-axis depicts the average exploration time. If the communication range is close to zero the coordinated and uncoordinated strategies behave similar because all robots act independently most of the time. As the communication range increases, the benefit of the coordinated approach improves. An interesting result of this experiment is that a communication range of 30% of the diameter of the environment appears to be sufficient to yield the same performance as with unlimited communication.

4.5 Comparisons to Other Coordination Techniques

In this section, we compare our approach to other existing techniques to assign targets to a team of robots. First, we compare our approach to the Hungarian method [79]. We then discuss a priorization technique to distribute the robots over the environment. Finally, we discuss exploration techniques that apply a solution to the traveling salesman problem (TSP) to coordinate the team of robots.

4.5.1 Target Assignment Using the Hungarian Method

In 1955, Kuhn [79] presented a general method to assign a set of jobs to a set of machines given a fixed cost matrix. This method is often referred to as the Hungarian method. Consider a given $n \times n$ cost matrix which represents the cost of all individual assignments of targets to robots. The Hungarian method, which is able to find the solution with the minimal cost given that matrix, can be summarized by the following three steps:

1. Compute a reduced cost matrix by subtracting from each element the minimal element in its row. Afterwards, do the same with the minimal element in each column.
2. Find the *minimal number* of horizontal and vertical lines required to cover all zeros in the matrix. In case exactly n lines are required, the optimal assignment is given by the zeros covered by the n lines. Otherwise, continue with Step 3.

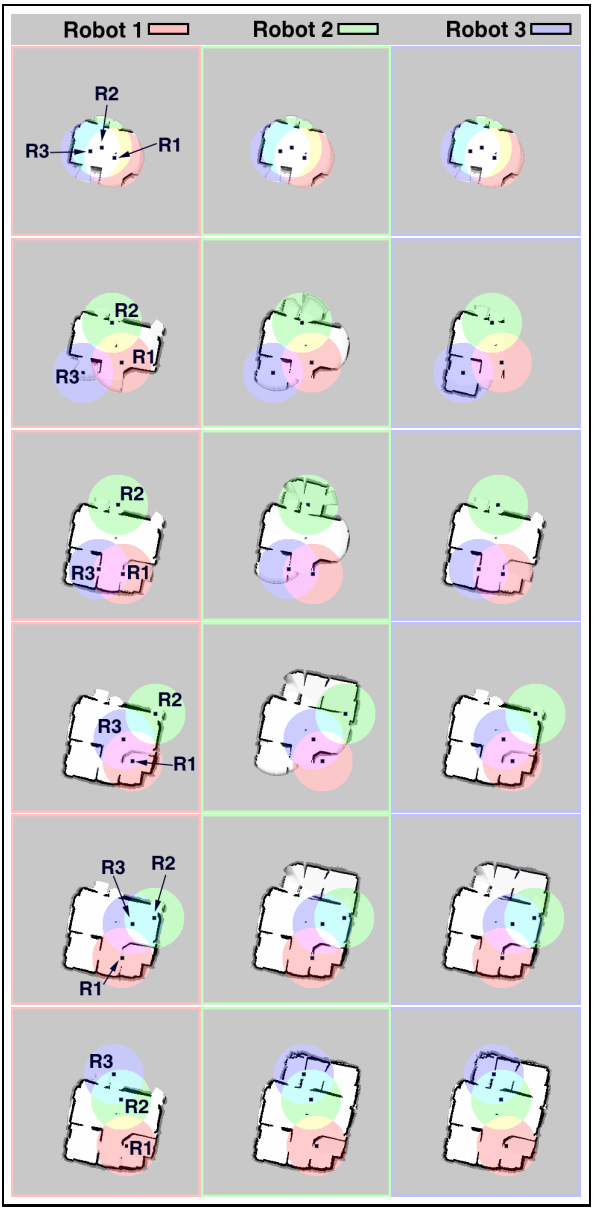


Fig. 4.11. Coordinated exploration by a team of three robots with limited communication abilities. Each column shows the evolution of the map of one robot over time. This experiment has been carried out by Mark Moors and Frank Schneider at the Forschungsgesellschaft für Angewandte Naturwissenschaften (FGAN), Wachtberg, Germany.

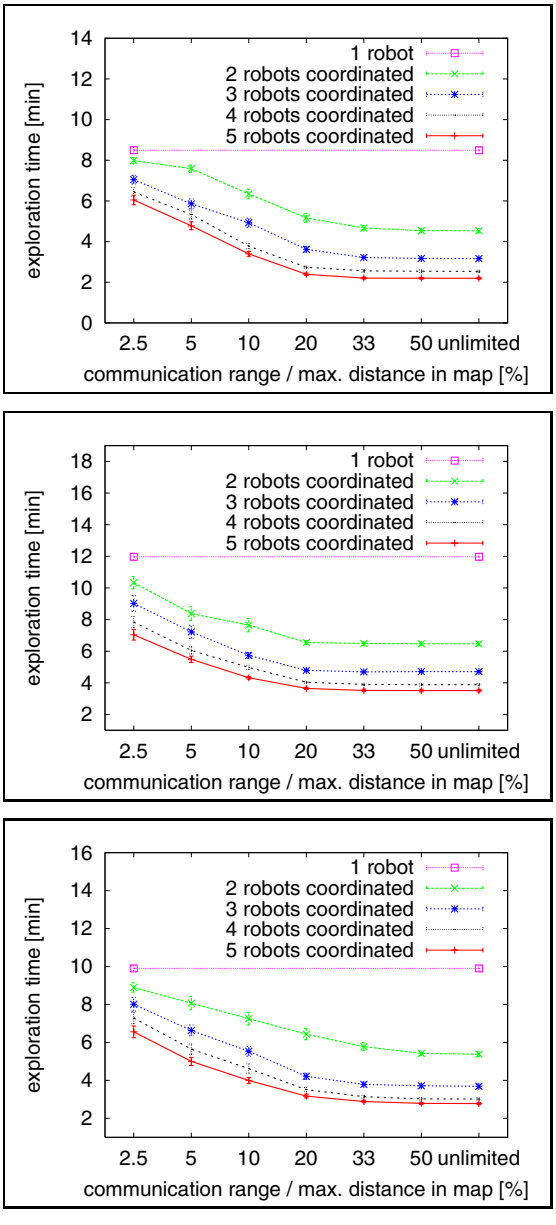


Fig. 4.12. Performance of the coordinated strategy with limited communication range for the different environments (unstructured (top), office (middle), and corridor environment (bottom)). The x-axis shows the communication range in relation to the size of the environment, the y-axis the average exploration time. As can be seen, the results of these experiments look very similar in all tested environments.

3. Find the smallest nonzero element in the reduced cost matrix that is not covered by a horizontal or vertical line. Subtract this value from each uncovered element in the matrix. Furthermore, add this value to each element in the reduced cost matrix that is covered by a horizontal and a vertical line. Continue with Step 2.

The computationally difficult part lies in finding the minimum number of lines covering the zero elements (Step 2). Details can be found in [79, 80]. The overall algorithm has a complexity of $O(n^3)$. This method can be applied to assign a set of frontiers or target locations to the individual robots. In such a scenario, the cost matrix is defined by the result of a deterministic value iteration carried out for each robot (see Section 4.2.1).

Since the implementation of the Hungarian method described above requires that the number of jobs and the number of machines is equal, we need to slightly adapt the cost matrix computed in that way. We can distinguish two situations:

1. In case $n < F$, where n is the number of robots and F the number of frontier cells, we add $F - n$ dummy robots which introduce zero cost for any assignment. The frontier cells to which these dummy robots are assigned to represent target locations that are not selected by a real robot.
2. In case $n > F$, some robots need to be assigned to the same target location. To achieve a balanced distribution of robots over the environment, we copy each frontier $\lceil \frac{n}{F} \rceil$ times so that not more than $\lceil \frac{n}{F} \rceil$ robots are assigned to a single target location. In case $n < F\lceil \frac{n}{F} \rceil$, we then add $F\lceil \frac{n}{F} \rceil - n$ dummy robots.

In this way, we obtain a square cost matrix even if $n \neq F$. In the worst case, the matrix has a dimension of $2 \max\{n, F\}$. Thus, the overall cost of coordinating a team of n robots given F possible target locations is $O(\max\{n, F\}^3)$.

The advantage of that method compared to our approach described in Algorithm 4.1 is that the Hungarian method computes the optimal assignment under the given cost matrix. In contrast to that, our algorithm applies a greedy technique to assigned robots to frontiers. On the other hand, the Hungarian method is not able to adapt the cost matrix during the assignment process. Such an adaption is performed by our algorithm in order to account for the fact, that assigning a frontier cell f to a robot affects the unassigned frontier cells close to f. This fact cannot be modeled when using the Hungarian method, since it requires a constant cost matrix.

We applied the Hungarian method in the same scenarios than our coordination technique presented in Algorithm 4.1 to evaluate its performance. We figured out, that the Hungarian method yields similar results as our coordination technique for large teams of robots. Plots showing the performance of both approaches are depicted in Figure 4.13. As can be seen from this figure, if the team of robots is small, our coordination approach performs better. This is due to the fact, that our technique considers the visibility between frontiers when computing their utility in the assignment process. This leads

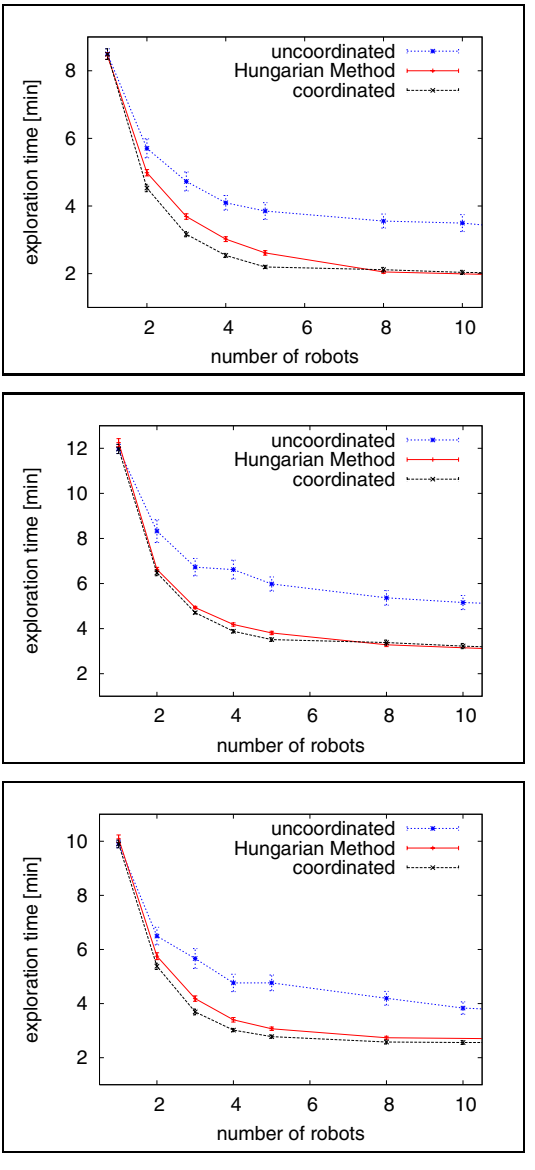


Fig. 4.13. Coordination performance of the Hungarian method evaluated in the unstructured (top), office (middle), and corridor environment (bottom).

to a better distribution of robots over the environment, which typically results in shorter exploration times. However, as soon as the size of the team gets bigger, this effect vanishes and both techniques perform equally well. An additional advantage of our approach is that it is much easier to implement compared to the Hungarian method.

4.5.2 Using a Priorization Scheme to Coordinate a Team of Robots

The second method we compared our approach to is the usage of a priorization scheme while selecting target locations. First, this approach assigns priorities to the individual robots. After a target location is assigned to a robot, this information is transmitted to all other robots. Each robot plans its action by taking into account the decisions made by robots with higher priority. Such a technique with a fixed priorization scheme typically performs worse than our coordination technique especially in the context of large robot teams. The reason for that is, that our approach can be regarded as a priorized approach where in each planning step the priorization scheme is adapted so that it promises the highest utility.

However, we compared this approach to our coordination scheme, since a fixed priorization scheme can be directly applied in multi-robot systems using a decentralized architecture. In contrast to this, our coordination algorithm requires a central coordinator (or a coordinating robot within each sub-team) that computes the assignment. Furthermore, such a decentralized priorization scheme needs less network bandwidth compared to the centralized approach. Therefore, it makes sense to apply such a technique, if only a slow network connection is available. This problem has been addressed in detail in a joined work with Daniel Meier (see [98]). In this approach, a polygonal approximation of the environment is computed. The polygons are incrementally refined depending on the available network bandwidth. The operations to carry out the refinement are computed using the minimum edit cost between the polygons. In this way one is able to substantially reduce the network traffic.

As can be seen in the plot in Figure 4.14, the quality of the priorized scheme is satisfactory for small teams of robots. However, as soon as the group gets larger, the performance of the algorithm decreases. In some situations using around 20 robots, this approach was even worse than the uncoordinated behavior. The reason for that is that the robots with a low priority do not gather any useful information and are often redirected before they really reach their desired goal location. At the same time, they cause interferences between robots. We believe that this method can be further optimized by, for example, reassigning priorities [10] or auction-based approaches that allow the robots to trade with their target locations [167].

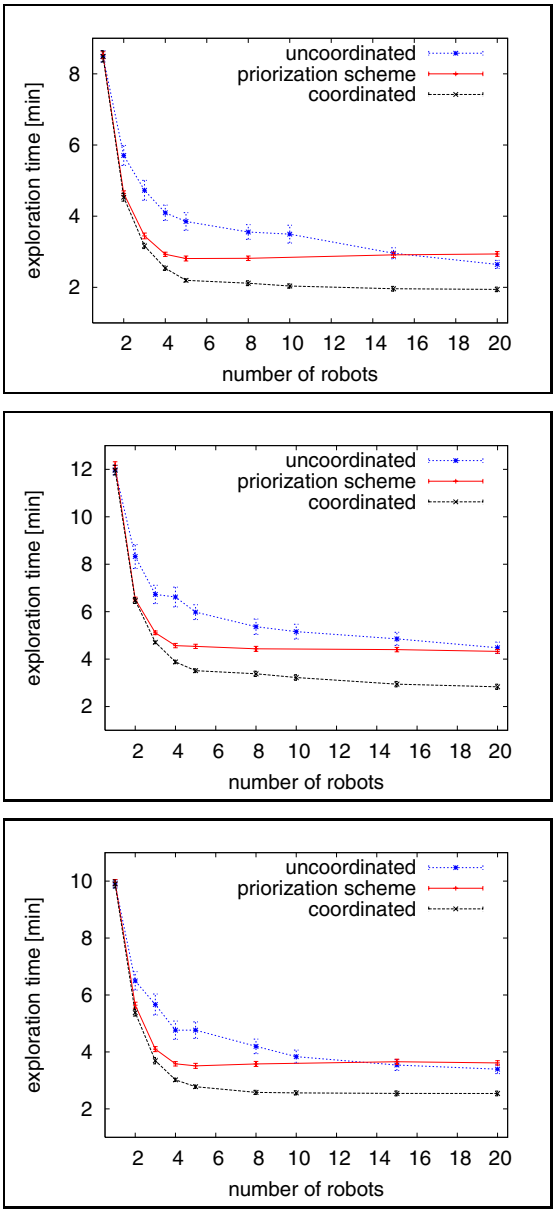


Fig. 4.14. Coordination performance of the priorized coordination method evaluated in the unstructured (top), office (middle), and corridor environment (bottom).

4.5.3 Coordination of a Team of Robots by Solving a TSP

An alternative solution to the multi-robot coordination problem is to solve a multi-agent traveling salesman problem (multi-agent TSP). In this approach, all available target locations are assigned to the robots and each robot builds a path by visiting multiple target locations instead of approaching a single one.

It should be noted that the computation of the optimal solution is in most cases impossible due to the online-characteristics of the exploration problem. Approximative solutions, however, open additional problems like the question how to balance the number of tasks assigned to the individual robots. Zlot *et al.* [167] who used an online auction-based approach similar to a TSP write *"since new frontiers generally originate from old ones, the robot that discovers a new frontier will often be the best suited to go on it (the closest)."* This observation indicates that often it is sufficient to consider only a single target location. A typical situation, in which a suboptimal solution is obtained when the workload is not balanced between the robots, is depicted in Figure 4.15. In this example, all the work is done by one robot and the other remains idle. This effect can get even stronger if the size of the team grows.

Applying TSP solutions in the context of exploration makes sense if, for example, the structure of the environment is (partly) known but the world needs to be covered by the sensors of the robots. This can be the case in the context of de-mining or cleaning tasks. There exits evaluations of different approximative solutions in the literature (compare [85]), but they typically assume that the environment is at least partly known.

4.6 Related Work

The various aspects of the problem of exploring unknown environments with mobile robots have been studied intensively in the past. Many approaches have been proposed for exploring unknown environments with single robots [22, 34, 36, 53, 82, 105, 142, 144, 165]. Most of these approaches guide the robot to the closest unexplored area, just as our approach does when applied to a single robot system. These techniques mainly differ in the way the environment is represented. Popular representations are topological [22, 82], metric [36], or grid-based [163, 164, 165]. Furthermore, there exists theoretical works providing a mathematical analysis of the complexity of exploration strategies including comparisons for single robots [1, 2, 26, 27, 78, 77, 166, 93, 119]. Additionally, Lee and Recce [86] provide an experimental analysis of the performance of different exploration strategies for one mobile robot.

Also the problem of exploring terrains with teams of mobile robots has received considerable attention in the past. For example, Rekleitis *et al.* [120, 121, 122] focus on the problem of reducing the odometry error during exploration. They separate the environment into stripes that are explored

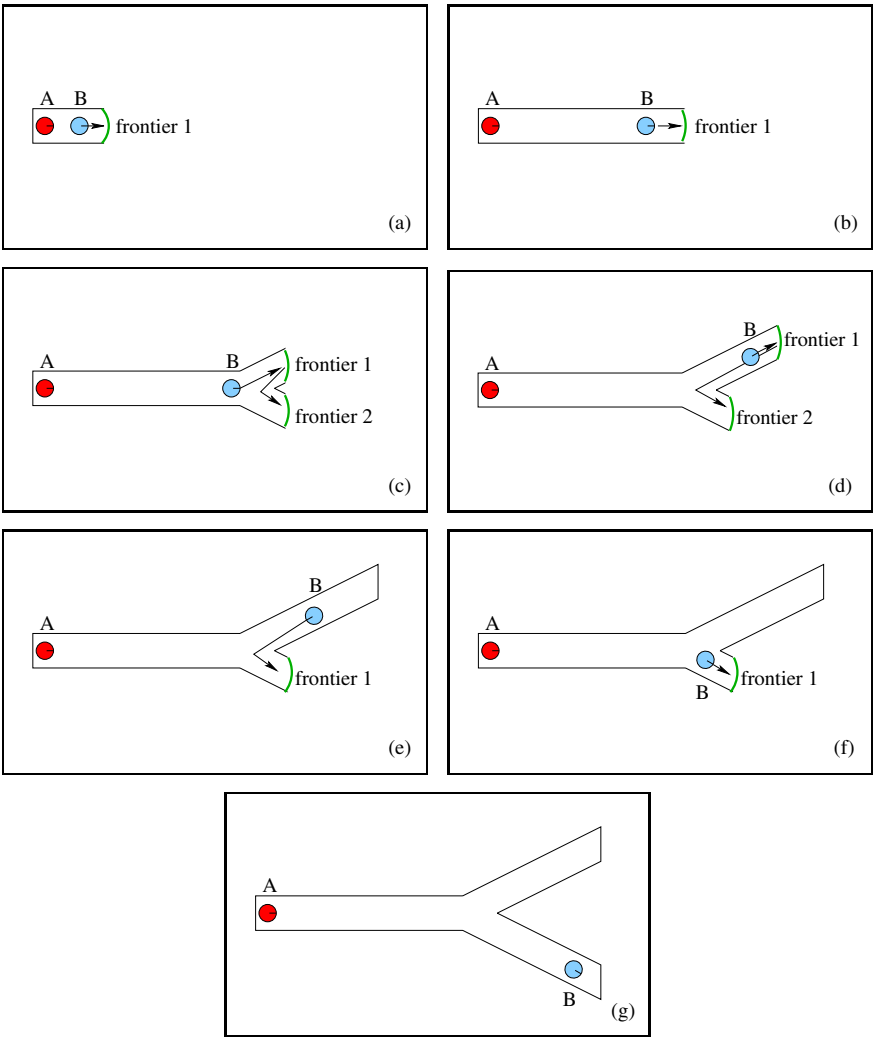


Fig. 4.15. An environment in which an online multi-agent TSP-solution can be problematic. (a) Two robots start in a Y-shaped corridor. The only frontier is assigned to robot B since it is the closest to this frontier. (b) The new frontier originates from the old one and so robot B is the best suited to go on it. (c) Robot B reaches the junction and the shortest path in this TSP is to guide robot B to frontier 1 and than to frontier 2. (d) Robot B enters the upper corridor, robot A has still no task assigned. (e) Robot B explores the upper corridor and turns back. Since the upper corridor is shorter than the horizontal one, robot B still has the frontier labeled as 1 in its route. (f) Robot B enters the lower corridor until the whole environment is explored (g). This solution is clearly suboptimal, since robot A was not used at all.

successively by the robot team. Whenever one robot moves, the other robots are kept stationary and observe the moving robot, a strategy similar to the one presented by Kurazume and Shigemi [84]. Whereas this approach can significantly reduce the odometry error during the exploration process, it is not designed to distribute the robots over the environment like our approach does. The robots are rather forced to stay close to each other in order to remain in the visibility range. Thus, using these strategies for multi-robot exploration one cannot expect that the exploration time is significantly reduced compared to single robot systems.

Yamauchi [163] presented a technique to learn maps with a team of mobile robots. He introduced the idea of frontiers between known and unknown areas in a grid map. The frontier technique is also used throughout this work, since it is well-suited to find potential target locations for singe as well as for multi-robot systems. In the approach of Yamauchi, the robots exchange information via a joint map that is continuously updated whenever new sensor input arrives. They also use map matching techniques [164] to improve the consistency of the resulting map. To acquire new information about the environment all robots move to the closest frontier cell. The authors do not apply any strategies to distribute the robots over the environment or to avoid that two or more robots exploring the same areas. This type of implicit coordination via a joint map is used as a reference technique for comparisons throughout this chapter. We called it the "uncoordinated technique" in this chapter. As shown in the experimental section, our coordination technique provides a more efficient coverage of terrain for multi-robot systems.

Cohen [23] considers the problem of collaborative of a *navigator* that has to reach an initially unknown target mapping and navigation of teams of mobile robots. The team consists location and a set of *cartographers* that randomly move through the environment to find the target location. When a robot discovers the goal point, the location is communicated among the cartographers to the navigation robot which then starts to move to that location. In extensive experiments, the author analyzes the performance of this approach and compares it to the optimal solution for different environments and different sizes of robot teams. In our approach, the robots do not have that different capabilities or different tasks to complete. Our systems allows the robots to travel with different speeds or to have a different size. Compared to Cohen [23], we do not consider robots supposed to solve only one a specific task within the exploration mission.

Koenig and colleagues [76, 77, 166] presented several evaluations of different terrain coverage techniques for single and multi-robot systems. Koenig and Tovey [77] for example demonstrated that for single robot systems, the greedy approach that guides the robot always to the closest frontier behaves reasonable well compared to the optimal solution. Recently, Zheng *et al.* [166] showed that under certain assumptions like fixed sensor range and grid cell ratios as well as unlimited communication, their greedy coverage algorithm needs in the worst case only eight time longer than the optimal solution.

Koenig *et al.* [76] analyze different terrain coverage methods for ants which are simple robots with limited sensing and computational capabilities. They consider environments that are discretized into equally spaced cells. Instead of storing a map of the environment in their memory like done in our exploration approach, the ants leave markers in the cells they visit. The authors consider two different strategies for updating the markers. The first strategy is "Learning Real-Time A*" (LRTA*), which greedily and independently guides the robots to the closest unexplored areas and thus results in a similar behavior of the robots as in the approach of Yamauchi [163]. The second approach is "Node Counting" in which the ants simply count the number of times a cell has been visited. The authors show that Learning Real-Time A* (LRTA*) is guaranteed to be polynomial in the number of cells, whereas "Node Counting" can be exponential.

Billard *et al.* [13] introduce a probabilistic model to simulate a team of mobile robots that explores and maps locations of objects in a circular environment. In several experiments, they demonstrate the correspondence of their model with the behavior of a team of real robots.

In [7], the authors analyze the effects of different kinds of communication on the performance of teams of mobile robots that perform tasks like searching for objects or covering a terrain. The "graze task" carried out by the team of robots corresponds to an exploration behavior. One of the results is that the communication of goal locations does not help if the robots can detect the "graze swathes" of other robots.

The technique presented by Kurabayashi *et al.* [83] is an off-line approach, which, given a map of the environment, computes a cooperative terrain sweeping technique for a team of mobile robots. In contrast to most other approaches, this method is not designed to acquire a map. Rather the goal is to minimize the time required to cover a known environment which can lead to a more efficient behavior in the context of cleaning or mowing tasks.

One approach towards cooperation between robots has been presented by Singh and Fujimura [138]. This approach especially addresses the problem of heterogeneous robot systems. During exploration each robot identifies "tunnels" to the so far unexplored area. If a robot is too big to pass through a tunnel it informs other robots about this task. Whenever a robot receives a message about a new task, it either accepts it or delegates it to smaller robots. In the case of homogeneous teams, the robots follow a strategy similar to the system of Yamauchi [163].

Howard *et al.* [64] presented an incremental deployment approach that is similar to the technique described here. Their approach explicitly deals with obstructions. They consider situations in which the path of one robot is blocked by another but they do not consider the problem of limited communication. Zlot *et al.* [167] have recently proposed an architecture for mobile robot teams in which the exploration is guided by a market economy. In contrast to our algorithm, they consider sequences of potential target locations for each robot like in a TSP. They then trade the tasks using single-item

first-price sealed-bid auctions. As illustrated in this chapter, the usage of a TSP-approach can be disadvantageous in unknown environments. Whenever a robot discovers a new frontier during exploration, this robot will often be the best suited to go on it (see [167]). As illustrated in Section 4.5.3, we found that this can lead to an unbalanced assignment of tasks to robots so that the overall exploration time is increased. Ko *et al.* [74] present a variant of our approach that uses the Hungarian method [79] to compute the assignments of frontier cells to robots. The main focus of this work is to cooperatively explore an environment with a team of robots in case the starting locations of the individual robots are not known in advance. Practical experiments presented in this chapter showed that the Hungarian method yields a similar performance as our coordination algorithm. Only in the case of small robot teams our approach appeared to be slightly superior since it provides a better distribution of the robots over the environment.

Furthermore, there are approaches that address the problem of coordinating two robots. The work presented by Bender and Slonim [9] theoretically analyzes the complexity of exploring strongly connected directed graphs with two robots. Roy and Dudek [127] focus on the problem of exploring unknown environments with two robots and present an approach allowing the robots with a limited communication range to schedule rendezvous. The algorithms are analyzed analytically as well as empirically using real robots.

There exist also coordination techniques optimized for a specific domain. For example, Weigel *et al.* [159] presented an approach to coordinate a team of soccer playing robots. This technique does not directly address the problem of exploring unknown environments but of assigning roles to the individual agents. These roles are soccer specific ones like, for example, "defense player". In this way, the team is able to adapt itself to the current situation of the soccer field.

Several researchers have focused on architectures for multi-robot cooperation. For example, Grabowski *et al.* [55] consider teams of miniature robots that overcome the limitations imposed by their small scale by exchanging mapping and sensor information. In this architecture, a team leader integrates the information gathered by the other robots. Furthermore, it directs the other robots to move around obstacles or to direct them to unknown areas. Jung and Zelinksy [70] present a distributed action selection scheme for behavior-based agents which has successfully been applied to a cleaning task. Stroupe *et al.* [143] recently presented the MVERT-approach. Their system uses a greedy approach that selects robot-target pairs based on proximity. The goal of the action selection is to maximize cooperative progress toward mission goals. In contrast to our algorithm, the MVERT system does not discount areas close to the selected goal locations. Matarić and Sukhatme [97] consider different strategies for task allocation in robot teams and analyze the performance of the team in extensive experiments. Parker [115] described a project in which a large team of heterogeneous robots is used to perform reconnaissance and surveillance tasks. This work differs from our approach

in that it investigates how to jointly accomplish a task with heterogeneous robots that cannot solve it individually.

4.7 Conclusion

In this chapter, we presented a technique for coordinating a team of robots while they are exploring their environment. The key idea of our technique is to simultaneously take into account the cost of reaching a so far unexplored location and its utility. The utility of a target location depends on the probability that this location is visible from target locations assigned to other robots. Our algorithm always assigns that target location to a robot which has the best trade-off between utility and costs. We also presented an extension of our technique to multi-robot systems that have a limited communication range. In this case, the robots form sub-teams so that they are able to communicate locally. The assignment problem is then solved within each sub-team.

Our technique has been implemented and tested on real robots and in extensive simulation runs. The experiments demonstrate that our algorithm is able to effectively coordinate a team of robots during exploration. They further reveal that our coordination technique significantly reduces the exploration time compared to exploration approaches that do not explicitly coordinate the robots. Further experiments demonstrate that our technique works well even if the robots can only partially exchange data. Additionally, we compared our approach to three alternative coordination techniques, namely the implicit coordination approach based on a joint map, the Hungarian method, and a coordination approach using a fixed priority scheme.

5

Multi-Robot Exploration Using Semantic Place Labels

5.1 Introduction

In the previous chapter, we introduced a technique to efficiently coordinate a team of exploring robots. So far, we made no assumption about the environment itself. In this chapter, we extend our coordination approach presented in Chapter 4 so that it takes into account additional information about the environment.

Indoor environments constructed by humans often contain certain structures like corridors with adjacent rooms or offices. However, it is mainly unexplored how robots can utilize such background information to more efficiently solve an exploration task. One of our observations is that the more potential target locations are known when assigning targets to robots, the faster the team can explore the environment. This is due to the fact that especially large teams of robots can be better distributed over the environment when more target locations are available. In this way, the amount of redundant work and the risk of interference is reduced. It therefore makes sense to focus on areas first which are likely to provide a large number of new target locations in order to obtain a better assignment of targets to robots.

The contribution of this chapter is a technique to estimate and utilize semantic information during collaborative multi-robot exploration. In our approach, the robots get a higher reward for exploring corridors, since they typically provide more branchings to unexplored areas like adjacent rooms compared to rooms itself. This allows us to make better assignments of target locations to robots. As a result, the overall completion time of an exploration task can be significantly reduced.

This chapter is organized as follows. First, we introduce our technique to estimate semantic labels of places. In Section 5.3, we then present a hidden Markov model-based extension to the labeling approach which improves the classification in the context of exploration. We then propose our coordination technique and describe how to utilize the place information in Section 5.4.

C. Stachniss: Robotic Mapping and Exploration, STAR 55, pp. 73–90.
springerlink.com

Section 5.5 presents experimental results on exploration using semantic place labels. Finally, Section 5.6 discusses related work.

5.2 Semantic Place Labeling

This section explains how semantic place labels can be obtained with mobile robots based on laser range observations. We apply a technique for place classification which was presented in a joint work with Martínez Mozos [95]. The techniques allows a mobile robot to robustly classify places into different semantic categories. In this chapter, we focus on learning a classifier, that is able to distinguish corridors from other kinds of indoor structure. To obtain such a classifier, we apply the AdaBoost algorithm introduced by Freund and Schapire [48].

The key idea of AdaBoost is to form a strong binary classifier given a set of weak classifiers. The weak classifiers h_j only need to be better than random guessing. Similar to the work of Viola and Jones [156], we construct our weak classifier based on simple, single-value features f_j as

$$h_j(x) = \begin{cases} 1 \text{ if } p_j f_j(x) < p_j \theta_j \\ 0 \text{ otherwise.} \end{cases} \tag{5.1}$$

This weak classifier returns 1 if the training example x is supposed to be a positive example and 0 otherwise. θ_j is a threshold value and p_j is either -1 or $+1$ and thus represents the direction of the inequality. The AdaBoost algorithm determines during the training process for each weak classifier h_j the optimal parameter tuple (θ_j, p_j), such that the number of misclassified training examples is minimized. To achieve this, it considers all possible combinations of p_j and θ_j, whose number is limited since only an finite number N of training examples is given. A training example is defined by the tuple (x_n, y_n) where x_n is the example and $y_n \in \{0, 1\}$ the class x_n belongs to. Using the training examples, (θ_j, p_j) is determined by

$$(\theta_j, p_j) = \underset{(\theta_i, p_i)}{\operatorname{argmin}} \sum_{n=1}^{N} |h_i(x_n) - y_n| \,. \tag{5.2}$$

Figure 5.1 illustrates the process to compute the optimal value of θ_j. First, one computes for each training example (x_n, y_n) the feature value $f_j(x_n)$ and adds it to a list which is sorted according to that value; Second, one iterates through this list and computes the error of the weak classifier using a θ_j value between the feature value of the current and the next element. The θ_j value which provides the highest classification rate is the optimal value for θ_j given the training set.

We compute two sets of simple features for each observation. The first set is calculated using the raw beams $z_{t,i}$, $i = 1, \ldots, M$ in the full range scan z_t.

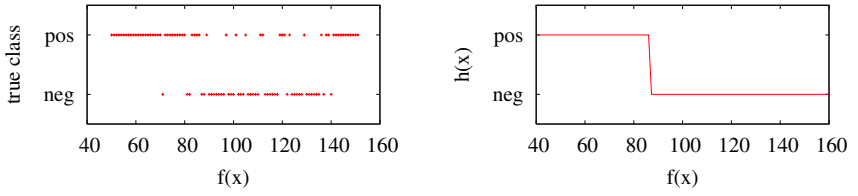


Fig. 5.1. This figure illustrates how the optimal value of θ_j is found. In the left image, the x-value of each data point represents the feature value of a training example and the y-value its true class. By iterating through this list of data points, one can determine the optimal value θ_j for the given training examples. The right image depicts the weak classifier h_j.

The second set of features is calculated from a polygonal approximation $\mathbf{P}(z_t)$ of the area covered by z_t. The vertices of the closed polygon $\mathbf{P}(z_t)$ correspond to the coordinates of the end-points of each beam relative to the robot.

$$\mathbf{P}(z_t) = \{(z_{t,k} \cos \alpha_k, z_{t,k} \sin \alpha_k) \mid k = 1, \ldots, M\}, \tag{5.3}$$

where α_k is the angle of the k-th beam z_t, k of the observation z_t.

Examples for features extracted from laser range data are depicted in Figure 5.2. Such features are, for example, the average distance between consecutive beams, the area covered by a range scan, or the perimeter of that area. All our features are rotational invariant to make the classification of a position dependent only on the (x, y)-position of the robot and not on its orientation. Most of the features are standard geometrical features used in shape analysis [51, 129]. Table 5.1 and 5.2 provide a full list of features used by our system to learn classifier for place recognition.

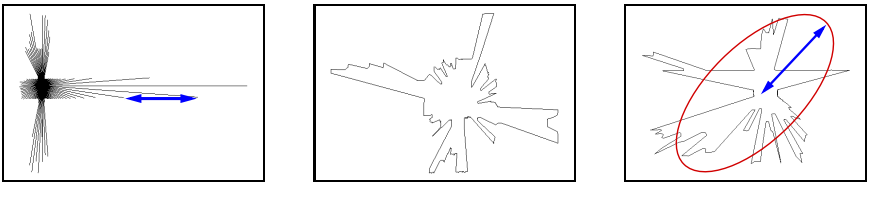


Fig. 5.2. Examples for features generated from laser data, namely the average distance between two consecutive beams, the perimeter of the area covered by a scan, and the length of the major axis of the ellipse that approximates the polygon described by the scan.

The input to the AdaBoost algorithm is a set of labeled, positive and negative training examples $\{x_n, y_n\}$. In our case, this is a set of laser range observations recorded in a corridor and a second set taken outside corridors.

Table 5.1. Simple features based on the individual beams of a laser range observation z

1. The average difference between the length of consecutive beams.
2. The standard deviation of the difference between the length of consecutive beams.
3. Same as 1), but considering different max-range values.
4. The average beam length.
5. The standard deviation of the length of the beams.
6. Number of gaps in the scan. Two consecutive beams build a gap if their difference is greater than a given threshold. Different features are used for different threshold values.
7. Number of beams lying on lines that are extracted from the range scan [131].
8. Euclidean distance between the two points corresponding to the two smallest local minima.
9. The angular distance between the beams corresponding to the local minima in feature 8).

Table 5.2. Features computed based on the polygon $\mathbf{P}(z)$

1. Area of $\mathbf{P}(z)$.
2. Perimeter of $\mathbf{P}(z)$.
3. Area of $\mathbf{P}(z)$ divided by Perimeter of $\mathbf{P}(z)$.
4. Mean distance between the centroid to the shape boundary.
5. Standard deviation of the distances between the centroid to the shape boundary.
6. 200 similarity invariant descriptors based on the Fourier transformation.
7. Major axis of the ellipse that approximates $\mathbf{P}(z)$ using the first two Fourier coefficients.
8. Minor axis of the ellipse that approximate $\mathbf{P}(z)$ using the first two Fourier coefficients.
9. The ratio of the major and minor.
10. Seven invariants calculated from the central moments of $\mathbf{P}(z)$.
11. Normalized feature of compactness of $\mathbf{P}(z)$.
12. Normalized feature of eccentricity of $\mathbf{P}(z)$.
13. Form factor of $\mathbf{P}(z)$.

In a series of T rounds, the algorithm repeatedly selects the weak classifier h_j with the lowest error under the training data. To do so, AdaBoost uses a weighted error function. The importance weight w_n for each example is updated in each round. The algorithm modifies the set of importance weights by increasing the weights of the most difficult training examples in each round. The optimal parameters (θ_j, p_j) for each weak classifier h_j are also computed using the weighed examples. As a result, a single feature can generate several weak classifiers with different parameters in the individual rounds of the AdaBoost algorithm.

The final strong classifier $\mathcal{H}$ is a weighted majority vote of the best T weak classifiers

$$\mathcal{H}(x) = \begin{cases} 1 \text{ if } \sum_{t=1}^{T} h_t(x)\alpha_t \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 \text{ otherwise,} \end{cases} \tag{5.4}$$

where the value of α_t is computed according to the weighted error rates of the individual weak classifiers. The full algorithm is given in Algorithm 5.1. In our system, the resulting strong classifier takes as input a single 360 degree laser range scan recorded by a robot and is able to determine whether the position from which the scan was taken belongs to the class *corridor* or not.

5.3 Estimating the Label of a Goal Location

The idea described in the previous section is well-suited to determine the type of place the robot is currently in given a 360 degree laser range scan. Even if the place to classify is not the current pose of the robot, one can simulated a laser range observation in the map of the environment and apply the classifier to the simulated scan. This works well for poses whose surroundings are completely known.

In the context of exploration, however, we are interested in classifying potential targets of the robot. Typically, target locations are located at the frontiers between known and unknown areas. This means that large neighboring areas have not been observed so far which makes it impossible to generate an appropriate observation taken from that location. As we will demonstrate in the experiments, classifying a place at a frontier with the approach presented in the previous section leads to high false classification rates of around 20%. In the following, we therefore introduce a HMM-based technique that takes into account spacial dependencies between nearby locations in order to obtain a lower error rate for places located at frontiers.

In our approach, we generate a potential target location for each group of frontier cells lying on the same frontier. This process is repeated for each frontier. As an example, the left image of Figure 5.3 depicts a potential target location extracted for the right-most frontier (the targets for the other two frontiers are not shown in that image).

Algorithm 5.1 The AdaBoost algorithm

Input: Input: set of examples $(x_1, y_1), \ldots, (x_N, y_N)$.

k = number of negatives examples
l = number of positive examples.
for $n = 1, \ldots, N$ **do**
  if $y_n = 1$ **then**
    $w_{1,n} = \frac{1}{l}$
  else
    $w_{1,n} = \frac{1}{k}$
  end if
end for
for $t = 1, \ldots, T$ **do**
  Normalize the weights $w_{t,n}$ so that $\sum_{n=1}^{N} w_{t,n} = 1$.
  for all features f_j **do**
    Train a weak classifier h'_j for the feature f_j.
    Compute the error ϵ'_j of a classifier h'_jaccording to

$$\epsilon'_j = \sum_{n=1}^{N} w_{t,n} \left| h'_j(x_n) - y_n \right|.$$

  end for
  Determine the weak classifier with the lowest error:

$$(h_t, \epsilon_t) = \underset{(h'_j, \epsilon'_j)}{\operatorname{argmin}} \epsilon'_j$$

  $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$
  for $n = 1, \ldots, N$ **do**
    $w_{t+1,n} = w_{t,n} \beta_t^{1-|h_t(x_n)-y_n|}$
  end for
  $\alpha_t = \log \frac{1}{\beta_t}$
end for
The final strong classifier is given by

$$\mathcal{H}(x) = \begin{cases} 1 \text{ if } \sum_{t=1}^{T} h_t(x)\alpha_t \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 \text{ otherwise,} \end{cases}$$

return $\mathcal{H}$

Due to the structure of environments made by humans, the semantic class does not change randomly between nearby poses. Therefore, it makes sense to consider smoothing or filtering between places located close together. To do so, we generate a short virtual trajectory to the desired goal location. We then simulate a laser range observation within the partially known map along the virtual trajectory. Whenever the ray-casting operation used to simulate a beam reaches an unknown cell in the grid map, the virtual sensor reports a maximum-range reading. We then apply a hidden Markov model (HMM) and maintain a posterior $Bel(L_x)$ about the type L_x of the place x the virtual sensor is currently at

$$Bel(L_x) = \eta p(o_x \mid L_x) \sum_{L_{x'}} p(L_x \mid L_{x'}) Bel(L_{x'}). \tag{5.5}$$

In this equation, o_x is the result of the classifier learned with AdaBoost for the place x and η is a normalizing constant ensuring that the left-hand side sums up to one over all semantic labels.

To implement this HMM, three components need to be known. First, we need to specify the observation model $p(o_x \mid L_x)$ which is the likelihood that the classification output is o_x given the actual class is L_x. The observation model is learned based on 5.000 observations, recorded at randomly chosen locations in different environments combined with the corresponding manually created ground truth labeling.

Second, we need to specify the transition model $p(L_x \mid L_{x'})$ which defines the probability that the virtual sensor moves from class $L_{x'}$ to class L_x. To determine this motion model, we evaluated typical trajectories obtained during exploration. We can directly compute $p(L_x \mid L_{x'})$ by counting the transitions between places on that trajectories. The correct labels were manually set.

Furthermore, we need to specify how the belief $Bel(L_{start})$ is initialized. In our current system, we choose a uniform distribution, which means that all classes (here *corridor* and *non-corridor*) have the same likelihood.

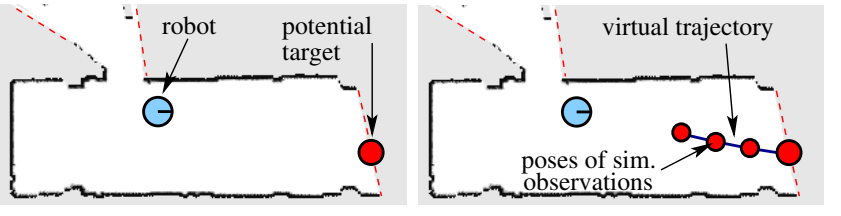


Fig. 5.3. This figure illustrates the generation of the virtual trajectory used for the HMM filtering. The left image depicts the current location of the robot, the frontiers (dashed lines), and a potential target location to be evaluated. To do so, the robot generates a virtual trajectory as shown in the right image and simulates observations at several positions located on the trajectory. These sequence of observations is used as the input of the HMM in order to obtain a more robust classification result.

Finally, we have to describe how the virtual trajectory is generated. The endpoint of the trajectory is the frontier cell to be classified. Since locations which have less unknown grid cells in their surroundings can typically be classified with a higher success rate, the other positions on that trajectory should be as far away from the unknown locations as possible. Therefore, we apply the euclidian distance transformation [99] with respect to unknown and occupied cells in the local area of the frontier. We then select the pose in the free space within that local area with the highest distance to unknown areas. An A* planner is used to generate the virtual trajectory to the target location. An illustrating example is depicted in Figure 5.3.

5.4 Using Semantic Place Information for Efficient Multi-Robot Exploration

In this section, we describe how to integrate the semantic information into the coordination technique presented in the previous chapter. As a result of that integration, the robots prefer to explore corridors first. In this way, they can identify more target locations in the beginning of the exploration run. As mentioned before, our observations is that the more unexplored target locations are known when assigning targets to robots, the faster the team can explore the environment. This is due to the fact that especially large teams of robots can be better distributed over the environment when more target locations are available.

The knowledge about the semantic labels is integrated into the utility function used to select the next target locations for the individual robots. The places which are supposed to provide several branchings to adjacent places are initialized with a high utility. In our current implementation, all corridor locations get a γ times higher initial utility (U_{init}) compared to all other potential target locations. In this way, the robots prefer targets in corridors and eventually make slight detours in order to explore them first. To determine the actual value of γ, we performed exploration runs in different environments with varying γ. We figured out that we obtained the best results using a γ-value of around 5. Algorithm 5.2 depicts the resulting coordination technique used in our current system (using the same notation as in Chapter 4).

Algorithm 5.2 Target assignment algorithm using semantic place labels.

1: Determine the set of frontier cells.
2: Compute for each robot i the cost V_f^i for reaching each frontier cell f.
3: Estimate for each frontier cell f the semantic labeling L_f (according to Section 5.3).
4: Set the utility U_f of all frontier cells f to $U_{init}(L_f, n)$ according to their semantic labeling L_f and the size n of the team (see text below).
5: **while** there is one robot left without a target point **do**
6: Determine a robot i and a frontier cell f which satisfy: $(i, f) = \text{argmax}_{(i',f')} \left(U_{f'} - V_{f'}^{i'} \right)$.
7: Reduce the utility of each target point f' in the visibility area according to $U_{f'} \leftarrow U_{f'} - P_{vis}(f, f')$.
8: **end while**

Our approach distributes the robots in a highly efficient manner over the environment and reduces the amount of redundant work by taking into account visibility constraints between targets and their semantic labels. The labels are used to focus the exploration on unexplored corridors, because they typically provide more branchings to adjacent rooms than other places. The high number of branchings from the places explored first results in a

higher average number of available target locations during exploration. This leads to a more balanced distribution of robots over the environment when doing the assignment. As we will demonstrate in the experiments, the integration of such semantic labels helps to reduce the overall exploration time of multi-robot exploration approaches for large robot teams.

Please note that for very small teams of robots, we do not achieve a reduction of the exploration time using our technique. This fact can be explained by considering the single-robot exploration scenario. In this case, it makes no sense to focus on exploring the corridors first, since the robot has to cover the overall environment with its sensors. Moving through the corridors first will in general lead to an increased trajectory length and in this way will increase the overall exploration time. We observed this effect for robots teams smaller than five robots.

To prevent a loss of performance compared to approaches which do not consider semantic place information for small robot teams, we trigger the influence of the semantic place information depending on the size of the team. We linearly decrease the influence γ for teams smaller than 10 robots. The linear interpolation of the influence of the semantic labels is encoded in the utility function $U_{init}(L_f, n)$ in Algorithm 5.2, where n denotes the number of robots.

5.5 Experimental Results

This section is designed to evaluate the improvements of our multi-robot coordination technique which makes use of semantic place information. Due to the high number of robots in the team, we evaluated our collaboration technique only in simulation experiments.

5.5.1 Performance Improvement Using Semantic Place Information

Fig. 5.4. Maps of the Fort Sam Huston hospital and the Intel Research Lab.

The first experiment has been carried out in the map of the Fort Sam Huston hospital which is depicted in the left image of Figure 5.4. This

environment contains a long horizontal corridor, vertical corridors, and several rooms adjacent to the corridors.

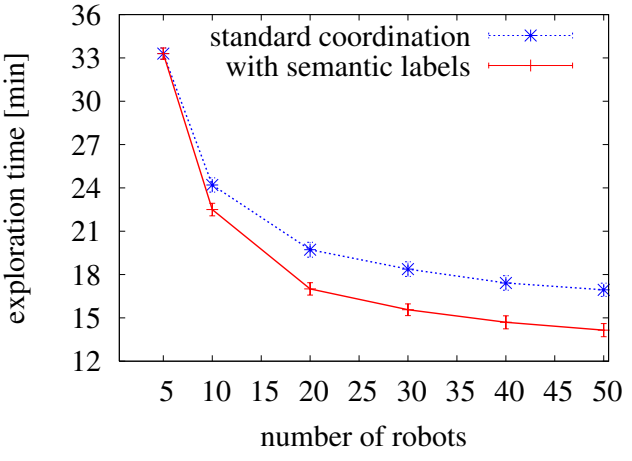


Fig. 5.5. Coordination results obtained in the Fort Sam Huston hospital map employing the coordination strategy with and without the use of semantic place labels.

We varied the size of the robot team from 5 to 50 robots and applied the coordination technique with and without taking into account semantic information about places. Figure 5.5 depicts the result of the exploration experiment by plotting the exploration time versus the number of robots. The error bars in that plot indicate the 5% confidence level. As can be seen, our technique significantly outperforms the collaboration scheme that does not consider the place information. This significant reduction of exploration time is due to the fact that the robots focus on exploring the corridors first. As a result, a big number of frontiers emerges due to typically numerous adjacent rooms. Especially in the context of large teams, this results in a better distribution of robots over the environments and thus speeds up the overall exploration process. This effect can be observed in Figure 5.6. The graphs plot the number of available target locations over time during an exploration task carried out using the Fort Sam Houston map. During the assignment process, most of the time the number of available target locations is higher compared to our previous approach. This leads to a better assignment of target locations to robots and as a result the amount of redundant work is reduced.

Furthermore, we observed a reduction of interferences between robots when they plan their paths through the environment. The interferences lead to a lower speed of the robots, since they often block their paths. Therefore, reducing the number of interferences allows the robots to accomplish their task faster. In our experiments, we observed a reduction of robot-robot interferences of up to 20%.

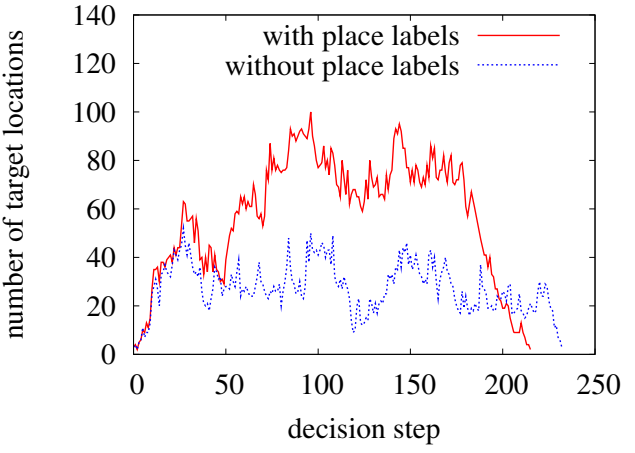


Fig. 5.6. The number of potential target locations at the different decision steps during exploration.

We performed similar experiments in different environments, like for example in the Intel Research Lab depicted in the right image of Figure 5.4. The result is comparable to the previous experiment and again the knowledge about the semantic categories of places allows the robots to complete the exploration task more efficiently. The actual evolution of the exploration time in this experiment is depicted in Figure 5.7. The same holds for experiments carried out using the floor plan of the DLR building shown in Figure 5.8.

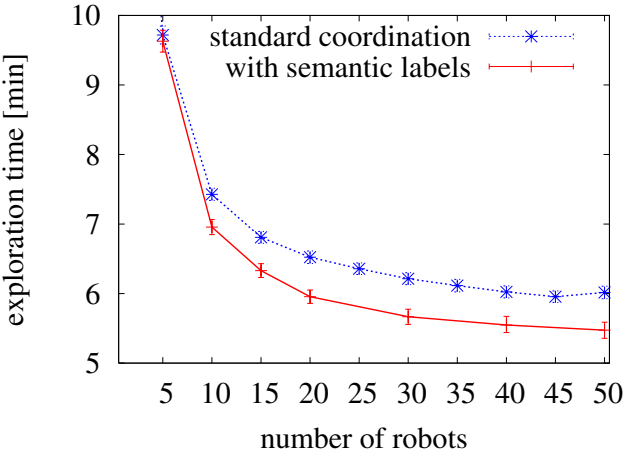


Fig. 5.7. Results obtained in the Intel Research Lab.

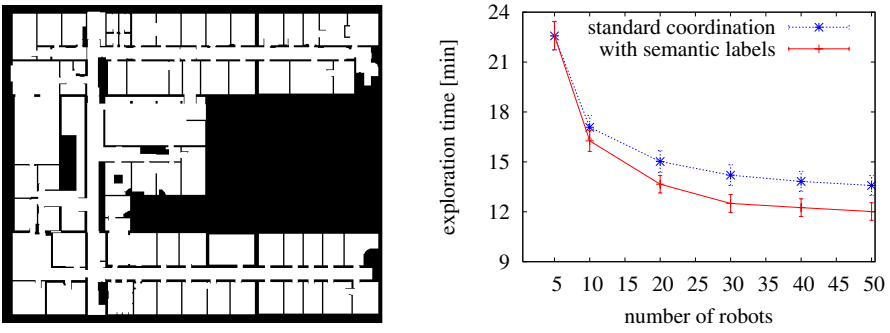


Fig. 5.8. Floor plan of the German Aerospace Center (DLR) and the corresponding results of our exploration system.

5.5.2 Influence of Noise in the Semantic Place Information

In the experiments presented above, we assumed that the robots are able to correctly classify the different target location into the semantic categories. This assumption, however, is typically not justified. In this experiment, we evaluate the performance of our approach for different classification error rates. We evaluated the exploration time for a classifier which randomly misclassified 5%, 10%, and 15% of the places. Figure 5.9 depicts a plot comparing the different error rates. As can be seen, even at a high error of 10%, our approach significantly outperforms the coordination technique that ignores the semantic information. When the error of the classification exceeds 15%, the exploration time is still reduced, although this result is not significant anymore.

5.5.3 Applying a Trained Classifier in New Environments

This experiment is designed to illustrate that it is possible to train a classifier in an environment and transfer it to a totally different one. Of course, the performance of the classifier decreases, however, we obtained promising result. Figure 5.10 shows two labeled maps. The one in the first row was labeled manually and used to learn the classifier using AdaBoost. For the environment depicted in the lower image, we simulated an observation for each grid cell and than used the trained classifier to label the positions. As can be seen, the spacial structures are quite different but the classification is good expect of a small areas which are wrongly classified. Large parts of the misclassified areas in this experiment are located at the ends of the corridors. This is mainly due to the fact that large parts of the area covered by scans recorded at these locations actually cover a corridor.

We then used this classification result to perform an exploration task. The results of this experiment are depicted in Figure 5.11. The figure plots the

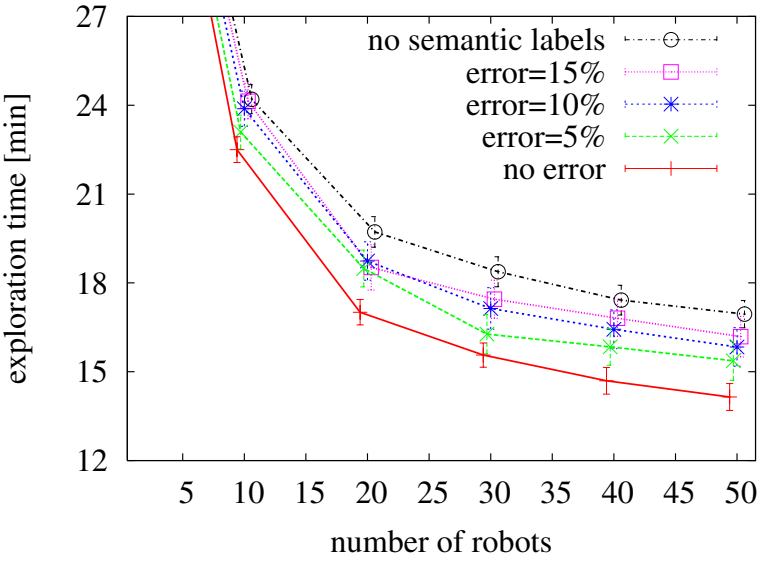


Fig. 5.9. Exploration results with wrongly labeled places.

time needed to explore the environment using our approach with the true labels, with the labels estimated by our classifier, and without using place information at all. As can be seen, there is only a small time overhead when using the estimated labels compared to the true ones. This indicates that even transferring such a classifier to unknown environments provides a speed-up in the context of multi-robot exploration.

5.5.4 Improvements of the HMM Filtering and Error Analysis of the Classifier

In this experiment, we want to analyze the actual error of our place classification system and illustrate the improvements of the HMM filtering. To do so, we labeled an environment, trained a corridor classifier using AdaBoost, and used a test set to evaluate the success rate. Whenever a single full 360 degree laser range scan was available, we obtained highly accurate classification results in different office environments. In this case, the error-rate was typically between 2% and 4%.

Figure 5.12 depicts the result of our classifier depending on the number of invalid readings caused by unknown grid cells close to frontiers. The x-axis shows the size of a continuous block of maximum range measurements (here with an angular resolution of the laser of 1 degree). As can be seen, if only half of the observations are available, the classification error rate is between 18% and 19%.

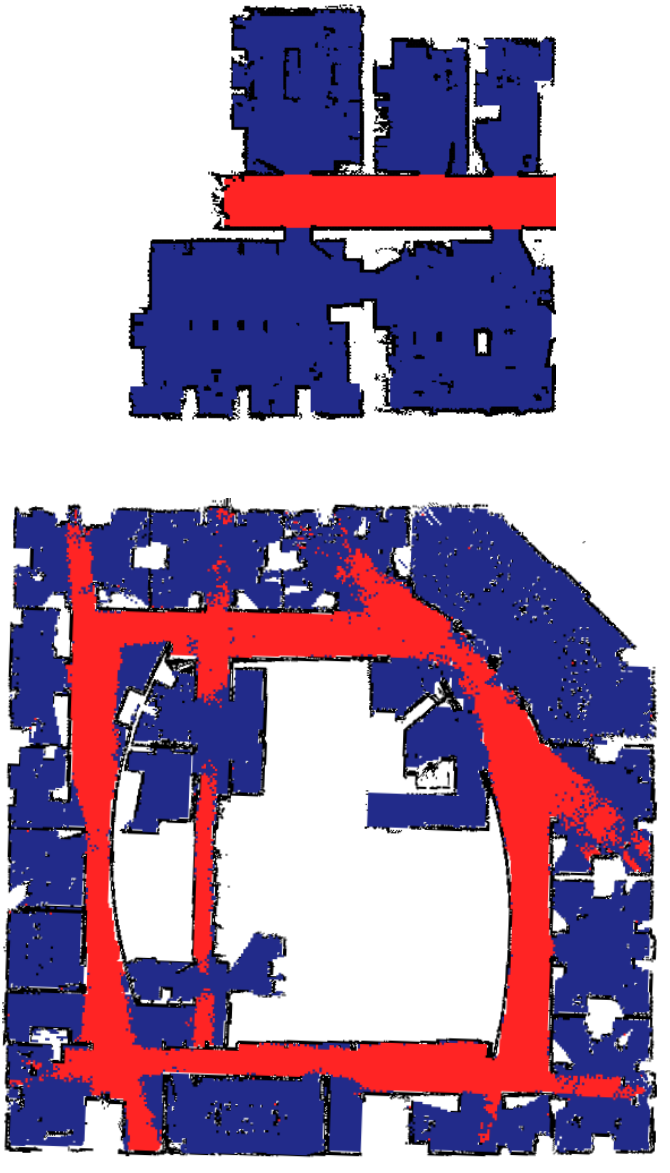


Fig. 5.10. The training examples for the classifier were trained in the map shown in the top image. In contrast to this, the lower image shows the resulting classification output. The classification for each place was performed based on a laser range scan simulated at the corresponding location in the map. As can be seen, even if the structure of the environment is significantly different, the classification output is reasonable. Red corresponds to corridor locations, blue to rooms.

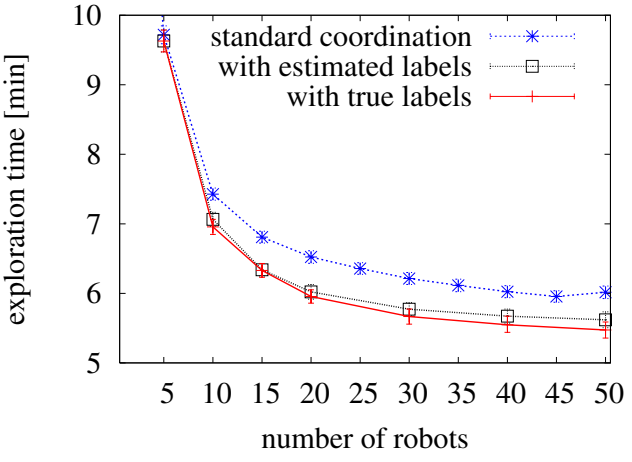


Fig. 5.11. Results obtained in the Intel Research Lab using the ground truth and the estimated semantic labels.

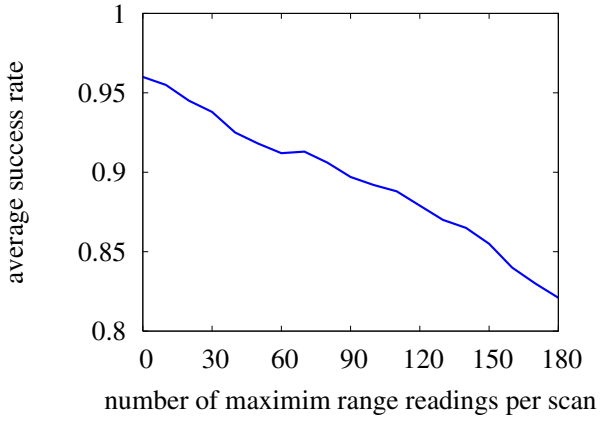


Fig. 5.12. This plot illustrates the classification performance of the standard classifier depending on how many consecutive beams of a 360 degree observation (1 degree angular resolution) are maximum range readings.

First, we determined the success rate of directly classifying frontier cells without using HMM filtering. In this case, the average classification rate was in average 81.2%. By considering the exploration speed-up depending on the classification rate depicted in Figure 5.9, such a high error rate is not sufficient to obtain an significant speed-up.

Second, we applied our HMM-based filtering approach that generates virtual trajectories towards frontiers and in this way incorporates the spatial

dependencies between the nearby locations. As a result, we obtained an average success rate of 92.8%. This is a good result considering that we obtained an average success rate in this scenario of 96.2% if all observations are perfectly known (see Figure 5.12). This fact illustrates that the HMM is an useful tool to improve the place labeling especially if not the full 360 degree range scan is available. It allows us to estimate the semantic labels with a comparably low error rate. In this way, our technique can be used to significantly speed up multi-robot exploration by considering semantic information about places in the environment.

In sum, our experiments demonstrate that semantic place information can significantly reduce the exploration time even under classification errors.

5.6 Related Work

In order to improve the navigation capabilities of a team of robots, we use semantic place information learned from sensor data. Several authors addressed the problem of classifying typical structures of indoor environments. For example, Koenig and Simmons [75] use a pre-programmed routine to detect doorways from range data. In [19], a virtual sensor is described which automatically segments the space into room and corridor regions, and computes a set of characteristic parameters for each region. The algorithm is incremental in the sense that it only maintains a local topological map of the space recently explored by the robot and generates information about each detected room whilst rooms are visited. Althaus and Christensen [3] use the Hough transform from sonar readings to detect two parallel lines which are considered to be part of a corridor. The detection of doorways is carried out using the gaps between these lines. With the detection of corridors and doorways, they construct a topological map for navigation and obstacle avoidance.

Some authors also apply learning techniques to localize the robot or to identify distinctive states in the environment. For example, Oore *et al.* [113] train a neural network to estimate the location of a mobile robot in its environment using the odometry information and ultrasound data. Kuipers and Beeson [81] apply different learning algorithms to learn topological maps of the environment. Additionally, Anguelov and colleagues [4, 5] apply the EM algorithm to cluster different types of objects from sequences of range data. In a recent work, Torralba *et al.* [151] use hidden Markov models for learning places from image data.

In our work, we apply a technique based on the place classification approach proposed in a joint work [95]. The idea is to use simple features extracted from laser range scans in order to train a set of classifiers using AdaBoost. In this way, it is possible to label a place given a single laser range observation. Furthermore, our filtering technique bears resemblance with the approach presented in a joint work with Rottmann *et al.* [125], in which a

hidden Markov model is applied to improve the classification result. In contrast to the work described here, we combine in [125] laser data and visual information to obtain more features and in this way are able to distinguish between more classes.

Different authors apply the AdaBoost algorithm to learn classifiers. Treptow *et al.* [152] use the AdaBoost algorithm to track a ball without color information in the context of RoboCup. Viola and Jones [156] presented a robust face detector using AdaBoost and single-value features. Their approach considers integral images in order to compute such features.

Our classifier used to label places can be seen as background knowledge about the environment. Fox *et al.* [45] presented a technique which aims to learn background knowledge in typical indoor environments and later on use them for map building. Their approach learns a Dirichlet prior over structural models from previously explored environments. The presented technique is applied to decide whether the robot is seeing a previously built portion of a map, or is exploring new territory. This can be especially useful if the pose information of the robots are affected by noise or they do not know their relative locations.

In the context of coordination techniques for multi-robot exploration, we would like refer the reader to Section 4.6 which discusses common approaches in detail. Due to the best of our knowledge, there is no work that investigates how semantic information about places in the environment can be used to optimize the collaboration behavior of a team of robots. The contribution of this chapter is an approach that estimates and explicitly uses semantic information in order to more efficiently spread the robots over the environment. This results in an more balanced target location assignment with less interferences between robots. As a result, the overall time needed to cover the whole environment with the robots' sensors can be significantly reduced.

5.7 Conclusion

In this chapter, we proposed a technique that takes into account semantic information about places in the context of coordinated multi-robot exploration. Since indoor environments are made by humans, they typically consist of structures like corridors and rooms. The knowledge about the type of place of a potential target location allows us to better distribute teams of robots over the environment and to reduce redundant work as well as the risk of interference between the robots. As a result, the overall exploration time can be reduced compared to collaboration approaches that ignore semantic place information. The semantic labels are determined by learning a classifier using AdaBoost in combination with an HMM to consider spacial dependencies.

Our approach has been implemented and tested in extensive simulation runs with up to 50 robots. Experiments presented in this chapter illustrate that a team of robots can complete their exploration mission in a significantly

shorter period of time using our approach. Furthermore, we believe that utilizing semantic information during exploration is not restricted to our exploration technique and that it can be easily integrated into other coordination approaches.

6

Efficient Techniques for Rao-Blackwellized Mapping

6.1 Introduction

So far, we focused on guiding robots through the environment in order to perceive it with their sensors. We assumed that the poses of the robots were known. This assumption, however, does not hold in real world situations. In the second part of this book, we therefore take into account the uncertainty about the pose of a mobile robot.

In this chapter, we focus on how to estimate the trajectory of a robot as well as the map of the environment given the perceived sensor data and the odometry information. In the literature, the mobile robot mapping problem under pose uncertainty is often referred to as the *simultaneous localization and mapping* (SLAM) or *concurrent mapping and localization* (CML) problem [28, 30, 37, 57, 59, 101, 104, 109, 145, 88]. SLAM is considered to be a complex problem because to localize itself a robot needs a consistent map and for acquiring the map the robot requires a good estimate of its location. This mutual dependency among the pose and the map estimates makes the SLAM problem hard and requires searching for a solution in a high-dimensional space.

Murphy, Doucet and colleagues [109, 30] introduced Rao-Blackwellized particle filters as an effective means to solve the simultaneous localization and mapping problem. The key idea of this approach is to first use a particle filter to estimate the trajectory of the robot. One then uses this trajectory estimate to compute a posterior about the map of the environment. The main problem of the Rao-Blackwellized approaches is their complexity, measured in terms of the number of particles required to build an accurate map. Reducing this quantity is one of the major challenges for this family of algorithms. Additionally, the resampling step is problematic since it can eliminate good state hypotheses. This effect is also known as the particle depletion problem [29, 155, 31].

In this work, we present two approaches to substantially increase the performance of a Rao-Blackwellized particle filter applied to solve the SLAM problem based on grid maps:

C. Stachniss: Robotic Mapping and Exploration, STAR 55, pp. 93–115.
springerlink.com

- A proposal distribution that considers the accuracy of the sensor of the robot and allows us to draw particles in an highly accurate manner.
- An adaptive resampling technique, which maintains a reasonable variety of particles and in this way enables the algorithm to learn an accurate map and to reduce the risk of particle depletion.

As explained in Chapter 2, the proposal distribution within a particle filter is used to draw the next generation of particles. In our approach, the proposal distribution is computed by evaluating the observation likelihood around a particle-dependent most likely pose obtained by a scan registration procedure. In this way, the last reading is taken into account while generating a new particle. This allows us to estimate the evolution of the system according to a more informed and thus more accurate model than the one obtained using only a scan-matching procedure with fixed covariance as done by Hähnel *et al.* [59]. The use of this refined model has two effects. The resulting map is more accurate because the current observation is taken into account when estimating the movement of the vehicle which yields a more accurate pose estimate. The reduced error additionally leads to a smaller number of particle required to represent the posterior. The second technique, the adaptive resampling strategy, allows us to perform a resampling step only when it is needed and in this way allows us to keep a reasonable particle diversity. This significantly reduces the risk of particle depletion. Our approach has been validated by a large set of experiments in indoor as well as in outdoor environments. In all experiments, our approach generated highly accurate metric maps. Additionally, the number of the required particles is around one order of magnitude smaller than with previous approaches.

This chapter is organized as follows. After explaining how a Rao-Blackwellized filter can be used to solve the SLAM problem, we describe our improved mapping technique in Section 6.3. Experiments carried out on real robots as well as in simulation are presented in Section 6.5. Section 6.4 then analyzes the complexity of the presented approach and finally Section 6.6 discusses related approaches.

6.2 The Concept of Rao-Blackwellized Mapping

Rao-Blackwellized particle filters for SLAM [109, 30] are used to estimate the posterior $p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1})$ about the trajectory $x_{1:t}$ of the robot and the map m given its observations $z_{1:t}$ and its odometry measurements $u_{1:t-1}$. The key idea of Rao-Blackwellized mapping is to separate the estimation of the trajectory from the map estimation process

$$
\begin{aligned}
& p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1}) \\
& \overset{\text{product rule}}{=} p(x_{1:t} \mid z_{1:t}, u_{1:t-1}) p(m \mid x_{1:t}, z_{1:t}, u_{1:t-1}) && (6.1) \\
& = p(x_{1:t} \mid z_{1:t}, u_{1:t-1}) p(m \mid x_{1:t}, z_{1:t}). && (6.2)
\end{aligned}
$$

where (6.2) is obtained from (6.1) by assuming that m is independent of the odometry measurements $u_{1:t-1}$ given the poses $x_{1:t}$ of the robot and the corresponding observations $z_{1:t}$.

This factorization, which is also called the Rao-Blackwellization, allows us to efficiently compute $p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1})$, since the posterior about the map $p(m \mid x_{1:t}, z_{1:t})$ can be computed analytically, given the knowledge of $x_{1:t}$ and $z_{1:t}$. Learning maps under given pose information is also called "mapping with known poses" (see Chapter 2).

To estimate the posterior $p(x_{1:t} \mid z_{1:t}, u_{1:t-1})$ about the potential trajectory, Rao-Blackwellized mapping uses a particle filter similar to Monte Carlo localization (MCL) [25]. In contrast to MCL, the Rao-Blackwellized particle filter for mapping maintains an *individual map* for each sample. Each map is built given the observations $z_{1:t}$ and the trajectory $x_{1:t}$ represented by the corresponding particle.

One of the most common particle filtering algorithms is the *sampling importance resampling* (SIR) filter. A Rao-Blackwellized SIR filter for mapping incrementally processes the observations and the odometry readings as they are available. This is achieved by updating a set of samples representing the posterior about the map and the trajectory of the vehicle. The process can be summarized by the following four steps:

1. **Sampling**: The next generation of particles is obtained from the current generation by sampling from a proposal distribution π.
2. **Importance Weighting**: An individual importance weight $w_t^{[i]}$ is assigned to each particle, according to

$$w_t^{[i]} = \frac{p(x_t^{[i]} \mid x_{1:t-1}^{[i]}, z_{1:t}, u_{1:t-1})}{\pi(x_t^{[i]} \mid x_{1:t-1}^{[i]}, z_{1:t}, u_{1:t-1})} w_{t-1}^{[i]}. \tag{6.3}$$

   The weights $w_t^{[i]}$ account for the fact that the proposal distribution π in general is not equal to the target distribution.
3. **Resampling**: Particles with a low importance weight are typically replaced by samples with a high weight. This step is necessary since only a finite number of particles is used to approximate a continuous distribution. Furthermore, resampling allows us to apply a particle filter in situations in which the true distribution differs from the proposal.
4. **Map Estimating**: The map of each particle is updated using "mapping with known poses."

An example for such a filter is illustrated in Figure 6.1. It depicts three particles with the individually estimated trajectories and the maps updated according to the estimated trajectory. In the depicted situation, the robot closed a loop and the different particles produced different maps. Particle 1 has a comparably accurate pose estimate, whereas the map of particle 3 shows big alignments errors. Therefore, particle 1 will get a higher importance weight compared to particle 3. The weight of particle 2 will be between the

weight of particle 1 and 3 because its alignment error is smaller than the one of particle 3 but bigger than the one of particle 1.

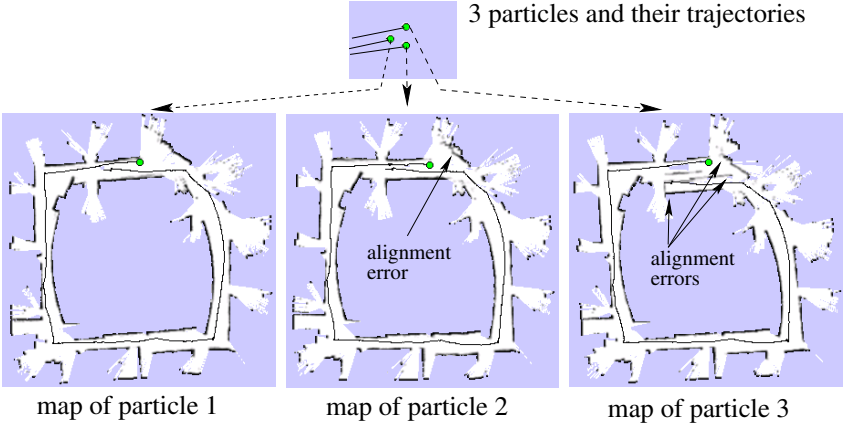


Fig. 6.1. Example for three particles used within Rao-Blackwellized mapping to represent $p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1})$. Each particle estimates the trajectory of the robot and maintains an individual map which is updated according to the estimated trajectory.

In the literature on particle filtering, several methods for computing improved proposal distributions and techniques for reducing the particle depletion problem have been described [29, 107, 118]. Our approach applies two concepts that have previously been identified as key pre-requisites for efficient particle filter implementations by Doucet [29]: the computation of improved proposal distributions and an adaptive resampling technique. Our idea of computing an improved proposal is similar to the technique applied by Montemerlo *et al.* [101] in FastSLAM-2. The major difference lies in the fact that we compute the proposal based on dense grid maps and not based on landmarks. To the best of our knowledge, adaptive resampling has never been investigated in the context of mapping with Rao-Blackwellized particle filters.

6.3 Improved Proposals and Selective Resampling

The generic algorithm specifies a framework for Rao-Blackwellized mapping but it leaves open how the proposal distribution is computed and when the resampling should be carried out. Throughout the remainder of this chapter, we describe a technique that computes an accurate proposal distribution and that adaptively determines when to resample.

As described in Section 6.2, one needs to draw samples from a proposal distribution π in the prediction step. In general, the proposal can be an arbitrary function (see Section 2.1 for further details). However, the more similar the proposal is to the target distribution, the better is the approximation of the next generation of samples. Therefore, π should approximate the true distribution as good as possible. Unfortunately, in the context of SLAM, a closed form of this posterior is not available. The samples are usually drawn from the transition model $p(x_t \mid x_{t-1}, u_{t-1})$ of the robot. Following the importance sampling principle, the weights $w_t^{[i]}$ can be computed as in the localization scenario (see Chapter 2)

$$w_t^{[i]} \propto p(z_t \mid m_{t-1}^{[i]}, x_t^{[i]}) w_{t-1}^{[i]}. \tag{6.4}$$

The motion model, however, is not the best choice for the proposal distribution. This fact has already been identified by Doucet *et al.* [31]. According to this work, the following equation is the optimal choice of the proposal distribution with respect to the variance of the particle weights and under the Markov assumption

$$p(x_t \mid m_{t-1}^{[i]}, x_{t-1}^{[i]}, z_t, u_{t-1}) = \frac{p(z_t \mid m_{t-1}^{[i]}, x_t) p(x_t \mid x_{t-1}^{[i]}, u_{t-1})}{\int p(z_t \mid m_{t-1}^{[i]}, x') p(x' \mid x_{t-1}^{[i]}, u_{t-1})\, dx'}. \tag{6.5}$$

We will now describe an efficient way for computing a per-particle proposal distribution, which uses the information of the most recent laser observation z_t.

6.3.1 Using Laser Range Data to Compute an Improved Proposal Distribution

In most particle filter applications [25, 104], the odometry motion model $p(x_t \mid x_{t-1}, u_{t-1})$ has been chosen as the proposal distribution. When modeling a mobile robot equipped with a laser range finder, this choice is suboptimal in most cases, since the accuracy of the laser range finder leads to extremely peaked likelihood functions. In such a situation, the likelihood function $p(z_t \mid m_{t-1}^{[i]}, x_t)$ dominates the product $p(z_t \mid m_{t-1}^{[i]}, x_t) p(x_t \mid x_{t-1}^{[i]}, u_{t-1})$ within the meaningful region $L^{[i]}$ of this distribution as illustrated in Figure 6.2.

In our current system, we therefore approximate $p(x_t \mid x_{t-1}^{[i]}, u_{t-1})$ by a constant k within the region $L^{[i]}$ given by

$$L^{[i]} = \left\{ x \mid p(z_t \mid m_{t-1}^{[i]}, x) > \epsilon \right\}. \tag{6.6}$$

Under this approximation, (6.5) turns into

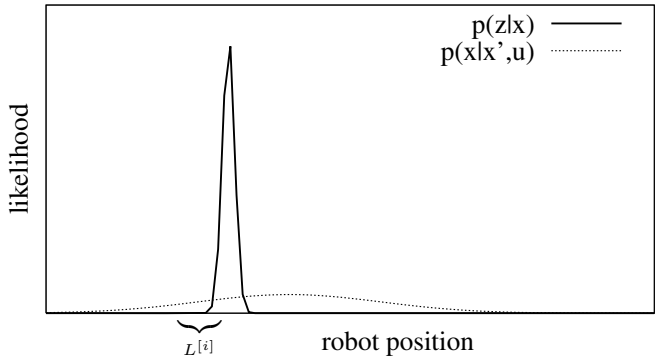


Fig. 6.2. The motion model for odometry as well as for laser data. Within the region $L^{[i]}$ the product of both functions is dominated by the observation likelihood. Accordingly, the model of the odometry error can safely be approximated by a constant value.

$$p(x_t \mid m_{t-1}^{[i]}, x_{t-1}^{[i]}, z_t, u_{t-1}) \simeq \frac{p(z_t \mid m_{t-1}^{[i]}, x_t)}{\int_{x' \in L^{[i]}} p(z_t \mid m_{t-1}^{[i]}, x')\, dx'}. \tag{6.7}$$

We furthermore have to specify the computation of the particle weights. For the importance weight $w_t^{[i]}$ of i-th particle, we obtain

$$w_t^{[i]} = \frac{p(x_{1:t}^{[i]} \mid z_{1:t}, u_{1:t-1})}{\pi(x_{1:t}^{[i]} \mid z_{1:t}, u_{1:t-1})} \tag{6.8}$$

$$\propto \frac{p(z_t \mid m_{t-1}^{[i]}, x_t^{[i]}) p(x_t^{[i]} \mid x_{t-1}^{[i]}, u_{t-1})}{\pi(x_t^{[i]} \mid m_{t-1}^{[i]}, x_{t-1}^{[i]}, z_t, u_{t-1})} \underbrace{\frac{p(x_{1:t-1}^{[i]} \mid z_{1:t-1}, u_{1:t-2})}{\pi(x_{1:t-1}^{[i]} \mid z_{1:t-1}, u_{1:t-2})}}_{w_{t-1}^{[i]}} \tag{6.9}$$

$$= w_{t-1}^{[i]} \frac{p(z_t \mid m_{t-1}^{[i]}, x_t^{[i]}) p(x_t^{[i]} \mid x_{t-1}^{[i]}, u_{t-1})}{\pi(x_t^{[i]} \mid m_{t-1}^{[i]}, x_{t-1}^{[i]}, z_t, u_{t-1})} \tag{6.10}$$

$$\overset{(6.7)}{\simeq} w_{t-1}^{[i]} \frac{p(z_t \mid m_{t-1}^{[i]}, x_t^{[i]}) p(x_t^{[i]} \mid x_{t-1}^{[i]}, u_{t-1})}{\frac{p(z_t | m_{t-1}^{[i]}, x_t^{[i]})}{\int_{x' \in L^{[i]}} p(z_t | m_{t-1}^{[i]}, x')\, dx'}} \tag{6.11}$$

$$\overset{\text{odometry const.}}{=} w_{t-1}^{[i]} \frac{p(z_t \mid m_{t-1}^{[i]}, x_t^{[i]}) k}{\frac{p(z_t | m_{t-1}^{[i]}, x_t^{[i]})}{\int_{x' \in L^{[i]}} p(z_t | m_{t-1}^{[i]}, x')\, dx'}} \tag{6.12}$$

This leads to

$$w_t^{[i]} = w_{t-1}^{[i]} k \int_{x' \in L^{[i]}} p(z_t \mid m_{t-1}^{[i]}, x')\, dx'. \tag{6.13}$$

Additionally, we locally approximate our proposal given in (6.7) around the maximum of the likelihood function by a Gaussian. This leads to the approximated proposal

$$p(x_t \mid m_{t-1}^{[i]}, x_{t-1}^{[i]}, z_t, u_{t-1}) \simeq p_{\mathcal{N}}(z_t \mid m_{t-1}^{[i]}, x_t). \tag{6.14}$$

where $p_{\mathcal{N}}$ refers to the Gaussian approximation of p. With this approximation, we obtain a closed form which is suitable for efficient sampling. For each particle i, the parameters $\mu_t^{[i]}$ and $\Sigma_t^{[i]}$ can be determined by evaluating the likelihood function for a set of points $\{x_j\}$ sampled around the corresponding local maximum found by the scan-matching process:

$$\mu_t^{[i]} = \frac{1}{\eta} \sum_{j=1}^{K} x_j p(z_t \mid m_{t-1}^{[i]}, x_j) \tag{6.15}$$

$$\Sigma_t^{[i]} = \frac{1}{\eta} \sum_{j=1}^{K} p(z_t \mid m_{t-1}^{[i]}, x_j) p(x_j - \mu_t^{[i]})(x_j - \mu_t^{[i]})^T \tag{6.16}$$

where

$$\eta = \sum_{j=1}^{K} p(z_t \mid m_{t-1}^{[i]}, x_j) \tag{6.17}$$

is a normalizer.

Observe that the computation of $\mu_t^{[i]}$ and $\Sigma_t^{[i]}$ as well as the scan-matching process are carried out for each particle individually. In our current system, we apply a scan-matching routine similar to that of Hähnel *et al.* [61]. The sampled points $\{x_j\}$ are chosen to cover an area dependent on the uncertainty of most recent odometry information

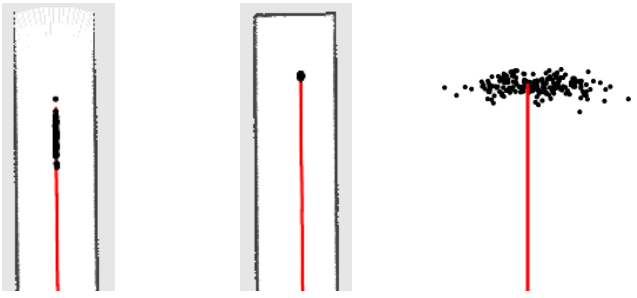
$$x_j \in \{x_t \mid p(x_t \mid x_{t-1}, u_{t-1}) > \epsilon\}. \tag{6.18}$$

By assuming that the Gaussian approximation of the observation likelihood is close to its real value (which is actually often the case) and by considering sampled points in $L^{[i]}$, the weights can be expressed by

$$w_t^{[i]} \propto w_{t-1}^{[i]} \int_{x' \in L^{[i]}} p(z_t \mid m_{t-1}^{[i]}, x')\, dx' \tag{6.19}$$

$$\overset{\text{use points } x_j \in L^{[i]}}{\simeq} w_{t-1}^{[i]} \sum_{j=1}^{K} p(z_t \mid m_{t-1}^{[i]}, x_j) \tag{6.20}$$

$$\overset{(6.17)}{=} w_{t-1}^{[i]} \eta, \tag{6.21}$$



(a) (b) (c)

Fig. 6.3. Particle distributions typically observed during mapping. In an open corridor, the particles distributes along the corridor (a). In a dead end corridor, the uncertainty is small in all dimensions (b). In a featureless open space the proposal distribution is the raw odometry motion model (c). The trajectory of the robot is depicted by the red line.

where η in (6.21) corresponds to the normalizer as given in (6.17).

The computations presented in this section allow us to determine the parameters of a Gaussian proposal distribution for each particle individually. The proposal takes into account the most recent laser observation and at the same time allows us efficient sampling. The resulting densities have a lower uncertainty than in the situation in which the odometry motion model is utilized. To illustrate this fact, Figure 6.3 depicts typical particle distribution obtained with our approach. In case of a straight featureless corridor, the samples are typically spread along the main direction of the corridor as depicted in Figure 6.3 (a). Figure 6.3 (b) illustrates the robot reaching the end of such a corridor. As can be seen, the uncertainty in the direction of the corridor decreases and all samples are centered around a single point. Figure 6.3 (c) shows how the particle spread out when they are draw from the odometry motion model.

During filtering, it can happen that the scan-matching process fails because of poor observations or a small overlapping area among the current scan and the previously computed map. In the case the scan-matcher reports an error, the raw motion model of the robot is used as a proposal. Such a situation in which the laser observation does not provide any information is depicted in Figure 6.3 (c). However, we observed that these kind of situations occur rarely in real datasets (see also Section 6.5.4 in the experiments of this chapter).

6.3.2 Selective Resampling

A further aspect that has a major influence on the performance of a particle filter is the resampling step. During resampling, the particles with a low importance weight $w_t^{[i]}$ are typically replaced by samples with a high weight. On

the one hand, resampling is necessary since only a finite number of particles are used. On the other hand, the resampling step can delete good samples from the sample set, causing particle depletion. In the context of map building, this is critical especially in the context of nested loops. During mapping an inner loop, hypotheses that are not necessarily the ones with the highest weight are often needed later on to correctly close an outer loop. Accordingly, it is important to find a criterion when to perform a resampling step.

Lui [90] introduced the so-called *effective number of particles* or *effective sample size* to estimate how well the current particle set represents the true posterior. This quantity is defined as

$$N_{\mathit{eff}} = \frac{1}{\sum_{i=1}^{N} \left(w_t^{[i]} \right)^2}, \tag{6.22}$$

where the weights $w_t^{[i]}$ are supposed to be normalized.

The intuition behind N_{eff} is as follows. If the samples were drawn from the true posterior, the importance weights of the samples would be equal to each other, due to the importance sampling principle. The worse the approximation, the higher the variance of the importance weights. N_{eff} can be regarded as a measure for the dispersion of the importance weights. Thus, it is a useful measure to evaluate how well the particle set approximates the true posterior. N_{eff} takes values between 1 and the number N of particles. Whenever the weights are equally distributed, its value is N. In case all the probability mass is concentrated in a single sample only, its value is 1. N_{eff} can be used to determine whether or not a resampling should be carried out. Whenever its value is high, resampling is typically not required since the approximation of the target distribution is good. We resample each time N_{eff} drops below a certain threshold. In our current implementation, this threshold was set to $N/2$. In extensive experiments, we found that this approach substantially reduces the risk of replacing good particles, because the number of resampling operations is reduced and resampling operations are only performed when needed.

6.4 Complexity

This section discusses the complexity of the presented approach to Rao-Blackwellized mapping using grid maps. Since our approach uses a particle filter to represent the joint posterior about the map and the trajectory, the number N of samples is the central quantity. To compute the proposal distribution for a single particle, our approach samples around the most likely position reported by the scan matcher. This sampling step is performed a constant number of K times for each sample and there is no dependency between the particles when computing the proposal.

The most recent observation which is used to compute $\mu^{[i]}$ and $\Sigma^{[i]}$, see (6.15) and (6.16), covers only an local area in the environment. Additionally,

the area of the sampled points is bounded by the odometry error. Since the computation needs to be done for each sample, the complexity of this computations depends only on the number N of particles. The same holds for the update of the individual maps associated to each particles.

The computation of the particle weights is done by computing the likelihood of the observation z_t according to (6.8). Again this leads only to a complexity linear in the number of particles.

During a resampling action, the information associated to a particle needs to be copied. In the worst case, $N-1$ samples are replaced by a single particle. In our current system, each particle stores and maintains its own grid map. To duplicate a particle, we therefore have to copy the whole map. As a result, a resampling action introduces a complexity of $O(NM)$, where M is the size of a grid map. However, the size of the environment in which the robot moves is typically limited. Furthermore, using our adaptive resampling technique, only a few resamplings are required during mapping. To decide whether or not a resampling is needed, the effective number of particles, see (6.22), needs to be taken into account. The computation of this quantity introduces a linear complexity in N.

Table 6.1 depicts the complexity of the individual operations. As a result, if no resampling operation is required, the overall complexity for integrating a single observation depends only linearly on the number of particles. If a resampling is required, the additional factor M which represents the size of the map is introduced and leads to a complexity of $O(NM)$.

Table 6.1. Complexity of the different operations for integrating one observation.

Operation	**Complexity**
Computation of the proposal distribution	$O(N)$
Update of the grid map	$O(N)$
Computation of the weights	$O(N)$
Test if resampling is required	$O(N)$
Resampling	$O(NM)$

6.5 Experimental Results

The approach described above has been implemented and tested using real robots and datasets gathered with real robots. Our implementation runs online on several platforms like ActivMedia Pioneer 2 AT, Pioneer 2 DX-8, and iRobot B21r robots equipped with a SICK LMS and PLS laser range finders (see Figure 6.4). The experiments carried out in a variety of environments have shown the effectiveness of our approach in indoor and outdoor

Fig. 6.4. Different types of robot used to acquire real robot data used for mapping (ActivMedia Pioneer 2 AT, Pioneer 2 DX-8, and an iRobot B21r).

environments. The quality of the resulting maps is extremely good, allowing in some cases to generate a map with a resolution of 1 cm, without observing considerable inconsistencies. Even in big real world datasets covering an area of approximately 250 m by 250 m, our approach never required more than 80 particles to build accurate maps. Except of the MIT dataset (see below), 30 particles where sufficient to build high quality maps of different environments. In this section, we discuss the behavior of the filter in different real world environments. Furthermore, we give a quantitative analysis of the performance of the presented approach.

Note that all corrected datasets presented here as well as the maps as high resolution images are available on the Internet [140]. We also provide a set of animations showing the evolution of the different trajectory hypotheses during mapping. Furthermore, an efficient open-source implementation of our mapping system is available at the OpenSLAM.org repository [141].

6.5.1 Mapping Results

The datasets discussed here have been recorded at the Intel Research Lab in Seattle, at the Killian Court at MIT, and on the campus at the University of Freiburg. The maps of these environments are depicted in Figures 6.5, 6.6, and 6.7.

Intel Research Lab

The Intel Research Lab is depicted in the left image of Figure 6.5 and has a size of 28 m by 28 m. The dataset has been recorded with a Pioneer 2 robot equipped with a SICK sensor. To successfully correct this dataset, our algorithm needed only 15 particles. As can be seen in the right image of Figure 6.5, the quality of the final map is so high that the map can be magnified up to a resolution of 1 cm without showing any significant errors or inconsistencies.

Freiburg Campus

The second dataset has been recorded outdoors at the Freiburg campus. Our system needs 30 particles to produce a good quality map such as the one

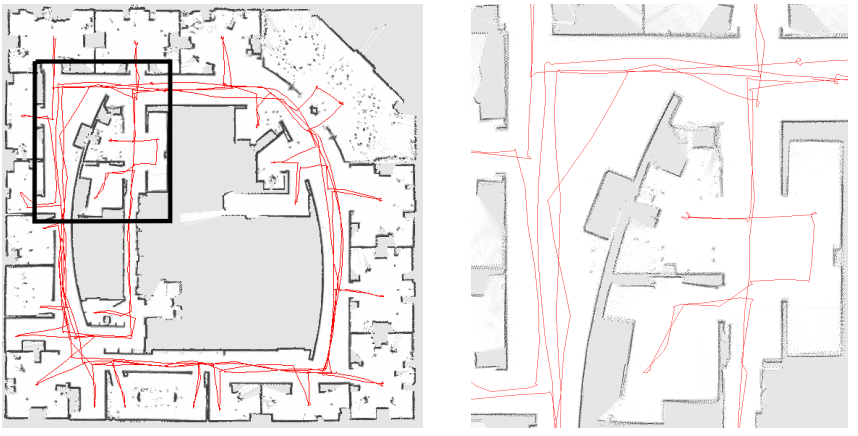


Fig. 6.5. The Intel Research Lab: The robot starts in the upper part of the circular corridor, and runs several times around the loop, before entering the rooms. The left image depicts the resulting map generated with 15 particles. The right image shows a magnified view with a grid resolution of 1 cm to illustrate the accuracy of the map in the loop closure point.

shown in Figure 6.6. Note that this environment partly violates the assumptions that the environment is planar. Additionally, there were objects like bushes and grass which are hard to be mapped with a laser range finder. Furthermore, there were moving objects like cars and people. Despite the resulting spurious measurements, our algorithm was able to generate an accurate map. Note that no GPS, compass, or IMU information is used in all our experiments.

MIT Killian Court

The third experiment was performed with a dataset acquired at the MIT Killian court and the resulting map is depicted in Figure 6.7. This dataset is extremely challenging since it contains several nested loops, which can cause a Rao-Blackwellized particle filter to fail due to particle depletion. Furthermore, there where people walking in front of the robot while it was moving through a nearly featureless corridor.

Using this dataset, our selective resampling procedure turned out to be extremely important. A consistent and topologically correct map can be generated with 60 particles. However, the resulting maps sometimes show artificial double walls. By employing 80 particles it is possible to achieve high quality maps. To give an impression about the size of this dataset, Figure 6.8 provides a satellite view showing the MIT campus around the Killian Court as well as the learned map on top of the satellite image.

Fig. 6.6. The Freiburg campus: The robot first runs around the external perimeter in order to close the outer loop. Afterwards, the internal parts of the campus are visited. The overall trajectory has a length of 1.75 km and covers an area of approximately 250 m by 250 m. The depicted map was generated using 30 particles.

6.5.2 Quantitative Results

In order to measure the improvement in terms of the number of particles, we compared the performance of our system using the informed proposal distribution to the approach done by Hähnel *et al.* [59]. Table 6.2 summarizes the number of particles needed by both RBPFs for providing a topologically correct map in at least 60% of all runs of our algorithm (initialized with different random seeds).

It turns out that in all of the cases, the number of particles required by our approach was approximately one order of magnitude smaller than the one required by the other approach. Moreover, the resulting maps are better due

Table 6.2. The number of particles needed by our algorithm compared to the approach of Hähnel *et al.* [59].

Proposal Distribution	Intel	MIT	Freiburg
our approach	8	60	20
approach of [59]	40	400	400

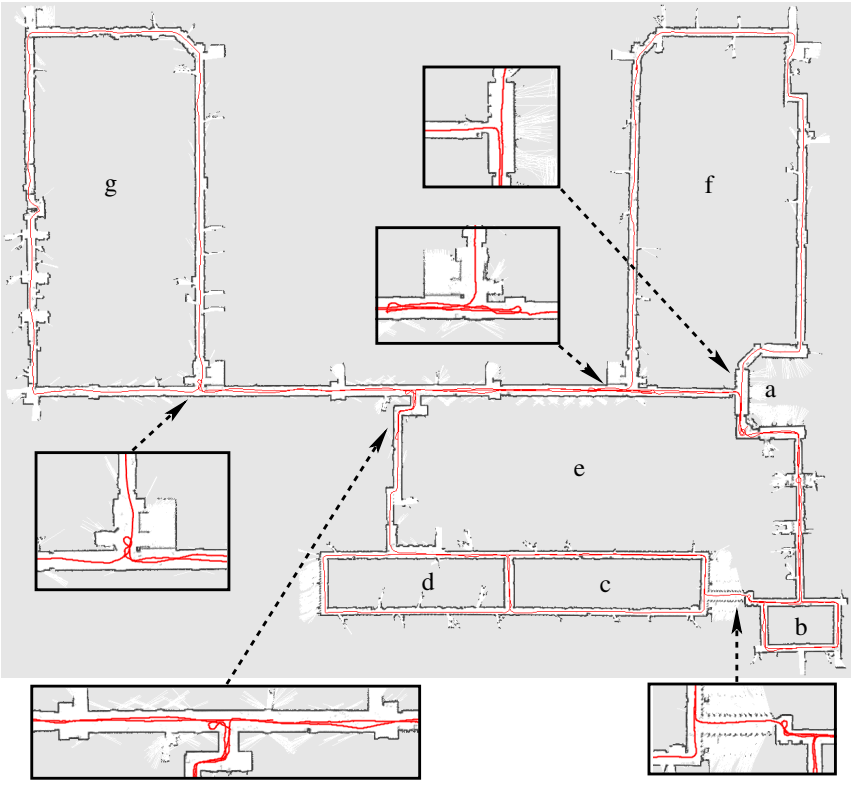


Fig. 6.7. The MIT Killian Court: The robot starts from the point labeled *a* and then traverses the first loop labeled *b*. It then moves through the loops labeled *c*, *d* and moves back to the place labeled *a* and the loop labeled *b*. It the visits the two big loops labeled *f* and *g*. The environment has a size of 250 m by 215 m and the robot traveled 1.9 km. The depicted map has been generated with 80 particles. The rectangles show magnifications of several parts of the map.

to our improved sampling process that takes into account the most recent sensor reading.

Figure 6.9 summarizes results about the success ratio of our algorithm in the environments considered here. The plots show the percentage of correctly generated maps, depending on the number of particles used. The binary decision if a run was successful or not was done by manual inspection of the resulting map. As a measure of success, we used the topological correctness. Map classified as incorrect typically showed double walls or corridors and/or wrongly aligned corridors.

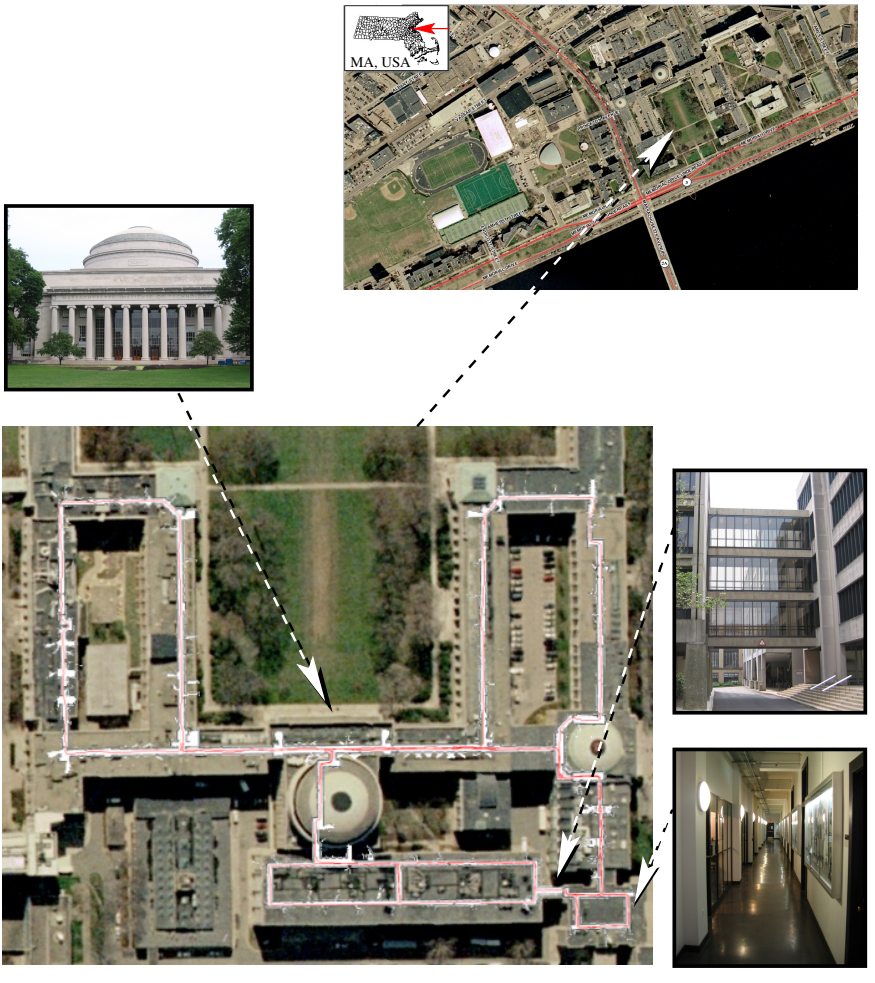


Fig. 6.8. The MIT Killian Court from a satellite perspective. The corridors plotted on top of the satellite view are the result of our mapping algorithm. Satellite image source: Massachusetts Geographic Information System (MassGIS).

6.5.3 Effects of Improved Proposals and Adaptive Resampling

The increased performance of our approach is due to the interplay of two factors, namely the improved proposal distribution, which allows to generate samples with an high likelihood, and the adaptive resampling controlled by monitoring N_{eff}. For proposals that do not consider the whole input history, it has been proven that N_{eff} can only decrease (stochastically) over time [29]. Only after a resampling operation does N_{eff} recover its maximum value. It is

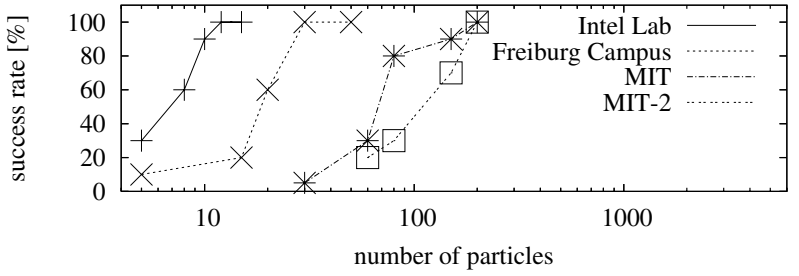


Fig. 6.9. Success rate of our algorithm in different environments depending on the number of particles. The binary decision if a run was successful or not was done by manual inspection of the resulting map. Each success rate in the plot was determined using 20 runs. For the experiment MIT-2, we disabled the adaptive resampling while correction the MIT dataset.

important to notice that the behavior of N_{eff} depends on the proposal: the worse the proposal, the faster N_{eff} drops.

In our experiments, we found that the evolution of N_{eff} using our proposal distribution shows three different behaviors depending on the information obtained from the robot's sensor. Whenever the robot moves through unknown terrain, N_{eff} typically drops slowly. This is because the proposal distribution becomes less peaked and the likelihoods of observations differ only slightly. The second behavior can be observed when the robot moves through a known area. In this case, each particle keeps localized within its own map due to the improved proposal distribution and the weights are more or less equal. This results in a constant evolution of N_{eff}. Finally, when closing a loop, some particles are correctly aligned with their map while others are not. The correct particles have a high weight, while the wrong ones have a low weight. Thus the variance of the importance weights increases and N_{eff} drops substantially. This behavior is illustrated in Figure 6.10.

Accordingly, our resampling criterion based on N_{eff} typically forces a resampling action when the robot is closing a loop. In most cases, the resampling is avoided which results in keeping the necessary variety of different hypotheses in the particle set. To analyze this, we performed an experiment in which we compared the success rate of our algorithm to that of a particle filter which resamples at every step. The experiment was carried out based on the MIT Killian Court dataset. As Figure 6.9 illustrates, our approach more often converged to the correct solution (MIT curve) for the MIT dataset compared to the particle filter with the same number of particles and a fixed resampling strategy (MIT-2 curve).

To give a more detailed impression about the accuracy of our new mapping technique, Figure 6.11 and 6.12 depict a collection of maps learned from commonly used and freely available real robot datasets [65]. The datasets

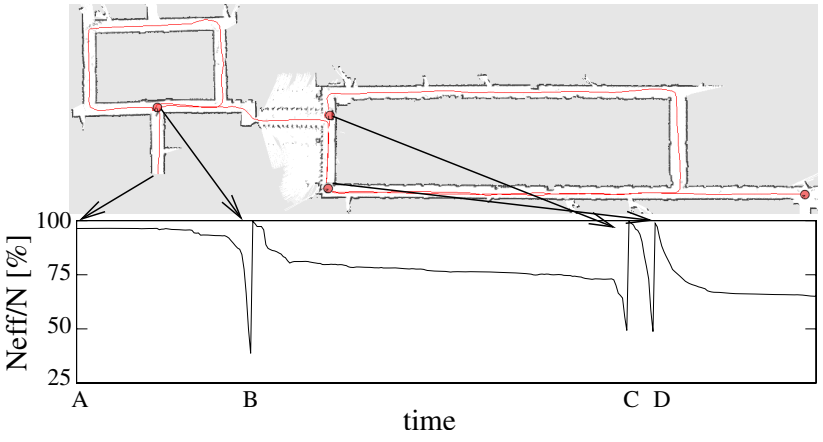


Fig. 6.10. The graph plots the evolution of the N_{eff} function over time during an experiment in the environment shown in the right image. At time B the robot closes the small loop. At time C and D resampling actions are carried after the robots closes the big loop.

used to build the maps shown in Figure 6.11 have been recorded at the MIT Computer Science and AI Lab, at the University of Washington, at Belgioioso, and at the University of Freiburg. Figure 6.12 depicts maps from the Bruceton mine, the University of Texas, and the Acapulco Convention Center. Each map was built using 30 particles to represent the posterior about the map of the environment and the trajectory of the vehicle.

6.5.4 Situations in Which the Scan-Matcher Fails

As reported above, it can happen that the scan-matcher is unable to find a good pose estimate based on the laser range data. In this case, we sample from the raw odometry model to create the next generation of particles. In most tested indoor dataset, however, such a situation never occurred at all. Only in the MIT dataset, this effect was observed once due to a person walking directly in front of the robot while the robot was moving though a corridor that mainly consists of glass panes. A picture of that glass corridor can be found in Figure 6.8.

In outdoor datasets, such a situation can occur if the robot moves through large open spaces and therefore the laser range finder mainly reports maximum range readings. During mapping the Freiburg campus, the scan-matcher also reported such an error at one point. In this particular situation, the robot entered the parking area (in the upper part of the map, compare Figure 6.6). On that day, all cars were removed from the parking area due to construction work. As a result, no cars or other objects caused reflections of the laser beams and most parts of the scan consisted of maximum range readings. In

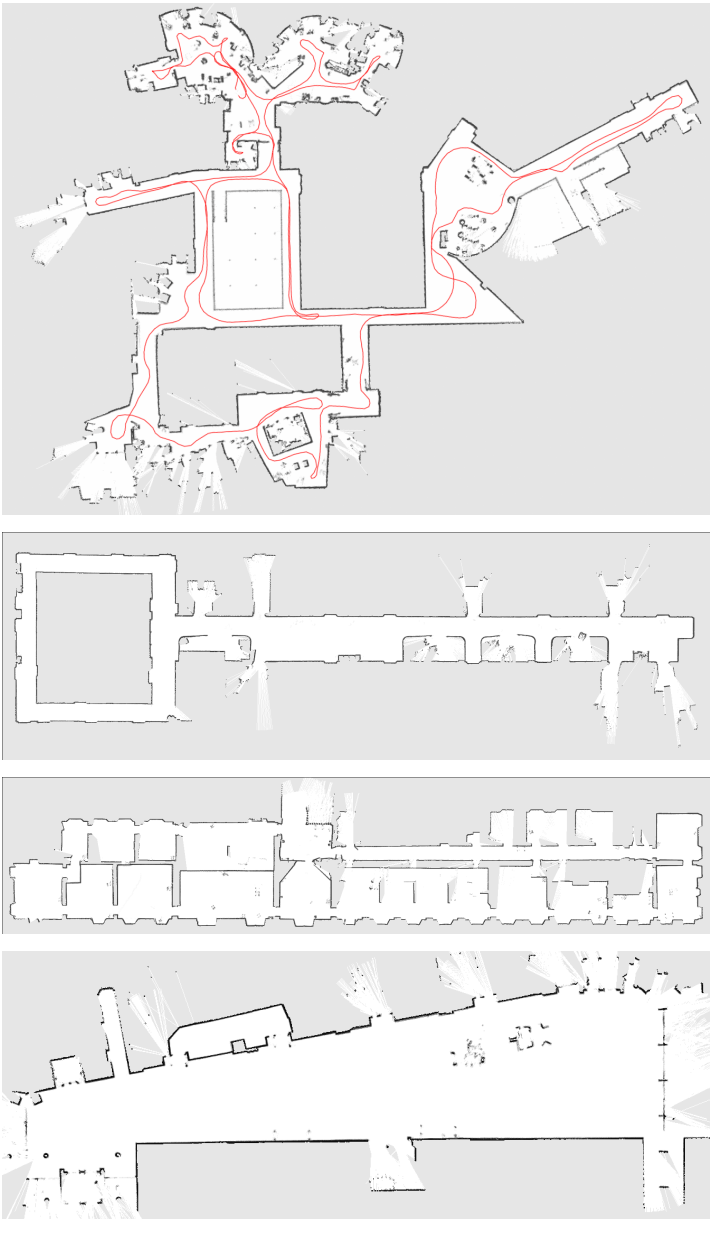


Fig. 6.11. Maps of the MIT Computer Science and AI Lab (showing also the trajectory of the robot), of the 4th floor of the Sieg Hall at the University of Washington, of the Belgioioso building, and of building 101 at the University of Freiburg.

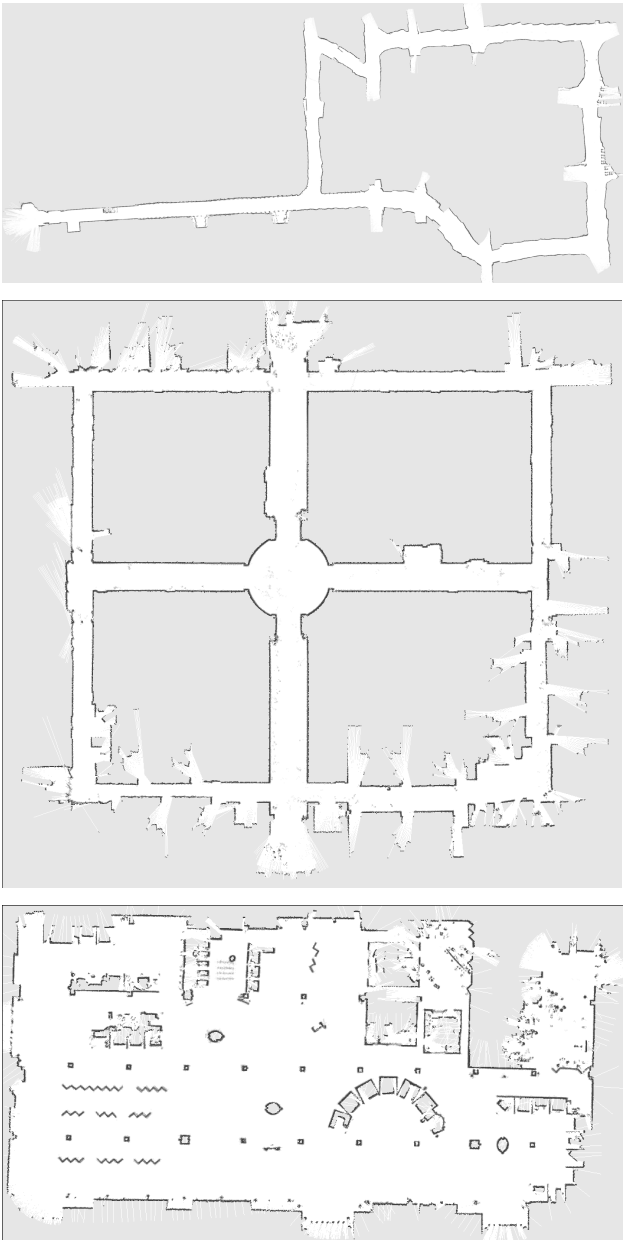


Fig. 6.12. Maps of the Bruceton mine, of the ACES building at University of Texas, and of the Acapulco Convention Center.

such a situation, the odometry information provides the best pose estimate and this information is used by our mapping system to predict the motion of the vehicle.

6.5.5 Computational Cost

In this last experiment, we analyze the memory and computational resources needed by our mapping system. We used a standard PC with a 2.8 GHz processor. We recorded the average memory usage and execution time using the default parameters that allows our algorithm to learn correct maps for all real world datasets provided to us. In this setting, 30 particles are used to represent the posterior about the map and the trajectory. A new observation which consists of a full laser range scan is integrated whenever the robot moved more than 0.5 m or rotated more than 25 degree. The Intel Research Lab dataset (see Figure 6.5) contains odometry and laser range readings which have been recorded over 45 min. Our implementation required around 200 MB of memory to store all the data using a map with a size of approx. 40 m by 40 m and a grid resolution of 5 cm. The overall time to correct the log file using our software was less than 30 min. This means that the time to record a log file is around 1.5 times longer than the time to correct the log file. Table 6.3 depicts the average execution time for the individual operations.

Table 6.3. Average execution time using a standard PC.

Operation	**Average Execution Time**
Computation of the proposal distribution, the weights, and the map update	1910 ms
Test if resampling is required	41 ms
Resampling	244 ms

6.6 Related Work

Mapping techniques for mobile robots can be roughly classified according to the map representation and the underlying estimation technique. One popular map representation are occupancy grid maps. Whereas such grid-based approaches typically require a lot of memory resources, they do not require a predefined feature extractor and provide detailed representations. Feature-based representations are attractive because of their compactness. However, they rely on feature extractors, which assumes that some structures in the environments are known in advance.

The estimation algorithms can be roughly classified according to their underlying basic principle. The most popular approaches are extended Kalman filters (EKFs), maximum likelihood techniques, sparse extended information filters (SEIFs), least square error minimization approaches, smoothing techniques, and Rao-Blackwellized particle filters (RBPFs). The effectiveness of the EKF approaches comes from the fact that they estimate a fully correlated posterior about landmark maps and robot poses [139, 87]. Their weakness lies in the strong assumptions that have to be made on both, the robot motion model and the sensor noise. Moreover, the landmarks are assumed to be uniquely identifiable. There exist techniques [111] to deal with unknown data association in the SLAM context, however, if certain assumptions are violated, the filter is likely to diverge [46]. Similar observations have been reported by Julier *et al.* [69] as well as by Uhlmann [153].

A popular least square error minimization algorithm computes the map given the history of sensor readings by constructing a network of relations that represents the spatial constraints among the poses of the robot [92]. Gutmann and Konolige [57] proposed an effective way for constructing such a network and for detecting loop closures while running an incremental estimation algorithm. When a loop closure is detected, a global optimization on the relation network is performed. Similar approaches use relaxation [32, 47] in order to find configurations that reduce the overall least square error in the network of relations between poses.

Hähnel *et al.* [60], proposed an approach which is able to track several trajectory and map hypotheses using an association tree. It expands always the best node in that tree. As a result, it switches to a different hypotheses as soon as the current one seems to lead to an inconsistent map. However, the necessary expansions of this tree can prevent the approach from being feasible for real-time operation. Furthermore, it is somewhat unclear, how the different hypotheses can be created autonomously.

Thrun *et al.* [150] proposed a method to correct the poses of robots based on the inverse of the covariance matrix. The advantage of sparse extended information filters (SEIFs) is that they make use of the approximative sparsity of the information matrix and in this way can perform predictions and updates in constant time. Eustice *et al.* [39] as well as Walter *et al.* [157] presented a techniques to more accurately compute the error-bounds within the SEIF framework and in this way reduces the risk of becoming overly confident. Paskin [116] presented a solution to the SLAM problem using thin junction trees. In this way, he is able to reduce the complexity compared to the EKF approaches since thin junction trees provide a linear time filtering operation.

Recently, Dellaert proposed a smoothing method called square root smoothing and mapping [24]. It has several advantages compared to EKF since it better covers the non-linearities and is faster to compute. In contrast to SEIFs, it furthermore provides an exactly sparse factorization of the information matrix.

Bosse *et al.* [14] describe a generic framework for SLAM in large-scale environments. They use a graph structure of local maps with relative coordinate frames and always represent the uncertainty with respect to a local frame. In this way, they are able to reduce the complexity of the overall problem and reduce the influence of linearization errors.

Modayil *et al.* [100] presented a technique which combines metrical SLAM with topological SLAM. The topology is utilized to solve the loop-closing problem and metric information is used to build up local structures. Similar ideas have been realized by Lisien *et al.* [89], which introduce a hierarchical map in the context of SLAM.

In a work by Murphy [109], Rao-Blackwellized particle filters (RBPF) have been introduced as an effective means to solve the SLAM problem. Each particle in a RBPF represents a possible robot trajectory and a map. The framework has been subsequently extended for approaching the SLAM problem with landmark maps [104, 103]. To learn accurate grid maps, RBPFs have been used by Eliazar and Parr [37] and Hähnel *et al.* [59]. Whereas the first work describes an efficient map representation, the second one presents an improved motion model that reduces the number of required particles.

It should be noted that improvements on particle filters resulting from an informed proposal distributions and an intelligent resampling technique are known techniques within the particle filter community. We would like to especially refer to the work of Doucet [29] who already addressed these issues in his work. However, due to the best of our knowledge, the adaptive resampling has never been used in the context of map learning. The computation of our proposal distribution is similar to the FastSLAM-2 algorithm presented by Montemerlo *et al.* [101]. In contrast to FastSLAM-2, our approach does not rely on predefined landmarks and uses raw laser range finder data to acquire accurate grid maps. Particle filters using proposal distributions that take into account the most recent observation are also called look-ahead particle filters. Morales-Menéndez *et al.* [107] proposed such a method to more reliably estimate the state of a dynamic system outside robotics where accurate sensors are available.

The work described in this chapter can be seen as an extension of the algorithm proposed by Hähnel *et al.* [59]. Instead of using a fixed proposal distribution, our algorithm computes an improved proposal distribution on a per-particle basis on the fly. This allows to directly use most of the information obtained from the sensor while evolving the particles. As a result, we require around one order of magnitude fewer samples compared to the approach of Hähnel *et al.*

The advantage of our approach is twofold. First, our algorithm draws the particles in a more effective way. Second, the highly accurate proposal distribution allows us to utilize the number of effective particles as a robust indicator to decide whether or not a resampling has to be carried out. This further reduces the risk of particle depletion.

One aspect which has not been addressed so far in this chapter is the question on how to deal with dynamically changing environments. Highly dynamic objects like walking persons or moving cars can be filtered [62] so that accurate maps without spurious objects can be obtained. The problem of dealing with low-dynamic or temporarily dynamic objects will be addressed in Chapter 10.

6.7 Conclusion

In this chapter, we presented an approach to learning highly accurate grid maps with Rao-Blackwellized particle filters. Based on the likelihood model of a scan-matching process for the most recent laser range observation, our approach computes an informed proposal distribution. This allows us to draw particles in an more accurate manner which seriously reduces the number of required samples. Additionally, we apply a selective resampling strategy based on the effective number of particles. This approach reduces the number of unnecessary resampling actions in the particle filter and thus substantially reduces the risk of particle depletion.

The approach has been implemented and evaluated on data acquired with different mobile robots equipped with laser range scanners. We furthermore successfully corrected a large number of available robotic datasets and published an open-source implementation of our mapping software. Tests performed with our algorithm in different large-scale environments have demonstrated its robustness and the ability of generating high quality maps. In these experiments, the number of particles needed by our approach often was by one order of magnitude smaller compared to previous approaches.

7

Actively Closing Loops During Exploration

7.1 Introduction

We presented so far approaches to autonomous exploration in Chapter 3-5 and a solution to the SLAM problem in Chapter 6. In general, the task of acquiring models of unknown environments requires to simultaneously address three tasks, which are mapping, localization, and path planning. In the this chapter as well as in the two following ones, we focus on integrated approaches which aim to solve these three tasks simultaneously in order to build accurate models of the environment.

A naïve approach to realize an integrated technique could be to combine a SLAM algorithm, which covers mapping and localization, with an exploration procedure. Since classical exploration strategies often try to cover unknown terrain as fast as possible, they avoid repeated visits to known areas. This strategy, however, is suboptimal in the context of the SLAM problem because the robot typically needs to revisit places in order to localize itself. A good pose estimate is necessary to make the correct data association, i.e., to determine if the current measurements fit into the map built so far. If the robot uses an exploration strategy that avoids multiple visits to the same place, the probability of making the correct association is reduced. This indicates that combinations of exploration strategies and SLAM algorithms should consider whether it is worth reentering already covered spaces or to explore new terrain. It can be expected that a system, which takes this decision into account, can improve the quality of the resulting map.

Figure 7.1 gives an example that illustrates that an integrated approach performing active place revisiting provides better results than approaches that do not consider reentering known terrain during the exploration phase. In the situation shown in the left image, the robot traversed the loop just once. The robot was not able to correctly determine the angle between the loop and the straight corridor because it did not collect enough data to accurately localize itself. The second map shown in the right image has been obtained after the robot traveled twice around the loop to relocalize before entering

C. Stachniss: Robotic Mapping and Exploration, STAR 55, pp. 117–134.
springerlink.com

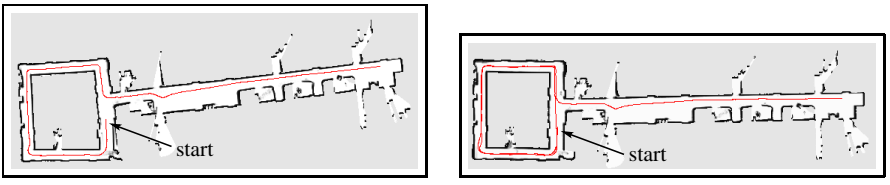


Fig. 7.1. This figure shows two maps obtained from real world data acquired at Sieg Hall, University of Washington. The left image depicts an experiment in which the robot traversed the loop only once before it entered the long corridor. As can be seen, the robot was unable to correctly close the loop, which led to an error of 7 degrees in the orientation of the horizontal corridor. In the case in which the robot revisited the loop, the orientation error was reduced to 1 degree (see right image).

the corridor. As can be seen from the figure, this reduces the orientation error from approximately 7 degrees (left image) to 1 degree (right image). This example illustrates that the capability to detect and actively close loops during exploration allows the robot to reduce its pose uncertainty during exploration and thus to learn more accurate maps.

The contribution of this chapter is an integrated algorithm for generating trajectories to actively close loops during SLAM and exploration. Our algorithm uses a Rao-Blackwellized mapping technique to estimate the map and the trajectory of the robot. It explicitely takes into account the uncertainty about the pose of the robot during the exploration task. Additionally, it applies a technique to reduce the risk that the robot becomes overly confident in its pose when actively closing loops, which is a typical problem of particle filters in this context. As a result, we obtain more accurate maps compared to combinations of SLAM with standard exploration.

This chapter is organized as follows. In Section 7.2, we present our integrated exploration technique. We describe how to detect loops and how to actively close them. Section 7.3 presents experiments carried out on real robots as well as in simulation. Finally, we discuss related work in Section 7.4.

7.2 Active Loop-Closing

Whenever a robot using a Rao-Blackwellized mapper explores new terrain, all samples have more or less the same importance weight since the most recent measurement is typically consistent with the part of the map constructed from the immediately preceding observations. Typically, the uncertainty about the pose of the robot increases. As soon as it reenters known terrain, however, the maps of some particles are consistent with the current measurement and some are not. Accordingly, the weights of the samples differ largely. Due to the resampling step, unlikely particles usually get eliminated and thus the uncertainty about the pose of the robot decreases. One typical example is

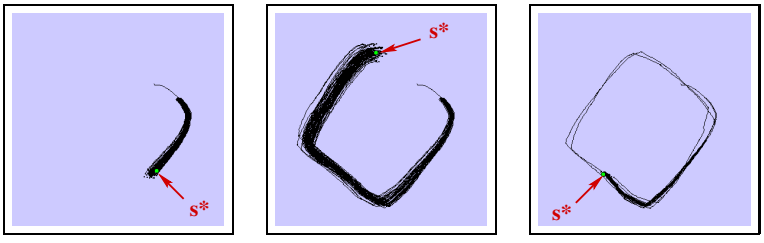


Fig. 7.2. Evolution of a particle set and the map of the most likely particle (here labeled as s^*) at three different time steps. In the two left images, the vehicle traveled through unknown terrain, so that the uncertainty increased. In the right image, the robot reentered known terrain so that samples representing unlikely trajectories vanished.

shown in Figure 7.2. In the two left images, the robot explores new terrain and the uncertainty of the sample set increases. In the right image, the robot travels through known terrain and unlikely particles have vanished.

Note that this effect is much smaller if the robot just moves backward a few meters to revisit previously scanned areas. This is because the map associated with a particle is generally locally consistent. Inconsistencies mostly arise when the robot reenters areas explored some time ago. Therefore, visiting places seen further back in the history has a stronger effect on the differences between the importance weights and typically also on the reduction of uncertainty compared to places recently observed.

7.2.1 Detecting Opportunities to Close Loops

The key idea of our approach is to identify opportunities for closing loops during terrain acquisition in order to relocalize the vehicle. Here, closing a loop means actively reentering the known terrain and following a previously traversed path. To determine whether there exists a possibility to close a loop, we consider two different representations of the environment. In our current system, we associate to each particle s an occupancy grid map $m^{[s]}$ and a topological map $\mathcal{G}^{[s]}$. Both maps are updated while the robot is performing the exploration task. In the topological map $\mathcal{G}^{[s]}$, the vertices represent positions visited by the robot. The edges represent the estimated trajectory of the corresponding particle. To construct the topological map, we initialize it with one node corresponding to the starting location of the robot. Let $x_t^{[s]}$ be the pose of particle s at the current time step t. We add a new node at the position $x_t^{[s]}$ to $\mathcal{G}^{[s]}$ if the distance between $x_t^{[s]}$ and all other nodes in $\mathcal{G}^{[s]}$ exceeds a threshold d (here set to 2.5m) or if none of the other nodes in $\mathcal{G}^{[s]}$ is visible from $x_t^{[s]}$

$$\forall n \in nodes(\mathcal{G}^{[s]}) \quad : \quad \Big[dist_{m^{[s]}}(x_t^{[s]}, n) > d \quad \vee \quad not\_visible_{m^{[s]}}(x_t^{[s]}, n) \Big] . \tag{7.1}$$

Whenever a new node is created, we also add an edge from this node to the most recently visited node. To determine whether or not a node is visible from another node, we perform a ray-casting operation in the occupancy grid $m^{[s]}$.

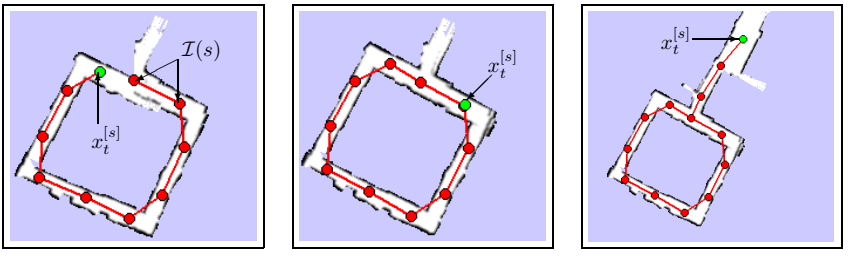


Fig. 7.3. The red dots and lines in these three image represent the nodes and edges of $\mathcal{G}^{[s]}$. In the left image, $\mathcal{I}(s)$ contained two nodes (indicated by the arrows) and in the middle image the robot closed the loop until the pose uncertainty is reduced. After this, it continued with the acquisition of unknown terrain (right image).

Figure 7.3 depicts such a graph for one particular particle during different phases of an exploration task. In each image, the topological map $\mathcal{G}^{[s]}$ is printed on top of metric map $m^{[s]}$. To motivate the idea of our approach, we would like to refer the reader to the left image of this figure. Here, the robot almost closed a loop. This can be observed by the fact that the length of the shortest path between the current pose of the robot and previously visited locations(here marked with $\mathcal{I}(s)$) *is large in the topological map* $\mathcal{G}^{[s]}$, whereas it is *small in the grid map* $m^{[s]}$.

The shortest path in from the current pose of the robot to those locations models a shortcut in the environment which has not been traversed to far. Following such a path exactly characterized a loop closure.

Thus, to determine whether or not a loop can be closed, we compute for each sample s the set $\mathcal{I}(s)$ of positions of interest. This set contains all nodes that are close to the current pose $x_t^{[s]}$ of particle s based on the grid map $m^{[s]}$, but are far away given the topological map $\mathcal{G}^{[s]}$

$$\mathcal{I}(s) = \{x_{t'}^{[s]} \in nodes(\mathcal{G}^{[s]}) \mid dist_{m^{[s]}}(x_{t'}^{[s]}, x_t^{[s]}) < d_1 \wedge dist_{\mathcal{G}^{[s]}}(x_{t'}^{[s]}, x_t^{[s]}) > d_2\}. \tag{7.2}$$

Here, $dist_{\mathcal{M}}(x_1, x_2)$ is the length of the shortest path from x_1 to x_2 given the representation $\mathcal{M}$. The distance between two nodes in $\mathcal{G}^{[s]}$ is given by the length of the shortest path between both nodes. The length of a path

is computed by the sum over the lengths of the traversed edges. Depending on the number of nodes in $\mathcal{I}(s)$, this distance information can be efficiently computed using either the A^* algorithm [112, 130] or Dijkstra's algorithm [114]. The terms d_1 and d_2 are constants that must satisfy the constraint $d_1 < d_2$. In our current implementation, the values of these constants are $d_1 = 6m$ and $d_2 = 20m$.

In case $\mathcal{I}(s) \neq \emptyset$, there exist at least one shortcut from the current pose $x_t^{[s]}$ of the particle s to the positions in $\mathcal{I}(s)$. These shortcuts represent edges that would close a loop in the topological map $\mathcal{G}^{[s]}$. The left image of Figure 7.3 illustrates a situation in which a robot encounters the opportunity to close a loop since $\mathcal{I}(s)$ contains two nodes which is indicated by two arrows. The key idea of our approach is to use such shortcuts whenever the uncertainty of the robot in its pose becomes large. The robot then revisits portions of the previously explored area and in this way reduces the uncertainty in its position.

To determine the most likely movement allowing the robot to follow a previous path, one in principle has to integrate over all particles and consider all potential outcomes of that particular action. Since this would be too time consuming for online-processing, we consider only the particle s^* with the highest accumulated logarithmic observation likelihood

$$s^* = \operatorname*{argmax}_s \sum_{t'=1}^{t} \log p(z_{t'} \mid m^{[s]}, x_{t'}^{[s]}). \tag{7.3}$$

If $\mathcal{I}(s^*) \neq \emptyset$, we select the node x_{t_e} from $\mathcal{I}(s^*)$ which is closest to $x_t^{[s^*]}$

$$x_{t_e} = \operatorname*{argmin}_{x \in \mathcal{I}(s^*)} dist_{m^{[s^*]}}(x_t^{[s^*]}, x). \tag{7.4}$$

In the sequel, x_{t_e} is denoted as the *entry point* at which the robot has the possibility to close a loop. t_e corresponds to the last time the robot was at the node x_{t_e}.

7.2.2 Representing Actions under Pose Uncertainty

One open question is how to express an action if the robot acts under pose uncertainty. A list of positions expressed in a global coordinate frame is usually not a good solution since this action is only valid for a single particle. Whenever the particle s^* which is used to compute the plan changes, the robot would need to recompute its action. An alternative solutions is to express the action as a sequence of relative motion commands. This works fine as long as the robot moves through unknown terrain or the pose uncertainty is not too big.

We use a slightly different method that provides more stable plans. Instead of using a sequence of relative motions commands with respect to the

current pose of the robot, we use the nodes in our topological maps as reference frames. An example for such an actions (expressed in human language) could be "move to the position 1 m north of the node 5, turn 90 degree right and move to node 7." As mentioned before, in our approach the actions are planned based on the particle s^*. In case a different particle becomes the particle s^*, it is likely that we do not need to replan our action since it is expressed relative to the nodes of the topological map. Often such a plan is still valid after s^* changed. Valid means in this context that the planned path does not lead to a collision. In case a collision with a wall is detected, the action is recomputed or the loop-closing procedure is aborted.

7.2.3 Stopping the Loop-Closing Process

To determine whether or not the robot should activate the loop-closing behavior, our system constantly monitors the uncertainty H about the robot's pose at the each point in time. The necessary condition for starting the loop-closing process is the existence of an entry point x_{t_e} and that $H(t)$ exceeds a given threshold. Once the loop-closing process has been activated, the robot approaches x_{t_e} and then follows the path taken after previously arriving at x_{t_e}. During this process, the uncertainty in the pose of the vehicle typically decreases because the robot is able to localize itself in the map built so far and unlikely particles vanish.

We have to define a criterion for deciding when the robot actually has to stop following a loop. A first attempt could be to introduce a threshold and to simply stop the trajectory following behavior as soon as the uncertainty becomes smaller than a given threshold. This criterion, however, can be problematic especially in the case of nested loops. Suppose the robot encounters the opportunity to close a loop that is nested within an outer and so far unclosed loop. If it eliminates all of its uncertainty by repeatedly traversing the inner loop, particles necessary to close the outer loop may vanish. As a result, the filter diverges and the robot fails to build a correct map (see Figure 7.4).

To remedy this so-called particle depletion problem, we introduce a constraint on the uncertainty of the robot. Let $H(t_e)$ denote the uncertainty of the posterior when the robot visited the entry point last time. Then the new constraint allows the robot to re-traverse the loop only as long as its current uncertainty $H(t)$ exceeds $H(t_e)$. If the constraint is violated the robot resumes its terrain acquisition process. This constraint is designed to reduce the risk of depleting relevant particles during the loop-closing process. The idea behind this constraint is that by observing the area within the loop, the robot does not obtain any information about the world outside the loop. From a theoretical point of view, the robot cannot reduce the uncertainty $H(t)$ in its current posterior below its uncertainty $H(t_e)$ when entering the loop since $H(t_e)$ is the uncertainty stemming from the world outside the loop.

To better illustrate the importance of this constraint, consider the following example: A robot moves from place A to place B and then repeatedly observes

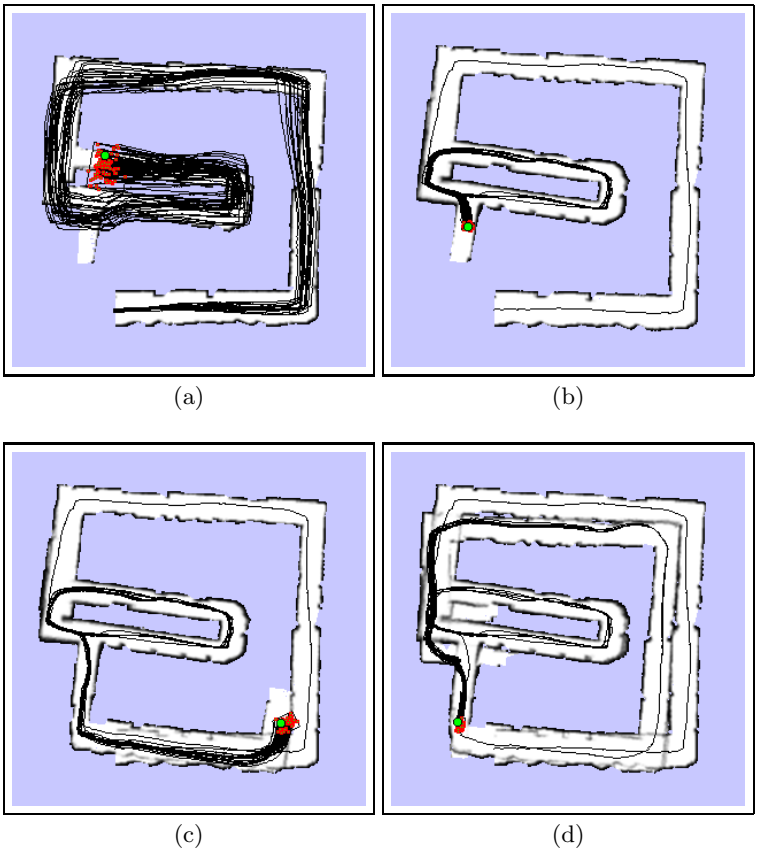


Fig. 7.4. An example for particle depletion. A robot explores an environment and travels through the inner loop (a) several times. Due to the repeated visits the diversity of hypotheses about the trajectory outside the inner loop decreases (b) too much and the robot is unable to close the outer loop correctly (c) and (d).

B. While it is mapping B, it does not get any further information about A. Since each particle represents a whole trajectory (and the corresponding map) of the robot, hypotheses representing ambiguities about A will also vanish when reducing potential uncertainties about B. Our constraint reduces the risk of depleting particles representing ambiguities about A by aborting the loop-closing behavior at B as soon as the uncertainty drops below the uncertainty stemming from A.

Finally, we have to describe how we actually measure the uncertainty in the position estimate. The typical way of measuring the uncertainty of a posterior is to use the entropy. To compute the entropy of a posterior represented by particles, one typically uses a multi-dimensional grid representing the possible

(discretized) states. Each cell c in this (virtual) grid stores a probability which is given by the sum of the normalized weights of the samples corresponding to that cell. The entropy is then computed by

$$H(t) = -\sum_{c} p(c) \log p(c) \tag{7.5}$$

$$= -\sum_{c} \left[\left(\sum_{i:x^{[i]} \in c} w_t^{[i]} \right) \log \left(\sum_{i:x^{[i]} \in c} w_t^{[i]} \right) \right], \tag{7.6}$$

where $i : x^{[i]} \in c$ refers to the indices of all particles which current poses $x^{[i]}$ lie within the area which is covered by the grid cell c.

In the case of multi-modal distributions, however, the entropy does not consider the distance between the different modes. This distance, however, is an important property when evaluating the pose uncertainty of a mobile vehicle. As a result, a set of k different pose hypotheses which are located close to each other but do not belong to the same cell c leads to the same entropy value than the situation in which k hypotheses are randomly distributed over the environment. The resulting maps, however, would look similar in the first case, but quite different in the second case. In our experiments, we figured out that we obtain better results if we use the volume expanded by the samples instead of the entropy. We therefore calculate the pose uncertainty by determining the volume of the oriented bounding box around the particle cloud. A good approximation of the minimal oriented bounding box can be obtained efficiently by a principal component analysis.

Note that the loop-closing process is also aborted in case a robot travels for a long period of time through the same loop in order to avoid a – theoretically possible – endless loop-closing behavior. In all our experiments, however, this problem has never been encountered.

7.2.4 Reducing the Exploration Time

The experiments presented later on in this chapter demonstrate that our uncertainty based stopping criterion is an effective way to reduce the risk of particle depletion. However, it can happen that the perceived sensor data after closing a loop does not provide a lot of new information for the robot. Moving through such terrain leads to an increased exploration time since the robot does redundant work which does not provide relevant information. It would be more efficient to abort the loop-closing procedure in situations in which the new sensor data does not help to identify unlikely hypotheses.

To estimate how well the current set of N particle represents the true posterior, Liu [90] introduced the effective number of particles N_{eff}, see (6.22). In the previous chapter, we described how to use N_{eff} to resample in an intelligent way but it is also useful in the context of active loop-closing. We monitor the change of N_{eff} over time, which allows us to analyze how the

new acquired information affects the filter. If N_{eff} stays constant, the new information does not help to identify unlikely hypotheses represented by the individual particles. In that case, the variance in the importance weights of the particles does not change over time. If, in contrast, the value of N_{eff} decreases over time, the new information is used to determine that some particles are less likely than others. This is exactly the information we need in order to decide whether or not the loop-closing should be aborted. As long as new information helps to identify unlikely particles, we follow the loop. As soon as the observations do not provide any new knowledge about the environment for a period of k time steps, we continue to explore new terrain in order to keep the exploration time small.

Note that this criterion is optional and not essential for a successful loop-closing strategy. It can directly be used if the underlying mapping approach applies an adaptive resampling technique. If no adaptive resampling is used, one needs to monitor the relative change in N_{eff} after integrating each measurement, because after each resampling step the weights of all particles are set to $\frac{1}{N}$. In the experimental section of this chapter, we illustrate that N_{eff} is a useful criterion in the context of active loop-closing and how it behaves during exploration.

As long as the robot is localized well enough or no loop can be closed, we use a frontier-based exploration strategy to choose a target location for the robot. As described before, a frontier is any known and unoccupied cell that is an immediate neighbor of an unknown, unexplored cell [163]. By extracting frontiers from a given grid map, one can easily determine potential target locations which guide the robot to so far unknown terrain. As in Chapter 4, the cost of the target locations is given by the cost function presented in Section 4.2.1. In our current system, we determine frontiers based on the map of the particle s^*.

A precise formulation of the loop-closing strategy is given by Algorithm 7.1. In our current implementation, this algorithm runs as a background process that is able interrupt the frontier-based exploration procedure.

Algorithm 7.1 The loop-closing algorithm

1: Compute $\mathcal{I}(s^*)$
2: **if** $\mathcal{I}(s^*) \neq \emptyset$ **then begin**
3: $\quad H = H(t_e)$
4: $\quad \mathit{path} = x_t^{[s^*]} + \mathit{shortest\_path}_{\mathcal{G}^{[s^*]}}(x_{t_e}, x_t^{[s^*]})$
5: $\quad$ **while** $H(t) > H \ \wedge \ \mathrm{var}(N_{\mathit{eff}}(n-k), \ldots, N_{\mathit{eff}}(n)) > \epsilon$ **do**
6: $\quad\quad \mathit{robot\_follow}(\mathit{path})$
7: **end**

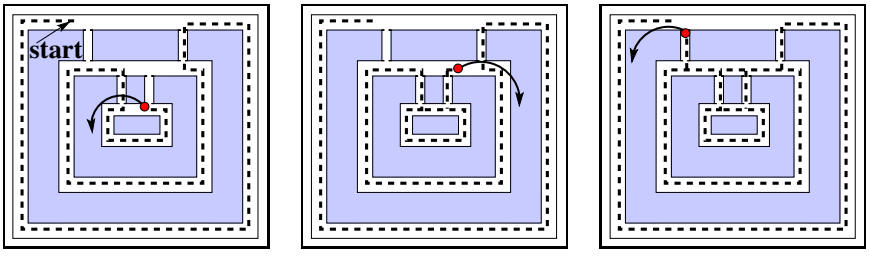


Fig. 7.5. Active loop-closing of multiple nested loops.

7.2.5 Handling Multiple Nested Loops

Note that our loop-closing technique can also handle multiple nested loops. During the loop-closing process, the robot follows its previously taken trajectory to relocalize. It does not leave this trajectory until the termination criterion (see line 5 in Algorithm 7.1) is fulfilled. Therefore, it never starts a new loop-closing process before the current one is completed. A typical example with multiple nested loops is illustrated in Figure 7.5. In the situation depicted in the left picture, the robot starts with the loop-closing process for the inner loop. After completing the most inner loop, it moves to the second inner one and again starts the loop-closing process. Since our algorithm considers the uncertainty at the entry point, it keeps enough variance in the filter to also close the outer loop correctly. In general, the quality of the solution and whether or not the overall process succeeds depends on the number of particles used. Since determining this quantity is still an open research problem, the number of particles has to be defined by the user in our current system.

7.3 Experimental Results

Our approach has been implemented and evaluated in a series of real world and simulation experiments. For the real world experiments, we used an iRobot B21r robot and an ActivMedia Pioneer 2 robot. Both are equipped with a SICK laser range finder. For the simulation experiments, we used the real-time simulator of the Carnegie Mellon robot navigation toolkit (CARMEN) [128].

The experiments described in this section illustrate that our approach can be used to actively learn accurate maps of large indoor environments. Furthermore, they demonstrate that our integrated approach yields better results than an approach which does not has the ability to actively close loops. Additionally, we analyze how the active termination of the loop closure influences the result of the mapping process.

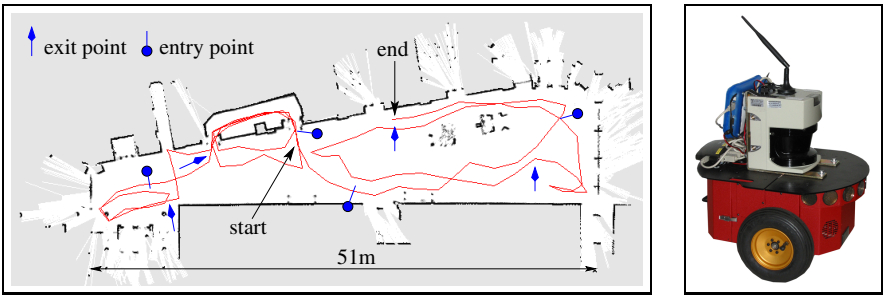


Fig. 7.6. The left image shows the resulting map of an exploration experiment in the entrance hall of the Department for Computer Science at the University of Freiburg. It was carried out using a Pioneer 2 robot equipped with a laser range scanner (see right image). Also plotted is the path of the robot as well as entry and exit points where the robot started and stopped the active loop-closing process.

7.3.1 Real World Exploration

The first experiment was carried out to illustrate that our current system can effectively control a mobile robot to actively close loops during exploration. To perform this experiment, we used a Pioneer 2 robot to explore the main lobby of the Department for Computer Science at the University of Freiburg. The size of this environment is 51 m by 18 m. Figure 7.6 depicts the final result obtained by a completely autonomous exploration run using our active loop-closing technique. It also depicts the trajectory of the robot, which has an overall length of 280 m. The robot decided four times to reenter a previously visited loop in order to reduce the uncertainty in its pose. Figure 7.6 shows the resulting map, the corresponding entry points as well as the positions where the robot left the loops ("exit points"). As can be seen, the resulting map is quite accurate.

7.3.2 Active Loop-Closing vs. Frontier-Based Exploration

The second experiment should illustrate the difference to approaches that do not consider loop-closing actions. We used real robot data obtained with a B21r robot in the Sieg Hall at the University of Washington. As can be seen from the motivating example in the introduction of this chapter (see Figure 7.1), the robot traversed the loop twice during map building. To eliminate the influence of unequal measurement noise and different movements of the robot, we removed the data corresponding to one loop traversal from the recorded data file and used this data as input to our SLAM algorithm. In this way, we simulated the behavior of a greedy exploration strategy which forces the robot to directly enter the corridor after returning to the starting location in the loop. As can be seen from the same figure, an approach that does not

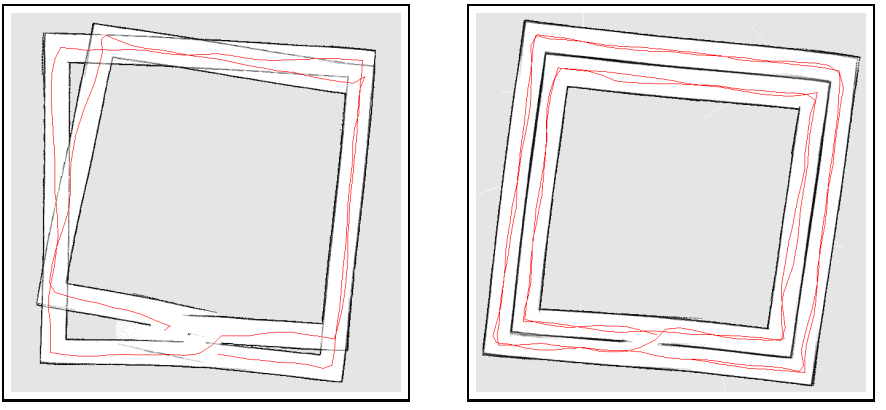


Fig. 7.7. This figure depicts an environment with two large loops. The outer loop has a length of over 220 m. The left image show the resulting map of a trajectory in which the robot drove through the loops only once. In the second run, the robot visited every loop twice and obtained a highly accurate map (see right image).

actively reenter the loop fails to correctly estimate the angle between the loop and the corridor which should be oriented horizontally in that figure. Whereas the angular error was 7 degrees with the standard approach, it was only 1 degree in the case where the robot traversed the loop twice. The depicted maps corresponded to the one of the particle s^*.

A further experiment that illustrates the advantage of place revisiting is shown in Figure 7.7. The environment used in this simulation run is 80 m by 80 m and contains two large nested loops with nearly featureless corridors. The left image shows the result of the frontier-based approach which traversed each loop only once. Since the robot is not able to correct the accumulated pose error, the resulting map contains large inconsistencies and two of the corridors are mapped onto each other. Our approach, in contrast, first revisits the outer loop before entering the inner one (see right image). Accordingly, the resulting map is more accurate.

7.3.3 A Quantitative Analysis

To quantitatively evaluate the advantage of the loop-closing behavior, we performed a series of simulation experiments in an environment similar to Sieg Hall. We performed 20 experiments, 10 with active loop-closing and 10 without. After completing the exploration task, we measured the average error in the relative distances between positions lying on the resulting estimated trajectory and the ground truth provided by the simulator. The results are depicted in Figure 7.8. As can be seen, the active loop-closing behavior significantly reduces the error in the position of the robot.

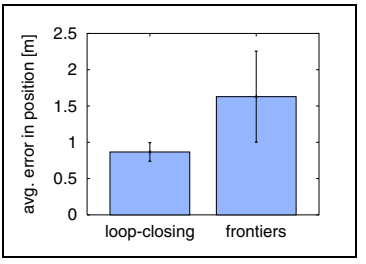


Fig. 7.8. This figure compares our loop-closing strategy with a pure frontier-based exploration technique. The left bar in this graph plots the average error in the pose of the robot obtained with our loop-closing strategy. The right one shows the average error when a frontier-based approach was used. As can be seen, our technique significantly reduces the distances between the estimated positions and the ground truth (confidence intervals do not overlap).

7.3.4 Importance of the Termination Criterion

In this experiment, we analyze the importance of the constraint that terminates the active loop-closing behavior as soon as the current uncertainty $H(t)$ of the belief drops under the uncertainty $H(t_e)$ of the posterior when the robot was at the entry point the last time.

In this simulated experiment, the robot had to explore an environment which contains two nested loops and is depicted in Figure 7.9 (d). In the first case, we simply used a constant threshold to determine whether or not the loop-closing behavior should be stopped. In the second case, we applied the additional constraint that the uncertainty should not become smaller than $H(t_e)$.

Figure 7.4 shows the map of the particle s^* obtained with our algorithm using a constant threshold instead of considering $H(t_e)$. In this case, the robot repeatedly traversed the inner loop (a) until its uncertainty was reduced below a certain threshold. After three and a half rounds it decided to again explore unknown terrain, but the diversity of hypotheses had decreased too much (b). Accordingly the robot was unable to accurately close the outer loop (c) and (d). We repeated this experiment several times and in none of the cases was the robot able to correctly map the environment. In contrast, our approach using the additional constraint always generated an accurate map. One example is shown in Figure 7.9. Here, the robot stopped the loop-closing after traversing half of the inner loop.

As this experiment illustrates, the termination of the loop-closing is important for the convergence of the filter and to obtain accurate maps in environments with several (nested) loops. Note that similar results in principle can also be obtained without this termination constraint if the number of particles is substantially increased. Since exploration is an online problem and each particle carries its own map it is of utmost importance to keep

the number of particles as small as possible. Therefore, our approach can also be regarded as a contribution to limit the number of particles during Rao-Blackwellized simultaneous localization and mapping.

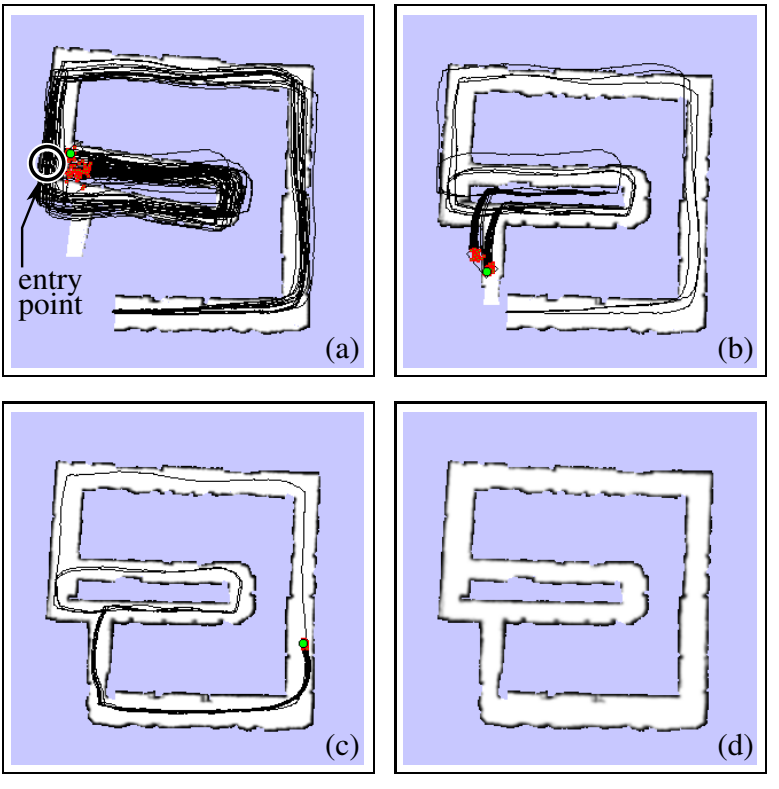


Fig. 7.9. In image (a), the robot detected an opportunity to close a loop. It traversed parts of the inner loop as long as its uncertainty exceed the uncertainty $H(t_e)$ of the posterior when the robot at the entry point and started the loop-closing process. The robot then turned back and left the loop (b) so that enough hypotheses survived to correctly close the outer loop (c) and (d). In contrast, a system considering only a constant threshold criterion fails to map the environment correctly as depicted in Figure 7.4.

7.3.5 Evolution of N_{eff}

In this experiment, we show the behavior of the optional termination criterion that triggers the active loop-closing behavior. Additionally, the constraint that the uncertainty $H(t)$ must be bigger than the uncertainty at the entry point $H(t_e)$ of the loop, the process is stopped whenever the effective number

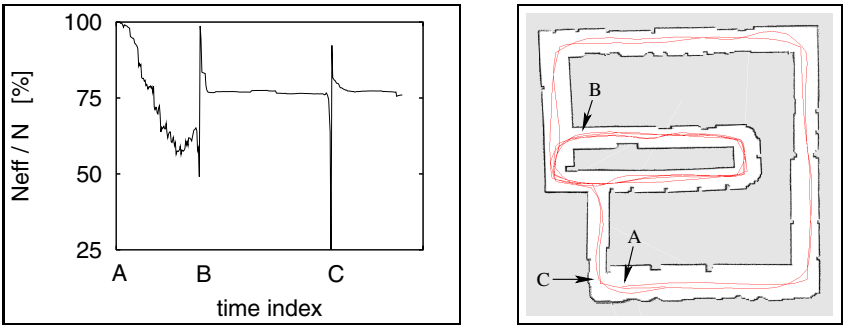


Fig. 7.10. The graph plots the evolution of the N_{eff} function over time during an experiment carried out in the environment shown in the right image. The robot started at position A. The position B corresponds to the closure of the inner loop and C corresponds to closure of the outer loop.

of particles N_{eff} stays constant for a certain period of time. This criterion was introduced to avoid that the robot moves through the loop even if no new information can be obtained from the sensor data. The robot re-traverses the loop only as long as the sensor data is useful to identify unlikely hypotheses about maps and poses.

One typical evolution of N_{eff} is depicted in the left image of Figure 7.10. To achieve a good visualization of the evolution of N_{eff}, we processed a recorded data file using 150 particles. Due to the adaptive resampling strategy, only a few resampling operations were needed. The robot started at position A and in the first part of the experiment moved through unknown terrain (between the positions A and B). As can be seen, N_{eff} decreases over time. After the loop has been closed correctly and unlikely hypotheses had partly been removed by the resampling action (position B), the robot re-traversed the inner loop and N_{eff} stayed more or less constant. This indicates that acquiring further data in this area has only a small effect on the relative likelihood of the particles and the system could not determine which hypotheses represented unlikely configurations. In such a situation, it therefore makes more sense to focus on new terrain acquisition and to not continue the loop-closing process.

Furthermore, we analyzed the length of the trajectory traveled by the robot. Due to the active loop-closing, our technique generates longer trajectories compared to a purely frontier-based exploration strategy. We performed several experiments in different environments in which the robot had the opportunity to close loops and measured the average overhead. During our experiments, we observed an overhead varying from 3% to 10%, but it obviously depends on number of loops in the environment.

7.3.6 Multiple Nested Loops

To illustrate, that our approach is able to deal with several nested loops, we performed a simulated experiment shown in Figure 7.11. The individual images in this figure depict eight snapshots recorded during exploration. Image (a) depicts the robot while exploring new terrain and image (b) while actively closing the most inner loop. After that, the robot focused on acquiring so far unknown terrain and moves through the most outer loop as shown in (c) and (d). Then the robot detects a possibility to close a loop (e) and follows its previously taken trajectory (f). After aborting the loop closing behavior, the robot again explores the loop in the middle (g), again closes the loop accurately, and finishes the exploration task (h).

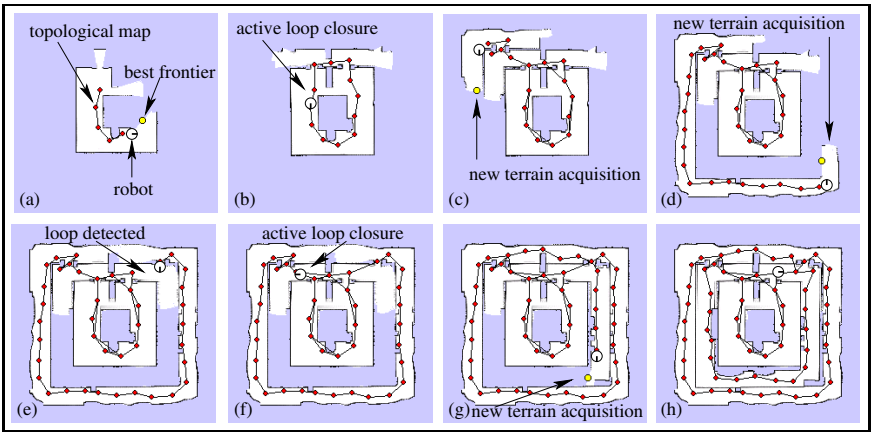


Fig. 7.11. Snapshots during the exploration of a simulated environment with several nested loops. The red circles represent nodes of the topological map plotted on top of the most likely grid map. The yellow circle corresponds to the frontier cell the robot currently seeks to reach.

7.3.7 Computational Resources

Note that our loop-closing approach needs only a few additional resources. To detect loops, we maintain an additional topological map for each particle. These topological maps are stored as a graph structure and for typical environments only a few kilobytes of extra memory is needed. To determine the distances based on the grid map in (7.1) and (7.2), our approach directly uses the result of a value iteration (alternatively Dijkstra's algorithm) based on the map of s^*, which has already been computed in order to evaluate the frontier cells. Only the distance computation using the topological map

needs to be done from scratch. However, since the number of nodes in the topological map is much smaller than the number of grid cells, the computational overhead is comparably small. In our experiments, the time to perform all computations in order to decide where to move next increased by around 10 ms on a standard PC when our loop-closing technique was enabled.

7.4 Related Work

Several previous approaches to SLAM and mobile robot exploration are related to our work. In the context of exploration, most of the techniques presented so far focus on generating motion commands that minimize the time needed to cover the whole terrain [77, 160, 163]. Other methods like, for example, the one presented in Chapter 3 or the work of Grabowski *et al.* [54] seek to optimize the viewpoints of the robot to maximize the expected information gain and to minimize the uncertainty of the robot about grid cells. Most of these techniques, however, assume that the location of the robot is known during exploration. A detailed discussion about those approaches is provided in the related work sections of Chapter 3 and 4.

In the area of SLAM, the vast majority of papers focuses on the aspect of state estimation as well as belief representation and update [28, 30, 37, 57, 59, 104, 109, 145]. A detailed discussion of related SLAM approaches can be found in Section 6.6. Classical SLAM techniques are passive and only consume incoming sensor data without explicitely generating controls.

Recently, some techniques have been proposed which actively control the robot during SLAM. For example, Makarenko *et al.* [94] as well as Bourgault *et al.* [15] extract landmarks out of laser range scans and use an extended Kalman filter to solve the SLAM problem. Furthermore, they introduce a utility function which trades off the cost of exploring new terrain with the utility of selected positions with respect to a potential reduction of uncertainty. The approaches are similar to the work done by Feder *et al.* [40] who consider local decisions to improve the pose estimate during mapping. Sim *et al.* [135] presented an approach in which the robot follows a parametric curve to explore the environment and considers place revisiting actions if the pose uncertainty gets too high. These four techniques integrate the uncertainty in the pose estimate of the robot into the decision process of where to move next. However, they rely on the fact that the environment contains landmarks that can be uniquely determined during mapping.

In contrast to this, the approach presented here makes no assumptions about distinguishable landmarks in the environment. It uses raw laser range scans to compute accurate grid maps. It considers the utility of reentering known parts of the environment and following an encountered loop to reduce the uncertainty of the robot in its pose. In this way, the resulting maps become more accurate.

There exist techniques to combine topological maps with other kind of spacial representations. This is typically done to handle large-scale maps or to simplify the loop-closing problem [14, 82, 89]. Those approaches can attach detailed local maps to the nodes of the topological map. Also our approach makes use of topological maps. However, building up such a hierarchy is not intended by our work, since we only use the topological map to detect loops in the environment.

7.5 Conclusion

In this chapter, we presented an approach for active loop-closing during autonomous exploration. We combined the Rao-Blackwellized particle filter for simultaneous localization and mapping presented in the previous chapter with a frontier-based exploration technique extended by the ability to actively close loops. Our algorithm forces the robot to re-traverse previously visited loops and in this way reduces the uncertainty in the pose estimate. The loop detection is realized by maintaining two different representations of the environments. By comparing a grid map with a topological map, we are able to detect loops in the environment that have not been closed so far. As a result, we obtain more accurate maps compared to combinations of SLAM algorithms with classical exploration techniques. As fewer particles need to be maintained to build accurate maps, our approach can also be regarded as a contribution to reduce the number of particles needed during Rao-Blackwellized mapping.

8

Recovering Particle Diversity

8.1 Introduction

We presented in Chapter 7 a technique that allows an exploring robot to detect loops and to carry out place revisiting actions. We showed that the quality of a map constructed by a mobile robot depends on its trajectory during data acquisition. This is due to the fact that the vehicle needs to relocalize itself during exploration in order to build an accurate model of the environment. Our loop-closing technique uses a heuristic stopping criterion in order to continue with the new terrain acquisition and to reduce the risk that the particle depletion problem affects the filter. We showed that such an approach works well in practice, however, the general problem of particle depletion in the context of loop-closing still exists. To overcome this limitation, we present in this chapter a technique that is able to approximatively recover lost hypotheses. It enables a mobile robot to store the particle diversity of the filter before entering a loop. When leaving the loop, the robot is then able to recover that diversity and continue the exploration process. This technique allows a mobile robot to stay – at least theoretically – arbitrarily long in a loop without depleting hypotheses needed to close an additional, outer loop.

Figure 8.1 illustrates the problem of vanished particles in the context of repeated loop traversals in environments with nested loops. Due to the risk of particle depletion, the robot should spend only a limited period of time in an inner loop. In situations in which the robot is forced to move through a loop for a longer period of time, the particle diversity is likely to get lost. This can, for example, be the case in environments with extremely long loops. Even if the robots seeks to explore new terrain, it may need to travel for long distances through the loop before it can reach a frontier (see Figure 8.2). Such a situation can lead to particle depletion too.

The contribution of this chapter is a technique to recover the variety of trajectory hypotheses represented by a Rao-Blackwellized particle filter in the context of nested loops. Our approach determines an approximation of the posterior given by the particles at the entry of a loop and propagates

C. Stachniss: Robotic Mapping and Exploration, STAR 55, pp. 135–142.
springerlink.com

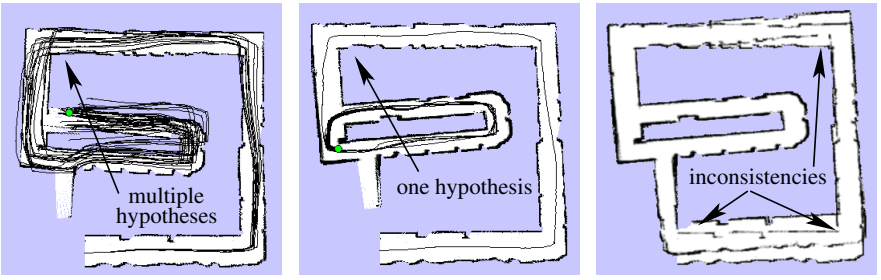


Fig. 8.1. This figure illustrates that a loss of particle diversity introduced by repeated loop closing can lead to a wrong solution in the context of mapping with a Rao-Blackwellized particle filter.

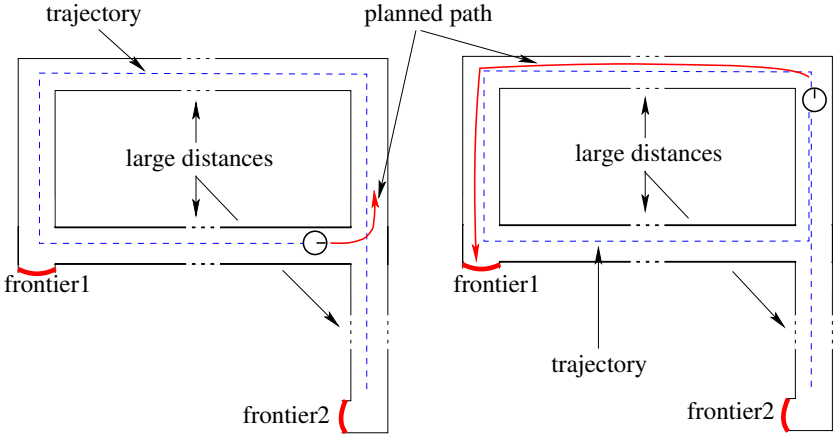


Fig. 8.2. In this experiment, the robot started the exploration process in the lower right corner of the environment. In the left image, is activates the loop-closing process and follows its previously taken trajectory. In the right image, the robot aborts the loop-closing process. However, to reach the next frontier, the robots needs to travel through known areas for a long time. This can lead to particle depletion.

its uncertainty through the loop. This way, hypotheses needed to close an outer loop are maintained. The major advantage of this approach is that the robot can, in principle, stay arbitrarily long in an inner loop without losing information necessary to close outer loops.

This chapter is organized as follows. Section 8.2 then describes how to recover the diversity of a particle filter when the robot leaves a loop. Section 8.3 contains experimental results carried out on real robots as well as in simulation. Finally, Section 8.4 discusses related work.

8.2 Recovering Particle Diversity after Loop Closure

To overcome the problems of particle depletion in the context of nested loops, we need a way to recover hypotheses vanished from a particle filter during the repeated traversal of an inner loop. Even if our active loop-closing technique in combination with the stopping criterion based on N_{eff} makes particle depletion unlikely, the vanishing of important hypotheses and the resulting problem of filter divergence remains. Note that the risk of particle depletion increases with the size of the environment. Also, the smaller the number of particles, the higher is that risk.

As an example, suppose a robot has accurately mapped an inner loop in an environments which contains nested loops. In such a case, its likely that the particle filter has converged to a highly peaked distribution and only one hypothesis present at the entry point has survived. Thus, it is not guaranteed that this hypothesis is the one which perfectly closes the outer loop. In principle, a robot therefore has to maintain a sufficient variety of particles allowing it to perform the next loop closure. Since the robot does not know in advance how many loops it will find in the environment, this problem cannot be solved in general with a finite number of particles only.

If one knew the starting point of such an inner loop in advance, one solution would be to suspend the particle filter and to start for each particle a new filter initialized with the current state of that particle. After the convergence of all filters one can then attach their solutions to the corresponding particles in the suspended filter. Apart from the fact that a loop cannot be recognized in advance this approach is not feasible for online tasks like exploration since the amount of computational resources needed grows exponentially in the number of loops.

The technique described in the following is an approximation of this approach. The key idea is to simulate this process as soon as the robot detects an opportunity to close a loop using our approach presented in the previous chapter. The robot computes the trajectory posterior *at the entry point of the loop* given the particles in its current belief. In this approximative particle set, the states and weights are computed according to

$$\tilde{x}_{t_e}^{[s]} \simeq \mathrm{pred}_{t_e}(x_t^{[s]}) \tag{8.1}$$

$$\tilde{w}_{t_e}^{[s]} \simeq w_t^{[s]}. \tag{8.2}$$

Here $\mathrm{pred}_{t_e}(x_t^{[s]})$ is the state of the ancestor of $x_t^{[s]}$ at time t_e. Note that especially the weight computation is an approximation. Typically, this value has changed between time t and t_e since new observations have been integrated.

Whenever the robot stops the loop closing behavior it uses this posterior to propagate the variety of the particles through the loop. In probabilistic terms, this corresponds to rewriting the term $p(x_{1:t} \mid z_{1:t}, u_{1:t-1})$ in (6.1) as

$$
\begin{aligned}
& p(x_{1:t} \mid z_{1:t}, u_{1:t-1}) \\
\overset{\text{product rule}}{=} \quad & p(x_{t_e+1:t} \mid x_{1:t_e}, z_{1:t}, u_{1:t-1}) p(x_{1:t_e} \mid z_{1:t_e}, u_{1:t_e-1}) && (8.3) \\
= \quad & p(x_{t_e+1:t} \mid x_{1:t_e}, z_{1:t}, u_{t_e:t-1}) p(x_{1:t_e} \mid z_{1:t_e}, u_{1:t_e-1}). && (8.4)
\end{aligned}
$$

The last transformation is valid under the assumption that previous odometry readings can be neglected given the poses are known.

In our current implementation, the first posterior of the last equation is approximated by importance sampling from $p(x_{t_e+1:t} \mid x_{1:t_e}, z_{1:t}, u_{t_e:t-1})$. Then, the trajectory drawn from this posterior is attached to each particle in $p(x_{1:t_e} \mid z_{1:t_e}, u_{1:t_e-1})$. This process propagates the different hypotheses from the entry point into the current belief before leaving the loop. If the robot then has to close a second loop, it is more likely to maintain appropriate hypotheses to close this loop accurately.

Equation (8.1) and (8.2) describe approximations of the sample set. Even if no resampling is carried out between t_e and t the observation likelihoods have been integrated into the weight of the particles. However, if a highly accurate proposal like our one presented in Chapter 6 is used the error is comparably small.

Note that in general a mapping system has to maintain a stack of saved states especially in environments with several nested loops. Due to the fact that we actively control the robot and never start a second loop-closing process before completing the current one, we only have to maintain a single saved state at each point in time.

As we demonstrate in the experiments, this technique is a powerful tool to recover vanished hypotheses without restarting the mapping algorithm from scratch. It only needs to attach a local trajectory to each particle which can be done within a few seconds (on a 2.8 GHz Pentium IV PC).

8.3 Experimental Results

Our approach has been implemented and evaluated on real robot data and in simulation. The experiments described here are designed to illustrate the benefit of our active loop closing technique with the ability to recover the diversity of the particles after loop closing.

This experiment is designed to show the effect of our technique to recover the particle variety when the robot leaves a loop. The environment used to carry out this experiment is depicted in the right image of Figure 8.3. The robot started in the outer loop, entered the inner loop, and moved through this loop for a long period of time. As shown in Figure 8.1, without our recovering technique the filter can converge to a wrong solution. The reason is that at the time when the robot leaves the loop only one hypothesis of the original particle set at the entry point has survived. Accordingly, the robot lacks an appropriate particle to accurately close the outer loop. Using

our algorithm, however, the robot can recover the hypotheses at the entry point and can propagate them through the loop (see left and middle image of Figure 8.3). The most likely map of the posterior after closing the outer loop is shown in the right image.

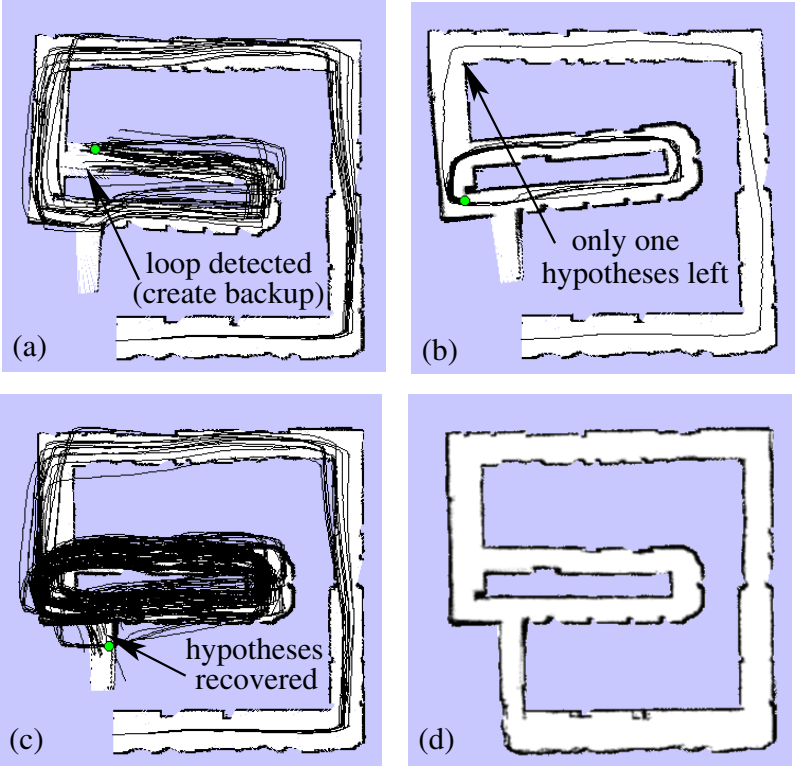


Fig. 8.3. This figure shows the same experiment as depicted in Figure 8.1, but using our recovering technique. In the left image the robots saves the set of approximated particles at time step t_e and later on recovers the vanished hypotheses (middle image). This allows the robot to correctly close the outer loop (right image).

To provide a more quantitative analysis, we mapped the environment 30 times without the capability of restoring the filter and 30 times with this option. The standard technique was able to build a correct map in only 40% of all runs. In all other cases the algorithm did not produce an accurate map. In contrast to this, our algorithm yielded a success rate of 93%. We repeated this experiment in different environments and got similar results. Figure 8.4 shows two (partial) maps of the Killian Court at the MIT. The left map has been built without the recovering technique using 40 particles and

shows inconsistencies due to vanished hypotheses. The right map has been constructed using our recovering technique in which the correct hypothesis has been restored. The average success rate of our approach was 55% whereas the standard approach found the correct data association in only 5% of all runs. We measured success by the fact that the map was topologically correct. This means that there exist no double corridors or large alignment errors. The evaluation if a map was topologically correct, was made by manual inspection of the resulting map.

Note that the second experiment was carried out based on real robot data taken from the MIT Killian Court dataset. Since we were unable to actively control the robot during the experiment at the Killian Court, we had to set the backup and restore points manually. The corresponding positions are labeled are depicted in Figure 8.4.

Our experiments show that our recovering technique is a powerful extension to autonomous exploration with mapping systems based on RBPFs especially in the context of (multiple) nested loops. Note that in general the success rate of the standard approach increases with number of particles used. Since each particle carries its own map, it is of utmost importance to keep this value as small as possible.

Additionally, we analyzed in our experiments the approximation error obtained by retrospectively recovering the particles at the entry point of a loop. Using this system without adaptive resampling, we observed that in our experiments typically around 75% of the particles in the filter at time step t_e had a successor in the current set and were therefore saved. In principle, this value drops for loops of increasing length. To provide a more quantitative comparison, we computed th Kullback-Leibler divergence (KL-divergence) between the recovered particle set and the true one. The KL-divergence between to probability distributions p and q is defied as

$$KLD(p,q) = \sum_x p(x) \log \frac{p(x)}{q(x)}. \tag{8.5}$$

In out experiments the KLD at time step t_e was typically between 1.0 and 1.5 compared to a value around 13 in the situation in which only a single hypothesis survived.

We then activated the adaptive resampling approach that carries out the resampling step only if the effective sample size was smaller than $N/2$, where N is the number of samples. As a result, the number of resamplings carried out in the whole experiment was comparably small. We did not observed more than one resampling step between the time t_e and t. The KL-divergence in this second groups of experiments was around one order of magnitude smaller compared to the set of experiments carried out without adaptive resampling.

The experiments presented in this section illustrate that our recovering technique is well-suited to propagate the uncertainty of trajectory hypotheses through a loop during Rao-Blackwellized mapping. Using the technique

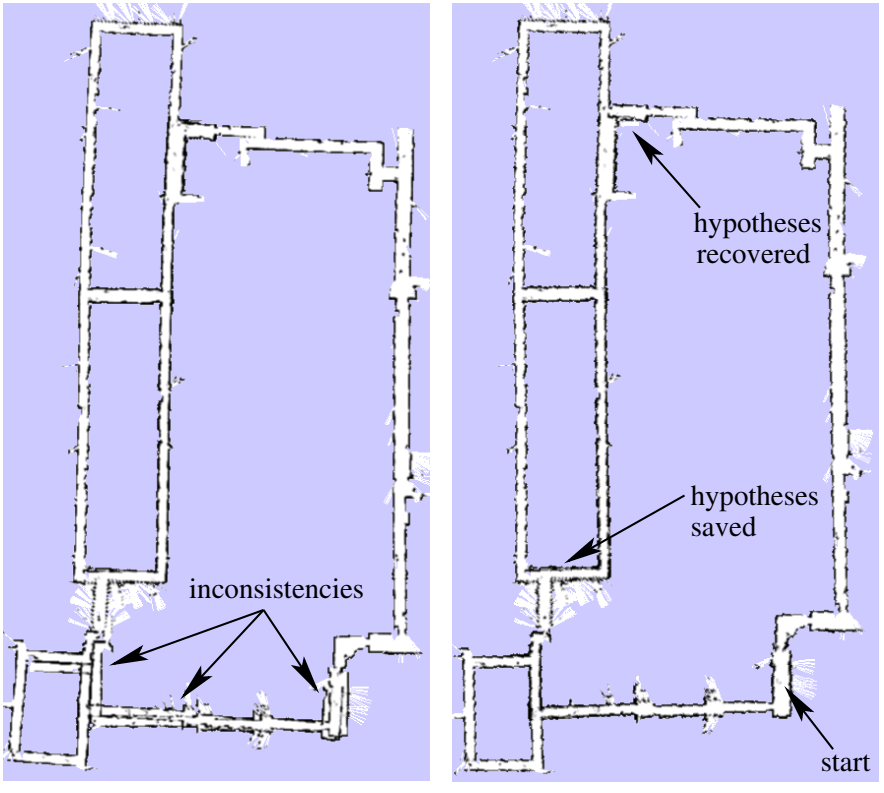


Fig. 8.4. This figure shows two maps of the Killian Court at the MIT. The size of the environment is 150 m by 80 m. The left map was constructed with the standard RBPF approach. If, in contrast, the robot is able to recover hypotheses the map becomes more accurate (right image).

described here, the robot can move arbitrarily long through a (nested) loop without depleting important state hypotheses.

8.4 Related Work

Most of the related work relevant for this chapter, has already been discussed in Section 7.4. Most of these papers focus on reducing the uncertainty during landmark-based SLAM or do not take into account the pose uncertainty in the context of grid-based exploration.

In the literature, only a few works address the problem of revoking a previously made decision in the SLAM context. For example, Hähnel *et al.* [60] maintain a data association tree in which each branch represents a sequence of associations. Whenever a branch becomes more likely than the current best

one, their approach switches to the alternative data association sequence. Their work can be regarded as orthogonal to our technique for recovering the uncertainty of a particle filter presented in this chapter. In fact, both approaches can be combined.

Particle depletion leads to problems similar to the one of an overly confident filters in the context of extended Kalman filters or sparse extended information filters (SEIFs). Especially the SEIF formulation of Thrun *et al.* [150] can lead to underestimated landmark covariance matrixes. Recently, Eustice *et al.* [39] as well as Walter *et al.* [157] presented a technique to more accurately compute the error-bounds within the SEIF framework and in this way reduces the risk of becoming overly confident.

Our approach presented here extends our work described in Chapter 7 and presents a way to recover particle diversity when applying a Rao-Blackwellized particle filter to solve the SLAM problem. Our technique allows the robot to stay – at least in theory – arbitrarily long within a loop without suffering from particle deletion. Therefore, our algorithm enhances the ability to correctly close loops, especially, in the context of nested loops.

8.5 Conclusion

In this chapter, we presented an extension of our loop-closing technique introduced in Chapter 7. Our approach is able to maintain the particle diversity while actively closing loops for mapping systems based on Rao-Blackwellized particle filters. When closing a loop, our approach determines an approximation of the particle set at the time the robot entered the loop. It uses this posterior to propagate the particle diversity through the loop after the robot successfully closed it. Compared to our previous approach which used a heuristic stopping criterion to abort the loop-closing, the technique presented here allows the robot to traverse a nested loop for an arbitrary period of time without depleting important particles. The approach has been implemented and tested on real robot data as well as in simulation. As experimental results demonstrate, we obtain a robust exploration algorithm that produces more accurate maps compared to standard combinations of SLAM and exploration approaches, especially in the context of nested loops.

9

Information Gain-based Exploration

9.1 Introduction

So far, we investigated different aspects of the map learning problem. We started in Chapter 3 with an information gain-based approach to exploration, where we assumed that the poses of the robot were known during exploration. After dealing with the problem of coordinating a team of robots, we addressed the SLAM problem to find a way to deal with the pose uncertainty of a mobile robot. We then presented in the previous two chapters an exploration system that takes into account the pose uncertainty and carries out loop-closing actions in order to relocalize the robot. This has been shown to provide better maps than exploration approaches focusing on new terrain acquisition only.

This chapter describes a decision-theoretic, uncertainty-driven approach to exploration which combines most of the previously presented techniques. We use a decision-theoretic framework similar to the one presented in the beginning of this book. However, we now reason about sequences of observations and not only about a single one. Furthermore, we integrate our SLAM approach in order to deal with the pose uncertainty of the vehicle. This allows us to simulate observations based on the posterior about maps. Last but not least, we consider loop-closing and place revisiting actions during exploration in order to relocalize the vehicle.

As illustrated in Chapter 7, the quality of the resulting map depends on the trajectory of the robot during data acquisition. In practice, the major sources of uncertainty about the state of the world are the uncertainty in the robot's pose and the uncertainty resulting from the limited accuracy of the sensor the robot uses to perceive its environment. Therefore, a robot performing an autonomous exploration task should take the uncertainty in the map as well as in its path into account to select an appropriate action.

As a motivating example consider Figure 9.1. The left image shows an exploring robot which has almost closed a loop. Suppose the vehicle has a high

C. Stachniss: Robotic Mapping and Exploration, STAR 55, pp. 143–160.
springerlink.com

Fig. 9.1. Suppose the robot has a high pose uncertainty and has to decide where to go next. Shown are three opportunities in the left image. Action 1 acquires new terrain and action 2 performs a loop closure without observing unknown areas. Action 3 does both: After closing the loop, it guides the robot to unknown terrain. Our map and pose entropy-driven exploration system presented in this chapter is able to predict the uncertainty reduction in the model of the robot. As a result, it chooses action 3 (as depicted in the right image) since it provides the highest expected uncertainty reduction.

pose uncertainty and now has to decide where to go next. Three potential actions are plotted on the map. Action 1 leads the robot to unknown terrain and action 2 performs a loop closure without observing unknown areas. Action 3 does both: After closing the loop, it guides the robot to unknown terrain.

Classical exploration approaches, which seek to reduce the amount of unseen area or which only consider the uncertainty in the posterior about the map would choose action 1, since this action guides the robot to the closest location from which information about unknown terrain can be obtained. In contrast to that, approaches to active localization consider only the uncertainty in the pose estimate of the robot. Therefore, they would choose either action 2 or 3 to relocalize the vehicle. Our loop-closing approach presented in Chapter 7 would select action 2 to reduce the entropy in the posterior about potential trajectories. However, the best action to reduce the uncertainty in the posterior about the map and the trajectory is action 3. Executing this action yields new sensor information to make the correct data association and closes the loop accurately. Additionally, it provides information about so far unknown terrain. As this example shows, exploration approaches should consider both sources of uncertainty to efficiently build accurate maps.

The contribution of this chapter is an integrated technique that combines simultaneous localization, mapping, and path planning. In contrast to our previous work described in Chapter 7, in which a heuristic was used to retraverse loops, the approach presented in this chapter is entirely decision-theoretic. Based on the expected uncertainty reduction in the posterior about the trajectory of the robot as well as about the map of the environment, we

select the action with the highest expected information gain. We take into account the sensor information, which is expected to be obtained along the path when carrying out an action, as well as the cost introduced by this action. Real world and simulation experiments show the effectiveness of our technique for autonomously learning accurate models of the environment.

This chapter is organized as follows. Section 9.2 and 9.3 present our decision-theoretic exploration technique and explain how to compute the expected change in entropy. Section 9.4 describes how the set of possible actions is generated. Then, Section 9.5 contains experimental results carried out on real robots as well as in simulation. Finally, we discuss related work.

9.2 The Uncertainty of a Rao-Blackwellized Mapper

In this approach to information gain-based exploration, we use the SLAM approach presented in Chapter 6 to estimate the pose of the vehicle as well as the map. The goal of our exploration task is to minimize the uncertainty in the posterior of the robot. The uncertainty can be determined by the entropy H. For the entropy of a posterior about two random variables x and y holds

$$
\begin{aligned}
& H(p(x,y)) \\
&= E_{x,y}[-\log p(x,y)] && (9.1)\\
&= E_{x,y}[-\log(p(x)p(y \mid x))] && (9.2)\\
&= E_{x,y}[-\log p(x) - \log p(y \mid x))] && (9.3)\\
&= E_{x,y}[-\log p(x)] + E_{x,y}[-\log p(y \mid x)] && (9.4)\\
&= H(p(x)) + \int_{x,y} -p(x,y) \log p(y \mid x)\ dx\ dy. && (9.5)
\end{aligned}
$$

The integral in (9.5) can be transformed as follows:

$$
\begin{aligned}
& \int_{x,y} -p(x,y) \log p(y \mid x)\ dx\ dy \\
&= \int_{x,y} -p(y \mid x)p(x) \log p(y \mid x)\ dx\ dy && (9.6)\\
&= \int_x p(x) \int_y -p(y \mid x) \log p(y \mid x)\ dy\ dx && (9.7)\\
&= \int_x p(x) H(p(y \mid x))\ dx && (9.8)
\end{aligned}
$$

Equation (9.5) and (9.8) can be combined to

$$
H(p(x,y)) = H(p(x)) + \int_x p(x) H(p(y \mid x))\ dx. \qquad (9.9)
$$

Based on (9.9), we can efficiently compute the entropy of a Rao-Blackwellized particle filter for mapping. For better readability, we use d_t instead of $z_{1:t}, u_{1:t-1}$:

$$H(p(x_{1:t}, m \mid d_t)) = H(p(x_{1:t} \mid d_t)) + \int_{x_{1:t}} p(x_{1:t} \mid d_t) H(p(m \mid x_{1:t}, d_t))\, dx_{1:t} \tag{9.10}$$

Considering that our posterior is represented by a set of weighted particles, we can approximate the integral by a finite sum:

$$H(p(m, x_{1:t} \mid d_t)) \simeq H(p(x_{1:t} \mid d_t)) + \sum_{i=1}^{\#particles} w_t^{[i]} H(p(m^{[i]} \mid x_{1:t}^{[i]}, d_t)) \tag{9.11}$$

Equation (9.11) shows that according to the Rao-Blackwellization, the entropy of the whole system can be divided into two components. The first term represents the entropy of the posterior about the trajectory of the robot and the second term corresponds to the uncertainty in the map weighted by the likelihood of the corresponding trajectory. Thus, to minimize the robot's overall uncertainty, one needs to reduce the map uncertainty of the individual particles as well as the trajectory uncertainty. In this section, we will describe how we determine both terms in our approach.

Throughout this work, we use grid maps to model the environment. Note that our technique is not restricted to this kind of representation, it only requires a way to compute the uncertainty for the used map representation. Using occupancy grids, the computation of the map entropy is straightforward. According to the common independence assumption about the cells of such a grid, the entropy of a map m is the sum over the entropy values of all cells. Since each grid cell c is represented by a binary random variable the entropy of m is computed as

$$H(m) = -\sum_{c \in m} p(c) \log p(c) + (1 - p(c)) \log(1 - p(c)). \tag{9.12}$$

Note that the overall entropy calculated for a grid map is not independent from the resolution of the grid. One potential solution to this problem is to weight the entropy of each cell with its covered area r^2 (where r is the resolution of the grid)

$$H(m) = -r^2 \sum_{c \in m} p(c) \log p(c) + (1 - p(c)) \log(1 - p(c)). \tag{9.13}$$

As a result, the entropy value stays more or less constant when changing the grid resolution. Slight differences in the entropy may be caused by discretization errors when changing the resolution.

Unfortunately, it is more difficult to compute the uncertainty $H(p(x_{1:t}|d_t))$ of the posterior about the trajectory of the robot since each pose x_t on the trajectory depends on the previous locations $x_{1:t-1}$. In the context of EKF-based exploration approaches, the pose uncertainty is often calculated by considering only the last pose of the robot, which corresponds to the approximation of $H(p(x_{1:t}|d_t))$ by $H(p(x_t|d_t))$. It is also possible to average over the uncertainty of the different poses along the path as done by Roy *et al.* [126]:

$$H(p(x_{1:t} \mid d_t)) \approx \frac{1}{t} \sum_{t'=1}^{t} H(p(x_{t'} \mid d_t)) \tag{9.14}$$

Instead, one can approximate the posterior about the trajectory by a high-dimensional (length of the trajectory times the dimension of the pose vector x_t of the robot) Gaussian distribution. The entropy of a n dimensional Gaussian $\mathcal{N}(\mu, \Sigma)$ is computed as

$$H(\mathcal{N}(\mu, \Sigma)) = \log((2\pi e)^{(n/2)}|\Sigma|). \tag{9.15}$$

Since a finite number of particles is used, the RBPF representation often generates a sparse trajectory posterior for points in time lying further back in the history. Unfortunately, this can lead to a reduced rank of Σ, so that $|\Sigma|$ becomes zero and the entropy $H(\mathcal{N}(\mu, \Sigma))$ approaches minus infinity.

Alternatively, one could consider the individual trajectories represented by the samples as vectors in a high-dimensional state space and compute the entropy of the posterior based on a grid-based discretization. Since the particles typically are extremely sparse, this quantity is in most cases equivalent to or slightly smaller than the logarithm of the number of particles, which is the upper bound for the entropy computed in this way.

In our current implementation, we use an approach that is similar to the one proposed by Roy *et al.* [126], who computed the entropy over the trajectory posterior as the average entropy of the pose posteriors over time, see (9.14). Instead of averaging only over the time steps, we additionally consider the different areas the robots visited. This allows us to give an area traversed only once by the vehicle the same influence than an area the robot visited several times. In our current implementation, the places are modeled by a coarse resolution grid. An example on how the trajectory entropy evolves over time using this measure is depicted in the left image of Figure 9.2.

9.3 The Expected Information Gain

To evaluate an action that guides the robot from its current location to a goal location, we compute the expected information gain, which is the expected change of entropy in the Rao-Blackwellized particle filter. In the last section, we described how to compute the entropy of the robot's world model and

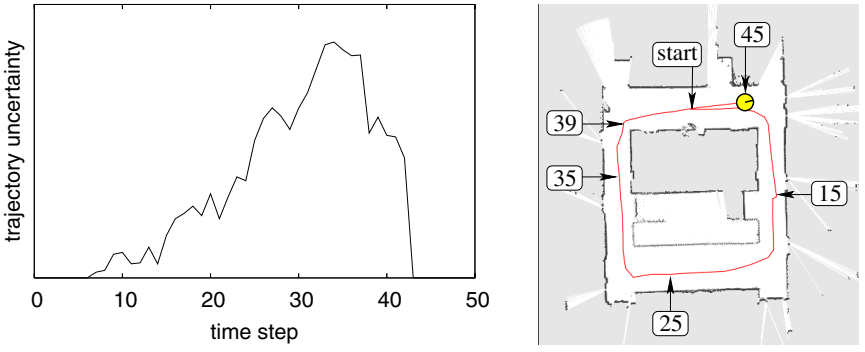


Fig. 9.2. The trajectory entropy of a robot during a real world experiment. The numbers in the right image illustrate the time steps when the robot was at the corresponding locations.

in this section we want to estimate the expected entropy after an action has been carried out.

An action a_t generated at time step t is represented by a sequence of relative movements $a_t = \hat{u}_{t:T-1}$ (see Section 7.2.2). During the execution of a_t, it is assumed that the robot obtains a sequence of observations $\hat{z}_{t+1:T}$ at the positions $\hat{x}_{t+1:T}$. In the following, all variables labeled with '$\hat{}$' correspond to points in time during the execution of an action a_t. For better readability, we replace $\hat{x}_{t+1:T}$ by $\hat{x}$ and $\hat{z}_{t+1:T}$ by $\hat{z}$.

To compute the information gain of an action, we have to calculate the change of entropy caused by the integration of $\hat{z}$ and a_t into the filter

$$I(\hat{z}, a_t) = H(p(m, x_{1:t} \mid d_t)) - H(p(m, x_{1:t}, \hat{x} \mid d_t, a_t, \hat{z})). \tag{9.16}$$

Since in general we do not know which measurements the robot will obtain along its path while executing action a_t, we have to integrate over all possible measurement sequences $\hat{z}$ to compute the expected information gain

$$E[I(a_t)] = \int_{\hat{z}} p(\hat{z} \mid a_t, d_t) I(\hat{z}, a_t)\, d\hat{z}. \tag{9.17}$$

In the following, we will explain how to approximate $p(\hat{z} \mid a_t, d_t)$ in order to reason about possible observation sequences $\hat{z}$. The posterior $p(\hat{z} \mid a_t, d_t)$ can be transformed into

$$\begin{aligned} & p(\hat{z} \mid a_t, d_t) \\ &= \int_{m, x_{1:t}} p(\hat{z} \mid a_t, m, x_{1:t}, d_t) p(m, x_{1:t} \mid d_t)\, dm\, dx_{1:t} && (9.18) \\ &= \int_{m, x_{1:t}} p(\hat{z} \mid a_t, m, x_{1:t}, d_t) p(x_{1:t} \mid d_t) p(m \mid x_{1:t}, d_t)\, dm\, dx_{1:t}. && (9.19) \end{aligned}$$

Equation (9.19) is obtained from (9.18) by using (6.1). If we again assume that our posterior is represented by a set of particles, we can rewrite (9.19) as follows:

$$p(\hat{z} \mid a_t, d_t) \approx \sum_{i=1}^{\#particles} p(\hat{z} \mid a_t, m^{[i]}, x_{1:t}^{[i]}, d_t) w_t^{[i]} p(m^{[i]} \mid x_{1:t}^{[i]}, d_t) \quad (9.20)$$

Based on (9.20), we can compute $\hat{z}$ for a given action a_t. The factor $p(m^{[i]} \mid x_{1:t}^{[i]}, d_t)$ in (9.20) is assumed to be computed analytically due to the assumptions made in the Rao-Blackwellization, see (6.1), namely that we can compute the map $m^{[i]}$ analytically given the positions $x_{1:t}^{[i]}$ as well as the data d_t. We can also estimate the term $p(\hat{z} \mid a_t, d_t)$ of that equation by simulation. This can be achieved by performing ray-casting operations in the map $m^{[i]}$ of the i-th particle to estimate possible observations $\hat{z}$. In other words, the (discretized) posterior about possible observations obtained along the path when executing the action a_t can be computed by ray-casting operations performed in the map of each particle weighted by the likelihood of that particle.

In cases where the ray-casting operation reaches an unknown cell in the map, we have to treat the beam differently. Touching an unknown cell means that we cannot say anything about the beam except that its length will be at least as long as the distance between robot pose and the unknown cell (with a high probability). Since such beams typically have a serious influence on the map uncertainty, we computed statistics about the average change of map entropy introduced by integrating a beam which reaches an unknown cell in the map. One example for such a statistics from recorded laser range data is shown in Figure 9.3. Note that in this situation, the change of entropy is approximatively proportional to the number of unknown cells covered by that beam. By computing the average beam length for such sensor observations from the statistics, one can predict the average change of entropy when approaching a frontier. In this way, the system also accounts for unknown areas which are visible from a planned path to any other destination.

This approximation dramatically reduces the number of potential observations that have to be simulated compared to the number of possible proximity measurements a laser range finder can generate. Several experiments showed the effectiveness of this approach for robots equipped with a laser range finder.

Despite this approximation, computing the expected information gain based on (9.17) requires a substantial amount of computational resources. Therefore, we furthermore approximate the posterior in this equation about possible sensory data, by not considering all possible map instances of the current posterior. We apply the computations only on a subset of potential maps. This subset is obtained by drawing particles v_i from the particle set, where each particle is drawn with a probability proportional to its weight. We then use the map associated to v_i to generate the measurements $\hat{z}(v_i)$ along the path. This reduces the computational complexity and allows us to run the exploration system on a real robot. Under this simplifying assumption,

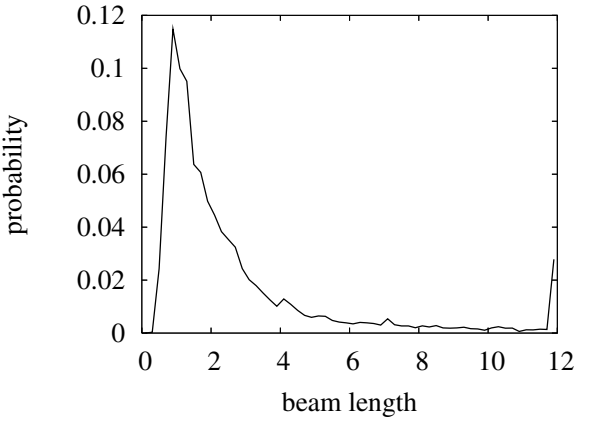


Fig. 9.3. The plot shows the likelihood of a laser beam that covers an unknown cell based on recorded sensor data. In this plot, a beam length of 12 m represents a maximum range reading.

we can rewrite the expected information gain in (9.17) by

$$E[I(a_t)] \approx \frac{1}{n} \sum_i I(\hat{z}(v_i), a_t). \tag{9.21}$$

where n is the number of drawn samples. An observation sequence $\hat{z}(v_i)$ is generated by a ray-casting operation in the map of v_i. Note that if more computational resources are available this approximation can easily be improved by drawing more particles. This computation can even be parallelized, since there is no interference between the integration of measurement sequences into different copies of the RBPF.

Now all necessary equations have been introduced to compute the expected information gain $E[I(a_t)]$ for an action a_t. To summarize, $E[I(a_t)]$ describes the expected change of entropy in the Rao-Blackwellized particle filter when executing a_t. To reason about possible observation sequences, the robot will obtain along the path, we draw a subset of particle according to their likelihood and perform a ray-casting operation in the corresponding maps. The expected measurements are then integrated into the filter and the entropies before and after the integration are subtracted.

The complexity of the computation of $E[I(a_t)]$ depends on two quantities. First, the filter needs to be copied to save its current state. This introduces a complexity linear in the size of the filter (which in turn depends on the number of particles). The second quantity is the length $l(a_t)$ of the planned path from the current pose of the robot to the desired goal location, because the expected observations along the path are taken into account. The number of particles drawn to generate observations is assumed to be constant. The

cost of integrating an observation is linear in the number N of particles. This leads to an overall complexity of $O(l(a_t)N)$ to evaluate an action a_t.

Besides the expected entropy reduction, there is a second quantity the robot should consider when selecting an action. This is the cost of carrying out an action measured in terms of traversability and trajectory length for reaching the target location. The cost of an action is computed based on the (convolved) occupancy grid map of the most likely particle. Traversing a cell introduces a cost proportional to its occupancy probability (see Section 4.2.1 for further details).

The expected utility $E[U(a_t)]$ of an action a_t in our exploration system is defined as

$$E[U(a_t)] = E[I(a_t)] - \alpha V(a_t). \tag{9.22}$$

where $V(a_t)$ refers to the cost of carrying out the action a_t. α is a weighting factor which trades off the cost with the entropy. This free parameter can be used to trigger the exploration process by adapting the influence of the traveling cost. In our work, we determined α experimentally.

After having computed the expected utility for each action under consideration, we select the action a_t^* with the highest expected utility

$$a_t^* = \underset{a_t}{\operatorname{argmax}} E[U(a_t)]. \tag{9.23}$$

Every time the robot has to make the decision where to go next, it uses (9.23) to determine the action a_t^* with the highest expected utility and executes it. As soon as no action provides an expected reduction of uncertainty and no frontiers to unseen areas are available, the exploration task is completed.

9.4 Computing the Set of Actions

So far, we have explained how to evaluate an action but have left open how potential actions are generated. One attempt might be to generate a vantage point for each reachable grid cell in the map. Since we reason about observations received along the path, we need to consider all possible trajectories to all reachable grid cells in the map. The number of possible trajectories, however, is huge which makes it intractable to evaluate all of them.

To find appropriate actions to guide a vehicle through the environment, we consider three types of actions, so called *exploration actions*, *place revisiting actions*, and *loop-closing actions*. Exploration actions are designed to acquire information about unknown terrain to reduce the map uncertainty. To generate exploration actions, we apply the frontier approach introduced by Yamauchi [163]. For each frontier between known and unknown areas, we generate an action leading the robot from its current pose along the shortest

path to that frontier. Furthermore, actions that guide a robot to cell which have a high uncertainty belong to the set of exploration actions.

Compared to the actions generated from frontiers, the place revisiting actions as well as the loop-closing actions do not focus on new terrain acquisition. They guide the robot back to an already known location or perform an active loop closure. The goal of these actions is to improve the localization of the vehicle, which means to reduce its trajectory uncertainty. In our current implementation, place revisiting actions are generated based on the trajectory of the robot. Such an action can simply turn the robot around and move it back along its previously taken path. Additionally, we generate so called loop-closing actions. To determine whether there exists a possibility to close a loop, we would like to refer the reader to Chapter 7 in which we describe how a mobile robot can robustly detect opportunities to actively close a loop.

Given this classification, the actions 1 and 3 depicted in the motivation example in Figure 9.1 are exploration actions, whereas action 2 is a place revisiting action performing an active loop closure.

9.5 Experimental Results

Our approach has been implemented and tested in real world and simulation experiments. The experiments described here are designed to illustrate the benefit of our exploration technique which takes into account the map as well as the trajectory uncertainty to evaluate possible actions.

9.5.1 Real World Application

The first experiment was a real world experiment carried out in building 106 at the University of Freiburg using an ActivMedia Pioneer 2 robot equipped with a SICK laser range finder. The exploration run was fully autonomous. The robot started in the lower left room (see Figure 9.4 (a)). The robot moved through the neighboring room and entered the corridor. After reaching its target location in the horizontal corridor (Figure 9.4 (b)), the robot decided to move back to in the lower left room to improve its pose estimate (Figure 9.4 (c)). The robot then explored the neighboring room and afterwards returned to the corridor (Figure 9.4 (d)). It then approached the lower horizontal corridor and moved around the loop (Figure 9.4 (e)). Finally, the robot returned to the lower left room and finished the exploration task. As can be seen from this experiment, as soon as the robot gets too uncertain about its pose, it performs place revisiting actions or chooses exploration actions which also reduce its pose uncertainty due to the information gathered along the path.

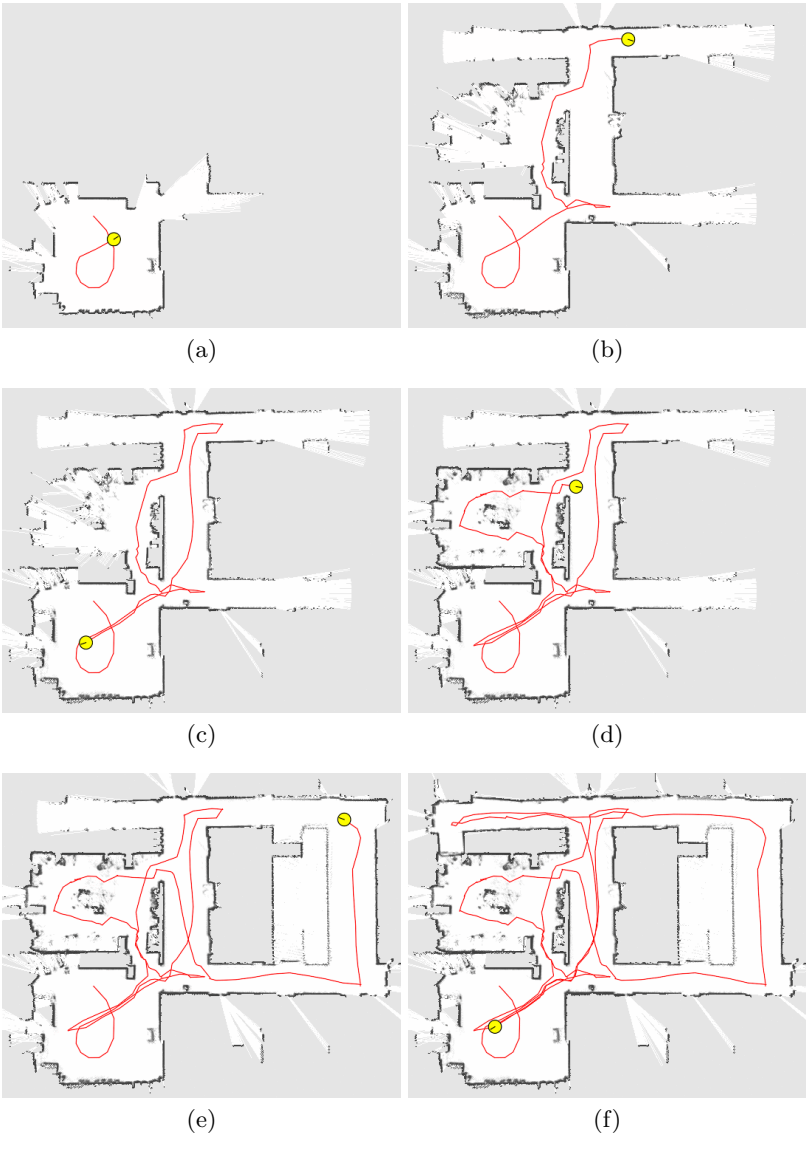


Fig. 9.4. Six different stages of an autonomous exploration run on the second floor of building 106 at the University of Freiburg. The map was acquired fully autonomously by our integrated approach.

9.5.2 Decision Process

The next experiment is designed to show how the robot chooses actions to reduce its pose uncertainty as well as its map uncertainty. Figure 9.5 depicts parts of a simulated exploration task performed in a map acquired at Sieg Hall, University of Washington. Each row depicts a decision step of the robot during autonomous exploration. In the first step shown in the first row, the robot has almost closed the loop. It had to decide whether it is better to move through the loop again or to focus on exploring the horizontal corridor. In this situation, the robot moved to target point 1 and actively closed the loop, since this provided the highest expected utility (see right plot in the first row of Figure 9.6). Target location 1 had the highest expected utility because the robot expected a chance to relocalize itself by closing the loop and to observe parts of the unknown areas close to the planned trajectory. Therefore, this actions provided an expected reduction of map and trajectory uncertainty. In the second decision, the robot focused on acquiring new terrain and approached the horizontal corridor, since target location 6 had the highest expected utility. The same happened in the third decision step, shown in the last row of this figure. Moving back through the known areas of the loop provided less expected entropy reduction and therefore the robot continued exploring the horizontal corridor (target location 5).

Figure 9.7 shows the map after reaching target location 5 from the last decision step. To visualize the change of entropy over time, the right plot shows the evolution of the map as well as the pose uncertainty. The labels in the left image show the time steps in which the robot was at the corresponding location. As can be seen, the entropy stayed more or less constant in the beginning, since the map uncertainty decreased while the pose uncertainty increased. After closing the loop at around time step 45, the pose uncertainty dropped so that the overall uncertainty was also reduced. Moving through known areas between time step 50 and 80 did not provide a lot of new information and did not change the entropy that much. As soon as the robot entered especially the wide part of the horizontal corridor, the overall uncertainty dropped again due to the serious reduction of map uncertainty compared to the moderate increase of pose uncertainty.

9.5.3 Comparison to Previous Approaches

The third experiment addresses the decision problem of the motivating example presented in the introduction of this chapter. It shows how our approach chooses the actions which lead to the highest uncertainty reduction in the posterior about poses and maps compared to previous techniques. As can be seen in Figure 9.8, the robot has almost closed the loop. Suppose the robot has a high pose uncertainty and considers three potential actions to approach different target locations (see left image of Figure 9.8). Action 1 is a new terrain acquisition action and action 2 performs a loop closure. Action 3 leads

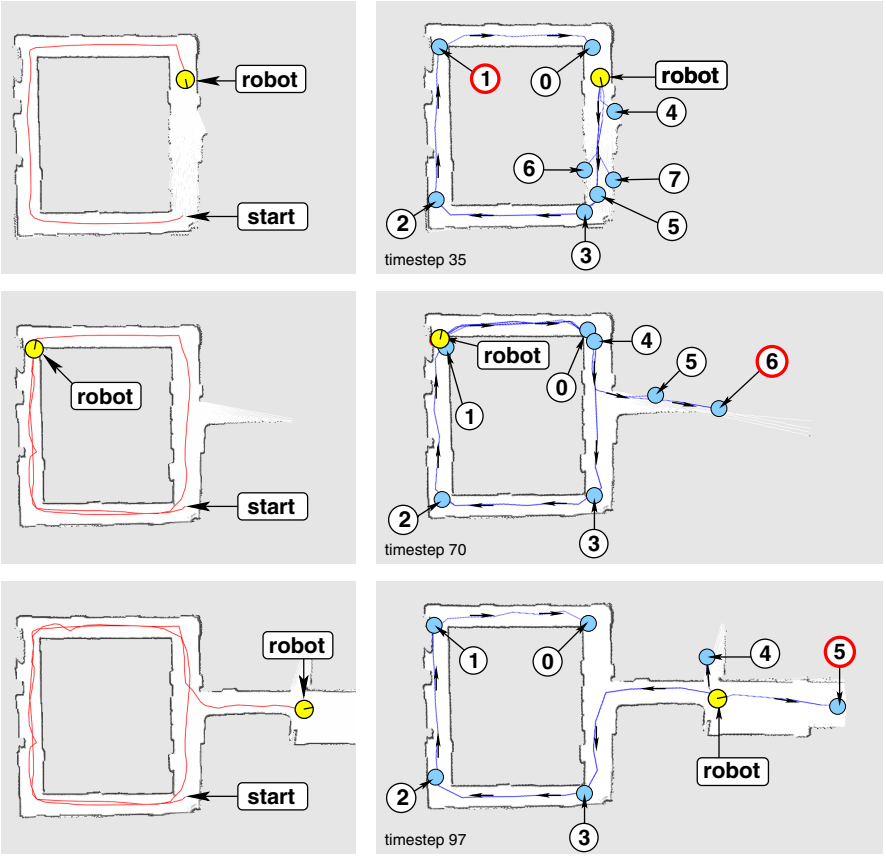


Fig. 9.5. Three points in time in which the robot had to decide where to move next. The left images depict the trajectory of the robot up the corresponding point in time and the right images illustrate the best maps and possible actions. Figure 9.6 depicts the corresponding utilities. The chosen target locations are marked red.

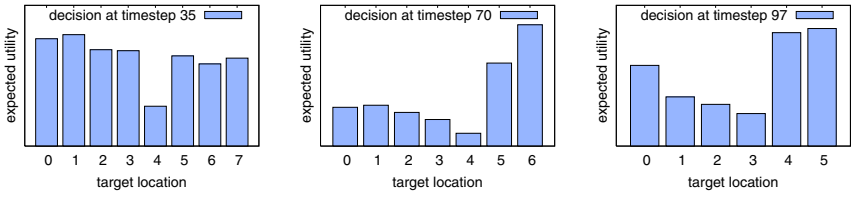
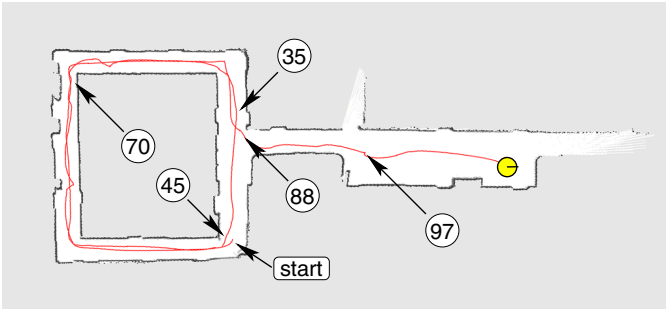


Fig. 9.6. The corresponding utilities of different target locations given the possible actions depicted in Figure 9.5.



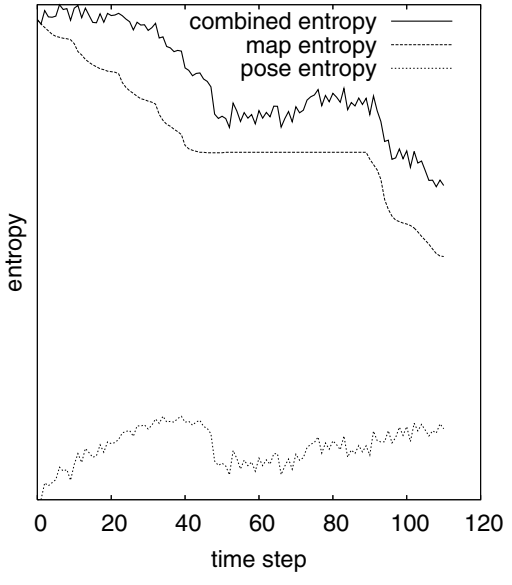


Fig. 9.7. This figure illustrates the evolution of the entropy during the experiment shown in Figure 9.5. The marker in the left image correspond to the different points in time when the robot was at the corresponding location. The right plot depicts the entropy during the data acquisition phase. It depicts the map entropy, the pose uncertainty, and the overall (combined) entropy over time.

the robot to unknown terrain while simultaneously closing the loop. Since action 3 combines a loop closure with new terrain acquisition, it provides the highest expected utility (see right image of Figure 9.8). Therefore, our approach chooses this target point. This is an advantage compared to other approaches which seek to actively close loops in an heuristic way. Such a technique (like the one we presented in Chapter 7) would typically choose action 2 to reduce the pose uncertainty of the vehicle. Classical exploration approaches, which only

take into account the map uncertainty or guide the robot to the closest unknown area [77, 160, 161, 163, 164] would select action 1. Even an active localization technique which seeks to reduce the pose uncertainty of the vehicle [71] would choose either action 2 or 3 (with a 50% chance each).

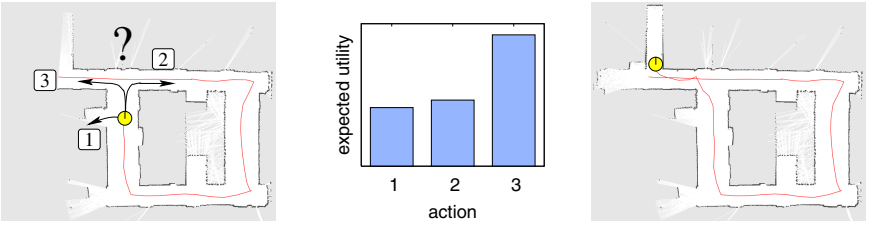


Fig. 9.8. This figure illustrates the decision process of where to go next. Shown in the left image are three potential actions in the left image and the corresponding expected utilities in the middle image. The situation after the robot has chosen action 3 is depicted in the right image.

9.5.4 Corridor Exploration

The last experiment was performed in building 79 at the University of Freiburg and is depicted in Figure 9.9. The environment has a long corridor and contains no loop. To make the pose correction more challenging, we restricted the range of the sensor to 3 m. According to the short sensor range used in this experiment, it was hard for the robot keep track of its own position. As can be seen, this technique leads to an intuitive behavior. Due to the large uncertainty in the pose of the vehicle, the robot chooses several times actions which guide it back to a well-known place (which is the starting location in this experiment) to reduce its pose uncertainty.

9.6 Related Work

In the context of exploration, most of the techniques presented so far focus on generating motion commands that minimize the time needed to cover the whole terrain [77, 160, 163, 17]. Most of these techniques, however, assume that an accurate position estimate is given during exploration. Whaite and Ferrie [161] present an approach that uses also the entropy to measure the uncertainty in the geometric structure of objects that are scanned with a laser range sensor. In contrast to the work described here, they use a parametric representation of the objects to be scanned and do not consider the uncertainty in the pose of the sensor. Similar techniques have been applied

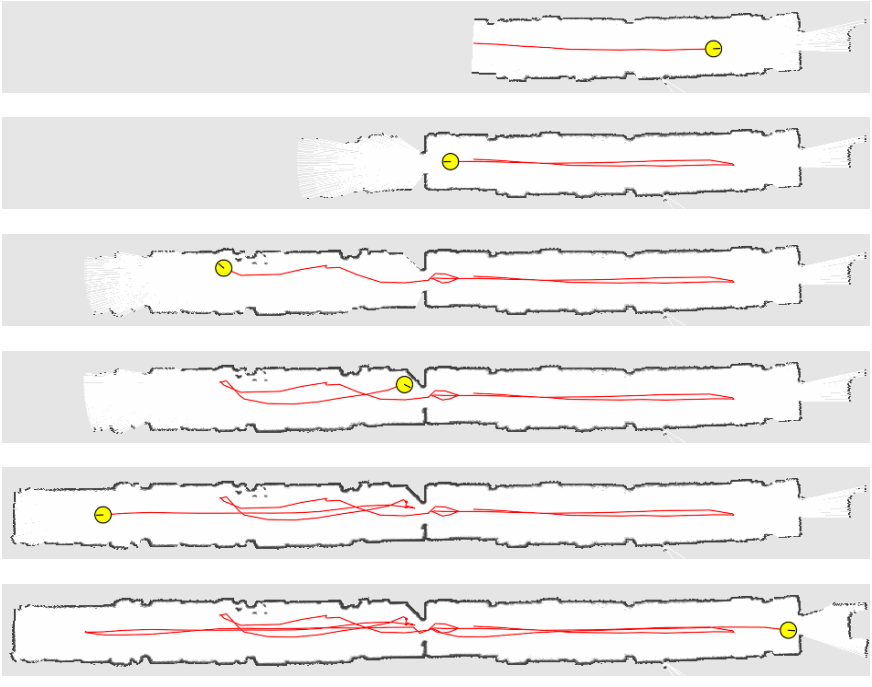


Fig. 9.9. The images depict six stages during the autonomous exploration of a long corridor. The maximum sensor range in this experiment was limited to 3 m. The short sensor range results in a comparably high pose uncertainty of the robot when moving through the environment, since the current scan has typically a small overlap with the previously seen area. Due to the high pose uncertainty, the exploration system chooses actions which guide the robot on a path close to the starting location in order to relocalize.

to mobile robots like, for example, our approach presented in Chapter 3 or the work of Rocha *et al.* [123]. However, none of the approaches mentioned above take the pose uncertainty into account when selecting the next vantage point. There are exploration approaches that have been shown to be robust against uncertainties in the pose estimates [32, 74, 82] but the selected actions do depend on the uncertainty of the system. Note that a detailed discussion about different exploration strategies for single and multi-robot systems has been presented in Section 4.6.

In the area of SLAM, the vast majority of papers have focused on the aspect of state estimation as well as belief representation and update [28, 30, 37, 56, 57, 59, 104, 109, 145]. These techniques are passive and only process incoming sensor data without explicitly generating control commands. Again, Section 6.6 presents a detailed discussion of SLAM approaches. In contrast

to these techniques, our approach considers the active control of the robot while learning accurate maps.

Recently, new techniques have been proposed which actively control the robot during SLAM. For example, Makarenko *et al.* [94] as well as Bourgault *et al.* [15] extract landmarks out of laser range scans and use an extended Kalman filter (EKF) to solve the SLAM problem. They furthermore introduce a utility function which trades off the cost of exploring new terrain with the expected reduction of uncertainty by measuring at selected positions. A similar technique has been applied by Sim *et al.* [135], who consider actions to guide the robot back to a known place in order reduce the pose uncertainty of the vehicle. These three techniques differ from the approach presented in this chapter in that they rely on the fact that the environment contains landmarks that can be uniquely determined during mapping. In contrast to this, our approach makes no assumptions about distinguishable landmarks and uses raw laser range scans to compute accurate grid maps. One disadvantage of feature based exploration systems is that the underlying models of the environment typically do not provide any means to distinguish between known an unknown areas. Therefore, an additional map representation needs to be maintained (like, e.g., an occupancy grid in [15, 94] or a visual map in [135]) to efficiently guide the vehicle. Approaches which do not maintain an additional model to identify unknown areas typically apply strategies in which the robot follows the contours of obstacles [162] or performs wall following combined with random choices at decision points [41].

Duckett *et al.* [32] use relaxation to solve the SLAM problem in their exploration approach. They condense local grid maps into graph nodes and select goal points based on that graph structure, but do not consider the expected change of uncertainty when choosing possible target locations.

In Chapter 7, we presented an approach to mobile robot exploration that is able to deal with pose uncertainty and seeks for opportunities to active close loops. Those loop-closing actions are used to relocalize the vehicle in order to reduce the uncertainty in the pose estimate. As we demonstrated, such an approach leads to better maps in the context of (nested) loops. The work presented in this chapter extents our loop-closing technique and is entirely decision-theoretic. It reasons about carrying out different types of actions, including loop-closing action, and selects the one which provides the highest expected uncertainty reduction considering also the cost of an action. Our active loop-closing approach can therefore be regarded as a component integrated in the technique presented in this chapter.

There are planning techniques that can compute optimal plans by maintaining a belief over possible states of the world and by computing the strategy that is optimal in expectation with respect to that belief. One solution to this is the partially observable Markov decision process, also known as POMDP [72]. The major disadvantage of POMDPs are their extensive computational cost and most solutions are not applicable to scenarios with more than around one thousand states [117]. Since we reason about a high-dimensional state

estimation problem, we have to be content with approximative solutions that rely on strong assumptions. In essence, our approach can be regarded as an approximation of the POMDP with an one step look-ahead.

Compared to the approaches discussed above, the novelty of the work reported here is that the algorithm for acquiring grid maps simultaneously considers the trajectory and map uncertainty when selecting an appropriate action. We furthermore reason about the information gathered by the sensor when the robot executes an action. Our approach also considers different types of actions, namely so-called exploration actions, which guide the robot to unknown areas and place revisiting actions as well as loop-closing actions, which allow the robot to reliably close loops and this way reduce its pose uncertainty.

9.7 Conclusion

In this chapter, we presented an integrated approach which simultaneously addresses mapping, localization, and path planning. We use a decision-theoretic framework related to the one presented in Chapter 3 for exploration. To deal with the noise in the position of the robot, we applied a Rao-Blackwellized particle filter presented in Chapter 6 to build accurate grid maps. Our exploration approach considers different types of actions, namely exploration actions forcing terrain acquisition as well as place revisiting and active loop-closing actions that reduce the robot's pose uncertainty. These actions are generated based on the active loop-closing technique presented in Chapter 7. By estimating the expected entropy of the particle filter after carrying out an action, we are able to determine the action which promises the highest expected uncertainty reduction, thereby taking potential measurements gathered along the whole path into account. The simulation of observations is done based on the posterior about the map.

We furthermore showed how the uncertainty in a Rao-Blackwellized particle filter can be separated into two components: The uncertainty in the trajectory estimate and the uncertainty in the individual maps weighted with the likelihood of the corresponding particle. Our approach has been implemented and tested on real robots and in simulation. As a result, we obtain a robust decision-theoretic exploration algorithm that produces highly accurate grid maps. In practical experiments, we showed how the robot is able to select the action that provides the highest expected uncertainty reduction in its posterior about poses and maps. This is an advantage compared to exploration approaches that seek to minimize the uncertainty in the map model only and to active localization techniques which consider only the uncertainty in the pose estimate.

10

Mapping and Localization in Non-Static Environments

10.1 Introduction

Throughout all previous chapters of this book, we assumed that the environment does not change over time. This assumption however is not realistic especially for environments populated by humans. People typically walk around, open and close doors, add or remove things, or even move objects like furniture. In the literature, most of the approaches to mapping with mobile robots are based on the assumption that the environment is static. As reported by Wang and Thorpe [158] as well as by Hähnel *et al.* [61], dynamic objects can lead to serious errors in the resulting map. A popular technique to deal with non-static environments is to identify dynamic objects and to filter out the range measurements reflected by these objects. Such techniques have been demonstrated to be more robust than traditional mapping approaches. They allow a robot, for example, to filter out walking people or passing cars. Their major disadvantage lies in the fact that the resulting maps only contain the static aspects of the environment.

Avoiding that walking people or moving cars leave spurious objects in the map is a desirable feature. However, there exist also non-static objects for which is makes sense to integrate them into the model of the environment. As an example, consider open and closed doors which can be classified as low-dynamic or non-static objects that do not move randomly.

In this chapter, we explore an alternative solution to deal with dynamic environments by explicitely modeling the low-dynamic or quasi-static states. Our approach is motivated by the fact, that many dynamic objects appear in only a limited number of possible configurations. As an example, consider the doors in an office environment, which are often either open or closed. Another scenario is cars in a parking space. Most of the time, a parking space is either occupied by a car or is empty. In such a situation, techniques to filter out dynamic objects produce maps which do not contain doors or parked cars at all. This can be problematic since, for example, in many corridor environments doors are important features for localization. The explicit knowledge about

C. Stachniss: Robotic Mapping and Exploration, STAR 55, pp. 161–175.
springerlink.com

the different possible configurations can improve the localization capabilities of a mobile robot. Therefore, it is important to integrate such information into the model of the environment. Our framework presented in this chapter allows that highly dynamic objects can be filtered so that they do not leave spurious objects in the map. This can be achieved by applying a filtering technique like the one of Hähnel *et al.* [62] in a slightly modified way.

As a motivating example consider the individual local maps depicted in Figure 10.1. These maps correspond to typical configurations of the same place and have been learned by a mobile robot operating in an office environment. They show the same part of a corridor including two doors and their typical states. The key idea of our work is to learn such local configurations and to utilize this information to improve the localization accuracy of the robot.

The contribution of this chapter is a novel approach to mapping in low-dynamic environments. Our algorithm divides the entire map into several sub-maps and learns for each of these sub-maps typical configurations for the corresponding part of the environment. This is achieved by clustering local grid maps. Furthermore, we present an extended Monte-Carlo localization algorithm, which uses these clusters in order to simultaneously estimate the current state of the environment and the pose of the robot. Experiments demonstrate that our map representation leads to an improved localization accuracy compared to maps lacking the capability to model different configurations of the environment.

This chapter is organized as follows. First, we introduce our mapping technique that models different configurations of non-static objects in Section 10.2. We then present our variant of Monte Carlo localization that estimates the pose of the vehicle as well as the state of the environment at the same time. In Section 10.4, we present a series of experiments using our technique for mapping and localization in non-static worlds. Finally, Section 10.5 discussed related approaches.

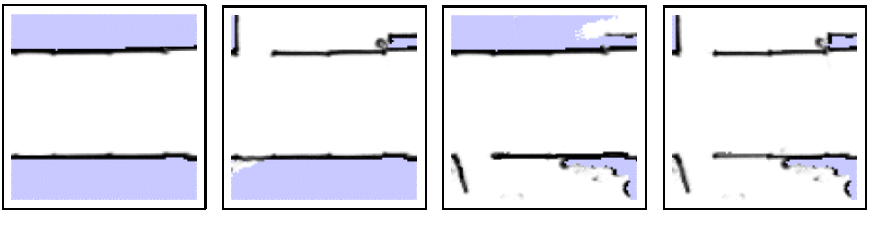


Fig. 10.1. Possible states of the same local area. The different configurations correspond to open and closed doors within a corridor.

10.2 Learning Maps of Low-Dynamic Environments

The key idea of our approach is to use the information about changes in the environment during data acquisition to estimate possible spatial configurations and store them in the map model. To achieve this, we construct a sub-map for each area in which dynamic aspects have been observed. We then learn clusters of sub-maps that represent possible spacial states in the corresponding areas.

10.2.1 Map Segmentation

In general, the problem of learning maps in dynamic environments is a high-dimensional state estimation problem. A naïve approach could be to store an individual map of the whole environment for each potential state. Obviously, using this approach, one would have to store a number of maps that is exponential in the number of dynamic objects. In real world situations, the states of the objects in one room are often independent of the states of the objects in another room. Therefore, it is reasonable to marginalize the local configurations of the individual objects.

Our algorithm segments the environment into local areas, called sub-maps. In this chapter, we use rectangular areas which inclose locally detected dynamic aspects to segment the environment into sub-maps. For each sub-map, the dynamic aspects are then modeled independently.

Note that in general the size of these local maps can vary from the size of the overall environment to the size of each grid cell. In the first case, we would have to deal with the exponential complexity mentioned above. In the second case, one heavily relies on the assumption that neighboring cells are independent, which is not justified in the context of dynamic objects.

In our current system, we first identify positions in which the robot perceives contradictory observations which are typically caused by dynamic elements. Based on a region growing technique, we determine areas which inclose dynamic aspects. By taking into account visibility constraints between regions, they are merged until they do not exceed a maximum sub-map size (currently set to 20m^2). This limits the number of dynamic objects per local map and in this way leads to a tractable complexity. Notice that each sub-map has an individual size and different sub-maps can also overlap.

10.2.2 Learning Configurations of the Environment

To enable a robot to learn different states of the environment, we assume that it observes the same areas at different points in time. We cluster the local maps built from the different observations in order to extract possible configurations of the environment. To achieve this, we first segment the sensor data perceived by the robot into observation sequences. Whenever the robot leaves a sub-map, the current sequence ends and accordingly a new

observation sequence starts as soon as the robot enters a new sub-map. Additionally, we start a new sequence whenever the robot moves through the same area for more than a certain period of time (30s). This results in a set Φ of observation sequences for each sub-map

$$\Phi = \{\phi_1, \ldots, \phi_n\}, \tag{10.1}$$

where each

$$\phi_i = z_{start(i)}, \ldots, z_{end(i)}. \tag{10.2}$$

Here z_t describes an observation obtained at time t. For each sequence ϕ_i of observations, we build an individual occupancy grid for the local area of the sub-map. Such a grid is then transformed into a vector of probability values ranging from 0 to 1 and one additional value ξ to represent an unknown (unobserved) cell. All vectors which correspond to the same local area are clustered using the fuzzy k-means algorithm [33]. During clustering, we treat unknown cells in a slightly different way, since we do not want to get an extra cluster in case the sensor did not cover the whole area completely. In our experiment, we obtained the best behavior using the following distance function for two vectors a and b during clustering

$$d(a,b) = \sum_i \begin{cases} (a_i - b_i) & a_i \neq \xi \wedge b_i \neq \xi \\ 0 & a_i = \xi \wedge b_i = \xi \\ \epsilon & \text{otherwise,} \end{cases} \tag{10.3}$$

where ϵ is a constant close to zero.

When comparing two values representing unknown cells, one in general should use the average distance computed over all known cells to estimate this quantity. Such a value, however, would be significantly larger than zero (except if the whole map is empty space). In our experiments, we experienced that using the average distance between cells leads to additional clusters in case a significant part of a sub-map contains unknown cells even if the known areas of the maps are nearly identical. Therefore, we use the distance function given in (10.3) which sets this distance value to zero.

Unfortunately, the number of different states is not known in advance. Therefore, we iterate over the number of clusters and compute in each iteration a model using the fuzzy k-means algorithm. We create a new cluster initialized using the input vector which has the lowest likelihood under the current model. We evaluate each model θ using the Bayesian information criterion (BIC) [133]:

$$BIC = \log p(d \mid \theta) - \frac{|\theta|}{2} \log n \tag{10.4}$$

10.2.3 Map Clustering

The BIC is a popular technique to score a model during clustering. It trades off the number $|\theta|$ of clusters in the model θ multiplied by the logarithm of the

number of input vectors n and the quality of the model with respect to the given data d. The model with the highest BIC is chosen as the set of possible configurations, in the following also called patches, for that sub-map. This process is repeated for all sub-maps.

The following example is designed to illustrate the map clustering process. The input to the clustering was a set of 17 local grid maps. The fuzzy k-means clustering algorithm started with a single cluster, which is given by the mean computed over all 17 maps. The result is depicted in the first row of Figure 10.2. The algorithm then increased the number of clusters and recomputed the means in each step. In the fifth iteration the newly created cluster is more or less equal to cluster 3. Therefore, the BIC decreased and the clustering algorithm terminated with the model depicted in the forth row of Figure 10.2.

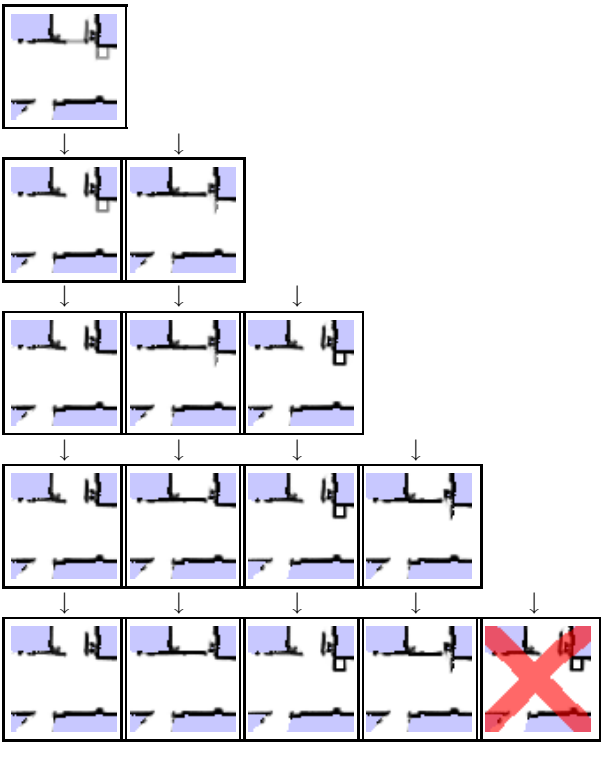


Fig. 10.2. Iterations of the map clustering process. Our approach repeatedly adds new clusters until no improvement is achieved by introducing new clusters (with respect to the BIC). Here, the algorithm ends up with 4 clusters, since cluster 3 and 5 are redundant.

In the introduction of this chapter, we claimed that our approach can also be used in environments which contain highly dynamic aspects like walking people. This is be done by applying the filtering technique introduced by Hähnel *et al.* [62] to the observations sequences $\phi_i, i = 1, \ldots, n$ individually and not to the whole set Φ at once. As a result, objects currently in motion are eliminated by that technique, but objects changing their location while the robot moves through different parts of the environment are correctly integrated into the local maps. The different configurations are then identified by the clustering algorithm.

Note that our approach is an extension of the classical occupancy grid map. It relaxes the assumption that the environment is static. In situations without moving objects, the overall map is equal to a standard occupancy grid map.

The complexity of our mapping approach depends linearly on the number T of observations multiplied by the number L of sub-maps. Furthermore, the region growing applied to build up local maps introduces in the worst case a complexity of $P^2 \log P$, where P is the number of grid cells considered dynamic. This leads to an overall complexity of $O(TL + P^2 \log P)$. Using a standard PC, our implementation requires around 10%-20% of the time needed to record the log file with a real robot.

10.3 Monte-Carlo Localization Using Patch-Maps

In this section, we show how our patch-map representation can be used to estimate the pose of a mobile robot moving through its environment. Throughout this chapter, we apply an extension of Monte-Carlo localization (MCL), which has originally been developed for mobile robot localization in static environments [25]. MCL uses a set of weighted particles to represent possible poses (x, y, and θ) of the robot. As explained in Chapter 2, the motion model is typically used to draw the next generation of samples. The sensor readings are used to compute the weight of each particle by estimating the likelihood of the observation given the pose of the particle and the map.

Besides the pose of the robot, we want to estimate the configuration of the environment in our approach. We do not assume a static map in like standard MCL and therefore need to estimate the map m_t as well as the pose x_t of the robot at time t

$$\begin{aligned} p(x_t, m_t \mid z_{1:t}, u_{1:t-1}) &\overset{\text{Bayes' rule}}{=} \\ \eta p(z_t \mid x_t, m_t, z_{1:t-1}, u_{1:t-1}) p(x_t, m_t \mid z_{1:t-1}, u_{1:t-1}). \end{aligned} \tag{10.5}$$

Here η is a normalization constant and u_{t-1} refers to the motion command which guides the robot from x_{t-1} to x_t. The main difference to approaches on simultaneous localization and mapping (see Chapter 6) is that we do not reason about all possible map configurations like SLAM approaches do. Our

patch-map restricts the possible states according to the clustering of patches and therefore only a small number of configurations are possible.

Under the Markov assumption, the second line of (10.5) can be transformed to

$$
\begin{aligned}
& p(x_t, m_t \mid z_{1:t-1}, u_{1:t-1}) \\
& \overset{\text{Markov \& total prob.}}{=} \int_{x_{t-1}} \int_{m_{t-1}} p(x_t, m_t \mid x_{t-1}, m_{t-1}, z_{1:t-1}, u_{t-1}) \\
& \qquad p(x_{t-1}, m_{t-1} \mid z_{1:t-1}, u_{1:t-2})\, dx_{t-1}\, dm_{t-1} \qquad (10.6) \\
& \overset{\text{product rule}}{=} \int_{x_{t-1}} \int_{m_{t-1}} p(x_t \mid x_{t-1}, m_{t-1}, z_{1:t-1}, u_{t-1}) \\
& \qquad p(m_t \mid x_t, x_{t-1}, m_{t-1}, z_{1:t-1}, u_{t-1}) \\
& \qquad p(x_{t-1}, m_{t-1} \mid z_{1:t-1}, u_{1:t-2})\, dx_{t-1}\, dm_{t-1} \qquad (10.7) \\
& = \int_{x_{t-1}} \int_{m_{t-1}} p(x_t \mid x_{t-1}, u_{t-1}) p(m_t \mid x_t, m_{t-1}) \\
& \qquad p(x_{t-1}, m_{t-1} \mid z_{1:t-1}, u_{1:t-2})\, dx_{t-1}\, dm_{t-1}. \qquad (10.8)
\end{aligned}
$$

Equation (10.6) is obtained by using the law of total probability and the Markov assumption. Furthermore, $u_{1:t-2}$ is assumed to have no influence on the estimate of x_t and m_t given x_{t-1} is known. In the recursive term of (10.6), u_{t-1} is assumed to have no influence on x_{t-1}, since u_{t-1} describes the odometry information between x_{t-1} and x_t.

Equation (10.8) is obtained from (10.7) by assuming that m_t is independent from $x_{t-1}, z_{1:t-1}, u_{t-1}$ given we know x_t and m_{t-1} as well as by assuming that x_t is independent from $m_{t-1}, z_{1:t-1}$ given we know x_{t-1} and u_{t-1}. Combining (10.5) and (10.8) leads to

$$
\begin{aligned}
& p(x_t, m_t \mid z_{1:t}, u_{1:t-1}) \\
& = \eta p(z_t \mid x_t, m_t, z_{1:t-1}, u_{1:t-1}) \\
& \quad \int_{x_{t-1}} \int_{m_{t-1}} p(x_t \mid x_{t-1}, u_{t-1}) p(m_t \mid x_t, m_{t-1}) \\
& \quad p(x_{t-1}, m_{t-1} \mid z_{1:t-1}, u_{1:t-2})\, dx_{t-1}\, dm_{t-1}. \qquad (10.9)
\end{aligned}
$$

Equation (10.9) describes how to extend the standard MCL approach so that it can deal with different spacial configurations. Besides the motion model $p(x_t \mid x_{t-1}, u_{t-1})$ of the robot, we need to specify a map transition model $p(m_t \mid x_t, m_{t-1})$, which describes the change in the environment over time.

In our current implementation, we do not reason about the state of the whole map, since each sub-map would introduce a new dimension in the state vector of each particle, which leads to a state estimation problem, that is exponential in the number of local sub-maps. Furthermore, the observations obtained with a mobile robot provide information only about the local environment of the robot. Therefore, we only estimate the state of the current

patch each particle is currently in. This leads to one additional dimension in the state vector of the particles compared to standard MCL.

In principle, the map transition model $p(m_t \mid x_t, m_{t-1})$ can be learned while the robot moves through the environment. In our current system, we use a fixed density for all patches. We assume, that with probability α the current state of the environment does not change between time $t-1$ and t. Accordingly, the state changes to another configuration with probability $1-\alpha$. Whenever a particle stays in the same sub-map between $t-1$ and t, we draw a new local map configuration for that sample with probability $1-\alpha$. If a particle moves to a new sub-map, we draw the new map state from a uniform distribution over the possible patches in that sub-map. Note that this is a valid procedure, since one can draw the next generation of samples from an arbitrary distribution according to the importance sampling principle (see Chapter 2). To improve the map transition model during localization, one in principle can update the values for α for each patch according to the observations of the robot. Adapting these densities can also be problematic in case of a diverged filter or a multi-modal distribution about the pose of the robot. Therefore, we currently do not adapt the values of α while the robot acts in the environment.

Note that our representation bears resemblance with approaches using Rao-Blackwellized particle filters to solve the simultaneous localization and mapping problem, as it separates the estimate of the pose of the robot from the estimate of the map (compare Chapter 6). Our approach samples the state of the (local) map and then computes the localization of the vehicle based on that knowledge. The main difference compared to Rao-Blackwellized SLAM is that we aim to estimate the current state of the sub-map based on the possible configurations represented in our enhanced map model.

10.4 Experimental Results

To evaluate our approach, we implemented and thoroughly tested it on an ActivMedia Pioneer 2 robot equipped with a SICK laser range finder. The experiments are designed to show the effectiveness of our method to identify possible configurations of the environment and to utilize this knowledge to more robustly localize a mobile vehicle.

10.4.1 Application in an Office Environment

The first experiment has been carried out in a typical office environment. The data was recorded by steering the robot through the environment while the states of the doors changed. To obtain a more accurate pose estimate than the raw odometry information, we apply a standard scan-matching technique. Figure 10.3 depicts the resulting patch-map. For the three sub-maps that contain the doors whose states were changed during the experiment our

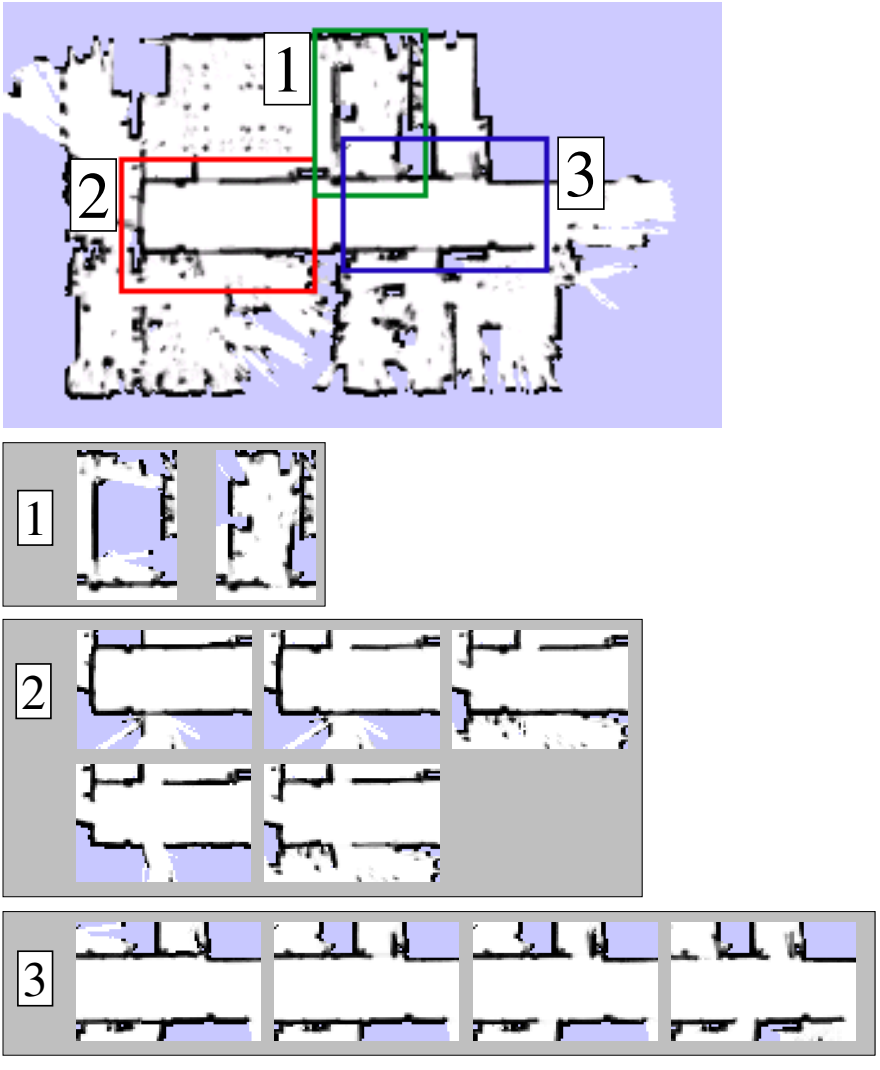


Fig. 10.3. The patch-map represents the different configurations learned for the individual sub-maps in a typical office environment.

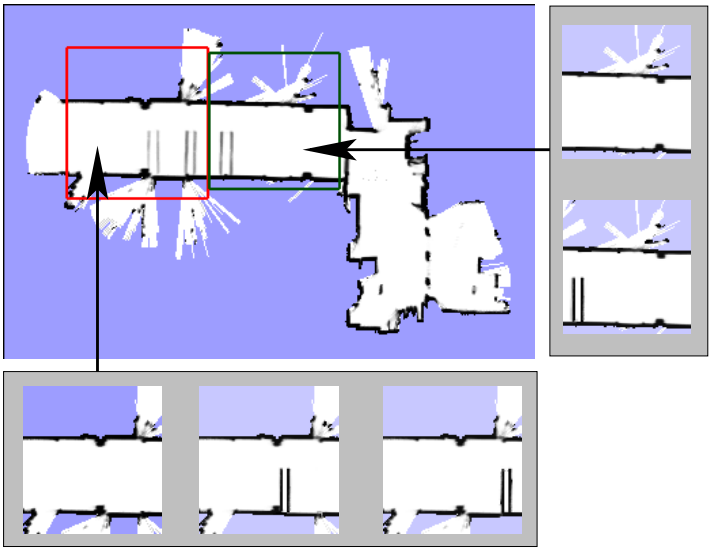


Fig. 10.4. The patch-map with the different configurations for the individual patches used in the localization experiment in Figure 10.5.

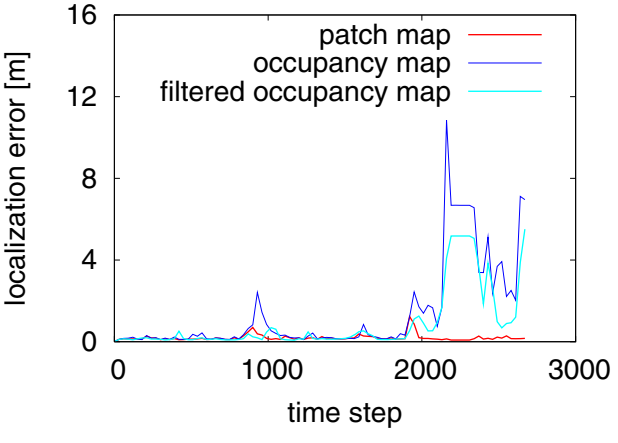


Fig. 10.5. The error in the pose estimate over time. As can be seen, using our approach the quality of the localization is higher compared to approaches using occupancy grid maps.

algorithm was able to learn all configurations that occurred. The sub-maps and their corresponding patches are shown in the same figure.

10.4.2 Localizing the Robot and Estimating the State of the Environment

The second experiment is designed to illustrate the advantages of our map representation for mobile robot localization in non-static environments compared to standard MCL. The data used for this experiment was obtained in the same office environment as above. We placed an obstacle at three different locations in the corridor. The resulting map including all patches obtained via clustering is depicted in Figure 10.4. Note that the tiles in the global map illustrate the average over the individual patches. To evaluate the localization accuracy obtained with our map representation, we compare the pose estimates to that of a standard MCL using a classical grid map as well as using a grid map obtained by filtering out dynamic objects according to [62].

Figure 10.5 plots the localization error over time for the three different representations. The error was determined as the weighted average distance from the poses of the particles to the ground truth. In the beginning of this experiment, the robot traveled through static areas so that all localization methods performed equally well. Close to the end, the robot traveled through the dynamic areas, which results in high pose errors for both alternative approaches. In contrast to that, our technique constantly yields a high localization accuracy and correctly tracks the robot.

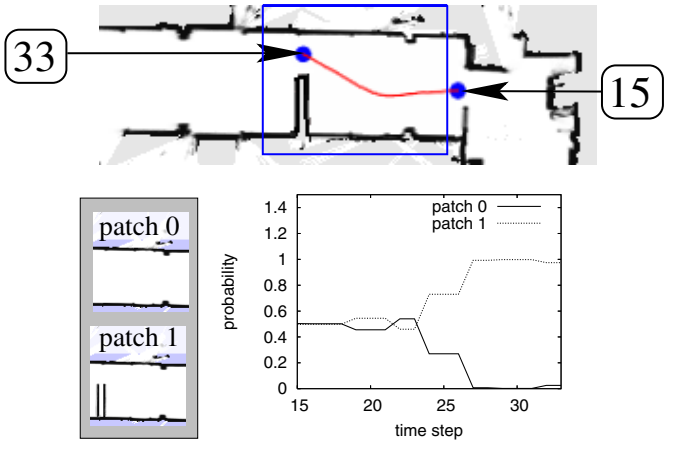


Fig. 10.6. The image in the first row illustrates the traveled path with time labels. The left images in the second row depict the two patches and the graph plots the probability of both patches according to the sample set. As can be seen, the robot identified that patch 1 correctly models the configuration of the environment.

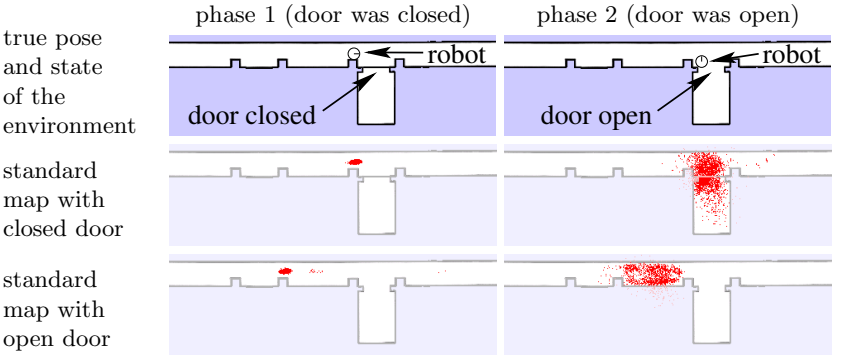


Fig. 10.7. This figure shows a global localization experiment using standard grid maps. The first row depicts the true pose of the robot and the true state of the door. The second row shows the same situation during a localization experiment using a map in which the door is modeled as closed. In the experiment depicted in the third row the used map was contained no doors at all. In the beginning of this experiment the door was closed (left column) but was later on opened (right column). As can be seen, both systems were unable to accurately localize the vehicle.

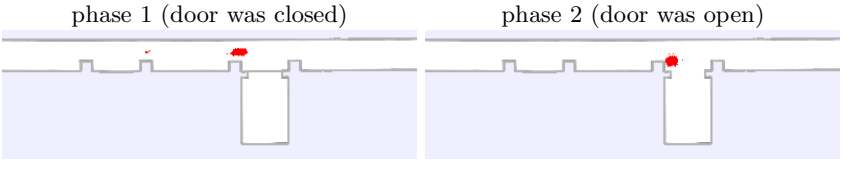


Fig. 10.8. Particle clouds obtained with our algorithm for the same situations as depicted in Figure 10.7.

To further illustrate how our extended MCL algorithm is able to estimate the current state of the environment, Figure 10.6 plots the posterior probabilities for two different patches belonging to one sub-map. At time step 17, the robot entered the corresponding sub-map. After a few time steps, the robot correctly identified, that the particles, which localize the robot in patch 1, performed better than the samples using patch 0. Due to the resamplings in MCL, particles with a low importance weight are more likely to be replaced by particles with a high importance weight. Over a sequence of integrated measurements and resamplings, this led to an probability close to 1 that the environment looked like the map represented by patch 1 (which corresponded to the ground truth in that situation).

10.4.3 Global Localization

Additionally, we carried out three global localization experiments in a simulated environment. First, we used a standard grid map which contains a closed door. In the second run, we used a map which did not contain a door at all and finally we used our patch-map representation using two patches to represent the state of the door. The experiments with standard MCL are depicted in Figure 10.7, the corresponding one using patch-maps is shown in Figure 10.8. During localization, the robot moved most of the time in front of the door, which was closed in the beginning and opened in the second phase of the experiment.

As can be seen in the left column of Figure 10.7 and 10.8, the MCL approach which uses the occupancy grid that models the closed door as well as our approach lead to a correct pose estimate. In contrast to that, the occupancy grid which models the open door causes the filter to diverge. In the second phase of the experiment, the door was opened and the robot again moved some meters in front of the door (see right column of the same figure). At this point in time, the MCL technique using the occupancy grid, which models the closed door cannot track the correct pose anymore, whereas our approach is able to correctly estimate the pose of the robot. This simulated experiment again illustrates that the knowledge about possible configurations of the environment is important for mobile robot localization. Without this knowledge, the robot is not able to correctly estimate its pose in non-static environments.

10.5 Related Work

In the past, several authors have studied the problem of learning maps in dynamic environments. A popular technique is to track dynamic objects and filter out the measurements reflected by those objects [61, 158]. Enhanced sensor models combined with the Expectation Maximization (EM) algorithm have been successfully applied to filter out arbitrary dynamic objects by Hähnel *et al.* [62]. The authors report that filtering out dynamic objects can improve the scan registration and lead to more accurate maps.

Anguelov *et al.* [4] present an approach which aims to learn models of non-stationary objects from proximity data. The object shapes are estimated by applying a hierarchical EM algorithm based on occupancy grids recorded at different points in time. The main difference to our approach is that we estimate typical configurations of the environment and do not focus on learning geometric models for different types of non-stationary obstacles. They furthermore presented a work in which they estimate the state of doors in an environment [5]. They apply the EM algorithm to distinguish door objects from wall objects as well as different properties like color, shape, and motion.

The problem of dealing with walking people has also been investigated in the context of mobile robot localization. For example, Fox *et al.* [44] use a probabilistic technique to identify range measurements which do not correspond to a given model. In contrast to our work, they use a fixed, static map model and do not reason about configurations the environment can be in. In a different project, a team of tour-guide robots has been reported to successfully act in highly populated environments [134]. Their system uses line features for localization resting on the assumption that such features more likely correspond to walls than to moving people. Montemerlo *et al.* [102] use a method to track walking people while localizing the robot to increase the robustness of the pose estimate.

Romero *et al.* [124] describe an approach to global localization that clusters extracted features based on similarity. In this way, the robot is able to reduce the number of possible pose hypotheses and can speed up a Markov localization process. The authors also perform a clustering of sub-maps, but compared to our work, they do not consider changes in the environment.

In contrast to most of the approaches discussed so far, we do not address the problem of filtering out or tracking dynamic objects. Our technique is complementary to and can be combined with those approaches. In this work, we applied the approach of Hähnel *et al.* [62] to eliminate high dynamic objects in the short observation sequences ϕ_i instead of in the whole dataset. We are interested in possible states of the environment like, for example, open and closed doors or moved tables. In this context, it makes sense to filter out measurements reflected by walking people, but to integrate those which correspond to obstacles like doors or moved furniture. Our approach learns possible states based on a clustering of local maps. The different state hypotheses enable a mobile robot to more reliably localize itself and to also estimate the current configuration of its surroundings.

In a recent work, Biber and Duckett [12] proposed an elegant approach that incorporates changes of the environment into the map representation. Compared to our work, they model temporal changes of local maps whereas we aim to identify the different configurations of the environment. In their work, they also construct local map but do not use grid maps like we do. For each local map they maintain five different map instances over different time scales . This is achieved by accepting changes differently fast. During Monte-Carlo localization, they estimate only the pose of the robot and not state of the environment. To compute the importance weight for a particle, they evaluate the observation likelihood in each map and then choose the mode. This is different to our Rao-Blackwellized approach in which each sample is evaluated based on its individual map estimate.

Van Den Berg *et al.* [154] presented an approach to motion planning in dynamic environments using randomized roadmaps. Their approach is able to deal with multiple configurations of local areas in the environment. This allows a mobile robot to replan its path given a passage is blocked by an

obstacle. Their technique focuses on path planning and leaves open how such dynamic areas can be identified and mapped.

10.6 Conclusion

In this chapter, we presented a technique to model quasi-static environments using a mobile robot. In areas where dynamic aspects are detected, our approach creates local maps and estimates for each sub-map clusters of possible configurations of the corresponding space in the environment. This allows us to model, for example, opened and closed doors or moved furniture.

Furthermore, we described how to extend Monte-Carlo localization to utilize the information about the different possible states of the environment while localizing a vehicle. We use a Rao-Blackwellized particle filter to estimate the current state of the environment as well as the pose of the robot.

Our approach as been implemented and tested on real robots as well as in simulation. The experiments demonstrate that our technique yields a higher localization accuracy compared to Monte-Carlo localization based on standard occupancy grids as well as grid maps obtained after filtering out measurements reflected by dynamic objects. As illustrated in this chapter, approaches which do not consider changes in the map model are unable to localize a vehicle correctly in certain situations. This is especially a problem when performing global localization while the environment is not static.

11

Conclusion

Learning map is one of the key problems in mobile robotics, since many applications require known spacial models. Robots that are able to acquire an accurate map of the environment on their own are regarded as fulfilling a major precondition of truly autonomous mobile vehicles. The autonomous map learning problem has several important aspects that need to be solved simultaneously in order to come up with accurate models. These problems are mapping, localization, and path planning. Additionally, most mapping approaches assume that the environment of the mobile robots is static and does not change over time. This assumption, however, is unrealistic since most places are populated by humans. Taking into account non-static aspects is therefore an desirable feature for mapping systems.

In this book, we focused on the problem of learning accurate maps with single- and multi-robot systems. We presented solutions to a series of open problems in this context. We started with the problem of exploring an environment with a mobile robot equipped with a noisy sensor. We presented a decision-theoretic framework that reasons about potential observations to be obtained at the robot's target locations. In this way, the robot is able to select the action that provides the highest expected uncertainty reduction in its map. This allows the robot to build accurate environment models not exceeding a given level of uncertainty. As the underlying representation, we defined coverage maps which can be seen as an extension of occupancy grid maps that allow us to model partly occupied cells. We then presented in Chapters 4 and 5 a technique to coordinate a team of robots during exploration. The main challenge in this context is to assign appropriate target locations to each robot so that the overall time to complete the exploration task is minimized. This collaboration between the robots is achieved by assigning utilities to all potential target locations. Whenever a target location is assigned to a robot, the utility of all locations that are visible from the assigned one are discounted. This leads to a balanced distribution of robots over the environment and reduces the amount of redundant work as well as the risk of interference. As a result, we obtained a significant reduction of the overall time needed to complete the exploration mission. We described

C. Stachniss: Robotic Mapping and Exploration, STAR 55, pp. 177–180.
springerlink.com

a way of dealing with limited communication in the network link. This was achieved by applying our centralized technique for sub-teams of robots which are currently able to communicate. Furthermore, we learned typical properties of indoor environments using the AdaBoost algorithm in combination with simple, single-valued features. By enabling the robots to add semantic labels to the individual places in the environment, the coordination of large robot teams can be optimized. We focused on the exploration of corridors, which typically have a high number of branchings to adjacent rooms, where large teams of robots can be better distributed over the environment. Using this technique to cover the environment with a team of robots, the task can be carried out in an even shorter period of time.

Whenever robots act in the real world, their actions and observations are affected by noise. Building spacial models under those conditions without consider active control is widely known as the simultaneous localization and mapping (SLAM) problem. It is often called a chicken or egg problem, since a map is needed to localize a vehicle while at the same time an accurate pose estimate is needed to build a map. We presented in Chapter 6 a solution to the SLAM problem which is based on a Rao-Blackwellized particle filter using grid maps. In such a filter, each sample represents a trajectory hypothesis and maintains its own map. Each map is updated based on the trajectory estimate of the corresponding particle. The main challenge in the context of Rao-Blackwellized mapping is to reduce its complexity, typically measured by the number of samples needed to build an accurate map. We presented a highly efficient technique which uses an informed proposal distribution to create the next generation of particles. We consider the most recent sensor observation to obtain an accurate proposal distribution. This allows us to draw samples only in those areas where the robot is likely to be located. We furthermore reduced the number of resampling actions in the particle filter which helps to make particle depletion less likely. As a result, our technique enables us to construct grid maps from large datasets in which the robots traveled for around 2 km in indoor as well as in structured outdoor environments. We are able to obtain maps with outstanding accuracy requiring around one order of magnitude less samples than other state-of-the-art Rao-Blackwellized mapping systems.

After having developed an efficient and accurate tool to deal with the uncertainty in the pose of the vehicle, we considered the problem of how to combine exploration and SLAM systems in Chapter 7. Since exploration strategies typically try to cover unknown terrain as fast as possible, they avoid repeated visits to known areas. This strategy, however, is suboptimal in the context of the SLAM problem because the robot needs to revisit places in order to localize itself. A good pose estimate is necessary to make the correct data association, i.e., to determine if the current measurements fit into the map built so far. In the case in which the robot uses an exploration strategy that avoids multiple visits to the same place, the probability of making the correct associations is reduced. To overcome this limitation, we developed a relocalization technique for exploration based on loop-closing actions. First,

the robot has to detect loops which have not been traversed to far. This is done by maintaining a dual representation of the environment. Beside a grid map, we construct a topological map based on the trajectory of the vehicle. By comparing both models, we are able to reliably detect loops. This information is then used to reenter the known parts of the environment in order to relocalize the vehicle. This often leads to better aligned maps especially at the loop closure point.

The problem of the presented technique lies in its heuristic estimation of when to abort the loop-closing process. If the robot moves for an extremely long period of time through known areas, the so-called particle depletion problem can affect the filter. Particle depletion is the phenomenon that hypotheses which are needed later on, for example to close a second loop, vanish while the robot stays in a first, inner loop. Chapter 8 describes a technique that allows a mobile robot to propagate the particle diversity through a loop after actively closing it. By creating a backup of the filter when entering a loop and recovering the uncertainty when leaving the loop, the robot can stay an arbitrary period of time in a loop without depleting important hypotheses. As shown in our experiments, this approach yields accurate maps while reducing the risk that the filter gets overly confident.

Chapter 9 presented an integrated approach that simultaneously addresses mapping, localization, and path planning. It extends the ideas on decision-theoretic exploration presented in Chapter 3 and allows us to deal with the pose uncertainty of the vehicle. It applies the Rao-Blackwellized particle filter presented in Chapter 6 to model the posterior about the trajectory of the vehicle and the map of the environment. The decision-theoretic action selection technique aims to minimize the uncertainty in joint posterior about the poses of the robot and the map. In this context, we showed that the entropy of a Rao-Blackwellized filter can be separated into two components: The uncertainty of the posterior about the trajectory and the uncertainty in the map multiplied with the likelihood of the corresponding sample.

Whenever our approach evaluates a set of actions, it takes into account sequences of potential observations in order to minimize the uncertainty in the overall posterior. This is achieved by simulating observation sequences based on the posterior about the map. The actions which are taken into account guide the robot in order to explore unknown areas, move it to places which are well-known in order to reduce the pose uncertainty, or actively close loops according to Chapter 7. As a result, we obtain a robust active map learning approach that

- minimizes the uncertainty in the robot's world model,
- considers the cost of carrying out an action,
- reasons about potential observation sequences based on the posterior about the map of the environment,
- considers exploration, place revisiting, and loop-closing actions, and
- is able to deal with uncertainty in the pose of the robot.

Finally, we addressed the problem of mapping and localization in non-static environments. The assumption of a static world is unrealistic since most places in which robots are deployed are populated by humans. In the last three years, different techniques that are able to deal with dynamic aspects during mapping were presented. This was typically achieved by filtering out the measurements which were reflected by dynamic objects. In Chapter 10, we chose a different approach. Instead of removing the non-static aspects from the map model, we presented a technique to map their typical configurations. The idea behind this approach is that several non-static objects occur only in a limited number of states. Doors, for example, are typically either open or closed and a parking space is either free or occupied by a car. Therefore, it makes sense to include their typical configurations into the environment model. By clustering local sub-maps, we are able to come up with a map model that maintains different possible configurations for local areas.

We then extended the standard Monte-Carlo localization approach to enable a mobile robot to localize itself in this kind of map and at the same time estimate the current state of the environment. This allows us to perform the localization task more robustly in case the environment is not static. In practical experiments, we showed that an approach that is not able to model different spacial states failed to localize a robot correctly whereas our approach succeeded.

The techniques described in this book are solutions to various previously unsolved or unaddressed aspects of the map learning problem with mobile robots. In Chapter 9, we developed an active map learning system that integrates most of our techniques described in the preceding chapters of this work. In summary, the presented approaches allow us to answer the following questions:

- How to coordinate a team of mobile robots so that the overall exploration time and the amount of redundant work is minimized?
- How to accurately and efficiently solve the grid-based simultaneous localization and mapping problem for robots equipped with a laser range finder?
- How to adapt an exploration technique to the needs of the underlying SLAM approach?
- How to reduce the risk of particle depletion in the context of active Rao-Blackwellized mapping?
- How to generate actions and reason about potential observation sequences for an exploring mobile robot with the goal to minimize the uncertainty in its world model?
- How to deal with non-static worlds in the context of map learning and localization?

A

A.1 Probability Theory

A.1.1 Product Rule

The following equation is called the product rule

$$p(x, y) = p(x \mid y)p(y) \tag{A.1}$$
$$= p(y \mid x)p(x). \tag{A.2}$$

A.1.2 Independence

If x and y are independent, we have

$$p(x, y) = p(x)p(y). \tag{A.3}$$

A.1.3 Bayes' Rule

The Bayes' rule, which is frequently used in this book, is given by

$$p(x \mid y) = \frac{p(y \mid x)p(x)}{p(y)}. \tag{A.4}$$

The denominator is a normalizing constant that ensures that the posterior of the left hand side adds up to 1 over all possible values. Thus, we often write

$$p(x \mid y) = \eta p(y \mid x)p(x). \tag{A.5}$$

In case the background knowledge e is given, Bayes' rule turns into

$$p(x \mid y, e) = \frac{p(y \mid x, e)p(x \mid e)}{p(y \mid e)}. \tag{A.6}$$

A.1.4 Marginalization

The marginalization rule is the following equation

$$p(x) = \int_y p(x, y)\, dy. \tag{A.7}$$

In the discrete case, the integral turns into a sum

$$p(x) = \sum_y p(x, y). \tag{A.8}$$

A.1.5 Law of Total Probability

The law of total probability is a variant of the marginalization rule, which can be derived using the product rule

$$p(x) = \int_y p(x \mid y)p(y)\, dy, \tag{A.9}$$

and the corresponding sum for the discrete case

$$p(x) = \sum_y p(x \mid y)p(y). \tag{A.10}$$

A.1.6 Markov Assumption

The Markov assumption (also called Markov property) characterizes the fact that a variable x_t depends only on its direct predecessor state x_{t-1} and not on $x_{t'}$ with $t' < t - 1$

$$p(x_t \mid x_{1:t-1}) = p(x_t \mid x_{t-1}). \tag{A.11}$$

References

1. Albers, S., Henzinger, M.R.: Exploring unknown environments. SIAM Journal on Computing 29, 1164–1188 (2000)
2. Albers, S., Kursawe, K., Schuierer, S.: Exloring unknown environments with obstacles. Algotithmica 32, 123–143 (2002)
3. Althaus, P., Christensen, H.I.: Behaviour coordination in structured environments. Advanced Robotics 17(7), 657–674 (2003)
4. Anguelov, D., Biswas, R., Koller, D., Limketkai, B., Sanner, S., Thrun, S.: Learning hierarchical object maps of non-stationary environments with mobile robots. In: Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI), Edmonton, Canada (2002)
5. Anguelov, D., Koller, D., Parker, E., Thrun, S.: Detecting and modeling doors with mobile robots. In: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), New Orleans, LA, USA, pp. 3777–3774 (2004)
6. Apostolopoulos, D., Pedersen, L., Shamah, B., Shillcutt, K., Wagner, M.D., Whittaker, W.R.L.: Robotic antarctic meteorite search: Outcomes. In: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), Seoul, Korea, pp. 4174–4179 (2001)
7. Balch, T., Arkin, R.C.: Communication in reactive multiagent robotic systems. Journal of Autonomous Robots 1(1), 27–52 (1994)
8. Bellman, R.E.: Dynamic Programming. Princeton University Press, Princeton (1957)
9. Bender, M., Slonim, D.: The power of team exploration: two robots can learn unlabeled directed graphs. In: Proc. of the 35th Annual Symposium on Foundations of Computer Science, pp. 75–85 (1994)
10. Bennewitz, M.: Mobile Robot Navigation in Dynamic Environments. PhD thesis, University of Freiburg, Department of Computer Science (2004)
11. Bentley, J.L.: Multidimensional divide and conquer. Communications of the ACM 23(4), 214–229 (1980)
12. Biber, P., Duckett, T.: Dynamic maps for long-term operation of mobile service robots. In: Proc. of Robotics: Science and Systems (RSS), Cambridge, MA, USA, pp. 17–24 (2005)
13. Billard, A., Ijspeert, A.J., Martinoli, A.: A multi-robot system for adaptive exploration of a fast changing environment: probabilistic modelling and experimental study. Connection Science 11(3/4), 357–377 (2000)

14. Bosse, M., Newman, P.M., Leonard, J.J., Teller, S.: An ALTAS framework for scalable mapping. In: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), Taipei, Taiwan, pp. 1899–1906 (2003)
15. Bourgoult, F., Makarenko, A.A., Williams, S.B., Grocholsky, B., Durrant-Whyte, F.: Information based adaptive robotic exploration. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Lausanne, Switzerland (2002)
16. Burgard, W.: Mapping with known poses. Lecture Notes on Introduction to Mobile Robotics. University of Freiburg, Germany (2005), `http://ais.informatik.uni-freiburg.de/lehre/ss05/robotics/slides/10.pdf`
17. Burgard, W., Moors, M., Fox, D., Simmons, R., Thrun, S.: Collaborative multi-robot exploration. In: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), San Francisco, CA, USA, pp. 476–481 (2000)
18. Burgard, W., Moors, M., Schneider, F.: Collaborative exploration of unknown environments with teams of mobile robots. In: Beetz, M., Hertzberg, J., Ghallab, M., Pollack, M.E. (eds.) Dagstuhl Seminar 2001. LNCS, vol. 2466, pp. 52–70. Springer, Heidelberg (2002)
19. Buschka, P., Saffiotti, A.: A virtual sensor for room detection. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Lausanne, Switzerland, pp. 637–642 (2002)
20. Cao, Y.U., Fukunaga, A.S., Khang, A.B.: Cooperative mobile robotics: Antecedents and directions. Journal of Autonomous Robots 4(1), 7–27 (1997)
21. Choset, H.: Coverage for robotics - a survey of recent results. In: Annals of Mathematics and Artifical Intelligence (2001)
22. Choset, H.: Topological simultaneous localization and mapping (SLAM): Toward exact localization without explicit localization. IEEE Transactions on Robotics and Automation 17(2), 125–137 (2001)
23. Cohen, W.: Adaptive mapping and navigation by teams of simple robots. Journal of Robotics & Autonomous Systems 18, 411–434 (1996)
24. Dellaert, F.: Square Root SAM. In: Proc. of Robotics: Science and Systems (RSS), Cambridge, MA, USA, pp. 177–184 (2005)
25. Dellaert, F., Fox, D., Burgard, W., Thrun, S.: Monte carlo localization for mobile robots. In: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), Leuven, Belgium (1998)
26. Deng, X., Kameda, T., Papadimitriou, C.: How to learn in an unknown environment. In: Proc. of the 32nd Symposium on the Foundations of Computational Science, pp. 298–303. IEEE Computer Society Press, Los Alamitos (1991)
27. Deng, X., Papadimitriou, C.: How to learn in an unknown environment: The rectilinear case. Journal of the ACM 45(2), 215–245 (1998)
28. Dissanayake, G., Durrant-Whyte, H., Bailey, T.: A computationally efficient solution to the simultaneous localisation and map building (SLAM) problem. In: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), San Francisco, CA, USA, pp. 1009–1014 (2000)
29. Doucet, A.: On sequential simulation-based methods for bayesian filtering. Technical report, Signal Processing Group, Dept. of Engeneering, University of Cambridge (1998)

30. Doucet, A., de Freitas, J.F.G., Murphy, K., Russel, S.: Rao-Blackwellized particle filtering for dynamic bayesian networks. In: Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI), Stanford, CA, USA, pp. 176–183 (2000)
31. Doucet, A., de Freitas, N., Gordan, N. (eds.): Sequential Monte-Carlo Methods in Practice. Springer, Heidelberg (2001)
32. Duckett, T., Marsland, S., Shapiro, J.: Fast, on-line learning of globally consistent maps. Journal of Autonomous Robots 12(3), 287–300 (2002)
33. Duda, R., Hart, P., Stork, D.: Pattern Classification. Wiley Interscience, Hoboken (2001)
34. Dudek, G., Jenkin, M., Milios, E., Wilkes, D.: Robotic exploration as graph construction. IEEE Transactions on Robotics and Automation 7(6), 859–865 (1991)
35. Dudek, G., Jenkin, M., Milios, E., Wilkes, D.: A taxonomy for multi-agent robotics. Journal of Autonomous Robots 3(4), 375–397 (1996)
36. Edlinger, T., von Puttkamer, E.: Exploration of an indoor-environment by an autonomous mobile robot. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Munich, Germany, pp. 1278–1248 (1994)
37. Eliazar, A., Parr, R.: DP-SLAM: Fast, robust simultainous localization and mapping without predetermined landmarks. In: Proc. of the Int. Conf. on Artificial Intelligence (IJCAI), Acapulco, Mexico, pp. 1135–1142 (2003)
38. Endres, H., Feiten, W., Lawitzky, G.: Field test of a navigation system: Autonomous cleaning in supermarkets. In: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), Leuven, Belgium, pp. 1779–1781 (1998)
39. Eustice, R., Walter, M., Leonard, J.J.: Sparse extended information filters: Insights into sparsification. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Edmonton, Cananda, pp. 641–648 (2005)
40. Feder, H., Leonard, J., Smith, C.: Adaptive mobile robot navigation and mapping. Int. Journal of Robotics Research 18(7) (1999)
41. Folkesson, J., Christensen, H.: Outdoor exploration and slam using a compressed filter. In: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), Taipei, Taiwan (2003)
42. Fox, D., Burgard, W., Kruppa, H., Thrun, S.: Collaborative multi-robot localization. In: Proc. of the 23rd German Conference on Artificial Intelligence, pp. 325–340. Springer, Heidelberg (1999)
43. Fox, D., Burgard, W., Kruppa, H., Thrun, S.: A probabilistic approach to collaborative multi-robot localization. Journal of Autonomous Robots 8(3) (2000)
44. Fox, D., Burgard, W., Thrun, S.: Markov localization for mobile robots in dynamic environments. Journal of Artificial Intelligence Research (JAIR) 11, 391–427 (1999)
45. Fox, D., Ko, J., Konolige, K., Stewart, B.: A hierarchical bayesian approach to the revisiting problem in mobile robot map building. In: Proc. of the Int. Symposium of Robotics Research (ISRR), Siena, Italy (2003)
46. Frese, U., Hirzinger, G.: Simultaneous localization and mapping - a discussion. In: Proc. of the IJCAI Workshop on Reasoning with Uncertainty in Robotics, Seattle, WA, USA, pp. 17–26 (2001)
47. Frese, U., Larsson, P., Duckett, T.: A multilevel relaxation algorithm for simultaneous localisation and mapping. IEEE Transactions on Robotics 21(2), 1–12 (2005)

48. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences 55(1), 119–139 (1997)
49. Friedman, J.H., Bentley, J.L., Finkel, R.A.: An algorithm for finding best matches in logarithmic expected time. ACM Transactions on Mathematical Software 3(3), 209–226 (1977)
50. Goldberg, D., Matarić, M.J.: Interference as a tool for designing and evaluating multi-robot controllers. Journal of Robotics & Autonomous Systems 8, 637–642 (1997)
51. Gonzalez, R.C., Wintz, P.: Digital Image Processing. Addison-Wesley Publishing Inc., Reading (1987)
52. González-Baños, H.H., Latombe, J.C.: Navigation strategies for exploring indoor environments. Int. Journal of Robotics Research (2001)
53. González-Baños, H.H., Mao, E., Latombe, J.C., Murali, T.M., Efrat, A.: Planning robot motion strategies for efficient model construction. In: Proc. Int. Symp. on Robotics Research (ISRR), Snowbird, UT, USA, pp. 345–352 (2000)
54. Grabowski, R., Khosla, P., Choset, H.: Autonomous exploration via regions of interest. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), pp. 1691–1696. Las Vegas, NV (2003)
55. Grabowski, R., Navarro-Serment, L.E., Paredis, C.J.J., Khosla, P.K.: Heterogeneous teams of modular robots for mapping and exploration. Journal of Autonomous Robots 8(3), 293–308 (2000)
56. Grisetti, G., Stachniss, C., Burgard, W.: Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), Barcelona, Spain, pp. 2443–2448 (2005)
57. Gutmann, J.-S., Konolige, K.: Incremental mapping of large cyclic environments. In: Proc. of the IEEE Int. Symposium on Computational Intelligence in Robotics and Automation (CIRA), Monterey, CA, USA, pp. 318–325 (1999)
58. Guzzoni, D., Cheyer, A., Julia, L., Konolige, K.: Many robots make short work. AI Magazine 18(1), 55–64 (1997)
59. Hähnel, D., Burgard, W., Fox, D., Thrun, S.: An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, pp. 206–211 (2003)
60. Hähnel, D., Burgard, W., Wegbreit, B., Thrun, S.: Towards lazy data association in slam. In: Proc. of the Int. Symposium of Robotics Research (ISRR), Siena, Italy, pp. 421–431 (2003)
61. Hähnel, D., Schulz, D., Burgard, W.: Map building with mobile robots in populated environments. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Lausanne, Switzerland (2002)
62. Hähnel, D., Triebel, R., Burgard, W., Thrun, S.: Map building with mobile robots in dynamic environments. In: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), Taipei, Taiwan (2003)
63. Hougen, D.F., Benjaafar, S., Bonney, J.C., Budenske, J.R., Dvorak, M., Gini, M., French, H., Krantz, D.G., Li, P.Y., Malver, F., Nelson, B., Papanikolopoulos, N., Rybski, P.E., Stoeter, S.A., Voyles, R., Yesin, K.B.: A miniature robotic system for reconnaissance and surveillance. In: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), San Francisco, CA, USA, pp. 501–507 (2000)

64. Howard, A., Matarić, M.J., Sukhatme, S.: An incremental deployment algorithm for mobile robot teams. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Lausanne, Switzerland, pp. 2849–2854 (2002)
65. Howard, A., Roy, N.: The robotics data set repository, Radish (2003), `http://radish.sourceforge.net/`
66. Howard, R.A.: Dynamic Programming and Markov Processes. MIT Press/ Wiley (1960)
67. Huang, Y., Cao, Z., Oh, S., Kattan, E., Hall, E.: Automatic operation for a robot lawn mower. In: SPIE Conference on Mobile Robots, Cambridge, MA, USA, vol. 727, pp. 344–354 (1986)
68. Jäger, M., Nebel, B.: Dynamic decentralized area partitioning for cooperating cleaning robots. In: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), Washington, DC, USA, pp. 3577–3582 (2002)
69. Julier, S., Uhlmann, J., Durrant-Whyte, H.: A new approach for filtering nonlinear systems. In: Proc. of the American Control Conference, Seattle, WA, USA, pp. 1628–1632 (1995)
70. Jung, D., Zelinksy, A.: An architecture for distributed cooperative planning in a behaviour-based multi-robot system. Journal of Robotics & Autonomous Systems 26(2-3), 149–174 (1999)
71. Kaelbling, L.P., Cassandra, A.R., Kurien, J.A.: Acting under uncertainty: Discrete Bayesian models for mobile-robot navigation. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Osaka, Japan (1996)
72. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Technical report, Brown University (1995)
73. Kalman, R.E.: A new approach to linear filtering and prediction problems. ASME-Journal of Basic Engineering, 35–45 (March 1960)
74. Ko, J., Stewart, B., Fox, D., Konolige, K., Limketkai, B.: A practical, decision-theoretic approach to multi-robot mapping and exploration. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, pp. 3232–3238 (2003)
75. Koenig, S., Simmons, R.: Xavier: A robot navigation architecture based on partially observable markov decision process models. In: Kortenkamp, D., Bonasso, R., Murphy, R. (eds.) Artificial Intelligence Based Mobile Robotics: Case Studies of Successful Robot Systems, pp. 91–122. MIT Press, Cambridge (1998)
76. Koenig, S., Szymanski, B., Liu, Y.: Efficient and inefficient ant coverage methods. Annals of Mathematics and Artificial Intelligence 31, 41–76 (2001)
77. Koenig, S., Tovey, C.: Improved analysis of greedy mapping. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA (2003)
78. Koenig, S., Tovey, C., Halliburton, W.: Greedy mapping of terrain. In: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), Seoul, Korea (2001)
79. Kuhn, H.W.: The hungarian method for the assignment problem. Naval Research Logistics Quarterly 2(1), 83–97 (1955)
80. Kuhn, H.W.: Variants of the hungarian method for assignment problems. Naval Research Logistics 3, 253–258 (1956)
81. Kuipers, B., Beeson, P.: Bootstrap learning for place recognition. In: Proc. of the National Conference on Artificial Intelligence (AAAI), Edmonton, Canada (2002)

82. Kuipers, B., Byun, Y.-T.: A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. Journal of Robotics & Autonomous Systems 8, 47–63 (1991)
83. Kurabayashi, D., Ota, J., Arai, T., Yoshida, E.: Cooperative sweeping by multiple mobile robots. In: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), Minneapolis, MN, USA, pp. 1744–1749 (1996)
84. Kurazume, R., Shigemi, N.: Cooperative positioning with multiple robots. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Munich, Germany, pp. 1250–1257 (1994)
85. Lagoudakis, M.G., Markakis, E., Kempe, D., Keskinocak, P., Kleywegt, A., Koenig, S., Tovey, C., Meyerson, A., Jain, S.: Auction-based multi-robot routing. In: Proc. of Robotics: Science and Systems (RSS), Cambridge, USA, pp. 343–350 (2005)
86. Lee, D., Recce, M.: Quantitative evaluation of the exploration strategies of a mobile robot. International Journal of Robotics Research 16(4), 413–447 (1997)
87. Leonard, J.J., Durrant-Whyte, H.F.: Mobile robot localization by tracking geometric beacons. IEEE Transactions on Robotics and Automation 7(4), 376–382 (1991)
88. Leonard, J.J., Feder, H.J.S.: A computationally efficient method for large-scale concurrent mapping and localization. In: Proc. of the Conf. on Neural Information Processing Systems (NIPS), Breckenridge, CO, USA, pp. 169–179 (2000)
89. Lisien, B., Silver, D., Morales, D., Kantor, G., Rekleitis, I.M., Choset, H.: Hierarchical simultaneous localization and mapping. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, pp. 448–453 (2003)
90. Liu, J.S.: Metropolized independent sampling with comparisons to rejection sampling and importance sampling. Statist. Comput. 6, 113–119 (1996)
91. Lowe, D.G.: Object recognition from local scale-invariant features. In: Proc. of the Int. Conf. on Computer Vision (ICCV), Kerkyra, Greece (1999)
92. Lu, F., Milios, E.: Globally consistent range scan alignment for environment mapping. Journal of Autonomous Robots 4, 333–349 (1997)
93. Lumelsky, S., Mukhopadhyay, S., Sun, K.: Dynamic path planning in sensor-based terrain acquisition. IEEE Transactions on Robotics and Automation 6(4), 462–472 (1990)
94. Makarenko, A.A., Williams, S.B., Bourgoult, F., Durrant-Whyte, F.: An experiment in integrated exploration. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Lausanne, Switzerland (2002)
95. Martínez-Mozos, O., Stachniss, C., Burgard, W.: Supervised learning of places from range data using adaboost. In: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), Barcelona, Spain, pp. 1742–1747 (2005)
96. Massios, N., Dorst, L., Voorbraak, F.: A strategy for robot surveillancs using the geometrical structure of the environment. In: IJCAI Workshop, Reasoning with Uncertainty in Robotics, Seattle, WA, USA (2001)
97. Matarić, M.J., Sukhatme, G.: Task-allocation and coordination of multiple robots for planetary exploration. In: Proc. of the Int. Conf. on Advanced Robotics (ICAR), Budapest, Hungary, pp. 61–70 (2001)

98. Meier, D., Stachniss, C., Burgard, W.: Coordinating multiple robots during exploration under communication with limited bandwidth. In: Proc. of the European Conference on Mobile Robots (ECMR), Ancona, Italy, pp. 26–31 (2005)
99. Meijster, A., Roerdink, J.B.T.M., Hesselink, W.H.: A General Algorithm for Computing Distance Transforms in Linear Time. In: Mathematical Morphology and its Applications to Image and Signal Processing, pp. 331–340. Kluwer Academic Publishers, Dordrecht (2000)
100. Modayil, J., Beeson, P., Kuipers, B.: Using the topological skeleton for scalable global metrical map-building. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Sendai, Japan, pp. 1530–1536 (2004)
101. Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B.: FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In: Proc. of the Int. Conf. on Artificial Intelligence (IJCAI), Acapulco, Mexico, pp. 1151–1156 (2003)
102. Montemerlo, M., Thrun, S.: Conditional particle filters for simultaneous mobile robot localization and people tracking. In: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), Washington, DC, USA (2002)
103. Montemerlo, M., Thrun, S.: Simultaneous localization and mapping with unknown data association using FastSLAM. In: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), Taipei, Taiwan, pp. 1985–1991 (2003)
104. Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B.: FastSLAM: A factored solution to simultaneous localization and mapping. In: Proc. of the National Conference on Artificial Intelligence (AAAI), Edmonton, Canada, pp. 593–598 (2002)
105. Moorehead, S.J., Simmons, R., Whittaker, W.L.: Autonomous exploration using multiple sources of information. In: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), Seoul, Korea (2001)
106. Moors, M.: Koordinierte Multi-Robot Exploration. Master's thesis, Department of Computer Science, University of Bonn (2000) (in German)
107. Morales-Menéndez, R., de Freitas, N., Poole, D.: Real-time monitoring of complex industrial processes with particle filters. In: Proc. of the Conf. on Neural Information Processing Systems (NIPS), Vancover, Canada, pp. 1433–1440 (2002)
108. Moravec, H.P., Elfes, A.E.: High resolution maps from wide angle sonar. In: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), St. Louis, MO, USA, pp. 116–121 (1985)
109. Murphy, K.: Bayesian map learning in dynamic environments. In: Proc. of the Conf. on Neural Information Processing Systems (NIPS), Denver, CO, USA, pp. 1015–1021 (1999)
110. Murphy, R.: Human-robot interaction in rescue robotics. IEEE Systems, Man and Cybernetics Part C: Applications and Reviews 34(2), 138–153 (2004)
111. Neira, J., Tardós, J.D.: Data association in stochastic mapping using the joint compatibility test. IEEE Transactions on Robotics and Automation 17(6), 890–897 (2001)
112. Nilsson, N.J.: A mobile automation: an application of artificial intelligence techniques. In: Proc. of the Int. Conf. on Artificial Intelligence (IJCAI), Washington, DC, USA, pp. 509–520 (1969)
113. Oore, S., Hinton, G.E., Dudek, G.: A mobile robot that learns its place. Neural Computation 9(3), 683–699 (1997)

114. Ottmann, T., Widmayer, P.: Algorithmen und Datenstrukturen. Spektrum Verlag, Heidelberg (1996)
115. Parker, L.E.: The effect of heterogeneity in teams of 100+ mobile robots. In: Multi-Robot Systems Volume II: From Swarms to Intelligent Automata. Kluwer Academic Publishers, Boston (2003)
116. Paskin, M.A.: Thin junction tree filters for simultaneous localization and mapping. In: Proc. of the Int. Conf. on Artificial Intelligence (IJCAI), Acapulco, Mexico, pp. 1157–1164 (2003)
117. Pineau, J., Gordon, G., Thrun, S.: Point-based value iteration: An anytime algorithm for pomdps. In: Proc. of the Int. Conf. on Artificial Intelligence (IJCAI), Acapulco, Mexico (2003)
118. Pitt, M.K., Shephard, N.: Filtering via simulation: auxilary particle filters. Technical report, Department of Mathematics, Imperial College, London (1997)
119. Rao, N., Hareti, S., Shi, W., Iyengar, S.: Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms. Technical Report ORNL/TM-12410, Oak Ridge National Laboratory (1993)
120. Rekleitis, I., Dudek, G., Milios, E.: Multi-robot exploration of an unknown environment, efficiently reducing the odometry error. In: Proc. of International Joint Conference in Artificial Intelligence (IJCAI), vol. 2, pp. 1340–1345 (1997)
121. Rekleitis, I., Dudek, G., Milios, E.: Accurate mapping of an unknown world and online landmark positioning. In: Proc. of Vision Interface (VI), Vancouver, Canada, pp. 455–461 (1998)
122. Rekleitis, I., Sim, R., Dudek, G., Milios, E.: Collaborative exploration for the construction of visual maps. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Maui, HI, USA (2001)
123. Rocha, R., Dias, J., Carvalho, A.: Exploring information theory for vision-based volumetric mapping. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Edmonton, Canada, pp. 2409–2414 (2005)
124. Romero, L., Morales, E., Sucar, E.: A hybrid approach to solve the global localozation problem for indoor mobile robots considering sensor's perceptual limitations. In: Proc. of the Int. Conf. on Artificial Intelligence (IJCAI), Seattle, WA, USA (2001)
125. Rottmann, A., Martínez-Mozos, O., Stachniss, C., Burgard, W.: Place classification of indoor environments with mobile robots using boosting. In: Proc. of the National Conference on Artificial Intelligence (AAAI), Pittsburgh, PA, USA, pp. 1306–1311 (2005)
126. Roy, N., Burgard, W., Fox, D., Thrun, S.: Coastal navigation – robot motion with uncertainty. In: Proceedings of the AAAI Fall Symposium: Planning with POMDPs, Stanford, CA, USA (1998)
127. Roy, N., Dudek, G.: Collaborative robot exploration and rendezvous: Algorithms, performance bounds and observations. Journal of Autonomous Robots 11(2), 117–136 (2001)
128. Roy, N., Montemerlo, M., Thrun, S.: Perspectives on standardization in mobile robot programming. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, pp. 2436–2441 (2003)
129. Russ, J.C.: The Image Processing Handbook. CRC Press, Boca Raton (1992)
130. Russel, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice-Hall, Upper Saddle River (1994)

131. Sack, D., Burgard, W.: A comparison of methods for line extraction from range data. In: Proc. of the IFAC Symposium on Intelligent Autonomous Vehicles (IAV), Lisbon, Portugal (2004)
132. Schneider-Fontan, M., Matarić, M.J.: Territorial multi-robot task division. IEEE Transactions on Robotics and Automation 14(5), 815–822 (1998)
133. Schwarz, G.: Estimating the dimension of a model. The Annals of Statistics 6(2) (1978)
134. Siegwart, R., Arras, K.O., Bouabdallah, S., Burnier, D., Froidevaux, G., Greppin, X., Jensen, B., Lorotte, A., Mayor, L., Meisser, M., Philippsen, R., Piguet, R., Ramel, G., Terrien, G., Tomatis, N.: Robox at Expo. 2002: A large-scale installation of personal robots. Journal of Robotics & Autonomous Systems 42(3-4) (2003)
135. Sim, R., Dudek, G., Roy, N.: Online control policy optimization for minimizing map uncertainty during exploration. In: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), New Orleans, LA, USA (2004)
136. Simmons, R., Apfelbaum, D., Burgard, W., Fox, D., Moors, M., Thrun, S., Younes, H.: Coordination for multi-robot exploration and mapping. In: Proc. of the National Conference on Artificial Intelligence (AAAI), Austin, TX, USA, pp. 851–858 (2000)
137. Simoncelli, M., Zunino, G., Christensen, H.I., Lange, K.: Autonomous pool cleaning: Self localization and autonomous navigation for cleaning. Journal of Autonomous Robots 9(3), 261–270 (2000)
138. Singh, K., Fujimura, K.: Map making by cooperating mobile robots. In: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), Atlanta, GA, USA, pp. 254–259 (1993)
139. Smith, R., Self, M., Cheeseman, P.: Estimating uncertain spatial realtionships in robotics. In: Cox, I., Wilfong, G. (eds.) Autonomous Robot Vehicles, pp. 167–193. Springer, Heidelberg (1990)
140. Stachniss, C.: Robotic datasets (2004), `http://www.informatik.uni-freiburg.de/~stachnis/datasets/`
141. Stachniss, C., Grisetti, G.: GMapping project at OpenSLAM.org (2007), `http://openslam.org`
142. Stentz, A.: Optimal and efficient path planning for partially-known environments. In: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), San Diego, CA, USA, pp. 3310–3317 (1994)
143. Stroupe, A.W., Ravichandran, R., Balch, T.: Value-based action selection for exploration and mapping with robot teams. In: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), New Orleans, LA, USA, pp. 4090–4197 (2004)
144. Tailor, C.J., Kriegman, D.J.: Exloration strategies for mobile robots. In: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), Atlanta, GA, USA, pp. 248–253 (1993)
145. Thrun, S.: An online mapping algorithm for teams of mobile robots. Int. Journal of Robotics Research 20(5), 335–363 (2001)
146. Thrun, S.: A probabilistic online mapping algorithm for teams of mobile robots. Int. Journal of Robotics Research 20(5), 335–363 (2001)
147. Thrun, S.: Learning occupancy grids with forward sensor models. Journal of Autonomous Robots 15, 111–127 (2003)
148. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press, Cambridge (2005)

149. Thrun, S., Hähnel, D., Ferguson, D., Montemerlo, M., Triebel, R., Burgard, W., Baker, C., Omohundro, Z., Thayer, S., Whittaker, W.: A system for volumetric robotic mapping of abandoned mines. In: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), Taipei, Taiwan (2003)
150. Thrun, S., Liu, Y., Koller, D., Ng, A.Y., Ghahramani, Z., Durrant-Whyte, H.: Simultaneous localization and mapping with sparse extended information filters. Int. Journal of Robotics Research 23(7/8), 693–716 (2004)
151. Torralba, A., Murphy, K., Freeman, W., Rubin, M.: Context-based vision system for place and object recognition. In: Proc. of the Int. Conf. on Computer Vision (ICCV), Nice, France (2003)
152. Treptow, A., Masselli, A., Zell, A.: Real-time object tracking for soccer-robots without color information. In: Proc. of the European Conference on Mobile Robots (ECMR), Radziejowice, Poland (2003)
153. Uhlmann, J.: Dynamic Map Building and Localization: New Theoretical Foundations. PhD thesis, University of Oxford (1995)
154. van den Berg, J.P., Nieuwenhuisen, D., Jaillet, L., Overmars, M.H.: Creating robust roadmaps for motion planning in changing environments. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Edmonton, Cananda, pp. 2415–2420 (2005)
155. van der Merwe, R., de Freitas, N., Doucet, A., Wan, E.: The unscented particle filter. Technical Report CUED/F-INFENG/TR380, Cambridge University Engineering Department (August 2000)
156. Viola, P., Jones, M.J.: Robust real-time object detection. In: Proc. of IEEE Workshop on Statistical and Theories of Computer Vision, Vancouver, Canada (2001)
157. Walter, M., Eustice, R., Leonard, J.J.: A provably consistent method for imposing sparsity in feature-based slam information filters. In: Proc. of the Int. Symposium of Robotics Research (ISRR), San Francisco, CA (2005)
158. Wang, C.-C., Thorpe, C.: Simultaneous localization and mapping with detection and tracking of moving objects. In: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), Washington, DC, USA (2002)
159. Weigel, T., Gutmann, J.S., Dietl, M., Kleiner, A., Nebel, B.: CS Freiburg: Coordinating robots for successful soccer playing. IEEE Transactions on Robotics and Automation 18(5), 685–699 (2002)
160. Weiß, G., Wetzler, C., von Puttkamer, E.: Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Munich, Germany, pp. 595–601 (1994)
161. Whaite, P., Ferrie, F.P.: Autonomous exploration: Driven by uncertainty. IEEE Transactions on Pattern Analysis and Machine Intelligence 19(3), 193–205 (1997)
162. Wullschleger, F.H., Arras, K.O., Vestli, S.J.: A flexible exploration framework for map building. In: Proc. of the Third European Workshop on Advanced Mobile Robots, Zurich, Switzerland (1999)
163. Yamauchi, B.: Frontier-based exploration using multiple robots. In: Proc. of the Second International Conference on Autonomous Agents, Minneapolis, MN, USA, pp. 47–53 (1998)
164. Yamauchi, B., Schultz, A., Adams, W.: Integrating exploration and localization for mobile robots. Adaptive Behavior 7(2), 217–229 (1999)

165. Zelinsky, A., Jarvis, R., Byrne, J., Yuta, S.: Planning paths of complete coverage of an unstructured environment by a mobile robots. In: Proc. of the Int. Conf. on Advanced Robotics (ICAR), Tokyo, Japan, pp. 533–538 (1993)
166. Zheng, X., Jain, S., Koenig, D., Kempe, D.: Multi-robot forst coverage. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Edmonton Cananda (2005)
167. Zlot, R., Stenz, A.T., Dias, M.B., Thayer, S.: Multi-robot exploration controlled by a market economy. In: Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA), Washington, DC, USA (2002)

Index

active localization, 4
active loop-closing, 118, 152
  stopping criterion, 122
AdaBoost, 74

Bayes' rule, 181
Bayesian information criterion, 164
BIC, *see* Bayesian information criterion

classifier
  strong, 77
  weak, 74
CML, *see* simultaneous localization and mapping
collaboration, *see* coordinated exploration
concurrent mapping and localization, *see* simultaneous localization and mapping
cost, 46

dataset
  Acapulco Convention Center, 111
  Aces, 110
  Belgioioso, 110
  Bruceton mine, 111
  DLR, 83
  Edmonton Convention Center, 111
  Fort Sam Huston, 82
  Freiburg building 101, 110
  Freiburg building 106, 152
  Freiburg building 79, 33, 157
  Freiburg campus, 103
  Intel Research Lab, 83, 103
  MIT Killian Court, 104, 140
  Seattle, 110, 154
dynamic objects, 161
  low-dynamic, 161
  non-static, 161

effective number of particles, 101
effective sample size, *see* effective number of particles
entropy, 29
exploration, 4, 29
  coordinated, 43, 48, 80
  information gain-based, 30, 143
  limited communication, 49

features, 74, 76

hidden Markov model, 78
HMM, *see* hidden Markov model
Hungarian method, 59

independence, 181
information gain, 30, 148
  expected, 31, 147
integrated approaches, *see* simultaneous planning localization and mapping
interference, 43

k-means, 164
KLD, *see* Kullback-Leibler divergence
Kullback-Leibler divergence, 140

law of total probability, 182
localization, 3

map
  clustering, 165
  coverage, 24
  feature, 13
  geometric, 13
  grid, 13
  occupancy grid, 14
  patch-map, 166
  reflection probability grid, 18
mapping, 3
  with known poses, 14
marginalization, 182
Markov assumption, 182
MCL, *see* Monte Carlo localization
Monte Carlo localization, 11, 166

particle diversity, 135
particle filter
  adaptive resampling, 94
  importance weighting, 8, 95
  particle depletion, 10, 135
  proposal distribution, 10, 94
  Rao-Blackwellized, 93
  resampling, 8, 95
  sampling, 8, 95
  sampling importance resampling filter, 8
  target distribution, 10
  uncertainty, 145
path planning, 3
priorization, 64
product rule, 181

scan counting, 37
semantic place label, 73
sensor model, 16, 25
simultaneous localization and mapping, 3, 93
simultaneous planning localization and mapping, 4, 144
SLAM, *see* simultaneous localization and mapping
SPLAM, *see* simultaneous planning localization and mapping

traveling salesman problem, 66
TSP, *see* traveling salesman problem

uncertainty
  of a Gaussian, 147
  of a grid map, 146
  of a Rao-Blackwellized particle filter, 145
  reduction, *see* information gain
  trajectory, 147
utility, 48, 151
  expected, 151

Springer Tracts in Advanced Robotics

Edited by B. Siciliano, O. Khatib and F. Groen

Further volumes of this series can be found on our homepage: springer.com

Vol. 55: Stachniss, C.
Robotic Mapping and Exploration
196 p. 2009 [978-3-642-01096-5]

Vol. 54: Khatib, O.; Kumar, V.;
Pappas, G.J. (Eds.)
Experimental Robotics:
The Eleventh International Symposium
579 p. 2009 [978-3-642-00195-6]

Vol. 53: Duindam, V.; Stramigioli, S.
Modeling and Control for Efficient Bipedal
Walking Robots
211 p. 2009 [978-3-540-89917-4]

Vol. 52: Nüchter, A.
3D Robotic Mapping
201 p. 2009 [978-3-540-89883-2]

Vol. 51: Song, D.
Sharing a Vision
186 p. 2009 [978-3-540-88064-6]

Vol. 50: Alterovitz, R.; Goldberg K.
Motion Planning in Medicine: Optimization
and Simulation Algorithms for
Image-Guided Procedures
153 p. 2008 [978-3-540-69257-7]

Vol. 49: Ott, C.
Cartesian Impedance Control of Redundant
and Flexible-Joint Robots
190 p. 2008 [978-3-540-69253-9]

Vol. 48: Wolter, D.
Spatial Representation and
Reasoning for Robot
Mapping
185 p. 2008 [978-3-540-69011-5]

Vol. 47: Akella, S.; Amato, N.;
Huang, W.; Mishra, B.; (Eds.)
Algorithmic Foundation of Robotics VII
524 p. 2008 [978-3-540-68404-6]

Vol. 46: Bessière, P.; Laugier, C.;
Siegwart R. (Eds.)
Probabilistic Reasoning and Decision
Making in Sensory-Motor Systems
375 p. 2008 [978-3-540-79006-8]

Vol. 45: Bicchi, A.; Buss, M.;
Ernst, M.O.; Peer A. (Eds.)
The Sense of Touch and Its Rendering
281 p. 2008 [978-3-540-79034-1]

Vol. 44: Bruyninckx, H.; Přeučil, L.;
Kulich, M. (Eds.)
European Robotics Symposium 2008
356 p. 2008 [978-3-540-78315-2]

Vol. 43: Lamon, P.
3D-Position Tracking and Control
for All-Terrain Robots
105 p. 2008 [978-3-540-78286-5]

Vol. 42: Laugier, C.; Siegwart, R. (Eds.)
Field and Service Robotics
597 p. 2008 [978-3-540-75403-9]

Vol. 41: Milford, M.J.
Robot Navigation from Nature
194 p. 2008 [978-3-540-77519-5]

Vol. 40: Birglen, L.; Laliberté, T.; Gosselin, C.
Underactuated Robotic Hands
241 p. 2008 [978-3-540-77458-7]

Vol. 39: Khatib, O.; Kumar, V.; Rus, D. (Eds.)
Experimental Robotics
563 p. 2008 [978-3-540-77456-3]

Vol. 38: Jefferies, M.E.; Yeap, W.-K. (Eds.)
Robotics and Cognitive Approaches to
Spatial Mapping
328 p. 2008 [978-3-540-75386-5]

Vol. 37: Ollero, A.; Maza, I. (Eds.)
Multiple Heterogeneous Unmanned Aerial
Vehicles
233 p. 2007 [978-3-540-73957-9]

Vol. 36: Buehler, M.; Iagnemma, K.;
Singh, S. (Eds.)
The 2005 DARPA Grand Challenge – The Great
Robot Race
520 p. 2007 [978-3-540-73428-4]

Vol. 35: Laugier, C.; Chatila, R. (Eds.)
Autonomous Navigation in Dynamic
Environments
169 p. 2007 [978-3-540-73421-5]

Vol. 34: Wisse, M.; van der Linde, R.Q.
Delft Pneumatic Bipeds
136 p. 2007 [978-3-540-72807-8]
Vol. 33: Kong, X.; Gosselin, C.
Type Synthesis of Parallel
Mechanisms
272 p. 2007 [978-3-540-71989-2]

Vol. 30: Brugali, D. (Ed.)
Software Engineering for Experimental Robotics
490 p. 2007 [978-3-540-68949-2]

Vol. 29: Secchi, C.; Stramigioli, S.; Fantuzzi, C.
Control of Interactive Robotic Interfaces – A
Port-Hamiltonian Approach
225 p. 2007 [978-3-540-49712-7]

Vol. 28: Thrun, S.; Brooks, R.; Durrant-Whyte, H. (Eds.)
Robotics Research – Results of the 12th
International Symposium ISRR
602 p. 2007 [978-3-540-48110-2]

Vol. 27: Montemerlo, M.; Thrun, S.
FastSLAM – A Scalable Method for the
Simultaneous Localization and Mapping
Problem in Robotics
120 p. 2007 [978-3-540-46399-3]

Vol. 26: Taylor, G.; Kleeman, L.
Visual Perception and Robotic Manipulation – 3D
Object Recognition, Tracking and Hand-Eye
Coordination
218 p. 2007 [978-3-540-33454-5]

Vol. 25: Corke, P.; Sukkarieh, S. (Eds.)
Field and Service Robotics – Results of the 5th
International Conference
580 p. 2006 [978-3-540-33452-1]

Vol. 24: Yuta, S.; Asama, H.; Thrun, S.;
Prassler, E.; Tsubouchi, T. (Eds.)
Field and Service Robotics – Recent Advances in
Research and Applications
550 p. 2006 [978-3-540-32801-8]

Vol. 23: Andrade-Cetto, J,; Sanfeliu, A.
Environment Learning for Indoor Mobile Robots
– A Stochastic State Estimation Approach
to Simultaneous Localization and Map Building
130 p. 2006 [978-3-540-32795-0]

Vol. 22: Christensen, H.I. (Ed.)
European Robotics Symposium 2006
209 p. 2006 [978-3-540-32688-5]

Vol. 21: Ang Jr., H.; Khatib, O. (Eds.)
Experimental Robotics IX – The 9th International
Symposium on Experimental Robotics
618 p. 2006 [978-3-540-28816-9]

Vol. 20: Xu, Y.; Ou, Y.
Control of Single Wheel Robots
188 p. 2005 [978-3-540-28184-9]

Vol. 19: Lefebvre, T.; Bruyninckx, H.;
De Schutter, J. Nonlinear Kalman Filtering
for Force-Controlled Robot Tasks
280 p. 2005 [978-3-540-28023-1]

Vol. 18: Barbagli, F.; Prattichizzo, D.;
Salisbury, K. (Eds.)
Multi-point Interaction with Real
and Virtual Objects
281 p. 2005 [978-3-540-26036-3]

Vol. 17: Erdmann, M.; Hsu, D.; Overmars, M.;
van der Stappen, F.A (Eds.)
Algorithmic Foundations of Robotics VI
472 p. 2005 [978-3-540-25728-8]

Vol. 16: Cuesta, F.; Ollero, A.
Intelligent Mobile Robot Navigation
224 p. 2005 [978-3-540-23956-7]

Vol. 15: Dario, P.; Chatila R. (Eds.)
Robotics Research – The Eleventh
International Symposium
595 p. 2005 [978-3-540-23214-8]

Vol. 14: Prassler, E.; Lawitzky, G.; Stopp, A.;
Grunwald, G.; Hägele, M.; Dillmann, R.;
Iossifidis. I. (Eds.)
Advances in Human-Robot Interaction
414 p. 2005 [978-3-540-23211-7]

Vol. 13: Chung, W.
Nonholonomic Manipulators
115 p. 2004 [978-3-540-22108-1]

Vol. 12: Iagnemma K.; Dubowsky, S.
Mobile Robots in Rough Terrain –
Estimation, Motion Planning, and Control
with Application to Planetary Rovers
123 p. 2004 [978-3-540-21968-2]

Vol. 11: Kim, J.-H.; Kim, D.-H.; Kim, Y.-J.;
Seow, K.-T.
Soccer Robotics
353 p. 2004 [978-3-540-21859-3]

Vol. 10: Siciliano, B.; De Luca, A.; Melchiorri, C.;
Casalino, G. (Eds.)
Advances in Control of Articulated and
Mobile Robots
259 p. 2004 [978-3-540-20783-2]

Vol. 9: Yamane, K.
Simulating and Generating Motions
of Human Figures
176 p. 2004 [978-3-540-20317-9]