Dissertation zur Erlangung des Doktorgrades der Fakultät für Angewandte Wissenschaften der Albert-Ludwigs-Universität Freiburg im Breisgau

Mobile Robot Navigation in Dynamic Environments

Maren Bennewitz

Betreuer:
Prof. Dr. Wolfram Burgard

Dekan der Fakultät für Angewandte Wissenschaften:
Prof. Dr. Thomas Ottmann

1. Gutachter: Prof. Dr. Wolfram Burgard

2. Gutachter: Prof. Dr. Raja Chatila

Tag der Disputation: 28. Juni 2004

Zusammenfassung

Service-Roboter, die in von Menschen bevölkerten Umgebungen agieren, sind in den letzten Jahren immer populärer geworden. Es existieren schon eine Reihe von Systemen, die beispielsweise in Krankenhäusern, Bürogebäuden, Kaufhäusern und Museen eingesetzt werden. Darüber hinaus sind auch verschiedene Mehrrobotersysteme entwickelt worden, da einige Aufgaben von einem Team von Robotern schneller und effizienter erledigt werden können als von einem einzelnen Roboter. Dazu gehören unter anderem Reinigungsarbeiten, Auslieferungsaufträge und das Erkunden von unbekannten Umgebungen.

Immer wenn Teams von mobilen Robotern in der selben Umgebung eingesetzt werden, müssen ihre Bewegungen koordiniert werden, damit die einzelnen Roboter sich nicht gegenseitig behindern. Außerdem sollte ihre gemeinsame Aufgabe so schnell wie möglich erledigt werden. Um diese Ansprüche zu erfüllen, werden komplexe Pfadplanungtechniken benötigt. Da der gemeinsame Konfigurationsraum der Roboter in der Regel extrem groß ist und exponentiell mit der Anzahl der Roboter wächst, können existierende Pfadplanungtechniken für einzelne Roboter nicht unmittelbar auf Mehrrobotersysteme übertragen werden.

Viele existierende Methoden für Mehrrobotersysteme sind „entkoppelt“, was bedeutet, dass sie zuerst Pfade für die einzelnen Roboter unabhängig voneinander planen. Anschließend überprüfen sie, ob sich die Roboter zu nahe kämen, wenn sie sich entlang dieser Pfade bewegten. In solchen Fällen werden die Pfade neu berechnet, um diese Konflikte zu umgehen. Dabei weisen viele entkoppelte Methoden den einzelnen Robotern Prioritäten zu. Diese geben an, in welcher Reihenfolge die Pfade neu berechnet werden. Bei der Berechnung des Pfades für einen Roboter werden dabei die Pfade aller Roboter mit höherer Priorität als gegeben und unveränderbar angesehen. Auf diese Weise wird der Suchraum extrem eingeschränkt und die Suche nach einer Lösung des kombinierten Planungsproblems beschleunigt. Die meisten existierenden entkoppelten Ansätze benutzen dabei ein festes Prioritätsschema, d.h. eine feste Ordnung der Roboter. Die Reihenfolge, in der Pfade berechnet werden, hat jedoch einen großen Einfluss darauf, ob überhaupt eine Lösung für das kombinierte Pfadplanungsproblem gefunden werden kann und darauf, wie effizient diese Lösung für das gesamte Mehrrobotersystem ist.

Im ersten Teil dieser Dissertation stellen wir einen Ansatz vor, der in dem Raum aller Prioritätsschemata nach einer Ordnung der Roboter sucht, für die eine Lösung des Pfadplanungsproblems berechnet werden kann. Dabei nutzt unser Verfahren Einschränkungen (Constraints) zwischen den Prioritäten der Roboter aus, welche automatisch von der Aufgabenspezifikation abgeleitet werden. Nachdem ein geeignetes Prioriätsschema gefunden wurde, versucht unser Verfahren dieses mit-

hilfe einer Hill-Climbing-Strategie zu verbessern. Unsere Suchmethode kann für beliebige entkoppelte Planungsysteme eingesetzt werden. In verschiedenen Experimenten mit einem realen Mehrrobotersystem sowie in Simulationen zeigen wir, dass unser Verfahren effiziente Lösungen auch für komplizierte Pfadplanungsprobleme finden kann.

Der zweite Teil dieser Dissertation konzentriert sich auf Roboter, die in von Menschen genutzten Umgebungen eingesetzt werden. Diese Systeme können den Service und ihr Verhalten gegenüber Personen verbessern, wenn sie auf die Aktivitäten der umgebenden Menschen reagieren und nicht mit ihnen interferieren. Im Gegensatz zu einem Mehrrobotersystem sind die zukünftigen Bewegungen von Menschen aber nicht bekannt. Deswegen müssen die Roboter in der Lage sein, die Menschen mittels ihrer Sensoren wahrzunehmen, zu identifizieren und ihre Intentionen zu lernen, damit sie bessere Vorhersagen über das Verhalten der Menschen machen können. In dieser Dissertation stellen wir eine Technik vor, die typische Bewegungsmuster von Personen aus Sensordaten mithilfe des EM-Algorithmus' lernt. Wir beschreiben außerdem, wie die gelernten Muster dazu benutzt werden können, um potentielle zukünftige Bewegungen der Personen vorherzusagen. Anschließend erklären wir, wie dieses Wissen im Pfadplanungprozess eines mobilen Roboters berücksichtigt werden kann. Danach führen wir eine Methode ein, die aus den gelernten Verhaltensmustern automatisch Hidden Markov Modelle (HMMs) ableitet. Diese HMMs können von einem mobilen Roboter benutzt werden, um die Positionen von mehreren Personen vorherzusagen, auch wenn sie außerhalb seines Sichtfelds sind. Um die HMMs mithilfe von Kamera- und Laserdaten zu aktualisieren, wenden wir Joint Probabilistic Data Association Filter an. In der Regel wird ein Roboter unsicher über die Positionen von Personen, wenn er sie längere Zeit nicht beobachtet. Deswegen untersuchen wir auch, wie entscheidungstheoretisch geeignete Beobachtungsaktionen bestimmt werden können, welche ausgeführt werden, während der Roboter seine sonstigen Aufgaben verrichtet.

Praktische Experimente, die wir mit unserem mobilen Roboter durchgeführt haben, zeigen, dass

- unsere Methode typische Bewegungsmuster von Personen lernen kann,
- das Navigationsverhalten des Roboters verbessert werden kann, indem er die gelernten Muster benutzt, um Vorhersagen über die Bewegungen von Personen zu machen,
- die abgeleiteten HMMs eingesetzt werden können, um zuverlässig einen probabilistischen Glauben über die Positionen von mehreren Personen zu behalten, auch wenn sie gerade nicht im Sichtfeld des Roboters sind, und

- unsere Technik effektive Aktionen generiert, welche die Unsicherheit des Roboters über die Positionen von Personen stark reduzieren.

Unser Ansatz ist nützlich für Serviceroboter verschiedenster Art, da es in vielen Anwendungen hilfreich ist zu wissen, wo sich die Personen in der Umgebung aufhalten. Beispielsweise kann ein Roboter persönliche Botendienste effizienter ausführen, wenn er weiß, wo sich die Personen gerade befinden. Für einen Putzroboter ist es ebenfalls interessant zu wissen, welche Räume gerade leer sind, damit er niemanden stört. Darüber hinaus kann ein Haushaltsrobototer sein Verhalten verbessern, wenn er weiß, wo eine Person gerade ist oder wo sie hingeht. Dadurch kann der Roboter sich beispielsweise so bewegen, dass er der Person nicht im Weg steht und er kann sich strategisch günstig für Interaktionen positionieren.

Zusammengefasst präsentieren wir Techniken, welche das Zusammenleben von Mensch und Roboter sowie deren Interaktion erleichtern.

Summary

Service robots which act in environments populated by humans have become very popular in the last few years. A variety of systems exists which act for example in hospitals, office buildings, department stores, and museums. Furthermore, several multi-robot systems have been developed for tasks which can be accomplished more efficiently by a whole team of robots than just by a single robot. These tasks include surface cleaning, deliveries, and the exploration of unknown terrain. Whenever teams of mobile robots are operating in the same environment their motions have to be coordinated in order to avoid congestions or collisions. At the same time the robots should perform their navigation tasks in a minimum amount of time. Thus, sophisticated path planning techniques are needed that fulfill these requirements. Since the joint configuration space of the robots is typically huge and grows exponentially with the number of robots, existing path planning methods for single robot systems cannot directly be transferred to multi-robot systems.

Many existing path planning methods for multi-robot systems are decoupled, which means that they first plan paths for the individual robots independently. Afterward, they check if the robots would get too close to each other if the paths were executed. In such a case the paths are recomputed to avoid these conflicts. Many decoupled methods assign priorities to the individual robots. These priorities define an order in which the paths of the robots have to be recomputed. By computing the path of a robot, the paths of the robots with higher priority are considered as fixed. This way, the size of the search space is extremely reduced. Most of the existing prioritized decoupled methods use a fixed priority scheme (order of the robots). However, the order in which the paths of the robots are recomputed has a serious influence on whether a solution can be found at all and on how efficient the solution is for the overall multi-robot system.

In the first part of this thesis we present an approach which searches in the space of all priority schemes to find an order of the robots for which a solution to the path planning problem can be computed. During the search, we utilize constraints between the priorities of the robots which are automatically derived from the task specification. After an appropriate priority scheme has been found, our technique tries to improve it by using a hill-climbing strategy. Our search method can be used to find and optimize paths generated by any prioritized path-planning technique. In several experiments with a real-robot system as well as in simulation we show that our approach produces efficient solutions even for difficult path planning problems.

The second part of this thesis is focused on robots acting in environments populated by humans. These systems can improve their behavior if they react appropriately to the activities of the surrounding people and do not interfere with

them. In contrast to a multi-robot path planning system, the future movements of people are not known. Therefore, the robots have to be able to detect people with their sensors, to identify them, and to learn their intentions in order to be able to make better predictions of their future behavior. In this thesis we present an approach to learn typical motion patterns of people from sensor data using the EM algorithm. Furthermore, we describe how the learned patterns can be used to predict future movements of the people. Afterward, we explain how this knowledge can be integrated into the path planning process of a mobile robot. Finally, we introduce a method which automatically derives Hidden Markov Models (HMMs) from the learned motion models. These HMMs can be used by a mobile robot to predict the positions of multiple persons even when they are outside its field of view. To update the HMMs based on laser-range data and vision information we apply Joint Probabilistic Data Association Filters. In practice, the robot becomes uncertain about the positions of people if it does not observe them for a long period of time. We therefore propose a decision-theoretic approach to determine observation actions that are carried out while the robot is executing its tasks. Practical experiments carried out with our mobile robot demonstrate

- that our method is able to learn typical motion patterns of people,
- that the navigation behavior of the robot can be improved by predicting the motions of people based on the learned motion patterns,
- that the derived HMMs can be used to reliably maintain a probabilistic belief about the current positions of multiple persons even if they are currently not in its field of view, and
- that our technique generates effective actions that seriously reduce the uncertainty in the belief about the positions of people.

Our approach is useful for service robots of various types that are designed to coexist with humans. In many tasks it is helpful to know the current locations of the people in the environment. For example, this knowledge enables a robot to more efficiently carry out personal delivery tasks since the number of detours is reduced. Also a cleaning robot that knows which rooms are currently empty can carry out its tasks without disturbing anyone. Furthermore, a home care robot can improve its behavior by knowing where the person it is providing service to currently is or where it is going to. The robot can then, for instance, generate motion actions that avoid interferences with the person. Additionally, this knowledge allows strategic positioning of the robot for providing personal assistance.

In summary, we present techniques which facilitate the coexistence of robots and humans in real world environments as well as the interaction between them.

Acknowledgments

Many people are responsible for the quality of this thesis. First of all I would like to thank Wolfram Burgard for supporting me over all these years. It was a pleasure to work with him as my advisor. Actually, I cannot think of a more competent, more enthusiastic and also a more critical advisor. I have learned a lot during the last few years. Special thanks go to Sebastian Thrun for coming up with many interesting ideas which highly influenced this thesis and for fruitful discussions during my visits at CMU. I also would like to thank Raja Chatila for being my very competent co-examiner.

I would like to thank all my friends and colleagues at our lab for the pleasant atmosphere. Everyone has been open-minded to discussions which contributed to this thesis. This thesis could not have been done without the help and support of numerous people. Especially, I would like to thank:

Dirk Hähnel for spending a lot of time helping me to solve soft- and hardware problems with Albert,

Greg Cielniak for the great collaboration on the vision-based people tracking system,

Kristian Kersting, Daniel Sack, Cyrill Stachniss, Moritz Tacke, Rudi Triebel, and Michael Veeck who helped me to carry out several experiments and gave a lot of helpful suggestions for improving the manuscript,

Dirk Zitterell and Patrick Pfaff for accompanying and supporting our group for many years,

Julio Pastrana for helping me to carry out experiments with Albert,

Luis Montesano for showing me how to optimize my Kalman filter implementation,

Thilo Weigel for his great support during the experiments with the robots of the CS-Freiburg RoboCup team,

and Susanne Bourjaillat for help on administrative matters and for cheering up the department.

Above all I wish to thank my family who have always supported me and Cyrill for always being at my side, for cooking marvelous meals, and for enjoying life with our friends and me.

Für meine Eltern, Cyrill und meine Freunde.

Contents

1 Introduction **1**

2 Multi-Robot Path Planning **7**
2.1 Introduction 7
2.2 Related Work 10
2.3 Prioritized A^*-based Path Planning and Path Coordination 13
2.3.1 A^*-based Path Planning 13
2.3.2 Prioritized Decoupled Path Planning for Teams of Robots 16
2.3.3 Using A^* for Prioritized Path Planning 17
2.4 Finding and Optimizing Solvable Priority Schemes 20
2.4.1 The Randomized Search Technique 21
2.4.2 Utilizing Constraints to Focus the Search 21
2.5 Experimental Results 26
2.5.1 An Example with Real Robots 26
2.5.2 Simulation Experiments 27
2.6 Conclusion 34

3 Learning Motion Patterns of People **39**
3.1 Introduction 39
3.2 Using EM to Learn Motion Patterns 40
3.2.1 The EM Algorithm 41
3.2.2 Representing Motion Patterns by Gaussian Distributions . 42
3.2.3 Monitoring Convergence and Local Maxima 46
3.2.4 Estimating the Number of Model Components 46
3.3 Computing the Likelihood of Motion Patterns 47
3.4 Laser-based Data Acquisition 48
3.5 Experimental Results 49
3.5.1 Learning Results 51
3.5.2 Prediction Accuracy 63
3.6 Representing Motion Patterns by Flow-Fields 65
3.6.1 Application of the EM Algorithm 66

3.6.2 Drawback of Flow-Fields in the Context of Clustering . . 67
3.7 Related Work 69
3.8 Conclusion 71

4 Adapting Navigation Strategies 73
4.1 Introduction 73
4.2 Integrating Predicted Motions of People into Path Planning 74
4.3 Detecting and Tracking People 77
4.3.1 The Kalman Filter 77
4.3.2 Tracking People Using Kalman Filters 80
4.3.3 Keeping Track of Multiple Persons 81
4.4 Experimental Results 82
4.4.1 Planning Detours 82
4.4.2 Giving Space to People 83
4.4.3 Multiple Persons 84
4.4.4 A Comparison to Linear Prediction 86
4.5 Related Work 89
4.6 Conclusion 91

5 Using HMMs to Estimate Positions of Multiple Persons 93
5.1 Introduction 93
5.2 Markov Chains and Hidden Markov Models 94
5.2.1 Markov Chains 94
5.2.2 Hidden Markov Models (HMMs) 95
5.3 Deriving an HMM from Learned Motion Patterns 96
5.4 Person Detection and Identification 97
5.5 Active Localization of People 103
5.6 Experimental Results 105
5.6.1 Tracking People 106
5.6.2 Deciding to Perform Observation Actions 112
5.7 Related Work 118
5.8 Conclusion 120

6 Conclusions 123

A Appendix 127
A.1 The B21r Robot Albert 127
A.2 Probability Theory 128
A.2.1 Bayes' Rule 128
A.2.2 Product Rule 128
A.2.3 Marginalization 128

A.2.4 Law of Total Probability 129
A.3 Proof: EM Monotonically Increases the Data Likelihood 129
A.4 Approximation of the Probability $P(\psi \mid z^{(1:t)})$ 130
A.5 Computation of the Assignment Probabilities λ_{sr} 131

Chapter 1

Introduction

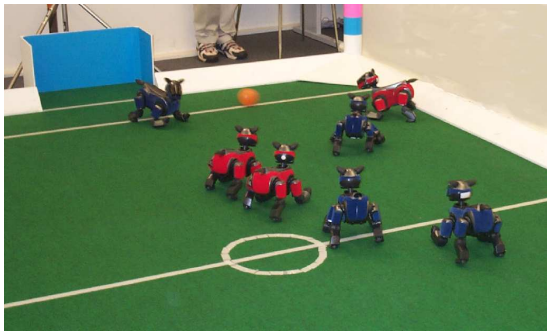
Service robots are envisioned to coexist with humans and to fulfill various kinds of tasks. In the last few years there has been a substantial progress in the field of service robots. A variety of mobile robots that are designed to operate in environments populated by humans has already been developed. These robots, for example, have been deployed in hospitals, office buildings, department stores, and museums. Existing robotic systems are already able to perform various services such as delivery, education, providing tele-presence, cleaning, or entertainment. Furthermore, there are prototypes of autonomous wheelchairs and intelligent service robots which are designed to assist people in their homes[1]. Figure 1.1 depicts four examples of existing robotic systems. The upper left image shows a cleaning robot which is designed to clean large surfaces, for example in supermarkets or airports [Hefter, 2004]. The robot in the upper right image has been developed within the EU project WebFAIR [2004]. The goal of this project is to build an interactive tele-presence system which provides individual access to exhibitions and trade-fairs by the Internet. The lower left image shows entertainment robots [Sony, 2003] and the lower right image depicts one of the robots installed at Swiss EXPO 2002 [Swiss Federal Institute of Technology Lausanne, 2002], which guided the visitors through a part of the exhibition.

Since some tasks can be carried out more efficiently by a team of robots than by just a single one, multi-robot systems have become popular. Application areas for multi-robot systems are for example surface cleaning, delivery tasks, the

[1]Hospitals: [King and Weiman, 1990, Engelberger, 1993], office buildings: [Horswill, 1993, Nourbakhsh *et al.*, 1995, Asoh *et al.*, 1997, Simmons *et al.*, 1997, Arras and Vestli, 1998, Alami *et al.*, 2000], department stores: [Gross and Boehme, 2000], museums: [Burgard *et al.*, 1999, Thrun *et al.*, 2000, Siegwart *et al.*, 2003], tele-presence: [Hirzinger *et al.*, 1994, Schulz *et al.*, 2000, Goldberg and Siegwart, 2001], cleaning: [Endres *et al.*, 1998], entertainment: [Weigel *et al.*, 2002], autonomous wheelchairs: [Lankenau and Röfer, 2000, Kluge *et al.*, 2001], home care: [Lacey and Dawson-Howe, 1998, Schaeffer and May, 1999, Montemerlo *et al.*, 2002a].

(a) 

(b) 

(c) 

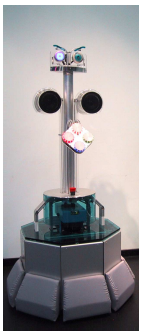
(d) 

Figure 1.1: Various types of service robots: (a) the *VARIOTECH* cleaning robot, (b) the interactive tourguide robot Albert, (c) AIBO Entertainment Robots, and (d) one of the robots installed at Swiss EXPO 2002.

exploration of unknown terrain[2] and robotic soccer (a scene from the 4-Legged League is depicted in Figure 1.1 (c)). Whenever teams of mobile robots are operating in the same environment their motions have to be coordinated in order to avoid deadlocks and congestions (see Figure 1.2) or even collisions. At the same time the robots should perform the navigation tasks in a minimum amount of time. Thus, sophisticated path planning techniques are needed to fulfill these requirements.

The path planning techniques for single robot systems (see e.g. the book by Latombe [1991]) cannot be directly transferred to multi-robot systems. Planning the paths for teams of mobile robots is significantly more complex than the path planning problem for single robots. This is due to the fact that the search space of a composite planning problem is typically extremely large. More precisely, the size of the joint state space of the robots grows exponentially with the number of robots.

The existing methods for solving the problem of motion planning for multi-

[2]Cleaning tasks: [Jäger and Nebel, 2002], delivery tasks: [Alami *et al.*, 1998b], exploration tasks: [Simmons *et al.*, 2000, Burgard *et al.*, 2000].

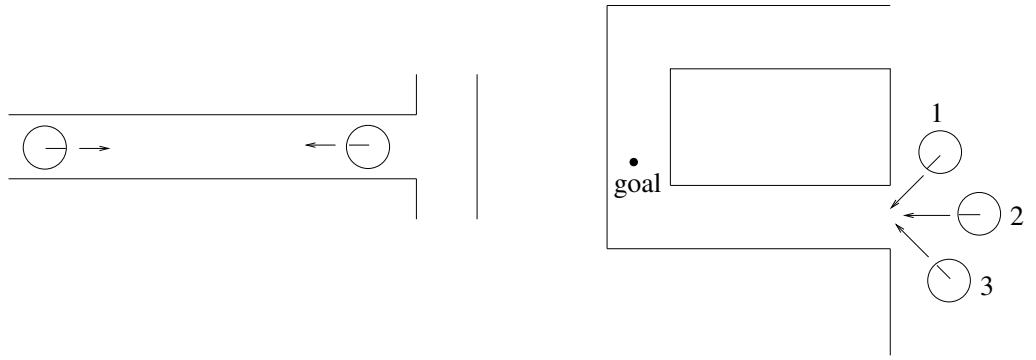


Figure 1.2: The situation depicted in the left image shows a deadlock between two robots which can occur in a narrow corridor. The right image depicts a congestion with several robots. In the second situation it would be better for robot 1 to make a detour and choose the way via the upper corridor. These two examples demonstrate that there is a need for coordinating the motions whenever a team of mobile robots is deployed in the same environment.

robot systems can be roughly divided into two categories: *Centralized approaches* combine the configuration spaces of the individual robots into one composite configuration space which is then searched for a solution for the whole composite system. *Decoupled approaches* in contrast first compute separate paths for the individual robots independently. Then they try to solve existing conflicts based on the independently computed paths. Conflicts are situations in which robots would get too close to each other if the paths were executed.

There are two important criteria to evaluate path planning methods:

1. *Completeness:* Is the path planning system able to compute a solution to any multi-robot path planning problem for which a solution exists?

2. *Optimality:* Is the solution as efficient as possible considering the whole team of robots?

While the general centralized approach, which performs an unconstrained search in the composite configuration space, is able to find the optimal solution to any planning problem for which a solution exists, its time complexity is exponential in the dimension of the composite configuration space. Therefore, it can typically not be applied to real world systems since those systems have to act under serious time constraints. In practice it is necessary to use heuristics for the exploration of the huge joint state space or to constrain the configuration space. As a consequence, practical centralized approaches cannot ensure completeness and optimality.

Many decoupled methods use a priority scheme for the robots. This means that a unique priority is assigned to each robot. The robots are then processed in the order implied by these priorities. During path planning for one robot the paths of the robots with higher priority are considered. This way the size of the search space is reduced to make the search tractable. Since all decoupled methods strongly restrict the search space they are generally incomplete and may also generate sub-optimal paths for the robots.

The order in which prioritized approaches compute the paths of the robots has a serious influence on whether a solution can be found and on the quality of the solution. No single prioritization will be sufficient for all possible multi-robot motion problems. In the first part of this thesis we present an approach to prioritized decoupled path planning that performs a hill-climbing search in the space of priority schemes. To find solvable priority schemes[3] even for large teams of robots, constraints derived from the task specification are used to guide the search. Extensive experiments on real robots and in simulation runs will show that our approach enables decoupled path planning methods to find efficient solutions even for complex multi-robot problems.

In the second part of this work we focus on robotic systems operating in environments populated by humans. Such systems can improve their service if they react appropriately to the activities of the people in their surrounding and do not interfere with them. In contrast to path planning for a team of mobile robots the intentions and future trajectories of people are not accessible. Therefore, it is necessary that the robots can locate and track people using their sensors. Furthermore, the robots need to be able to identify and potentially learn intentions of people so that they can make better predictions about their future actions. In the past few years various approaches have been presented to track the positions of people and to predict their short-term motions. All these approaches assume that motion models of the people are given. A lot of research has already been focused on the problem of learning and recognizing behaviors or plans of humans. Additionally, systems have been developed to detect atypical behaviors or unusual events.

In this thesis we present an approach that, in contrast to the previous approaches, enables a mobile robot

- to learn typical motion patterns of people from sensor data,
- to adapt its navigation behavior by predicting trajectories of people, and
- to utilize the learned motion patterns to maintain a belief about where the people are.

[3] We denote a priority scheme as solvable if collision-free paths for all robots can be found if the paths of the robots are planned in the order implied by the priority scheme.

Such capabilities can be useful in various kinds of situations. For example, they allow a robot to reliably predict the trajectory of a person so that it avoids blocking the path of that person. Furthermore, a home care robot can more robustly keep track of the person it is providing service to and this way increase the time it stays in the vicinity of the person, for example to support interactions [Chatila *et al.*, 2002]. Thus, the knowledge about motion patterns of a person is quite useful for various tasks such as collision avoidance, strategic positioning, and verbal assistance.

The remainder of this thesis is organized as follows: In the following chapter we consider the problem of planning the paths for teams of robots. We present our approach to prioritized decoupled path planning that searches in the space of priority schemes. After this we focus on environments populated by humans and describe in Chapter 3 how to learn typical motion patterns of people from sensor data. In Chapter 4 we demonstrate how a mobile robot can use the learned motion patterns to improve its navigation behavior. In Chapter 5 we explain how to derive Hidden Markov Models (HMMs) from the learned motion patterns. These HMMs are used to estimate the positions of multiple persons and are updated based on observations made by a mobile robot.

Parts of thesis have been published in the following articles, conference and workshop papers:

- M. Bennewitz, W. Burgard, G. Cielniak, and S. Thrun. Learning Motion Patterns of People for Compliant Robot Motion. In: *The International Journal of Robotics Research*, 2004, to appear.

- M. Bennewitz, G. Cielniak, and W. Burgard. Utilizing Learned Motion Patterns to Robustly Track Persons. In: *Proceedings of the Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS)*, 2003.

- G. Cielniak, M. Bennewitz, and W. Burgard. Where is ...? Learning and Utilizing Motion Patterns of Persons with Mobile Robots. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

- M. Bennewitz, W. Burgard, and S. Thrun. Adapting Navigation Strategies Using Motions Patterns of People. In: *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2003.

- M. Bennewitz, W. Burgard, and S. Thrun. Using EM to Learn Motion Behaviors of Persons with Mobile Robots. In: *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, 2002.

- M. Bennewitz, W. Burgard, and S. Thrun. Learning Motion Patterns of Persons for Mobile Service Robots. In: *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2002.

- M. Bennewitz, W. Burgard, and S. Thrun. Finding and Optimizing Solvable Priority Schemes for Decoupled Path Planning Techniques for Teams of Mobile Robots. In: *Robotics and Autonomous Systems*, Vol. 41, 2002.

- M. Bennewitz, W. Burgard, and S. Thrun. Exploiting Constraints During Prioritized Path Planning for Teams of Mobile Robots. In: *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, 2001.

- M. Bennewitz, W. Burgard, and S. Thrun. Optimizing Schedules for Prioritized Path Planning of Multi-Robot Systems. In: *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2001.

- M. Bennewitz and W. Burgard. An Experimental Comparison of Path Planning Techniques for Teams of Mobile Robots. In: *Tagungsband des 16. Fachgesprächs Autonome Mobile Systeme (AMS)*, 2000, in German.

- M. Bennewitz and W. Burgard. Coordinating the Motions of Multiple Mobile Robots Using a Probabilistic Model. In: *Proceedings of the 8th International Symposium on Intelligent Robotic Systems (SIRS)*, 2000.

Chapter 2

Multi-Robot Path Planning

2.1 Introduction

Path planning is one of the fundamental problems in mobile robotics. As stated by Latombe [1991], the capability of effectively planning its motions is "eminently necessary since, by definition, a robot accomplishes tasks by moving in the real world."

The problem of coordinating multiple mobile robots has received considerable attention in the robotics literature. Whenever several robots are deployed in the same environment there is the need for coordinating their movements. Trajectories for the individual robots have to be computed such that collisions between the robots and static obstacles as well as between the robots among themselves are avoided. Especially in the context of multi-robot systems different undesirable situations can occur, such as congestions or deadlocks. As an example, consider the situation with three robots depicted in Figure 2.1. The starting positions of the robots are indicated by large circles whereas the small dots correspond to the goal locations. The lines are the individual optimal paths for the robots. Assuming that the corridors are too narrow to allow two robots to pass by, no path can be found for robot 1, if robot 3 enters the corridor before robot 1 has left it. In that case robot 3 blocks the way of robot 1 such that it cannot reach its designated target point G_1. This example shows that there is the need of coordinating the motions whenever teams of robots are operating in the same environment.

The existing methods for solving the problem of motion planning for multiple robots can roughly be divided into two major categories [Latombe, 1991]: the *centralized* and the *decoupled* techniques. In the centralized approach the configuration spaces of the individual robots are combined into one composite configuration space which is then searched for a path for the whole composite system [Schwartz and Scharir, 1983, Tournassoud, 1986, Barraquand and

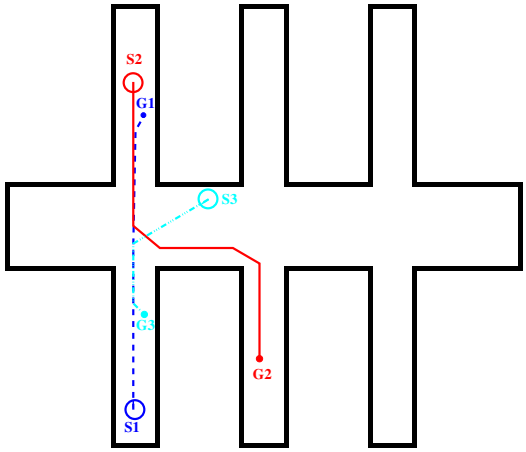


Figure 2.1: Path planning problem with three robots. The lines are the individual optimal paths for the robots between their current positions (indicated by large circles) and their goal locations (indicated by small dots).

Latombe, 1990, Barraquand *et al.*, 1992, McHenry, 1998]. Because the size of the joint configuration space grows exponentially with the number of robots, this approach, in general, suffers intrinsic scaling limitations. The major alternative are decoupled approaches [Erdmann and Lozano-Pérez, 1987, O'Donnell and Lozano-Pérez, 1989, Liu *et al.*, 1989, Buckley, 1989, Warren, 1990, Chu and EiMaraghy, 1992, Chai *et al.*, 1995, Souccar and Roderic, 1996, Azarm and Schmidt, 1996, Ferrari *et al.*, 1998, Leroy *et al.*, 1999]. Decoupled path planning systems first compute an individual path for each robot independently. Subsequently, they apply heuristics for resolving conflicts between the paths of different robots. Conflicts are situations in which the robots attempt to occupy the same location at the same time or in which they would get too close to each other.

A centralized path planning method which searches in the unconstrained composite configuration space is able to find the optimal solution to any planning problem for which a solution exists. Its time complexity, however, is exponential in the number of robots [Reif, 1979, Schwartz *et al.*, 1987]. Practical centralized approaches therefore either use heuristics to explore the huge joint state space, or constrain the configuration space to make the search feasible. As a result, they are typical neither complete nor optimal. Which means that they may fail to find a solution even if there is one and that the solution they generate may not be the optimal one.

As explained before decoupled planners first determine the paths of the individual robots independently and then employ different strategies to resolve possible conflicts. To deal with the still large search space it is common practice to as-

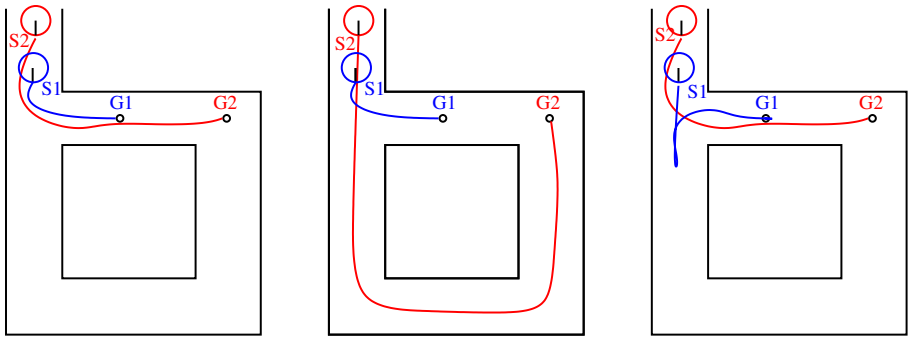


Figure 2.2: Independently planned paths for two robots (left image), sub-optimal solution if robot 1 has higher priority (center image), and more efficient solution which results if the path for robot 2 is planned first (right). As can be seen, then robot 1 has to move aside in order to let robot 2 pass by.

sign *priorities* to the individual robots [Erdmann and Lozano-Pérez, 1987, Buckley, 1989, Warren, 1990, Azarm and Schmidt, 1996, Ferrari *et al.*, 1998]. The replanning step is then performed in accordance with these priorities. Thus, in the case of conflicts, prioritized approaches try to compute a new collision-free path for each robot given the paths of the robots with higher priority. Priority schemes provide an effective mechanism for resolving conflicts that is computationally extremely efficient. Since they strongly restrict the search space, all decoupled techniques are also incomplete and generate potentially sub-optimal solutions.

For decoupled methods the order in which prioritized approaches compute the paths are planned has a serious influence on whether at all a solution can be found and on how long the resulting paths are. To illustrate this, let us consider two examples. Figure 2.1 shows a situation in which no solution can be found if robot 3 has a higher priority than robot 1. Since then the path of robot 3 is planned without considering robot 1, it will enter the corridor containing its target location (marked G_3) before robot 1 has left this corridor. Because the corridors are too narrow to allow two robots to pass by, robot 3 will block the way of robot 1 so that it cannot reach its target point G_1. However, if we change the priorities and plan the trajectory of robot 1 before that of robot 3, then robot 3 considers the trajectory of robot 1 during path planning and thus will wait in the hallway until robot 1 has left the corridor.

Another example is shown in Figure 2.2. The left image depicts the optimal paths of two robots. As can be seen, if the path of robot 1 is planned first without considering robot 2, then the collision-free path of robot 2 includes a large detour (see center image of Figure 2.2). This is because robot 1 blocks the upper corridor. However, if the path of robot 2 is planned first and after that the path for robot 1

is planned taking into account the path of robot 2 we obtain a much more efficient solution (see right image of Figure 2.2).

These two examples illustrate that the priority scheme, which specifies the order in which the paths of the robots are computed, has a serious influence on whether a solution can be found and on the quality of the solution. Moreover, it suggests that no single prioritization will be sufficient for all possible multi-robot motion problems.

In this chapter, we present a technique that searches in the space of all priority schemes while solving complex multi-robot planning problems. Our approach performs a randomized hill-climbing search in the space of priority schemes. Since each change of a scheme requires the computation of the paths for many of the robots, it is important to focus the search. Our method achieves this by utilizing constraints between the different robots, which are derived from the task specification. This has two serious advantages. First, it reduces the time required to find a solution, and second, it increases the number of problems for which a solution can be found in a given amount of time. Additionally, our algorithm is able to reduce the overall move cost once a solution has been found. It has anytime characteristics, which means that the quality of the solution depends on the available computation time.

This chapter is organized as follows. In the next section we present related work to multi-robot path planning emphasizing on prioritized decoupled methods. In Section 2.3 we introduce the two prioritized decoupled path planning techniques that are used throughout this work. Section 2.4 describes our approach to searching for solvable priority schemes during planning. Finally, in Section 2.5, we present systematic experimental results illustrating the capabilities of our approach to finding and optimizing solvable priority schemes.

2.2 Related Work

As already mentioned the idea of centralized methods is to consider the composite configuration space of all robots and to search for a solution for the whole composite system. To make centralized methods applicable in practice good heuristics for the exploration of the composite configuration space are needed or, the search space has to be constrained, thereby loosing completeness and optimality.

Many centralized methods use potential field techniques to guide the search [Tournassoud, 1986, Barraquand and Latombe, 1990, Barraquand *et al.*, 1992]. These techniques apply different approaches to deal with the problem of local minima in the potential function. Other methods restrict the motions of the robots to reduce the size of the search space. For example, LaValle and Hutchinson [1996] and Sveska and Overmars [1995] restrict the trajectories of the robots to lie on

independent roadmaps. The coordination is achieved by searching in the Cartesian product of the separate roadmaps. The individual roadmaps are constructed beforehand by randomly generating collision-free configurations and connecting them using some local planner. Kavraki *et al.* [1996] presented a similar approach for robotic systems with many degrees of freedom. They directly build a probabilistic roadmap (PRM) for the whole system. Once a roadmap has been learned it can be used to get collision-free paths for different configurations of the robots as long as the environment does not change. Bohlin and Kavraki [2000] proposed a variant which reduces the number of collision checks for the sampled configurations. Their goal has been to speed up the roadmap construction phase to efficiently answer single planning queries. The latter two roadmap methods, however, are not feasible for path planning problems with many robots. For this purpose the authors suggest to use a prioritized variant.

Decoupled planners, in contrast, first compute paths for the individual robots independently and then try to solve possible conflicts between these paths. A popular decoupled approach is planning in the configuration time-space [Erdmann and Lozano-Pérez, 1987], which extends the configuration space of the robot by a time axis. Techniques of this type assign priorities to the individual robots and compute the paths of the robots based on the order implied by these priorities. Thereby, they incorporate the paths of the robots with higher priority into the configuration time-space of the robot under consideration. The method presented by Warren [1990] uses a fixed order and applies potential field techniques in the configuration time-space to avoid collisions. The approach proposed by Ferrari *et al.* [1998] also uses a fixed priority scheme and chooses random detours for the robots with lower priority. A further approach to decoupled planning is the path coordination method which was first introduced by O'Donnell and Lozano-Pérez [1989]. This approach is based on scheduling techniques for limited resources [Yannakakis *et al.*, 1979]. The key idea of this technique is to keep the robots on their individual paths and let the robots stop, move forward, or even move backward *on their trajectories* in order to avoid collisions (see also [Kant and Zucker, 1986, Bien and Lee, 1992, Chang *et al.*, 1994, Lee *et al.*, 1995a]).

Unfortunately, the problem of finding the optimal schedule for the path coordination method is NP-hard. To see this, we notice that the NP-hard Job-Shop Scheduling problem with the goal to minimize maximum completion time with unit processing time for each job [Martin and Shmoys, 1996, Lawler *et al.*, 1989] can be regarded as a special instance of the path coordination method. To reduce the complexity in the case of large teams of robots Leroy *et al.* [1999] presented a technique to separate the overall coordination problem into sub-problems. This approach, however, assumes that the overall problem can be divided into very small sub-problems, a serious assumption which, as various examples described below demonstrate, is often not justified. In general, therefore, a prioritized vari-

ant has to be applied.

Sánchez and Latombe [2002] presented experiments using a probabilistic roadmap planner to compare centralized and decoupled planning methods. They compared the performance in finding solutions to path planning problems using three different techniques: centralized planning, path coordination with global coordination and path coordination with pairwise coordination using some random priority scheme. In their experiments it came out that the implemented decoupled planning techniques are substantially incomplete for applications "which require tight robot coordination".

As explained in the introduction of this chapter for the prioritized decoupled approaches the order in which the robots' paths are recomputed has a serious influence on whether a solution can be found at all and on how long the resulting paths are. The prioritized methods described above leave open how to assign the priorities to the individual robots. In the past, different techniques for selecting priorities were used. Buckley [1989] proposed to apply a heuristics which assigns higher priority to robots which can move on a straight line from the starting point to their target location. In the work presented by Azarm and Schmidt [1996] all possible assignments are considered. Due to its exponential complexity this approach has only been applied to groups of up to three robots.

Since no single priority scheme will be sufficient to solve all multi-robot path planning problems and since the complexity of the search problems is too high to try all possible orders one has to find sophisticated strategies to explore the search space in order to end up with a solution. Therefore, we present an approach to optimize priority schemes for arbitrary decoupled path planning methods. Our approach performs a randomized hill-climbing search in the space of priority schemes. We thereby interleave the search for an optimal priority scheme with the planning of the paths of the robots. To guide the search, our algorithm utilizes constraints between the robots that are derived from the task description. As a result, our approach seriously reduces the time needed to find a solution (if one exists) for a given path planning problem. Once a solution has been found, our algorithm is able to optimize the priority scheme in order to minimize the overall move cost.

In addition to the literature reviewed so far, there exist techniques that do not fall in the two presented categories. For example, Alami *et al.* [1995] proposed an approach in which the robots merge their plans incrementally. Whenever a robot receives a new goal it tries to coordinate its elaborated plan with the plans of the other robots (see also [Alami *et al.*, 1998a]). Since there is the potential for deadlocks Qutub *et al.* [1997] presented a method to detect and solve them. Their approach assumes that most conflicts can be solved locally which is not guaranteed in general. In the worst case their method switches to a centralized planning system. Gravot and Alami [2001] extended the system in order to be

able to handle different priorities of the involved robots.

Lumelsky and Harinarayan [1997] and Chun *et al.* [1999] use reactive approaches for real-time motion planning. In these approaches each robot plans its path towards its target dynamically based on its current position and sensory feedback. Jäger and Nebel [2001] and Clark *et al.* [2003] also presented more reactive techniques. Whenever a collision between two robots is anticipated Jäger and Nebel try to solve this problem by inserting idle times between certain segments of the pre-computed trajectories of the robots (similar to the path coordination method). If this does not lead to a solution alternative trajectories have to be planned. Clark *et al.* use a combination of centralized and decentralized planning for robots which have only incomplete world models. The authors assume that the robots have a limited communication range and build communication networks whenever they get close enough. A centralized PRM planner is used to compute collision-free trajectories for all robots which are involved in the network. All these approaches, however, tend to fail or to produce sub-optimal solutions especially in narrow environments since there is a high possibility of deadlocks.

Finally, some authors proposed techniques based on heuristics like traffic rules to resolve conflicts [Grossman, 1988, Wang and Premvuti, 1995, Lee *et al.*, 1995b]. These techniques are only applicable if the environment is modeled as a route network and can easily lead to deadlock situations.

2.3 Prioritized A^*-based Path Planning and Path Coordination

In this section we introduce the two prioritized decoupled approaches we apply our search algorithm to.

The basic search algorithm to compute paths for the individual robots that is used throughout this chapter is the popular A^* procedure, which is introduced first. Afterward, we present the prioritized decoupled path planning approach and describe how the A^* procedure can be utilized to plan the motions of a team of robots in a prioritized manner.

2.3.1 A^*-based Path Planning

Our system applies the A^* procedure [Nilsson, 1982] to compute the cost-optimal paths for the individual robots. A^* addresses the problem of finding an optimal path according to a given cost function from an initial state to a goal state in a graph. To search efficiently, the A^* procedure takes into account the accumulated cost of reaching a state n from the initial state $start$ as well as an estimate for

the cost of reaching the goal state $goal$ from n. The estimated cost is also called heuristics. A typical implementation of the A^* algorithm uses a priority queue which contains the already "visited" nodes along with their associated A^* costs. The A^* cost $f(n)$ of a node n are the accumulated costs $cost\_from\_start[n]$ for reaching n from the initial state plus the estimated cost $h(n, goal)$ for reaching the goal state from n. In each iteration the element with the minimum A^* cost is extracted from the priority queue. If necessary its neighbors' costs are updated by taking into account the cost between two neighbor states. These costs are given by the function c. The complete A^* procedure is shown in Algorithm 1.

By using a good heuristics for the cost of reaching the goal state, A^* tends to focus the search in parts of the state space most relevant to the problem of finding a cost-optimal path. This property makes A^* an efficient search algorithm and has given it an enormous popularity in the robotics community. To ensure that the algorithm computes the optimal path the heuristics has to be admissible (see for example [Russell and Norvig, 1995]), which means that it does not overestimate the true cost to reach the goal.

It should be mentioned that A^* requires a discrete search graph, whereas the configuration space of a robot is continuous. Furthermore, each state needs to have a finite number of successor states. In our case we assume that the environment is readily represented by a discrete occupancy grid map – which is common in the mobile robotics literature. Occupancy grids [Moravec and Elfes, 1985] separate the environment into a grid of equally spaced cells and store in each cell $\langle x, y\rangle$ the probability $P_{occ}(\langle x, y\rangle)$ that it is occupied by a static object. An occupancy grid map can be seen as a discrete graph: Each cell of the grid represents a node of the graph. For all neighbor cells with an occupancy value lower than a threshold, an edge between the nodes is inserted. The cost for traversing a cell $\langle x, y\rangle$ is proportional to its occupancy probability $P_{occ}(\langle x, y\rangle)$. To avoid that paths lead through walls etc. we apply a threshold function $\gamma(P_{occ}(\langle x, y\rangle))$ which is infinite if $P_{occ}(\langle x, y\rangle)$ exceeds 0.8, and $P_{occ}(\langle x, y\rangle)$ otherwise.

Furthermore, the estimated cost for reaching the target location $\langle x^*, y^*\rangle$ is approximated by $min_{occ} \cdot ||\langle x, y\rangle - \langle x^*, y^*\rangle||$ where $min_{occ} > 0$ is chosen as the minimum occupancy probability in the map and $||\langle x, y\rangle - \langle x^*, y^*\rangle||$ is the straight-line distance between $\langle x, y\rangle$ and $\langle x^*, y^*\rangle$. Since this heuristics is admissible A^* determines the cost-optimal path from a starting position to the target location.

Figure 2.3 shows a typical space explored by A^*. In this situation the robot starts in the corridor of our environment. Its target location is in the third room to the south. The figure also shows the accumulated costs of the states considered by the planning process. As can be seen A^* only expands a small fraction of the overall state space and therefore is highly efficient.

The disadvantage of the A^* procedure lies in the assumption that all actions are carried out with absolute certainty. To deal with the uncertainty in the robot's

Algorithm 1 Algorithm to compute optimal paths.
$A^*(s, g, succ(*), c(*,*), h(*,*))$

Input: Initial state $start$, goal state $goal$, successor function $succ$, cost function c, heuristics h.

Output: Optimal path from $start$ to $goal$ or `NULL` if no path exists.

```
initializePriorityQueue(PQ);
for all e ∈ searchspace do
   cost_from_start[e] = ∞;
end for
cost_from_start[s] = 0;
predecessor[s] = s;
insert start in PQ with f(start) = 0;
while (PQ not empty) do
   e = extractMin(PQ);
   if (e == g) then
      return optimal path from start to goal given by predecessor[];
   end if
   for all n ∈ succ(e) do
      if (cost_from_start[n] > cost_from_start[e] + c(e, n)) then
         cost_from_start[n] = cost_from_start[e] + c(e, n);
         predecessor[n] = e;
         if (n ∈ PQ) then
            update f(n) = cost_from_start[n] + h(n, goal) in PQ;
         else
            insert n with f(n) = cost_from_start[n] + h(n, goal) in PQ
         end if
      end if
   end for
end while
/* no path from start to goal exists */
return NULL.
```

actions one in principle would have to use the value iteration algorithm for non-deterministic actions (see for example [Sutton and Barto, 1998]) which is less efficient than A^*. To incorporate the uncertainty of the robots motions into the A^* approach we convolve the grid map using a Gaussian kernel. This has a similar effect as generally observed when considering non-deterministic motions: It introduces a penalty for traversing narrow passages or staying close to obstacles. As a result, according to the paths computed by A^* our robots generally prefer trajectories which stay away from obstacles.

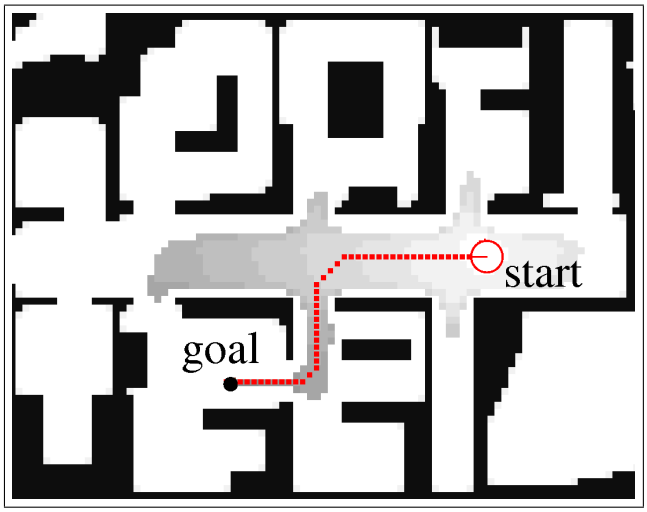


Figure 2.3: Result of a path planing process for a single robot using A^*. The accumulated costs of the cells considered during the search are indicated in grey (the darker the cell the higher the cost).

2.3.2 Prioritized Decoupled Path Planning for Teams of Robots

Recall that in the multi-robot path planning problem, many robots simultaneously seek to traverse an environment. If the robots could move freely regardless of other robots the problem could easily be separated into many local path planning problems. In such a situation each robot could apply A^* to determine its optimal path. However, the impossibility for robots to occupy the same location at the same point in time introduces non-trivial restrictions that have to be incorporated into the paths of the individual robots.

A common prioritized decoupled path planning approach is the following. In a first step, for each robot its optimal path is computed using A^* without considering the paths of the other robots. In the remainder we denote these paths as the independently planned optimal paths. Clearly, these paths might not be acceptable since they would lead to collisions if they were executed. Thus, in a second step, a check for possible conflicts is performed. If conflicts occur, a prioritized planning method tries to avoid them by recomputing the paths of the individual robots, thereby using a priority scheme for the robots. Such a priority scheme Π determines the order in which the robots are processed. $\Pi[0]$ is the robot with the highest priority which is processed first. For each robot it is checked whether its independently planned optimal path has a conflict with one or more of the paths of the robots with higher priority. If so, a path planning method is applied to compute a conflict-free path for the robot under consideration by taking into account

Algorithm 2 Prioritized approach to compute conflict-free paths.
$computeConflictFreePaths(in, \Pi)$

Input: Individually planned paths in of R robots, ordering of the robots (priority scheme) Π.

Output: Conflict-free paths out of the robots or `NULL` if for one robot no conflict-free path can be computed.

for $t = 1$ to $R - 1$ **do**
  if ($existsConflict(in_{\Pi[t]}, \{out_{\Pi[0]}, \ldots, out_{\Pi[t-1]}\})$) **then**
    /* compute a conflict-free path for robot $\Pi[t]$ given the paths of the robots with higher priority */
    $out_{\Pi[t]} = computePath(\{out_{\Pi[0]}, \ldots, out_{\Pi[t-1]}\}, start_{\Pi[t]}, goal_{\Pi[t]})$;
    if ($out_{\Pi[t]} ==$ `NULL`) **then**
      return `NULL`
    end if
  else
    /* the individual optimal path of robot $\Pi[t]$ has no conflicts with the paths of the robots with higher priority */
    $out_{\Pi[t]} = in_{\Pi[t]}$;
  end if
end for
return out.

the paths of the robots with higher priority. The complete prioritized planning method is listed in Algorithm 2. The input are the individually planned paths of the robots and a priority scheme.

2.3.3 Using A^* for Prioritized Path Planning

The A^* algorithm can also be used as the planning method for prioritized path planning. In this case the path of a robot is replanned in its configuration time-space [Erdmann and Lozano-Pérez, 1987]. The configuration time-space of each robot is computed based on the map of the environment and the paths of the robots with higher priority.

In the following we introduce the costs used for planning in the configuration time-space. While planning in the configuration time-space we take into account possible deviations of the individual robots from their planned paths. For this purpose we use a probabilistic model which allows us to derive the probability that a robot will be at location $\langle x, y \rangle$ at time t given it is planned to be at location $\langle x', y' \rangle$ at that time. To estimate the parameters of this model we performed a series of 28 experiments with a robot which was moving with constant velocity.

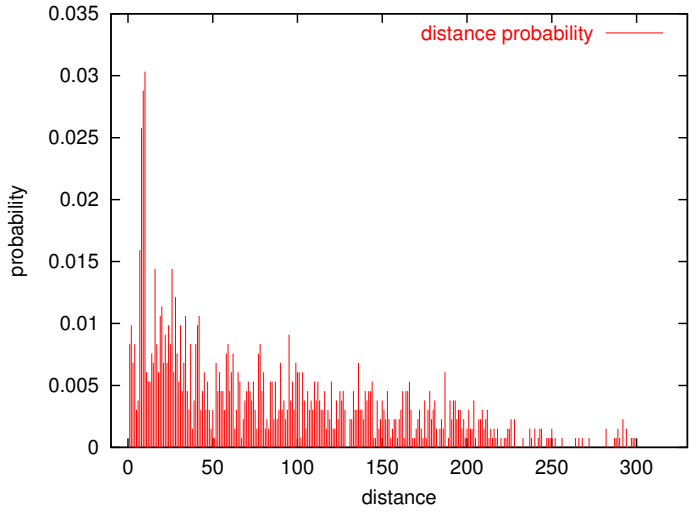


Figure 2.4: Average deviation of a robot from its pre-planned path during plan execution. In a series of experiments we constantly measured the distance of the robot's current position from its planned position at the same point in time.

In each run we constantly measured the distance of the robot from its planned position at the same point in time. As a result we obtained for a discrete set of distances the number of times the robot deviated from its planned path by that distance. The resulting probabilities are depicted in Figure 2.4. In our current implementation this histogram is approximated by a set of linear functions in order to avoid over-fitting. Given these data, we can easily determine the probability $P_i(\langle x, y, t\rangle)$ that the robot $\Pi[i]$ is at a location $\langle x, y\rangle$ at time t. This probability is then used to define a function which allows us to determine the cost $C_k(\langle x, y, t\rangle)$, which is the cost for robot $\Pi[k]$ of traversing cell the $\langle x, y\rangle$ at time t:

$$C_k(\langle x, y, t\rangle) \quad = \quad \gamma(P_{occ}(\langle x, y\rangle)) + \sum_{i=0}^{k-1} P_i(\langle x, y, t\rangle). \tag{2.1}$$

A typical application example of the prioritized decoupled planning technique is illustrated in the left image of Figure 2.5. In this case, the robot depicted in green was supposed to move into the fourth room in the north. The second robot depicted in black had its starting position in the corridor and its target location was close to the starting point of the first robot. When both paths were planned independently, they imposed a conflict between the two robots as can be seen in the left image of the figure. After applying the A^* procedure in the configuration time-space of the black robot (which we assumed to have lower priority), the conflict had been resolved (see right image of Figure 2.5). According to the path

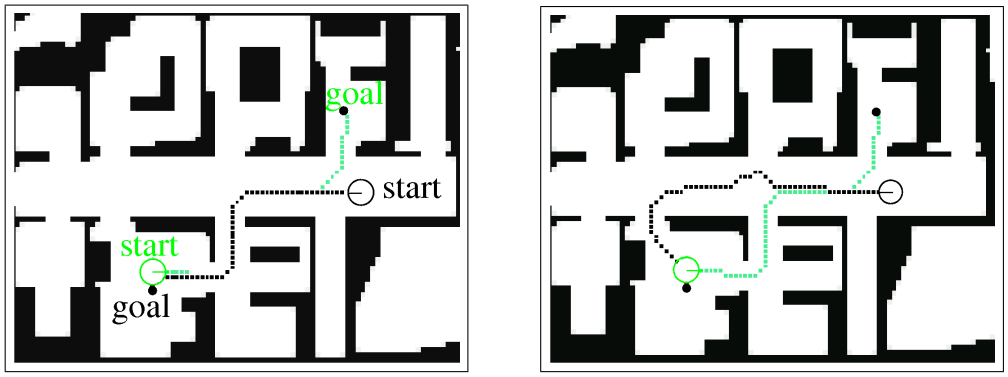


Figure 2.5: Conflict situation for two robots (left image) and resulting conflict-free paths after planning in the configuration-time space of the black robot.

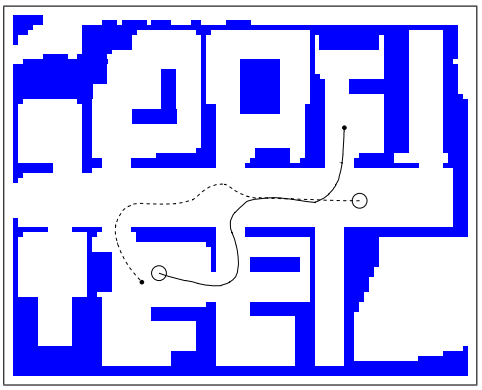


Figure 2.6: Resulting trajectories of two robots carrying out the planned paths shown in the right image of Figure 2.5.

computed by A^*, the black robot had to avoid the conflict with the green robot by moving to the north just at the door where the other robot is supposed to enter the corridor. After this collision avoidance action, the path through the next doorway has less cost. Figure 2.6 shows the trajectories of two robots carrying out the computed plans.

In addition to the general A^*-based planning in the configuration time-space there exists a restricted version of this method which only explores a subset of the configuration in order to reduce the search time. The path coordination technique [O'Donnell and Lozano-Pérez, 1989] restricts the search space to those states which lie on the independently planned optimal paths of the robots. Thus, it forces the robots to stay on their initially computed paths. In our work we use a prioritized variant of this approach. Due to the restriction of the search space,

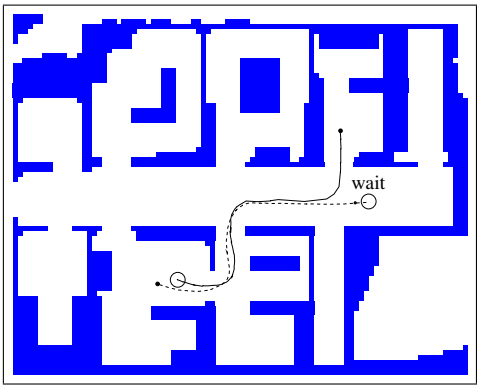


Figure 2.7: Trajectories of two robots obtained by using the path coordination technique to resolve the conflict shown in the right image of Figure 2.5. As can be seen, one robot has to wait until the other passed by and needed twice as long to arrive at its target location compared to the experiment shown in Figure 2.6.

the path coordination method is more efficient than the general A^* search. Its major disadvantage, however, lies in the fact that it fails more often and that it often produces inefficient solutions.

Consider for example the situation depicted in Figure 2.5. In this situation the path coordination technique cannot find a path for the green robot if the black robot has higher priority. Only if the green robot has higher priority the path planning problem can be solved by letting the black robot wait at its initial position until the green robot passed by. Figure 2.7 shows the corresponding paths obtained with the path coordination technique. Please note that in this situation the coordination technique performs significantly worse than general A^*-based planning in the configuration time-space. Since the coordination technique restricts the robots to stay on their pre-planned paths, the robot starting in the corridor has to wait until the other robot passed by. Therefore, the time to arrive at its target location is almost twice as long as it would be without any conflict. In contrast to that, the two robots arrive almost at the same time using unconstrained A^*-based planning in the configuration time-space.

2.4 Finding and Optimizing Solvable Priority Schemes

As already discussed above, the introduction of a priority scheme for the decoupled path planning leads to a serious reduction of the search space. Whereas there are schemes leading to a viable solution with conflict-free paths for a multi-robot

path planning problem, there also exist schemes for which no solution can be found. The examples given in the introduction of this chapter illustrate that the order in which the paths are planned has a profound impact on whether a solution can be found and on how long the resulting paths are. This raises the question of how to find a priority scheme for which the decoupled approach does not fail and for which the move costs of the resulting paths are minimized. Unfortunately, the problem of finding the optimal priority scheme is a non-trivial matter since paths for all $R!$ possible orders of the R robots have to computed. In this section we describe our approach to searching in the space of priority schemes during decoupled path planning. We describe a naive randomized hill-climbing search procedure first, which is subsequently improved by utilizing constraints derived from the task specification.

2.4.1 The Randomized Search Technique

Our algorithm for finding eligible priority schemes for decoupled path planning techniques interleaves the search for conflict-free paths with the search for a solvable priority scheme. Recently, randomized search techniques have been used with great success to solve constraint satisfaction or satisfiability problems [Selman *et al.*, 1992]. Our algorithm presented here is a variant which performs a randomized and hill-climbing search in order to optimize the planning order for prioritized decoupled path planning techniques. It starts with an arbitrary initial priority scheme and tries to compute conflict-free paths for all robots in the order given by the priority scheme. It then randomly exchanges the priorities of two robots in the current scheme and tries to find paths for the robots given the new priority scheme. If the new order results in a solution with paths with lower move costs than the best one found so far, it continues with this new order. Since hill-climbing approaches like this frequently get stuck in local minima, it performs random restarts with different initial orders of the robots. The number of restarts and priority exchanges are controlled by the two parameters `maxTries` and `maxFlips`. The complete method is listed in Algorithm 3. Note that in our implementation, we reuse the paths p if they are conflict-free. After swapping the priorities of two robots, only the paths of the robots with a priority index higher or equal i have to be recomputed. However, we omitted the corresponding statements to enhance readability.

2.4.2 Utilizing Constraints to Focus the Search

Whereas the plain randomized search technique produces good results, it has the major disadvantage that often a lot of iterations are necessary to come up with a solution. For example, we found that for a situation with ten robots in the envi-

Algorithm 3 The naive algorithm to optimize priority schemes.
$naiveOptimization(p_{indep})$

Input: Independently planned optimal paths p_{indep}.
Output: Conflict-free paths p^* with the lowest found move cost or `NULL` if conflict-free paths for all robots cannot be found.

```
p* = NULL;
for tries = 1 to MAX_TRIES do
  select random order Π;
  p = computeConflictFreePaths(p_indep, Π);
  for flips = 1 to MAX_FLIPS do
    choose random i, j with i < j;
    Π' = swap(Π, i, j);
    p' = computeConflictFreePaths(p_indep, Π');
    if (p == NULL or moveCost(p') < moveCost(p)) then
      p = p'; Π = Π';
    end if
  end for
  if (p* == NULL or moveCost(p) < moveCost(p*)) then
    p* = p;
  end if
end for
return p*.
```

ronment shown in Figure 2.8 more than 20 iterations on average were necessary to find a solvable priority scheme. Since each change of a scheme requires the recomputation of the paths for many of the robots, it is of utmost importance to minimize the time required to find priority schemes for which a solution to the path planning problem can be computed. In this section we therefore present a technique to focus the search that tends to reduce the search time significantly.

Our approach can be motivated through the situation depicted in Figure 2.1. In this situation, it is impossible to find a path for robot 1 if the path of robot 3 is planned first, because the goal location of robot 3 lies on the optimal path for robot 1. The key idea of our approach is to introduce a constraint between the priorities of two robots i and j, when the goal position of robot j lies too close to the independently planned optimal path of robot i. Such a constraint would be robot $i <$ robot j, which means that the path of robot i has to be planned before the path of robot j. In our example we thus obtain the constraint robot $1 <$ robot 3 between the robots 1 and 3. Additionally, we get the constraint robot $2 <$ robot 1, since the goal location of robot 1 lies too close to the trajectory of robot 2.

Although the satisfaction of the constraints by a certain priority scheme does not guarantee that valid paths can be found, orders satisfying the constraints more often have a solution than priority schemes violating constraints. Unfortunately, depending on the environment and the number of the robots, it is possible that there is no order satisfying all constraints. In such a case the constraints produce a cyclic dependency. The key idea of our approach is to initially reorder only those robots that are involved in such a cycle in a constraint graph. Thus, we separate all robots into two sets. The first group R_1 contains all robots that, according to the constraints, do not lie on a cycle and have a higher priority than the robot with highest priority which lies on a cycle. This set of robots is ordered arbitrarily with respect to the constraints. This order is initially not changed during the search. The second set, denoted as R_2 contains all other robots. Initially, our algorithm only changes the order of the robots in this second group. After a certain number of iterations, we include all robots in the search for a priority scheme. In our experiments we figured out that this leads to better results with respect to the overall move cost, especially for large numbers of iterations. The complete procedure is listed in Algorithm 4. Note that if it is the case that no conflict-free paths for the robots in R_1 can be computed using the initial order, a new order which satisfies the constraints is chosen for these robots. To enhance readability we omitted the corresponding statements in Algorithm 4.

The advantages of this approach are quite obvious. Initially, the search for a solvable priority scheme is focused. However, through the strong restriction of the search space we loose many solutions which even might be more efficient. Therefore, after a certain number of iterations we do not restrict the search space anymore in order to have a higher chance to find better solutions. The number of iterations carried out changing only the priorities of the robots in R_2 is controlled by a parameter denoted k in the remainder of this work.

Note that this procedure has any-time character [Zilberstein and Russell, 1995], which means that – if a solution can be found – the quality of the solution depends on the available computation time.

To illustrate our approach, consider again the situation with ten robots shown in the left image of Figure 2.8. Whereas the starting locations are marked by $S_0, \ldots, S_9$ the corresponding goal positions are marked by $G_0, \ldots, G_9$. The independently planned optimal trajectories are indicated by solid lines. Given these paths we obtain the constraints depicted in Figure 2.9. The robots 3, 2, 6, 4, 7, and 9 can be sorted according to the constraints and their order initially remains unchanged during the search process (the corresponding nodes are colored red in the figure). Figure 2.10 shows the part of the constraint graph with the robots of R_2. Given the restricted search space R_2 our system quickly finds a solution. In this example, after one iteration we obtained the order 0, 1, 5, and 8 for the robots in R_2. The resulting corresponding conflict-free paths for all robots are

Algorithm 4 The algorithm for finding and optimizing priority schemes. $findAndOptimizePriorityScheme(p_{indep}, R_1, R_2, k)$

Input: Independently optimal paths p_{indep}, set of robots R_1 which can be sorted according to the constraints, set of all other robots R_2, factor k specifying after which iteration the constraints are to be ignored.

Output: Conflict-free paths p^* with the lowest found move cost or `NULL` if conflict-free paths for all robots cannot be found.

$count = 0$
$p^* =$ `NULL`;
for $tries = 1$ to `MAX_TRIES` **do**
  if ($count < k$) **then**
    select Π which satisfies the constraints for R_1 and randomly orders R_2
  else /* extensive search after k iterations */
    select random order Π;
  end if
  $p = computeConflictFreePaths(p_{indep}, \Pi)$;
  for *flips* $= 1$ to `MAX_FLIPS` **do**
    if ($count < k$) **then**
      choose random i, j with $i < j$ and $\Pi[i], \Pi[j] \in R_2$;
    else /* extensive search after k iterations */
      choose random i, j with $i < j$;
    end if
    $\Pi' = swap(\Pi, i, j)$;
    $p' = computeConflictFreePaths(p_{indep}, \Pi')$;
    if ($p ==$ `NULL` **or** $moveCost(p') < moveCost(p)$) **then**
      $p = p'$; $\Pi = \Pi'$;
    end if
    $count = count + 1$;
  end for
  if ($p^* ==$ `NULL` **or** $moveCost(p) < moveCost(p^*)$) **then**
    $p^* = p$;
  end if
end for
return p^*.

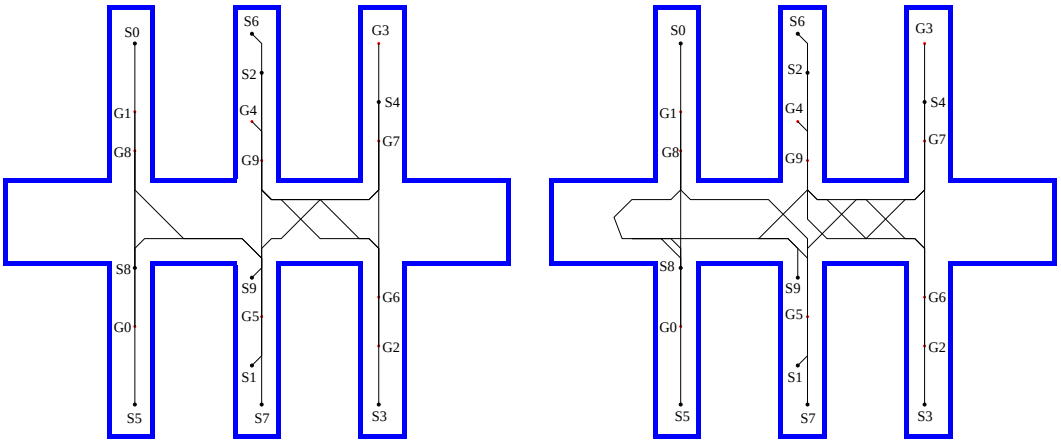


Figure 2.8: Independently planned paths for ten robots (left) and the paths resulting after a solvable priority scheme has been found (right).

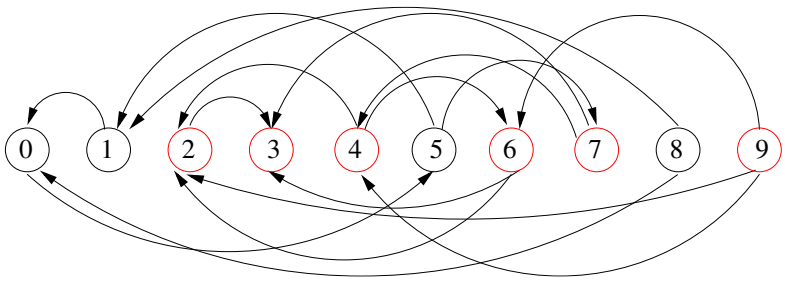


Figure 2.9: Constraint graph generated according to the paths shown in Figure 2.8. Indicated red are those robots which can be sorted according to the constraints.

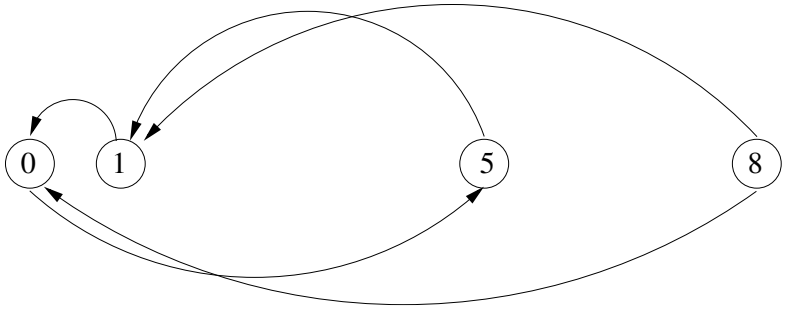


Figure 2.10: Constraint graph for the robots of R_2.

shown in the right image of Figure 2.8. This demonstrates that the constraints can drastically reduce the search space and allow the system to quickly find solvable priority schemes.

2.5 Experimental Results

Our approach has been tested thoroughly on real robots and in extensive simulation runs. The key questions addressed in our experiments were:

Practicability: Is our approach relevant and applicable to real robot systems?

Solvability: Does our approach succeed more frequently in finding valid multi-robot paths than approaches with fixed prioritization?

Optimality: When our approach can find a solution, does it generate more efficient plans?

All experiments were carried out using different environments. To evaluate the general applicability, we applied our method to the two prioritized decoupled path planning techniques described above. In our current implementation we regard it as a conflict whenever the distance between two robots is below $1.2m$ at the same time step. In our implementation it requires less than 0.02 seconds on a 3 GHz Pentium 4 to plan a conflict-free path for one robot in all environments described below. The whole optimization for 10 robots with 10 restarts and 10 iterations per restart requires approximately 12 seconds.

2.5.1 An Example with Real Robots

The goal of the first experiment is to demonstrate the applicability of our approach to real robot systems. This experiment was carried out using the Pioneer I robots of the CS-Freiburg RoboCup [Dietl *et al.*, 2002] team. The task of the robots was to get into their initial formation which has to be done at the beginning of each match. Thereby, the robots have to avoid colliding with other robots that are on the field (shown here as filled circles). In the particular example described here, the robots were deployed on one side of the field and had to move to their home positions on the other side. The left image of Figure 2.11 shows this initial configuration along with the independently planned optimal paths. As can be seen from this figure, these paths cross each other close to the center of the field leading to several several conflicts[1]. Therefore, we applied our randomized search method

[1]Note that in the experiments on this field we regard it as a conflict whenever the distance between two robots is below $0.8m$ at the same time step.

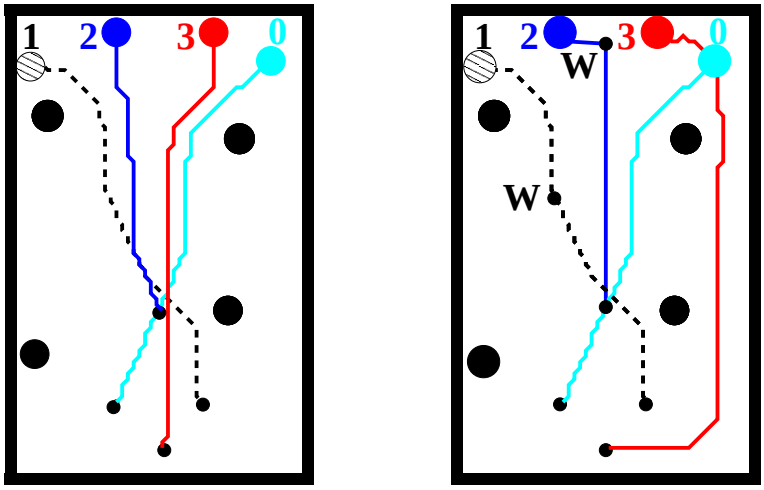


Figure 2.11: An application example with the Robots of the CS-Freiburg RoboCup team. The left image shows the independently planned optimal paths for the four robots and the right image depicts the resulting conflict-free paths computed by our algorithm.

using the general A^* search in the configuration time-space to compute conflict-free paths for all robots. Our algorithm came up with the following order for the robots: 0, 1, 3, 2. The corresponding paths are depicted in the right image of Figure 2.11. As can be seen, the paths of the robots were changed in order to avoid collisions. Whereas robot 1 and robot 2 shortly had to wait in order to let robot 0 pass by (the corresponding positions are marked with a "W" in the right image of Figure 2.11), robot 3 had to take a detour.

Figure 2.12 shows the robots carrying out the navigation plans. The upper left image depicts the initial situation. In the top right image you can see robot 2 making space for robot 1 and robot 3 starting its detour. In the lower left image robot 1 waits to let robot 0 pass by. Finally, the lower right image shows the robots at their final locations. This experiment demonstrates that our approach is applicable to real robot systems.

2.5.2 Simulation Experiments

To elucidate the scaling properties of our approach to larger number of robots, we performed extensive simulation experiments. In particular, we were interested in characterizing the dependence between the performance of our system on various components of our approach. We analyzed the number of planning problems that can be solved, the speed-up obtained by utilizing the constraints, and the reduction

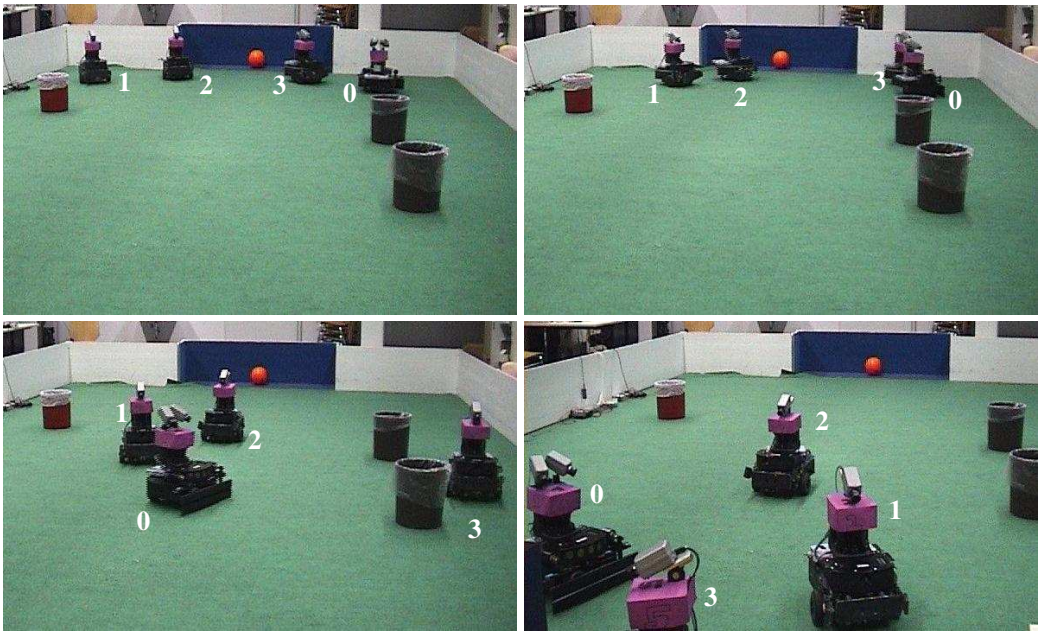


Figure 2.12: The robots carrying out the navigation plans. The top left images depicts the initial situation. In the top right image robot 2 makes space for robot 1 while robot 3 takes a detour. The lower left images shows robot 1 waiting to let robot 0 pass by. The lower right image shows the robots at their final locations.

of the overall move cost. In all experiments, we found that our approach produces highly efficient motion plans even for very large teams of robots, for different environments, and regardless of the specific baseline path planning technique (e.g., general A^* or the path coordination method).

Solved Planning Problems

This first set of experiments was designed to characterize the effect of our search strategy on the overall number of planning problems that can be solved. For each number of robots considered, we performed 100 experiments. In each experiment we randomly chose the starting and target locations of the robots. We applied four different strategies to find solvable priority schemes:

1. A *single* randomly chosen order for the robots.

2. A *single* order which *satisfies* the constraints for the robots in R_1 and consists of a randomly chosen order for the robots in R_2.

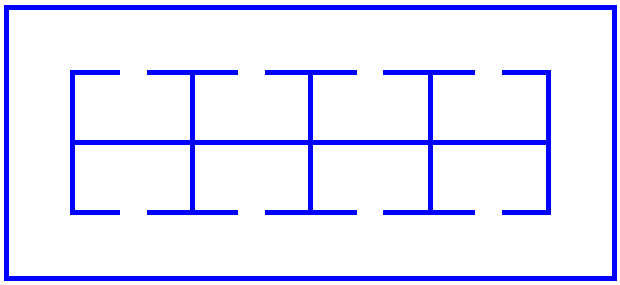


Figure 2.13: Cyclic corridor environment used for the simulation runs.

3. *Unconstrained* randomized search starting with a random order and without considering the constraints.

4. *Constrained* randomized search starting with an order computed in the same way as strategy (2).

All four strategies can be cast as special cases of our algorithm. In the first two strategies the corresponding values for `maxTries` and `maxFlips` are 1. For the first strategy the value of the threshold `k` is 0. The strategies 3 and 4 only differ in the value of the threshold `k`. Whereas the unconstrained search is obtained by setting $\texttt{k} = 0$, the constrained search corresponds to a value of $\texttt{k} = \infty$. Which means that the search space of the unconstrained search comprises all robots from the beginning and the search space of the constrained search comprises only the robots in R_1 all the time.

Note that in these experiments we chose a small number of iterations for the last two strategies in order to assess the advantages of the constrained search under serious time constraints. Particularly, we chose a value of 3 for the parameters `maxFlips` and `maxTries`. Obviously, the larger the number of iterations, the higher is the probability that a solution can be found by an arbitrary randomized search. However, larger numbers of iterations drastically increase the computation time. For each technique, we performed A^*-based planning in the configuration time-space and counted the number of solved planning problems.

Figure 2.14 summarizes the results we obtained for the cyclic corridor environment depicted in Figure 2.13. The horizontal axis represents the number of robots, and the vertical axis depicts the percentage of solved path planning problems. As this result illustrates, our constrained search technique succeeds more often than any of the alternative strategies. It is interesting to note that the second strategy, which utilizes the constraints but considers only one scheme in each experiment, shows a similar performance than the unconstrained randomized search.

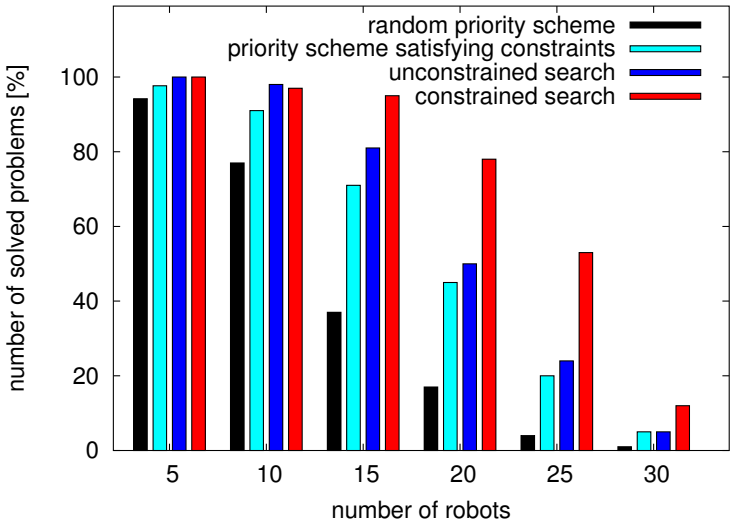


Figure 2.14: Number of solved planning problems for different strategies using A^*-based planning in the configuration time-space in the cyclic corridor environment depicted in Figure 2.13.

To complement these results, we performed a similar series of experiments for the noncyclic corridor environment depicted in Figure 2.8. The results are shown in Figure 2.15. Again, the constrained-based search leads to a much higher success rate.

To investigate the performance using a different baseline path planning algorithm, we applied all four strategies using the path coordination method instead of plain A^*. We used a variant of the environment depicted in Figure 2.8 with five corridors on both sides. Since the path coordination method restricts the robots to stay on their independently planned optimal trajectories, the number of unsolvable problems is much higher compared to the general A^*-based planning in the configuration time-space. As can be seen from Figure 2.16, again the constrained search outperforms all other strategies.

These experiments demonstrate that our approach to finding solvable priority schemes leads to a serious higher number of solved planning problems.

Speed-up Obtained by Utilizing the Constraints

The second set of experiments was performed to investigate the ability of our approach to guide the search in the space of all priority schemes. We were especially interested in the question how much the computation time necessary to find a solution can be reduced by constraining the search. During these experiments we

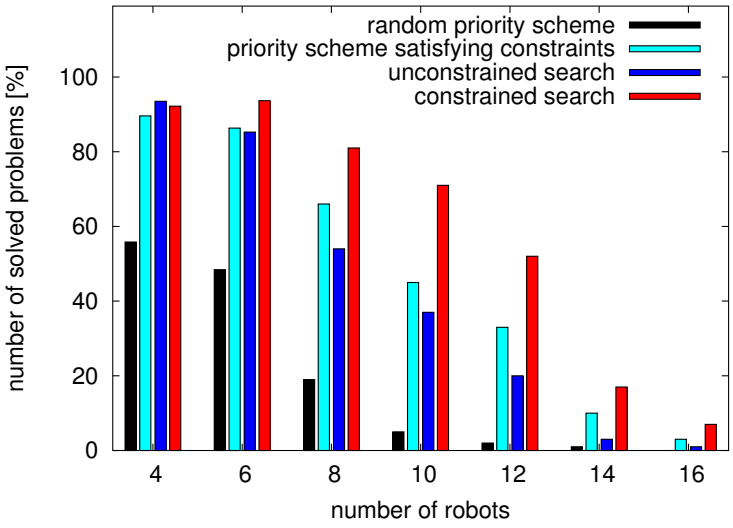


Figure 2.15: Number of solved planning problems for all four strategies using A^*-based planning in the configuration time-space in the noncyclic environment depicted in Figure 2.8.

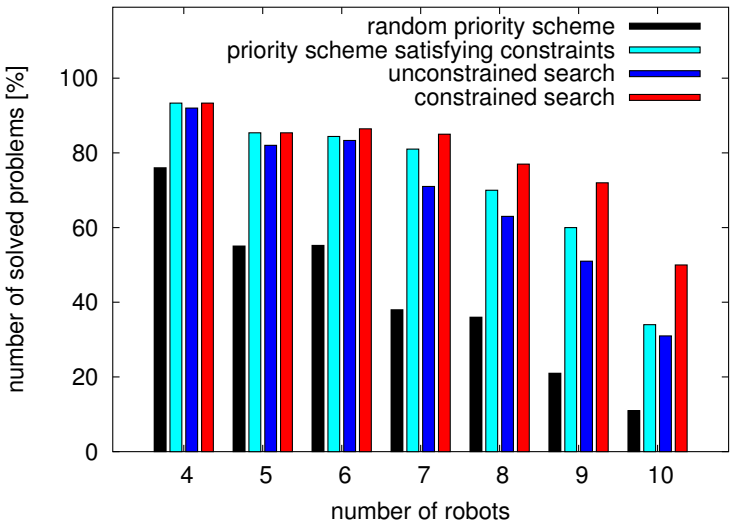


Figure 2.16: Number of solved planning problems for all four strategies using the path coordination method in a variant of the noncyclic environment depicted in Figure 2.8.

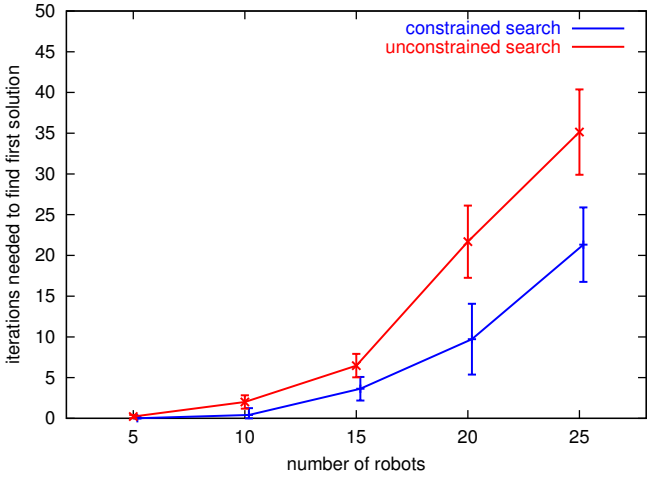


Figure 2.17: This figure plots for the cyclic corridor environment the iteration in which the first solution was found if the planning problem could be solved. Significantly fewer iterations are needed when using the constrained search method.

increased the values of `maxFlips` and `maxTries` to 10 and evaluated in which iteration the first solution was found if the planning problem could be solved. Figure 2.17 plots the results obtained for different number of robots in the cyclic corridor environment and Figure 2.18 shows the corresponding evaluation for the noncyclic environment. We only evaluated planning problems which could be solved by both search methods. As can be seen, the unconstrained search needs significantly more iterations than the constrained search to generate a solution for both environments. Thus, the advantages of our constrained search are two-fold. On one hand, it requires fewer iterations than the unconstrained counter-part. On the other hand, it requires less computation, since the search is restricted to a subset of the robots, which reduces the number of paths that have to be generated in each iteration during the search.

Influence on the Overall Move Cost

The next experiments were carried out to analyze the performance of our algorithm with respect to the overall move cost. Since our algorithm considers in the beginning only a restricted set of priority schemes, and after `k` iterations explores the whole set of priority schemes, we were especially interested in how long the resulting paths are compared to the unconstrained search. We performed 100 experiments in the cyclic corridor-environment and determined the average overall move cost at each iteration. The corresponding graphs are shown in Figure 2.19.

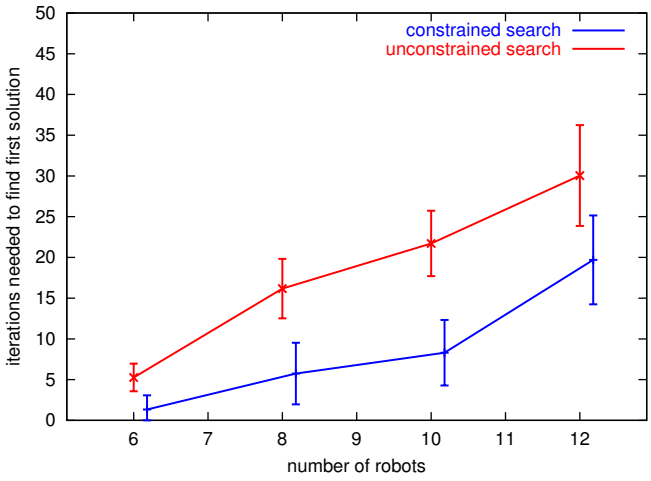


Figure 2.18: In the noncyclic corridor environment we also have a significant difference between the two search strategies considering the iteration in which the first solution was found if the planning problem could be solved.

This plot contains the average move costs for three different strategies at each iteration. The first data set was obtained for the constrained search which corresponds to $\mathtt{k} = \infty$. Using this strategy we reorder only the robots of R_2. The data for the unconstrained search was obtained using $\mathtt{k} = 0$. In this case our algorithm chooses arbitrary priority schemes regardless of the constraints which were derived given the task specification. Finally, the third function labeled "combination of both techniques" corresponds to the results obtained with our algorithm given $\mathtt{k} = 20$.

Since the constrained search focuses the search on the robots that pose the most serious restrictions to the other robots, it finds a solution faster and accordingly has more time to optimize it. Thus, in the beginning, the constrained search outperforms the unconstrained search. After 20 iterations, however, the situation completely changes. Because the unconstrained search can explore many more priority schemes, it more often finds better solutions than the constrained search. Thus, after 20 iterations, the unconstrained search leads to better results than the constrained search. As can be seen from the figure, our approach combines the advantages of both methods. In the beginning, it applies the constraints to focus the search and to quickly find a first solution which is optimized subsequently. After 20 iterations it considers arbitrary priority schemes so that the resulting move costs can be reduced as in the unconstrained search.

Accordingly, our randomized search, which initially uses the constraints to focus the search for a viable solution and afterward uses the unconstrained search

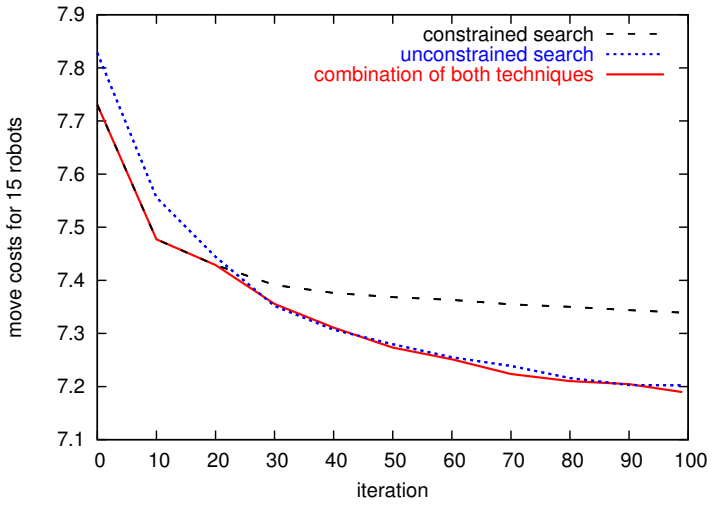


Figure 2.19: Summed move cost plotted over time averaged over 100 planning problems for 15 robots in the cyclic environment.

to optimize this solution, inherits the advantages of both techniques with respect to efficiency and the resulting move costs.

Planning Paths for Large Teams of Robots

The final experiment is designed to illustrate that our system can be used to solve planning problems for even large numbers of robots.

The left image of Figure 2.20 shows the independently planned optimal paths for a team of 30 robots in an unstructured environment. In this particular example our system was able to generate a first solution in less than one second. The paths shown in the right image of this figure are the best solution found after 10 restarts with 10 iterations in each round.

Figure 2.21 plots the evolution of the summed move cost of the best solution found so far over time. As can be seen from the figure, after 100 iterations the overall move cost is reduced by 15%. Figure 2.22 shows images of the robots carrying out in simulation the navigation plans of the best solution which was found.

2.6 Conclusion

In this chapter we presented the main problems of the multi-robot path planning problem and explained the drawbacks of existing approaches. We introduced the

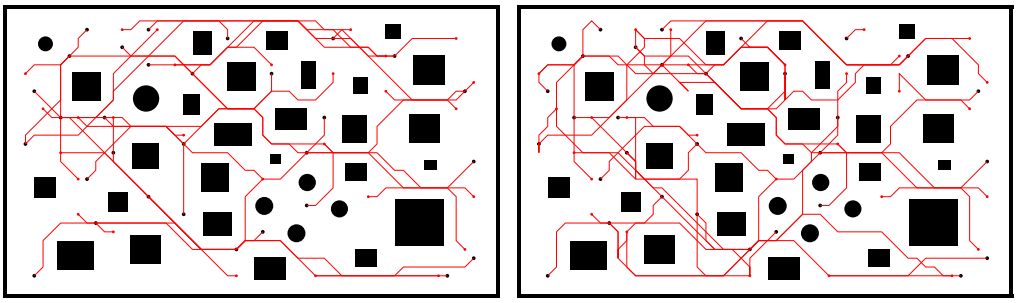


Figure 2.20: Independently planned optimal paths for 30 robots (left image) and the resulting paths after optimizing the priority scheme during 100 iterations (right image).

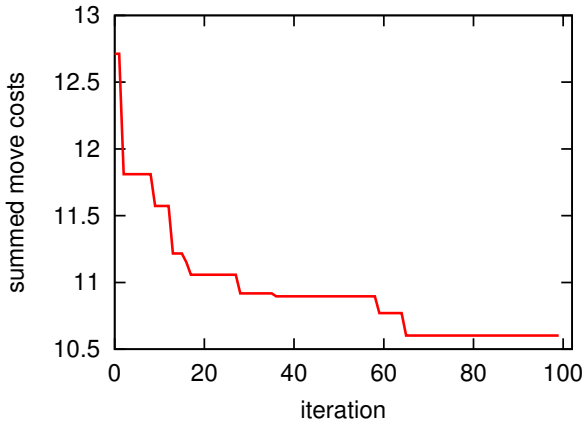


Figure 2.21: Summed move cost plotted over time for the planning problem with 30 robots shown in Figure 2.20.

prioritized decoupled path planning approach which searches in the configuration time-spaces of the robots for conflict-free paths. As pointed out, no single priority scheme for the robots will be sufficient to solve all possible multi-robot motion problems. We therefore proposed a randomized and hill-climbing search technique in the space of priority schemes, which is used to find a solution to a given path planning problem and to minimize the overall path length. To guide the search for a solvable priority scheme, our approach utilized constraints extracted from the task specification. These constraints specify an order in which certain paths have to be planned.

We presented experiments which were designed to evaluate the general applicability of our approach to finding and optimizing solvable priority schemes for

decoupled path planning methods. Our approach has been successfully applied to real robots. These results have been complemented by extensive simulations to characterize the relation between the planning performance and various problem parameters. The experiments suggest that our technique seriously decreases the number of cases in which no solution can be found. Additionally, our approach leads to a reduction of the overall path length. In all experiments, we have found that our approach produces highly efficient motion plans even for very large teams of robots and in different types of environments. One further advantage of our method lies in its general applicability. Although we applied our optimization technique only to two different baseline path-planning techniques here, it is not limited to these two techniques. Rather, it can be used to find and optimize paths generated by arbitrary prioritized path-planning techniques.

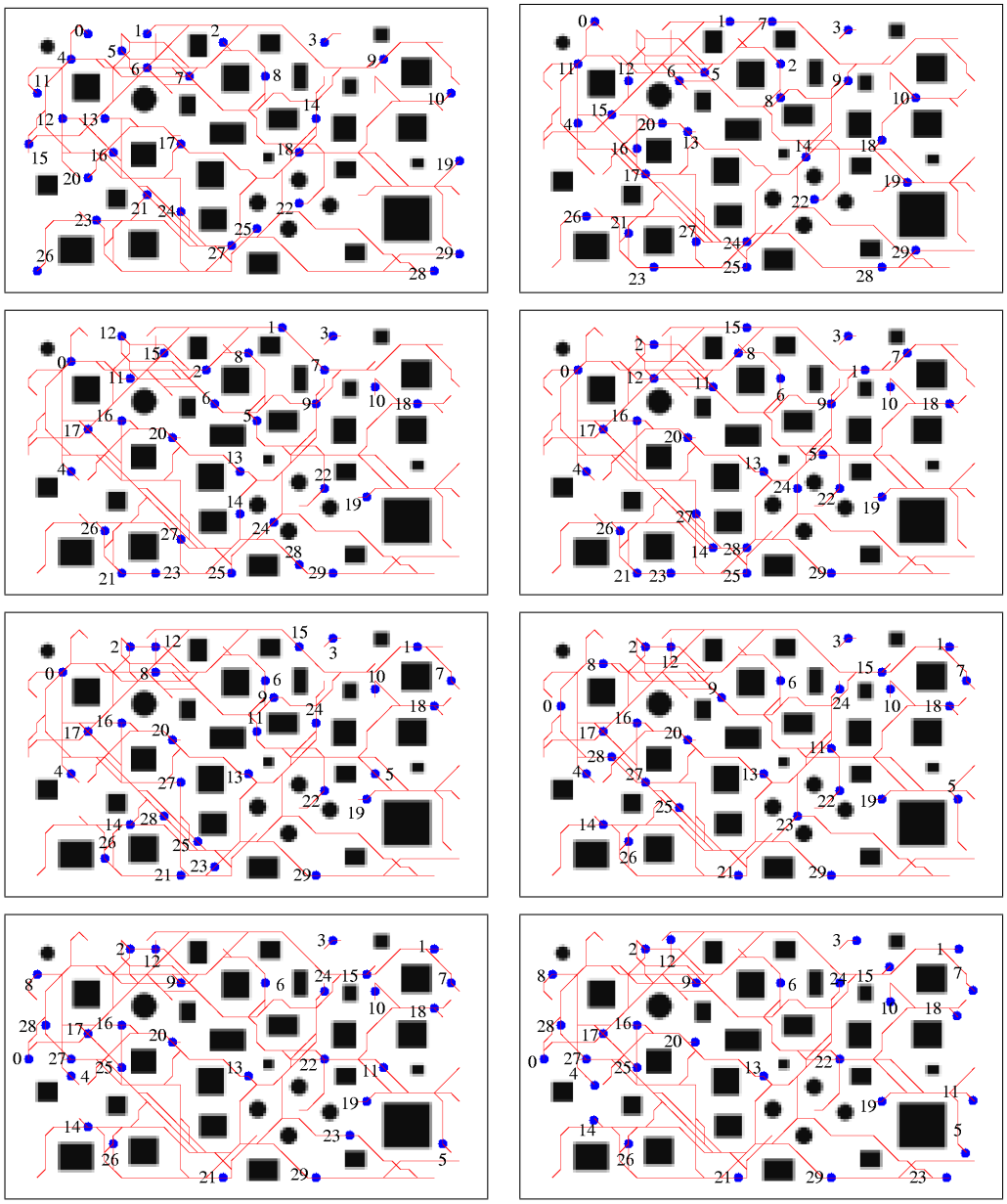


Figure 2.22: This figure shows images of a simulation run in which the robots carry out the paths of the best solution which was found. The upper left image shows the robots at their initial position, the upper right images shows the position of the robots at time step 9, the left image in the second row at time step 17 and so on. In the situation depicted in the image in the lower right all robots have arrived at their goal position.

Chapter 3

Learning Motion Patterns of People

3.1 Introduction

Whenever mobile robots are designed to operate in environments populated by humans they need to be able to perceive the people in their environment and to adapt their behavior according to the activities of the people. The knowledge about typical motion patterns of people can be used in several ways to improve the behavior of a mobile robot. For example, it enables the robot to develop improved people following and obstacle avoidance skills. Furthermore, knowledge about typical motion patterns can be important for robotic security devices [Everett, 1998] to identify potential intruders based on the deviations in the movements and for intelligent homes. Those have the objectives to anticipate the inhabitants needs, like for example lighting conditions and energy conservation [Mozer, 1998].

In this chapter we present an approach that allows to learn motion patterns of people, while they are carrying out their every-day activities. Our approach to learning motion patterns of people is purely probabilistic. It is motivated by the observation that people typically do not move randomly when they walk through their environments. Instead, they usually engage in motion patterns, related to typical activities or specific locations they are interested in approaching. The input to our algorithm is a set of trajectories of people between so-called resting places where the people often stop and stay for a certain period of time. Such places can be desks in office environments or the TV set in at home. Our approach clusters these trajectories into so-called motion patterns using the EM algorithm (see for example [McLachlan and Krishnan, 1997]). Our method is an extension of the k-Means algorithm to the multi-step case that independently applies k-Means (see e.g. [Duda *et al.*, 2001]) to each step in a normalized trajectory.

In the following section we introduce our probabilistic representation of the motion patterns and describe how to learn them using EM. Section 3.3 describes

our approach to compute the likelihoods that an observed trajectory belongs to the individual motion patterns. In Section 3.4 we provide implementation details of the data acquisition procedure. Section 3.5 presents experimental results regarding the learning process as well as regarding the prediction accuracy of the learned models. Furthermore, this section contains a discussion about properties of our algorithm. Before presenting related work in Section 3.7 we describe a different approach to represent the motion patterns in Section 3.6 and give reasons why this approach is not applicable in the context of clustering trajectories.

3.2 Using EM to Learn Motion Patterns

When people perform everyday activities in their environments they do not move permanently. They usually stop at several locations denoted as resting places and stay there for a certain period of time, depending on what activity they are currently carrying out. The task of the algorithm presented in this section is to cluster similar trajectories between resting places into single motion patterns in a completely unsupervised manner. Accordingly, we assume that the input to our algorithm is a collection of observed trajectories $s = \{s_1, \ldots, s_I\}$ (called: the data) between resting places. The output is a number of different types of motion patterns $\theta = \{\theta_1, \ldots, \theta_M\}$ a person might exhibit in its natural environment. Each trajectory s_i consists of a sequence $s_i = \{s_i^1, s_i^2, \ldots, s_i^{T_i}\}$ of positions s_i^t, where s_i^1 is the resting place the person leaves and $s_i^{T_i}$ is the destination.

Our goal is to find a motion model θ (i.e., a set of motion patterns) that maximizes the likelihood of the data. The likelihood of a data set s under a model θ is the probability $P(s \mid \theta)$ of obtaining s given θ. Thus, the model θ with the maximum likelihood is given by:

$$\theta = \operatorname*{argmax}_{\theta'} P(s \mid \theta'). \tag{3.1}$$

To define the likelihood of the data under the model θ, it will be useful to introduce a set of *correspondence variables*, denoted c. Each correspondence c_{im} is a binary variable, i.e., it is either 0 or 1. Here i is the index of the trajectory s_i and m is the index of the motion pattern θ_m. c_{im} is 1 if and only if s_i belongs to θ_m. If we think of the motion pattern as a specific motion activity a person might be engaged in, c_{im} is 1 if the person was engaged in motion activity m when following trajectory i.

In the sequel, we denote the set of all correspondence variables for the i-th data item by c_i, i.e., $c_i = \{c_{i1}, \ldots, c_{iM}\}$. For any data item s_i, the fact that exactly

one correspondence is 1 translates into the following:

$$\sum_{m=1}^{M} c_{im} \quad = \quad 1. \tag{3.2}$$

The goal is to find the set of motion patterns which has the highest data likelihood. Finding the model that maximizes the likelihood of the observed data s is equivalent to finding the model which maximizes the joint likelihood of s and the correspondence variables c. Given a model θ we can compute the joint likelihood of s and c as the product of the likelihoods of the individual data items and their correspondence variables:

$$P(s, c \mid \theta) = \prod_{i=1}^{I} P(s_i, c_i \mid \theta). \tag{3.3}$$

Since the logarithm is a monotonic function we can maximize the log likelihood instead of the likelihood. The logarithm of Eq. (3.3) is given by:

$$\ln P(s, c \mid \theta) = \sum_{i=1}^{I} \ln P(s_i, c_i \mid \theta). \tag{3.4}$$

Since the values of the correspondence variables c are hidden (i.e., not known) we have to integrate over them and optimize the *expected* log likelihood, denoted $E_c[\ln P(s, c \mid \theta) \mid \theta, s]$, instead which is defined as:

$$E_c[\ln P(s, c \mid \theta) \mid \theta, s] = E_c[\sum_{i=1}^{I} \ln P(s_i, c_i \mid \theta) \mid \theta, s]. \tag{3.5}$$

Optimizing this term is usually not an easy endeavor since it is a non-linear optimization. The EM algorithm, which is introduced in the following, iteratively maximizes expected log likelihood functions by optimizing a sequence of lower bounds. In particular, it generates a sequence of models, denoted $\theta^{[1]}, \theta^{[2]}, \ldots$ of increasing log likelihood.

3.2.1 The EM Algorithm

The EM algorithm is often used in statistics to compute maximum-likelihood estimates in the case of hidden data [Dempster *et al.*, 1977, McLachlan and Krishnan, 1997].

A common method to find the model which has the maximum expected data likelihood is to use a function $Q(\theta' \mid \theta)$ which depends on the current estimated

model θ and yields the expected data log likelihood as a function of θ' given the observed data s:

$$Q(\theta' \mid \theta) \;=\; E_c[\ln P(s, c \mid \theta') \mid \theta, s]. \tag{3.6}$$

The sequence of models generated in the iterations of the EM algorithm is then given by calculating

$$\theta^{[j+1]} \;=\; \operatorname*{argmax}_{\theta'} Q(\theta' \mid \theta^{[j]}), \tag{3.7}$$

starting with some initial model $\theta^{[0]}$. In particular, the EM algorithm repeats the following two steps until convergence:

Estimation (E) step: Compute expected values for the hidden variables c given the current model $\theta^{[j]}$. Define the expected data log likelihood as a function of θ using these values:

$$Q(\theta \mid \theta^{[j]}) = E_c[\ln P(s, c \mid \theta) \mid \theta^{[j]}, s]. \tag{3.8}$$

Maximization (M) step: Maximize this expected likelihood. Replace $\theta^{[j]}$ by the model θ' that maximizes the Q function:

$$\theta^{[j+1]} \;=\; \operatorname*{argmax}_{\theta'} Q(\theta' \mid \theta^{[j]}). \tag{3.9}$$

In each iteration the M-step chooses a new model $\theta^{[j+1]}$ that monotonically increases the Q function and thus the EM algorithm increases monotonically the data likelihood ([Dempster *et al.*, 1977], see Appendix A.3 for the proof). Whenever the Q function is continuous this approach is guaranteed to converge to a stationary point, typically to a local maximum (see [Wu, 1983] for convergence properties). However, it is not guaranteed to converge to a global maximum. The choice of the initial estimate $\theta^{[0]}$ can have a serious influence on the final result of the EM algorithm.

3.2.2 Representing Motion Patterns by Gaussian Distributions

In our implementation a motion pattern, denoted θ_m with $1 \leq m \leq M$ where M is the number of different motion patterns the person might be engaged in, is represented by K probability distributions $P(x \mid \theta_m^k)$. In particular, we use Gaussian distributions with a fixed standard deviation σ. Accordingly, the application of EM leads to an extension of the k-Means algorithm [Forgy, 1965, MacQueen, 1967, Mitchell, 1997, Duda *et al.*, 2001] to trajectories.

In order to be able to apply the k-Means algorithm the input to our algorithm must consist of trajectories which have the same number of observed positions, i.e., $T_i = T$ for all i. To achieve this, we transform the trajectories of the input set s into a set d of I trajectories such that each $d_i = \{x_i^1, x_i^2, \ldots, x_i^T\}$ has a fixed length T and is obtained from s_i by a linear interpolation (which means that we insert new data points into the sequences). The length T of these trajectories corresponds to the maximum length of the input trajectories in s. The learning algorithm described below operates solely on $d_1, \ldots, d_I$ and does not take into account the velocities of the people during the learning phase. In our experiments we never found evidence that the linear interpolation led to wrong results or that the walking speed of a person depends on the typical activity it is carrying out. Note, however, that one can extend our algorithm to also incorporate the velocities of the people. This can be achieved by introducing further dimensions to the state variables.

For each θ_m^k the probability distribution $P(x \mid \theta_m^k)$ is computed based on $\beta = \lceil T/K \rceil$ subsequent positions on the trajectories. Accordingly, $P(x \mid \theta_m^k)$ specifies the probability that the person is at location x after $t \in [(k-1) \cdot \beta + 1; k \cdot \beta]$ observations given that it is engaged in motion pattern m. Thus, we calculate the likelihood of a trajectory under the m-th motion pattern θ_m as:

$$P(d_i \mid \theta_m) \;=\; \prod_{t=1}^{T} P(x_i^t \mid \theta_m^{\lceil t/\beta \rceil}). \tag{3.10}$$

Note that we assume consecutive positions on the trajectories to be independent. This is generally not justified, however, in our experiments we never found evidence that this led to wrong results.

Expectation Maximization

As mentioned above we assume that each motion pattern θ_m is represented by K Gaussian distributions with means μ_m^k and a fixed standard deviation σ. Given the individual Gaussians for a model θ we can compute the joint likelihood of a single trajectory d_i and its correspondence vector c_i as follows:

$$P(d_i, c_i \mid \theta) \;=\; \prod_{t=1}^{T} \frac{1}{\sqrt{2\pi}\sigma} \prod_{m=1}^{M} e^{-\frac{1}{2\sigma^2} c_{im} \| x_i^t - \mu_m^{\lceil t/\beta \rceil} \|^2}. \tag{3.11}$$

Here we make use of the fact that only one of the correspondence variables c_{im} in the inner product is 1 and all others are 0.

Accordingly, the expected data log likelihood which has to be maximized is defined as:

$$
\begin{aligned}
& E_c[\ln P(d, c \mid \theta) \mid \theta, d] \\
&= E_c\left[\sum_{i=1}^{I} \ln \prod_{t=1}^{T} \frac{1}{\sqrt{2\pi}\sigma} \prod_{m=1}^{M} e^{-\frac{1}{2\sigma^2} c_{im} \|x_i^t - \mu_m^{\lceil t/\beta \rceil}\|^2} \;\middle|\; \theta, d \right] && (3.12) \\
&= E_c\Bigg[I \cdot T \cdot \ln \frac{1}{\sqrt{2\pi}\sigma} \\
&\qquad -\frac{1}{2\sigma^2} \sum_{i=1}^{I} \sum_{t=1}^{T} \sum_{m=1}^{M} c_{im} \|x_i^t - \mu_m^{\lceil t/\beta \rceil}\|^2 \;\Big|\; \theta, d \Bigg]. && (3.13)
\end{aligned}
$$

Since the expectation is a linear operator we can move the expectations inside the expression so that we finally get:

$$
\begin{aligned}
& E_c[\ln P(d, c \mid \theta) \mid \theta, d] \\
&= const - \frac{1}{2\sigma^2} \sum_{i=1}^{I} \sum_{m=1}^{M} E[c_{im} \mid \theta, d] \sum_{t=1}^{T} \|x_i^t - \mu_m^{\lceil t/\beta \rceil}\|^2. && (3.14)
\end{aligned}
$$

Application of the EM Algorithm

In accordance with Eq. (3.14), the Q-function is factored as follows:

$$
Q(\theta' \mid \theta) = const - \frac{1}{2\sigma^2} \sum_{i=1}^{I} \sum_{m=1}^{M} E[c_{im} \mid \theta, d] \sum_{t=1}^{T} \|x_i^t - \mu_m'^{\lceil t/\beta \rceil}\|^2. \quad (3.15)
$$

During the iterations of the EM algorithm a sequence of models $\theta^{[j]}$ is generated starting with some initial model $\theta^{[0]}$ as explained above. The optimization involves two steps: Calculating the expectations $E[c_{im} \mid \theta^{[j]}, d]$ given the current model $\theta^{[j]}$ (E-step) and finding the new model $\theta^{[j+1]}$ that has the maximum expected likelihood under these expectations (M-step).

The expected values for the correspondence variables are easily calculated via Bayes' Rule (see Appendix A.2.1), by obeying obvious independence assumptions between different data trajectories and assuming that the c_{im} are a priori equally likely:

$$
\begin{aligned}
E[c_{im} \mid \theta^{[j]}, d] &= P(c_{im} \mid \theta^{[j]}, d) && (3.16) \\
&= P(c_{im} \mid \theta^{[j]}, d_i) && (3.17) \\
&= \eta P(d_i \mid c_{im}, \theta^{[j]}) P(c_{im} \mid \theta^{[j]}) && (3.18) \\
&= \eta' P(d_i \mid \theta_m^{[j]}). && (3.19)
\end{aligned}
$$

Thus, $E[c_{im} \mid \theta^{[j]}, d]$ is the probability that trajectory d_i belongs to the m-th component given the current model $\theta^{[j]}$. The normalization constants η and η' ensure that the expectations sum up to 1 over all m.

If we combine Eq. (3.10) and Eq. (3.19) making use of the fact that the distributions are represented by Gaussians we obtain:

$$E[c_{im} \mid \theta^{[j]}, d] = \eta' \prod_{t=1}^{T} e^{-\frac{1}{2\sigma^2}\|x_i^t - \mu_m^{\lceil t/\beta \rceil [j]}\|^2}. \tag{3.20}$$

The M-step calculates a new model $\theta^{[j+1]}$ by maximizing the expected likelihood. This is done by computing for each model component θ_m and for each probability distribution $P(x \mid \theta_m^{k[j+1]})$ a new mean $\mu_m^{k[j+1]}$ of the Gaussian distribution. The new model $\theta^{[j+1]}$ is computed as follows:

$$\begin{aligned}
&\theta^{[j+1]} \\
&= \operatorname*{argmax}_{\theta'} Q(\theta' \mid \theta^{[j]}) && (3.21)\\
&= \operatorname*{argmax}_{\theta'} \left\{ const - \frac{1}{2\sigma^2} \sum_{i=1}^{I} \sum_{m=1}^{M} E[c_{im} \mid \theta, d] \sum_{t=1}^{T} \|x_i^t - \mu_m'^{\lceil t/\beta \rceil}\|^2 \right\} && (3.22)\\
&= \operatorname*{argmin}_{\theta'} \left\{ \sum_{i=1}^{I} \sum_{m=1}^{M} E[c_{im} \mid \theta^{[j]}, d] \sum_{t=1}^{T} \|x_i^t - \mu_m'^{\lceil t/\beta \rceil}\|^2 \right\}. && (3.23)
\end{aligned}$$

To compute the model $\theta^{[j+1]}$ which minimizes this expression we compute the first derivation and set it to zero. Each of the K means of the probability distributions which represent the motion patterns is computed based on β subsequent positions on the trajectories. Thus, for each component of the mean $\mu_m^{k[j+1]}$ it must hold that (to enhance readability we do not use a further index for the component here):

$$\begin{aligned}
\sum_{i=1}^{I} E[c_{im} \mid \theta^{[j]}, d] \cdot 2 \cdot \sum_{t=(k-1)\cdot\beta+1}^{k\cdot\beta} (x_i^t - \mu_m^{k[j+1]}) \;&\stackrel{!}{=}\; 0 \qquad \Longleftrightarrow \qquad (3.24)\\
\sum_{i=1}^{I} E[c_{im} \mid \theta^{[j]}, d] \sum_{t=(k-1)\cdot\beta+1}^{k\cdot\beta} x_i^t \;&\stackrel{!}{=}\; \sum_{i=1}^{I} E[c_{im} \mid \theta^{[j]}, d] \cdot \beta \cdot \mu_m^{k[j+1]}.
\end{aligned}$$

Thus, we get for $\mu_m^{k[j+1]}$ a weighted sum of the β subsequent positions on the trajectories:

$$\mu_m^{k[j+1]} \;=\; \frac{1}{\beta} \cdot \frac{\sum_{i=1}^{I} E[c_{im} \mid \theta^{[j]}, d] \sum_{t=(k-1)\cdot\beta+1}^{k\cdot\beta} x_i^t}{\sum_{i=1}^{I} E[c_{im} \mid \theta^{[j]}, d]}. \tag{3.25}$$

3.2.3 Monitoring Convergence and Local Maxima

The EM algorithm is well-known to be sensitive to local maxima during the search. In the context of clustering local maxima correspond to situations in which data items are associated to wrong model components or clusters. In our application this involves situations in which trajectories of different motion patterns are with high probability associated to the same model component θ_m. In such cases, θ cannot correctly represent a model of the people's motions. Luckily, such cases can be identified quite reliably during EM.

Let us first assume that the correct number of motion patterns is given. Our approach continuously monitors two types of occurrences to detect local maxima:

Low data likelihood: If a trajectory d_i has low likelihood under the model θ, this is an indication that no appropriate model component for d_i has yet been identified that explains this trajectory.

Low model component utility: The aim of this criterion is to discover multiple model components that basically represent the same motion pattern. To detect such cases, the expected data log likelihood is calculated with and without a specific model component θ_m. If the difference in this likelihood is smaller than a pre-specified threshold, the effect of removing θ_m from the model is negligible.

Whenever the EM algorithm has converged, our approach extracts those two statistics and considers "resetting" individual model components. In particular, if a low data likelihood trajectory is found, a new model component is introduced that is initialized using this very trajectory (this is an adaption of the partition expansion presented by Li *et al.* [2001]). At the same time the model component which has the lowest utility is eliminated from the model. If no model component exists with a utility lower than a predefined threshold our algorithm terminates and returns the current set of model components.

In our experiments we found this selective resetting and elimination strategy extremely effective in escaping local maxima. Without this mechanism, the EM algorithm frequently got stuck in local maxima and generated models that were significantly less predictive of human motion.

3.2.4 Estimating the Number of Model Components

The approach presented above works well in the case that the actual number of different motion patterns is known. In general, however, the correct number of motion patterns is not known in advance. Thus, we need to determine this quantity during the learning phase. If the number of model components is wrong, we

can distinguish two different situations. First, if there are too few model components, there must be trajectories that are not explained well by any of the current model components. On the other hand, if there are too many model components there must be trajectories that are explained well by different model components. Whenever the EM algorithm has converged, our algorithm checks whether the model can be improved by increasing or decreasing the number of model components. During the search, we continuously monitor the two types of occurrences mentioned above: low data likelihood and low model component utility. If a low data likelihood trajectory is found, a new model component is introduced which is initialized using this very trajectory. Conversely, if a model component with low utility is found, it is eliminated from the model.

To limit the model complexity and to avoid overfitting we use the Bayesian Information Criterion (BIC) [Schwarz, 1978] to evaluate a model $\theta^{[j]}$:

$$E_c[\log P(d, c \mid \theta^{[j]}) \mid \theta^{[j]}, d] - \frac{M^{[j]}}{2} \log I. \tag{3.26}$$

BIC is a popular approach to score a model during clustering (see also [Fraley and Raftery, 1998]). It trades off the number of model components $M^{[j]}$ multiplied by the logarithm of the number of input trajectories with the quality of the model with respect to the given data[1].

Our algorithm terminates and returns the model with the best overall evaluation found so far after the maximum number of iterations has been reached or when the overall evaluation cannot be improved by increasing or decreasing the number of model components.

3.3 Computing the Likelihood of Motion Patterns

To be able to use the learned motion patterns to classify trajectories and to predict future movements, we have to compute the probability that an observed sequence belongs to the individual motion patterns. Suppose the robot observes a sequence $z = \{z^1, z^2, \ldots, z^T\}$ of positions of a person. What we are interested in is a distribution which gives us for each motion pattern θ_m the probability $P(\theta_m \mid z)$ that the person is engaged in θ_m given z. According to Bayes' Rule, this corresponds to

$$P(\theta_m \mid z) = \alpha P(z \mid \theta_m) P(\theta_m). \tag{3.27}$$

[1] Note that BIC bears a resemblance to the Minimum Description Length Principle (MDL) from the information theory [Rissanen, 1984]. MDL says that the best model is the one that minimizes the number of bits needed to describe the model plus the number of bits needed to describe the data given the model.

Here, $P(z \mid \theta_m)$ is the likelihood of the data given θ_m, $P(\theta_m)$ is the prior for θ_m, and α is a normalizer ensuring that the left-hand side sums up to one over all θ_m.

It remains to describe how $P(z \mid \theta_m)$ is computed. Unfortunately, z does not necessarily start at the initial position of the corresponding motion pattern. Suppose θ_m^k with $1 \leq k \leq K$ is the position in θ_m the first observed position z^1 corresponds to. Furthermore, suppose $\theta_m^{k'}$ with $k \leq k' \leq K$ is the position of θ_m the final observation $z^{\mathcal{T}}$ of z corresponds to. Since both k and k' are unknown, we apply the law of total probability (see Appendix A.2.4) and compute $P(z \mid \theta_m)$ by summing over all possible combinations of k and k':

$$P(z \mid \theta_m) = \sum_{k=1}^{K} \sum_{k'=k}^{K} P(z \mid \theta_m, k, k') P(k, k' \mid \theta_m). \tag{3.28}$$

The prior probability $P(k, k' \mid \theta_m)$ depends on the difference between the length of the given segment on the motion pattern θ_m and on the distance $\|z^1 - z^{\mathcal{T}}\|$. To get $P(z \mid \theta_m, k, k')$ we compute the product of the likelihoods of each observation z^τ in z given that it starts at k and ends at k':

$$P(z \mid \theta_m, k, k') = \prod_{\tau=1}^{\mathcal{T}} P(z^\tau \mid \theta_m, k, k') \tag{3.29}$$

$$= \prod_{\tau=1}^{\mathcal{T}} P(z^\tau \mid \theta_m^{\lceil f(\tau,k,k') \rceil}) \tag{3.30}$$

where

$$f(\tau, k, k') = \frac{k' - k}{\mathcal{T} - 1}\tau + \frac{k\mathcal{T} - k'}{\mathcal{T} - 1} \tag{3.31}$$

realizes a linear mapping of the individual observations $z^1, \ldots, z^{\mathcal{T}}$ to the components $\theta_m^k, \ldots, \theta_m^{k'}$ of θ_m.

3.4 Laser-based Data Acquisition

The EM-based learning procedure has been implemented for data acquired with laser range sensors. To acquire the data we used several laser range scanners which were installed in the environment such that all relevant parts of the environment were covered. The laser scanners were mounted on a height of approximately $30cm$. During the data acquisition we assume that only one person is moving through the environment. But it should be noted that if more sensors to distinguish people (e.g. a camera system) were available one could easily learn

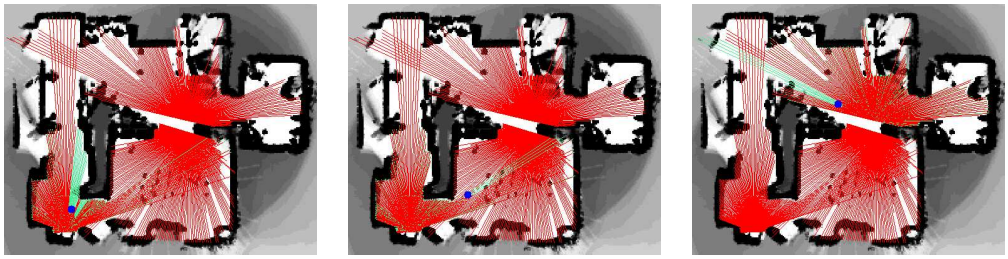


Figure 3.1: Typical laser range data obtained in a home environment equipped with three laser-range scanners. This data is used to extract resting places and trajectories of people between these resting places.

different motion models for individual people at the same time. Figure 3.1 depicts typical laser range data obtained during the data acquisition phase.

To determine the trajectories that are the input to our algorithm our system first extracts features which are local minima in the range scans that come from the person's legs[2]. Additionally, it considers changes in consecutive scans to identify a moving person. After determining the positions of the person based on the range scans we proceed with the next step and determine the resting places, i.e., the places where the person frequently stays for a while. This can easily be done by identifying time periods in which the person does not move. Figure 3.2 shows a map of a domestic residence as well as identified resting places.

Then we perform a segmentation of the data into different slices in which the person moves. Furthermore, we smooth the data to filter out measurement noise. Finally, we compute the trajectories which are the input to the learning algorithm described above, i.e. the sequence of positions covered by the person during that motion. When computing these trajectories we ignore positions which lie closer than $15cm$ to each other. A typical result of this process is shown in Figure 3.3.

3.5 Experimental Results

To evaluate the capabilities of our approach, we performed extensive experiments. The first set of experiments described here are designed to illustrate that our algorithm can learn complex motion patterns of people in different types of environments. In the second set of experiments we analyze the classification performance of learned models.

[2]A key precondition of our approach for extracting the positions of the person out of the laser range data is to know the relative positions of the laser range scanners. We used the system developed by Tacke [2002] to determine these positions.

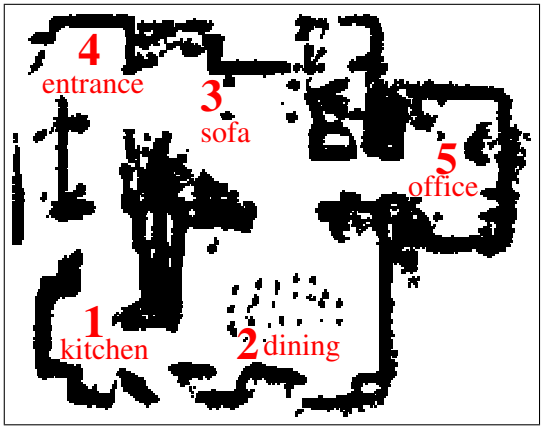


Figure 3.2: Map of a domestic residence. Indicated in red are identified resting places where the person frequently stayed for a while.

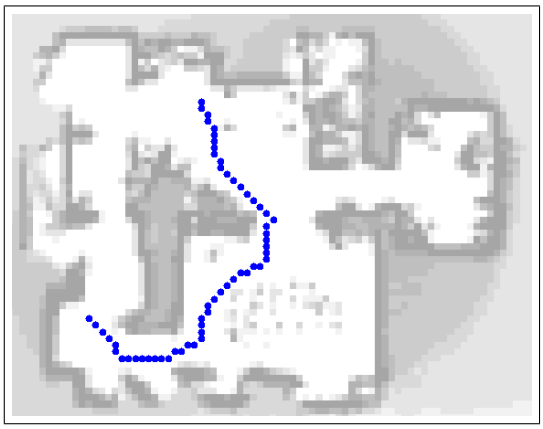


Figure 3.3: A single trajectory extracted from the laser data.

3.5.1 Learning Results

To analyze the ability of our approach to learn different motion patterns from a set of trajectories we performed extensive experiments in different environments. This included a domestic residence, an office environment, and a large hallway. The following section describes an experiment using data collected in the home environment. In this experiment the actual number of motion patterns is known beforehand. In the second set of experiments the number of motion patterns is unknown and has to be determined during the clustering process.

To get the random initial model we initialize the expectations with a unimodal distribution for each trajectory, i.e., for each d_i the expectations $E[c_{i1} \mid \theta^{[0]}, d], \ldots, E[c_{iM^{[0]}} \mid \theta^{[0]}, d]$ form a distribution with a unique randomly chosen peak. In all experiments we set the parameter β to 5 which means that the mean of each probability distribution is computed based on 5 subsequent positions on the trajectories. The standard deviation σ was set to $170cm$. We experimentally found out that these values yield good results.

Known Number of Motion Patterns

To see how our EM-based learning procedure works in practice consider Figure 3.4. In this example, a model for nine trajectories belonging to three different motion patterns has to be learned. There are three trajectories leading from resting place 3 to resting place 1, three trajectories leading from 3 to 2, and three trajectories leading from 2 to 3.

The leftmost image shows the initial model (the means of the three model components are indicated by circles). In the following images one can see the evolution of the model components during different iterations of the EM algorithm. Finally, the rightmost image shows the model components after convergence of the EM algorithm. As can be seen, the trajectories have been approximated quite well by the individual model components.

Unknown Number of Motion Patterns

In the remaining experiments the task was to correctly learn the motion patterns of the people along with their number. In principle, one could start our algorithm with a single model component and just introduce in each iteration (after convergence of the EM) a new model component for the trajectory which has the lowest likelihood given the current model. When the overall evaluation cannot be improved anymore by increasing the number of components, the system automatically alternates decreasing and increasing operations until the evaluation of the best model cannot be improved any more. However, to speed up the process we usually start our algorithm with a model that contains one component for six

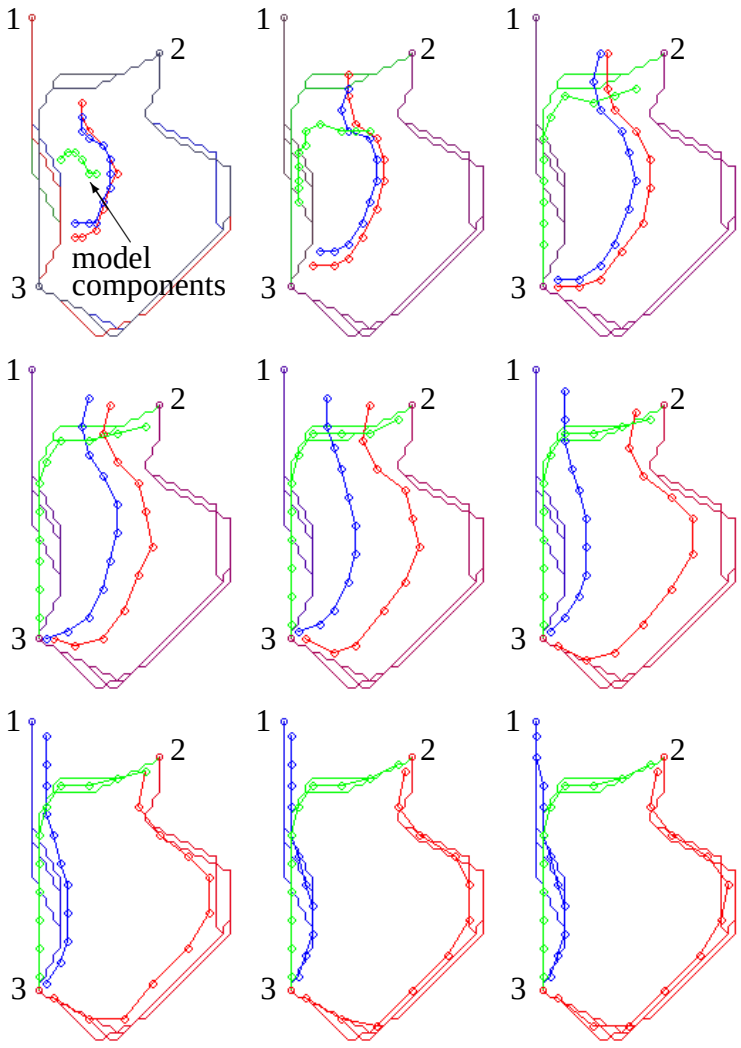


Figure 3.4: Trajectories of three different motion patterns and evolution of the model components during different iterations the EM algorithm. The means of the three model components are indicated by circles and the numbers indicate the three resting places.

trajectories in the input data. This reduces the learning time since typically fewer increasing operations are needed. In general, it is not easy to guess a good initial value of the number of motion patterns. Even if there is a heuristic about the correct number of motion patterns, initializing our algorithm with this number does not automatically lead to a small number of iterations. This is because the EM often gets stuck in local maxima which means that there exist redundant model

components that basically represent the same motion pattern. Those redundant model components are first eliminated before new model components are introduced for trajectories with low likelihood.

The first experiment was carried out with 42 trajectories from a data set of 61 trajectories recorded in a home environment (a map of this environment is depicted in Figure 3.7). To enhance the readability in this experiment we chose exactly three trajectories for each of 14 motion patterns. We grouped the trajectories belonging to the same motion pattern so that they appear as blocks of three in Figure 3.5. In general, of course, our algorithm works with an arbitrary set of trajectories.

We started our algorithm with a model of size $M = 10$. Figure 3.5 shows for different rounds of the EM during this run the resulting expectations $E[c_{i1} \mid \theta^{[j]}, d], \ldots, E[c_{iM^{[j]}} \mid \theta^{[j]}, d]$ for each trajectory d_i under the current model $\theta^{[j]}$ (the darker the color the higher the value). The x-axis of each plot represents the trajectories $d_1, \ldots, d_I$ and the y-axis contains the different model components $\theta_i^{[j]}$, for $i = 1, \ldots, M^{[j]}$.

The topmost image of Figure 3.5 labeled A shows the initialization of the expectations. The second image (plot B) shows the expectations after a few iterations. In the situation corresponding to C the EM has converged to a maximum in the log likelihood space given ten different model components. As can be seen from the figure, there are four model components that explain two different trajectories (the model components are indicated by arrows in the figure). In the next step (image D) our algorithm therefore tries to improve the data likelihood by introducing a new model component to which it assigns trajectory 25 which has the lowest likelihood given the current model. The situation when the EM has again converged to a maximum is shown in E. As before, another model component is introduced. We omit the corresponding images for the sake of brevity. The correct classification is found after our algorithm has introduced four additional model components (plot F). As can be seen from the figure, the system has determined a model in which all trajectories are correctly clustered, which means that the expected values of the correspondence variables of trajectories belonging to the same motion pattern have a peak close to 1 in the same model component. Nevertheless, our algorithm still tries to further improve this model by removing and adding a model component. However, since none of these operations increases the overall evaluation, our algorithm terminates and outputs the model corresponding to F. Plot G shows the expectations after convergence for a model with 15 components. As can be seen from the histogram three trajectories are assigned to two different model components. Because of the penalty term our algorithm prefers the lower complexity model corresponding to histogram F.

Figure 3.6 shows for the whole data set (61 trajectories) recorded in the home

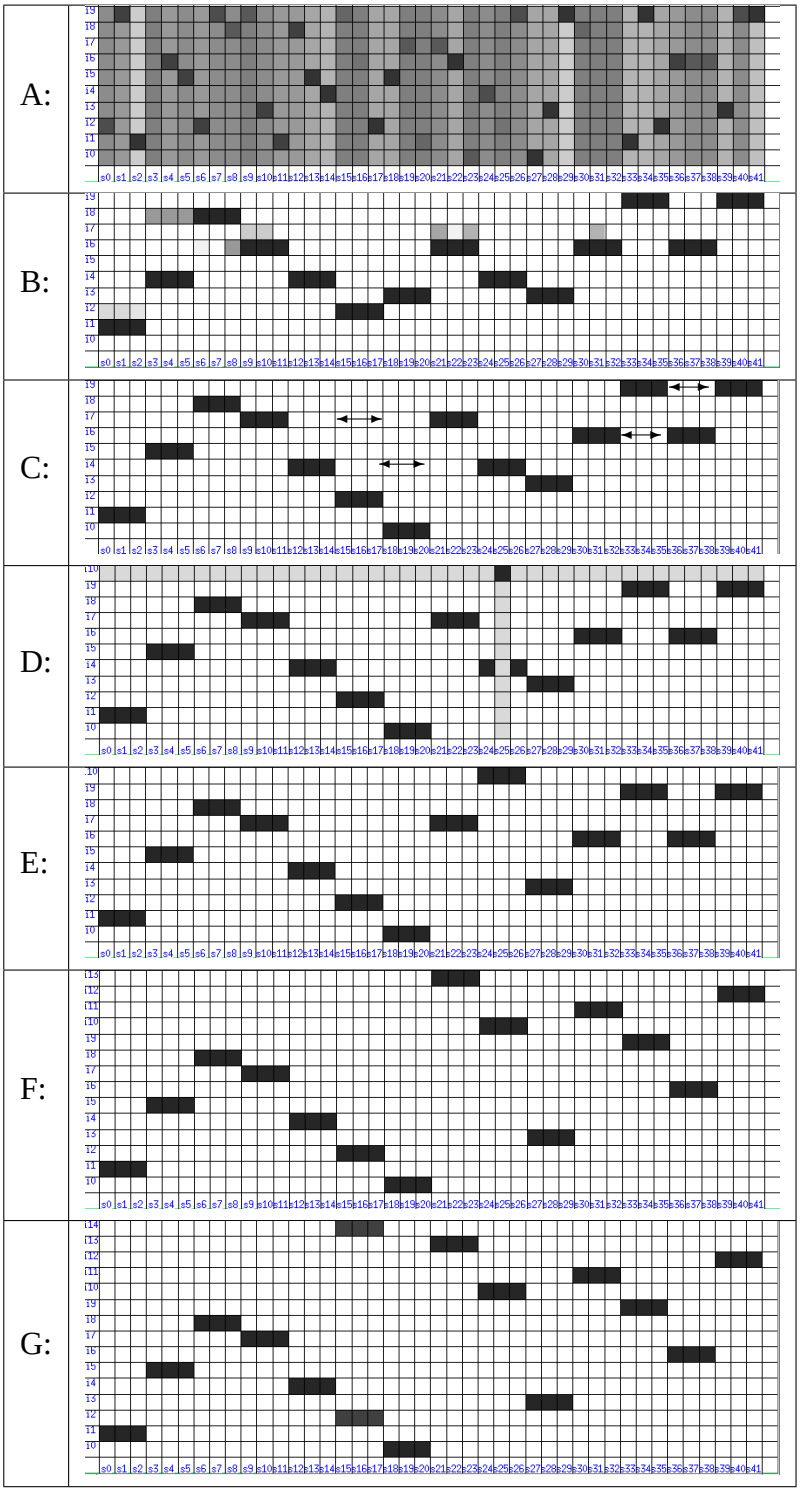


Figure 3.5: Expectations $E[c_{im} \mid \theta^{[j]}, d]$ computed in the different iterations of the EM algorithm (the darker the color the higher the value). The x-axis represents the trajectories and the y-axis the model components. See text for a detailed explanation.

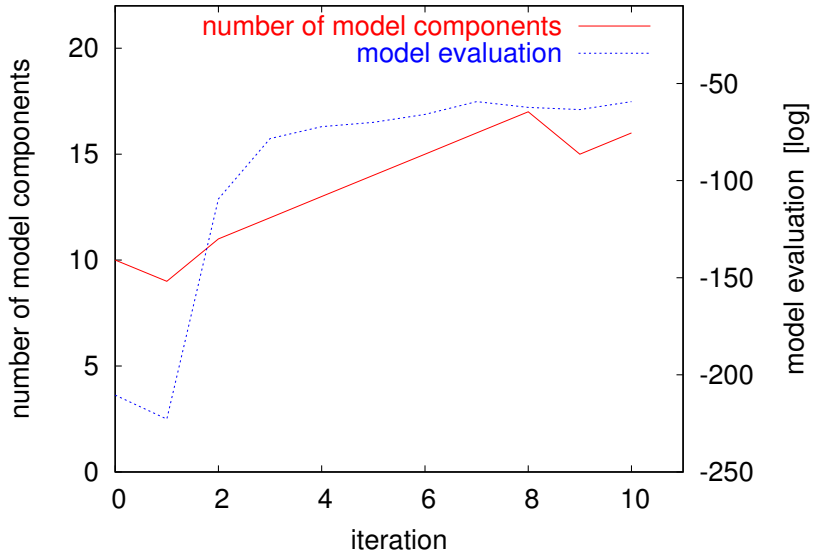


Figure 3.6: Evolution of the number of model components and the overall evaluation of the model for the domestic residence (see map in Figure 3.7).

environment the evolution of the overall evaluation as well as the number of model components. As can be seen from the figure, we started our algorithm with ten model components and our algorithm first tried to reduce the model complexity. Decreasing the number of model components did not increase the model evaluation and therefore our algorithm introduced new model components for the trajectories with low likelihood. In the end it stopped with 16 different model components when no improvement of the model could be achieved by increasing or decreasing the number of components. To illustrate that our algorithm correctly clustered the trajectories Figure 3.7 shows trajectories of two different model components after the convergence of the EM. Figure 3.8 depicts the learned motion patterns as well as the resting places which are indicated by numbers.

Figure 3.9 shows the evolution of the number of model components and the overall evaluation of the model when starting our algorithm with two model components. As can be seen, our algorithm needs 70% more iterations until it has clustered the trajectories.

We additionally applied our algorithm to data recorded in our office environment (see map in Figure 3.11). From the collected data we extracted 129 trajectories. Figure 3.10 shows the model complexity and model evaluation for one run in which we started with 20 different model components. As can bee seen from the figure, the algorithm decreased the model complexity until only 17 (non-redundant) components remained. Afterwards it increased the number of

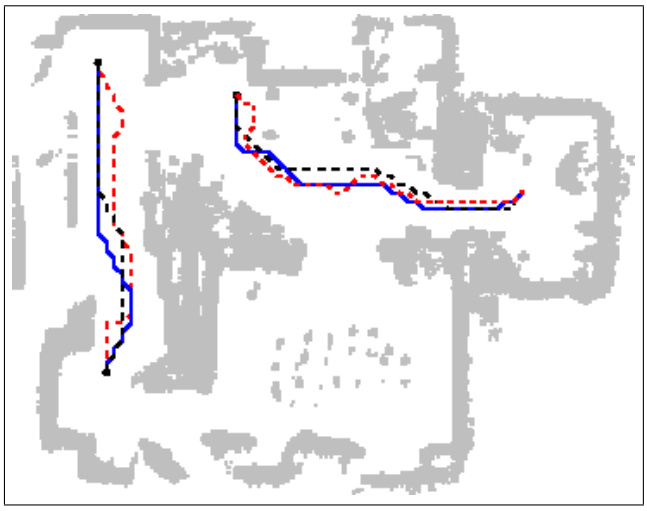


Figure 3.7: Trajectories of two different model components.

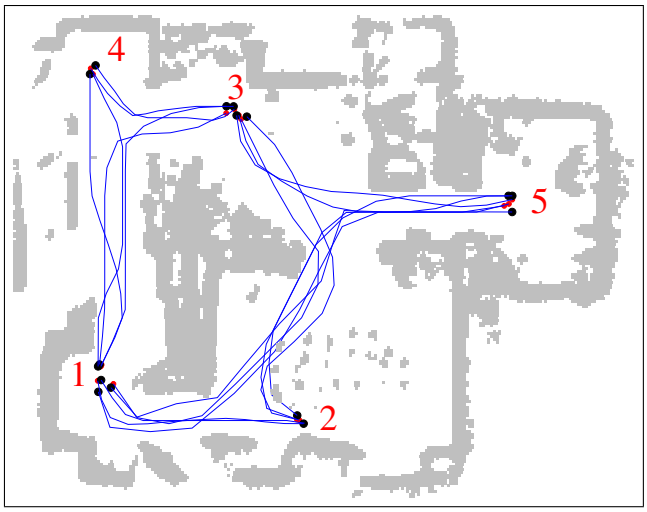


Figure 3.8: The 16 learned motion patterns between the identified resting places. Note that there are pairs of resting places for which we have two motion patterns, one for either direction.

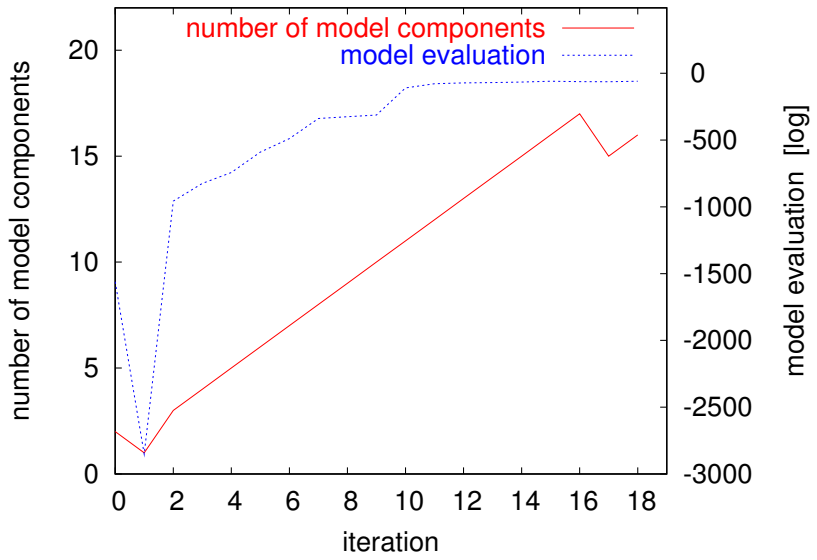


Figure 3.9: Evolution of the number of model components and of the model evaluation in the case in which our algorithm started with two model components and was applied to the same set of trajectories as in the experiment shown in Figure 3.6.

model components to improve the model evaluation. Finally, it terminated with the model correctly representing 49 different motion patterns. The trajectories of the learned model can be seen in Figure 3.11. The identified resting places are again indicated by numbers.

Figure 3.12 shows the evolution of the number of model components and the overall evaluation of the model for the case in which our algorithm was started with two model components on the same data set. As can be seen, now the algorithm needed more iterations (in this case over 35% more) to cluster the trajectories.

Corresponding results for a third data set recorded in the large entrance hall of a building on our campus are shown in Figure 3.13. In this experiment our algorithm was started with ten different model components for 59 trajectories and first reduced the number of model components. When there were 8 different model components left our algorithm tried to improve the model by introducing new components for the trajectories with low likelihood and finally stopped with 20 model components. A map of the environment as well as the learned motion patterns can be seen in Figure 3.15. Figure 3.14 shows the evolution of the number of model components and the overall evaluation of the model when starting our algorithm with two model components on the same data set. Again, the number of iterations was over 35% higher.

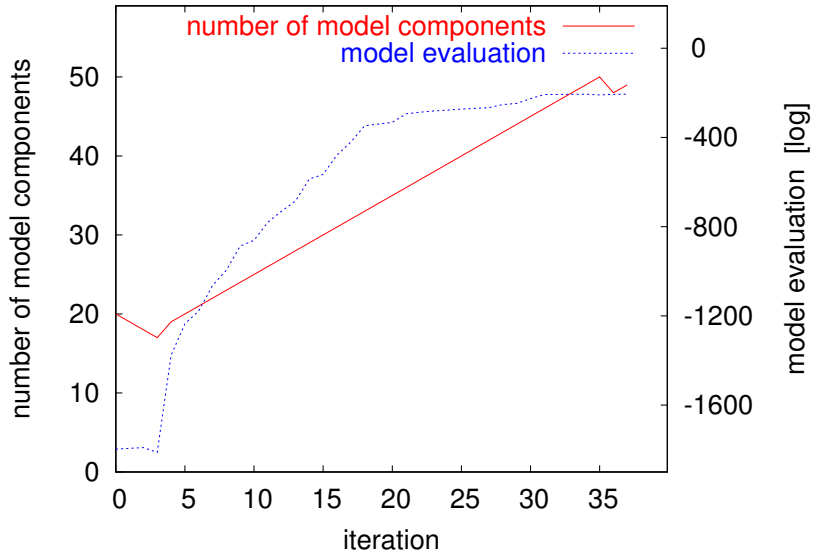


Figure 3.10: Evolution of the number of model components and the overall evaluation of the model during the application of our algorithm. In this case a model for 129 trajectories collected in our office environment (see map in Figure 3.11) has to be learned.

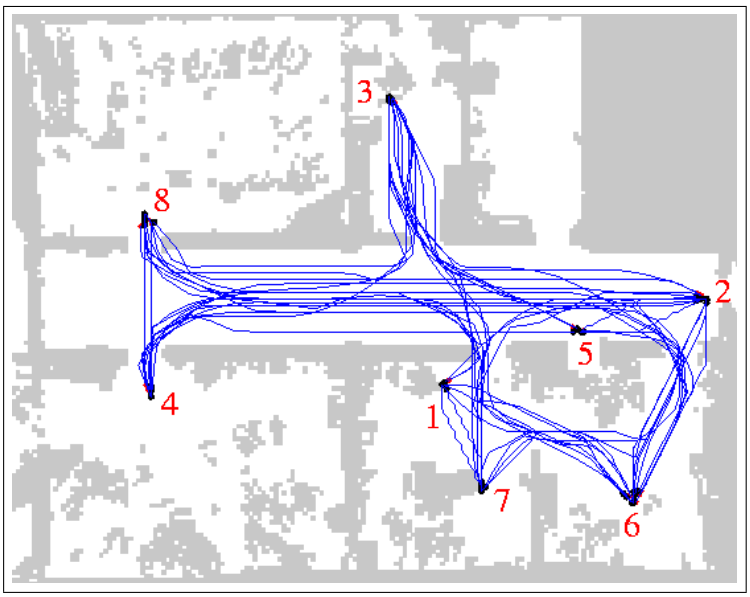


Figure 3.11: The 49 learned motion patterns in the office environment as well as the corresponding resting places.

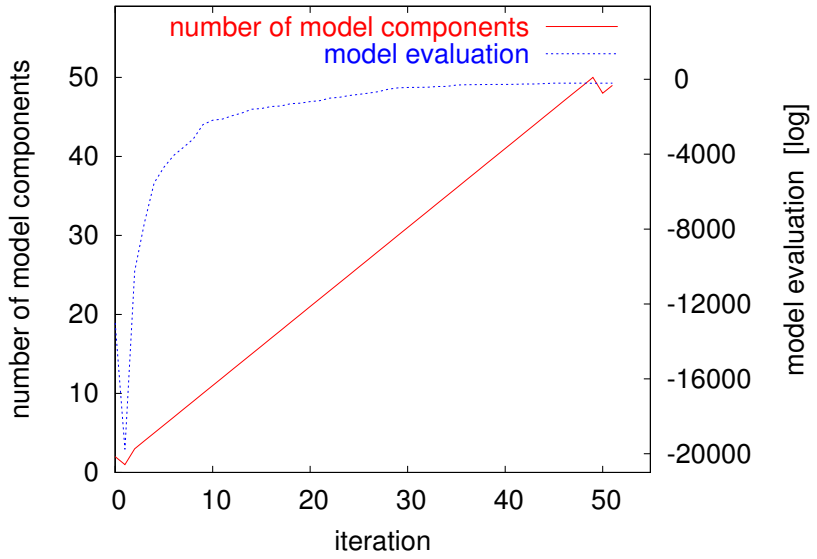


Figure 3.12: Corresponding evolution when starting our algorithm with two model components on the data set recorded in the office environment.

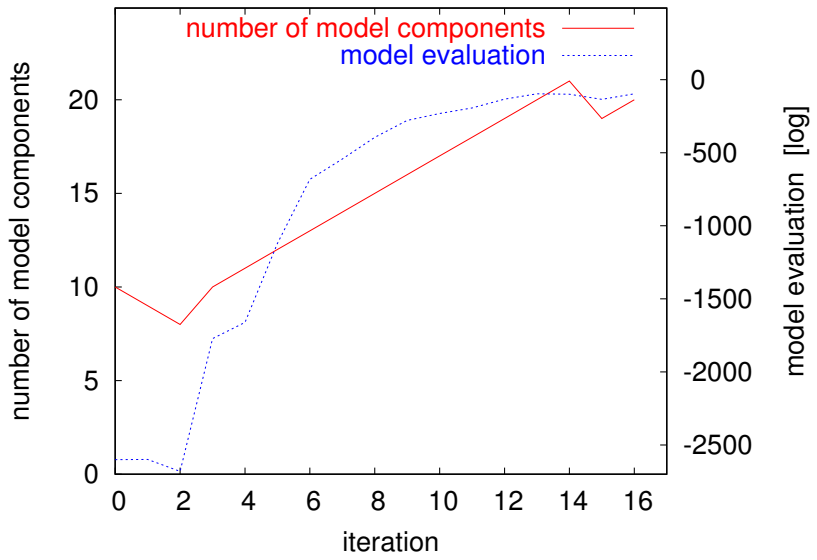


Figure 3.13: Evolution of the number of model components and the overall evaluation of the model for a data set recorded in the entrance hall of one of our main buildings on the campus (see map in Figure 3.15).

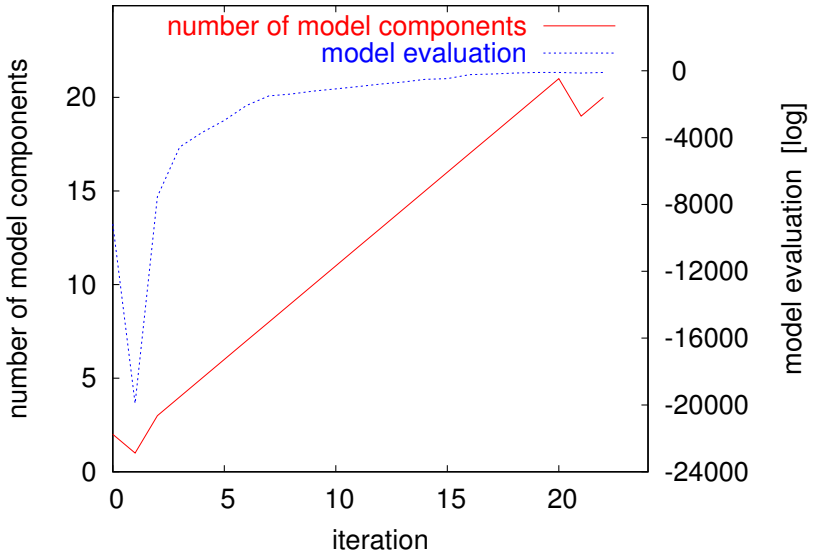


Figure 3.14: Evolution of the number of model components and the overall evaluation of the model when our algorithm was started with two model components on the same set of trajectories which was used in the experiment shown in Figure 3.13.

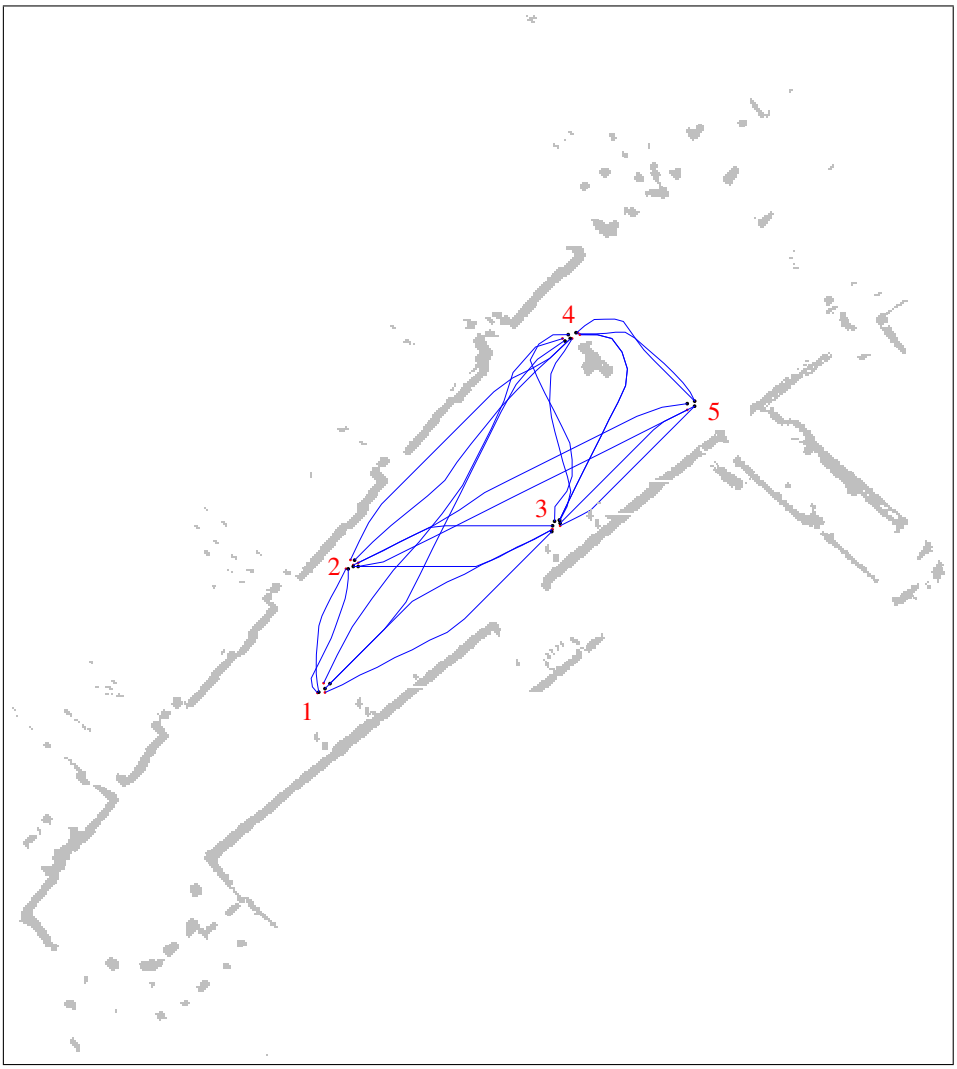


Figure 3.15: The 20 learned motion patterns for the entrance hall as well as identified resting places.

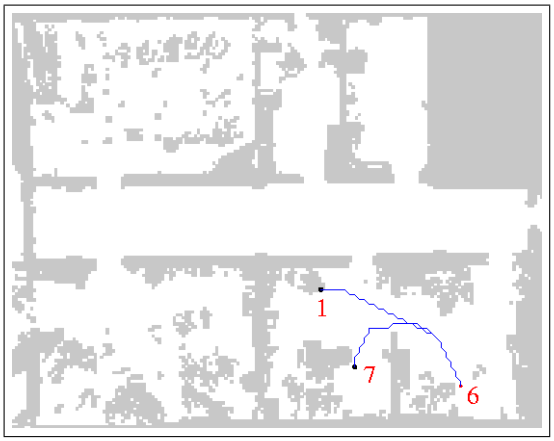


Figure 3.16: Two of the trajectories which are sometimes falsely classified into the same model component.

To evaluate the performance of our approach we carried out a series of experiments using several data sets. In each experiment we chose a random set of trajectories and counted the number of correct classifications. It turned out that our algorithm was able to learn the correct model in 96% of all cases. Thus, even if the initialization of the expectations seriously influences to which situation the EM algorithm converges, using the selective resetting and elimination strategy our algorithm is able to compute the correct classification of the trajectories and thus the correct motion patterns in most of the cases. We furthermore did not discover evidence that the number of model components we initialized our algorithm with has an influence on the overall result.

To analyze why our algorithm sometimes fails to correctly cluster the trajectories consider Figure 3.16 which shows two trajectories belonging to two different motion patterns that are often clustered into the same model component. As can be seen from the figure, both trajectories are extremely similar. Even if the model which has an individual component for both trajectories has a higher expected data likelihood, our algorithm prefers the model with just one component due to the penalizing of higher complexity models.

Discussion

Note that the output of our learning algorithm depends on the standard deviation σ and on the number of probability distributions K used to represent the motion patterns. It is clear that smaller values of σ will result in a higher number of model components. Furthermore, if there is only a relatively small number of trajectories for one motion pattern compared to the other motion patterns, our algorithm tends

to underestimate the number of model components and to assign these trajectories to other clusters. Due to the assumption that all model components have the same length, our algorithm prefers to cluster longer trajectories into single components rather than short trajectories. This is because the distance between long trajectories and their cluster is typically higher than for short trajectories. A solution to this problem would be to additionally estimate the number of probability distributions constituting each particular motion pattern. This aspect will be subject of future research.

Furthermore, there are several alternative clustering approaches that might be applicable to our problem of learning motion patterns. For the purpose of comparison we also implemented the classical k-Means algorithm. This algorithm differs from the approach used here in that in each iteration each data trajectory d_i is assigned to the "closest" model component, i.e., the model component θ_m which has the highest probability that d_i is observed given the person is engaged in θ_m. In our experiments it turned out that the classical k-Means algorithm yields a similar success rate compared to our approach. The advantage of our technique, which allows a "fuzzy" assignment of the trajectories to the model components, lies in the probabilistic framework and its correspondence to the EM algorithm as a general optimization procedure.

3.5.2 Prediction Accuracy

To evaluate the capability of our learned motion patterns to predict human motions we performed a series of experiments.

In the first experiment we used the data collected in home and in the office environment. In each experiment we randomly chose fractions of test trajectories and computed the likelihood of the correct motion pattern using the approach described in Section 3.3. Figure 3.17 shows the average likelihood of the correct motion pattern depending on the length of the observed fraction. As can be seen from the figure, the classification results are quite good and our approach yields motion patterns allowing a mobile robot to reliably identify the correct motion pattern. For example, if the robot observes 50% of a trajectory, then the probability of the correct motion pattern is about 0.6 in both environments.

Figure 3.18 illustrates for one trajectory of the person in the office environment the evolution of the set of possible motion patterns. The dashed black line corresponds to the trajectory of the person, which started to move at resting place 2. In the beginning all resting places are possible target locations. When the location labeled A is reached the motion pattern which leads to resting place 6 can be eliminated from the set of hypotheses because the corresponding likelihood gets too low. Thus, even if our system is not able to uniquely determine the intended goal location, it can already predict that the person will follow the corridor during the

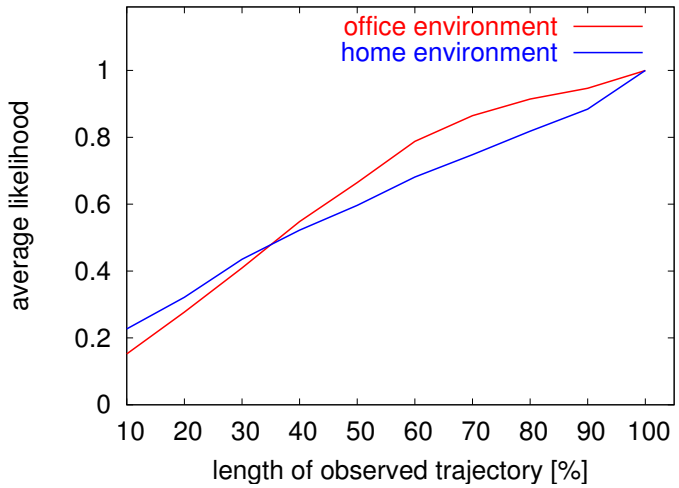


Figure 3.17: Average likelihood of the correct motion pattern after observing fractions of trajectories.

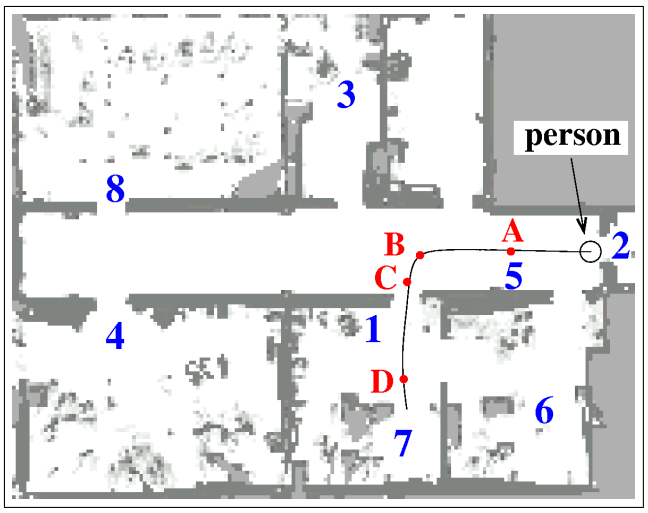


Figure 3.18: The black line corresponds to the trajectory of a person. The labels A, B, C, and D indicate locations at which motion patterns can be excluded because their likelihoods have become too low.

next steps. When the person reaches location B the system can also exclude resting place 5. When the person reaches position C resting place 3 becomes unlikely as well and the motion patterns leading to 7 and 1 become the most probable. Finally, at location D out system correctly predicts that resting place 7 is the target location. This experiment illustrates, that the results of the prediction are useful even in situations in which there are ambiguities about the actual intention of the person.

3.6 Representing Motion Patterns by Flow-Fields

In this section we discuss a different approach to represent the motion patterns and explain why this approach is not applicable in our context.

The idea is to learn probabilistic "flow-fields" of motion, which define local transition probabilities for each state in the two dimensional space. A flow-field F_m, with $1 \leq m \leq M$, is composed of two types of probability distributions (similar to a Markov chain [Cox and Miller, 1965], see also Section 5.2). The first distribution assigns to each environment location x in the discretized 2D space the probability $P(x \mid F_m)$ that this location x is the initial location of the person given it is engaged in the m-th motion pattern. Second, it also assigns to each environment location x a conditional probability distribution over possible successor locations x'. This probability, denoted $P(x' \mid x, F_m)$, specifies our expectations over successor locations x' given the person was at location x one time step earlier, and if we know for a fact that the person is engaged in motion pattern m. The totality of all such conditional probabilities for a specific value of m is called a probabilistic flow-field.

It is straightforward to define the likelihood of a trajectory $s_i = \{s_i^1, s_i^2, \ldots, s_i^{T_i}\}$ under the flow-field F_m as the product of the probabilities which correspond to the transitions of the trajectory times the probability of the initial location:

$$P(s_i \mid F_m) \;=\; P(s_i^1 \mid F_m) \prod_{t=1}^{T_{i-1}} P(s_i^{t+1} \mid s_i^t, F_m). \tag{3.32}$$

The representation of motion patterns by flow-fields seems to be very natural. However, a major problem exists which makes it impossible to come up with the correct number of different motion patterns using this representation. In the following we first explain how to compute the set of flow-fields which has the maximum expected data likelihood using the EM algorithm. After that we discuss the problem which arises by using this approach for clustering.

3.6.1 Application of the EM Algorithm

The E-step computes the expectation $E[c_{im} \mid F^{[j]}, s]$ that the i-th trajectory belongs to the m-th flow-field in the current model $F^{[j]}$ by combining Eq. (3.32) and Eq. (3.19):

$$\begin{aligned} & E[c_{im} \mid F^{[j]}, s] && (3.33)\\ &= \eta' P(s_i \mid F_m^{[j]}) && (3.34)\\ &= \eta' P(s_i^1 \mid F_m^{[j]}) \prod_{t=1}^{T_{i-1}} P(s_i^{t+1} \mid s_i^t, F_m^{[j]}). && (3.35) \end{aligned}$$

The M-step calculates new transition probabilities $P(x' \mid x, F_m^{[j+1]})$ for each flow-field $F_m^{[j+1]}$, and each pair of positions x and x' and by identifying the probability $P(x \mid F_m^{[j+1]})$ for each cell x that it is the starting location of the represented motion pattern. The maximum likelihood estimates of such probabilities are the empirical frequency count ratios. The equations are basically equivalent to those used in Hidden Markov Models (HMMS) for learning model parameters [Rabiner and Juang, 1986, Bilmes, 1997]:

$$P(x \mid F_m^{[j+1]}) = \frac{1}{I} \sum_{i=1}^{I} I_{s_i^1 = x} \; E[c_{im} \mid F^{[j]}, s], \tag{3.36}$$

$$P(x' \mid x, F_m^{[j+1]}) = \frac{\sum_{i=1}^{I} E[c_{im} \mid F^{[j]}, s] \sum_{t=1}^{T_i - 1} I_{s_i^t = x \wedge s_i^{t+1} = x'}}{\sum_{i=1}^{I} E[c_{im} \mid F^{[j]}, s] \sum_{t=1}^{T_i - 1} I_{s_i^t = x}}. \tag{3.37}$$

Here I is the indicator function that is 1 if its argument is true, and 0 otherwise. Notice that our empirical ratios are weighted by the expectations $E[c_{im} \mid F^{[j]}, s]$ computed in the E-step. If data is scarce (as it is the case in our experiments), these probabilities have to be smoothed over neighboring locations so as to avoid overfitting of scarce data. In our implementation, this is achieved by convolving the estimated transition probabilities with a Gaussian kernel. The resulting flow-fields are, strictly speaking, not maximum likelihood estimators, but they generalize much better to unseen data when training data is scarce. Furthermore, in order to be able to compute the log likelihood, for each s_i and F_m it must be ensured that the likelihoods $P(s_i \mid F_m)$ are greater than zero. This can be achieved by setting the transition probabilities to a minimum value if they have a lesser value.

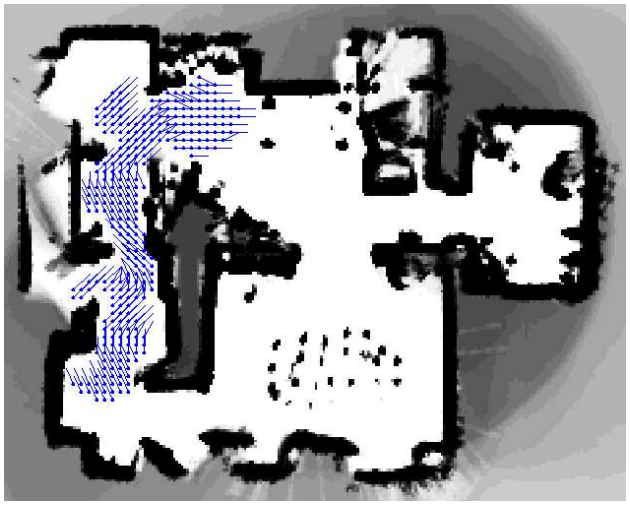


Figure 3.19: Maximum likelihood transitions for one motion pattern (shown here are only transitions with a probability greater than 0.15).

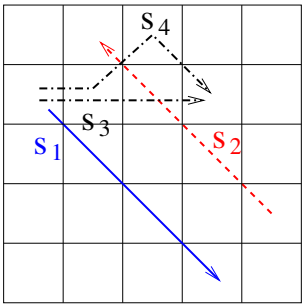


Figure 3.20: Four trajectories for which flow-fields have to be learned.

Figure 3.19 shows the maximum likelihood transitions for the typical motions carried out by the person when it goes from the kitchen to the living room in the home environment (we only show transitions which have a probability greater than 0.15).

3.6.2 Drawback of Flow-Fields in the Context of Clustering

As can be seen, the flow-fields model typical motion patterns in a natural way. However, one major problem arises during the application of the EM: When using these flow-fields to cluster the input trajectories it is not really penalized when totally different motion patterns are represented by the same flow-field. As an

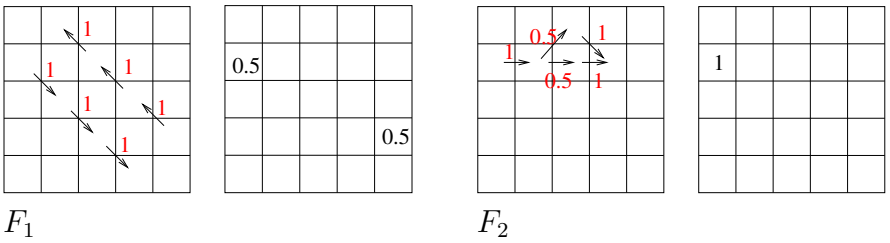


Figure 3.21: Two model components which can result after convergence of EM. The left hand side of each model component depicts the transition probabilities and the right hand side shows the probability distributions over the initial states.

example consider Figure 3.20. Here, we have four trajectories, each of which has three transitions. Let us assume EM converges to a local maximum, in which the trajectories s_1 and s_2 and the trajectories s_3 and s_4 are clustered together into model components F_1 and F_2 respectively. The model components depicted in Figure 3.21 are the resulting ones after convergence of the EM. The left hand side of each model component depicts the transition probabilities of the corresponding flow-field[3]. The right hand side of each model component shows the probability distributions, which indicate the probability that the corresponding states are the initial locations of the trajectory the person. For the likelihoods of the trajectories we have $P(s_1 \mid F_1) = 0.5$, $P(s_2 \mid F_1) = 0.5$, $P(s_3 \mid F_2) = 0.5$, and $P(s_4 \mid F_2) = 0.5$ according to Eq. (3.32). These results are rather implausible because the model component F_1 represents two totally different trajectories. Actually, it should be penalized that the trajectories s_1 and s_2 are clustered together and these trajectories should have a lower likelihood given this model than s_3 and s_4.

The described problem often leads to a wrong estimation of the number of model components. If the penalty term in Eq. (3.26) is high than this approach outputs a model in which totally different trajectories are clustered into one component since the gain in clustering them into different model components is not high enough. To find a good strategy to penalize a model in which totally different trajectories are assigned to the same model component is not easy. This is because the input trajectories can traverse a different number of states. Therefore, for example, it does not make sense to count the number of transitions in each flow-field which have a probability greater than a pre-defined threshold and to discount the likelihood according to this number.

In contrast, our approach, which uses Gaussian distributions to represent the positions of the people at different time steps, does not have the described unde-

[3]Note that we did not use a minimum transition probability and also omitted smoothing over neighboring locations for simplicity reasons in this illustrative example.

sired property. Using our approach it is ensured that trajectories of totally different motion patterns have a much lower likelihood when they are clustered together compared to the case when they are clustered into individual components.

3.7 Related Work

A variety of techniques has been developed that allows a robot to estimate the positions of people in its vicinity or to predict future poses. For example, the approaches presented by Tadokoro *et al.* [1995] and Zhu [1991] use given probabilistic motion models in order to predict future poses of observed objects. The techniques developed by Schulz *et al.* [2003a] and Montemerlo *et al.* [2002b] are able to robustly track multiple persons in the sensor range of a mobile robot equipped with laser range finders using particle filters [Bar-Shalom and Fortmann, 1988]. Kluge *et al.* [2001] also implemented a system to track people but they do not apply a motion model to the objects. Thus, they cannot reliably keep track of individual moving objects over time and deal with temporary occlusion. The same drawback has the approach presented by Lindström and Eklundh [2001] which uses Gaussian hypotheses to detect and track moving objects from a moving platform in furnished rooms. Rosencrantz *et al.* [2003] introduced variable-dimension particle filters to track the location of moving objects even if they are temporarily occluded. There are also many vision-based techniques that are designed to keep track of moving objects and which show robust behavior even in the case of temporary occlusions (for example [Rosales and Sclaroff, 1998, MacCormick and Blake, 1999]). Feyrer and Zell [2000] use a combination of vision and laser data to detect people in front of stationary background and to track them. Fod *et al.* [2002] use multiple statically mounted laser range finders and apply Kalman filters [Kalman, 1960] to maintain an estimate of the positions of people in everyday environments whereas Krumm *et al.* [2000] deployed multiple stationary cameras for multi-person tracking in a living room. Lavalle *et al.* [1997] presented a system that is able to follow of a moving target. They explicitely take into account possible short-term occlusions by static obstacles in the environment. Riley and Veloso [2002b] use predefined opponent models to predict opponent agents behavior in the RoboCup domain. Foka and Trahanias [2002] suggested to predict the movements of people using manually defined “hot points” which the people might be interesting in approaching.

The majority of the described techniques assume the existence of some models of the motion behavior of the tracked objects. Our approach, in contrast, is able to learn such models. The system described by Kruse [1998] uses a camera system mounted on the ceiling to track people in the environment and to learn where the people usually walk in their workspace. However, the system does not take

into account that the behavior of the people may vary depending on their intentions. Nguyen *et al.* [2003] proposed to use an Abstract Hidden Markov mEmory Model (AHMEM) to infer intentions of people. The idea of an AHMEM is to model higher level behaviors by a stochastic sequence of more simple behaviors at the lower levels. The authors apply an EM-based learning method for labeled trajectories to determine the transition probabilities for the states at the lowest level (grid cells) and assume that the landmarks the people want to approach are given. Our approach in contrast applies an unsupervised clustering method to the observed trajectories and is also able to automatically infer resting places which correspond to the landmarks in the AHMEM. The approach recently presented by Liao *et al.* [2004] learns the structure and parameters of an 3-level Abstract Hidden Markov Model. The authors apply EM to GPS data collected by a user which wears a GPS unit. The learned model is able to infer the user's daily movements and use of mode of transportation. Furthermore, it supports detecting novel events that may indicate user errors. Walter *et al.* [Walter *et al.*, 2001] proposed a method to learn hand gestures. Their approach assumes that the individual gestures can be viewed as repetitive sequences of atomic components and that they start and end in resting positions (this is similar to our approach to extract the trajectories). The authors model the atomic components by a mixture of Gaussians. They apply the EM algorithm to learn the atomic components and use the Minimum Description Length (MDL [Rissanen, 1984]) to estimate their number. After they have clustered the extracted trajectories into atomic components they analyze the observation sequence to concatenate consecutive atomic components. Since they found out that this way the number of different gestures tends to be over-estimated they apply EM and MDL a second time to reduce the number of gestures.

The approaches to clustering data sequences presented by Li *et al.* [2001] and Smyth [1997] are based on HMMs. Li *et al.* start with one cluster and gradually increase the number of clusters by introducing a new component for the data sequences which have the lowest likelihood given the current partition. Besides estimating the number of clusters this technique also estimates the number of states for the HMMs by using BIC. The approach, which penalized HMMs with a higher number of states, is not applicable for learning motion patterns of people as explained in the previous section. Smyth first learns an individual HMM for each single data sequence. Afterward, this approach uses the symmetrized distance matrix to cluster the data sequences into M groups. M is then chosen as that value which yields the best model. Sebastiani *et al.* [1999] follow a similar approach. They first learn an individual Markov chain for each data sequence and then merge one pair of chains in each step. They try out to merge all pairs of chains and stop the merging process when the model cannot be improved. The latter two approaches are not applicable in our context of learning trajectories because they are vulnerable to clustering totally different trajectories into the same

cluster as pointed out in the previous section.

Illmann *et al.* [2002] apply a fusion of omnidirectional vision and information of a laser range finder to acquire basic motion patterns like straight motion, wandering aimlessly, or entering a queue. The authors use the symmetric Kullback-Leibler divergence to compare pairwise trajectories and to cluster them. However, it is not obvious how to choose an appropriate stopping criterion. Gaffney and Smyth [1999] proposed a mixture regression model to cluster trajectory data. The authors applied their approach to cluster hand movements extracted from video streams into a given number of classes. Johnson and Hogg [1995] learn probability density functions of typical object trajectories to detect atypical behaviors. Compared to the our work, these approaches lack a technique to estimate the number of different motion patterns. The goal of the work by Stauffer and Grimson [2000] is also to detect unusual events. They learn codebooks of a given number of prototypes which is used for automatic hierarchical object classification. Rosales and Sclaroff [2003] analyze 3D trajectories to learn and recognize a given number of typical actions like walking, running, and biking. They use an extended Kalman filter [Welch and Bishop, 1997] to track objects in a video sequence. Oliver *et al.* [2002] use data obtained from various sensors (like microphone, camera, keyboard, and mouse) as input to a two layer HMM architecture to infer the state of a user's activity. Each layer is connected to the next higher layer via its inferential results and is trained independently with different feature vectors. The authors train independent HMMs for a given number of sounds/behaviors. Finally, Guralnik and Haigh [2002] use sequential pattern mining to learn typical behaviors of humans in their homes. They installed 10-20 sensors of different types in a home. Their algorithm uses this data to learn sequences of rooms in which room the person was acting. Their algorithm uses domain knowledge to extract sequences of rooms the person was acting in. These sequences are then analyzed by a human expert to identify complex behavior models. The approach described here can be seen as an alternative way of learning the sequences of rooms that does not require domain knowledge.

3.8 Conclusion

In this chapter we presented a method for learning motion patterns of people. Our approach applies the popular EM algorithm to cluster similar trajectories into single patterns. Additionally, it is able to estimate the number of motion patterns using the Bayesian Information Criterion. The output of our algorithm is a collection of motion patterns, each corresponding to a principle motion activity of a person. Using the resulting motion patterns our system can predict the motions of people based on observations made by a robot.

Our approach has been implemented and applied to range data recorded with laser range sensors. In practical experiments we demonstrated that our method is able to accurately learn typical motion patterns of a person in a domestic residence, in an office building, and in an large entrance hall. We furthermore described how to use the resulting models to predict the motions of people and presented reliable classification results for observed trajectories.

Chapter 4

Adapting Navigation Strategies Using Motion Patterns of People

4.1 Introduction

In this chapter we consider the problem of how knowledge about typical motion patterns of people can be utilized to improve the navigation behavior of the robot. In particular, we are interested in predicting the motions of people and instructing the robot to choose appropriate detours so that the risk of interferences with people is minimized.

As an example, consider the situations illustrated in Figure 4.1. In the left image a robot is moving from right to left in a corridor. At the same time, a person is walking down the corridor from left to right. In this particular situation, the robot needs to be able to detect the person and to predict its future actions in order to prevent interfering with it. A similar situation is depicted in the right image.

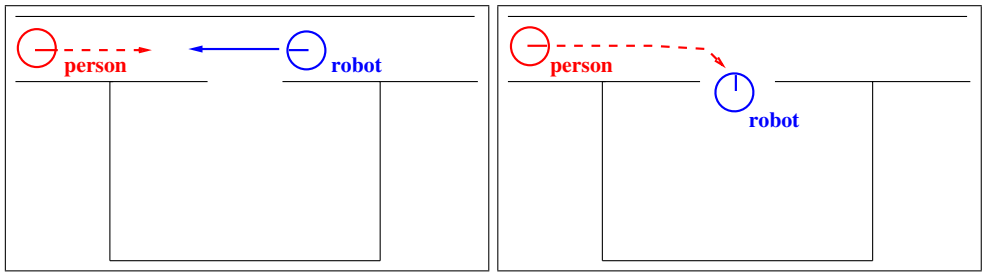


Figure 4.1: Situations in which a robot interferes with a person. In both cases the knowledge that the path of the person leads through the position of the robot would help the robot to avoid this conflict.

Here, the robot is standing in a doorway and a person, that wants to enter the room, is approaching the robot. Again, if the robot fails to detect the intention of the person and to react appropriately, the person cannot enter the room immediately. Whereas the conflict indicated in the left image could possibly be avoided with a linear prediction of the person's path, the conflict depicted in the right image can only be avoided if the robot anticipates that the person will enter the room. Then the robot can react early enough and the person does not need to wait until the robot moves aside.

We introduce a technique that allows a mobile robot to predict future motions of detected people and to incorporate this knowledge into its navigation plans. In particular, we describe a probabilistic technique to determine potential motion patterns the people might follow. The knowledge about potential intentions is then used to plan the actions of the robot in its configuration time-space. This way, the robot is able to avoid interfering with people by choosing trajectories that stay away from the predicted paths of the people in its vicinity. As a result, the robot can actively move out of the way of the people already at an earlier stage than with purely reactive approaches. Such a behavior is especially useful in environments with narrow passages since the robot comes too close to the people and forces them to slow down if it does not correctly predict their movements.

In the following section we show how the learned motion patterns can be integrated into the path planning process. Section 4.3 introduces our approach to detect and keep track of people using laser-range data. In Section 4.4 we present several experiments. These experiments demonstrate that our approach can improve the behavior of a mobile robot by predicting trajectories of people in the vicinity using learned motion patterns. Finally, we discuss related work in Section 4.5.

4.2 Integrating Predicted Motions of People into Path Planning

Our approach uses motion patterns of people that are learned using the technique described in the previous chapter. Thus, we assume we have a set θ of M different motion patterns $\theta = \{\theta_1, \ldots, \theta_M\}$ a person might exhibit in its environment. A motion pattern θ_m with $1 \leq m \leq M$, is represented by K Gaussian distributions $\theta_m^k = N(\mu_m^k, \sigma)$ with mean μ_m^k and fixed standard deviation σ for all m and k. Each such Gaussian specifies for each data point x^t and each θ_m^k the likelihood $P(x^t \mid \theta_m^k)$ that the person is at location x^t given that step t of the trajectory x

corresponds to step k of motion pattern m:

$$P(x^t \mid \theta_m^k) \quad = \quad \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}\|x^t - \mu_m^k\|^2}. \tag{4.1}$$

In Section 3.3 we explained how to estimate the probability that a person is engaged in a motion pattern θ_m given an observation sequence z. Within this section we now focus on the question of how a robot can make use of this information to improve its navigation behavior. In particular, we want to focus on the question how the belief of the robot about possible motion patterns of people in its vicinity can be considered during the path planning process.

We use the A^* procedure [Nilsson, 1982] for path planning and for searching the minimum-cost path in the robot's three dimensional configuration time-space [Erdmann and Lozano-Pérez, 1987]. The configuration time-space of the robot is computed based on a static occupancy grid map [Moravec and Elfes, 1985] of the environment and the predicted positions of the surrounding people at any point in time (these techniques have been introduced in Section 2.3). To obtain a good heuristics for A^* we compute the cost-optimal path for each pair of grid cells in the two dimensional environment once using the deterministic value iteration (see for example [Sutton and Barto, 1998]). More precisely, we compute for each grid cell a value iteration with that cell as start and store the costs of the paths to all other cells. These costs are used as a heuristics for A^* in the three dimensional configuration time-space. This allows the robot to quickly replan its path whenever new measurements have arrived and the belief about the intended trajectories of the people has changed.

The cost for traversing a grid cell $\langle x, y\rangle$ is proportional to its occupancy probability $P_{\mathsf{occ}}(\langle x, y\rangle)$. To incorporate the robot's belief about future trajectories of the people, each cell $\langle x, y\rangle$ is additionally discounted according to the probability, that one of the persons covers $\langle x, y\rangle$ at a given time t. Suppose our robot has observed R persons and suppose $P_{\mathsf{cov}}(\langle x, y, t\rangle \mid z_r)$ is the probability that person r covers $\langle x, y\rangle$ at time t given the observation sequence z_r corresponding to this person. If we consider the individual persons independently, we can compute the cost $C_{\mathsf{cov}}(\langle x, y, t\rangle)$ introduced by the fact that $\langle x, y\rangle$ might be covered by one of the observed persons at time t as follows:

$$\begin{aligned} & C_{\mathsf{cov}}(\langle x, y, t\rangle) \\ & \quad = \quad \sum_{r=1}^{R} P_{\mathsf{cov}}(\langle x, y, t\rangle \mid z_r) \qquad (4.2) \\ & \quad = \quad \sum_{r=1}^{R} \sum_{m=1}^{M} \sum_{k=1}^{K} \sum_{k'=k}^{K} P_{\mathsf{cov}}(\langle x, y, t\rangle \mid \theta_m, k, k', z_r) \cdot P(\theta_m, k, k' \mid z_r). \qquad (4.3) \end{aligned}$$

The last transformation is due to the fact that we do not know where z_r starts on the motion pattern θ_m and therefore we have to sum over all possible combinations of start and end points (law of total probability; see also Section 3.3, Eq. (3.28)). It remains to describe how the individual terms $P_{\mathsf{cov}}(\langle x, y, t\rangle \mid \theta_m, k, k', z_r)$ and $P(\theta_m, k, k' \mid z_r)$ are computed. For the latter term, according to Bayes' Rule and the product rule (see Appendix A.2.1 and A.2.2), we have:

$$\begin{aligned} & P(\theta_m, k, k' \mid z_r) \\ &= \eta P(z_r \mid \theta_m, k, k') P(\theta_m, k, k') && (4.4) \\ &= \eta P(z_r \mid \theta_m, k, k') P(\theta_m) P(k, k' \mid \theta_m). && (4.5) \end{aligned}$$

where η is a normalizer. How we compute the likelihood $P(z_r \mid \theta_m, k, k')$ of an observation sequence z_r given motion pattern θ_m has been explained in Section 3.3, Eq. (3.29). $P(\theta_m)$ is the prior for θ_m, whereas the prior probability $P(k, k' \mid \theta_m)$ depends on the difference between the length of the given segment on the motion pattern θ_m and on the distance the person moved in the observation sequence z_r. Finally, we need to describe how we compute the probability

$$P_{\mathsf{cov}}(\langle x, y, t\rangle \mid \theta_m, k, k', z_r)$$

that a person engaged in motion pattern θ_m will cover $\langle x, y\rangle$ at time t given the observations z_r and given that z_r starts at θ_m^k and ends at $\theta_m^{k'}$. We use a Gaussian distribution to represent the uncertainty about the position of the person at time step t. The mean of this Gaussian is computed as that position on θ_m which has the distance $v \cdot \Delta_t$ from the latest observed position $x^{t'}$ at time step t'. Here v is the velocity of the person in the observations z_r and $\Delta_t = t - t'$. Thus, we predict the motion of the person starting from location $x^{t'}$ according to the average velocity v and the trajectory given by θ_m.

The overall cost for the robot to traverse a cell $\langle x, y\rangle$ at time t is then computed as:

$$C(\langle x, y, t\rangle) \;=\; \gamma(P_{\mathsf{occ}}(\langle x, y\rangle)) + C_{\mathsf{cov}}(\langle x, y, t\rangle). \qquad (4.6)$$

Thereby $\gamma(P_{\mathsf{occ}}(\langle x, y\rangle))$ is a threshold function which is infinite if $P_{\mathsf{occ}}(\langle x, y\rangle)$ exceeds 0.8, and $P_{\mathsf{occ}}(\langle x, y\rangle)$ otherwise. This avoids that paths for example lead through walls. Note, that this computation is similar to the computation of the cost of traversing a cell in the case of multi-robot systems (see Section 2.3, Eq. (2.1)).

Our approach can be used to predict motions of multiple persons and it can even deal with people not engaged in any of the learned motion patterns. If the trajectory of a person r cannot be associated well with any of the known motion patterns, the belief $P(\theta_m \mid z_r)$ is uniformly distributed which introduces higher costs for fields on learned motion patterns.

Note that the system presented in this chapter does not distinguish between individual persons. In the next chapter we present an approach that is able to identify people using vision data and which applies Hidden Markov Models (HMMs) [Rabiner and Juang, 1986] to maintain individual beliefs about the positions of people. These HMMs can be used to provide the costs C_{cov} needed for planning in the configuration-time space. This way, the technique introduced so far can be extended to incorporate different motion patterns for individual persons.

4.3 Detecting and Tracking People

To apply the technique described above, a robot must be able to detect people in its vicinity and to keep track of them. Similar to Schulz *et al.* [2003a] and Montemerlo *et al.* [2002b], our system extracts features which are local minima in the range scans that come from the legs of the people. It additionally considers changes in consecutive scans to identify moving people more reliably. We extract only features not belonging to static objects. Given these features we compute the observed position of a person by taking the mean of close-by features. When the robot is moving its sensor information is not as reliable as when it is standing. Therefore we use a Kalman filter to keep track of a person. A Kalman filter estimates the current position of the person given the estimated position at the previous time step and given noisy sensor input as it is explained in the following.

4.3.1 The Kalman Filter

A Kalman filter [Kalman, 1960, Welch and Bishop, 1997] is a recursive procedure to process sensor data. Figure 4.2 depicts a typical application scenario of the Kalman filter (this figure is adapted from [Maybeck, 1979]). A system is controlled by some known process and a measurement device provides information about the state of the system. The Kalman filter is needed in the case that influencing factors of various types exist which cannot be measured directly by the available sensors. A Kalman filter represents the posterior estimate of the state of a system by a Gaussian distribution.

Using a Kalman filter one can compute the optimal estimate of the state of a system given the system can be described by a linear model and given the noise of the process and the measurements is Gaussian and white. The latter means that the noise is not correlated in time. According to Maybeck [1979] the Kalman filter "processes all available measurements, regardless of their precision, to estimate the current value of the variables of interest, with the use of (1) knowledge of the system and measurement device dynamics, (2) the statistical description of the system noises, measurement errors, and uncertainty in the dynamic models, and

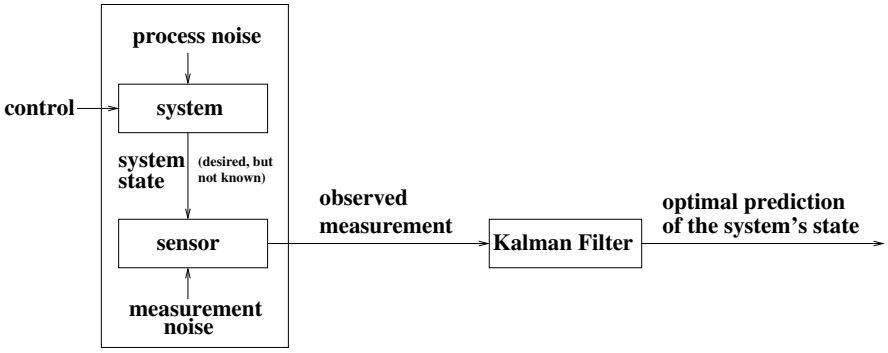


Figure 4.2: The task of the Kalman filter is the exact estimation of the state of a system which cannot be measured perfectly by sensors.

(3) any available information about initial conditions of the variables of interest." The result of the fusion process which combines the estimates and the measurements is a new estimation with a new error which is also Gaussian. The Kalman filter is optimal in the sense that the estimate of the state of the system minimizes the resulting error between the predicted and the actual state.

Figure 4.3 illustrates the individual steps of the Kalman filter (this figure is adapted from[Welch and Bishop, 1997]). In essence, it is a cycle which consists of a *prediction* and a *correction* step. The prediction step projects the state ahead in time and in the update step the projected estimate is adjusted using the current measurement. Given initial estimates about the state of the system and the error covariance, the Kalman filter iteratively computes the Kalman gain which indicates how much the new measurement is weighted during the following update of the new estimation of the state. Once the state is newly estimated the error covariance is updated, the prediction step is invoked which computes new projections of the state of the system and the corresponding error covariance.

In the following we introduce the mathematical equations of the Kalman filter. The task of the Kalman filter is to estimate the state $x \in \mathbb{R}^n$ of a discrete-time controlled process which can be modeled by two linear stochastic equations:

$$x_t = A_{t-1}x_{t-1} + Bu_t + w_{t-1} \tag{4.7}$$

and

$$z_t = Hx_t + v_t. \tag{4.8}$$

The matrix A_{t-1} specifies how the state x_t changes in relation to the previous state x_{t-1} without taking into account any control input or process noise. The matrix B

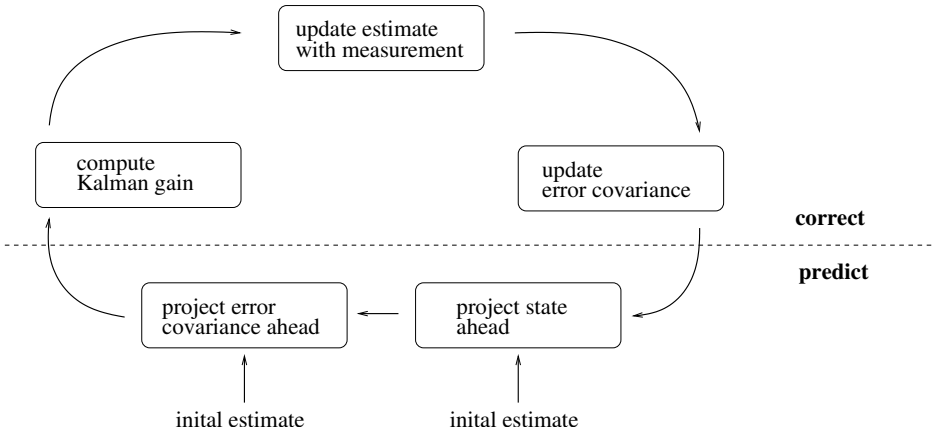


Figure 4.3: Diagram illustrating the operations during the iterations of the Kalman filter.

describes the influence of the control input u_t to the state x_t, whereas the measurement matrix H indicates the relation between the state x_t and the measurement z_t. H might also change each time step but in our case we assume it to be constant. The random variables w_{t-1} and v_t represent the mentioned Gaussian process and measurement noise. Since w_{t-1} and v_t are vectors, instead of variances we have covariance matrices Q and R for the process and measurement noise which we assume to be constant.

The *a priori* state estimate at step t given knowledge about the systems state prior to step t is denoted as $\hat{x}_t^-$. The corrected state estimate at step t given the measurement z_t is referred to as $\hat{x}_t$ and is called the *a posteriori* state estimate.

In the prediction step the following computations are carried out to get the a priori estimates for the state of the system $\hat{x}_t^-$ and the corresponding error covariance matrix P_t^-:

$$\hat{x}_t^- = A_{t-1}\hat{x}_{t-1} + Bu_t \tag{4.9}$$

and

$$P_t^- = A_{t-1}P_{t-1}A'_{t-1} + Q. \tag{4.10}$$

Here A'_{t-1} denotes the transpose of matrix A_{t-1}. Initially, $\hat{x}_0$ and P_0 have to be specified to compute $\hat{x}_1^-$ and P_1^-.

In the following correction step the a posteriori state estimate $\hat{x}_t$ is computed using the current measurement z_t according to the following equation:

$$\hat{x}_t = \hat{x}_t^- + K_t(z_t - H\hat{x}_t^-). \tag{4.11}$$

As can be seen, depending on K_t the difference between the predicted measurement $H\hat{x}_t^-$ and the actual measurement z_t is weighted more or less heavily in this linear combination. In the Kalman filter, K_t is chosen so that it minimizes the a posteriori error covariance (see for example [Brown and Hwang, 1992]):

$$K_t = P_t^- H'(HP_t^- H' + R)^{-1}. \tag{4.12}$$

The a posteriori error covariance matrix P_t for the corrected state $\hat{x}_t$ is then computed as:

$$P_t = (Id - K_t H)P_t^-. \tag{4.13}$$

4.3.2 Tracking People Using Kalman Filters

In our application the state of a person is represented by a vector $[x, y, v_x, v_y]'$. Whereas x and y stand for the position of the person, the terms v_x and v_y represent the velocity of the person in x- and y-direction.

The matrix B in Eq. (4.9) is in our case the zero matrix because we do not have an extern control input. The matrix A_t defines how the state of the person changes if the person continues walking with the estimated current velocities v_x and v_y:

$$A_t = \begin{pmatrix} 1 & 0 & \Delta_t & 0 \\ 0 & 1 & 0 & \Delta_t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{4.14}$$

Here Δ_t is the time elapsed until the next measurement. According to Eq. (4.9), the predicted state at the time of the next measurement is then:

$$\hat{x}_{t+1}^- = A_t\hat{x}_t = \begin{pmatrix} x + \Delta_t \cdot v_x \\ y + \Delta_t \cdot v_y \\ v_x \\ v_y \end{pmatrix}. \tag{4.15}$$

In our case, a measurement is the position of the person extracted out of the laser range data as explained in the beginning of this section. Since the sensors we are using generally do not provide the velocities v_x and v_y, which are also part of our state space, the measurement matrix H projects onto the first two components of the state space:

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}. \tag{4.16}$$

Accordingly, the predicted measurement for the next time step is:

$$z^-_{t+1} = H\hat{x}^-_{t+1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \hat{x}^-_{t+1} = \begin{pmatrix} x + \Delta_t \cdot v_x \\ y + \Delta_t \cdot v_y \end{pmatrix}. \quad (4.17)$$

Using the defined quantities the prediction and correction steps are iteratively executed whenever a new measurement of the position of the person is provided.

4.3.3 Keeping Track of Multiple Persons

To track multiple persons in the range scans, we apply independent Kalman filters, one for each feature. One problem which has to be solved in the case of multiple persons is to decide which feature is caused by which person. To solve this data association problem, we apply a nearest neighbor approach [Cox, 1993], i.e. we update a filter using the observation z_t that is closest to the predicted observation $z^-_t = H\hat{x}^-_t$.

To measure the distance between the predicted observation and a given observation we apply the Mahalanobis distance [Mahalanobis, 1930], which is often used in cluster analysis as a distance measurement [Everitt, 1974]. The squared Mahalanobis distance d^2_C between two data points x_i and x_j is defined as follows:

$$d^2_C = (x_i - x_j)' \cdot C^{-1} \cdot (x_i - x_j). \quad (4.18)$$

Here C^{-1} is the inverted covariance matrix of the data set. The advantage of the Mahalanobis distance compared to the Euclidean distance is that it takes into account the correlation between the parameters.

Accordingly, in our application the distance between the predicted observation z^-_t and a given observation z_t is computed as

$$d^2_{P_t} = (z^-_t - z_t)' \cdot (P^-_t)^{-1} \cdot (z^-_t - z_t), \quad (4.19)$$

where $(P^-_t)^{-1}$ is the inverted covariance matrix for the predicted state $\hat{x}^-_t$.

Since the robot can only cover a restricted area of the environment with its laser scanner we have to take into account that people disappear out of the field of view of the robot and that new people enter it. Thus, filters are removed when no updates can be made (because all observed features are farther away than a predefined threshold) to a filter for a given period of time (one second in our implementation). Furthermore, new filters are introduced for features which could not be associated with any of the existing filters.

As an application example, Figure 4.4 shows laser-range data and the estimated positions of two persons at different time steps. The upper left image shows the situation in which a second Kalman filter was initialized for the person on the right. In the following images one can see how the robot keeps track of the positions of the persons.

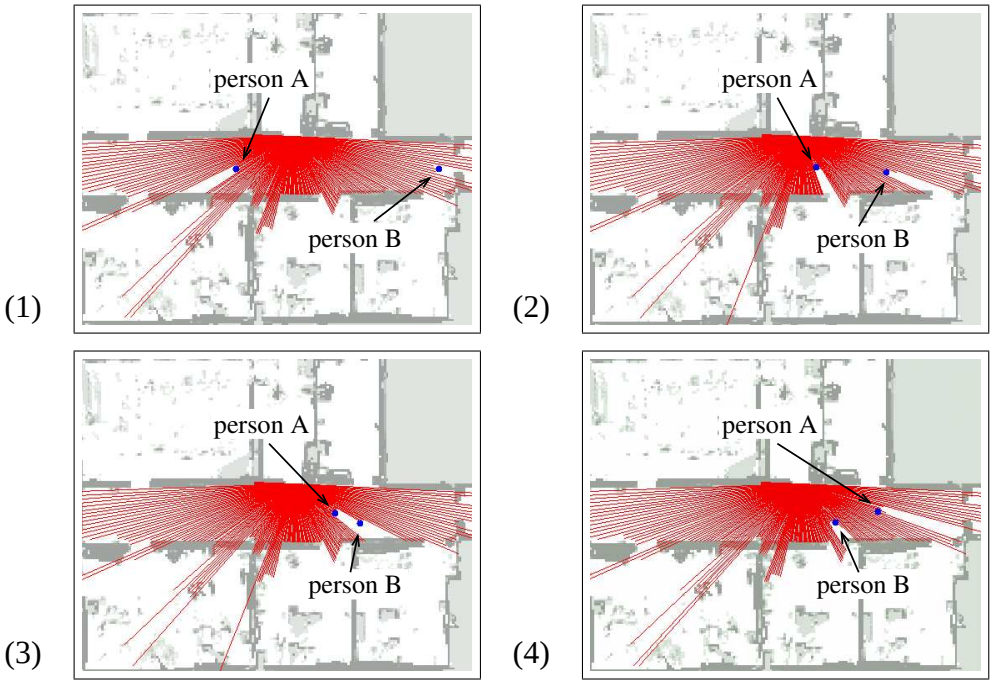


Figure 4.4: Tracking two persons using laser-range data and independent Kalman filters. A nearest neighbor approach is used to update the filters based on new observations.

4.4 Experimental Results

To evaluate the capabilities of our approach, we performed extensive experiments. The experiments illustrate that our method enables a mobile robot to correctly predict motions of people and to adapt its own behavior accordingly. We furthermore present an experiment which demonstrates that the behavior of a robot can significantly be improved by predicting the trajectories using learned motion patterns compared to the case when the robot performs only a linear prediction. All experiments were carried out using our B21r platform Albert (see Appendix A.1) in the corridor environment of our laboratory at the University of Freiburg.

4.4.1 Planning Detours

The first experiment is designed to demonstrate that our approach allows a mobile robot to reliably predict the possible trajectories of a person and to appropriately incorporate this information into its motion plans.

Here the robot was traveling along the corridor of our building. At the same time, a person walking in the opposite direction was approaching the robot. Fig-

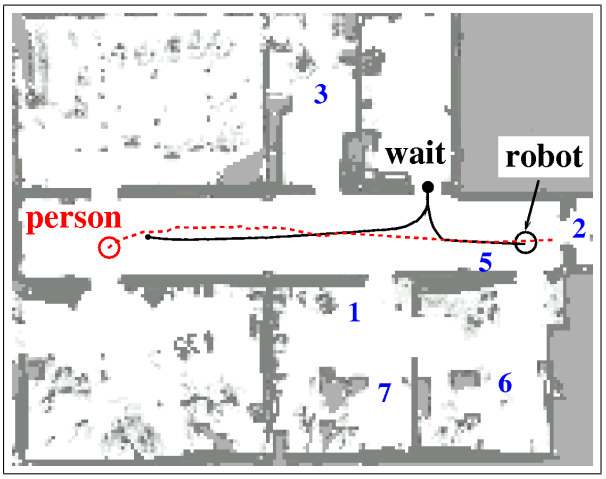


Figure 4.5: Albert anticipates that it probably will interfere with the person if it follows its original trajectory. Therefore the robot moves into a doorway to let the person pass by (images taken during this experiment are shown in Figure 4.6).

ure 4.5 shows the initial position of the robot and the position of the person when it was detected by the robot for the first time. Given the existing motion patterns, possible trajectories led to the resting places 1, 2, 3, 5, 6, and 7 which are also shown in Figure 4.5. Obviously, all trajectories led through the corridor so that the robot was likely to interfere with the person. Furthermore, the probability that the person followed the corridor to go to resting place 2 was very high. Accordingly, the cost-optimal action for the robot was to drive into the doorway to the right in front of it and to wait there until the person eventually had passed by.

Figure 4.5 shows the trajectory of the person (red line) as well as the trajectory of the robot (black line). As can be seen from the figure, the person went to the location 2 and the robot continued to move towards its designated goal point after the person had walked by. Figure 4.6 shows images of the robot and the person taken during this experiment.

4.4.2 Giving Space to People

The next experiment described here demonstrates that our technique can also be used to improve the behavior of the robot even in situations in which the robot is not performing a navigation task. Furthermore, it demonstrates that the knowledge about typical motion patterns of people can be highly useful to achieve an intelligent behavior of the robot.

In this particular situation (see Figure 4.7) the robot had no goal point and

Figure 4.6: To avoid that the person has to slow down Albert moves into the doorway and continues its way after the person had passed by.

rested in a doorway waiting for instructions. Then it realized that a person was approaching from the left. According to the learned motion patterns, Albert inferred a high probability that the person would enter the room through the doorway that it was blocking. The cost-optimal action according to the path planner was to drive to the middle of the corridor in order to give space to the person. Figure 4.8 shows images taken during this experiment.

If the robot did not take into account learned motion patterns and just performed a linear prediction about the person's movements or if it did not predict the movements at all it would not be able to infer the intention of the person to enter the room. Thus, the robot would not react early enough and the person would have to slow down and wait until the robot moved out of its way.

4.4.3 Multiple Persons

Figure 4.9 shows a situation in which Albert was about to leave a room while two persons were walking along the corridor and approaching the robot from either direction. The trajectories of the persons are depicted as solid lines whereas the designated path of the robot is indicated as a dashed line. Since Albert was not able to leave the room before person P_1 had walked by it stayed in the doorway.

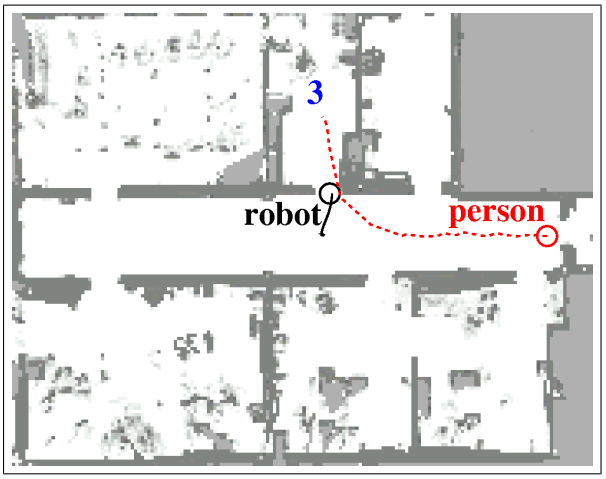


Figure 4.7: Albert anticipates that the person wants to enter the room through the doorway which it is blocking. Thus, Albert moves aside in order to let the person pass by (images taken during this experiment are shown in Figure 4.8).

Figure 4.8: Albert moves away from the doorway in order to let the person enter the corresponding room.

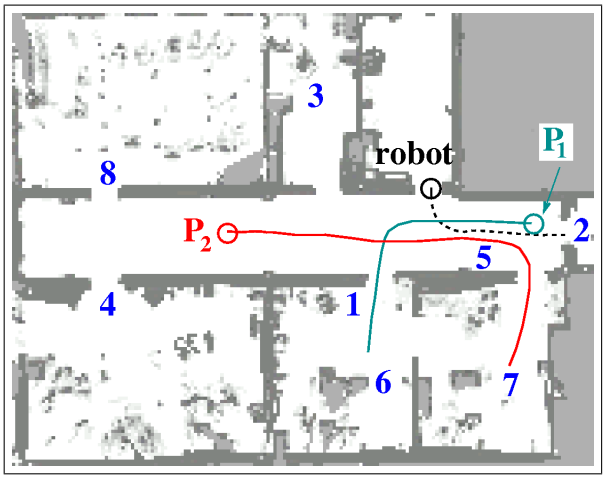


Figure 4.9: Albert observes two persons and waits in the doorway until both have passed it. The dashed path corresponds to the intended path of the robot.

After P_1 had passed the doorway the behavior of the robot was mainly influenced by the trajectory of P_2. Since P_2 continued walking along the corridor Albert waited until also P_2 had passed the doorway. If, in contrast, P_2 had walked into one of the offices before passing the robot or P_2 had moved at a lower speed, Albert would have started moving immediately after P_1 had unblocked the robot's path.

We repeated this experiment with a reactive collision avoidance system that does not predict the motions of people [Stachniss and Burgard, 2002]. The robot directly left the doorway to drive to its designated goal point and blocked the way of person P_1.

4.4.4 A Comparison to Linear Prediction

The following experiment is designed to show that a robot, which takes into account different motion patterns during path planning, performs significantly better than a robot that just relies on a linear prediction of the movements of people.

Consider the situation depicted in the left image of Figure 4.10. The robot was moving along the corridor from left to right to reach its target location, which is labeled C in the figure. At the position labeled with A the robot detected a person approaching it. According to the learned motion patterns the person was most probably walking to resting place 3. The motion patterns which had a sufficiently high probability are depicted in Figure 4.11 (the thicker the trajectory, the higher the probability that the person will follow this specific trajectory). Since the prob-

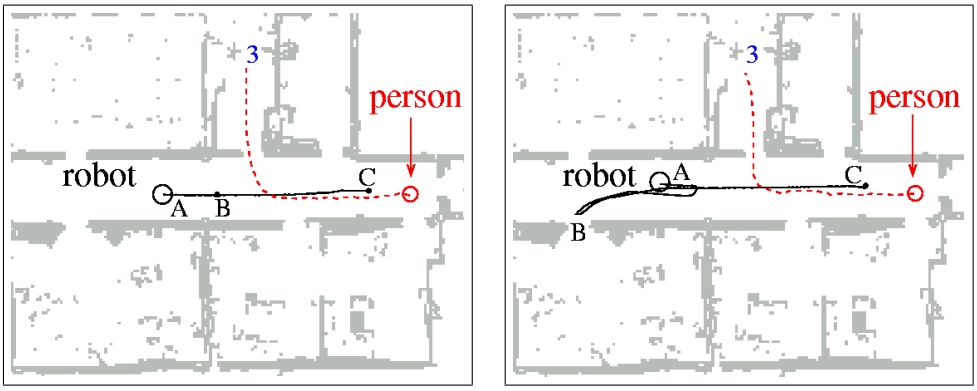


Figure 4.10: In the left image the robot used the motion patterns to predict future poses of people. At the position labeled A the robot observed a person approaching it. Based on the learned motion patterns it inferred that the person will enter the room to the north to go to resting place 3 with very high likelihood (see Figure 4.11). Accordingly, the robot moved forward and waited at the position labeled B until the person left the corridor. In the right image the robot used a linear prediction of the trajectory of the person. It anticipated that it will block the person's way and thus it moved to the position labeled B. After the person left the corridor the robot continued approaching its target location.

abilities of the motion patterns with target locations 4 and 8 were very low, the additional costs introduced to the configuration time-space did not prevent the robot from driving further along the corridor. Thus, the robot moved to the location labeled B in the left image of Figure 4.10 and waited there until the person entered the room in the north. After that it moved to its target location, which is labeled C. Figure 4.12 shows images that were taken during this experiment.

We repeated this experiment with a system that does not use the motion patterns and instead predicts the trajectories of people only linearly. The trajectory of the robot in this experiment is shown in the right image of Figure 4.10. After the robot detected the person it continued to move and simultaneously replanned its path using the configuration time space computed based on a linear prediction of the movements of the person. When it detected that it would block the path of the person it turned around and moved to the location labeled B in the right image of Figure 4.10. After it noticed that the person disappeared the robot continued to its designated target location C.

We performed ten similar experiments for each of the prediction strategies and measured the time needed to complete the navigation task. The average time for both systems is shown in Figure 4.13. As can be seen from the graph, the time can be significantly reduced when taking into account learned motion patterns

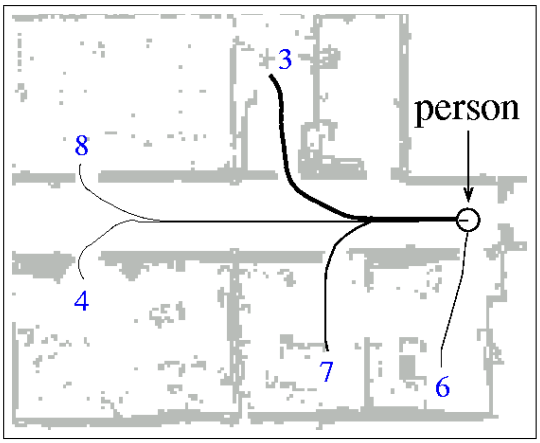


Figure 4.11: Most probable trajectories of the person in the experiment shown in Figure 4.10. The thicker the trajectory, the higher the probability that the person follows this motion pattern.

Figure 4.12: Albert moves forward and waits until the likelihood of interfering with the person is low enough.

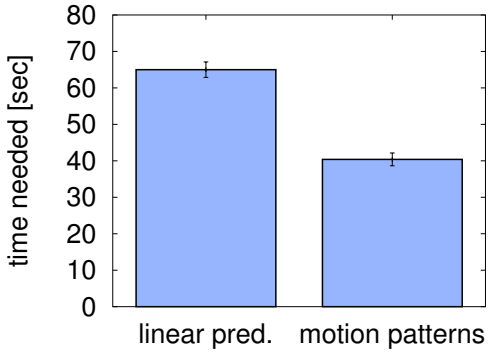


Figure 4.13: Average time needed to complete the navigation task when the robot performs linear prediction (left column) and when it uses the learned motion patterns (right column) for ten experiments.

compared to the approach that only performs a linear prediction.

We also performed this experiment with the reactive collision avoidance system that does not predict the motions of people [Stachniss and Burgard, 2002]. In a situation like the one considered here, the person always had to slow down because the robot was blocking its path.

4.5 Related Work

In the past, different techniques were introduced to adapt the behavior of a robot according to a belief about the future motions of people in its surrounding. For example, the approaches presented by Zhu [1991] and Tadokoro *et al.* [1995] apply Hidden Markov Models [Rabiner and Juang, 1986] to predict the motions of moving obstacles in the vicinity and to choose adequate trajectories for a robot. Jensen *et al.* [2003] defined a probabilistic navigation function that takes into account possible future motions of people for which they define a probabilistic motion model. Arkin *et al.* [1993] present a behavior-based technique to avoid moving obstacles. Their approach computes a collisions zone and repulsive vectors which are used to adapt the trajectory of the robot. Thereby, they assume that the robot and the moving obstacles maintain their current velocities. Since those approaches do not learn typical motion patterns, they can only predict short-term motions and not complete trajectories. The system described by Kruse [1998] uses cameras to track people and to learn where people typically walk in the environment. A collision probability field similar to a potential field is computed which incorporates the average motion behavior of the people. Accordingly, the

path of the robot is determined so that the likelihood of conflicts is reduced. In contrast to our system, their goal is *not* predict the motions of observed people and to actively adapt the trajectory of the robot during motion. When the path of the robot is blocked the robot stops and replanning is invoked, treating the unforeseen obstacle as a static obstacle. Foka and Trahanias [2002] proposed to use manually defined "hot points" which people might be interested in approaching to predict their movements. Depending on the direction of the movements of the people they compute probabilities for the hot points. These probabilities are then integrated into the reward function of an POMDP (Partially Observable Markov Decision Process [Kaelbling *et al.*, 1998]) which is used to compute the robot's path. Illmann *et al.* [2002] presented an approach to acquire basic motion patterns like straight motion, wandering aimlessly, or entering a queue. Their goal is to predict short-term motions of surrounding people using the learned motion patterns so that a mobile robot can chose adequate behaviors. Riley and Veloso [2002a] try to predict the behavior of opponent agents in the RoboCup domain using given models. Depending on the belief about the strategy of the opponents they create new setplays for their own team by performing a hill-climbing search in their plan space. However, since they do not have monitoring actions they are not able to react to unforeseen events. They only create new setplays for the robots when the game is stopped. Kasper *et al.* [2001] presented an approach to improve the behavior of a robot by following the activities of a teacher. Lavalle *et al.* [1997] developed a system which is able to keep track of a moving target even in the case of possible occlusions by static obstacles in the environment. González-Baños *et al.* [2002] presented a modified version that works without a prior map of the environment. A major difference between the latter approaches and our technique lies in the different evaluation functions. Whereas Kasper *et al.* seek to optimize the navigation skills, Lavalle *et al.* and González-Baños *et al.* generate actions to maximize the probability of future visibility of a moving object even in the presence of obstacles. The approach presented here, in contrast, has the goal to minimize the risk of interfering with people given knowledge about their typical motion patterns. In the work of Rosencrantz *et al.* [2003] a team of robots tries to locate and tag "enemies" which are not always in their perceptual field of view. The movements for the observer robots are coordinated so that the information gain is maximized and the search time minimized. Feyrer and Zell [2000] use an artificial potential field to pursue people in real-time using a combination of vision and laser data while avoiding collisions with obstacles. Furthermore, some multi-robot systems have been developed that keep track of multiple moving objects [Parker, 1997, Murrieta-Cid *et al.*, 2002, Jung and Sukhatme, 2002] or surround a moving target [Pirjanian and Matarić, 2000]. Those systems also focus on the generation of appropriate actions in order to be able to keep track of moving targets.

4.6 Conclusion

In this chapter we presented a method for adapting motion strategies of a mobile robot according to the activities of people in its vicinity. Our approach uses the motion patterns learned from real data which were introduced in the previous chapter. These patterns are utilized to predict the motions of people sensed by the robot. To compute cost-optimal paths that minimize the risk of interfering with a person we consider the configuration time-space of the robot. The integration of the estimated motion patterns of the people into the path planning allows the robot to adapt its behavior more appropriately and at an earlier stage than purely reactive approaches.

Our technique has been implemented and applied to data recorded by our mobile robot equipped with a laser-range sensor. The current implementation is highly efficient and allows the robot to quickly react to its sensory input. In several experiments we demonstrated that the behavior of a mobile robot can appropriately be adapted by predicting the motions of people in the vicinity. The experiments furthermore illustrated advantages over standard reactive systems and over systems that do not take into account the motion patterns and instead predict the trajectories of people only linearly.

Chapter 5

Using Hidden Markov Models to Estimate the Positions of Multiple Persons

5.1 Introduction

Mobile robots that provide service to people can carry out their tasks more efficiently if they know where the people are. In this chapter we investigate the problem of how a robot can effectively maintain a probabilistic belief about the positions of the people in its environment. The robot is equipped with knowledge about typical motion behaviors of people in form of Hidden Markov Models (HMMs) [Rabiner and Juang, 1986], which are derived from given motion patterns. The motion patterns which are learned using the technique presented in Chapter 3 can be regarded as the "input" to the derivation process presented this chapter. We do not learn the structure and parameters of the HMMs used here. Instead, they are automatically derived from the learned motion patterns. The HMMs are updated based on vision and laser information. By incorporating vision data the robot is able to distinguish between different persons. Thus, different motion patterns can be used for the individual persons which move through the environment. In contrast to the approach presented in Chapter 4, using the HMMs the robot not only is able to estimate the positions of people and infer their future movements when they are in its field of view. The robot maintains a belief about the positions of the people at *any* point in time. It can robustly estimate the positions of multiple persons even if they are currently outside its field of view. While the robot is carrying out its task it applies a decision-theoretic approach to actively select points in the environment that are expected to provide information about the positions of people.

This chapter is organized as follows. In Section 5.2 we first introduce Markov Chains and Hidden Markov Models. Section 5.3 contains a description of how we derive HMMs from the learned motion patterns to predict motions of people. In Section 5.4 we explain our technique to detect and identify people using sensor data. Section 5.5 introduces our strategy how to decide if and which observation action should be executed to update the robot's belief. Section 5.6 describes experimental results illustrating the robustness of our approach to maintain an accurate belief about the positions of people using laser and vision data with a mobile robot. Finally, Section 5.7 contains a discussion of related work.

5.2 Markov Chains and Hidden Markov Models

In this section we first explain the key concepts of Markov Chains and then introduce Hidden Markov Models, which are an extension of Markov Chains.

5.2.1 Markov Chains

A Markov chain [Cox and Miller, 1965] is a finite state machine and is used to model the behavior of a system. The parameters of this model are the initial state distribution and the probabilities for each state transition. Let ξ^t denote the state of the system at time step t. ξ^t is a random variable ranging over the set of possible states $\rho = \{\rho_1, \ldots, \rho_N\}$. The initial state distribution indicates for each $\rho_i \in \rho$ the probability that the systems starts in this particular state, i.e., it specifies the probability $P(\xi^1 = \rho_i)$ for all ρ_i. The transition probabilities of a Markov chain indicate which state follows which other state, i.e., they specify for each pair ρ_j and ρ_i the probability $P(\xi^{t+1} = \rho_j | \xi^t = \rho_i)$ that the next state is ρ_j given that the current state is ρ_i.

For any sequence of states $\xi = \{\xi^1, \xi^2, \ldots, \xi^T\}$ we can write the probability of ξ as:

$$\begin{aligned} P(\xi) &= P(\xi^T, \ldots, \xi^1) \\ &= P(\xi^T | \xi^{T-1}, \ldots, \xi^1) \cdot \ldots \cdot P(\xi^2 | \xi^1) \cdot P(\xi^1), \end{aligned} \tag{5.1}$$

by applying the product rule (see Appendix A.2.2) several times. The key property of a Markov chain is that the probability of the value of the successor state ξ^{t+1} depends only on the value of the current state ξ^t. The history of the states of the system does not add any new information. Accordingly, the following holds:

$$P(\xi^{t+1} | \xi^t, \ldots, \xi^1) = P(\xi^{t+1} | \xi^t). \tag{5.2}$$

Applying this to Eq. (5.1) yields:

$$\begin{aligned} P(\xi) &= P(\xi^T, \ldots, \xi^1) \\ &= P(\xi^T|\xi^{T-1}) \cdot P(\xi^{T-1}|\xi^{T-2}) \cdot \ldots \cdot P(\xi^2|\xi^1) \cdot P(\xi^1). \end{aligned} \tag{5.3}$$

The property that only the current state gives information about the future behavior of the system is called the *Markov property*. Thus, in applications in which the Markov property is satisfied, knowledge about the history of the states of the system does not add any new information and does not help to predict future states. In our application domain the Markov property is generally *not* satisfied.

To illustrate this, see the situation depicted in the left image of Figure 5.1. Suppose the person starts moving at location A. When it reaches location B one cannot predict which direction the person will go assuming that the next state (the next position of the person) only depends on the current state. However, if we know that the person started at location A and given we have information about typical motion patterns of people (see center image of Figure 5.1), we can infer that the person will go to the right as depicted in the right image of Figure 5.1[1].

We model the fact that the behavior of the person varies depending on the actual intention the person has by taking into account the learned motion patterns. We encode in each state the position and additionally the motion patterns to which it belongs. As a result, our model is able to differentiate between various motion patterns the person might follow and automatically chooses the correct transitions. Using this representation of a person's state the Markov property is satisfied.

5.2.2 Hidden Markov Models (HMMs)

When using a Markov chain to model the behavior of a system it is assumed that there is a one-to-one correspondence between states and observations. This is no longer satisfied for Hidden Markov Models (HMMs) [Rabiner and Juang, 1986]. Thus, when using an HMM to model the behavior of a system, it is assumed that it is not possible to tell what state the system is in when an observation is made.

We have to use an HMM since the current states of the people are not directly observable. The robot's sensors are generally not working perfectly and we cannot completely be sure about the position of a person when sensor information is provided. Thus, we have to compute a probability distribution over the possible positions given an observation.

In addition to the initial probability distribution and the state transition model, the observation probabilities are further parameters which have to be specified when using an HMM. For each state of the system a probability distribution

[1]Note that such observations that the incorporation of higher level knowledge improves the prediction performance have also been reported by Bui *et al.* [2001] and Murphy [2002].

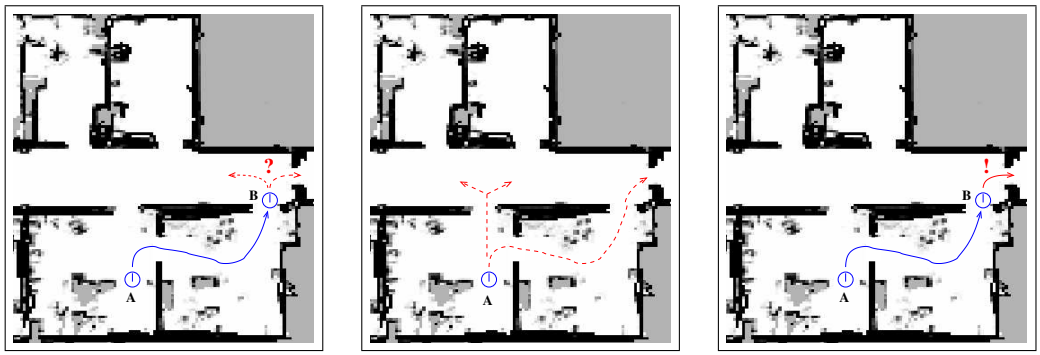


Figure 5.1: If one assumes that the next state only depends on the current state the next position of the person cannot be predicted when it is at location B (left image). However, if we know that the person started at location A and if we take into account the different motion patterns (see center image) than it is easy to conclude that the person will go to the right (right image).

over the possible observations has to be defined. Let z^t denote the observation at time step t. Assuming there are W possible observations we have $z^t \in \{o_1, \ldots, o_W\}$. The observation model specifies for each pair of o_w and ρ_i the probability $P(z^t = o_w | \xi^t = \rho_i)$. To incorporate an observation z^t into an HMM Bayes' Rule is applied:

$$P(\xi^t = \rho_i \mid z^t = o_w) = \eta \cdot P(z^t = o_w \mid \xi^t = \rho_i) \cdot P(\xi^t = \rho_i). \tag{5.4}$$

Here η is a normalizer which ensures that the probabilities some up to 1 over all possible states. Thus, the probability that the system is in state ρ_i at time step t, given the observation o_w, is computed by multiplying the likelihood of the observation o_w given the system is in state ρ_i with the prior probability of the system to be in state ρ_i. Figure 5.2 illustrates the dependencies of the individual components of an HMM.

5.3 Deriving an HMM from Learned Motion Patterns

In Chapter 3 we described our approach to learning motion patterns of people. In the following we show how to derive the transition probabilities of the HMMs from these learned patterns that can be used to predict the motions of people.

As explained before, people usually do not permanently move. Rather they typically move between resting places where stay for a certain period of time.

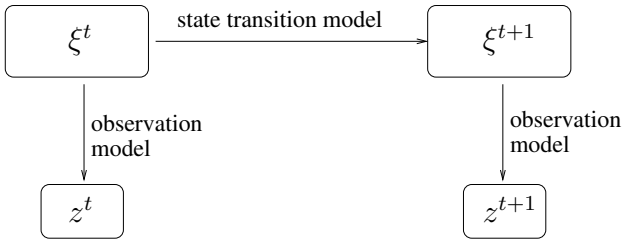


Figure 5.2: Illustration of dependencies in an HMM.

To derive an HMM from the learned motion patterns of a person we therefore distinguish two types of states. The first class of states correspond to the resting places. To connect these states we introduce so-called intermediate states which lie on the trajectories given by the learned motion patterns. In our current system we use a sequence of L_m intermediate states $\rho_m^1, \ldots, \rho_m^{L_m}$ for each motion pattern θ_m. The intermediate states are distributed over θ_m such that the distance between two consecutive states is $\Delta_\rho = 50cm$. Given this equidistant distribution of the intermediate states and assuming a constant speed v with standard deviation σ_v of the person, the transition probabilities of this HMM depend on the length Δ_t of the time interval between consecutive updates of the HMM as well as on v and σ_v. In our current system this value is set to $\Delta_t = 0.5sec$. Accordingly, we compute the probability $P(\rho'_m|\rho_m, \Delta_t)$ that the person is in state ρ'_m after ρ_m in the direction of the motion pattern given its current state is ρ_m and given that the time Δ_t has elapsed as:

$$P(\rho'_m|\rho_m, \Delta_t) = \int_{\rho'_m - \frac{\Delta_\rho}{2}}^{\rho'_m + \frac{\Delta_\rho}{2}} \mathcal{N}(\rho_m + v \cdot \Delta_t, \sigma_v, \rho)\, d\rho. \tag{5.5}$$

Here $\mathcal{N}(\rho_m + v \cdot \Delta_t, \sigma_v, \rho)$ is the quantity obtained by evaluating the Gaussian with mean $\rho_m + v \cdot \Delta_t$ and standard deviation σ_v at ρ.

The transition probabilities for the resting places are computed based on a statistics about the average time period that elapses until the person starts to move on a particular trajectory after arriving at the corresponding resting place.

5.4 Person Detection and Identification

In Chapter 4 we introduced the Kalman filter approach to track the positions of people. One obvious drawback of this approach is that it can only represent Gaussian (unimodal) probability distributions. In several situations it can be more appropriate to represent the belief about the position of a person by a multimodal

distribution. This can be the case if, for example, the person is not in the field of view anymore. Thus, one advantage of using HMMs is that they can represent such multimodal beliefs. Furthermore, by using HMMs one is able to utilize even negative information to update the belief. Such negative information is, for example, obtained in situations in which the robot does not detect a specific person. In these situations the robot can update the states *in* its field of view as well as the states *outside* its field of view with this observation. Accordingly, the probabilities of the states outside the field of view will increase whereas the probabilities of the states in the field of view will decrease. In the following we describe how we use HMMs to maintain a belief about the positions of multiple persons.

To keep track of multiple persons in an environment, one in principle would have to maintain a belief over the joint state space of all persons. This approach, however, is usually not feasible since the complexity of the state estimation problem grows exponentially with the number of persons. Additionally, learning the joint transition probability distribution would require a huge amount of training data. Therefore, we approximate the posterior by factorizing the belief over the joint state space and consider independent beliefs over the states of all persons. Thus, we use an individual HMM to represent the belief about the position of each person. To maintain the individual beliefs we need to be able to update the HMMs for the persons based on observations made by the robot, which requires the ability to reliably detect people and to identify them. To achieve this, our current systems combines laser and vision information.

To detect people in the laser-range scans obtained with the robot our system extracts features which are local minima that correspond to the people's legs. We also need to be able to identify a person in order to appropriately update the belief about the location of that person. To achieve this we employ the vision system of our robot and learn an image database beforehand. For each person this database contains one histogram which is built from 20 images. To identify a person, we proceed as follows. Every time the laser-based people detection system reports a feature in the field of view of the camera, an image is collected and the following three steps are applied:

1. *Segmentation:* We extract a rectangular area containing the person from the image. To determine the area in the image corresponding to a feature detected by the laser-based people detection system, we rely on an accurate calibration between the camera and the laser. We use a perspective projection to map the 3D position of the person in world coordinates to 2D image coordinates.

2. *Color histograms:* We compute a color histogram for the area selected in the previous step. Whereas color histograms are robust with respect to transla-

Figure 5.3: Typical scene with two persons walking along the corridor (left image) and corresponding estimate of the laser-based people detection system (right image).

tion, rotation, scale and to any kind of geometric distortions they are sensitive to varying lighting conditions. To handle this problem we consider the HSV (Hue-Saturation-Value) color space. In this color model the intensity factor can be separated so that its influence is reduced. In our current system we ignore this factor and consider only the hue and saturation values. As a result our system shows a quite robust behavior even under varying lighting conditions.

3. *Database matching:* To determine the likelihood that the area extracted in the segmentation step contains a particular person, we compare the histogram computed in step 2 to all prototypes existing in the database. As a measure of similarity between a query histogram q with a prototype π in the database we use the normalized intersection norm [Swain and Ballard, 1991]. This quantity is computed as:

$$H(q,\pi) \quad = \quad \frac{\sum_{b=1}^{B} \min(q_b, \pi_b)}{\sum_{b=1}^{B} \pi_b}, \tag{5.6}$$

where q and π are color histograms both having B bins. One advantage of this norm is that it also allows to compare partial views, i.e.,when the person is close to the camera and only a part of it is visible.

As an application example consider the situation depicted in the left image of Figure 5.3. In this particular situation two persons (person B and person C) were walking along the corridor within the perceptual field of the robot. The right image of Figure 5.3 shows the estimate of the laser-based people detection system at the same point in time. The corresponding image obtained with the robot's camera is shown in the left image of Figure 5.4. The two segments of

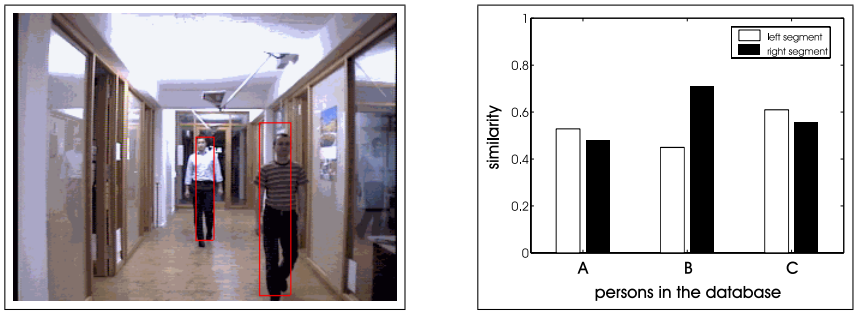


Figure 5.4: Segmentation of the two persons from the image grabbed with the camera of the robot (left image) and the similarity between the color histograms (see Figure 5.5) of the extracted segments and the data base prototypes (right image).

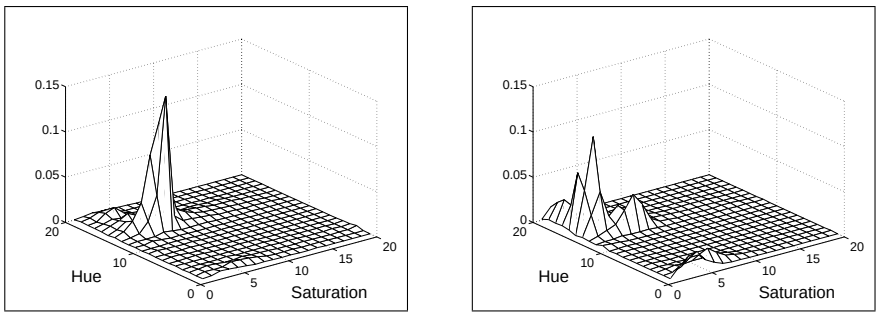


Figure 5.5: Corresponding color histograms for the left and right segment shown in Figure 5.4 (left image).

the image that correspond to the two features detected by the laser-based people detection system are also shown in this image. Figure 5.5 depicts the resulting color histograms of the two extracted segments. The right image of Figure 5.4 plots the similarities between these histograms and the three prototypes stored in the data base.

Since we consider independent beliefs over the states of the persons we have to determine which feature is caused by which person and we have to update each HMM depending on the likelihood that the corresponding person has been observed. For that purpose we apply Joint Probabilistic Data Association Filters (JPDAFs) [Cox, 1993] which are introduced in the following.

Let $\xi^t = \{\xi_1^t, \ldots, \xi_R^t\}$ denote the state of the R persons we are tracking at time t. Each ξ_r^t is a random variable ranging over the state space of a single person. A measurement at time t is denoted as $z^t = \{z_1^t, \ldots, z_{S^t}^t\}$. Here, S^t is the

number of features detected at time t. In our case, each z_s^t is the position of a feature provided by the laser-based people detector together with the corresponding similarity values provided by the vision system. If the detected feature is not in the field of view of the camera we assume that the similarity values are uniformly distributed over all data base images.

To incorporate observations $z^{(1:t)}$ into the HMM of person r we apply the recursive Bayesian update scheme (see also Eq. (5.4)):

$$P(\xi_r^t \mid z^{(1:t)}) = \eta \; P(z^t \mid \xi_r^t) \; P(\xi_r^t \mid z^{(1:t-1)}). \tag{5.7}$$

Here, η is a normalization factor and $z^{(1:t)}$ denotes the sequence of all measurements up to time t.

Since we do not know which of the features in z^t is caused by person r, we follow the idea of Joint Probabilistic Data Association Filters and integrate the single features according to the assignment probability λ_{sr} that feature z_s^t corresponds to person r:

$$P(\xi_r^t \mid z^{(1:t)}) \;\; = \;\; \eta \sum_{s=0}^{S^t} \lambda_{sr} P(z_s^t \mid \xi_r^t) \; P(\xi_r^t \mid z^{(1:t-1)}). \tag{5.8}$$

To compute λ_{sr} one considers so-called joint association events. Each such event ψ is a set of pairs $(s, r) \in \{0, \ldots, S^t\} \times \{1, \ldots, R\}$ that determines which feature is assigned to which person. Note that the feature z_0^t is used here to model situations in which a person r has not been detected, i.e., no feature has been obtained for r. This is represented in the association event as $(0, r)$.

The set of all joint association events which assign feature z_s^t to person r is denoted as Ψ_{sr}. At time t the JPDAF computes the posterior probability that feature z_s^t is caused by person r according to [Cox, 1993]:

$$\lambda_{sr} \;\; = \;\; \sum_{\psi \in \Psi_{sr}} P(\psi \mid z^{(1:t)}). \tag{5.9}$$

As derived in Appendix A.4 the term $P(\psi \mid z^{(1:t)})$ can be approximated by:

$$P(\psi \mid z^{(1:t)}) \approx \eta' \int P(z^t \mid \psi, \xi^t) \; P(\psi \mid \xi^t) \; P(\xi^t \mid z^{(1:t-1)}) \; d\xi^t. \tag{5.10}$$

Here, η' is a normalizer ensuring that $P(\psi \mid z^{(1:t)})$ sums up to one over all ψ. Under the assumption that all assignments have the same likelihood, $P(\psi \mid \xi^t)$ can be approximated by a constant.

The term $P(z^t \mid \psi, \xi^t)$ denotes the likelihood of making an observation given the state of the persons and a specific assignment ψ between the features and the

persons. In order to determine this quantity, we have to consider false alarms which occur if a feature is not caused by any of the persons that are tracked using the HMMs. Let γ denote the probability that an observed feature is a false alarm. The number of false alarms contained in an association event ψ is given by $\phi = S^t - A$ where $A = R - \|\{(0,\cdot) \in \psi\}\|$ is the number of persons to which features have been assigned. Thus, γ^ϕ is the probability assigned to all false alarms in z^t given ψ. If we assume that the features are detected independently of each other, the term $P(z^t \mid \psi, \xi^t)$ can be computed as the product over all association pairs:

$$P(z^t \mid \psi, \xi^t) \;=\; \gamma^\phi \prod_{(j,i)\in\psi} P(z_j^t \mid \xi_i^t). \tag{5.11}$$

By inserting Eq. (5.11) into Eq. (5.10), assuming that $P(\psi \mid \xi^t)$ is constant, and after inserting the result into Eq. (5.9) we obtain the following equation for the probability that feature z_s^t is caused by person r:

$$\lambda_{sr} = \sum_{\psi\in\Psi_{sr}} \left[\eta' \, \gamma^\phi \int \prod_{(j,i)\in\psi} P(z_j^t \mid \xi_i^t) \; P(\xi^t \mid z^{(1:t-1)}) d\xi^t \right]. \tag{5.12}$$

This term can be rearranged to (see Appendix A.5):

$$\lambda_{sr} = \sum_{\psi\in\Psi_{sr}} \left[\eta' \, \gamma^\phi \prod_{(j,i)\in\psi} \int_{\xi_i^t} P(z_j^t \mid \xi_i^t) \; P(\xi_i^t \mid z^{(1:t-1)}) d\xi_i^t \right]. \tag{5.13}$$

Since we use HMMs to represent the belief about the states of the persons the integration in Eq. (5.13) corresponds to summing over all states of the HMM for the particular person.

It remains to describe how the term $P(z_j^t \mid \xi_i^t)$ is computed. As explained above, each z_j^t consists of the position y_j^t of the feature j at time t and the similarity measure $H(q_j^t, \pi_i)$ between the query histogram q_j^t of the corresponding segment in the camera image and the database histogram of person i. In our current system we use the following approximation to compute the likelihood $P(y_j^t, H(q_j^t, \pi_i) \mid \xi_i^t)$, which has turned out to yield satisfactory results in practice:

$$P(z_j^t \mid \xi_i^t) = P(y_j^t, H(q_j^t, \pi_i) \mid \xi_i^t) \approx H(q_j^t, \pi_i) \; P(y_j^t \mid \xi_i^t). \tag{5.14}$$

Here, $P(y_j^t \mid \xi_i^t)$ is the probability that the laser-based people detection system reports a feature detection at location y_j^t given that the person is in state ξ_i^t. We compute this quantity using a mixture of a uniform distribution and a bounded Gaussian with mean y_j^t. Note that we also take into account visibility constraints, i.e., states that are occluded are regarded as states outside the bounded Gaussian.

In the case that no feature has been obtained for a person we use the likelihood of false negative observations for such states that are within the range of the robot's sensors. For all other states we use the average likelihood that the robot does not detect a person given it is outside the robot's sensor range.

5.5 Active Localization of People

As the experiments in Section 5.6.1 will show using the derived HMMs and the presented observation model the robot is able to track the positions of multiple persons. The robot is even able to reliably maintain a belief when the people leave its field of view. However, since the robot becomes uncertain about the position of a person when it has not been observing the person for a longer period of time, it should consider to actively perform observation actions to update its belief. An observation action corresponds to moving to a place in the environment and obtaining a sensor measurement there. In this context two aspects are relevant. On one hand, the information gain should be as large as possible, and on the other hand, the cost of performing observation actions should be minimized.

To determine the uncertainty in the belief about the positions of the people we consider the entropy of the posteriors which is a general measure for the uncertainty. The entropy $\mathcal{H}$ of the posterior Bel_r about possible states $\rho_1, \ldots, \rho_{N_r}$ of person r is defined as:

$$\mathcal{H}(Bel_r) \;=\; -\sum_{i=1}^{N_r} Bel_r(\rho_i) \cdot \log Bel_r(\rho_i). \tag{5.15}$$

$\mathcal{H}$ is maximal in case of a uniform distribution. The minimal value zero is obtained if the robot is absolutely certain about the current position of person r.

To take into account the information provided by the sensors of the robot, we compute the expected information gain which is the expected change of entropy given that the robot obtains sensor information. The information gain I for the posterior Bel_r given an observation z is defined as:

$$I(Bel_r \mid z) \;=\; \mathcal{H}(Bel_r) - \mathcal{H}(Bel_r \mid z). \tag{5.16}$$

Here, $\mathcal{H}(Bel_r|z)$ is the entropy of the posterior about the position of person r after integrating the observation z.

Note that the problem considered here – choosing the optimal action sequence – can be regarded as a partially observable Markov decision process (POMDP) problem (see the article by Kaelbling *et al.* [1998] for a comprehensive overview). Since solving the POMDP for applications of the size considered here is not feasible in practice we follow an approach that makes several simplifying assumptions

and has been applied successfully for a similar problem in the past [Fox *et al.*, 1998]. First, we use a restricted set of potential observation actions that the robot can carry out. Since we assume that the people stay at the resting places most of the time we only consider observation actions at viewpoints for the resting places. Furthermore, we consider only one observation per task and do not consider all potential measurements perceived by the robot while it is carrying out its task.

Additionally, we take the possibility into account that the robot can observe a resting place whenever it arrives at the final location of its current task. The case that no part of the HMM can be observed after the robot arrived at its goal can be regarded as a special case of this.

Since we do not know what the robot will perceive when it has executed its task a, we have to sum over all possible observations z_a to compute the expected information gain for Bel_r:

$$E(Bel_r|a) \;=\; \sum_{z_a} P(z_a \mid Bel_r) \cdot I(Bel_r \mid z_a). \tag{5.17}$$

To efficiently compute the likelihood $P(z_a \mid Bel_r)$ we do not integrate over all possible measurements. Instead, we consider abstract observations namely that the robot identifies/does not identify the person given it is at the observed resting place or not. The corresponding likelihoods are the average detection respectively failure rates. The overall expected information gain $E(a)$ for the task a is given by the sum of the individual expected information gains for the posteriors of all R persons after executing a:

$$E(a) \;=\; \sum_{r=1}^{R} E(Bel_r|a). \tag{5.18}$$

The expected utility of a can now be defined as:

$$EU(a) = reward(a) - cost(a) + E(a). \tag{5.19}$$

Here, $reward(a)$ specifies a reward function which depends on the utility of finishing a and $cost(a)$ are the cost of executing a.

During the execution of its current task, the robot considers additional observations of resting places. To reduce complexity we compute viewpoints for the resting places. From these viewpoints the robot can observe if a person is currently staying at the corresponding resting place.

To compute the viewpoints we proceed as follows: We perform two deterministic value iterations (see for example [Sutton and Barto, 1998]) in the static 2D occupancy grid map of the environment: The start of first value iteration is the target location of the current task and the start of the second value iteration is

the robot's current position. For each resting place l we compute a visibility area, i.e. the set of grid cells from which l is visible. The view-point for l is then defined as the cell in its visibility area which has the lowest move cost. To determine the move cost for a cell $\langle x, y \rangle$ we can simply add the costs for reaching $\langle x, y \rangle$ from the start location and for reaching the final location from $\langle x, y \rangle$.

The action of moving to the viewpoint corresponding to a resting place l to get an observation is referred to as a_1 and the action of driving from l to the original target location is referred to as a_2. Let $a_l = a_1 \oplus a_2$ be the whole action consisting of a_2 executed after a_1. The expected utility of a_l is given by:

$$EU(a_l) = reward(a_1 \oplus a_2) - cost(a_1 \oplus a_2) + E(a_1 \oplus a_2). \tag{5.20}$$

Note that both a_1 and a_2 depend on the resting place l. However, to enhance readability we omitted the argument l. The expected information gain for $a_1 \oplus a_2$ can be computed as:

$$E(a_1 \oplus a_2) = \sum_{r=1}^{R} \sum_{z_{a_1}} \sum_{z_{a_2}} P(z_{a_1} \mid Bel_r) \cdot P(z_{a_2} \mid Bel_r) \cdot I(Bel_r \mid z_{a_1}, z_{a_2}). \tag{5.21}$$

During the execution of its current task a the robot moves to the viewpoint corresponding to resting place l^* with

$$l^* = \operatorname*{argmax}_{l \in \{l_1, \ldots, l_N\}} EU(a_l) \tag{5.22}$$

whenever $EU(a_{l^*}) > EU(a)$. Here, $l_1, \ldots, l_N$ are the resting places.

5.6 Experimental Results

To analyze the applicability of the HMMs for the prediction of the locations of a person we performed several experiments with our B21r robot Albert (see Appendix A.1) in our office environment. The Hidden Markov Model used to carry out these experiments was computed based on data recorded in our office environment. During the acquisition phase the average speed of the person was $v = 107 cm/sec$ with a standard deviation of $\sigma_v = 25 cm/sec$. The possible transitions of the Hidden Markov Model that was derived from the learned motion patterns in the office environment is shown in Figure 5.6. Whereas the numbered squares indicate the eight resting places, the small circles on the trajectories are the intermediate states.

The experiments described in this section are designed to illustrate that our approach can be used to maintain a robust belief about the positions of multiple

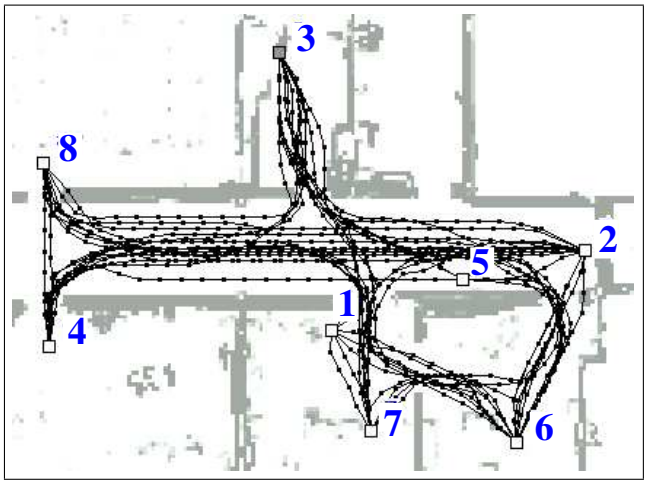


Figure 5.6: Possible transitions of the Hidden Markov Model derived from learned motion patterns. The numbered squares indicate the eight resting places. The small circles on the trajectories correspond to the intermediate states.

persons. We furthermore demonstrate that by using our decision-theoretic approach a mobile robot can choose appropriate observation actions that reduce its uncertainty.

5.6.1 Tracking People

The first experiments are designed to illustrate that our approach enables a robot to reliably estimate the positions of people in its environment, even when they leave its field of view.

Tracking a Single Person

In the first experiment, a single person was moving in our department and the task of the robot, was to estimate the positions of this person. Especially, we were interested in the probability that the person stayed at the correct resting place.

Figures 5.7 and 5.8 show a scene overview (left hand side) for a part of an experiment in which the person was walking through the environment. The robot could only cover a part of the environment with its sensors but even though it was able to maintain and update the belief about the position of the person at any point in time. The center images of the figures depict the results of the laser-based feature detection system and the images on the right hand side show the corresponding beliefs about the position of the person after integrating the observations. In this case we did not use vision information because we assumed only

one person was moving in the environment. In the images on the right hand side the red dot corresponds to the position of the person provided by the laser-based feature detection system. The size of the squares of the intermediate states of the HMM represents the probability that the person is currently in the corresponding state. Additionally, the resting places are labeled with the probability that the person stays at this particular place.

In the images depicted the first two rows of Figure 5.7 the robot observed the person walking through the corridor. Accordingly, it assigned high likelihood to the states close to the detected feature. After the person entered the room and moved outside the field of view of the robot most of the probability mass "wandered" to resting place 7 (third and forth row of Figure 5.7) which was according to the transition probabilities encoded in the HMM. In the situation shown in the images of the first two rows depicted in Figure 5.8 the robot still had not been perceiving the person. As can be seen, even if the robot was not observing the person it inferred that, according to the transition probabilities, there was a high chance that the person left resting place 7 in direction of resting place 6. Finally, the person entered the corridor again (penultimate row of images). Thus, the robot could update its belief according to the new observations.

Figure 5.9 plots for different resting places the probability that the person stayed at this particular place over time. The red, green, and blue curve correspond to resting places 3, 7, and 6 respectively. Whereas the x-axis represents the individual time steps, the y-axis indicates the probability. The graph also includes the ground truth, which is indicated by the corresponding horizontal line-pattern at the .9 level. The images shown in Figures 5.7 and 5.8 correspond roughly to the time steps 37-57. As can be seen from the figure, the system can reliably determine the location of the person. During this experiment the robot predicted the correct position of the person in 93% of the time.

Estimating the Locations of Multiple Persons

As an application example with multiple persons consider the situation depicted Figure 5.10. In this experiment the database of the robot consisted of three persons. For all three persons we used identical motion patterns and transition probabilities in the HMMs. In the situation described here the robot was initially quite certain that persons A and B were in the room containing resting place 3. Then the robot observed a person leaving the room (see images in the first row of Figure 5.10). The grey circles labeled with names indicate the position provided by the laser-based feature detection system. Since the robot did not get vision information at this particular time step it was uncertain who the person was. Note that we use uniform similarity values for all database images in such situations. The intensity of the circle represents the similarity between the extracted segment

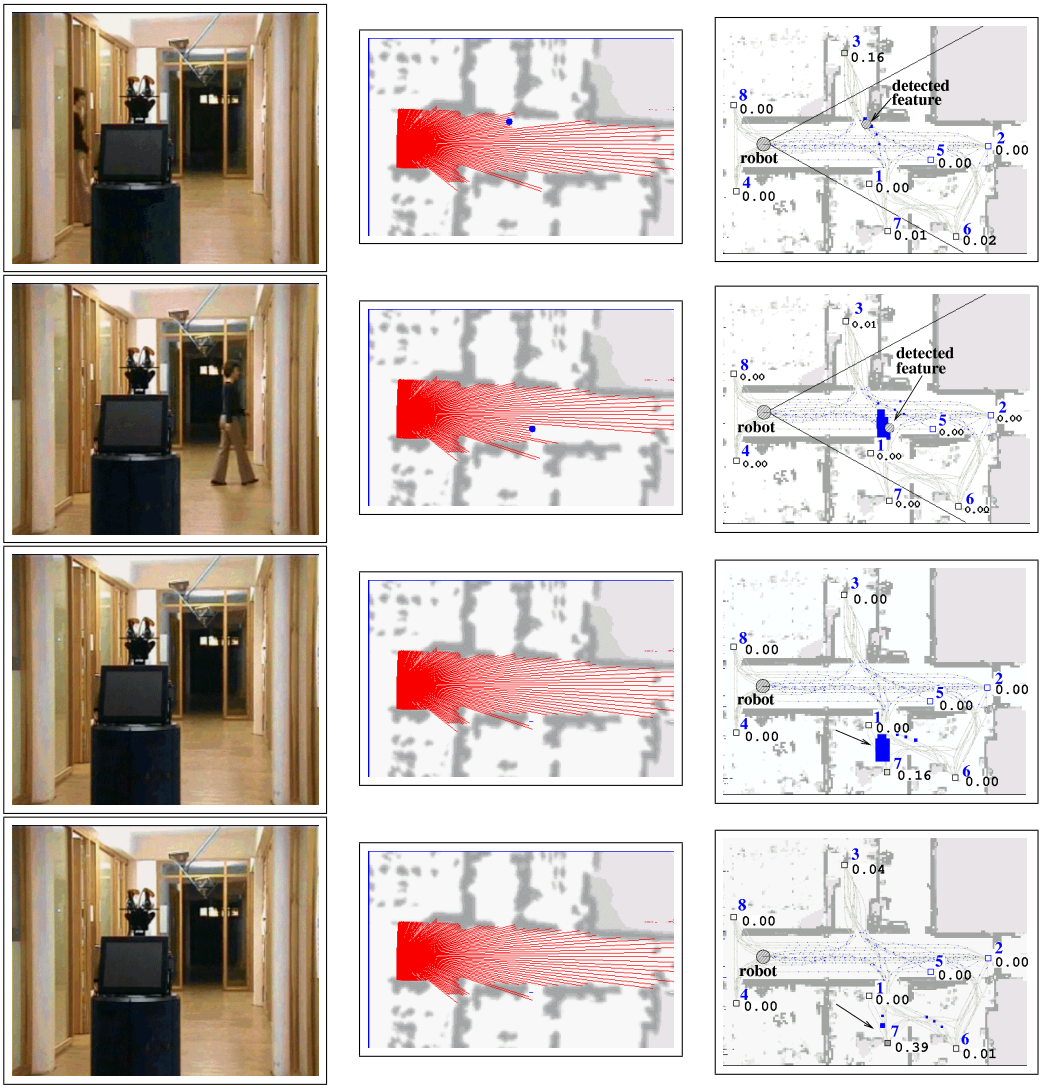


Figure 5.7: Albert tracking a person while she is moving through the environment. The center images depict the results of the laser-based feature detection system. The images on the right hand side show the evolution of the belief over the position of the person after integrating the observations. The size of the squares represents the probability that the person is currently in the corresponding intermediate state and the resting places are also labeled with the probability that the person stays currently there. Even if the robot does not observe the person any more it is able correctly infer that the person is going to resting place 7 (last row). See Figure 5.8 for the continuation of the experiment.

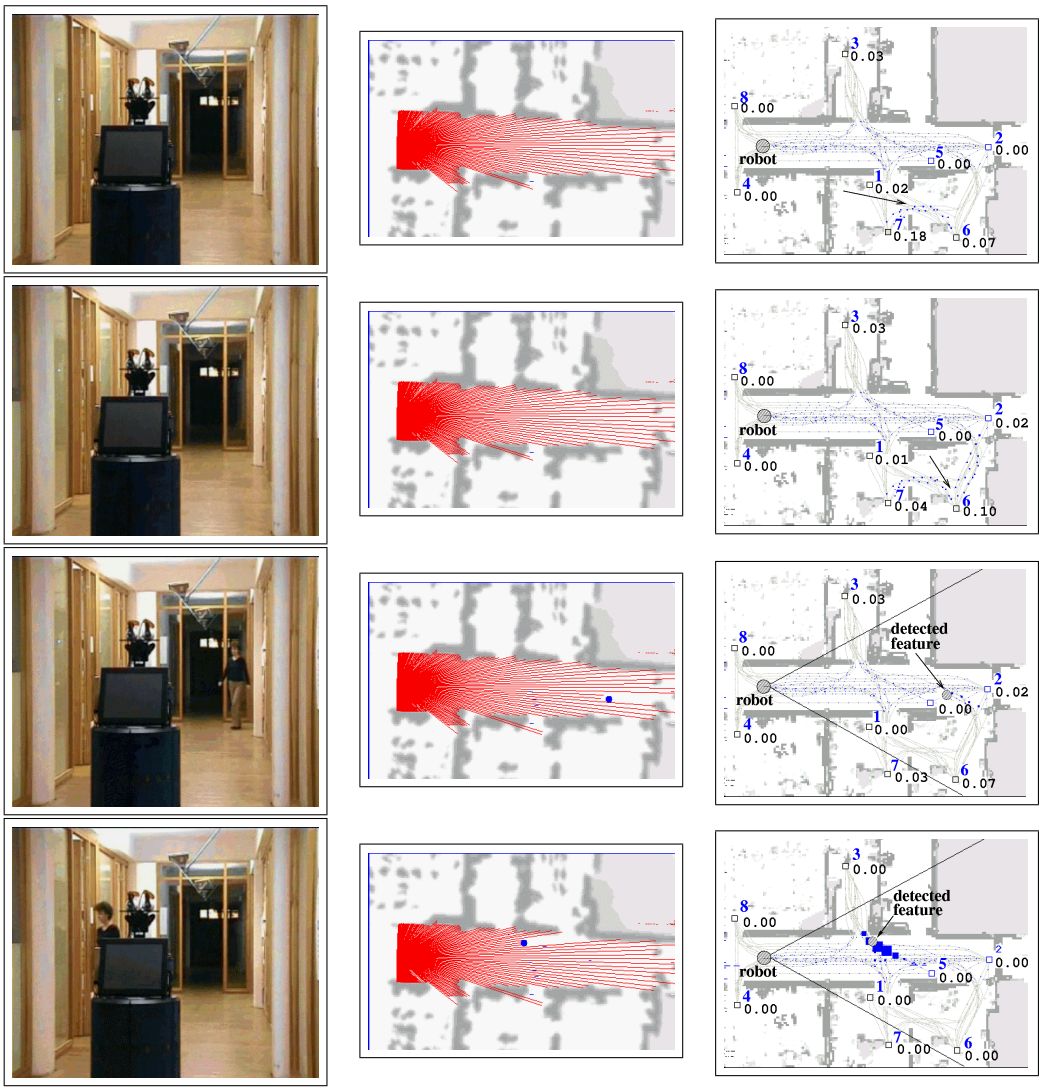


Figure 5.8: Continuation of the experiment shown in Figure 5.7. Even if the robot is not observing the person (situations in the first two rows) it is able to maintain a belief about her position. As soon as the person enters its field of view again the robot updates its belief accordingly (last two rows).

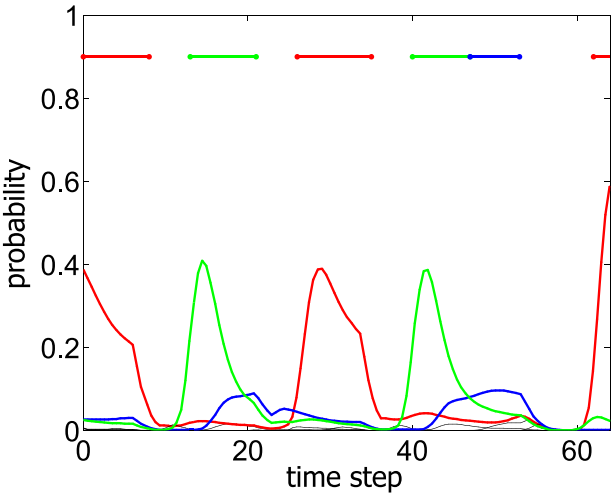


Figure 5.9: Evolution of the probability of the person to be at the different resting places over the time. The ground truth is indicated by the horizontal line-pattern at the .9 level.

and the database images of the person corresponding to the HMM (the darker the more likely). Again, the size of the squares represents the probability that the person is currently in the corresponding intermediate state and the probabilities of the resting places are indicated by numbers. In the images shown in the second row a second person entered the corridor. Now the robot received vision information and updated the individual HMMs according to the data association probabilities computed by the JPDAF. At that time step we had the following information from the vision-based people identification system: The person which had first entered the corridor was person A with a likelihood of .65 and person B with likelihood .31, the second person was person A with a likelihood of .2 and person B with likelihood .76 (note that our database contained a further person). In the situation shown in the images in the third row the robot was still observing the two persons. At that time step we had the following likelihoods: The person which had first entered the corridor was person A with a likelihood of .73 and person B with likelihood .23, the second person was person A with a likelihood of .21 and person B with likelihood .75. During the next time steps both persons left the field of view of the robot but nevertheless the robot was able to maintain an adequate belief about their positions (see images in the fourth and fifth row of the figure).

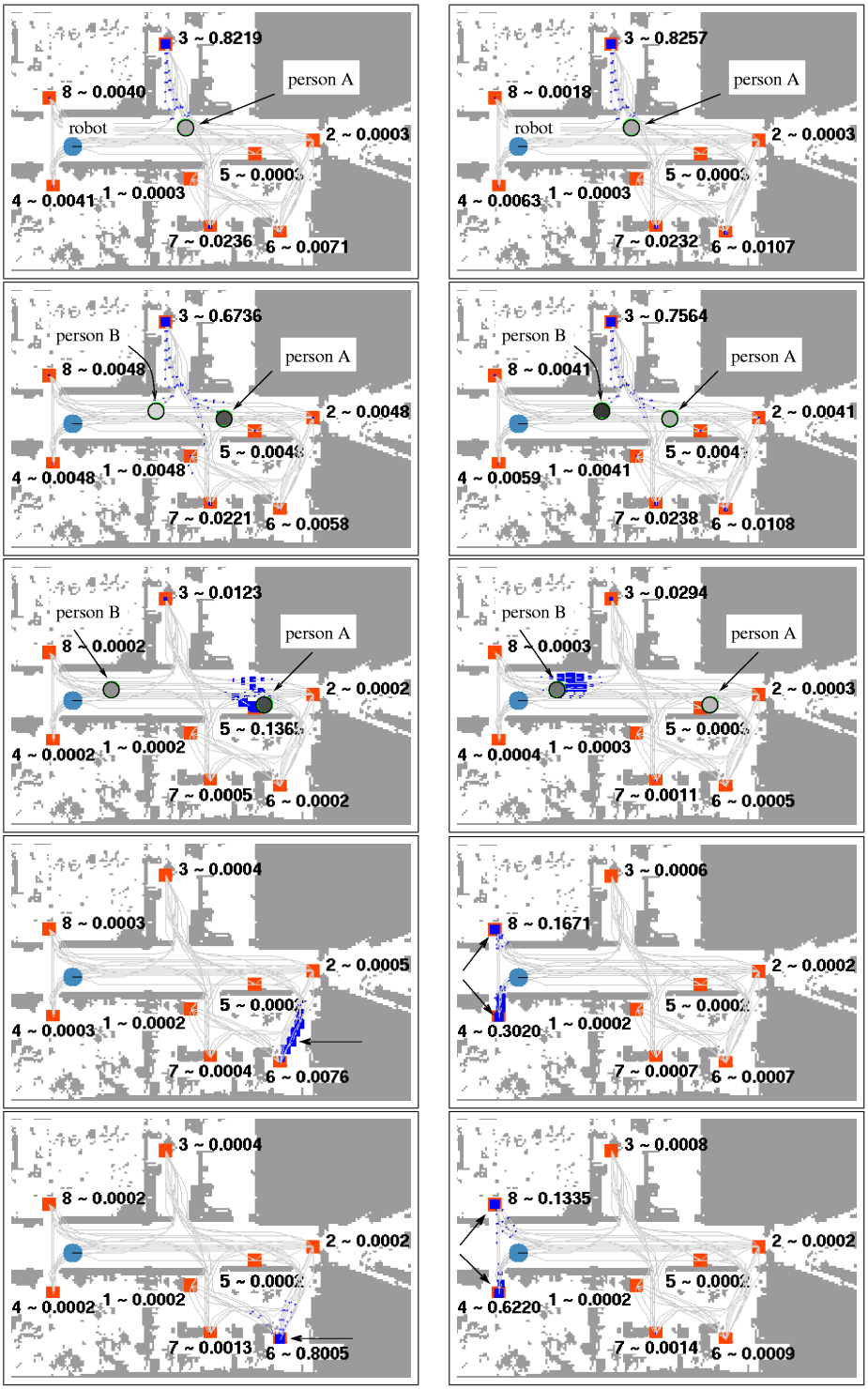


Figure 5.10: This figure shows an experiment with two persons. Whereas the left column depicts the belief about the position of person A the right column shows the belief about the position of person B. The circles labeled with names are detected features. The grey value of each circle represents the similarity to the person corresponding to the HMM.

5.6.2 Deciding to Perform Observation Actions

During the following experiments the robot considered to interrupt its current task to perform an observation actions every 0.5 seconds. In the experiments shown here we assume that $reward(a)$ is equal for all actions a. The results obtained illustrate that our system allows the robot to actively perform observation actions and to use these actions to reduce its uncertainty about the current positions of the people in its environment.

Performing an Observation Action During Task Execution

The goal of the first experiment is to illustrate that our algorithm can effectively guide the robot to viewpoints that provide information about positions of people when needed. The task of the robot was to move from the position marked with $t = 0$ in Figure 5.11 to resting place 4. In the initial situation the viewpoint of resting place 3 had the highest expected utility because the robot was uncertain about the current position of person A (see initial belief in Figure 5.12) and because the additional move cost for viewpoint 3 were very low. Therefore, the robot decided to observe resting place 3 at time step $t = 50$. The robot detected person A and as can be seen in Figure 5.12 updated its belief accordingly. If, in contrast to this, the robot in the same situation does not perform an observation action at viewpoint 3 and moves directly to its target location, its belief about the position of person A would not improve over time. This fact is illustrated in Figure 5.13. In all the figures we only show posteriors of the relevant resting places to enhance readability. In a similar experiment, in which the robot was quite sure that person A was in her office, it did not stop at the viewpoint because the expected utility was not high enough. Note that the posterior probability for being at resting place 7 decreased at time step $t = 20$ (Figures 5.12 and 5.13), because the robot did not observe person A there when it traveled along the corridor and happened to look into the room containing resting place 7.

Actively Searching for a Person

The second experiment has been designed to illustrate that the robot can deal with ambiguities and that it can decrease its uncertainty about the positions of the person by integrating negative information. Here the robot was standing in the middle of the corridor looking to the east (see right image of Figure 5.14) and had currently no task to execute. At around time step $t = 20$ the robot observed a person walking to the east. A scene overview is depicted in the left image of Figure 5.14. According to the camera information the detected person was most likely person B, who had previously been staying at resting place 4. The resulting posterior about the position of person B after integrating a part of the observation

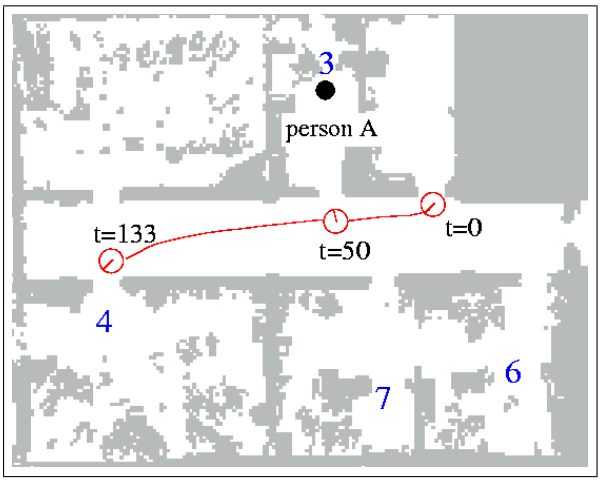


Figure 5.11: The robot decides to stop at the viewpoint to check whether person A is in her room at resting place 3 or not while it is executing a task.

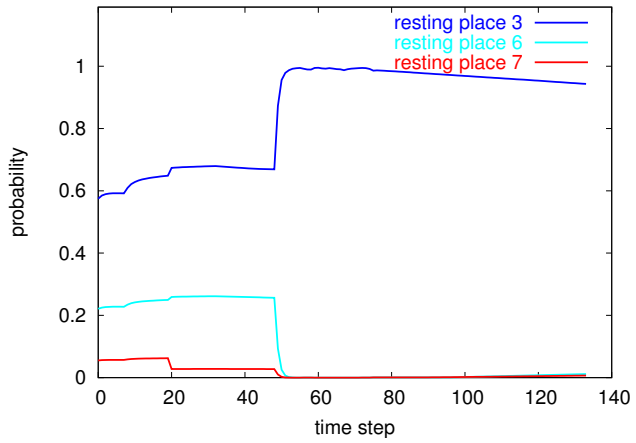


Figure 5.12: Evolution of the probability of person A to be at different resting places over time for the experiment shown in Figure 5.11. As can be seen at time step $t = 50$ when the robot detects person A at resting place 3 it updates its belief accordingly.

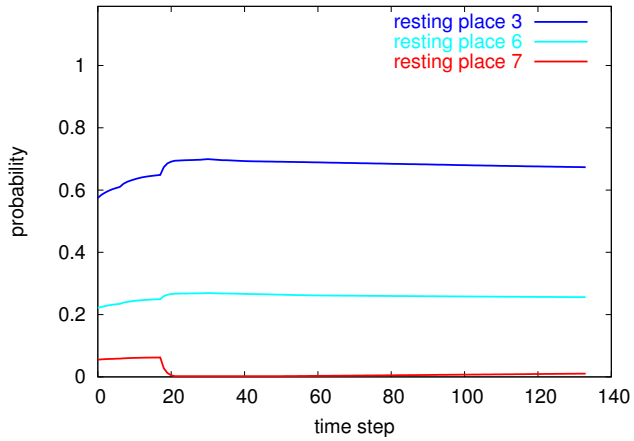


Figure 5.13: Evolution of the probability of person A to be at different resting places over time when the robot does *not* stop at the viewpoint to perform an observation action.

sequence is depicted in Figure 5.15. The circle labeled person B corresponds to the position provided by the laser-based feature detection system. At around time step $t = 30$ person B disappeared out of the field of view of the robot and walked through the office containing resting place 7 to resting place 6. Since the robot was uncertain to which resting place person B was going to, it decided to search for him.

According to the transition probabilities of the resting places, the robot believed that person B most likely walked to resting place 6. Still, the probability of resting place 7 was quite high (see Figure 5.16). Even the probability of resting place 3 increased since the robot did not assume that its sensor information is perfect and since it knew that – according to the HMM – person B sometimes walks to resting place 3 and stays there for a short period of time. Since the viewpoint corresponding to resting place 7 had the highest expected utility (see Figure 5.17), the robot decided to move there to perform the corresponding observation action (see right image of Figure 5.14). In this example, the robot did not observe person B at resting place 7 so that, after the update of the HMM, person B was most likely at resting place 6. Figure 5.16 shows the evolution of the belief about the position of person B during this experiment. As can be seen, the probability of person B to be at resting place 6 rapidly increased at time step $t = 73$ when the robot checked resting place 7 and did not detect him there.

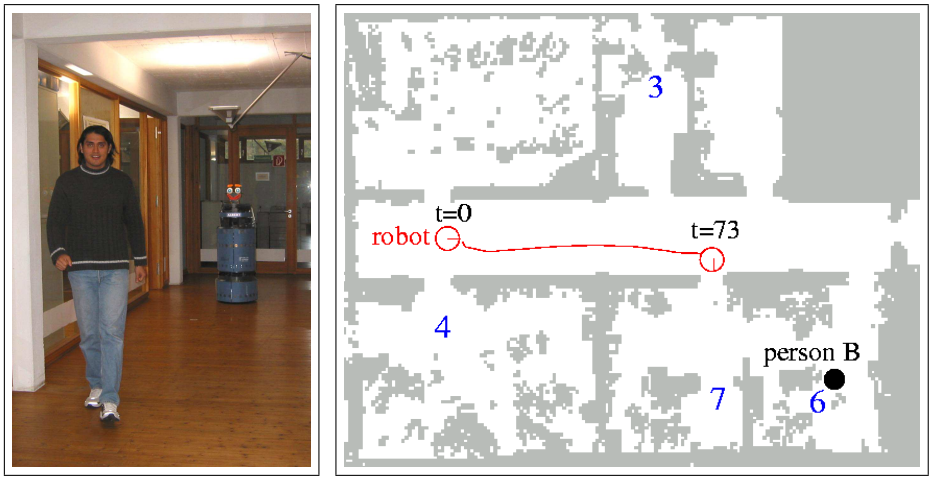


Figure 5.14: The left image shows person B walking down the corridor. It enters the room containing resting place 7 and walks to resting place 6 in the neighbor room. Since the robot is uncertain where person B is going to after he moved out of the field of view, the robot decides to search for him. Since the viewpoint corresponding to resting place 7 has the highest expected utility the robot moves there to perform an observation action (right image).

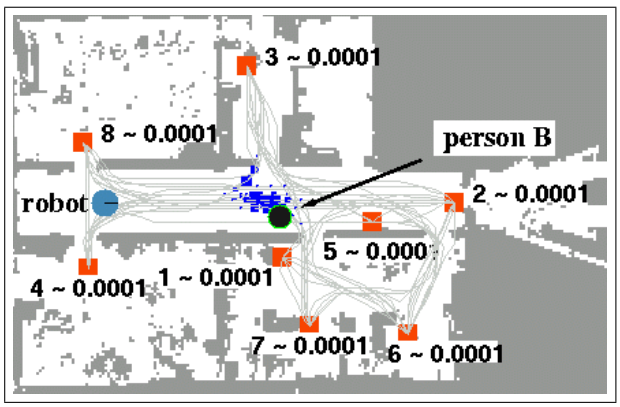


Figure 5.15: The robot detects that person B walks away through the corridor. Shown here is the belief about the position of person B after integrating part of the observation sequence.

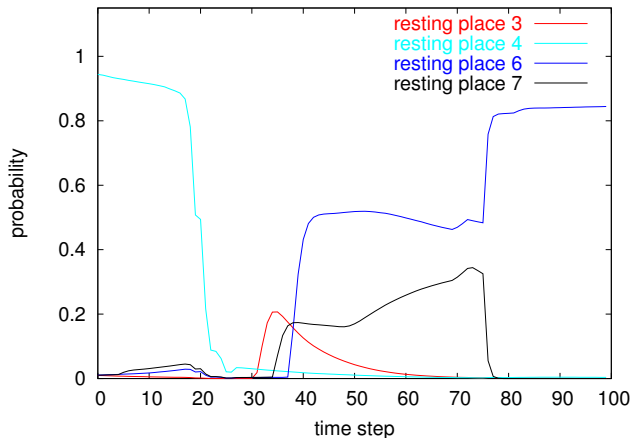


Figure 5.16: Evolution of the probability of person B to be at different resting places over time. When the robot does not detect a person at resting place 7 (time step $t = 73$) it infers that person B is probably staying at resting place 6.

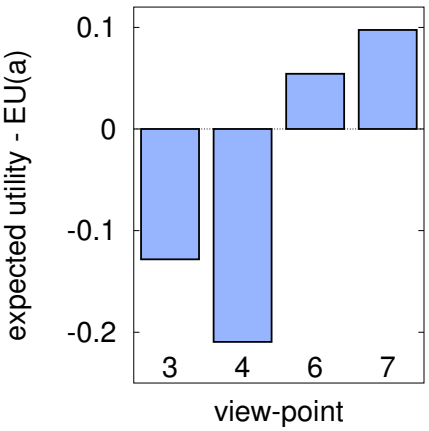


Figure 5.17: Expected utilities of different viewpoints at the time step where the robot decides to move. Here a is the current task the robot is executing.

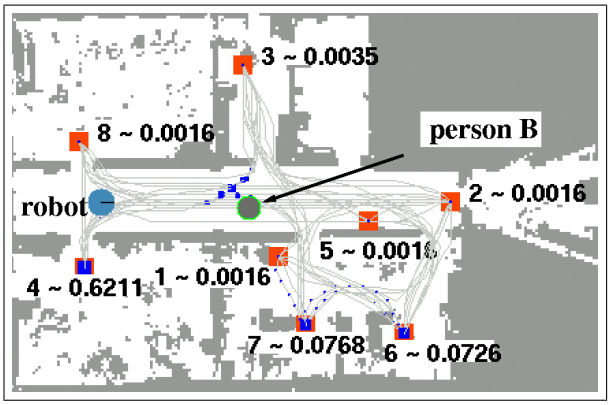


Figure 5.18: In this experiment the robot detects a person in the corridor. Since the robot is uncertain who the person is the probability that person B still stays at resting place 4 remains high even if it was him who walked down the corridor. Shown here is the belief about the position of person B after integrating part of the observation sequence.

Keeping Track of Multiple Person

The final experiment described in this section is designed to illustrate that the robot can actively maintain a belief about the positions of multiple persons. In this particular experiment the robot was keeping track of two persons and was standing at the same place as in the beginning of the previous experiment.

Initially the robot believed that person A and B were most likely at resting place 4. At around time step $t = 15$ the robot observed one person walking to the east along the corridor and entering the office containing resting place 7. Since the similarity measures between the extracted segment in the camera image and the database histograms of person A and B were ambiguous, the robot was rather uncertain which person it had observed. The similarity between the corresponding segment in the camera image and the database image of person B was only slightly higher than the similarity to the database image of person A. More precisely, the likelihood that the detected person was person B was 0.57 and the likelihood of person A was 0.43. Therefore, the probability of person B, who actually walked down the corridor, to be at resting place 4 remained high. Figure 5.18 depicts the posterior about the position of person B after integrating part of the observation sequence. At around time step $t = 25$ the robot decided to turn to resting place 4 to check which person was still there (Figure 5.19 depicts the situation). The robot identified person A at time step $t = 35$ and updated its belief accordingly. As can be seen in the upper image of Figure 5.20 the probabilities that person B was at the resting places 6 and 7 immediately increased after the inspection of resting

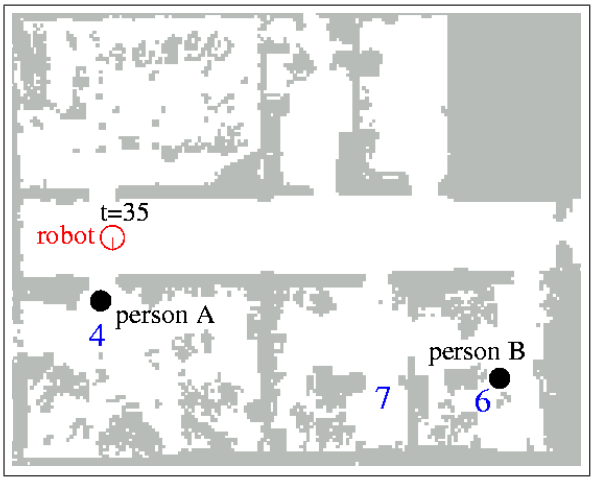


Figure 5.19: Initially the robot was quite certain that persons A and B both were in the room containing resting place 4 (see initial belief in Figure 5.20). After the robot observed a person walking away it turns to resting place 4 to check which person is still there.

place 4. The lower image shows the evolution of the belief about the position of person A. As can be seen from the graph the robot was now very sure that person A was at resting place 4.

When the robot starts turning to resting place 4 after time step $t = 25$ when the person left its field of view we do not use observations to update the HMMs. This is because the laser-based feature detection system has too many false positive detections while the robot is turning. As a result, the probabilities for both persons to be at resting place 4 decrease according to the transition probabilities.

5.7 Related Work

The problem of localizing people in the environment of mobile robots has been studied intensively in the past. Several techniques exist to keep track of moving people in the surrounding of a mobile robot (for example [Rosales and Sclaroff, 1998, MacCormick and Blake, 1999, Kluge *et al.*, 2001, Lindström and Eklundh, 2001, Montemerlo *et al.*, 2002b, Fod *et al.*, 2002, Schulz *et al.*, 2003a]). We introduced them in Chapter 3. These approaches can only deal with temporary occlusions and cannot keep track of the positions of the people when they are not in the perceptual field of view of the robot anymore. In contrast to that using our derived HMMs which model typical motion behaviors the robot is able to maintain an estimate about the positions of multiple persons even if they are not

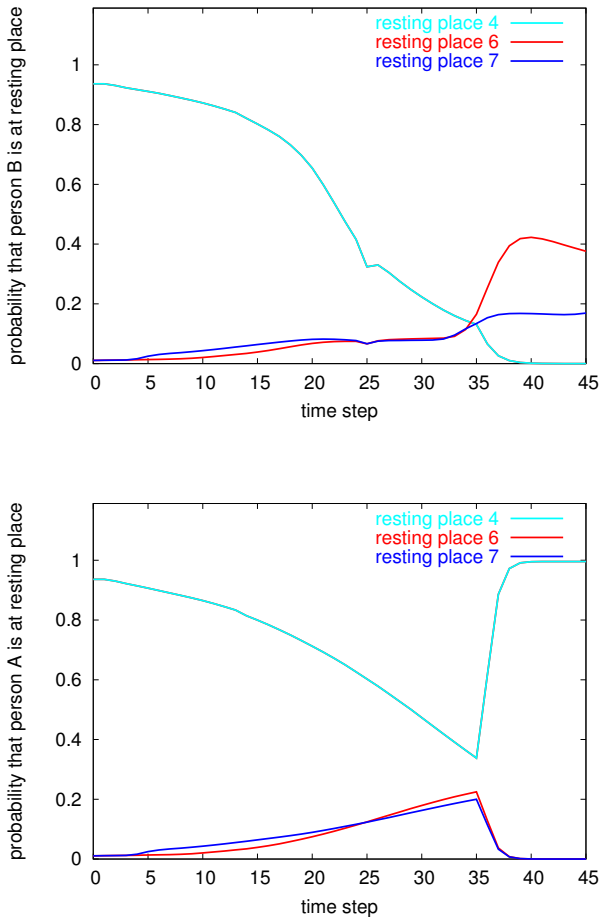


Figure 5.20: The upper image shows the evolution of the probability of person B to be at different resting places over time for the case that the robot is uncertain who the detected person is and therefore decides to inspect resting place 4. When the robot observes person A there the probabilities of resting places 6 and 7 immediately increase for person B (time step $t = 35$). The lower image shows that at the same time the probability of person A to be at resting place 4 seriously increases.

in the sensor range of the robot for a long period of time.

Rosencrantz *et al.* [2003] use variable-dimension particle filters to track the location of the moving objects under prolonged periods of occlusion. However, they are not able to estimate the probability of certain positions when the objects are outside the sensor range. They can only predict that the objects are in specific areas (rooms). Furthermore, they do not distinguish between the tracked objects. Schulz *et al.* [2003b] presented a technique which uses a combination of anonymous sensors (like laser-range finders) and ID-sensors to track and identify multiple persons. This approach can reliably estimate the IDs and trajectories of people in the range of the anonymous sensors even if currently no information from the ID-sensors is available.

Galata and Hogg [2001] use Variable Length Markov Models (VLMMs) to model structured behaviors. One problem to be solved in the context of VLMMs is the estimation of the optimal size of the time window in order to correctly predict the next states. In contrast to this, we follow a conservative approach and assume all steps in the past to be relevant. Bui *et al.* [2001] use an Abstract Hidden Markov Model (AHMM) to track and predict trajectories of people given camera information. Thereby they do not distinguish between different persons. This model uses a hierarchical representation for the higher-level and lower-level goals of people. Whereas the higher-level goals in such an AHMM correspond to the resting places of the people, the lower-level goals correspond the states along the paths between the resting places. Since we do not apply any abstractions because of the limited size of our model, we use a single HMM in which we encode in each state the position of the person and its intention.

Additionally, as mentioned in Chapter 4 several approaches to maintain the visibility of moving objects exist [Lavalle *et al.*, 1997, Parker, 1997, Pirjanian and Matarić, 2000, Feyrer and Zell, 2000, González-Baños *et al.*, 2002, Murrieta-Cid *et al.*, 2002, Jung and Sukhatme, 2002].

All these approaches are either passive in the sense that they just maintain a belief about the positions of the targets being tracked or are reactive and generate short-term plans to maintain or maximize the visibility of the objects being tracked. Our approach, in contrast, deals with the problem of actively maintaining an accurate belief about the positions of people while the robot has to carry out navigation tasks such as office delivery.

5.8 Conclusion

In this chapter we first explained the basic concepts of Hidden Markov Models. We then introduced a method for automatically deriving an HMM from typical motion patterns of people. To update the resulting HMMs based on laser-range

data and vision information we apply Joint Probabilistic Data Association Filters. Furthermore, we considered the problem of actively maintaining an accurate belief about the positions of multiple person in the environment of a mobile robot. Our approach uses a decision-theoretic approach to determine observation actions that are carried out while the robot is executing its tasks. The utility of an observation action is computed by trading-off move costs and the expected reduction of uncertainty.

In practical experiments we demonstrated that, by using our HMMs, a mobile robot equipped with a laser-range sensor and a vision system can reliably estimate the positions of multiple persons even if it cannot observe them for a long period of time. Furthermore, the experimental results demonstrate that our decision-theoretic approach generates effective actions that seriously reduce the uncertainty in the belief about the positions of people.

Chapter 6

Conclusions

In this thesis we focused on mobile robots which share their workspace with humans or with other robots. While considering the problem of path planning for teams of mobile robots we assume the existence of a central system which computes collision-free paths for all robots. However, searching for the optimal path in the composite state space of all robots is generally not feasible since this joint state space grows exponentially with the number of robots. To make the search tractable we therefore consider prioritized approaches which assign a unique priority to each robot. The paths of the robots are successively computed in the order implied by the priority scheme and by taking into account the paths of the robots with higher priority. As we illustrated in various experiments, the order in which the paths of the robots are planned has a serious influence on whether a solution can be found at all and on how long the resulting paths are. We therefore developed a method which performs a hill-climbing search in the space of priority schemes. We interleave the search for an optimal priority scheme with the planning of the paths of the robots. To find solvable schemes even for large teams of robots, constraints derived from the task specification are used to guide the search. We demonstrated in extensive experiments on real robots as well as in simulations that our approach enables prioritized path planning methods

- to seriously increase the number of planning problems which can be solved and
- to generate efficient solutions even for complex multi-robot problems.

In the second part of this work we focused on mobile robots acting in environments populated by humans. In contrast to a multi-robot system, the future trajectories of people are not known. Therefore, in order to not interfere with people, a robot needs to be able to locate and track people and to react appropriately to their activities. We observed that people usually do not move randomly. Instead,

they follow specific trajectories or motion patterns corresponding to their intentions. Obviously, a robot can improve its behavior by knowing about such typical motion patterns of people. This way, the robot can make better prediction about future motions of the people and also about the positions of the people when they are currently not in its field of view.

We presented a technique for learning collections of trajectories that characterize typical motion patterns of people. Our approach clusters data recorded with laser-range finders using the popular expectation maximization algorithm. We described how to use the learned motion patterns to derive a belief about potential trajectories of people. Afterward, we explained how to incorporate this belief into the path planning process of a mobile robot. We furthermore introduced a method for automatically deriving Hidden Markov Models (HMMs) from the motion patterns of people. These HMMs are used by a mobile robot to estimate the positions of multiple persons even when they are outside its sensor range. To update the HMMs based on laser-range data and vision information we apply Joint Probabilistic Data Association Filters. In practice, the robot becomes uncertain about the position of a person when it has not been observing the person for a longer period of time. We therefore proposed a decision-theoretic approach to determine observation actions that are carried out while the robot is executing its tasks.

Practical experiments carried out with a mobile robot and using different environments demonstrate:

- that our method is able to learn typical motion patterns of people,
- that the navigation behavior of the robot can be improved by predicting the motions of people based on the learned motion patterns,
- that the derived HMMs can be used to reliably maintain a probabilistic belief about the current positions of multiple persons even if they are currently not in its field of view, and
- that our technique generates effective actions that seriously reduce the uncertainty in the belief about the positions of people.

Despite this encouraging results, several areas for improvements exist. For example, one reasonable extension is to incorporate information about the current time in order to make our models more predictive. In many cases, the behavior of the people varies depending on the time of day. Furthermore, it should be monitored how reliably the HMMs predict the motions of people. Since the behavior of people can change over time the transition probabilities of the HMMs should be adapted accordingly.

Recently, radio frequency identification (RFID) tags have been become rather popular. While these ID-sensors provide the identity information of a person

within the range of a receiver, they do not provide an accurate location information. Therefore, it would be interesting to use this technology in combination with our HMMs to estimate and predict the positions of people. These aspects are subject for future research.

In summary, we presented techniques which facilitate the coexistence of robots and humans in real world environments as well as the interaction between them. Our approach is useful for service robots that are designed to coexist with humans and to fulfill various tasks such as delivery, cleaning, entertainment, and assistance of people in their everyday activities.

Appendix A

A.1 The B21r Robot Albert

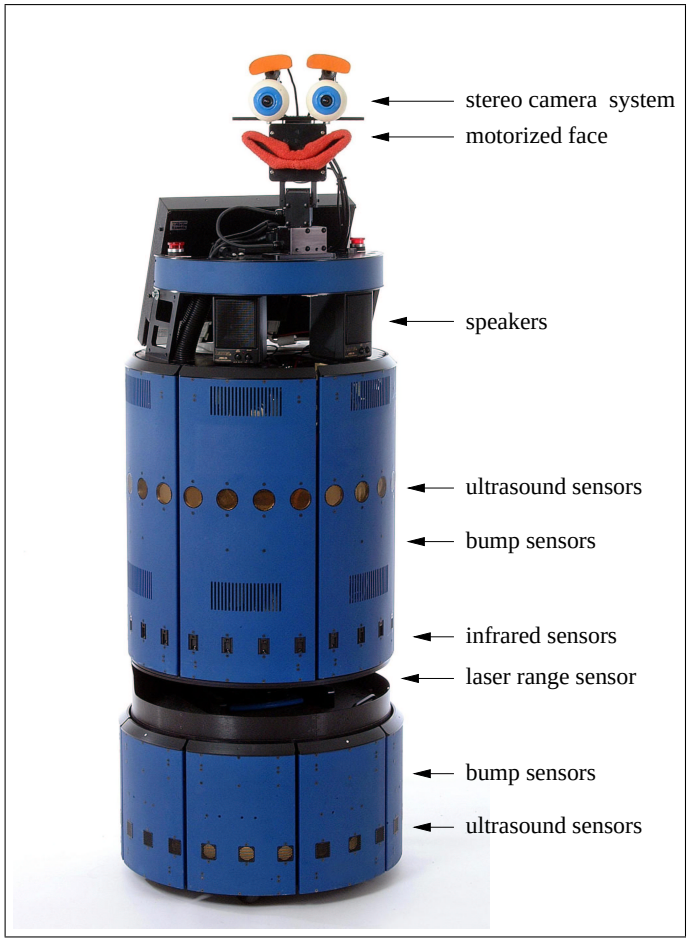


Figure A.1: Mobile robot Albert which has been used to carry out the experiments.

Parameter	Value
Laser scanner	1 SICK PLS (1° resolution)
Camera system	stereo color camera system
Number of ultrasound sensors	48
Number of infrared sensors	24
Number of bump sensors	56
CPU	450 MHz Pentium II
Translational velocity	max. $90\,cm/sec$
Rotational velocity	max. $167°/sec$
Drive	4-wheel synchro-drive
Radius	$26.25\,cm$
Height	$150\,cm$
Weight	approx. $120\,kg$

Figure A.2: The mobile Robot Albert.

A.2 Probability Theory

A.2.1 Bayes' Rule

The Bayes' Rule, which is used in this thesis several times, is given by the following equation:

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}. \tag{A.1}$$

The denominator is just a normalizing constant that ensures that the posterior of the left hand side adds up to 1 over all possbile values of A. Thus, we often write:

$$P(A \mid B) = \eta \cdot P(B \mid A) \cdot P(A). \tag{A.2}$$

A.2.2 Product Rule

The following equation is called the product rule:

$$P(A, B) = P(A \mid B) \cdot P(B) = P(B \mid A) \cdot P(A). \tag{A.3}$$

A.2.3 Marginalization

The marginalization rule is the following equation:

$$P(A) = \sum_{b_i} P(A, B = b_i). \tag{A.4}$$

A.2.4 Law of Total Probability

The law of total probability is a variant of the marginalization rule, which can be derived using the product rule:

$$P(A) = \sum_{b_i} P(A \mid B = b_i) \cdot P(B = b_i). \tag{A.5}$$

A.3 Proof: EM Monotonically Increases the Data Likelihood

Given the observed data s and the hidden data c the log likelihood of s under the model θ is given by:

$$\ln P(s \mid \theta) = \ln \sum_c P(s, c \mid \theta) \tag{A.6}$$

using marginalization (see A.2.3). Because of the concavity of $\ln$ and using Jensen's Inequality[1] we get for any distribution $q(c)$ over the hidden variables [Ghahramani, 1998]:

$$\begin{aligned}
\ln P(s \mid \theta) &= \ln \sum_c P(s, c \mid \theta) && \text{(A.7)}\\
&= \ln \sum_c q(c) \frac{P(s, c \mid \theta)}{q(c)} && \text{(A.8)}\\
&\geqq \sum_c q(c) \ln \frac{P(s, c \mid \theta)}{q(c)} && \text{(A.9)}\\
&= \sum_c q(c) \ln P(s, c \mid \theta) - \sum_c q(c) \ln q(c) && \text{(A.10)}\\
&= \mathcal{F}(q, \theta). && \text{(A.11)}
\end{aligned}$$

The EM algorithm alternates between maximizing $\mathcal{F}$ with respect to q and θ, starting with some initial model $\theta^{[0]}$:

E step: $q^{[j+1]} = \operatorname{argmax}_q \mathcal{F}(q, \theta^{[j]})$

M step: $\theta^{[j+1]} = \operatorname{argmax}_\theta \mathcal{F}(q^{[j+1]}, \theta)$.

[1] Suppose f is a continuous strictly concave function and for $1 \leqq i \leqq n : \sum_i a_i = 1$ and $a_i > 0$. Then Jensen's Inequality says: $f(\sum_i a_i x_i) \geqq \sum_i a_i f(x_i)$.

It is easy to proof that the maximum value for $q^{[j+1]}$ in the E step is $P(c \mid \theta^{[j]}, s)$:

$$\begin{aligned}
\mathcal{F}(P(c \mid \theta^{[j]}, s), \theta^{[j]}) &= \sum_c P(c \mid \theta^{[j]}, s) \ln \frac{P(s, c \mid \theta^{[j]})}{P(c \mid s, \theta^{[j]})} && \text{(A.12)} \\
&= \sum_c P(c \mid \theta^{[j]}, s) \ln P(s \mid \theta^{[j]}) && \text{(A.13)} \\
&= \ln P(s \mid \theta^{[j]}) \sum_c P(c \mid \theta^{[j]}, s) && \text{(A.14)} \\
&= \ln P(s \mid \theta^{[j]}) \cdot 1 && \text{(A.15)}
\end{aligned}$$

Thus, by setting $q^{[j+1]} = P(c \mid \theta^{[j]}, s)$ the bound in Eq. (A.11) becomes an equality.

Since the second term of $\mathcal{F}(q, \theta)$ does not depend on θ we need only to consider the first term, which is denoted as $E_c[\ln P(s, c \mid \theta) \mid \theta, s]$ in Chapter 3, in the M-step.

Since $\ln P(s \mid \theta) = \mathcal{F}(q, \theta)$ at the beginning of each M step, and since the E step does not change θ, the likelihood is guaranteed not to decrease after one iteration of EM.

A.4 Approximation of the Probability $P(\psi \mid z^{(1:t)})$

Using the law of total probability (see A.2.4) and under the assumption that the estimation problem is Markovian the probability $P(\psi \mid z^{(1:t)})$ of an individual joint association event ψ given the observation sequence $z^{(1:t)}$ can be computed as [Schulz *et al.*, 2003a]:

$$\begin{aligned}
P(\psi \mid z^{(1:t)}) &= P(\psi \mid z^t, z^{(1:t-1)}) && \text{(A.16)} \\
&= \int P(\psi \mid z^t, z^{(1:t-1)}, \xi^t)\ P(\xi^t \mid z^t, z^{(1:t-1)})\ d\xi^t && \text{(A.17)} \\
&= \int P(\psi \mid z^t, \xi^t)\ P(\xi^t \mid z^t, z^{(1:t-1)})\ d\xi^t. && \text{(A.18)}
\end{aligned}$$

Accordingly, the state ξ^t of the persons has to be known in order to determine the assignments ψ. On the other hand, ψ has to be known in order to determine the positions of the tracked persons. To overcome this problem $P(\xi^t \mid z^t, z^{(1:t-1)})$ is approximated by the belief $P(\xi^t \mid z^{(1:t-1)})$ about the predicted state of the persons, which is computed using all measurements perceived before time step t:

$$\begin{aligned}
P(\psi \mid z^{(1:t)}) &\approx \int P(\psi \mid z^t, \xi^t)\ P(\xi^t \mid z^{(1:t-1)})\ d\xi^t. && \text{(A.19)} \\
&= \eta \int P(z^t \mid \psi, \xi^t)\ P(\psi \mid \xi^t)\ P(\xi^t \mid z^{(1:t-1)})\ d\xi^t. && \text{(A.20)}
\end{aligned}$$

Here η is a normalizer and the last transformation corresponds to Bayes' Rule.

A.5 Computation of the Assignment Probabilities λ_{sr}

The probability that feature s is caused by person r is given by (the notations correspond to those introduced in Section 5.4):

$$\lambda_{sr} = \sum_{\psi \in \Psi_{sr}} \left[\eta' \, \gamma^{\phi} \int \prod_{(j,i)\in\psi} P(z_j^t \mid \xi_i^t) \; P(\xi^t \mid z^{(1:t-1)}) \; d\xi^t \right]. \tag{A.21}$$

Since we assume that the individual state spaces are independent the integral part can be written as:

$$\int \prod_{(j,i)\in\psi} P(z_j^t \mid \xi_i^t) \; P(\xi^t \mid z^{(1:t-1)}) \; d\xi^t \tag{A.22}$$

$$= \int \prod_{(j,i)\in\psi} P(z_j^t \mid \xi_i^t) \prod_{r=1}^{R} P(\xi_r^t \mid z^{(1:t-1)}) \; d\xi^t \tag{A.23}$$

$$= \int \prod_{(j,i)\in\psi} P(z_j^t \mid \xi_i^t) \; P(\xi_i^t \mid z^{(1:t-1)}) \; d\xi^t \tag{A.24}$$

$$= \int_{\xi_R^t} \dots \int_{\xi_2^t} \int_{\xi_1^t} \prod_{(j,i)\in\psi} P(z_j^t \mid \xi_i^t) \; P(\xi_i^t \mid z^{(1:t-1)}) \; d\xi_1^t \, d\xi_2^t \dots d\xi_R^t \,. \tag{A.25}$$

Here Eq. (A.24) uses the fact that ψ assigns each person i one feature j. Let z_{j_i} denote the feature which is assigned to person i in ψ. Then Eq. (A.25) can be written as:

$$\int_{\xi_R^t} \dots \int_{\xi_2^t} \int_{\xi_1^t} P(z_{j_1} \mid \xi_1^t) \; P(\xi_1^t \mid z^{(1:t-1)}) \cdot \prod_{(j,i)\in\psi\setminus\{(j_1,1)\}} P(z_j^t \mid \xi_i^t) \; P(\xi_i^t \mid z^{(1:t-1)}) \; d\xi_1^t \, d\xi_2^t \dots d\xi_R^t \tag{A.26}$$

$$= \int_{\xi_1^t} P(z_{j_1} \mid \xi_1^t) \; P(\xi_1^t \mid z^{(1:t-1)}) \; d\xi_1^t \cdot \int_{\xi_R^t} \dots \int_{\xi_2^t} \prod_{(j,i)\in\psi\setminus\{(j_1,1)\}} P(z_j^t \mid \xi_i^t) \; P(\xi_i^t \mid z^{(1:t-1)}) \; d\xi_2^t \dots d\xi_R^t. \tag{A.27}$$

These transformations can be applied for each i so that we finally obtain:

$$\prod_{i=1}^{R} \int_{\xi_i^t} P(z_{j_i} \mid \xi_i^t)\ P(\xi_i^t \mid z^{(1:t-1)})\ d\xi_i^t \tag{A.28}$$

$$= \prod_{(j,i)\in\psi} \int_{\xi_i^t} P(z_j \mid \xi_i^t)\ P(\xi_i^t \mid z^{(1:t-1)})\ d\xi_i^t. \tag{A.29}$$

Which is the justification for Eq. (5.13). The final transformation is correct according to the definition of z_{j_i}.

Bibliography

[Alami *et al.*, 1995] R. Alami, F. Robert, F. Ingrand, and S. Suzuki. Multi-robot cooperation through incremental plan-merging. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 1995.

[Alami *et al.*, 1998a] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand. An architecture for autonomy. *International Journal of Robotic Research*, 17(4):315–337, 1998.

[Alami *et al.*, 1998b] R. Alami, S. Fleury, M. Herb, F. Ingrand, and F. Robert. Multi robot cooperation in the Martha project. *IEEE Journal of Robotics and Automation*, 5, 1998.

[Alami *et al.*, 2000] R. Alami, R. Chatila, S. Fleury, M. Herrb, F. Ingrand, M. Khatib, B. Morissett, P. Moutarlier, and T. Siméon. Around the lab in 40 days... In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2000.

[Arkin *et al.*, 1993] R.C. Arkin, W.M. Carter, and D.C. Mackenzie. Active avoidance: Escape and dodging behaviors for reactive control. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(1):177–192, 1993.

[Arras and Vestli, 1998] K.O. Arras and S.J. Vestli. Hybrid, high-precision localization for the mail distributing mobile robot system MOPS. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 1998.

[Asoh *et al.*, 1997] H. Asoh, S. Hayamizu, I. Hara, Y. Motomura, S. Akaho, and T. Matsui. Socially embedded learning of office-conversant robot Jijo-2. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1997.

[Azarm and Schmidt, 1996] K. Azarm and G. Schmidt. A decentralized approach for the conflict-free motion of multiple mobile robots. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1667–1674, 1996.

[Bar-Shalom and Fortmann, 1988] Y. Bar-Shalom and T.E. Fortmann. *Tracking and Data Association*. Academic Press, 1988.

[Barraquand and Latombe, 1990] J. Barraquand and J.C. Latombe. A monte-carlo algorithm for path planning with many degrees of freedom. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 1990.

[Barraquand *et al.*, 1992] J. Barraquand, B. Langois, and J.C. Latombe. Numerical potential field techniques for robot path planning. *IEEE Transactions on Robotics and Automation, Man and Cybernetics*, 22(2):224–241, 1992.

[Bien and Lee, 1992] Z. Bien and J. Lee. A minimum-time trajectory planning method for two robots. *IEEE Transactions on Robotics and Automation*, 8(3):414–418, 1992.

[Bilmes, 1997] J. Bilmes. A gentle tutorial on the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical report, University of Berkeley, 1997.

[Bohlin and Kavraki, 2000] R. Bohlin and L. Kavraki. Path planning using lazy prm. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2000.

[Brown and Hwang, 1992] R. G. Brown and P. Y. C. Hwang. *Introduction to Random Signals and Applied Kalman Filtering*. John Wiley & Sons, 1992.

[Buckley, 1989] S.J. Buckley. Fast motion planning for multiple moving robots. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 1989.

[Bui *et al.*, 2001] H. Bui, S. Venkatesh, and G. West. Tracking and surveillance in wide-area spatial environments using the abstract hidden markov model. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(1):177–195, 2001.

[Burgard *et al.*, 1999] W. Burgard, A.B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(1-2), 1999.

[Burgard *et al.*, 2000] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun. Collaborative multi-robot exploration. In *ICRA*, 2000.

[Chai *et al.*, 1995] A.-H. Chai, T. Fukuda, F. Arai, T. Ueyama, and A. Sakai. Hierachical control architecture for cellular robotic system-simulations and -experiments. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, pages 1191–1196, 1995.

[Chang *et al.*, 1994] C. Chang, M.J. Chung, and B.H. Lee. Collision avoidance of two robot manipulators by minimum delay time. *IEEE Transactions on Robotics and Automation, Man and Cybernetics*, 24(3):517–522, 1994.

[Chatila *et al.*, 2002] R. Chatila, R. Alami, T. Siméon, J. Pettre, and L. Jaillet. Safe, reliable and friendly interaction between humans and humanoids. In *Proc. of the 3rd IARP International Workshop on Humanoid and Human Friendly Robotics*, 2002.

[Chu and EiMaraghy, 1992] H. Chu and H.A. EiMaraghy. Real-time multi-robot path planner based on a heuristic approach. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 1992.

[Chun *et al.*, 1999] L. Chun, Z. Zheng, and W. Chang. A decentralized approach to the conflict-free motion planning for multiple mobile robots. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 1999.

[Clark *et al.*, 2003] C.M. Clark, S.M. Rock, and J.-C. Latombe. Motion planning for multiple mobile robots using dynamic networks. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2003.

[Cox and Miller, 1965] D.R. Cox and H.D. Miller. *The Theory of Stochastic Processes*. Wiley, New York, 1965.

[Cox, 1993] I.J. Cox. A review of statistical data association techniques for motion correspondence. *International Journal of Computer Vision*, 10(1):53–66, 1993.

[Dempster *et al.*, 1977] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39 (Series B):1–38, 1977.

[Dietl *et al.*, 2002] M. Dietl, F. Diesch, S. Gutmann, A. Kleiner, B. Nebel, B. Stiegeler, B. Szerbakowski, and T. Weigel. CS Freiburg – The robotic soccer team. http://www.informatik.uni-freiburg.de/~robocup/, 2002.

[Duda *et al.*, 2001] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley-Interscience, 2001.

[Endres *et al.*, 1998] H. Endres, W. Feiten, and G. Lawitzky. Field test of a navigation system: Autonomous cleaning in supermarkets. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 1998.

[Engelberger, 1993] J. F. Engelberger. Health-care robotics goes commercial: The 'helpmate' experience. *Robotica*, 11:517–523, 1993.

[Erdmann and Lozano-Pérez, 1987] M. Erdmann and T. Lozano-Pérez. On multiple moving objects. *Algorithmica*, 2:477–521, 1987.

[Everett, 1998] H.R. Everett. Breaking down the barriers. *Unmanned Vehicles*, 3(1), 1998.

[Everitt, 1974] B.S. Everitt. *Cluster Analysis*. John Wiley & Sons, New York, 1974.

[Ferrari *et al.*, 1998] C. Ferrari, E. Pagello, J. Ota, and T. Arai. Multirobot motion coordination in space and time. *Robotics and Autonomous Systems*, 25:219–229, 1998.

[Feyrer and Zell, 2000] S. Feyrer and A. Zell. Robust real-time pursuit of persons with a mobile robot using multisensor fusion. In *6th International Conference on Intelligent Autonomous Systems (IAS-6)*, 2000.

[Fod *et al.*, 2002] A. Fod, A. Howard, and M.J. Matarić. A laser-based people tracker. In *ICRA*, 2002.

[Foka and Trahanias, 2002] A.F. Foka and P.E. Trahanias. Predictive autonomous robot navigation. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 490–495, 2002.

[Forgy, 1965] E. Forgy. Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications. *Biometrics*, 21(3), 1965.

[Fox *et al.*, 1998] D. Fox, W. Burgard, and S. Thrun. Active markov localization for mobile robots. *Robotics and Autonomous Systems*, 25:195–207, 1998.

[Fraley and Raftery, 1998] C. Fraley and A.E. Raftery. How many clusters? which clustering method? answers via model-based cluster analysis. *The Computer Journal*, 41(8):578–588, 1998.

[Gaffney and Smyth, 1999] S. Gaffney and P. Smyth. Trajectory clustering with mixtures of regression models. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1999.

[Galata and Hogg, 2001] N. Galata, A.and Johnson and D. Hogg. Learning variable length markov models of behaviour. *Computer Vision and Image Understanding (CVIU) Journal*, 81(3), 2001.

[Ghahramani, 1998] Z. Ghahramani. Learning dynamic Bayesian Networks. *Lecture Notes in Computer Science*, 1387:168–197, 1998.

[Goldberg and Siegwart, 2001] K. Goldberg and R. Siegwart. *Robots on the Web: Physical Interaction through the Internet*. MIT-Press, 2001.

[González-Baños *et al.*, 2002] H.H. González-Baños, C.-Y. Lee, and J.-C. Latombe. Real-time combinatorial tracking of a target moving unpredictably among obstacles. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2002.

[Gravot and Alami, 2001] F. Gravot and R. Alami. An extension of the plan-merging paradigm for multi-robot coordination. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2001.

[Gross and Boehme, 2000] H.-M. Gross and H.-J. Boehme. Perses – a vision-based interactive mobile shopping assistant. In *Proc. of the IEEE International Conference on Systems, Man and Cybernetics*, 2000.

[Grossman, 1988] D. Grossman. Traffic control of multiple robot vehicles. *IEEE Journal of Robotics and Automation*, 4:491–497, 1988.

[Guralnik and Haigh, 2002] V. Guralnik and K.Z. Haigh. Learning models of human behaviour with sequential patterns. In *Proc. of the AAAI-02 workshop "Automation as Caregiver"*, 2002.

[Hefter, 2004] Hefter. Cleaning robot. http://www.hefter.de/cleantech/roboter.htm, 2004.

[Hirzinger *et al.*, 1994] G. Hirzinger, B. Brunner, J. Dietrich, and J. Heindl. ROTEX the first remotely controlled robot in space. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 1994.

[Horswill, 1993] I. Horswill. Polly: A vision-based artificial agent. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 1993.

[Illmann *et al.*, 2002] J. Illmann, B. Kluge, and E. Prassler. Statistical recognition of motion patterns. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2002.

[Jäger and Nebel, 2001] M. Jäger and B. Nebel. Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2001.

[Jäger and Nebel, 2002] M. Jäger and B. Nebel. Dynamic decentralized area partitioning for cooperating cleaning robots. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2002.

[Jensen *et al.*, 2003] B. Jensen, R. Philippsen, and R. Siegwart. Motion detection and path planning in dynamic environments. In *Proc. of the IJCAI-03 Workshop "Reasoning with Uncertainty in Robotics"*, 2003.

[Johnson and Hogg, 1995] N. Johnson and D. Hogg. Learning the distribution of object trajectories for event recognition. In *British Machine Vision Conference*, 1995.

[Jung and Sukhatme, 2002] B. Jung and G.S. Sukhatme. A region-based approach for cooperative multi-target tracking in a structural environment. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2002.

[Kaelbling *et al.*, 1998] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101, 1998.

[Kalman, 1960] R.E. Kalman. A new approach to linear filtering and prediction problems. *ASME-Journal of Basic Engineering*, 82:35–45, 1960.

[Kant and Zucker, 1986] K. Kant and S.W. Zucker. Toward efficient trajectory planning: The path-velocity decomposition. *IJRR*, 5(3):72–89, 1986.

[Kasper *et al.*, 2001] M. Kasper, G. Fricke, K. Stuernagel, and E. von Puttkamer. A behavior-based mobile robot architecture for learning from demonstration. *Journal of Robotics and Automation*, 34(2-3):153–164, 2001.

[Kavraki *et al.*, 1996] L. Kavraki, P. Svestka, J.C. Latombe, and M. Overmars. Probabilistic road maps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[King and Weiman, 1990] S. King and C. Weiman. Helpmate autonomous mobile robot navigation system. In *Proc. of the SPIE Conference on Mobile Robots*, pages 190–198, Boston, MA, November 1990.

[Kluge *et al.*, 2001] B. Kluge, C. Köhler, and E. Prassler. Fast and robust tracking of multiple moving objects with a laser range finder. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2001.

[Krumm *et al.*, 2000] J. Krumm, S. Harris, B. Meyers, B. Brumitt, M. Hale, and S. Shafer. Multi-camera multi-person tracking for Easyliving. In *Proc. of Third IEEE International Workshop on Visual Surveillance*, 2000.

[Kruse, 1998] F. Kruse, E. und Wahl. Camera-based monitoring system for mobile robot guidance. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1248–1253, 1998.

[Lacey and Dawson-Howe, 1998] G. Lacey and K. Dawson-Howe. The application of robotics to a mobility aid for the elderly blind. *Journal of Robotics and Autonomous Systems (RAS)*, 23:245–252, 1998.

[Lankenau and Röfer, 2000] A. Lankenau and T. Röfer. Smart wheelchairs – state of the art in an emerging market. *Zeitschrift KI mit Schwerpunkt Autonome Mobile Systeme*, 4, 2000.

[Latombe, 1991] J.C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991. ISBN 0-7923-9206-X.

[LaValle and Hutchinson, 1996] S.M. LaValle and S.A. Hutchinson. Optimal motion planning for multiple robots having independent goals. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 1996.

[Lavalle *et al.*, 1997] S.M. Lavalle, H.H. González-Banos, G. Becker, and J.-C. Latombe. Motion strategies for maintaining visibility of a moving target. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 1997.

[Lawler *et al.*, 1989] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys. Sequencing and scheduling: Algorithms and complexity. Technical report, Centre for Mathematics and Computer Science, 1989.

[Lee *et al.*, 1995a] J. Lee, H.S. Nam, and J. Lyou. A practical collision-free trajectory planning for two robot systems. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 1995.

[Lee *et al.*, 1995b] J.-H. Lee, BH. Lee, M.H. Choi, J.D. Kim, K.-T. Joo, and H. Park. A real time traffic control scheme for a multiple agv system. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, pages 1625–1630, 1995.

[Leroy *et al.*, 1999] S. Leroy, J.P. Laumond, and T. Siméon. Multiple path coordination for mobile robots: A geometric algorithm. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1999.

[Li *et al.*, 2001] C. Li, G. Biswas, M. Dale, and P. Dale. Building models of ecological dynamics using hmm based temporal data clustering. In *Proc. of the Fourth International Conference on Intelligent Data Analysis*, 2001.

[Liao *et al.*, 2004] L. Liao, D. Patterson, D. Fox, and H. Kautz. Learning and inferring transportation routines. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 2004. to appear.

[Lindström and Eklundh, 2001] M. Lindström and J.-O. Eklundh. Detecting and tracking moving objects from a mobile platform using a laser range scanner. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2001.

[Liu *et al.*, 1989] Y.H. Liu, S. Kuroda, T. Naniwa, H. Noborio, and S. Arimoto. A practical algorithm for planning collision-free coordinated motion of multiple mobile robots. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, pages 1427–1432, 1989.

[Lumelsky and Harinarayan, 1997] V.J. Lumelsky and K.R. Harinarayan. Decentralized motion planning for multiple mobile robots: The cocktail party model. *Journal of Autonoumous Robots*, 4:121–135, 1997.

[MacCormick and Blake, 1999] J. MacCormick and A. Blake. A probabilistic exclusion principle for tracking multiple objects. In *Proc. of 7th International Conference on Computer Vision (ICCV)*, pages 572–587, 1999.

[MacQueen, 1967] J. MacQueen. Some methods for classification and analysis of mulivariate observations. In *Proc. of the Fifth Berkeley Symposium on Math., Stat. and Prob.*, pages 281–296, 1967.

[Mahalanobis, 1930] P.C. Mahalanobis. On tests and meassures of groups divergence I. *Journal of the Asiatic Society of Benagal*, 26(541), 1930.

[Martin and Shmoys, 1996] P. Martin and D.B. Shmoys. A new approach to computing optimal schedules for the job-shop scheduling problem. In *Proc. of the 5th International IPCO Conference*, pages 389–403, 1996.

[Maybeck, 1979] P. S. Maybeck. *Stochastic Models, Estimation, and Control*, volume 1. Academic Press, 1979.

[McHenry, 1998] M. McHenry. *Slice-Based Path Planning*. PhD thesis, University of Southern California, 1998.

[McLachlan and Krishnan, 1997] G.J. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley Series in Probability and Statistics, 1997.

[Mitchell, 1997] T.M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[Montemerlo *et al.*, 2002a] M. Montemerlo, J. Pineau, N. Roy, S. Thrun, and V. Verma. Experiences with a mobile robotic guide for the elderly. In *Proc. of the AAAI National Conference on Artificial Intelligence*, 2002.

[Montemerlo *et al.*, 2002b] M. Montemerlo, S. Thrun, and W. Whittaker. Conditional particle filters for simultaneous mobile robot localization and people-tracking. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2002.

[Moravec and Elfes, 1985] H.P. Moravec and A.E. Elfes. High resolution maps from wide angle sonar. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 116–121, 1985.

[Mozer, 1998] M.C. Mozer. The neural network house: An environment that adapts to its inhabitants. In *Proceedings of the American Association for Artificial Intelligence Spring Symposium on Intelligent Environments*, 1998.

[Murphy, 2002] K.P. Murphy. *Dynamic Bayesian Networks: Representation, Inference, and Learning*. PhD thesis, University of California, Berkely, 2002.

[Murrieta-Cid *et al.*, 2002] R. Murrieta-Cid, H.H. González-Baños, and B. Tovar. A reactive motion planner to maintain visibility of unpredictable targets. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2002.

[Nguyen *et al.*, 2003] N.T. Nguyen, H.H. Bui, S. Venkatesh, and G. West. Recognising and monitoring high-level behaviours in complex spatial environments. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003.

[Nilsson, 1982] N. J. Nilsson. *Principles of Artificial Intelligence*. Springer Publisher, Berlin, New York, 1982.

[Nourbakhsh *et al.*, 1995] I. Nourbakhsh, R. Powers, and S. Birchfield. Dervish: An office-navigating robot. *AI Magazine*, 16(2), 1995.

[O'Donnell and Lozano-Pérez, 1989] P.A. O'Donnell and T. Lozano-Pérez. Deadlock-free and collision-free coordination of two robot manipulators. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 1989.

[Oliver *et al.*, 2002] N. Oliver, E. Horvitz, and A. Garg. Layered representations for learning and inferring office activity from multiple sensory channels. In *Proc. of the International Conference on Multimodal Interfaces (ICMI)*, 2002.

[Parker, 1997] L.E. Parker. Cooperative motion control for multi-target observation. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1997.

[Pirjanian and Matarić, 2000] P. Pirjanian and M. Matarić. Multi-robot target acquisition using multiple objective behaviour coordination. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2000.

[Qutub *et al.*, 1997] S. Qutub, R. Alami, and S. Ingrand, F. How to solve deadlock situations within the plan-merging paradigm for multi-robot cooperation. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1997.

[Rabiner and Juang, 1986] L.R. Rabiner and B.H. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1):4–16, 1986.

[Reif, 1979] J.H. Reif. Complexity of the mover's problem and generalizations. In *Proc. of the 20th IEEE Symposium on Foundations of Computer Science*, pages 421–427, 1979.

[Riley and Veloso, 2002a] P. Riley and M. Veloso. Planning for distributed execution through use of probabilistic opponent models. In *Proc. of the Sixth International Conference on Artificial Intelligence Planning Systems*, 2002.

[Riley and Veloso, 2002b] P. Riley and M. Veloso. Recognizing probabilistic opponent movement models. In A. Birk, S. Coradeschi, and S. Tadokoro, editors, *RoboCup-2001: The Fifth RoboCup Competitions and Conferences*. Springer Verlag, Berlin, 2002.

[Rissanen, 1984] J. Rissanen. Universal encoding, information, prediction, and estimation. *IEEE Transactions on Information Theory*, 30(4):629–636, 1984.

[Rosales and Sclaroff, 1998] R. Rosales and S. Sclaroff. Improved tracking of multiple humans with trajectory prediction and occlusion modeling. In *Proc. of the IEEE Workshop on Interpretation of Visual Motion*, 1998.

[Rosales and Sclaroff, 2003] R. Rosales and S. Sclaroff. A framework for heading-guided recognition of human activity. *Computer Vision and Image Understanding (CVIU) Journal*, 2003.

[Rosencrantz *et al.*, 2003] M. Rosencrantz, G. Gordon, and S. Thrun. Locating moving entities in dynamic indoor environments with teams of mobile robots. In *Proc. of the Second Joint International Conference on Autonomous Agents & Multi Agent Systems (AAMAS)*, 2003.

[Russell and Norvig, 1995] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, New Jersey, 1995.

[Sánchez and Latombe, 2002] G. Sánchez and J.-C. Latombe. Using a prm planner to compare centralized and decoupled planning for multi-robot systems. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2002.

[Schaeffer and May, 1999] C. Schaeffer and T. May. Care-o-bot - a system for assisting elderly or disabled persons in home environments. In *Assistive technology on the threshold of the new millenium*. IOS Press, Amsterdam, 1999.

[Schulz *et al.*, 2000] D. Schulz, W. Burgard, D. Fox, S. Thrun, and A.B Cremers. Web interfaces for mobile robots in public places. *IEEE Robotics and Automation Magazine*, 7(1), 2000.

[Schulz *et al.*, 2003a] D. Schulz, W. Burgard, D. Fox, and A.B. Cremers. People tracking with a mobile robot using sample-based joint probabilistic data association filters. *International Journal of Robotics Research (IJRR)*, 2003.

[Schulz *et al.*, 2003b] D. Schulz, D. Fox, and J. Hightower. People tracking with anonymous and id-sensors using rao-blackwellised particle filers. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[Schwartz and Scharir, 1983] J.T. Schwartz and M. Scharir. On the "piano movers" problem: III. Coordinating the motion of several independent bodies moving amidst polygonal obstacles. *International Journal of Robotic Research*, 2(3):46–75, 1983.

[Schwartz *et al.*, 1987] J.T. Schwartz, M. Scharir, and J. Hopcroft. *Planning, Geometry and Complexity of Robot Motion*. Ablex Publishing Corporation, Norwood, NJ, 1987.

[Schwarz, 1978] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[Sebastiani *et al.*, 1999] P. Sebastiani, M. Ramoni, P. Cohen, J. Warwick, and J. Davis. Discovering dynamics using Bayesian Clustering. In *Proc. of the 3rd International Symposium on Illigent Data Analysis*, 1999.

[Selman *et al.*, 1992] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard instances of satisfiability. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 1992.

[Siegwart *et al.*, 2003] R. Siegwart, K.O. Arras, B. Jensen, R. Philippsen, and N. Tomatis. Design, implementation and exploitation of a new fully autonomous tour guide robot. In *Proc. of the 1st International Workshop on Advances in Service Robotics (ASER)*, 2003.

[Simmons *et al.*, 1997] R. Simmons, R. Goodwin, K. Haigh, S. Koenig, and J. O'Sullivan. A layered architecture for office delivery robots. In *Proc. of the First International Conference on Autonomous Agents*, 1997.

[Simmons *et al.*, 2000] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, S. Moors, M.and Thrun, and Younes H. Coordination for multi-robot exploration and mapping. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 2000.

[Smyth, 1997] P. Smyth. Clustering sequences with hidden markov models. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9. The MIT Press, 1997.

[Sony, 2003] Sony. Entertainment robots. http://www.aibo.com/, 2003.

[Souccar and Roderic, 1996] K. Souccar and A.G. Roderic. Distributed motion control for multiple robotic manipulators. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 1996.

[Stachniss and Burgard, 2002] C. Stachniss and W. Burgard. An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2002.

[Stauffer and Grimson, 2000] C. Stauffer and W.E.L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, 2000.

[Sutton and Barto, 1998] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[Sveska and Overmars, 1995] P. Sveska and M. Overmars. Coordinated motion planning for multiple car-like robots using probabilistic roadmaps. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 1995.

[Swain and Ballard, 1991] M. Swain and D. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1), 1991.

[Swiss Federal Institute of Technology Lausanne, 2002] Swiss Federal Institute of Technology Lausanne. Expo 2002. http://robotics.epfl.ch/, 2002.

[Tacke, 2002] M. Tacke. Errechnung der relativen Positionen einer Gruppe von Laserentfernungsmessern. Studienarbeit. University of Freiburg, 2002. In German.

[Tadokoro *et al.*, 1995] S. Tadokoro, M. Hayashi, Y. Manabe, Y. Nakami, and T. Takamori. On motion planning of mobile robots which coexist and cooperate with human. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 518–523, 1995.

[Thrun *et al.*, 2000] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A. B. Cremers, D. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, J. Schulte, and D. Schulz. Probabilistic algorithms and the interactive museum tour-guide robot Minerva. *International Journal of Robotics Research*, Vol. 19, Number 11:972–999, 2000.

[Tournassoud, 1986] P. Tournassoud. A strategy for obstacle avoidence and its application to multi-robot systems. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, pages 1224–1229, 1986.

[Walter *et al.*, 2001] M. Walter, A. Psarrou, and S. Gong. Data driven model acquisition using minimum description length. In *Proc. of the British Machine Vision Conference*, 2001.

[Wang and Premvuti, 1995] J. Wang and S. Premvuti. Distributed traffic regulation and control for multiple autonomous mobile robots operating in discrete space. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, pages 1619–1624, 1995.

[Warren, 1990] C. Warren. Multiple robot path coordination using artificial potential fields. In *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, pages 500–505, 1990.

[WebFAIR, 2004] WebFAIR. Web access to commercial fairs through mobile agents. http://www.ics.forth.gr/webfair/, 2004.

[Weigel *et al.*, 2002] T. Weigel, J.-S. Gutmann, M. Dietl, A. Kleiner, and B. Nebel. CS Freiburg: Coordinating robots for successful soccer playing. *IEEE Transactions on Robotics and Automation*, 2002.

[Welch and Bishop, 1997] G. Welch and G. Bishop. An introduction to the kalman filter. Technical report, University of North Carolina at Chapel Hill, 1997.

[Wu, 1983] C.F.J. Wu. On the convergence properties of the EM algortihm. *The Annals of Statistics*, 11(1):95–103, 1983.

[Yannakakis *et al.*, 1979] M.Z. Yannakakis, C.H. Papadimitriou, and H.T. Kung. Locking policies: Safety and freedom for deadlock. In *Proc. of the 20th Annual Symposium on Foundations of Computer Science*, 1979.

[Zhu, 1991] Q. Zhu. Hidden Markov model for dynamic obstacle avoidance of mobile robot navigation. *IEEE Transactions on Robotics and Automation*, 7(3):390–397, 1991.

[Zilberstein and Russell, 1995] S. Zilberstein and S. Russell. Approximate reasoning using anytime algorithms. In S. Natarajan, editor, *Imprecise and Approximate Computation*. Kluwer Academic Publishers, Dordrecht, 1995.