Techniques for Multi-Robot Coordination and Navigation

Kai M. Wurm

Technische Fakultät
Albert-Ludwigs-Universität Freiburg im Breisgau

UNI FREIBURG

Dissertation zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften

Betreuer: Prof. Dr. Wolfram Burgard

27. Januar 2012

Techniques for Multi-Robot Coordination and Navigation

Kai M. Wurm

Dissertation zur Erlangung des akademischen Grades Doktor der Naturwissenschaften
Technische Fakultät, Albert-Ludwigs-Universität Freiburg im Breisgau

Dekan:	Prof. Dr. Bernd Becker
Erstgutachter:	Prof. Dr. Wolfram Burgard
Zweitgutachter:	PD Dr. Cyrill Stachniss
Tag der Disputation:	12.09.2012

Abstract

A team of mobile robots that work in parallel has the potential to finish a given task faster than a single robot. When a task can be decomposed into several independent sub-tasks, the problem can be approached using a divide-and-conquer strategy: Each robot in the team solves one sub-task and the partial solutions are then combined to complete the overall mission. One of the tasks that can be distributed well among a team of robots is mapping an environment. In this thesis, we focus on creating maps with teams of autonomous robots. There are two fundamental challenges that need to be addressed: The first challenge is the coordination of the team, in other words, assigning actions to robots so that the efficiency of the team is maximized. The second challenge is model estimation, the task of generating a map of the environment with a team of robots.

The contributions of this thesis are innovative techniques that address both of these challenges to enable teams of robots to explore environments more efficiently. In the first part of this thesis, we present approaches to coordinate teams of robots. We introduce an approach that makes use of the typical structure of buildings. It partitions the environment into regions such as rooms and corridors and then assigns robots to regions. In this way, our approach achieves a balanced distribution of the robots in the building.

Moreover, we present a method to coordinate teams of exploring robots that are able to perform actions that go beyond navigation. Such actions include opening doors or deploying other robots. Our coordination technique integrates a symbolic planning system and a robotic path planner. By explicitly considering actions other than navigating to goal positions, our approach is more flexible and more efficient than previous approaches.

In the second part of the thesis, we address challenges that arise during model estimation with multiple robots. We first present a technique to combine mapping results of individual robots into a joint map of the environment. To compute relative transformations between maps, we identify locations that occur in several of the local maps. As a result, our approach is able to estimate a consistent joint model from overlapping partial maps without relying on global estimates of robot poses.

We furthermore provide techniques for efficient 3D mapping. Our proposed mapping framework generates maps that are probabilistic, compact, and that can be created by teams of autonomous robots. Furthermore, we address the question of how semantic information can be incorporated into 3D maps. We present a hierarchical 3D mapping method that is based on knowledge about spatial dependencies in the environment.

In addition, we present an approach that robustly detects flat vegetation from laser measurements. Our method makes use of laser remission measurements and it can be used with data of most laser range finders. It reliably detects flat vegetation and enables vehicles that have not been designed to drive on vegetation to safely navigate in structured outdoor environments such as parks or campus sites.

Zusammenfassung

Seit mehr als 50 Jahren beschäftigt sich die Forschung mit der Entwicklung mobiler Roboter. In der Vergangenheit haben stationären Roboter die Arbeit in Fabriken revolutioniert, auf ähnliche Weise könnten in Zukunft mobile Roboter unsere Arbeit und unser Leben verändern. Die Fähigkeit, sich in ihrer Umgebung zu bewegen, ermöglicht mobilen Robotern, Reinigungsarbeiten zu übernehmen, Gegenstände zu transportieren, gefährliche und unzugängliche Umgebungen zu erkunden oder nach Überlebenden von Naturkatastrophen zu suchen. Mobile Roboter könnten demnach in Zukunft Aufgaben übernehmen, die anstrengend, monoton oder gefährlich sind und bisher von menschlichen Arbeitern ausgeführt werden mussten.

Eine Gruppe von mobilen Robotern bietet eine Reihe von Vorteilen gegenüber einzelnen Robotern. So ist eine Gruppe von Robotern etwa in der Lage, eine gegebene Aufgabe schneller zu bewältigen. Lässt sich die Aufgabe in unabhängige Teilaufgaben zerlegen, so kann jeder Einzelroboter zunächst eine Teilaufgabe lösen. Anschließend werden die Teillösungen zu einer Lösung des Gesamtproblems vereinigt. Beispiele für Aufgaben, die sich auf diese Weise aufteilen lassen, sind die Kartierung von Umgebungen, das Suchen nach Gegenständen oder Personen und das Reinigen einer Umgebung. Des Weiteren ist eine Gruppe von Robotern weniger fehleranfällig. Wenn mehrere Roboter eine Aufgabe gemeinsam lösen, können fehlerhafte Roboter durch andere Gruppenmitglieder ersetzt werden. Weiterhin können Robotergruppen auch Aufgaben gemeinsam bewältigen, die über die Fähigkeiten der einzelnen Roboter hinausgehen. So kann eine Gruppe von verschiedenartigen Robotern verwendet werden, die jeweils auf bestimmte Aufgaben spezialisiert sind. Eine solche Gruppe ist flexibler als ein hochspezialisierter Einzelroboter und mit wenig Aufwand kann sie auch für weitere Aufgaben eingesetzt werden.

Bestehende Mehrrobotersysteme sind oft auf Roboter beschränkt, die sich in planaren Innenräumen bewegen. Darüber hinaus wird häufig davon ausgegangen, dass alle Roboter die selben Fähigkeiten besitzen. Sieht man von solchen Einschränkungen ab, ergibt sich eine Vielzahl von neuen Einsatzmöglichkeiten, die im Folgenden erläutert werden.

Im ersten Teil dieser Arbeit werden innovative Ansätze zur Koordinierung von Robotergruppen vorgestellt. Es wird zunächst ein Verfahren präsentiert, das zusätzliches Wissen über die Struktur von Gebäuden verwendet. Mobile Roboter können viele Aufgaben allein dadurch erfüllen, dass sie über Wissen über die Position und Größe von Hindernissen verfügen. Beispielsweise kann ein Roboter auf der Basis dieses Wissens eine Umgebung kartieren, sich lokalisieren, Navigationspfade bestimmen oder sich entlang von Navigationspfaden bewegen. Häufig kann jedoch zusätzliches semantisches Wissen aus Sensormessungen abgeleitet werden. So existieren bereits Verfahren, um Räume zu erkennen und von Korridoren zu unterscheiden, um Türen und Objekte zu erkennen und

um die Beschaffenheit von Oberflächen zu analysieren. In dieser Arbeit wird untersucht, wie eine Gruppe von Robotern solches Zusatzwissen nutzen kann, um eine Umgebung effizienter zu kartieren. Konkret wird das Wissen über den typischen Aufbau von Gebäuden genutzt, um die Koordinierung von Robotergruppen zu verbessern. Solches Wissen wurden von bisherigen Ansätzen nicht berücksichtigt – häufig führte dies zu Situationen, in denen mehrere Roboter den selben Raum oder Korridor kartierten. Dies reduziert die Effizienz der Gruppe, da sich die Messungen der Roboter überschneiden. Das hier vorgestellte Verfahren teilt eine Karte des bisher explorierten Bereichs zunächst in Regionen auf, die Räumen und Korridoren entsprechen. Einzelne Roboter werden dann auf verschiedene Regionen der Umgebung verteilt. Auf diese Weise erreicht das vorgestellte Verfahren eine ausgeglichene Verteilung der Roboter im Gebäude und kann die Umgebung somit schneller explorieren als Verfahren, die dieses Wissen nicht nutzen.

Wie eingangs erwähnt, können Gruppen von verschiedenen Robotern flexibler sein als Gruppen gleichartiger Roboter. Allerdings zieht diese gesteigerte Flexibilität eine höhere Komplexität bei der Koordinierung der Roboter nach sich. Roboter können sich in ihren physischen Eigenschaften unterscheiden, wie ihrer Größe oder Geschwindigkeit. In dieser Arbeit werden jedoch Roboter betrachtet, die sich in den Aktionen unterscheiden, die sie ausführen können. Beispielsweise könnten einige Roboter in der Lage sein, Gegenstände zu bewegen oder andere Roboter abzusetzen. Solche Aktionen unterscheiden sich deutlich von Navigationsaktionen, also dem Bewegen zwischen Start- und Zielpunkten. Im Folgenden werden solche Zusatzaktionen als *symbolische Aktionen* bezeichnet. Leider gibt es keine effiziente Methode, um symbolische Aktionen auf Kosten- und Nutzenmaße abzubilden, wie sie üblicherweise in bisherigen Koordinierungsverfahren verwendet werden. In dieser Arbeit wird ein Verfahren vorgestellt, um Gruppen von Robotern zu koordinieren und dabei Aktionen zu berücksichtigen, die über die bloße Navigation hinausgehen. Das vorgestellte Verfahren integriert ein symbolisches Planungssystem und einen Pfadplaner für Roboter. Der symbolische Planer ermöglicht es dem Koordinierungsverfahren, symbolische Aktionen zu berücksichtigen. Da die Gesamtkosten der Kartierung in der Regel jedoch entscheidend von den Kosten abhängen, die durch das Bewegen der Roboter in der Umgebung entstehen, werden diese Kosten durch den effizienten Pfadplaner geschätzt. Die Verbindung eines modernen symbolischen Planungsansatzes mit einem effizienten Pfadplaner ermöglicht die explizite Berücksichtigung von symbolischen Aktionen, während gleichzeitig die Ausführungszeit minimiert wird.

Im zweiten Teil dieser Arbeit werden Fragestellungen behandelt, die bei der Erstellung von Umgebungsmodellen mit mehreren Robotern entstehen. Im Speziellen werden Verfahren vorgestellt, um mehrere Teilkarten einer Umgebung in ein Gesamtmodell zu integrieren. Weiterhin werden Ansätze präsentiert, um dreidimensionale Karten zu erstellen und es wird eine Methode vorgestellt, um Vegetation in Außenumgebungen zu erkennen.

Die Aufgabe, eine Umgebung mit mehreren Roboter zu kartieren, kann grob in zwei

Schritte unterteilt werden: Im ersten Schritt wird das Kartierungsproblem unter den einzelnen Robotern aufgeteilt, so dass jeder Roboter einen Teilbereich der Umgebung kartiert. Im zweiten Schritt werden die Teilresultate zu einer gemeinsamen Lösung zusammengefügt. Falls keine globale Schätzung der Roboterpositionen zur Verfügung steht, zieht der zweite Schritt ein Datenassoziationsproblem nach sich: Die Roboter müssen Bereiche in den Teilkarten identifizieren, die auch in anderen Teilkarten vorkommen. Können solche Bereiche gefunden werden, ermöglicht dies dem System relative Transformationen zwischen den Teilkarten zu berechnen und so eine gemeinsame Lösung zu erstellen. In dieser Arbeit werden Verfahren vorgestellt, um Transformationen zwischen überlappenden Teilkarten einer Umgebung zu berechnen. Dabei werden die beiden vorherrschenden Kartentypen betrachtet: Rasterkarten und Landmarkenkarten. Um identische Bereiche in mehreren Rasterkarten zu identifizieren, werden Daten eines Roboters in der Karte eines anderen Roboters lokalisiert. Ist diese Lokalisierung erfolgreich, wird eine relative Ausrichtung der Karten beider Roboter berechnet. Dieses Verfahren lässt sich ebenfalls einsetzen, um Gebäude mit mehreren Stockwerken zu kartieren. Um korrespondierende Bereiche in Landmarkenkarten zu bestimmen, kommt ein Triangulierungsverfahren zum Einsatz. Dieses ermöglicht die effiziente Suche nach überlappenden Bereichen in zwei Teilkarten anhand von geometrischen Merkmalen. Das vorgestellte Verfahren lässt sich mit geringem Rechenaufwand anwenden und assoziiert Landmarkenkarten zuverlässig.

Dreidimensionale Modelle der Umgebung stellen eine volumetrische Beschreibung der Umgebung zur Verfügung. Solche Modelle sind beispielsweise wichtig für fliegende Roboter und Roboter mit Manipulatoren. Während bereits robuste Verfahren existieren, um umfangreiche zweidimensionale Karten zu erstellen, gibt es bisher keine effiziente Möglichkeit, diese Verfahren auf die 3D-Kartierung zu übertragen. Um 3D-Umgebungen mit Gruppen von autonomen Robotern zu kartieren, müssen drei Voraussetzungen erfüllt werden: Zunächst muss die verwendete Karte kompakt und schnell aktualisierbar sein. Des Weiteren muss die Karte unkartierte Bereiche repräsentieren. Diese Eigenschaft ist wichtig, um eine Umgebung autonom kartieren zu können. Wichtig ist auch die Möglichkeit, Messdaten mehrerer Quellen probabilistisch zu einer Gesamtschätzung zu vereinen. Auf diese Weise kann eine Karte von mehreren Robotern oder mit mehreren Sensoren erstellt werden. Diese Arbeit stellt ein effizientes Verfahren zur 3D-Kartierung vor, das die genannten Voraussetzungen erfüllt. Eine der Neuerungen des Verfahrens ist ein verlustfreies Kompressionsverfahren, das den Speicherbedarf um mehr als 40% senken kann. Des Weiteren wird in dieser Arbeit die Frage behandelt, wie semantisches Wissen bei der 3D-Kartierung berücksichtigt werden kann, beispielsweise das Wissen über Ebenen in der Umgebung und darauf platzierte Objekte. Die vorliegende Arbeit stellt eine Methode vor, die solches Wissen verwendet, um eine hierarchische 3D-Karte der Umgebung zu erstellen.

Die meisten autonomen Fahrzeuge, wie Transportfahrzeuge, autonome Rollstühle oder

autonome Autos, wurden für das Fahren auf Straßen und befestigten Wegen entwickelt. Innerhalb von Parks oder auf Betriebsgeländen lassen sich befestigte Wege nicht in jedem Fall sicher erkennen. Solche Wege sind oft vergleichsweise schmal und grenzen an bewachsene Bereiche an. Das Befahren solcher bewachsenen Bereiche stellt eine Gefahr für autonome Fahrzeuge dar. Einerseits könnten die Räder des Fahrzeugs darin stecken bleiben. Andererseits könnte beim Befahren von Gras oder ähnlicher Vegetation die Bodenhaftung reduziert werden, was zu einer Beeinträchtigung der Positionsschätzung führen würde. Beide Fälle stellen ein Risiko für die Sicherheit des Fahrzeugs dar. Ist das Fahrzeug jedoch in der Lage, flache Vegetation zu erkennen, so kann es sein Navigationsverhalten verbessern, in dem es bewachsene Bereiche vermeidet. In dieser Arbeit wird ein Verfahren vorgestellt, um flache Vegetation aus den Daten von Lasermesssystemen zu erkennen. Das Verfahren nutzt die Reflektionseigenschaften von Vegetation, um eine sichere Erkennung zu erreichen. Die Erkennung wird durch eine support vector machine realisiert und geschieht auf der Basis von Beispielmessungen. Zuverlässige Beispieldaten werden durch die Verwendung eines zweiten Vegetationsklassifikators erzeugt. Als Eingabe verwendet dieser Klassifikator gemessene Vibrationen, die während der Fahrt über den jeweiligen Untergrund entstehen. Das vorgestellte Verfahren erreicht hohe Erkennungsraten von über 99%. Auf diese Weise ermöglicht es Fahrzeugen, die nicht für das Befahren von Vegetation geeignet sind, die sichere Navigation in strukturierten Außenumgebungen.

Zusammenfassend werden in dieser Arbeit die folgenden Fragestellungen behandelt:

- Wie kann semantisches Wissen verwendet werden, um die Koordinierung von Robotergruppen zu verbessern?
- Wie können Gruppen von verschiedenartigen Robotern effizient koordiniert werden?
- Wie kann eine Gruppe von Robotern gemeinsam ein konsistentes Modell der Umgebung erstellen?
- Wie können mobile Roboter auf effiziente Weise dreidimensionale Modelle der Umgebung erstellen?
- Wie können Roboter effizient und sicher in strukturierten Außenumgebungen navigieren?

Acknowledgments

This thesis would not have been possible without the support of a number of people and I would like to thank everybody that contributed to this work.

First of all, I would like to thank my advisor Wolfram Burgard. I am thankful for the opportunities he gave me and his support in every regard of my scientific work. He shaped my view for what is important in science, inspired and encouraged me, and most of all helped me to focus on the right questions.

Furthermore, I would like to thank my co-advisor Cyrill Stachniss. Many projects would not have been possible without his prior work on multi-robot systems. Over the years, we worked together on many ideas, projects, and papers and I am thankful that he shared his creativity, pragmatism, and his intuition for efficient solutions.

I would like to thank all co-authors and colleagues that contributed to this thesis. Needless to say, many projects could not have been realized if it wasn't for fruitful collaborations with fellow researchers. My special thanks for our joint work during the last years go to Christian Dornhege, Armin Hornung, Giorgio Grisetti, and Rainer Kümmerle. For their contributions to joint publications I would furthermore like to thank Maren Bennewitz, Alexander Cunningham, Frank Dellaert, Patrick Eyerich, Daniel Hennes, Dirk Holz, Michael Karg, Henrik Kretzschmar, and Bernhard Nebel. I thank Kurt Konolige, Radu Bogdan Rusu, and Willow Garage Inc. for giving me the opportunity to apply my work in new fields. For their collaboration on research projects, for providing datasets, and for helpful discussions I thank Kai Oliver Arras, Mark Edgington, Felix Endres, Dieter Fox, Barbara Frank, Lutz Frommberger, Marc Gissler, Slawomir Grzonka, Jürgen Hess, Dominik Joho, Gil Jones, Michael Kaess, Yohannes Kassahun, Norman Kohler, Boris Lau, Jörg Müller, Christian Plagemann, Axel Rottmann, Michael Ruhnke, Gabe Sibley, Jürgen Sturm, Christoph Sprunk, Bastian Steder, Rudolph Triebel, and Diedrich Wolter.

For their help on administrative and technical matters, my thanks go to Susanne Bourjaillat, Michael Keser, and Dagmar Sonntag. Furthermore, I thank the University of Freiburg and the members of the SFB/TR-8 for providing me with a professional, interdisciplinary, and enjoyable work environment.

Finally, I would like to thank my family, my friend Janne Schulz, and my wife Andrea for their unconditional support throughout the years.

This work has been partly supported by the German Research Foundation (DFG) under contract number SFB/TR-8 and through the Gottfried Wilhelm Leibniz Program.

To my loving wife and family

Contents

Chapter 1
Introduction

For more than 50 years, researchers have worked towards the development of autonomous mobile robots. Just as stationary manipulation robots have revolutionized factory assembly lines, mobile robots have the potential to change the way humans live and work. The ability to move around allows mobile robots, for example, to clean the environment, to fetch and deliver goods, to examine hostile or remote areas, to search and rescue victims in a disaster scenario – in short: to accomplish strenuous, repetitive, or dangerous tasks that previously had to be done by human workers.

A team of multiple mobile robots that work in parallel offers a number of advantages over single robots systems. Multiple robots have the potential to finish a given task faster than a single robot. When a task can be decomposed into several independent sub-tasks, the problem can be approached using divide-and-conquer strategies: Each robot in the team solves one sub-task and the partial solutions are then combined to complete the overall task. Examples of such tasks include mapping, searching for objects or persons, and covering of environments. Furthermore, teams of robots introduce redundancy into the system that makes it more robust to failure. When multiple robots jointly solve a task, failing robots can be replaced by other teammates thus avoiding the single point of failure of systems in which only one robot is used. Moreover, multiple robots can join their efforts to accomplish tasks that go beyond the ability of each individual robot. Instead of equipping a single robot with all capabilities that a given task requires, a heterogeneous team of robots with different abilities can be employed. Such a team is more flexible than one highly specialized robot and with little effort the team can be reconfigured to accomplish other tasks.

Existing multi-robot approaches often consider teams of robots that operate in planar indoor environments and they often assume that all robots have the same capabilities. As we will discuss in the following, extending this basic setting greatly enhances the potential of teams of robots.

Mobile robots can solve many tasks solely based on geometric information. For example, tasks such as mapping, localization, path planning, and navigation can be accomplished solely based on information about the positions and sizes of obstacles. Yet, additional information may help robots to be more efficient. Often, semantic knowledge can be derived from sensor measurements and there exist approaches to identify, for example, rooms, corridors, and doors, known objects, or the type of terrain surface. In this thesis, we investigate how a team of robots can make use of information about the structure of a building during exploration in order to perform this task more efficiently.

As mentioned above, heterogeneous teams of robots offer a greater flexibility than homogeneous teams. However, the increase in flexibility also leads to an increase in the complexity of their coordination. Heterogeneous teams may differ in their physical properties or the type of terrain they are able to traverse. They may also differ in the actions they are able to perform. For instance, robots might be equipped with manipulators, or they could be able to deploy other robots. Such actions stand in contrast to navigation actions that move robots to a given goal position and we will refer to them as *symbolic actions* in the following. Unfortunately, it is not straightforward to efficiently map symbolic actions to cost or utility measures, such as those used by popular coordination approaches. We will address the problem of coordinating teams of robots that are able to perform actions that go beyond navigation.

The task of mapping an environment with a team of mobile robots can be rephrased as a divide-and-conquer problem. In the first step (divide), the mapping problem is distributed among the robots and each robot solves a so-called local mapping problem. In the second step (conquer), the team combines the local results into a joint solution. If no global estimates of the robot positions are available, the conquer step entails a data association problem. The team has to identify locations that occur in several of the local robot maps. Finding such associations between local maps then allows to compute relative transformations and to create a joint solution. We provide techniques to compute transformations between overlapping maps for the two prevailing map representations, grid maps and landmark maps.

Three-dimensional models provide a volumetric representation of space which is important, for example, for flying robots and for robots that are equipped with manipulators. While techniques exist to estimate large 2D maps, there is no efficient extension of these methods to the case of 3D mapping. We identified three prerequisites that allow 3D maps to be acquired autonomously by teams of robots. First, the map representation needs to be compact and updates have to be computationally efficient. Second, unmapped areas have to be encoded in the map. This is important when the map is created autonomously to identify potential sensing locations. Third, the map has to support the probabilistic fusion of data from multiple robots or multiple sensors. A consistent joint model can then be generated by integrating data of multiple sources. We present an efficient mapping technique that addresses these three challenges. Furthermore, we address the question of

how semantic information can be incorporated into 3D maps, for example, knowledge about supporting planes. We present a 3D mapping technique that makes use of such information by constructing a hierarchy of 3D maps.

Most autonomous systems that navigate outdoors, such as transportation vehicles, surveillance robots, or autonomous cars have been designed to drive on streets and paved paths. In urban areas these drivable surfaces can be hard to detect. Especially paths in parks or campus sites are often narrow and there may be no detectable boundary between the path and neighboring regions that contain vegetation. Traversing vegetation increases the risk that the vehicle gets stuck or that wheel slippage leads to errors in the location estimation process. Both of these cases pose a danger to the safety of the system. If the vehicle is able to detect vegetation, however, it is able to avoid such areas and thus improve its navigation in structured outdoor environments. We will present an approach that robustly detects flat vegetation from laser measurements. The approach makes use of laser remission measurements and can be used with most laser sensors commonly used on mobile robots.

In sum, we have identified the following open research questions with respect to multi-robot systems:

- How can semantic knowledge be utilized to improve the coordination of multi-robot systems?
- How can heterogeneous teams of robots be coordinated efficiently?
- How can a team of robots create a consistent map of the environment?
- How can 3D environments be efficiently mapped by mobile robots?
- How can robots efficiently navigate in structured outdoor environments?

1.1. Contributions

The contributions of this thesis are innovative approaches that address the research questions discussed above. We provide techniques to coordinate teams of robots that go beyond the state-of-the-art by taking into account semantic information and by explicitly considering heterogeneous teams of robots. Furthermore, we present techniques that improve multi-robot navigation and mapping. The following approaches are the key contributions of this thesis:

- An approach to explore buildings with teams of robots that uses a partitioning of the environment into rooms and corridors (Chapter 2).
- A technique that applies temporal symbolic planning to coordinate heterogeneous teams of robots and that explicitly considers symbolic actions (Chapter 3).

- Two solutions to the data association problem which arises in multi-robot SLAM (Chapter 4).
- An efficient technique to model 3D environments probabilistically which is suited for 3D exploration (Chapter 5).
- An approach that robustly identifies flat vegetation in outdoor environments (Chapter 6).

1.2. Outline

This thesis is organized in two parts. Part I addresses the problem of coordinating teams of robots while Part II presents approaches to estimate models of the environment. These two techniques are the fundamental building blocks of an efficient multi-robot system: coordination enables the team of robots to exploit its full potential while a model of the environment is the basis of efficient coordination.

In Chapter 2, we present a technique to coordinate teams of exploring robots that makes use of semantic information. We analyze the structure of indoor environments and partition the environment into regions such as rooms and corridors. Previous approaches often generate exploration targets on the frontier between mapped and unmapped areas. In typical environments, multiple exploration targets are often generated in the same room or corridor. When several robots explore the same physical region, however, there often is a considerable overlap of the sensor measurements that leads to inefficiency. To overcome this source of inefficiency, our coordination method identifies regions such as rooms and clusters targets according to the structure of the environment. We assign robots to regions instead of assigning them to target locations directly, thus achieving a balanced distribution of the team.

Heterogeneous teams of robots are considered in Chapter 3. We present a technique to coordinate teams of robots that integrates a symbolic planning system and a robotic path planner. This combination allows our system to consider planning problems that include symbolic actions such as opening doors or deploying other robots. To coordinate a team of robots, we generate a symbolic description of the current state of the system and of the goal state. These descriptions serve as input to the symbolic planner. To solve the coordination problem, the symbolic planner uses the path planner to efficiently plan paths and estimate travel costs for the robots. Our approach then uses the solution determined by the planning system to compute actions that are send to the individual robots. We apply our approach to two settings. First, we consider teams of robots that are able to deploy and pick up smaller robots. Second, we simulate a disaster scenario where the task is to clear blockades and to perform pre-defined actions at certain critical locations in the environment.

In the second part of this thesis, we present techniques to estimate models of the environment. In Chapter 4, we present an approach to simultaneous localization and mapping (SLAM) with multiple robots. To extend single-robot SLAM approaches to multi-robot systems, we align the maps and trajectories of the individual robots. We make use of a graphical formulation of the SLAM problem, that maintains graphs of poses and observations for each robot. To connect the initially independent graphs, we solve a data association problem: Areas that have been covered by multiple robots are identified from the observations of the robots. From the corresponding matches, we extract alignment constraints between the trajectories of the robots. The resulting global graph is then optimized to estimate a consistent model. We present two methods to solve the data association problem in distributed systems. The first approach matches grid maps while the second approach matches landmark maps. Our grid-based approach applies Monte Carlo localization and performs global pose estimation. By localizing one robot in the map of another robot, it associates positions in multiple grid maps. The second approach we present aligns maps of landmark positions. It computes a Delaunay triangulation of the maps and then uses geometric features to efficiently search for similar triangles. Matching triangles are then used to compute alignment constraints between pairs of overlapping maps.

We present efficient techniques to estimate 3D models in Chapter 5. Our mapping approach meets three goals that make it suitable for 3D exploration: It models the environment probabilistically, it represents unmapped areas, and it is efficient both in runtime and in its memory requirement. We make use of probabilistic state estimation techniques and efficient data structures to achieve these goals. A lossless map compression method is introduced to keep 3D models compact. Furthermore, we present an approach that uses semantic information to construct hierarchies of 3D maps. In addition to that, we present an information-driven exploration algorithm to learn 3D models autonomously.

In Chapter 6, we present an approach to detect vegetation from laser measurements. In structured outdoor environments, such as parks or campus sites, large areas are often covered with grass. Our approach reliably detects flat vegetation from remission values of laser scanners. To predict the terrain type, we use a support vector machine. The input to this classifier are individual laser measurements consisting of the remission value, the distance to the surface, and the angle of incidence. To avoid the need to label training data manually, we label example sets in a self-supervised fashion by means of a pre-trained vibration-based terrain classifier. Our classifier can be used to improves traversability estimates and therefore enables vehicles that have not been designed to drive on vegetation to safely navigate in structured outdoor environments. Furthermore, we can obtain a segmentation of such environments into regions by clustering vegetation detections.

1.3. Publications

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

- K.M. Wurm, H. Kretzschmar, R. Kümmerle, C. Stachniss, and W. Burgard. Identifying Vegetation from Laser Data in Structured Outdoor Environments. In *Journal of Robotics and Autonomous Systems*, Special issue on semantic mapping, 2012. Conditionally accepted for publication.

- A. Cunningham, K.M. Wurm, W. Burgard, and F. Dellaert. Fully Distributed Scalable Smoothing and Mapping with Robust Multi-robot Data Association. In *Proc. of the IEEE/RSJ International Conference on Robotics & Automation (ICRA)*, Saint Paul, MN, USA, 2012. Accepted for publication.

- K.M. Wurm, D. Hennes, D. Holz, R.B. Rusu, C. Stachniss, K. Konolige, and W. Burgard. Hierarchies of Octrees for Efficient 3D Mapping. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, CA, USA, 2011.

- K.M. Wurm, C. Dornhege, P. Eyerich, C. Stachniss, B. Nebel, and W. Burgard. Coordinated Exploration with Marsupial Teams of Robots using Temporal Symbolic Planning. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, 2010.

- A. Hornung, K.M. Wurm, and M. Bennewitz. Humanoid Robot Localization in Complex Indoor Environments. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, 2010.

- K.M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: A Probabilistic, Flexible, and Compact 3D Map Representation for Robotic Systems. In *Proc. of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, Anchorage, USA, 2010.

- M. Karg, K.M. Wurm, C. Stachniss, K. Dietmayer, and W. Burgard. Consistent Mapping of Multistory Buildings by Introducing Global Constraints to Graph-based SLAM. In *Proc. of the IEEE/RSJ International Conference on Robotics & Automation (ICRA)*, Anchorage, USA, 2010.

- K.M. Wurm, C. Stachniss, and G. Grisetti. Bridging the Gap Between Feature- and Grid-based SLAM. In *Journal of Robotics and Autonomous Systems*, 58(2):140 - 148, 2010.

- K.M. Wurm, R. Kümmerle, C. Stachniss, and W. Burgard. Improving Robot Navigation in Structured Outdoor Environments by Identifying Vegetation from Laser Data. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, St. Louis, MO, USA, 2009

- K.M. Wurm, C. Stachniss, and W. Burgard. Coordinated Multi-Robot Exploration using a Segmentation of the Environment. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nice, France, 2008

1.4. Contributions to Open-Source Software

The approach presented in Chapter 5 has been made available as a self-contained C++ library under the BSD open source license and The library, called OctoMap, can be obtained from `http://octomap.sf.net`. It received a considerable uptake from the scientific community and has been used in a variety of research projects. It has also been integrated into the Robot Operating System (ROS) where it is applied, for example, to perform 3D navigation.

1.5. Collaborations

Parts of the approaches presented in this thesis have been developed in collaboration with other researchers. The coordination approach presented in Chapter 3 was jointly developed with Christian Dornhege, who provided his expertise on the modular temporal planning framework TFD/M. I co-supervised several bachelor and master theses – the grid-based multi-robot SLAM technique presented in Chapter 4 was originally developed in the master thesis of Michael Karg. The landmark-based multi-robot SLAM technique was developed together with Frank Dellaert and Alexander Cunningham, who provided the optimization framework for the real-world experiments. The 3D mapping approach OctoMap, presented in Chapter 5, was developed in close collaboration with Armin Hornung. The hierarchical extension and 3D exploration system was developed in the course of a research collaboration with Willow Garage Inc. and the tabletop mapping system was developed with the support of Radu Bogdan Rusu and Kurt Konolige. Finally, the laser-based vegetation classifier was developed in collaboration with Rainer Kümmerle and Henrik Kretzschmar.

Part I.

Coordinated Exploration

Chapter 2
Multi-Robot Exploration using a Segmentation of the Environment

2.1. Introduction

There are several applications in which it is advantageous for mobile robots to create a map of their environment autonomously. Autonomous exploration is most useful in scenarios where human operators cannot command the robot efficiently such as in planetary exploration or in disaster missions. Exploring an environment with a team of robots instead of a single robot offers a number of advantages (Cao et al., 1997; Dudek et al., 1996; Gerkey and Matarić, 2004). Clearly, multiple robots that work in parallel have the potential to reduce the overall mapping time. If the team is coordinated so that each robot maps a different part of the environment then a significant speedup can be achieved (Guzzoni et al., 1997). Another important aspect is that a single robot introduces a single point of failure into an exploration system. Especially in hostile environments, it may be impossible to repair defective robots or to recover a robot that became immobilized due to unforeseen circumstances. A team of robots will therefore be more robust to this type of failure.

In this chapter, we consider the problem of efficient exploration with teams of mobile robots. The goal of our coordination approach is to minimize the overall time required to cover the environment completely. The task of coordinating a team of robots during exploration can roughly be separated into two tasks. First, one needs to identify possible exploration actions for the team of robots. Typically, such actions correspond to sensing

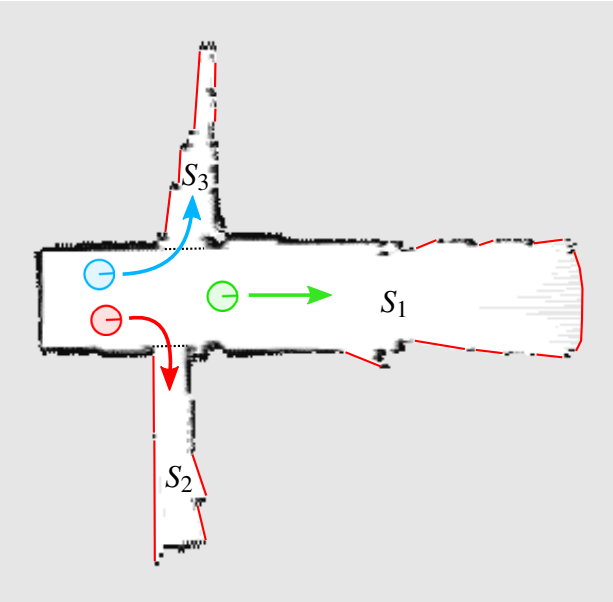


Figure 2.1: Typical coordination of robots obtained by assigning them to different segments of the partially explored map. Segment boundaries are visualized as dashed black lines and exploration frontiers are depicted as red lines.

locations in the environment and we will refer to them as *exploration targets* in the following. After determining the set of targets, we need to assign them to the team of robots. In most cases, there will be more targets than robots. Since our goal is to explore the environment as fast as possible, we need to choose an assignment that covers unmapped areas fast and at the same time avoids redundant work among the team. It is this assignment strategy which affects the efficiency of the robot team the most and we will present a novel strategy that takes into account the structure of the environment.

Most exploration approaches apply the target generation method that was proposed by Yamauchi *et al.* (1998). This approach computes so called frontiers, which are defined as the borders between the explored and the unexplored space. The method then defines exploration targets along those frontiers. Many previous coordination approaches use a cost function to assign robots to frontier targets. Popular approaches define cost functions that take into account the expected path costs or travel time to the target as well as an utility function that covers aspects such as the expected gain in information (Singh and Fujimura, 1993a; Zlot et al., 2002; Gerkey and Matarić, 2002; Burgard et al., 2005; Stachniss et al., 2006). These coordination strategies have in common that they consider individual exploration targets in the environment. In indoor environments, however, several frontier targets are often generated close to each other. Due to sensor occlusions that result from furniture and other obstacles, multiple frontiers often exist in the same room. An illustration of this problem is given in Figure 2.1. Because of occlusions, both rooms S_2 and S_3 as well as the corridor contain several exploration frontiers. An exploration

strategy that focuses on individual targets and ignores this effect may lead to comparably inefficient exploration behavior. Several robots may be assigned to the same physical region and this will often result in a considerable overlap of the sensor measurements in the team. Furthermore, robots that operate close-by run into the risk of affecting each other's measurements and of blocking each other's paths. In the example given in Figure 2.1, the worst strategy would be to assign all robots to the frontiers in S_3.

In the following, we introduce a new online coordination strategy for multi-robot exploration. It assigns robots to regions instead of directly assigning them to exploration targets. In indoor environments, we define regions as separate parts of the building such as rooms and corridors. We apply a segmentation technique to divide a map of the already explored area into separate regions. By assigning robots to regions, we avoid exploring the same region with multiple robots. This strategy distributes the robots over the environment more effectively than previous approaches. The balanced distribution leads to a reduction of redundant work and it also avoids interferences between robots. As a result, the exploration time is significantly reduced. The key idea of our approach is summarized in the illustration in Figure 2.1. Our approach assigns each of the three robots in the example to a different region and thus achieves a good distribution of the team in the environment.

This chapter is organized as follows. In the next section, we introduce the method of generating exploration targets using the frontiers approach. In Section 2.3, we describe the Hungarian target assignment method which we use in our coordination approach. In Section 2.4, we introduce a graph-based approach for map segmentation and in Section 2.5 we present our multi-robot coordination technique. Section 2.6 presents simulated as well as real world experiments conducted to evaluate our approach. Finally, we give an overview of the related work in Section 2.7.

2.2. Exploration Targets

To explore an environment with a team of robots, most coordination approaches generate a set of possible exploration targets and assign robots to a subset of those targets. An exploration target is defined as a specific location in the environment and, in general, also includes the orientation of the robot or sensor. To explore a target, the robot approaches the specified position and takes a sensor measurement there.

The method of generating exploration targets depends on the representation of the environment. The most popular representation for exploration is the occupancy grid map that discretizes the environment into a grid of map cells (Moravec, 1988). A binary Bayes filter is applied to estimate the occupancy of each cell. Each cell is initialized as unknown. When sensor measurements are integrated, the estimate of the cells will converge to being

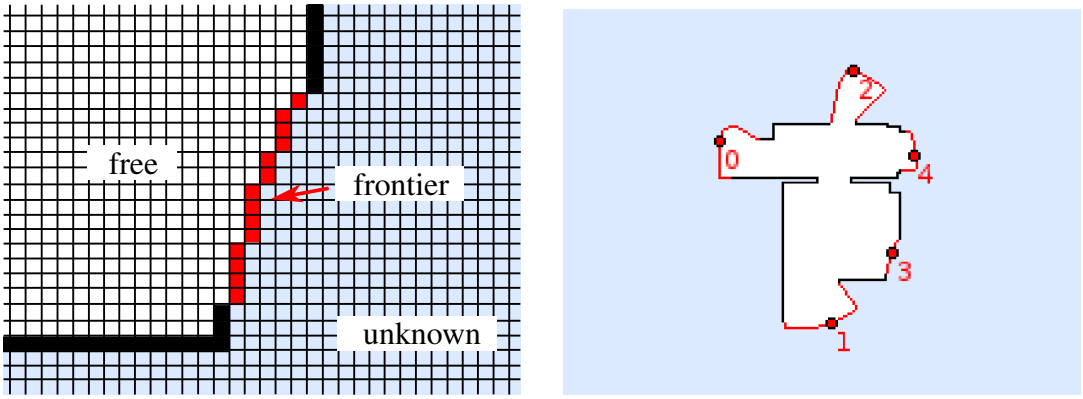


Figure 2.2: Frontier extraction. Left: Frontier cells are determined at the boundary between *free* and *unknown* map cells. Unknown cells are shown light blue, free cells are depicted in white and occupied cells in black. Frontier cells are shown in red. Right: Five exploration targets, visualized as red circles, are generated at frontiers.

occupied or *free* depending on whether the corresponding area in environment contains an obstacle.

The information about which parts of the environment have been sensed by a robot and which parts are unknown is used by the frontiers approach (Yamauchi, 1997) to generate exploration targets. It identifies unknown cells that are adjacent to cells which have been sensed as being free before. Each of these frontier cells is reachable by one of the robots and leads to unmapped areas. In practice, adjacent frontier cells are often combined into a single exploration target as long as they lie within the field of view of the robot's sensor. This process is illustrated in Figure 2.2.

2.3. Target Assignment using the Hungarian Method

Once a set of exploration targets has been determined, our coordination algorithm assigns these targets to the team of robots. Previous approaches applied either iterative or batch assignment methods: Iterative methods assign targets to one robot after the other while batch approaches assign targets to all robots at once. Iterative techniques, for example the one proposed by Burgard *et al.* (2005), allow costs and gains of targets to be adapted after each assignment. Batch approaches, for example the method presented by Ko *et al.* (2003), compute all costs and gains once but based on these quantities they are able to determine the optimal assignment. In experimental evaluations, both approaches achieved a similar performance whenever constant exploration costs could be assumed. In our approach, we assume the cost to explore a region to be constant and therefore perform a batch assignment.

Kuhn (1955) introduced a method to assign a set of *jobs* to a set of *machines* given a fixed cost matrix. This method, which is often referred to as the Hungarian method,

computes an assignment that minimizes the total cost. Starting from an $n \times n$ cost matrix which represents the cost of all individual assignments of n jobs to n machines, the Hungarian method is able to determine the cost-optimal solution. The algorithm can be summarized as follows (Stachniss, 2006):

1. Identify the minimal element in each row and compute a reduced cost matrix by subtracting this element from each element in the corresponding row. Afterwards, reduce the costs in each column in the same way.

2. Find the minimal number of horizontal and vertical lines required to cover all zeros in the matrix. In case exactly n lines are required, the optimal assignment can be derived from the positions of the zeros covered by the n lines. Otherwise, continue with Step 3.

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

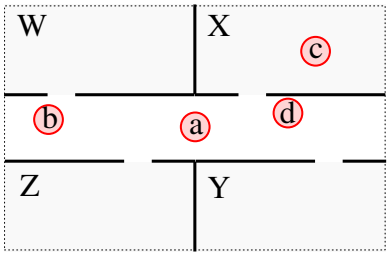
An illustration of the algorithm is given in Figure 2.3. Here, a team of four robots is assigned to four exploration areas.

The computationally expensive part lies in finding the minimum number of lines covering the zero elements (Step 2). The overall assignment algorithm has a complexity of $O(n^3)$ (Stachniss, 2006). In practice, coordination problems that involve several hundreds of robots and tasks can be solved online (Gerkey and Matarić, 2004).

We apply the method described above to assign a set of target locations (tasks) to a team of robots (machines). A straightforward way of generating the cost matrix is, for example, to estimate the length of the path that each robot has to travel to reach the corresponding target location. Since the implementation of the Hungarian method described above requires the number of jobs and machines to be identical, we may need to adapt the cost matrix to fulfill this requirement. Whenever there are more targets than robots, we expand the cost matrix by introducing virtual robots which will result in target locations that are not approached by any of the real robots. In those cases where there are more robots than targets we duplicate existing targets and hence multiple robots will be assigned to the same target.

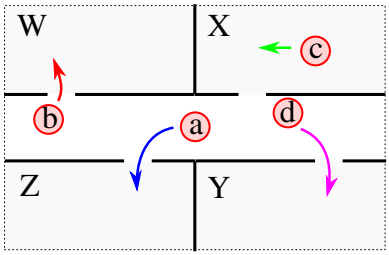
2.4. Map Segmentation for Exploration

The coordination approach presented in this chapter makes use of the structure of indoor environments. A typical office building consists of rooms and corridors and each doorway can be understood as a natural boundary between these regions. During exploration,



targets

robots	W	X	Y	Z
a	3	2	3	2
b	2	5	6	3
c	7	1	3	5
d	6	2	3	5



targets

robots	W	X	Y	Z
a	1	0	0	**0**
b	**0**	3	3	1
c	6	**0**	1	4
d	4	0	**0**	3

Figure 2.3: Illustration of the Hungarian method. Top row: four robots (a,b,c,d) need to be assigned to four areas (W,X,Y,Z). The costs for reaching each area is given in the cost matrix depicted on the right. Bottom row: The cost matrix is transformed using the Hungarian method and an assignment of robots to areas is computed.

we partition the map of the building into such regions. Several frontier targets may be generated in the same room or corridor but by considering the regions we identified, we are able to cluster frontier targets according to the structure of the environment. By assigning robots to regions instead of assigning them to frontier targets directly, our coordination approach achieves a balanced distribution of the team. In the following, we describe a segmentation method that partitions the grid map of a building into structurally important regions. The segmentation is based on the detection of narrow passages in the grid map that correspond to doorways.

2.4.1. Voronoi Graphs

We partition the map of the building by applying graph-based image segmentation methods. Several methods to segment grid maps have been proposed that are based on graph cuts (Kuipers and Byun, 1991; Thrun, 1998; Beeson et al., 2005; Zivkovic et al., 2006; Friedman et al., 2007). In this context, Voronoi graphs (Choset et al., 2000) are a popular method to represent environments. Let C denote the free-space of a given grid map m. To compute the Voronoi Graph $G(m) = (V,E)$ of m, we consider the set $O(p,m)$ that contains the closest obstacle points for each point p in C. The nodes of the Voronoi graph are defined as the set of points in C for which there are at least two obstacle points with equal minimal distance:

$$V = \{p \in C \mid |O(p,m)| \geq 2\} \tag{2.1}$$

For each pair of nodes in $G(m)$ we add an (undirected) edge if their corresponding points in m are adjacent:

$$E = \{(p,q) \mid p,q \in V,\ p \text{ adjacent to } q \text{ in } m\} \tag{2.2}$$

The Voronoi graph can be generated from metric maps of the environment such as occupancy grid maps (Choset and Burdick, 1995; Thrun, 1998). Let N be the number of cells in the map. The straightforward solution would be to compute the set $O(p,m)$ for each cell in the map. We would need to perform a nearest neighbor search for each cell. Using search trees, each neighbor search takes $O(\log N)$ time and the tree can be set up in $O(N \log N)$, thus the overall runtime complexity of the naive method is $O(N \log N)$. However, the complexity can be reduced to $O(N)$ using image manipulation methods. The Voronoi graph can be efficiently constructed by first applying the Euclidean distance transform (Meijster et al., 2000) to an occupancy grid map. The result of this transform is stored in a distance map which for each cell in the occupancy grid map holds the distance to the closest obstacle. A Voronoi graph can then be constructed using skeletonization on the distance map. This method computes local maxima in the distance map. We use an approach similar to the one proposed by Niblack *et al.* (1990). Both the distance

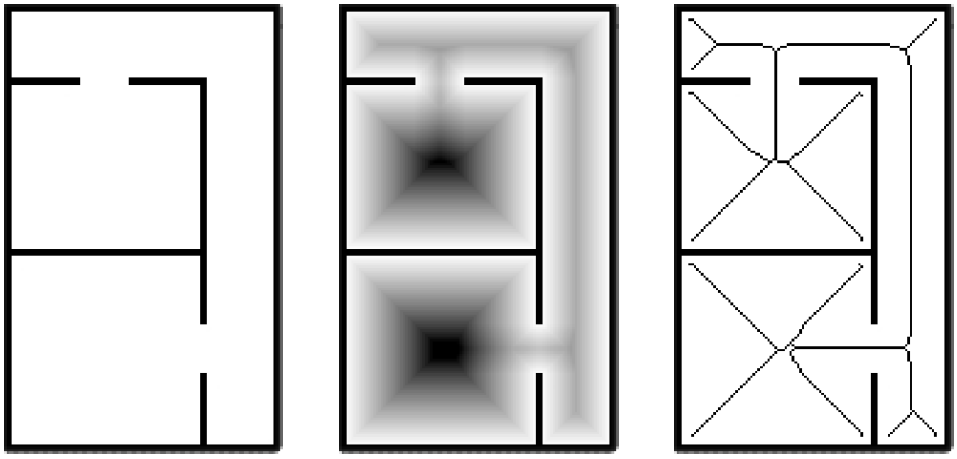


Figure 2.4: Generation of the Voronoi graph. Left: Example grid map. Center: Grid map superimposed with visualization of the Euclidean distance transform (the darker a point is, the larger the distance to the closest obstacle). Right: Grid map superimposed with result of skeletonization of the distance transform.

transform and skeletonization have a runtime complexity of $O(N)$. Figure 2.4 illustrates the process of generating a Voronoi graph from an occupancy grid map.

2.4.2. Graph Reduction

The initial Voronoi graph computed from a partially explored environment contains a large number of nodes and edges that can be safely ignored to compute a segmentation of the environment as specified above. For this reason we compute a reduced Voronoi graph using the method that is summarized in Algorithm 1. The Voronoi graph computed from a grid map often consists of several unconnected sub graphs. In general, there will be one large connected component that corresponds to the already explored area and there may be several smaller components that are created in areas that have not been sufficiently mapped. In a first pre-processing step, we therefore restrict the graph to the biggest connected component of the initial graph. The connected components of a graph can be efficiently computed using graph-search (Hopcroft and Tarjan, 1973). This step removes parts of the graph that are not accessible from the already explored area. An example can be seen in Figure 2.5. Here, small unconnected sub graphs can be seen on the top right part of the initial graph (left figure) that are removed in the reduced graph (right figure).

In the following steps, we remove nodes from the graph that are not necessary to compute a partition of the grid map. Let the degree of a node denote the number of edges associated with the node in the undirected graph. We remove nodes and associated edges with degree 2 (bridging nodes) from the graph that are not a local minimum with respect to the distance to the closest obstacle points in the map m used for generating the graph. This distance can be computed for any Voronoi node $v \in E$ using the sets $O(v,m)$ defined above and is returned by the method $mindist(v)$. We preserve nodes that are local minima

Algorithm 1 Computation of Reduced Voronoi Graph

Input:
$G = (V, E)$, initial graph
Output:
$G' = (V', E')$, reduced graph

1: $G' = (V', E') \leftarrow \text{biggestConnectedComponent}(G)$
2: *// remove redundant nodes with degree 2*
3: **for** all nodes $v \in \{v \in V' \mid \text{degree}(v) = 2\}$ **do**
4: $n_l = \text{leftNeighbor}(v)$
5: $n_r = \text{rightNeighbor}(v)$
6: **if** $(\text{mindist}(n_l) \leq \text{mindist}(v)) \vee (\text{mindist}(n_r) \leq \text{mindist}(v))$ **then**
7: $V' \leftarrow V' \setminus v$
8: $E' \leftarrow E' \setminus \{(n_l, v), (v, n_r)\}$
9: $E' \leftarrow E' \cup \{(n_l, n_r)\}$
10: **end if**
11: **end for**
12: *// remove nodes with degree 1*
13: **for** all nodes $v \in \{v \in V' \mid \text{degree}(v) = 1\}$ **do**
14: $V' \leftarrow V' \setminus v$
15: $E' \leftarrow E' \setminus \{(v, n) \mid \exists n, (v, n) \in E'\}$
16: **end for**

since they are used in the partitioning step described below. In the algorithm, we use *leftNeighbor*(v) to access the the first neighbor of a degree-2-node and *rightNeighbor*(v) to access the second neighbor. In the last step, we remove nodes that are of degree 1 (endpoints). Such nodes usually lead to the boundary of the map and thus can be ignored in the case of map segmentation. The result of the reduction algorithm applied to an exemplary graph can be seen in Figure 2.5.

2.4.3. Graph Partitioning

After generating the Voronoi graph, we are now interested in creating a partitioning of the graph into k disjoint sets $V_1, V_2, \ldots, V_k$ with

$$V = \bigcup_{i=1}^{k} V_i \tag{2.3}$$

such that each cluster of nodes V_i is a segment of the environment that we can assign robots to. Thrun *et al.* (1998) proposed to segment the Voronoi graph of indoor environments at so-called critical points. In their approach, critical points are defined as those nodes in the graph at which the distance to the closest obstacle in the map is a local minimum. This is usually the case in doorways and other narrow passages.

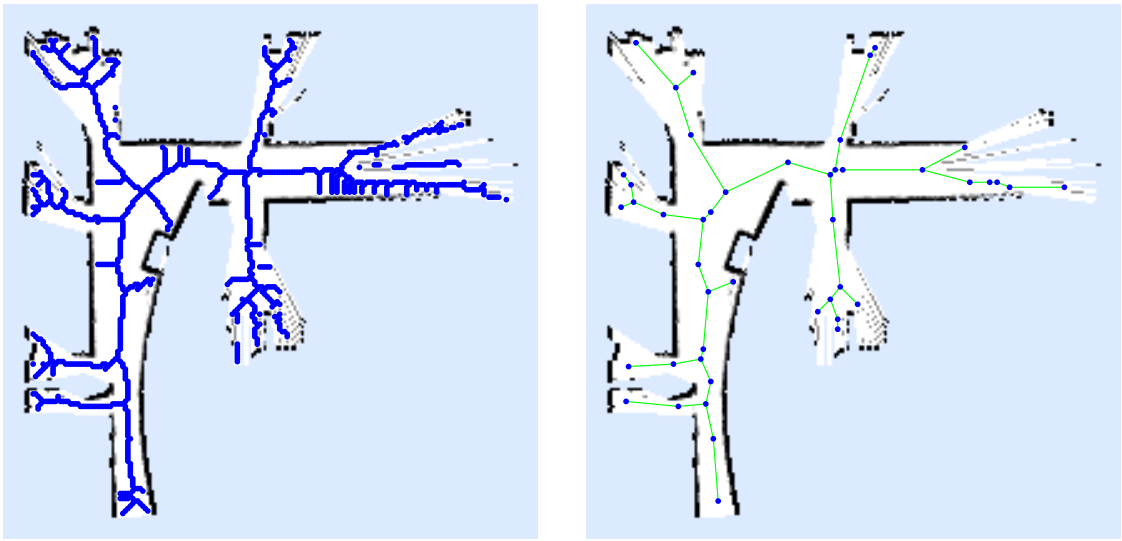


Figure 2.5: Example of a reduced Voronoi graph in a small section of an environment. The partial grid map is superimposed with the initial Voronoi graph (left) and the reduced Voronoi graph (right). Nodes are depicted as blue dots and edges are depicted as green lines. Light blue areas have not been explored yet.

Whereas this criterion can be used to reliably identify doorways, it also generates a number of false positive candidates in cluttered environments (see Figure 2.6 a). To eliminate false positive boundary detections, we constrain the generation of critical points in two ways: First, critical points are required to be of degree 2 (two edges) and they need to have at least one neighboring node of degree 3 (a junction node). Second, we require the points to lead from known into unknown areas. Intuitively, the first constraint ensures that critical points in the graph correspond to decision point in the topology (junctions). The second constraint is motivated by the observation that segments which do not contain unknown areas can safely be ignored in an exploration task.

For each node in the graph our segmentation approach computes the distance to the closest reachable unknown cell. This value is then used to verify that critical points are created between parts of the environment that only contain known areas and parts that also contain unknown areas. This can be done efficiently using a method that is similar to the computation of the Euclidean distance transform.

Figure 2.6 b) depicts the critical points determined by our algorithm when applied to the example graph shown in Figure 2.5. In this figure, the distance of every free cell to the closest unknown cell is visualized as a gray-scale value: the darker the value, the longer the path to the closest unknown cell (avoiding obstacles in the map which are shown in black). It can be seen, that all doorways have been identified successfully and that no critical points are generated in areas that have been explored already.

Our evaluation of this segmentation technique in real world experiments, which is presented in Section 2.6, showed that a segmentation can be computed sufficiently fast and

a)

b)

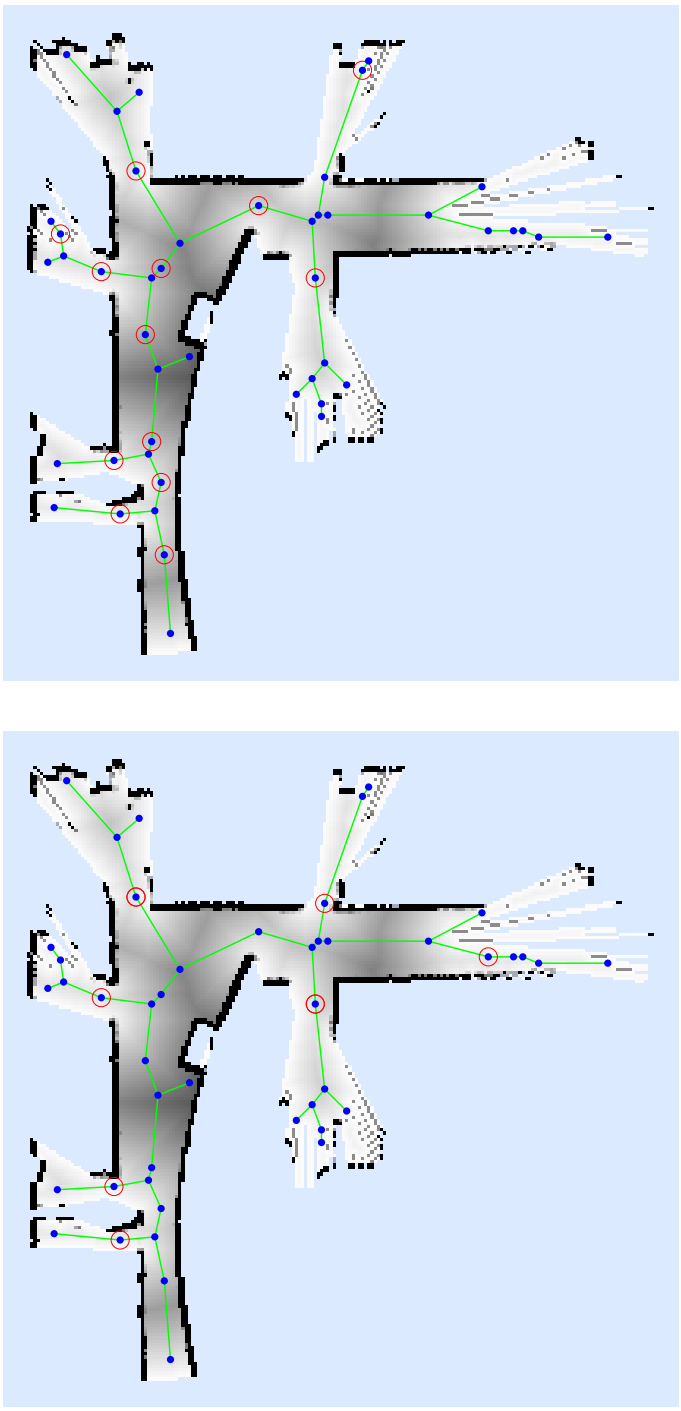


Figure 2.6: Critical points in the Voronoi graph of a small section of an environment. a) The nodes highlighted in red are the critical points determined using the local minima criterion only, as proposed by Thrun *et al.* Nodes are depicted as blue dots and edges are depicted as green lines. Light blue areas have not been explored yet. b) Critical points chosen by our approach that considers the distance to unknown cells. The distance of every free cell to the closest unknown cell is visualized as a gray-scale value: the darker the value, the longer the path to the closest unknown cell.

results in a partition that can be used to efficiently coordinate a team of robots. In typical office environments, we could reliably separate rooms and segments of a corridor. Other, more complex environments, may however require more sophisticated segmentation algorithms which rely on hand-labeled training data (Brunskill et al., 2007; Friedman et al., 2007) or more complex reasoning (Beeson et al., 2005; Zivkovic et al., 2006).

Urban outdoor environments, such as parks or campus sites, offer a structure that can be partitioned into several segments as well. In Chapter 6, we will describe a technique to identify areas that contain vegetation to differentiate them from areas that contain flat, drivable surfaces. In this way, a segmentation is defined that can be used, for instance, to explore such an environment.

2.5. Multi-Robot Coordination using a Segmentation of the Environment

When exploring an environment with a team of robots, one typically seeks to minimize the time to cover the complete environment. Clusters of robots that have a substantial overlap in the field of view of their sensors do not exploit their full potential. Since their measurements overlap, the robots carry out redundant work. For this reason, it is important to assign robots to exploration targets such that the robots do not get too close to each other during exploration. The described effect occurs mostly in situations where rooms or other confined spaces are explored by more than one robot. In general, it is therefore more efficient to explore separate regions of the environment with different robots.

Buildings are usually divided into rooms which can be reached via corridors. In many cases, it is a disadvantage to assign more than one robot to the same room. The room might, for example, be too small for a second robot to speed up it's exploration although there initially is more than one exploration target in the room. When the room is fully explored, robots might even block each other while trying to leave the room which will result in an increase in exploration time. In our approach, we assign individual robots to separate segments of unexplored space. In an indoor environment, such segments are rooms, corridors, or parts of larger corridors or rooms. Our approach takes into account the structure of the environment and prevents the forming of inefficient clusters of robots. As we will show in the following, the algorithm is equivalent to a purely cost-based coordination method in those cases in which the environment cannot be partitioned.

Our assignment algorithm is summarized in Algorithm 2. An assignment is determined whenever one of the robots requests a new exploration target. First, a partition of the partial map of the environment is created, for instance using the graph-based method described in Section 2.4. The algorithm then determines the set of exploration targets T_i within each segment s_i using the frontiers approach described in Section 2.2.

Algorithm 2 Target Assignment Using Map Segmentation.

```
1: Determine a segmentation S = {s_1, ..., s_m} of the map.
2: for all segments s_i ∈ S do
3:    Determine the set of exploration targets T_i within s_i
4: end for
5: for all segments s_i ∈ S do
6:    for all robots r_j, j ∈ {1, ..., n} do
7:       Compute the cost C_i^j of r_j reaching the nearest target t_min ∈ T_i in segment s_i.
8:    end for
9: end for
10: Assign robots to segments using the Hungarian Method.
11: for all segments s_i ∈ S do
12:    if any robot assigned to s then
13:       Locally assign robots to exploration targets T_i using the Hungarian Method.
14:    end if
15: end for
```

We determine the cost C_i^j of robot r_j to explore segment s_i by estimating the expected path cost to the nearest exploration target within $t_{min} \in T_i$. This quantity can be estimated efficiently using a path planning algorithm, for example, Dijkstra's algorithm. As an additional heuristic, we determine the segment that each robot is currently exploring. The estimated cost is then discounted by a constant factor if robot r_j is already located in segment s_i. This has the effect that the robots stay in their assigned segment until it is completely explored.

After computing the costs of exploring a segment, an assignment is determined by applying the Hungarian method (see Section 2.3) based on the cost matrix given by $C = [C_i^j]_{i \in \{1,\cdots,m\},\ j \in \{1,\cdots,n\}}$. The Hungarian method does not assign more than one robot to the same segment unless there are more robots available than there are unexplored segments. To appropriately handle those cases in which multiple robots are assigned to a single segment, we perform a local assignment of robots to individual exploration targets within a segment. Because of this local assignment, our algorithm is equivalent to a purely cost-based assignment if the environment cannot be partitioned, i.e., there is only one segment.

By assigning robots to separate segments, a wide distribution of the robots over the environment can be achieved. As we will demonstrate in the experiments, this leads to a significant reduction in exploration time. In a typical office environment, for example, each of the corridors is explored completely by one of the robots. In this way, the rough structure of the building will quickly be revealed. Meanwhile other robots will be assigned to the rooms reachable from the corridors, one at a time. This behavior does not only appear to be a natural way of exploring an unknown environment, our experiments

also revealed that it significantly increases the efficiency of the robot team compared to approaches which ignore the structure of the building.

Note that our coordination algorithm is not limited to homogeneous teams of robots. Consider the situation in which one particular robot cannot enter a certain part of the environment while another robot can. Such a situation may, for example, arise in an outdoor exploration scenario where several types of terrain are encountered and some robots are better suited to explore a given type of terrain than others (see Chapter 6). The assignment algorithm described above can be applied in this case by using modified exploration costs $\bar{C}_i^j$ defined as:

$$\bar{C}_i^j = \begin{cases} C_i^j & \text{if robot } r_j \text{ can enter segment } s_i \\ \infty & \text{otherwise.} \end{cases} \tag{2.4}$$

The algorithm assumes a central planning agent that coordinates the team. For this reason, reliable communication among the team of robots is required. If the communication range of the robots is limited, then clusters of robots can be coordinated by choosing one individual planning agent per cluster (Ko et al., 2003; Stachniss, 2006).

2.6. Evaluation

Our approach has been implemented and evaluated using simulated as well as real teams of robots. The simulated experiments have been designed to verify that our exploration approach leads to a significantly shorter exploration time compared to a standard cost-based approach that ignores the structure of the environment. We used the Carnegie Mellon Robot Navigation Toolkit (Montemerlo et al., 2002) to simulate teams of robots. This framework simulates robot movements and sensor readings on the basis of ground truth maps.

In addition to simulated teams we also evaluated our approach using a real team of robots. The team consisted of two ActivMedia Pioneer II robots equipped with a laser range finder with a 180° field of view. The real world experiments have been conducted to demonstrate the applicability of the overall system under realistic conditions.

During our experiments, the robots updated a joint occupancy grid map. Initially empty, this map was generated from the sensor readings of all robots under the assumption that the positions of all vehicles are known. This map was used for coordination, path planning, and path execution. Coordination took place on a central planning component that could communicate with all robots to assign exploration targets to them.

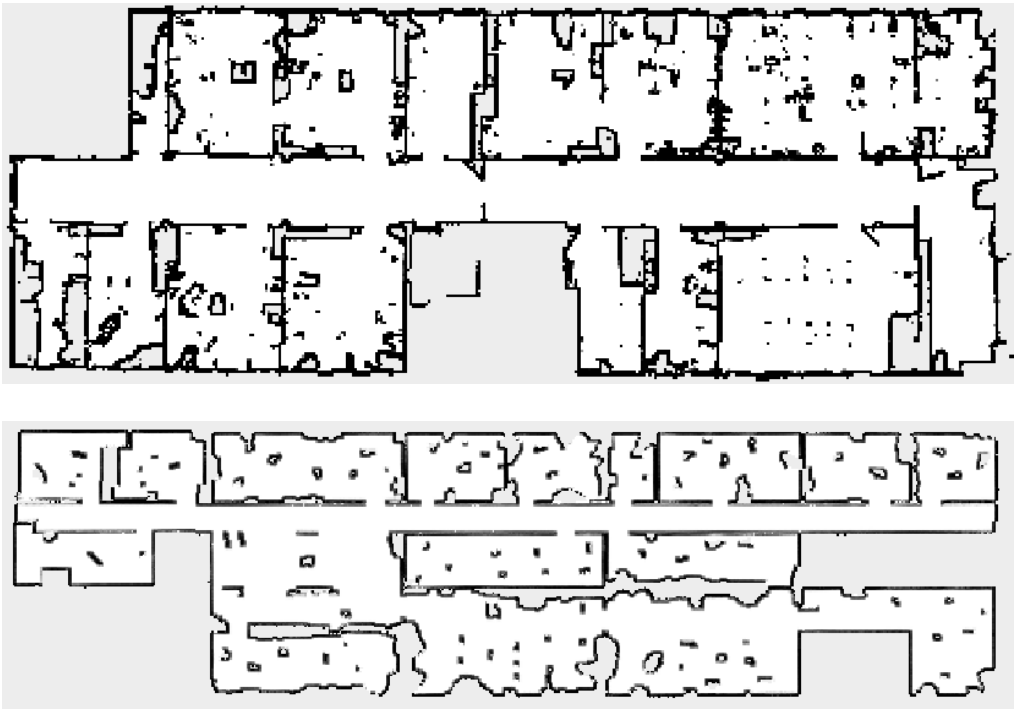


Figure 2.7: Maps of the the environment used for our simulated experiments: Building 079 of the computer science campus of the Freiburg University (top) and the Cartesium building at the University of Bremen (bottom).

2.6.1. Simulated Experiments

To evaluate our robot coordination algorithm, we simulated teams of robots in various environments. We compared our segmentation-based approach to a cost-based approach in which each robot is assigned to the closest frontier that has not been assigned to another robot yet. This baseline approach is similar to the approach introduced by Ko *et al.* (2003). Since this strategy does not consider the structure of the environment, it will in general assign more than one robot to a room or corridor if it contains more than one exploration frontier. To eliminate influences of the segmentation algorithm, we manually specified a segmentation of the environment into rooms and corridors in our simulation experiments. As mentioned above, such a segmentation could also be generated reliably from the partial map.

Figure 2.7 depicts the maps of the two buildings that were simulated in the evaluation, the Freiburg map (top) and the Bremen map (bottom). Both of these buildings are office environments, one is situated at the University of Freiburg and the other building is part of the University of Bremen. To create a more challenging scenario, we added clutter to the map representing the office environment located at the University of Bremen. At a width of 54 m, the Bremen map is considerably bigger than the Freiburg map (37 m) and we simulated larger teams of robots there. We varied the size of the simulated team

from two to six robots (Freiburg map) respectively from two to eight robots (Bremen map). For each team size, we conducted a series of 20 simulated exploration runs starting from different positions. Each of these experiments was performed once using our segmentation-based approach and was repeated using the baseline coordination method.

We compared both approaches in terms of the overall runtime to completely explore the environment. The results of our evaluation can be seen in Figure 2.8. The figure shows the relative runtime improvement of our approach compared to the baseline approach, where the improvement is defined as $(t_{segmentation} - t_{baseline}/t_{baseline}) * 100$, with $t_{baseline}$ denoting the absolute exploration time of the baseline approach and $t_{segmentation}$ denoting the absolute exploration time of the segmentation-based approach. A paired t-test showed that our target assignment method significantly outperformed the baseline approach.

We observed a bigger runtime gain in the evaluation of the Bremen map. This map features several large rooms while the Freiburg map is composed of two large corridor segments and a number of smaller rooms. This observation points to scenarios in which our approach will lead to especially good results: Whenever the environment can be divided into reasonably large and separated segments, our technique substantially reduces the overall exploration time.

When there are more segments than robots, our strategy assigns one robot to one segment. As soon as there are more robots than segments, multiple robots may be assigned to the same segment as mentioned in Section 2.5. For this reason, the runtime gain of our strategy will decrease for large teams of robots in small environments. This can be seen in Figure 2.8. Note, however, that the overall time to complete the mission is still reduced when more robots are added to the task – the plot only shows the improvement of our approach compared to the baseline approach.

2.6.2. Real Robot Experiments

To demonstrate the applicability of the overall system under realistic conditions, we used our segmentation-based approach to explore a typical office building with a team of real robots. For this experiment, we used two identical Pioneer II robots equipped with a SICK LMS laser range finder and a standard laptop-computer. During the experiment, both robots were connected via a wireless network. The robot localization was achieved using an incremental scan-matching approach as described by Hähnel (2005). The relative starting poses of the robot were determined manually in the beginning. Figure 2.9 depicts the two robots during the exploration mission.

During exploration, the team was coordinated using the segmentation-based approach introduced in this chapter and the segmentation of the map was determined utilizing the graph-based segmentation described in Section 2.4. The experiments were conducted in the lower floor of building 079 of the Freiburg computer science campus (a map of this building was also used in the simulated experiments, see Figure 2.7). The building has a

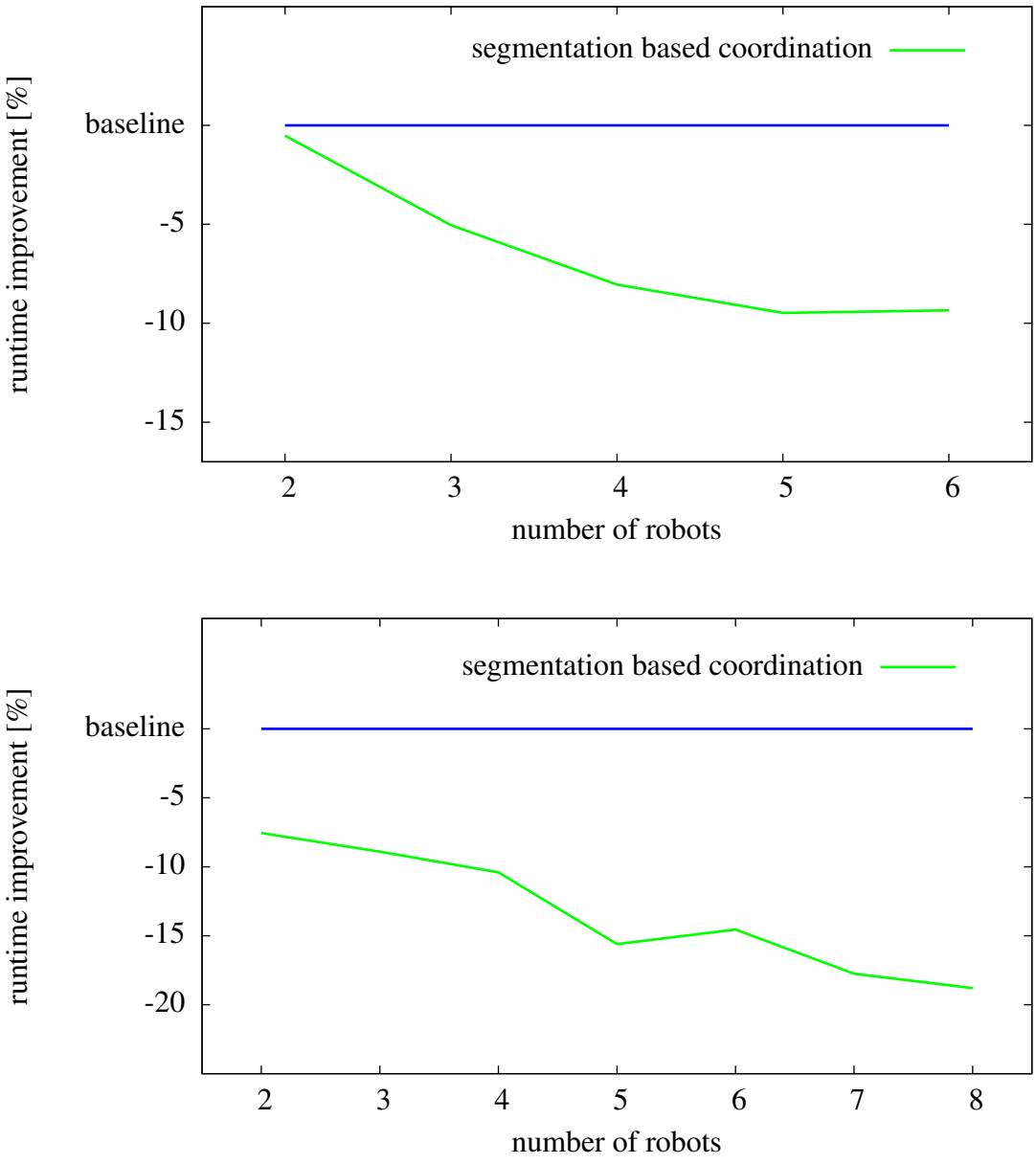


Figure 2.8: Average exploration runtime improvement of our segmentation-based approach over a frontier-based baseline approach. The results for the Freiburg map are shown at the top and the results for the Bremen map are shown at the bottom.

Figure 2.9: Two Pioneer II robots exploring the AIS laboratory of the University of Freiburg using our segmentation-based coordination approach.

size of approximately 37 m x 14 m and consists of numerous office rooms and two long corridors divided by a door.

The team of robots was able to successfully explore the environment using our coordination approach. The result of one of the experiments can be seen in Figure 2.10. The figure shows the map generated from the sensor measurements of both robots after the exploration had finished. It also shows the trajectories of both robots during the exploration. The total exploration time was less than nine minutes, each of the robots traveled approximately 120 m.

It can be seen that each of the rooms was explored by exactly one of the robots. It can also be observed that both corridors have been explored completely by one of the robots while the other one was exploring rooms reachable from the corridor. Another noteworthy effect of the segmentation-based coordination is that the robots did not interfere or block each other during the execution of their tasks.

2.7. Related Work

The problem of autonomous mapping environments with mobile robots is well studied and can be considered "a primary application domain in robotics systems development" (Thrun et al., 2005). An early approach described by Kuipers *et al.* (1991) autonomously builds a topological representation of the environment while the approach presented by Dudek *et al.* (1991) uses a graph-structure.

Two of the earliest approaches that learn a metric model of indoor environments have been presented by Thrun *et al.* (1993) and Edlinger and Puttkamer (1994). An explo-

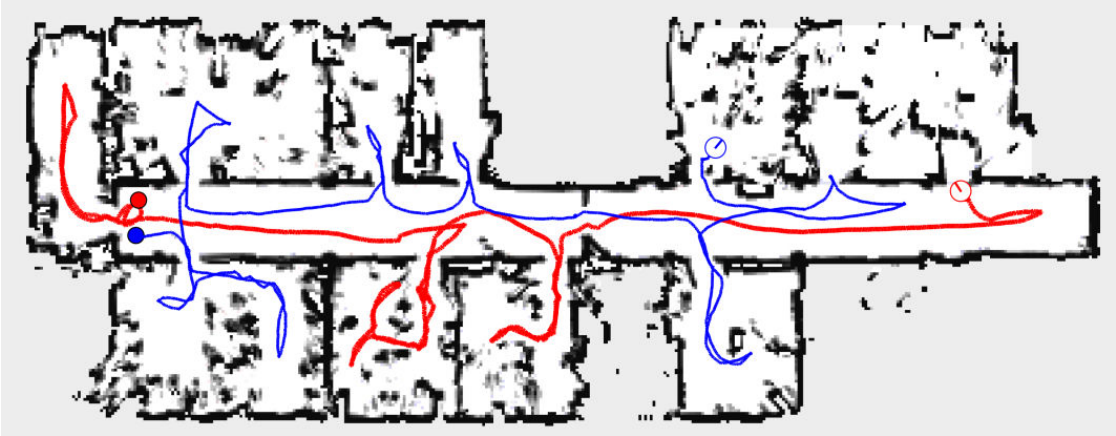


Figure 2.10: Resulting map of the real world experiment superimposed with the trajectories of the two robots.

ration system that uses occupancy grid maps has been presented by Yamauchi (1997). He introduced the concept of frontiers between known and unknown areas in a grid map, which are widely used to select potential target locations during exploration.

The extension from single exploring robots to teams of robots has received considerable attention in the past. The efficiency of the team is guided by the coordination strategy to a large extent and a host of different approaches have been presented.

An implicit approach that assigns each robot to the closest exploration target was proposed by Yamauchi (1998).

Ko *et al.* (2003) presented an approach that uses the Hungarian method (Kuhn, 1955) to compute the assignments of frontier cells to robots. In contrast to our approach, Ko *et al.* mainly focus on finding a common frame of reference in case the start locations of the robots are not known and do not utilize a segmentation of the environment.

As a way to trade off costs and gains of exploring a target, Burgard *et al.* (2005) compute the utility of frontier targets. They simulate sensor measurements, estimate path costs, and take visibility constraints into account. Targets are then assigned to the robots iteratively.

Zlot and colleagues (2002) proposed an architecture for teams of mobile robots in which the exploration is guided by a market economy. They consider sequences of potential target locations for each robot and trade tasks between the robots using single-item first-price sealed-bid auctions. Such auction-based techniques have also been applied by Gerkey and Matarić (2002) and Berhault *et al.* (2003) to efficiently solve the task allocation problem with a group of robots.

In a formal analysis of task allocation methods, Gerkey *et al.* (2004) compared centralized coordination methods and distributed methods such as auction-based approaches. They concluded that "for small- to medium-scale systems, say $n < 200$, a broadcast-based centralized assignment solution is likely the better choice."

An early approach towards cooperation in heterogeneous robot systems was presented by Singh and Fujimura (1993a). In their system, whenever a robot is too big to pass through a narrow passage during exploration, other robots are informed about the task. Howard *et al.* (2002) presented an incremental deployment approach that explicitly deals with obstructions, i.e., situations in which the path of one robot is blocked by another robot. Koenig *et al.* (2001) analyze different terrain coverage methods for small robots with limited sensing and computational capabilities.

The problem of integrating semantic background information into the coordination procedure was previously considered by Stachniss *et al.* (2006). This technique is related to the method proposed in this chapter, even if the methodology is substantially different. Their approach reduces the overall exploration time for teams of more than five robots. In contrast to that, our approach is also able to reduce the exploration time significantly for small teams of robot.

Learning topological maps is a research field on its own and different methods have been proposed (Brunskill et al., 2007; Friedman et al., 2007; Kuipers and Byun, 1991; Thrun, 1998; Zivkovic et al., 2006). These approaches segment the environment into regions and they have originally been designed to facilitate topological localization and loop closing or they have been used to reduce planning costs. In contrast to this, we apply segmentation techniques to coordinate a team of exploring robots. Since our coordination approach is not specific to the graph-based segmentation method we presented in this chapter, it can make use of any of these segmentation approaches to improve the coordination of an exploring team of robots.

Our findings have recently been confirmed by other researchers. In a comparative evaluation of exploration strategies, Holz *et al.* (2011) confirmed that using a segmentation of the environment, as it is done in our approach, leads to substantial improvements:

> "Using map segmentation (SEG) reduces the traveled distance especially in the hospital environment, where SEG provides a good quality segmentation. Enabling SEG prevents to leave out corners and occlusions and exploring them in the last steps of the exploration. Without SEG, multiple visits to the same room can be necessary, e.g., when the current robot's room is not completely explored and the best frontier location happens to be outside that room."

2.8. Conclusion

In this chapter, we presented a novel technique for coordinating a team of exploring robots. Our approach makes use of the structure of the environment. We compute a segmentation that partitions the environment into structurally important regions. In buildings, such regions correspond to rooms and corridors. Previous approaches often gener-

ate exploration targets on the frontier between mapped and unmapped areas and multiple exploration targets are usually generated in the same room or corridor. When several robots explore the same physical region, however, there often is a considerable overlap of the sensor measurements that leads to inefficiency. By considering the regions we identified in our approach, we are able to cluster targets according to the structure of the environment. We assign robots to regions instead of assigning them to target locations directly, thus achieving a balanced distribution of the team. This leads to a significantly shorter overall exploration time compared to previous approaches which do not consider the structure of the environment. The distribution achieved by our approach reduces not only the amount of redundant work but also the risk of interference between robots. In addition to the coordination strategy, we introduced an efficient graph-based segmentation technique for partially explored environments that identifies regions in buildings that are separated by narrow passages such as doorways. Our multi-robot coordination approach has been implemented and evaluated in simulation as well as with a team of real robots. The experiments show a significant improvement of our approach compared to a standard frontier-based approach.

Our approach is not limited to the partition of indoor environments into rooms and corridors. Structurally important regions could, for example, also be defined on the basis of traversability information. In Chapter 6, we will present a technique to detect grass in outdoor environments. Using this information, our approach can be used to explore paths and areas covered with grass in a similar way as we applied it to explore corridors and rooms in a building.

Chapter 3
Coordinating Heterogeneous Teams of Robots

3.1. Introduction

In many applications, a coordinated team of robots offers advantages over a single robot. Multi-robot systems have the potential of being more fault tolerant and of reducing the overall time to complete a given task (Dudek et al., 1996; Cao et al., 1997). A well-studied problem is the exploration of an unknown environment with a cooperative team of robots. Previous approaches often addressed the problem of exploring an environment with groups of robots that have identical capabilities. To coordinate such *homogeneous* teams, popular approaches determine a set of exploration targets and assign robots to them numerically (Zlot et al., 2002; Ko et al., 2003; Berhault et al., 2003; Burgard et al., 2005; Stachniss, 2009). These approaches consider the cost and the expected information gain of each exploration target. The coordination technique we presented in Chapter 2 is also an example of such cost-based numeric approaches.

In this chapter, we address the problem of coordinating exploring teams of robots with differing capabilities. The robots in such *heterogeneous teams* may differ in their physical properties such as their sensor setup, their size and payload, their maximum traveling speed, or the type of terrain they are able to traverse. In our approach, we especially consider robots that differ in the actions they are able to perform. For instance, robots might be equipped with manipulators, they could be able to deploy localization beacons, or they could even deploy other robots. Numeric, cost-based coordination approaches

are able to consider differing properties of robots if navigation costs are affected. For example, the cost of reaching a target depends on the travel speed of a robot and it is also possible to encode that a robot cannot reach a target due to constraints on the size of the robot or the type of terrain it can traverse. Unfortunately, it is not straightforward to take into account actions other than navigating to exploration targets since there is no efficient mapping of such actions to cost or utility measures. While we can usually specify the time it takes to perform an action such as deploying a robot, it is not meaningful to compare this cost to the cost of exploring a given frontier target. The reason for this incompatibility lies in the fact that performing an action that does not explore parts of the environment has no apparent immediate reward to an exploration system but it effects its performance in the future.

From a conceptual point of view, the ability to perform actions that go beyond navigating to goal positions introduces corresponding *symbolic actions*. This term is commonly used in classical planning approaches that encode the state of a system using logical symbols. Classical planning approaches execute symbolic actions to change the state of the system towards a pre-defined goal state. In the context of multi-robot exploration, we define a symbolic action as any action that does not explore parts of the environment directly. Examples of such actions include opening doors, operating elevators, moving obstacles, and deploying other robots. There exist relatively few approaches that coordinate teams of robots and take into account symbolic actions. One specific set of actions that has previously been considered is the deployment and retrieval of robots by other robots. To execute symbolic actions, previous approaches rely on manually designed strategies (Singh and Fujimura, 1993b; Murphy et al., 1999; Dellaert et al., 2002). One strategy, for example, is to deploy a smaller robot whenever a large robot cannot reach a given goal. These strategies, however, are specific to a certain type of robot and environment and it is unclear whether they are able to efficiently coordinate large teams of robots.

The approach presented in this chapter considers symbolic actions explicitly by applying symbolic planning techniques. In the context of multi-robot exploration our goal is to explore all exploration targets as fast as possible. We assume that to reach some targets it is necessary to execute additional symbolic actions such as removing an obstacle or operating an elevator. The key idea of our approach is to integrate a symbolic planning system and a robotic path planner. Since it is unclear how we can translate symbolic actions into a cost measure as it is used in numeric coordination approaches, we instead treat navigating to an exploration target as an action in a symbolic planning system. We generate a symbolic formulation of the coordination problem that includes navigation goals as well as symbolic actions that have to be executed. This description serves as the input to a symbolic planning system that solves the coordination problem. However, classical planning approaches do not consider the execution cost of the solutions they generate. Adding costs measures to actions turns the coordination problem into a *temporal*

planning problem and there exist efficient algorithms to solve such problems (Gerevini et al., 2008; Eyerich et al., 2009). We estimate execution costs for each navigation action using a robotic path planner.

In contrast to cost-based coordination approaches, our system is able to explicitly plan for the execution of symbolic actions. Still, the use of action costs that are determined by the robotic path planner allows us to generate time-efficient solutions. In this way, our approach combines the strength of cost-based coordination approaches with the flexibility of symbolic planning systems. To evaluate our coordination approach, we apply our framework to two popular multi-robot applications: exploration of unknown environments with heterogeneous teams of robots and disaster recovery with heterogeneous teams.

An interesting coordination problem arises when a team of robots includes members that are able to deploy and retrieve other, smaller robots. For a task such as the autonomous exploration of lunar craters, one can imagine robots that approach the crater and then deploy a specialized robot which descents into the crater (Cordes et al., 2011; ESA, 2008). Such systems are frequently referred to as marsupial robots (Murphy et al., 1999). The coordination of marsupial teams generally requires to carefully plan deployment and retrieval actions, both are symbolic actions according to our definition. In addition, one has to take into account the different properties of the robots such as their sensor setup, their size and payload, their maximum traveling speed, or the type of terrain they are able to traverse. The coordination framework proposed in this chapter is applied to the exploration of an unknown environment using marsupial teams of robots.

A further class of applications for teams of heterogeneous robots is centered around the scenario that important parts of an environment have collapsed after a natural disaster. In such a setting, teams of robots can, for instance, be used to perform search and rescue missions. We consider the task of carrying out pre-defined actions at certain critical locations in the collapsed structure, for example, to re-establish safe operation of a failed power station or chemical plant. Such a disaster recovery task requires robots with diverse capabilities. First of all, robots are required to clear collapsed paths throughout the structure. At the same time, there is a need for robots with good sensors or precise manipulation skills to open doors, operate valves, closely inspect parts of the building, or repair damage. A heterogeneous team of specialized robots can be used to provide these capabilities in a flexible way. We apply our proposed framework to coordinate a team of exploring and clearing robots in a simulated disaster scenario. In addition to exploratory navigation actions, we explicitly plan for symbolic clearing actions.

Both application scenarios serve as motivating examples for this work. We developed and implemented a coordination approach for both applications and compared our approach to ad-hoc extensions of cost-based numeric coordination approaches. As we will show in the evaluation, our approach produces significantly better plans leading to a de-

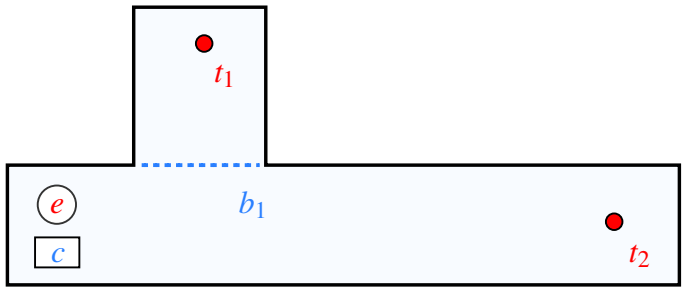


Figure 3.1: An exploring robot e has to explore both targets t_1 and t_2. The path to t_1 is blocked at b_1 (dashed line). A clearing robot c can clear the blockade.

crease in both the overall runtime and the sum of traveled distances. Additionally, we applied our approach to coordinate a real-world heterogeneous robotic system.

The remainder of this chapter is organized as follows. We begin by giving a short introduction to the temporal planning approach we apply in our system. In Section 3.3, we describe our coordination framework and its components. In the following Section 3.4, we describe a specific application of our approach to the problem of exploration with marsupial robots. In Section 3.5, we present a further application of our framework in the context of disaster recovery. Our experimental evaluation is presented in Section 3.6. Finally, we discuss related work in Section 3.7.

3.2. Temporal Planning

Our approach employs a symbolic planning system to compute actions for a team of robots. Algorithms for classical domain independent planning generate plans that consist of sequences of actions. There is a substantial body of literature on this topic and a description of planning algorithms can, for example, be found in (Russell and Norvig, 2010). In contrast to these classical planners, temporal planning systems explicitly allow for concurrent actions. Consider the initial state of the simple planning problem shown in Figure 3.1. The environment contains two targets t_1 and t_2 and two robots e and c. The task is to explore both targets using robot e. Robot c can clear the blockade b_1 using the action `(clear c b1)` and robot e can explore targets using, for example, the action `(explore e t2)`. Target t_1 is blocked by b_1 and cannot be explored initially.

A classical planner is able to produce a sequence of actions that solves the problem of exploring all targets. For example the following plan is a valid solution:

```
(clear c b1)
(explore e t2)
(explore e t1)
```

This, however, is not the best solution to the task. The first two actions are not dependent

on each other, so they can be executed in parallel. Temporal planning allows to express such concurrent actions and would result in the following exemplary plan

```
0.0: (clear c b1)   [1.2]
0.0: (explore e t2) [2.0]
2.0: (explore e t1) [1.8],
```

where the actions are preceded by the start time and the durations of actions are given in square brackets. For example, `(explore e t2)` starts at time 0.0 and takes 2.0 time units to complete.

In our approach, we define temporal planning problems using PDDL 2.1 (Fox and Long, 2003), which is a language for defining planning problems and allows for durative actions. Durative actions are an extension of classical planning actions. In addition to conditions and effects they allow the problem description to specify an execution time. Note that solutions to temporal planning problems do not define a strict sequence of actions but merely a partially ordered set and especially that actions might be executed in parallel.

In realistic planning problems, there will be restrictions on which actions can be executed in parallel and which actions have to be completed before another action can be started. These restrictions are modeled as conditions. More precisely, a condition is defined as a tuple $(C_{\vdash}, C_{\leftrightarrow}, C_{\dashv})$, where $C_{\vdash}$ is a start condition that must hold at the start time of the action, $C_{\dashv}$ is an end condition that must hold at the end of the action and $C_{\leftrightarrow}$ is an overall condition that must hold during the execution of the action.

The effects of durative actions are specified in a similar way. An effect is defined as a tuple $(E_{\vdash}, E_{\dashv})$, where $E_{\vdash}$ is a start effect that is applied at the start of the action and $E_{\dashv}$ is an end effect that is applied at the end of the action. For more details on the definition of temporal planning tasks we refer to the work of Eyerich *et al.* (2009).

When coordinating a cooperative team of robots, specific tasks are assigned to individual robots. In most domains, robots work in parallel. Still, their actions might depend on each other, for example, when a robot is clearing a path for another robot. Coordination problems that include such cases can only be solved efficiently when the concurrency of the domain, but also interdependencies of robot tasks, are accounted for. Temporal planning allows for the specification of such conditions and is therefore well suited for multi-robot coordination.

3.2.1. Planning Domain Definition Language

A wide range of problem types can be modeled as a general planning problem, ranging from transportation problems and single-player games to general combinatorial problems. In recent years, the Planning Problem Definition Language (PDDL) (Fox and Long, 2003) has been established as the prevalent planning language.

To generate a PDDL task description, one needs to define (i) the objects involved in the planning process, (ii) the predicates that define the state of the planner, (iii) actions that change the state, and (iv) a start state and a goal condition. We will give several examples of PDDL statements in the following sections. The PDDL description then forms the input to symbolic planners.

In our approach, PDDL is used to encode the coordination problem that has to be solved in a multi-robot system. Based on this description, the temporal planner computes concurrent actions for the team of robots. We use PDDL/M (Dornhege et al., 2009), an extension to PDDL that allows for the definition of external module calls. Using this method, we combine symbolic planning and a robotic path planner in our approach. Details are given below, see Section 3.2.3.

3.2.2. The TFD/M Planning System

TFD/M is a domain-independent progression search planner developed by Dornhege *et al.* (2009). It extends the temporal planner framework Temporal Fast Downward (TFD) by Eyerich *et al.* (2009). TFD in turn is based on a planning classical system called Fast Downward that was developed by Helmert (2006). TFD extends the original system to support durative actions and numeric expressions while TFD/M adds support for external modules.

TFD/M solves a planning problem in three phases: First, the PDDL planning task is translated from its original encoding into a more concise representation using finite-domain variables. This is used by the planner to guide the search by employing hierarchical dependencies between state variables and leads to an increased search performance. In the second step, efficient internal data structures are generated that are used by the search component and the search guidance function. The most important ones are *domain transition graphs* for each variable that encode how state variables can change their values and the *causal graph* that represents the hierarchical dependencies between different state variables. Finally, a best-first progression search is performed, guided by a numeric temporal variant of the context-enhanced additive method (Helmert and Geffner, 2008).

In contrast to several other temporal planning systems, TFD/M does not decompose the search into an action selection phase and a scheduling phase but searches directly in the space of time-stamped states. This usually leads to plans of significantly higher quality (Eyerich et al., 2009). Note, however, that the first plan that is generated by this method is not necessarily optimal. This is because the search guidance function is inadmissible, in other words, it does not guarantee an underestimation of the true execution cost of a plan.

TFD/M is implemented as an anytime algorithm that does not terminate after the first solution is generated. It produces a potentially non-optimal solution quickly and then

prunes the search space to those time-stamped states which can potentially be extended to solutions with a lower overall duration. If all states in the resulting state space are expanded, the produced solution is guaranteed to be optimal.

3.2.3. Semantic Attachments in TFD/M

TFD/M features semantic attachments that are a means of evaluating components of the planning task externally. This is implemented as a module interface for predicates, numerical effects, and durations. In our coordination algorithm, semantic attachments are used to specify durations of actions in the planning task description. Consider, for example, the following module specification:

```
(costClear ?r - robot ?b - blockade cost
 costClear@libcost_module.so)
```

This defines a semantic attachment for cost computation (indicated by the `cost` keyword) and is used to specify the duration of actions. It has two parameters, a robot *r* and a blockade *b*. To define a semantic attachment, a function call is provided that performs the actual computation externally (here, `costClear@libcost_module.so`).

We can now use the semantic attachment defined above to specify the duration of actions. The `clear` action in the initial problem (see Figure 3.1), for example, can be defined in the following way:

```
(:durative-action clear
:parameters (?c - clearer ?t - blockade)
:duration (= ?duration [costClear ?c ?t])
:condition (and (at start (at ?c ?t))
                (at start (not (cleared ?t))) )
:effect (and (at end (cleared ?t)) ) )
```

Note that the use of semantic attachments is indicated by squared brackets in the definition. For details on the implementation of semantic attachments in forward chaining state space planners such as TFD/M, we refer to the work of Dornhege *et al.* (2009).

When the planner expands actions in the planning phase, it detects semantic attachments and executes the associated dynamic library calls. These external calls compute the appropriate action costs. In our approach, for example, the cost of navigation actions are computed by a robot path planner.

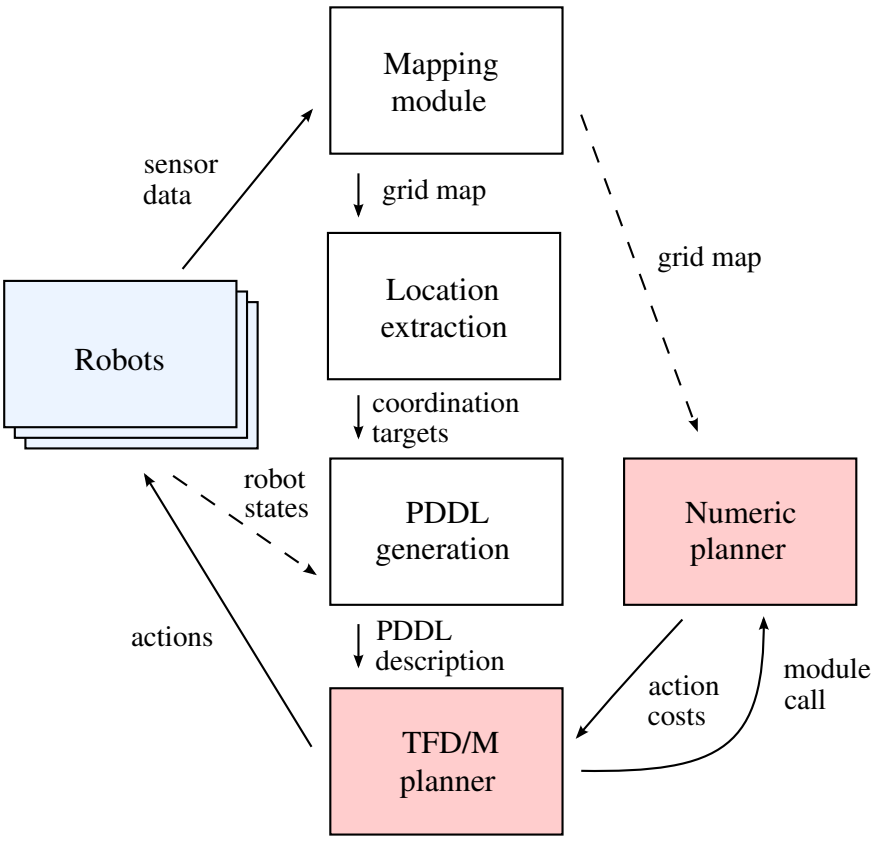


Figure 3.2: System overview of our coordination system. Solid arrows illustrate the control flow and dashed arrows represent data passed between modules.

3.3. Coordination of Heterogeneous Teams of Robots Using Temporal Planning

In the following, we will describe our coordination framework for heterogeneous teams of robots. We assume that to solve a given task, the robots have to move in the environment and additionally need to perform symbolic actions. In this context, symbolic actions are defined as actions that change the state of the overall system but whose primary purpose is not to change the position of the robots. Numeric coordination approaches usually consist of a method to evaluate the utility of navigation goals and a target assignment technique. Since it is unclear how we can translate symbolic actions into a utility measure as it is used in numeric coordination approaches, we instead treat navigating to an exploration target as an action in a symbolic planning system. In our approach, the target evaluation and assignment modules of numeric approaches are replaced by a temporal symbolic planner, a method to generate a problem description for this planner and a numeric path planner that is used to estimate the path costs of navigation actions.

The architecture of our system is illustrated in Figure 3.2. In our approach, we assume global and unlimited communication between the robots and thus employ a centralized coordination approach. Furthermore, all robots are assumed to know their individual position. The team of robots provides the sensor data and states of the platforms, such as their positions, current actions and navigation goals, to our centralized coordination system. We use the sensor measurements and poses of the robots to build a grid map of the environment. From this map, we extract locations that are relevant for coordination. In an exploration task, relevant locations would be exploration targets that can, for example, be determined using the approach described in Section 2.2. To execute symbolic actions we furthermore extract positions at which the actions need to be executed. If some of the robots were able to open doors, for example, we would determine the positions of doors in the map.

We solve the coordination problem of assigning actions to the robots using the temporal symbolic planning system TFD/M. From the current state of the robots and the locations we extracted from the map of the environment we generate a problem description in PDDL. This description serves as the input for the planner. Since the goal is to solve the given task as fast as possible, we need to consider the costs of traveling to navigation goals. To this end, we make use of the modular interface of TFD/M to call a numeric path planner for mobile robots. The numeric planner returns the estimated path cost of reaching a goal position from a given start position. Based on these estimates, the temporal planner is able to compute an efficient plan that solves the mission goal. From the solution of the symbolic planner, we extract actions and send them to the robots. The loop depicted in Figure 3.2 is executed constantly: Whenever new information about

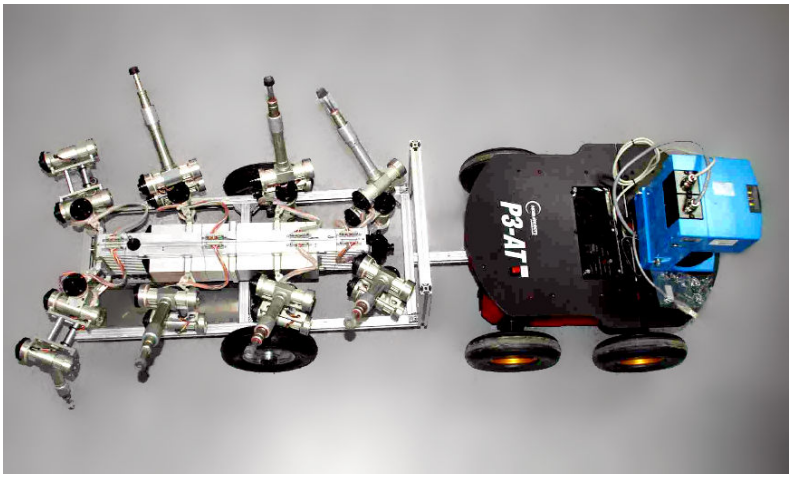


Figure 3.3: Example of a marsupial robot team. A versatile but slow legged platform is deployed by a faster wheeled robot. (Image courtesy of DFKI Robotics Innovation Center)

the environment arrives, for example, new sensor data is perceived or one of the robots reaches a target, we replan.

Some of the modules in our coordination framework are specific to the application. Depending on which actions the robots are able to perform, we need to adapt the PDDL description of the coordination problem. The possible actions furthermore influence which locations we need to extract from the map. In the following, we will present two applications and describe how to adapt the framework to them.

3.4. Coordination of Marsupial Teams

In the first application, we use our planning framework to coordinate a team of marsupial robots. In such a team, one group of robots, the *carriers*, are able to carry, deploy and retrieve a group of other robots, the *rovers*. An example of a real carrier and rover robot is depicted in Figure 3.3. Here, a versatile but slow legged platform is deployed by a faster wheeled robot.

We assume that carriers and rovers have different navigation capabilities and that certain areas of the environment can only be explored by the rovers and others only by the carriers. We furthermore assume that the robots are able to determine which areas are traversable by which robot based on their sensor observations, for example based on techniques presented in Chapter 6.

In our experiments, a marsupial team consists of n carrier robots, where each carrier initially carries m rover robots. The goal is to completely explore the environment, that is, to cover the traversable area with the sensors of the robots. Figure 3.4 depicts a situation where a carrier has to choose between exploring the environment itself and deploying a rover in an area that it cannot reach. This illustrates the key challenges that

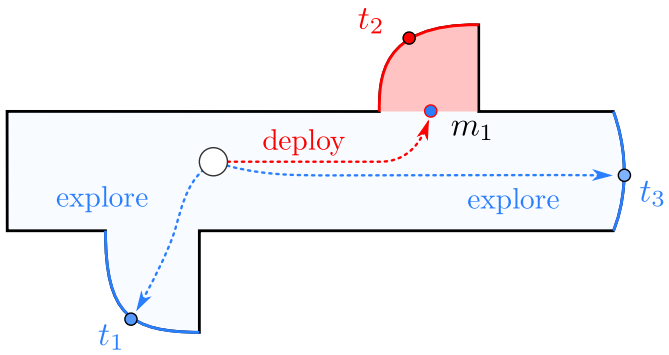


Figure 3.4: An exploring robot (white circle) has to choose between three possible actions: explore target t_1, explore target t_3, or deploy a smaller robot at m_1 to let it explore t_2 in the red area that it cannot explore itself.

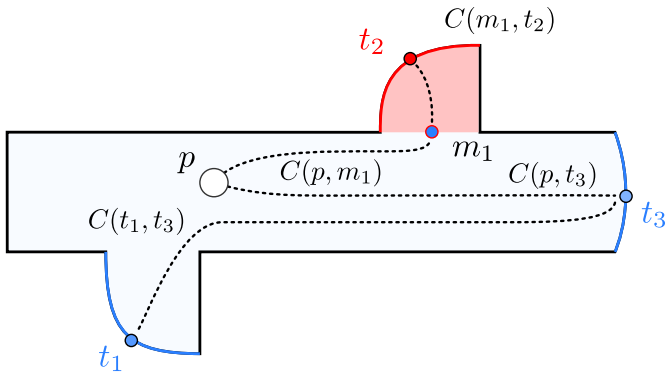


Figure 3.5: Example of the costs that have to be considered. Dotted lines illustrate the estimated path costs between the robot at position p and the different target positions t_i, the costs between meeting points m_i and robot or targets positions, and costs between target positions. For the sake of better visibility, we did not display all costs in this figure.

the coordination method faces: It needs to generate exploration targets, to assign robots to those targets, and to schedule deployment and retrieval actions.

3.4.1. Target Locations and Cost Estimation

To identify potential target locations, a set of exploration targets T is generated from the partially explored grid map. In addition to this, a set of meeting points M is determined. These meeting points are situated at the border between those parts of the environment that can only be traversed by the rovers and the parts that can only be traversed by the carriers. They are used for deployment and retrieval of the rovers (see Figure 3.5 for an illustration). To determine the targets and meeting points a frontier extraction algorithm is used as described in Section 2.2.

There are two basic types of actions a carrier can perform: exploring a target or visiting

a meeting point to deploy or retrieve a rover (see Figure 3.4). While deployment and retrieval are assumed to have constant costs, the cost of traveling between two locations in the environment is defined as the estimated travel time of a given robot. This cost depends on the path length as well as on the traversability constraints and travel speed of the corresponding robot. Let *type* be a robot type (here: carrier or rover), x a location in the environment and $t \in \{T \cup M\}$ a target. We define the cost for reaching t as:

$$C_{type}(x,t) = \begin{cases} \text{estimatePathCost}(x,t) & \text{if robots of type } type \text{ can reach } t \text{ from } x \\ \infty & \text{otherwise,} \end{cases}$$

where $estimatePathCost(x,t)$ returns the estimated path cost. Finally, the exploration task is completed as soon as the set of exploration targets T is empty.

3.4.2. Formulating the Exploration Problem as a Temporal Planning Problem

To apply the coordination approach described above, we need to encode the coordination problem that arises in a given situation as a PDDL description. First, we define which types of objects are involved. In the exploration scenario, possible objects are robots that can be either rovers or carriers and locations that can be meeting points or exploration targets. The corresponding PDDL statements are given in Figure 3.6 a. Second, we specify the predicates that are used to define the state. The major predicates we use to describe the exploration problem are

```
(at ?r - robot ?x - location),
```

which describes that the robot r is at position x and

```
(on ?e - rover ?c - carrier)
```

which is used to determine whether a rover e is docked at a carrier c. For each target $t \in T$, we also define if it has been explored

```
(explored ?t - target).
```

Additionally, we use a numeric state variable

```
(num_docked ?c - carrier)
```

that keeps track of the number of rovers that are docked at a carrier c.

Third, the actions that change the state are provided. We define four actions, namely `dock`, `undock`, `move`, and `explore`. The actions `dock` and `undock` are used to deploy or pick up a rover. They require that the carrier and the rover are at the same meeting point, which is ensured using the `at` predicate. For docking, the number of docked rovers has to be lower than the carrier's capacity and the action changes a rover's state from being at a meeting point to being on a carrier (encoded using the `on` predicate).

a)
```
(:types
  robot
  carrier rover - robot
  location
  target meeting - location )
(:predicates
  (at ?r - robot ?x - location)
  (on ?e - rover ?c - carrier)
  (explored ?t - target)
  (can_explore ?r - robot ?t - target))
```

b)
```
(:durative-action explore
  :parameters (?r - robot
     ?s - location ?g - target)
  :duration (= ?duration
     [pathCost ?r ?s ?g])
  :condition (and (at start (at ?r ?s))
     (at start (not (explored ?g)))
     (at start (can_explore ?r ?g)) ... )
  :effect
    (and
      (at start (not (at ?r ?s)))
      (at end (at ?r ?g))
      (at start (explored ?g))
      ... ))
```

c)
```
(:init
  (at robot0 p)
  (on robot1 robot0)
  (can_explore robot0 t1)
  (can_explore robot1 t2)
  (can_explore robot0 t3)
)
(:goal (and
  (explored t1)
  (explored t2)
  (explored t3)
))
```

Figure 3.6: Examples of PDDL definitions used in the exploration domain. a: Definition of the types and predicates. b: Definition of the `explore` action. c: Example that shows how to specify the current state of the world for the TFD/M planner (see illustration shown in Figure 3.4).

The other two actions `move` and `explore` model the possible motions of the robots. The `move` action moves a robot to a meeting point for deployment or retrieval while the `explore` action moves the robot to a target and explores it. To define the duration of the `move` and `explore` actions, we employ the module interface of the temporal planner. Instead of specifying a constant duration or a fixed formula, we call an external module that determines the duration of the action. In our setting, the external module is realized by an efficient path planner for mobile robots that plans the optimal trajectory of the robot to the given target location based on the current occupancy grid map constructed by the robots so far. Figure 3.6 b depicts the PDDL statements that describe the action `explore`. The term `[pathCost ?r ?s ?g]` represents the call to the external module. In our current implementation, we use Dijkstra's algorithm for cost estimation as it allows to efficiently compute the path cost from a given start position to all goal positions. Finally, the initial state of the current planning procedure and the goal state are specified. For the situation depicted in Figure 3.4, this is exemplified in Figure 3.6 c.

3.5. Coordinated Disaster Recovery

As a further application of the coordination approach described in Section 3.3, we investigate teams of robots that operate in a disaster recovery scenario. More specifically, we assume that a known building is partially collapsed so that paths within the building are blocked at unknown positions. The goal is to visit a given set of targets in the building using a team of robots. In a real world application, the robots could, for instance, provide high-resolution views of the situation, close a set of valves, or deploy a set of beacons. We assume that there are two types of robots: *exploring robots* that visit targets by performing a specific action at a given location and *clearing robots* that are able to clear blocked paths. Both types of robots are able to detect blockades using their sensors. All robots are provided with a map of the environment that includes the mission targets. This prior map does not, however, include blocked paths.

The key challenge in this scenario is to coordinate the team of robots so that the exploring robots do not waste time by waiting for blocked paths to be cleared. An illustration of a typical situation in this problem domain is given in Figure 3.1.

3.5.1. Target Locations and Cost Estimation

The initial mission map includes all target locations that have to be visited. These locations are extracted by the coordination module and stored in a set $T = \{t_1, \ldots, t_n\}$. Whenever a target is visited by one of the exploring robots, it is removed from the set T and the mission map.

While the robots navigate in the environment they update their maps using their sensors. Whenever a blockade b is sensed, this information is distributed to all other robots.

The central coordinator keeps track of blockades in a set $B = \{b_1, \ldots, b_m\}$ which is used to coordinate the clearing robots.

Furthermore, the coordination system maintains a representation of the environment that is composed of the prior building map and the set of blockades sensed by the robots. This map is used by the robot path planner to plan collision free paths for the team of robots and to estimate travel costs during target assignment. It estimates path costs from a location x to a target $t \in \{T \cup B\}$ according to

$$C(x,t) = \begin{cases} \text{estimatePathCost}(x,t) & \text{path from } x \text{ to } t \text{ traversable} \\ \infty & \text{path blocked,} \end{cases}$$

where *estimatePathCost*(x,t) returns the estimated path cost in the map that contains known blockades. This is an optimistic estimate, as we do not assume any knowledge about where additional blockades may appear.

A disaster recovery mission is considered completed as soon as the set of targets T is empty, that is, all of the targets have been visited by an exploring robot.

3.5.2. Formulation as a Temporal Planning Problem

To apply our coordination approach to the disaster recovery problem described above, we encode the system's state and possible actions using PDDL statements. This PDDL description then serves as input to our coordination algorithm as depicted in the system overview in Figure 3.2.

First, we define which types of objects are involved. In this scenario, the relevant objects are the robots that can be either exploring robots or clearing robots and the map locations that can be mission targets or blockades. Figure 3.7 a shows the corresponding PDDL statements. Second, we specify the predicates that we use to define the problem states. The state in the disaster recovery problem can be specified using

```
(at ?r - robot ?x - location)
```

which describes that the robot r is at position x,

```
(cleared ?b - blockade)
```

which describes that blockade b has been cleared, and

```
(explored ?t - mission_target)
```

which describes that mission target t has been visited by an exploring robot.

Third, we provide the actions that change the state of the system. There are four kinds of movement actions: We distinguish between actions that explore a mission target and actions that have a goal position other than a mission target. We furthermore differentiate actions whose start location is a blockade and actions that start at a location in free space. As an example, the PDDL definition given in Figure 3.7 b defines a movement action that

a)
```
(:types
  robot - object
  clearer explorer - robot
  location - object
  free_location - location
  mission_target - free_location
  blockade - location)
(:predicates
    (at ?r - robot ?x - location)
    (cleared ?t - blockade)
    (explored ?t - mission_target) )
```

b)
```
(:durative-action explore-from-blockade
  :parameters (?r - explorer
               ?s - blockade
               ?g - mission_target)
  :duration (= ?duration [pathCost ?r ?s ?g])
  :condition (and
               (at start (at ?r ?s))
               (at start (not (explored ?g)))
               (at start (not (= ?s ?g)))
               (at start (cleared ?s)) )
  :effect (and
            (at start (not (at ?r ?s)))
            (at end (at ?r ?g))
            (at start (explored ?g)) ) )
```

Figure 3.7: Examples of PDDL definitions used in the disaster recovery domain. a: Definition of the types and predicates. b: Definition of the `explore-from-blockade` action.

explores a target starting from a blockade. It can be seen, that movement actions which start at a blockade require the blockade to be cleared before execution. All movement actions require the robot to be at the start location and result in the robot being at the goal location. The costs of these actions are defined as the estimated path costs and are determined via the modular interface much the same as in the marsupial exploration scenario. The goal location of all exploration actions is a mission target t. Once such an action is executed, the current state is changed so that the corresponding predicate `(explored t)` is set to true.

In addition to exploration actions, we define a `clear` action to coordinate the group of clearing robots. This action expresses that a given blockade b is cleared by a clearing robot that is positioned at the corresponding location. The duration of the action is defined by the time the robot takes to clear the blockade. In our experiments we assume that this value depends on the size of the blockade. After the action is executed, the predicate `(cleared b)` is set to true in the current state. Finally, the initial and goal state are defined. In the initial state, no blockades have been discovered and none of the mission targets have been explored. The goal state is defined as the state that describes all mission targets as explored.

The coordination loop depicted in the system overview (see Figure 3.2) is executed continuously until the goal state is reached. While the description of the actions remain unchanged during consecutive planning cycles, the current state of the system is updated to reflect the current state of the system and the environment. The updated state description defines all robots to be at their specific locations and blockades are added to the state whenever they are discovered. Previously cleared blockades or targets that have been explored are irrelevant to the current problem and thus are ignored in the problem description.

3.6. Evaluation

The approach described in this chapter has been implemented and evaluated thoroughly using a multi-robot simulation system. The experiments are designed to show that explicitly planning symbolic actions leads to a significantly more efficient coordination than using a heuristic extension of previous coordination approaches. We evaluated our coordination framework in the two problem domains introduced in Section 3.4 and Section 3.5. In addition, we show that our approach is able to produce efficient coordination plans in reasonable time and thus can be employed in a real-world system.

3.6.1. Simulation System

To evaluate our coordination approach quantitatively, we developed a simulation system that is able to simulate large teams of heterogeneous robots. In our current system, we

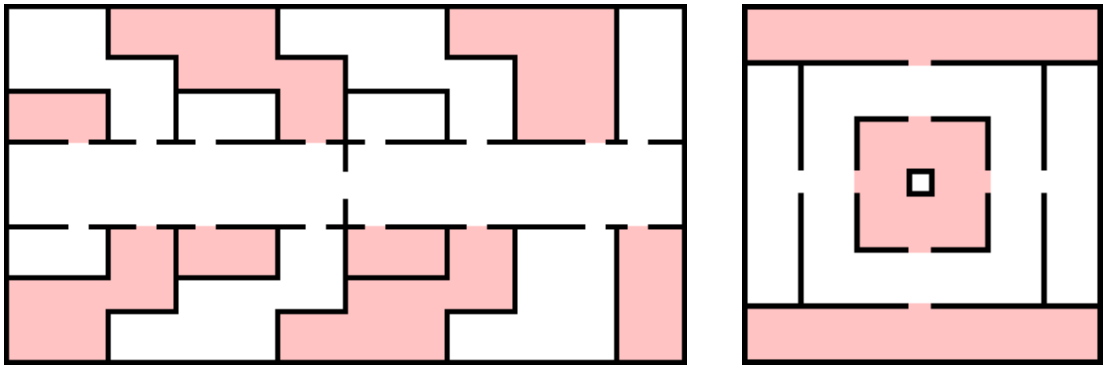


Figure 3.8: Simulated environments in the marsupial exploration experiments: *office* (left) and *maze* (right). White areas can only be traversed by carriers while red areas can only be explored by rovers.

also simulate laser range sensors. Sensor and odometry noise are not considered since we focus on the coordination aspects of the problems. The environment is modeled using a grid map with additional traversability information and semantic annotations such as mission targets or blockades.

3.6.2. Exploration with Marsupial Teams

We simulated teams of marsupial robots to evaluate our approach introduced in Section 3.4. The simulated environments contain areas that can only be traversed by rovers and areas that can only be traversed by carriers. The rover robots in most real marsupial systems are considerably slower than the carrier robots. In the evaluation we account for this difference by simulating carrier robots that are twice as fast as rovers. Furthermore, the maximum sensor range of the carriers is also twice as far as the sensor range of the rover robots.

We evaluated robot teams of varying sizes and different environments have been used in the simulation. Two of the environments we used in our experiments can be seen in Figure 3.8. The *office* environment resembles a typical office building with two corridors and a number of rooms. Some of the rooms can only be explored by rovers, those are depicted as red areas in the maps. The second environment, in the following referred to as the *maze* environment, features a central area that can only be explored by rovers. In contrast to the areas in the *office* environment, it has multiple meeting points that can be used for deployment. A visualization generated by the simulator is depicted in Figure 3.9. It shows a representative coordination problem where exploration targets as well as symbolic actions need to be considered.

To get an intuition of the extent of the environment, one can look at the ratio between the size of the environment and the maximum sensor range of the robots. In many real office buildings, such as building 079 on the Freiburg campus (see Figure 2.10), this ratio

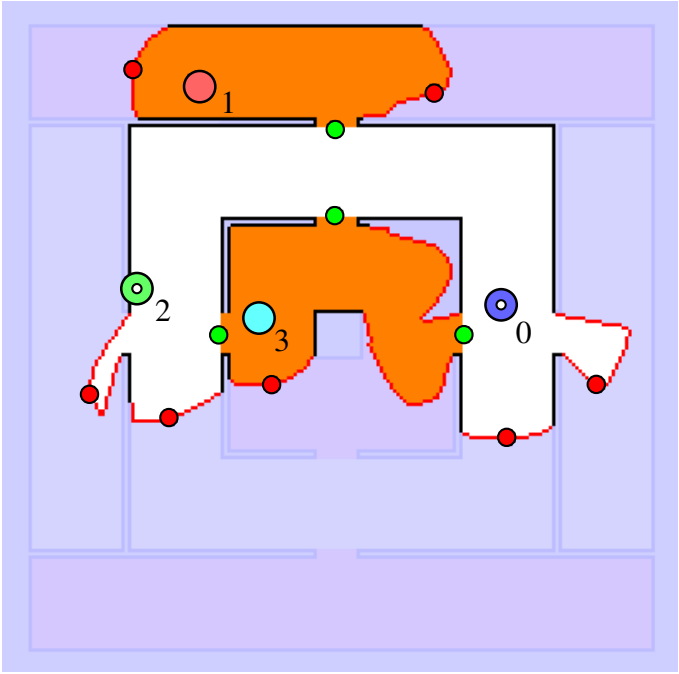


Figure 3.9: Visualization of a simulated run in the *maze* environment. Robots are depicted as disks with id numbers and carriers are marked by a white dot. Small circles depict exploration targets (red) and meeting points (green). In this situation, rover 3 could be retrieved by carrier 0 or carrier 2 at any of the three meeting points in its vicinity.

is around 10 for a maximum sensor range of 4 m. The ratio for the *maze* environment is 10.2 for rovers and 5.1 for carriers. In the considerably larger *office* environment the ratio increases to 23.8 and 11.9 respectively.

In both environments, we simulated 30 exploration runs with random initial robot positions. Exploration targets were determined using the frontiers approach and neighboring exploration targets were clustered using visibility constraints similar to the approach proposed by Burgard *et al.* (2005).

3.6.3. Baseline Approach

We compared our coordination algorithm to a heuristic extension of a method that assigns robots to target locations based on cost estimates (Stachniss, 2009). In this approach, the Hungarian method is used to compute the cost-optimal assignment of robots to exploration targets (see Section 2.3). To adapt this method to the problem of exploration with marsupial teams, we first assign the carriers to exploration targets independent of whether they can access those targets or not. The assignment is solely based on the estimated travel cost of the carriers, ignoring traversability constraints in the estimation. The rovers are then deployed as follows: Whenever a carrier is assigned to a target that it cannot explore itself, it will move to the nearest connecting meeting point and deploy a rover there.

This rover then explores the targets reachable from the meeting point. As soon as it has finished exploring them, it returns to the meeting point. In our experiments, we assume a limited number of rovers per carrier. This will lead to situations in which a carrier needs to deploy a rover but has none available. The baseline approach then requires the carrier to first retrieve a rover.

The baseline approach as described above closely resembles heuristic deployment rules that have been applied in previous marsupial systems (Murphy et al., 1999). Note that the baseline approach could be improved by introducing more complex reasoning. For example, it could anticipate cases in which no rovers are available to a carrier and then avoid assigning this carrier to targets that are only reachable by rovers. This would introduce additional state variables that the coordination system would need to maintain. In fact, this kind of reasoning would approximate a planning method such as the one we apply in our approach.

3.6.4. Results

An overview of the results obtained in the experiments is given in Figure 3.10. It can be seen that our approach explores the environment significantly faster than the baseline method in all configurations. By considering deployment and pick-up actions explicitly, our approach avoids long travel paths and detours that result from the pick-up heuristic in the baseline. In addition, it uses its larger planning horizon to plan sequences of actions while the baseline approach computes exactly one action per robot. Even though we execute only the first action in the sequence, robots often are in a good initial position for the next action.

Especially smaller teams of up to six robots profit considerably from our coordination method. Larger teams can, to some degree, compensate inefficiencies of the coordination when a good distribution in the environment is achieved and many targets can be approached simultaneously. In the settings we evaluated, this effect can be noticed when more than three carriers are used. The number of robots for which this effect occurs clearly depends on the structure and size of the environment.

To evaluate the amount of unnecessary movement in larger teams, we evaluated a further benchmark. We computed the exploration quality according to (Zlot et al., 2002) as

$$Q = \frac{1}{A} \sum_{i=1}^{n} d_i, \tag{3.1}$$

where A is the total area of the environment and d_i denotes the distance traveled by robot i. This measure can intuitively be understood as the average area each robot explores per movement.

The results given in Figure 3.11 show that our approach reaches a significantly higher

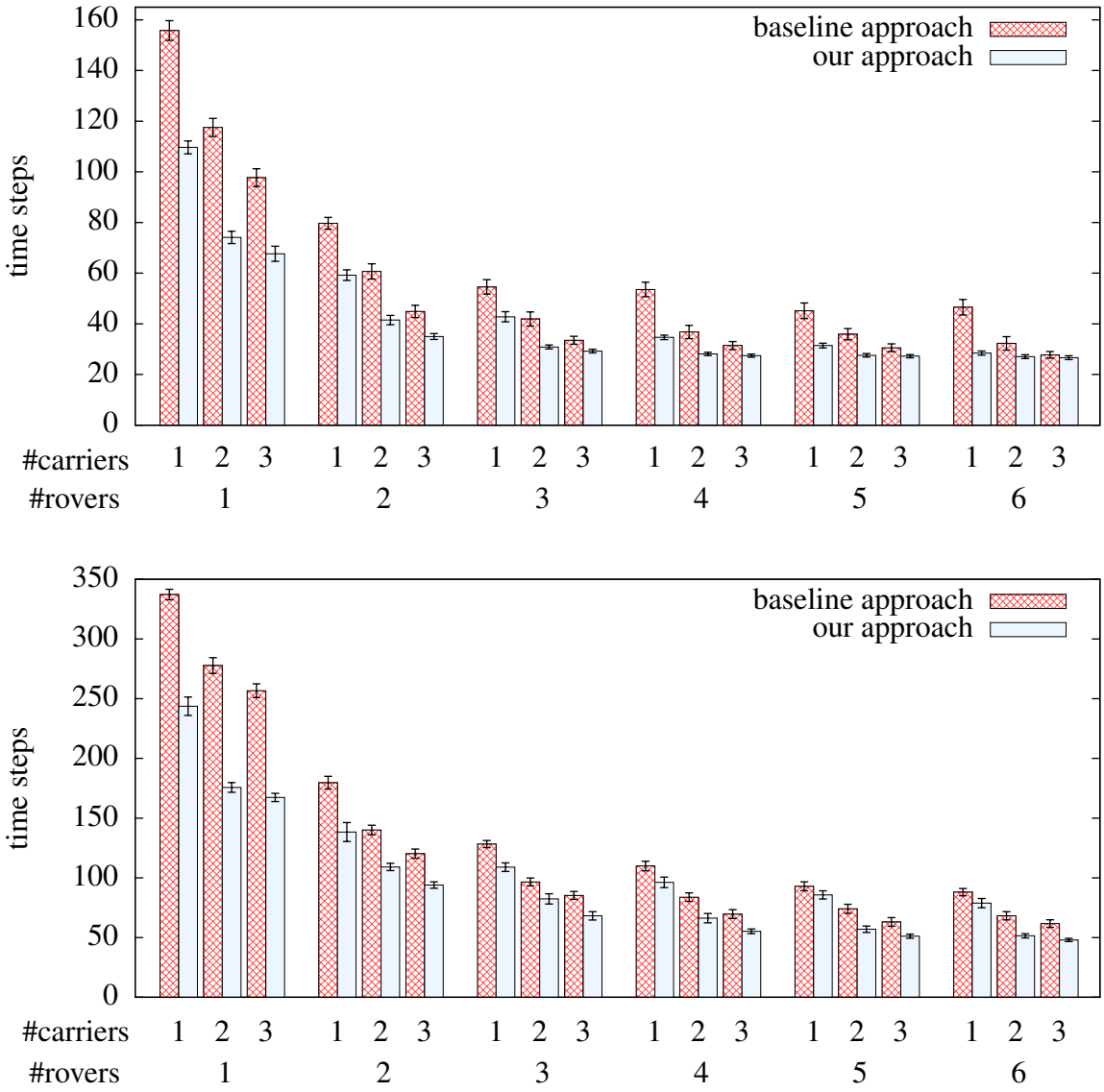


Figure 3.10: Exploration time obtained with our approach compared to the ad-hoc method in the *maze* environment (top) and in the *office* environment (bottom) for varying team sizes (number of carriers and number of rovers per carrier). The error bars indicate the 95% confidence intervals.

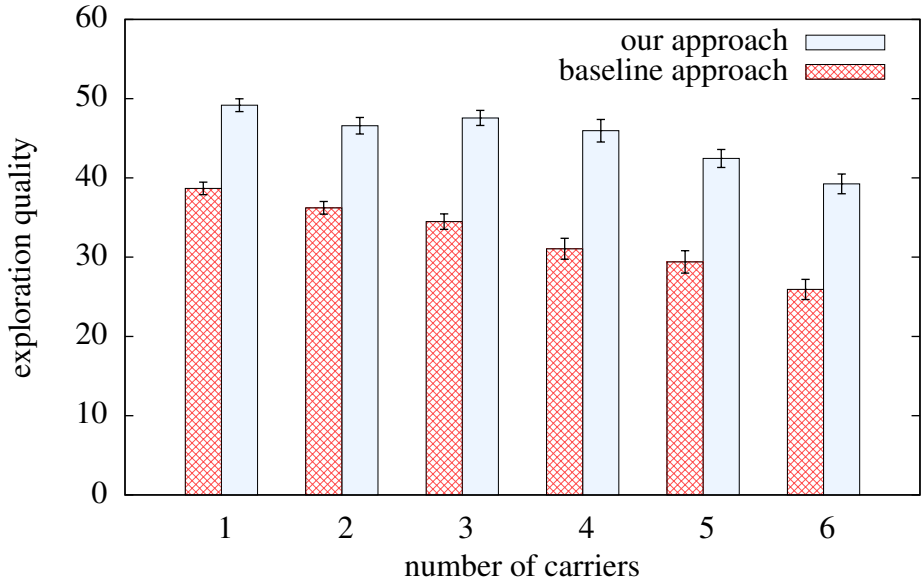


Figure 3.11: Exploration quality in the *office* environment over 30 runs using two rovers per carrier. The higher the value, the better the coordination. The error bars indicate the 95% confidence intervals. Note that similar results were obtained for the *maze* environment.

exploration quality. Especially larger teams of robots are coordinated more efficiently, so that unnecessary movements are avoided. The reason for this improvement lies in the larger planning horizon of our approach compared to the baseline. Our approach anticipates future states of the system, especially the position of the robots, while the baseline approach computes actions solely on the basis of the current state of the system. Avoiding unnecessary movements is a significant advantage on its own since there usually is a direct correlation between the distance traveled and the power consumption of the vehicles. In this way, our approach will save resources in larger teams.

3.6.5. Coordinated Disaster Recovery

To evaluate the performance of our coordination approach in the disaster recovery domain, we simulated teams of exploring and clearing robots. While the exploring robots are able to visit mission targets, the clearing robots are able to clear blocked paths in the environment.

Several environments have been evaluated and two of those can be seen in Figure 3.12. The first environment is based on the Fort Sam Houston hospital, in the following called hospital map. It features long corridors and a large number of rooms. In our experiments, a set of 15 mission targets and 18 blockades are placed throughout the building. The ratio between the size of the environment and the maximum sensor range of the robots is 27.05. The second environment is based on the blueprint of a nuclear power plant (referred to

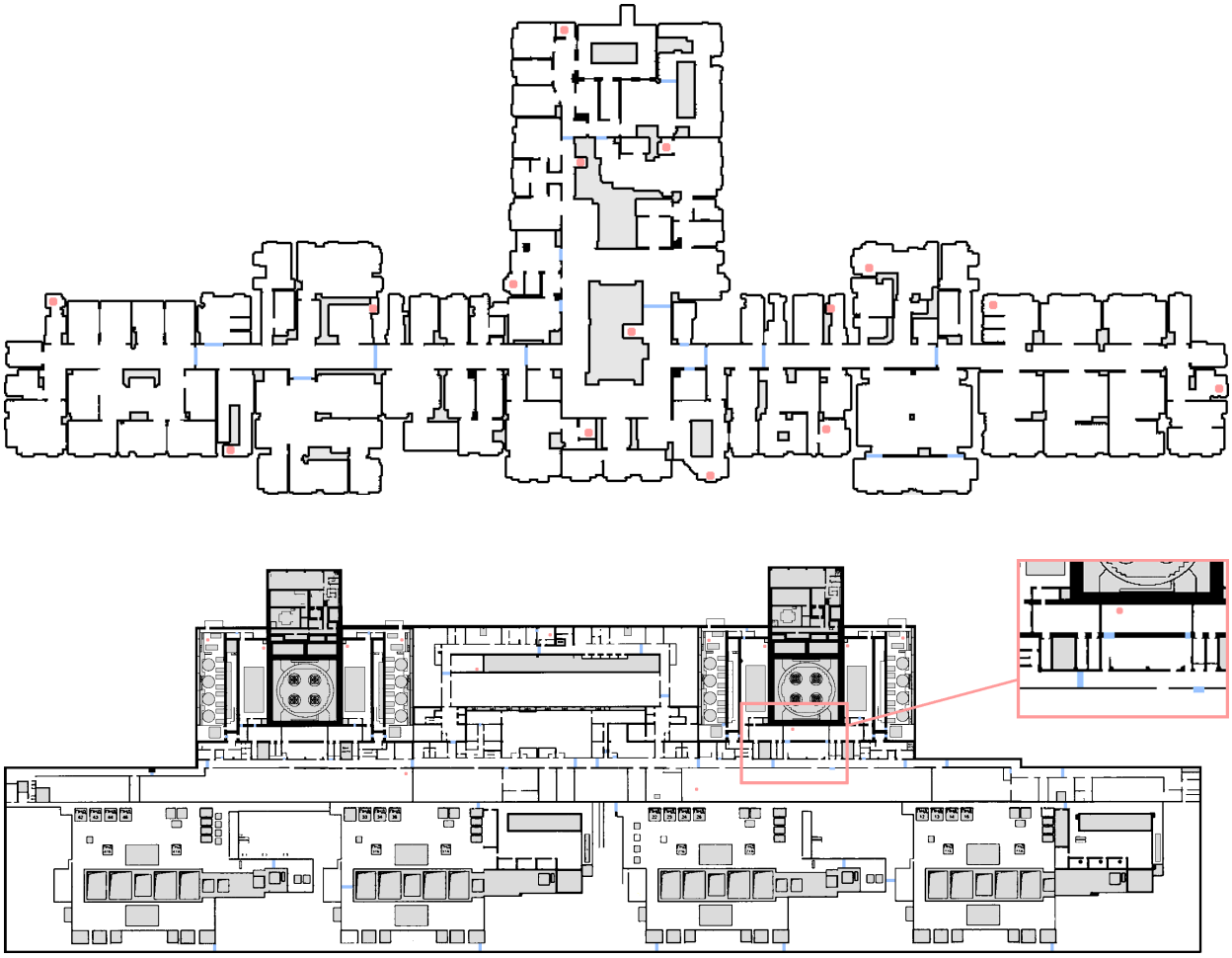


Figure 3.12: Simulated environments in the disaster recovery experiments: *hospital* (top) and *power plant* (bottom). Mission targets are visualized by red dots, blockades are shown as blue lines. Gray areas can not be accessed by the robots. In the experiments, blockades were not known to the robots a priori but could be detected by their sensors.

as the power plant map). In this map, 13 mission targets are grouped around the two reactor cores and the two control rooms. Paths are blocked at 58 locations throughout the building. The sensor ratio in this fairly large environment is 65.9.

We simulated teams of one to four exploring robots that are accompanied by one to four clearing robots. Simulated clearing robots and exploring robots have an identical maximum sensor range and driving speed. The time to clear a blockade depends on its size which can be determined by the robots using their sensors. For each team size, we simulated 30 runs starting at random positions.

3.6.6. Baseline Approach

We compared our coordination approach to an ad-hoc extension of a cost-based coordination method. In this baseline approach, the exploring robots are assigned to mission targets using the Hungarian Method (see Section 2.3). Intuitively, this method assigns each exploring robot to the closest mission target while avoiding to assign multiple robots to the same target. The assignment process is repeated whenever one of the robots reaches a target, a path is sensed blocked, or a blockade has been cleared.

As soon as blockades are discovered in the environment, clearing robots are assigned to clear those blockades. This assignment is determined using the Hungarian Method based on the estimated travel time to the blockades. Note that in the baseline approach, the coordination of the clearing robots does not depend on the mission targets or the assignment of the exploring robots.

3.6.7. Results

The results obtained in the evaluation of the disaster recovery experiments are shown in Figure 3.13. In the *hospital* environment, our approach performed significantly better in all simulated settings. By planning sequences of clearing and exploration actions our approach is able to minimize the time that clearing robots spent waiting for blockades to be cleared.

Similar results could be obtained in the *power plant* map. Due to the large extent of this map and the random placement of start locations, there is a higher variance in the overall runtimes. Using the paired t-test (Hsu and Lachenbruch, 2007), however, a significant improvement of our approach over the baseline approach could be shown in this map, too.

It is worth mentioning that a significant improvement could not be shown for all simulated team sizes in the *power plant* map when the number of blockades was reduced by 20% and the team size was small (in our experiments: one explorer and two clearing robots). In this special case, the independent assignment of clearing and exploring robots using the Hungarian method provided comparable overall runtimes (see Figure 3.14).

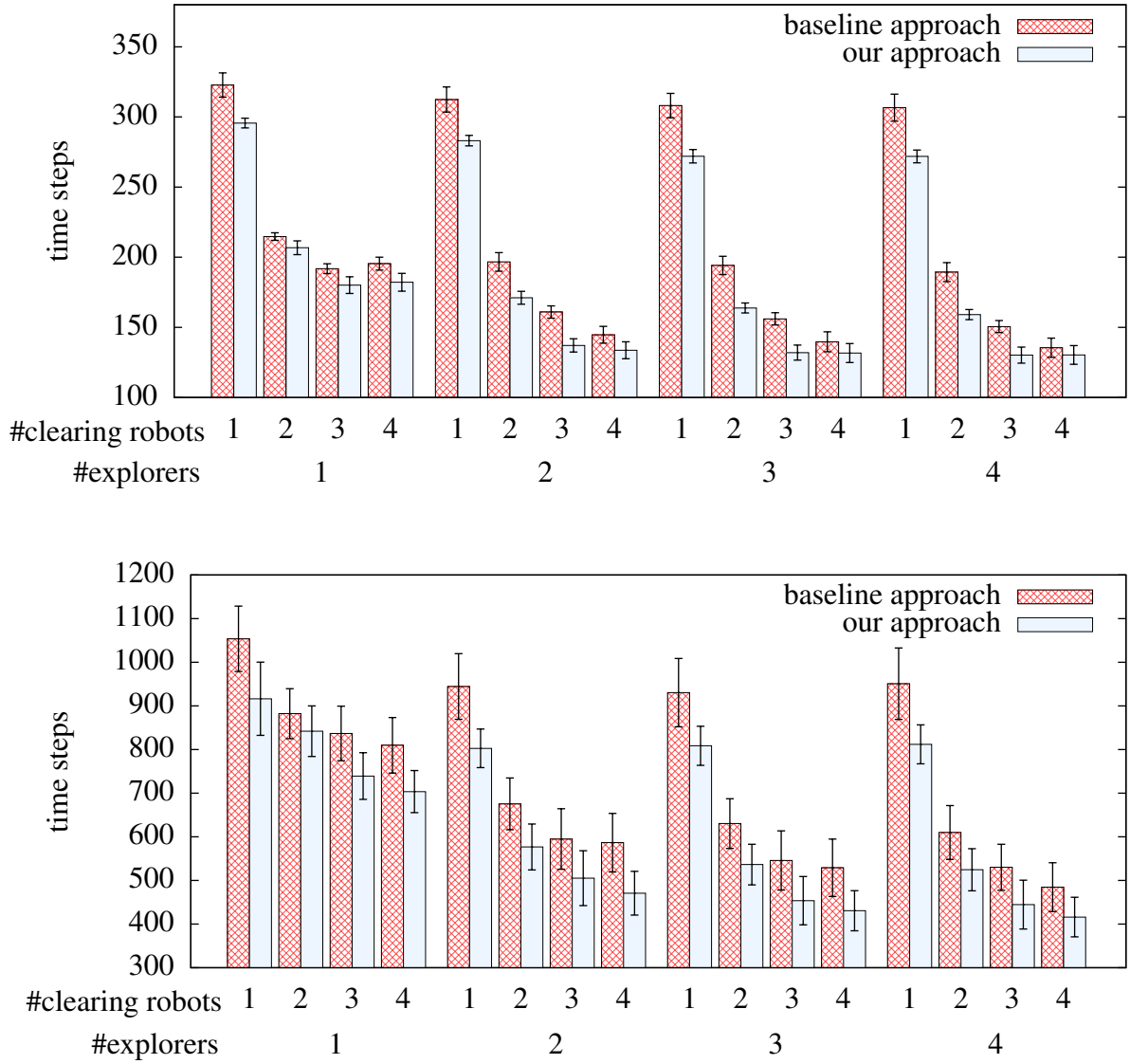


Figure 3.13: Runtime until all mission targets are visited using our approach compared to the result using the ad-hoc method in the *hospital* environment (top) and in the *power plant* environment (bottom) for varying team sizes (number of exploring and clearing robots). The error bars indicate the 95% confidence intervals.

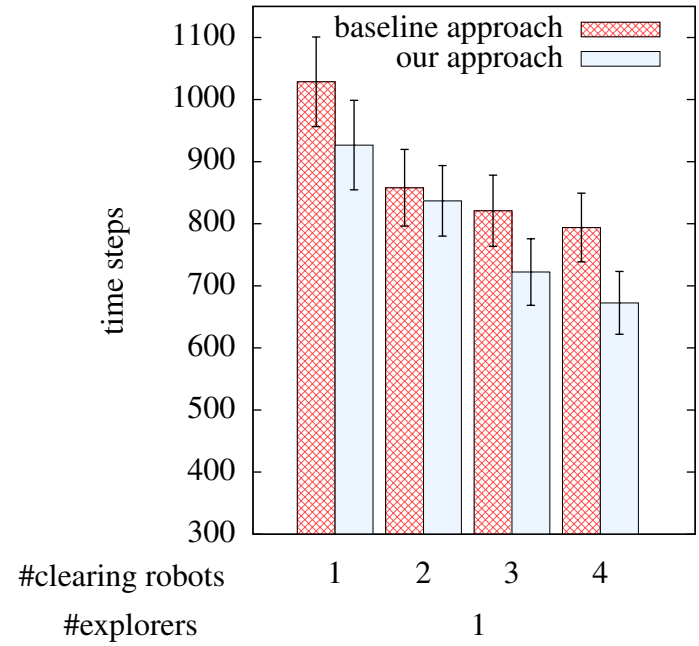


Figure 3.14: Time to visit all mission targets using our approach compared to the result using the baseline method in the *power plant* environment with 20% fewer blockades. The error bars indicate the 95% confidence intervals.

The low number of blockades enables the baseline approach to simply clear all blockades independent of whether the exploring robot needs a particular path cleared. For this reason, our coordination approach does not profit from its additional lookahead and the ability to plan sequences of actions to the same degree as it does in the more complex settings.

3.6.8. Planner Runtime

As TFD/M, the temporal planner used in our approach, cannot guarantee optimality, we seek to determine close-to-optimal plans. To analyze the tradeoff between planner runtime and plan quality, we simulated a recovery mission using a team of three exploring and two clearing robots in the *power plant* map. In this experiment, the planner was given a maximum planning time of 900 s and we measured the time until a plan was generated that was at most 5% worse than the best plan found until the timeout was reached. The experiment was performed on a 2.7 GHz AMD Opteron 2384 system using one core per task.

The result of this evaluation is shown in Figure 3.15. It can be seen that the number of planning targets (exploration targets and blockades combined) has a dominant influence on the planning time. It can also be seen that for a team size of five robots the close-to-optimal plan can be found in less than around 10 s as long as there are less than ten

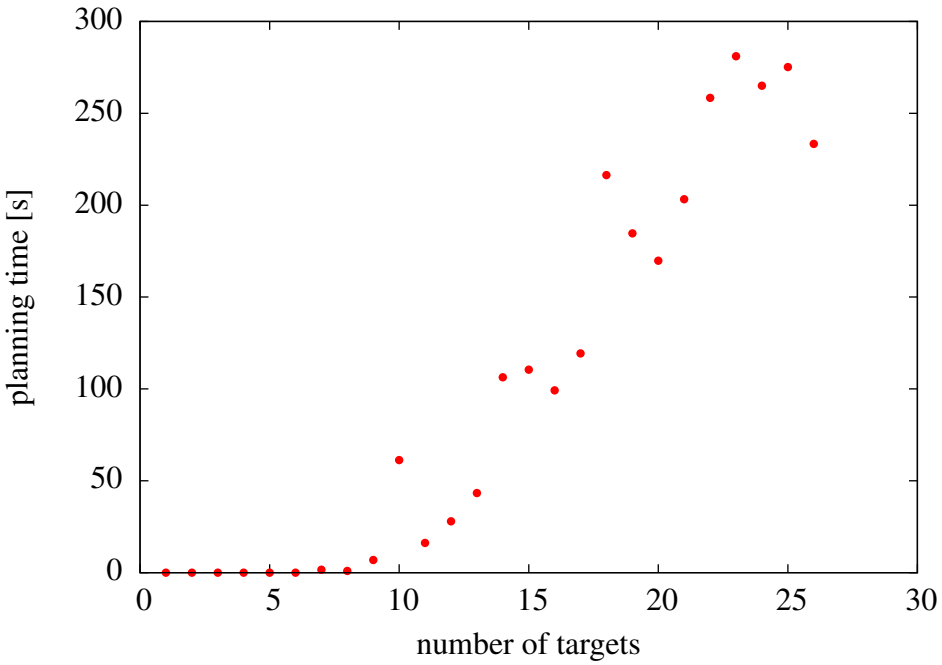


Figure 3.15: Average runtime until a plan is computed that is at most 5% worse than the best plan computed until a timeout of 900 s is reached. For this evaluation, three exploring and two clearing robots were coordinated in the *power plant* map.

planning targets. Note however that a feasible plan is usually found much faster than the best plan and in all our experiments, it was found within a 30 s planning limit.

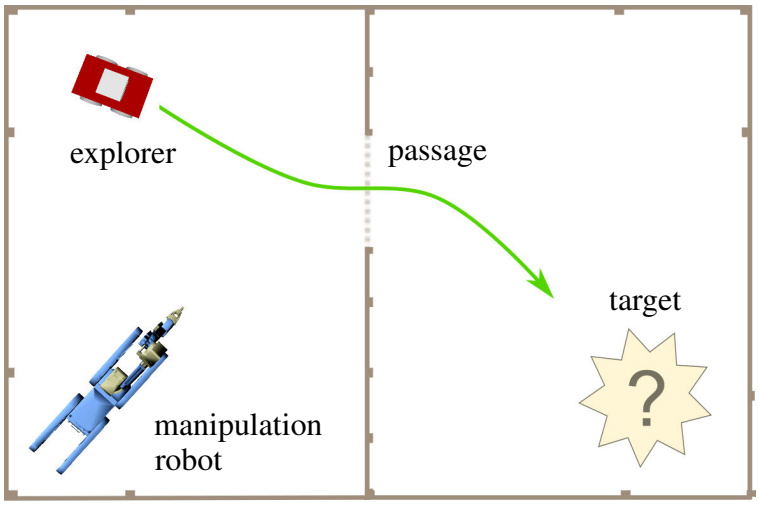


Figure 3.16: Two robots cooperate to explore an area. The goal of the robots is to explore the area to the right. A narrow passage connects both areas but can be blocked by an obstacle.

3.6.9. Real World Application

In this experiment, we applied our approach to coordinate a real team of robots. The team consisted of an ActiveMedia Pioneer 2AT equipped with a tilting laser scanner (the explorer) and of a Telemax mobile manipulation platform (the manipulation robot). The goal of the robots was to explore an area which was reachable through a narrow passage that only the exploring robot could traverse. The setting is illustrated in Figure 3.16. In our experiments, the passage could be free or it could be blocked by movable obstacles. The explorer was able to detect such obstacles and to inform the coordination module about the blockade. The manipulation robot was able to grasp detected obstacles and to remove them from the passage. Once the goal area was reached by the exploring robot, it was explored by sweeping its sensor to acquire a 3D scan and the mission was then considered completed.

We solved the coordination problem using the approach described in the disaster recovery application in Section 3.5. A sequence of actions that was planned in a particular situation can be seen in Figure 3.17. Here, the passage was first blocked by an obstacle. The obstacle was then detected by the exploring robot. The coordination approach assigned the manipulation robot to clear the passage. After the passage was cleared and the manipulation robot moved to a waiting position, the exploring robot was able to reach the goal area and complete the task.

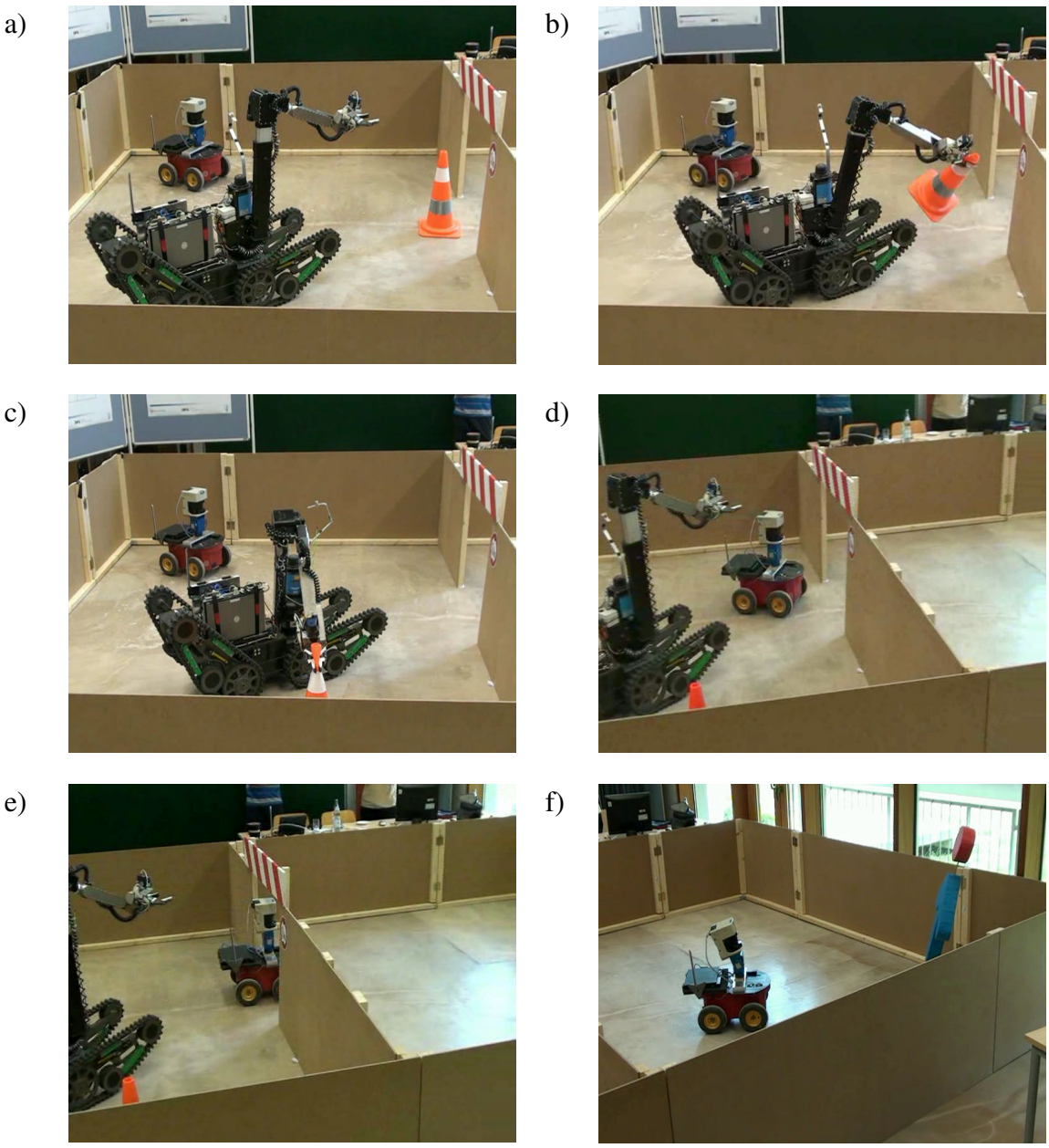


Figure 3.17: Sequence of actions in real-world experiment. From a) to f): The obstacle is detected by the wheeled exploring robot, the obstacle is removed by the manipulation robot, the manipulation robot returns to its initial position and the exploring robot traverses the narrow passage. Finally, the explorer takes a 3D scan of the target area.

3.7. Related Work

When multiple robots explore an unknown environment the coordination strategy has the biggest influence on the performance of the system. Coordination methods for homogeneous teams, in which all members are equipped equally, have been studied extensively in the past. In those cases, the coordination task is often formulated as an assignment problem. Such approaches assign robots to exploration targets based on a suitable cost measure.

Several methods have been presented to determine an assignment of robots to exploration targets. Most methods determine targets by extracting the frontier between explored and unexplored areas. This target generation approach was introduced by Yamauchi (1997) who also proposed a decentralized multi-robot coordination technique that assigns each robot to the nearest frontier (Yamauchi, 1998). The drawback of this coordination method is that several robots can be assigned to the same target.

To improve the overall performance, later approaches introduced techniques to avoid redundant work. Zlot and colleagues (2002) proposed an architecture in which the exploration is guided by a market economy. They consider sequences of potential target locations for each robot and trade tasks between the robots using single-item first-price sealed-bid auctions. Such auction-based techniques have also been applied by Berhault *et al.* (2003) to assign robots to bundles of targets so that synergy effects between targets are exploited.

Burgard *et al.* (2005) presented an iterative assignment method based on the estimated cost of reaching a target. It additionally considers visibility constraints between targets. Ko *et al.* (2003) and Stachniss (2009) presented algorithms that use the Hungarian method to compute the assignments of robots to exploration targets.

Heterogeneous teams of robots consist of robots with different abilities. An early approach towards coordination of heterogeneous robot systems during exploration was presented by Singh and Fujimura (1993b). In this approach, if a robot is too big to pass through a narrow passage, other robots are informed about this task. A further heterogeneous system was presented by Grabowski and Navarro-Serment (2000). Coordination, however, is performed manually in this approach.

Heterogeneous teams have also been studied in the RoboCup domain that is concerned with soccer-playing teams of robots. In this dynamic environment, most approaches choose a decentralized coordination approach that relies on the sensing and reasoning ability of each individual robot (Iocchi et al., 2003; Kok et al., 2005; Kiener and Von Stryk, 2010). In this way such teams are more robust against network failures and cumulative measurement uncertainty.

Whenever small robots with low traveling speeds or limited power resources are used in a heterogeneous robot team, it is advantageous to have larger robots transport the smaller ones to avoid a serious penalty in exploration time or power consumption (Mur-

phy et al., 1999). The larger carrier robots are frequently referred to as *marsupial robots*. One of the first physical implementation of a marsupial system was presented by Murphy *et al.* (1999) who also described ad-hoc methods to deploy the smaller robot. Kadioglu and Papanikolopoulos (2003) presented a further physical implementation of a marsupial system. Dellaert *et al.* (2002) deployed a team of legged robots using a carrier robot in a rescue scenario. Denner and Papanikolopoulos (2007) presented a deployment method for marsupial teams that explicitly considers power constraints. In all of the previously described exploration systems, deployment and retrieval of robots in marsupial teams is determined by handcrafted methods. In contrast to that, the approach presented in this chapter explicitly takes these symbolic actions into account when coordinating the exploration.

The approach presented in this chapter employs domain independent planning techniques to coordinate multiple robots. Domain independent planning is a sub-field of artificial intelligence that has been investigated thoroughly. A classical planning problem consists of a set of state-variables with finite domains, an initial state, a set of actions and a goal condition. An action is defined by a precondition and its effects, which is a set of variable assignments. A solution for a classical planning problem is then defined as a finite sequence of actions that leads from the initial state to a goal state. There exist several efficient planning systems for classical planning problems (Bonet and Geffner, 2001; Hoffmann and Nebel, 2001; Chen et al., 2006; Helmert, 2006; Richter and Westphal, 2010).

In multi-robot coordination, actions need to be executed concurrently and they usually have variable durations. These temporal constraints cannot be handled by classical planning approaches and constitute a new class of problems. The task of finding solutions to these problems is known as temporal planning and several efficient approaches have been presented (Gerevini et al., 2008; Eyerich et al., 2009). The predominant approach of solving temporal planning problems is forward search with a search guidance function using A^* or similar algorithms. Most approaches support the usage of numerical state variables. In contrast to binary and multi-valued state variables, numeric state variables have an infinite continuous value domain. While numeric state variables lead to undecidability even when used in a very limited form (Helmert, 2002), they are considered to be of high importance when modeling real world domains.

While temporal planners generate sequences of abstract actions, most real-world applications in robotics require explicit motion commands that can be executed by the robots. For this reason, a number of approaches have been proposed that integrate other reasoners, such as motion planners, to generate low-level actions. The work by Cambon *et al.* (2009) focuses on the integration of manipulation planning with a symbolic planner. Kaelbling and Lozano-Perez (2011) address the problem of successor generation during planning. They use so called suggesters that provide, for example, new motion paths or goal locations. The work described in this chapter uses the planner TFD/M (Dorn-

hege et al., 2009), a variant of the temporal fast downward planning system originally developed by Eyerich *et al.* (2009). TFD/M provides a generic interface for the integration of external modules that compute the values of state variables, action costs, and numerical effects during the planning process using sub-processes. By means of these sub-processes, we combine temporal planning with path planners traditionally used for robot navigation and coordination.

Autonomous disaster recovery in partially known environments, as introduced in this chapter, has not been studied in depth. An auction-based coordination method as well as a method based on genetic algorithms is presented by Jones *et al.* (2011). In their work, they coordinate a simulated team of fire trucks and bulldozers leading to a coordination problem that is similar to the disaster recovery domain considered in this chapter. In contrast to our approach, however, blockades are assumed to be known beforehand. It is unclear, how this approach can be extended to partially known environments. Koes *et al.* (2005) described an approach that employs mixed integer linear programming (MILP) to coordinate a heterogeneous team in a search and rescue scenario. In this domain, disaster victims need to be found and identified in a known environment. This task involves assigning robots with different capabilities to appropriate tasks. Similar to our approach, path costs between target locations are explicitly taken into account using a robotic path planner. The static nature of the environment, however, allows all paths to be pre-computed for the entire runtime. Such a pre-computation is not possible in the case of partially known environments.

3.8. Discussion

In this chapter we presented a novel technique to coordinate heterogeneous teams of exploring robots. We assumed that to explore the environment, the robots have to move in the environment and additionally are required to execute actions such as removing an obstacle or operating an elevator. These so-called symbolic actions stand in contrast to navigation actions that move robots to a given goal position in order to explore target locations. Our method integrates a symbolic temporal planning system and a robotic path planner. This combination allows our system to consider planning problems that include symbolic actions. To coordinate a team of robots, we generate a symbolic description of the current state of the system and of the goal state. These descriptions serve as input to the symbolic planner. To solve the coordination problem, the symbolic planner uses the path planner to efficiently estimate travel costs for the robots. In contrast to cost-based coordination approaches, our system is able to explicitly plan for the execution of symbolic actions. Still, the use of action costs that are determined by the robotic path planner allows us to generate time-efficient solutions. In this way, our approach combines the strength of cost-based coordination approaches with the flexibility of symbolic planning

systems. By planning symbolic actions explicitly, the system is able to take into account actions such as manipulation actions or deployment and retrieval of robots.

The approach was implemented and evaluated extensively in simulated environments. We evaluated two distinct settings. First, we coordinated teams of marsupial robots that were able to deploy and pick up smaller robots. Second, we simulated a disaster scenario in which the task was to clear blockades and to perform pre-defined actions at certain critical locations in the environment. A similar setting was also investigated using a team of real robots.

The experimental results demonstrate that our approach can effectively coordinate large teams of robots and significantly outperforms handcrafted strategies. In addition to that, our planning framework adds a substantial degree of flexibility to the system. For example, additional constraints such as power constraints for individual robots can be specified by adding adequate predicates to the problem description. Furthermore, other symbolic actions such as recharging batteries or deploying sensor nodes can be integrated in a straightforward way.

3.8.1. Limitations of the Approach

The planning system described in this chapter generates temporal plans for the robots based on the current knowledge about the world. We do not assume a model of the unknown information and thus rely on an optimistic problem formulation. While the robots move, their state changes and new information about the environment is perceived. Therefore, we execute the planning cycle (illustrated in Figure 3.2) in a continuous loop. The symbolic planer is implemented as an anytime algorithm that does not terminate after the first solution is generated. It produces a potentially non-optimal solution quickly and then improves upon this solution. If more than one solution is found, we set the planning timeout to 30 s. In the marsupial coordination setting, we evaluated our approach with up to 24 robots (6 carriers plus 18 rovers). For significantly larger teams, however, the problem complexity increases substantially and the solution reported by the anytime planner after 30 s may be suboptimal.

The exact problem size that can be solved close-to-optimally in a given planning time depends on the structure of the problem and on the team size. The evaluation of the planning time presented in Section 3.6.8 shows that the best plan will usually be found in less than 10 s when coordinating a team of five robots in a disaster recovery mission with up to ten mission targets. Many real-world scenarios will feature a similar complexity.

Part II.

Model Estimation for Navigation

Chapter 4
Multi-Robot SLAM

4.1. Introduction

To move autonomously from one place to another a robot needs to know the layout of the environment and this knowledge is typically encoded in a map. In an integrated system, it is advantageous that the map is generated by the robot itself in a training phase. If the robot knew its trajectory precisely, it could create a map from its sensor readings in a straightforward way using *mapping with known poses* (Thrun et al., 2005). Most robots, however, cannot precisely determine their position without a map. This codependency of mapping and localization leads to the problem referred to as *simultaneous localization and mapping* (SLAM): The robot has to estimate both its trajectory and a map of the environment from sensor data. A large variety of techniques has been proposed to solve the SLAM problem. Extended Kalman filters (Leonard and Durrant-Whyte, 1991), sparse extended information filters (Thrun et al., 2004), graph-based maximum likelihood methods (Lu and Milios, 1997; Frese et al., 2005; Olson et al., 2006; Grisetti et al., 2007c), particle filters (Montemerlo and Thrun, 2003), and several other techniques have been applied. While efficient solutions exist to approach the SLAM problem with single robots, the problem of SLAM with multiple robots has not received the same amount of attention.

In this chapter, we address the problem of mapping environments with teams of several robots. In the experiments presented in Chapter 2, the relative positions of the robots were known because the robots started from the same area. In the approach presented in this chapter, we do not assume any such global pose estimates, nor do we require robots to meet and recognize each other. Our approach treats the multi-robot SLAM problem as an extension of single-robot SLAM and assumes that each robot in the team applies existing estimation techniques to solve a local SLAM problem. To combine the local estimates into one joint estimate, we have to determine the relative positions of the

robots. Therefore, we need to solve a global data association problem that answers the question: Which locations in the map of one robot correspond to locations in the map of another robot?

Our multi-robot SLAM approach extends existing single-robot SLAM solutions that are based on graphs. These methods maintain a graph of robot poses and observations and apply graph optimization algorithms to estimate the most likely trajectory of the robot given its observations. In our system, each robot in the team initially creates an independent pose graph. We then connect these graphs by determining the relative transformation between pairs of robot maps. Optimization of the joint graph is performed to estimate a globally consistent model.

We present two different data association modules for multi-robot SLAM. Both approaches generate constraints that connect local SLAM graphs but they differ in the underlying map representation that they use. Most mapping approaches use either grid maps or sets of landmarks to represent the environment. The decision which model to use in a practical application is largely influenced by the type of environment: In large open spaces with predefined landmarks, for example, feature-based approaches are often preferred, whereas grid maps have been widely used in unstructured environments and buildings. We present data association modules for both map representations. Our first method generates constraints between SLAM graphs by identifying structural similarities in grid maps of the environment. The approach is based on global Monte-Carlo localization: By localizing one robot in the grid map of another robot we obtain an estimate of the relative transformation between both maps. These estimates serve as global constraints that connect the SLAM graphs of the robots. The second data association approach is based on maps of landmark detections. We analyze the relative position of landmarks in the maps of each robot. To this end, we compute a triangulation of the landmarks and then identify matching triangles based on features that are invariant to Euclidean transformations. Triangle matches are used to compute the relative transformation between the local landmark maps of the robots.

In our experiments, we evaluated both scenarios described above, outdoor environments that contain predefined landmarks and structured indoor environments. To evaluate our grid-based approach, we applied it to the problem of consistently mapping buildings with multiple floors. An illustration of this application is given in Figure 4.1. In this setting, each floor of a building is mapped independently of the others. We apply our approach to generate constraints between the resulting independent SLAM graphs which are then used to estimate a globally consistent model of the complete building. This problem can be considered a generalization of the multi-robot SLAM problem since the case in which each robot maps parts of the same floor of a building is a special case of the more general multi-floor problem. As we will show in the evaluation, our approach is able to generate constraints between floors of buildings and that it estimates maps that are more accurate than those obtained with an approach that does not consider global

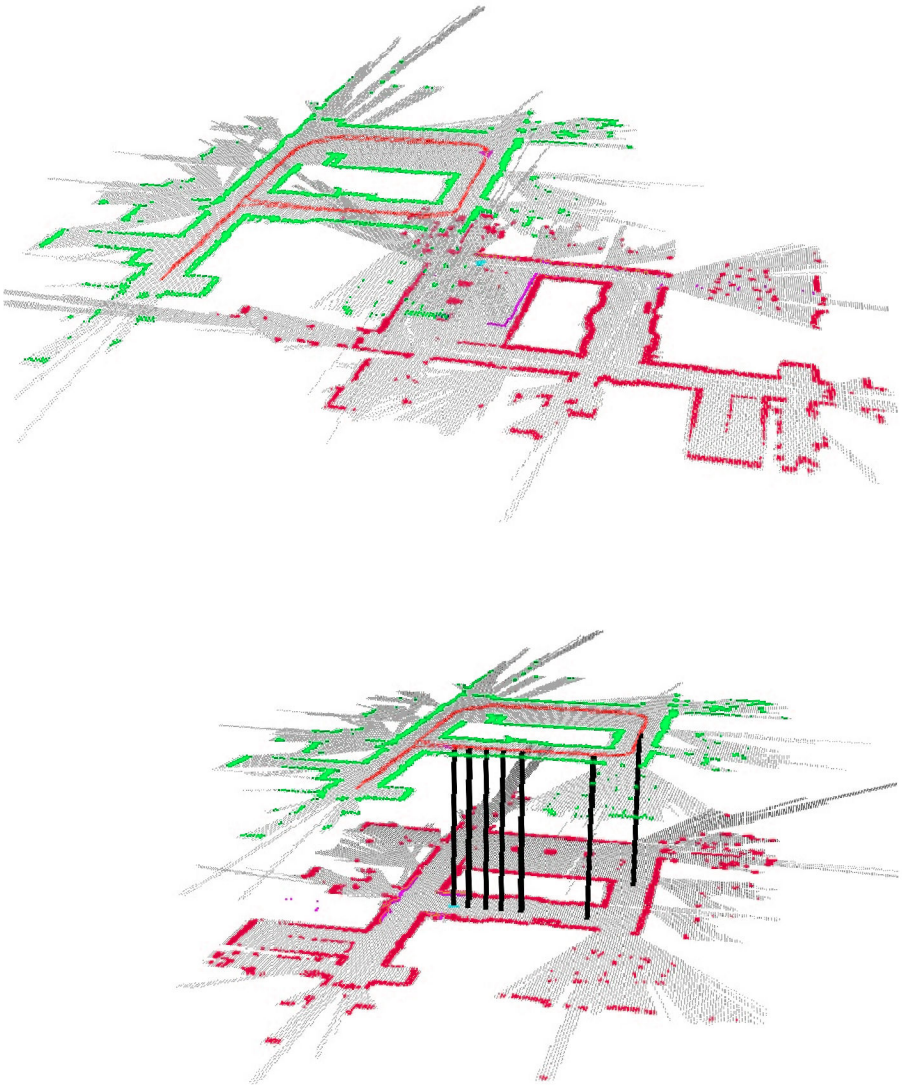


Figure 4.1: Illustration of inter-graph constraints in multi-floor building maps. With standard SLAM techniques the correct alignment of separate floor maps can not be derived in general (top). Our multi-robot SLAM approach, in contrast, generates constraints between the individual SLAM graphs recorded at different floors of a building. After optimizing the graph, the individual floors are aligned correctly (bottom).

constraints. The landmark-based approach was evaluated in the context of large outdoor environments that contain predefined landmarks. In our experiments, we detected poles and tree trunks from laser data. We applied our method to real world and simulated datasets and the results of the evaluation show that the approach is able to robustly align landmark maps of multiple robots.

This chapter is organized as follows. After introducing the graph-based SLAM technique we apply in our approach, we present the grid-based data association approach in Section 4.3 and the landmark-based approach in Section 4.4. The evaluation of both approaches is presented in Section 4.5 and Section 4.6 respectively. Finally, related work is discussed in Section 4.7.

4.2. Graph-based SLAM

The graph-based formulation of the SLAM problem, as introduced by Lu and Milios (1997), maintains a graph of poses. In this pose graph, the set of nodes V consists of robot poses p_i and the set of edges E represent spatial constraints between these poses, where each edge $\delta_{ji} \in E$ encodes a constraint between two poses p_j and p_i:

$$G = (V, E) \tag{4.1}$$

$$V = \{p_i\}\,,\ i \in \{1, \cdots, n\} \tag{4.2}$$

$$E = \{\delta_{ji}\}\,,\ j, i \in \{1, \cdots, n\}. \tag{4.3}$$

Usually, the poses in the graph approximate the trajectory of the robot and the constraints are computed from the robot's odometry and its sensor measurements. Both types of observations are afflicted with noise in realistic scenarios. For this reason, we store an information matrix Ω_{ji} along with each constraint δ_{ji} that expresses the uncertainty in the observation.

The goal of SLAM often is to create a map of the environment. To generate a map, we assume that a sensor measurement z_i is associated with each node in the pose graph p_i. By applying mapping with known poses, we can then compute a map based on the poses in the SLAM graph.

In the graph-based formulation, the SLAM problem can be divided into two independent sub-problems: First, one has to construct the pose graph from the data of the robot by generating poses and constraints. In the second step, one then has to estimate the most likely configuration of poses given the constraints. The first part of the problem is usually referred to as the SLAM front-end, while the optimization part is referred to as the SLAM back-end.

4.2.1. SLAM Back-end

The SLAM back-end seeks to find a configuration of the nodes in the pose graph $G = (V, E)$ that maximizes the likelihood of the observations. Most SLAM back-ends do not use the poses V directly but compute a parameterization x by applying a bijective mapping $g(V)$:

$$x = g(V) \tag{4.4}$$

$$x = (x_1, \cdots, x_n)^T. \tag{4.5}$$

The parameterization is specific to the SLAM back-end, our work applies the approach of Grisetti *et al.* (2009) which is based on a tree parameterization.

To derive an optimization criterion for pose graphs, let us look at a single constraint $\delta_{ji} \in E$ and its information matrix Ω_{ji}. Without loss of generality, we assume that δ_{ji} expresses an observation of node j as seen from node i. Let $f_{ji}(x)$ be a function that computes a zero noise observation according to the current configuration of the nodes j and i in the pose graph as expressed by x. Using this noise free observation, we define the error introduced by a constraint as

$$e_{ji}(x) = f_{ji}(x) - \delta_{ji} \tag{4.6}$$

and the residual is defined as

$$r_{ji} = -e_{ji}(x). \tag{4.7}$$

In an error free pose graph, both the error and the residual of a constraint are zero, since in this case $f_{ji}(x)$ equals δ_{ji}. To optimize the complete pose graph we need to consider the error of all constraints. Let $\mathcal{C}$ be the set of pairs of indices for which a constraint exists

$$\mathcal{C} = \{(j,i) \mid \delta_{ji} \in E\}. \tag{4.8}$$

Maximum likelihood approaches, such as the one we apply in our SLAM system, maximize the likelihood of the observations by minimizing their negative log likelihood. Assuming the constraints to be independent and the observation error to be Gaussian, this can be written as

$$x^* = \operatorname*{argmin}_x \sum_{(j,i)\in\mathcal{C}} r_{ji}(x)^T \, \Omega_{ji} \, r_{ji}(x). \tag{4.9}$$

To solve Equation (4.9), various techniques have been proposed (Duckett et al., 2000; Frese et al., 2005; Konolige, 2004; Lu and Milios, 1997; Olson et al., 2006). The ap-

proach of Grisetti *et al.* applies stochastic gradient descent to compute the most likely configuration of the nodes in the network. The problem of optimizing the graph is divided into a set of sub-problems by iteratively optimizing the graph for individual constraints. For each selected constraint δ_{ji} the configuration of the nodes in the pose graph is modified in the direction of the residual $r_{ji}(x)$, thereby decreasing the error $e_{ji}(x)$. For more details on the optimization step, we refer the reader to the work of Grisetti *et al.* (2009).

4.2.2. Extension to Multi-Robot SLAM

So far, we assumed that the poses in the graph were generated from the trajectory of a single robot. When several robots are used in a distributed system, an independent pose graph is generated for each robot. SLAM graphs are considered independent when no constraints exist that connects nodes from one graph to nodes of another graph. Such a situation can occur in different scenarios, for example, when multiple robots map an environment and their relative positions are unknown or when the same robot maps different parts of the environment in several runs and the transition between runs is lost. Both settings are essentially equivalent as long as the sensor data generated in either case are comparable, that is, a similar sensor was used in a similar setup. In the rest of our description we will assume that multiple robots were used to generate the data. However, the techniques we will present can be applied to both cases.

There exists a straightforward extension of the graph SLAM approach to the multi-robot case. In a multi-robot mapping system, each individual robot maintains a local pose graph $G_i = (V_i, E_i)$, where the SLAM front-end of each robot generates constraints between poses in V_i. To compute a solution to the multi-robot SLAM problem, the individual pose graphs need to be connected by further inter-graph constraints. Finding such constraints is the task of a multi-robot SLAM front-end. In the following we present two different multi-robot SLAM front-ends: the approach presented in Section 4.3 is based on grid maps while the approach given in Section 4.4 uses landmark maps.

4.3. Front-End for Multi-Robot SLAM based on Grid Maps

In this section, we will first present a SLAM front-end that generates pose graphs based on the laser range data of a single robot. We will then present a technique to generate constraints between SLAM graphs of several robots that performs global localization in grid maps.

4.3.1. SLAM Front-End for Single Robots

We assume that sensor data are generated by a mobile robot that provides odometry information and that is equipped with a laser range finder. As the robot traverses the environment, a number of laser measurements z_1 to z_t are obtained. To construct a pose graph, we first align the laser measurements using a gradient descent-based scan matcher that takes the odometry of the robot as its initial guess (Grisetti et al., 2007b). Then we add a node p_i to the pose graph for every measurement z_i. The pose of these nodes is determined based on the result of the scan matching procedure. Two consecutive nodes are connected by a constraint δ_{ji} that expresses the relative 2D transformation between the poses p_j and p_i. The information matrix Ω_{ji} associated with the constraint is determined using the covariance matrix returned by the scan matching method – the information matrix is given by the inverse of the covariance matrix (Olson et al., 2006).

In addition to constraints between consecutive nodes, we seek for constraints that result from loop closures, that is, when the robot revisits previously mapped areas. We look for such loop closing constraints when we add a new node to the pose graph by considering existing neighboring nodes.

4.3.2. Inter-Graph Constraints from Grid Maps

We generate constraints between independent SLAM graphs G_1 to G_n by computing grid maps from the graphs and then identifying similarities in those maps. To identify structurally similar areas, we use a global localization method based on the Monte-Carlo localization (MCL) approach (Dellaert et al., 1998). MCL estimates the distribution of the robot pose x_t in a map m at time step t given all previous measurements $z_{1:t}$ and actions $u_{1:t-1}$:

$$P(x_t \mid z_{1:t}, u_{1:t-1}, m) = \eta\, P(z_t \mid x_t, m) \int P(x_t \mid u_{t-1}, x_{t-1})\, P(x_{t-1} \mid z_{1:t-1}, u_{1:t-2}, m)\, dx_{t-1} \tag{4.10}$$

Here, the term $P(x_t \mid u_{t-1}, x_{t-1})$ is referred to as the motion model and $P(z_t \mid x_t, m)$ as the sensor model while η is a normalizer which ensures that $P(x_t \mid z_{1:t}, u_{1:t-1}, m)$ sums up to one. For all quantities, we use the standard models described in (Thrun et al., 2005).

Our method to generate inter-graph constraints is summarized in Algorithm 3. The algorithm is initialized by choosing a reference graph $G_{ref} = (V_{ref}, E_{ref})$ randomly among the set of pose graphs. The reference graph serves as a common reference frame in which constraints can be expressed. Global localization is performed in the map computed from G_{ref} and the sensor observations associated with the remaining graphs.

The localization approach is implemented using a particle filter. To start the global localization, its particles are initialized uniformly in the freespace of the reference map.

Algorithm 3 Constraint Generation Using MCL

Input: $\{G_1, \cdots, G_n\}$, independent graphs
Output: G, connected graph

```
1: ref ← random(1, ··· , n)
2: m_ref ← mapWithKnownPoses(G_ref)
3: for all G_i ∈ {G_1, ··· , G_n}\{G_ref} do
4:    C_i ← ∅
5:    for k = 1 to |G_i| do
6:       // check for convergence
7:       if x ← MCL(m_ref, z_{k:k+d}) then
8:          k ← k + d
9:          l̂ ← computeMeasurementLikelihood(z_d)
10:         if l̂ > l^min then
11:            q ← findNearestNeighbor(G_ref, x)
12:            c ← computeConstraint(q, p_k)
13:            C_i = C_i ∪ {c}
14:         end if
15:      end if
16:   end for
17:   C_i ← RANSAC(C_i)
18: end for
19: G ← mergeGraphs({G_1, ··· , G_n}, {C_1, ··· , C_n})
```

The algorithm then replays a sequence of d observations stored in graph G_i and updates the particle filter accordingly. The sequence should be long enough to allow the particle filter to converge, in our experiments we used $d = 40$, but this value certainly depends on the sensor model and on the density of the graph. After each localization trial, we check whether the filter has converged to an uni-modal distribution. This is achieved by verifying whether 99% of the probability mass is concentrated in a small predefined area, in our implementation we set this area to a circle with a radius of 0.5 m.

When the global localization step converges, we obtain a pose estimate x in the reference map m_{ref}. This allows us to compute a constraint between the poses p_k in G_i and pose x but x will in general not be equivalent to a pose $q \in G_{ref}$ in the pose graph G_{ref}. For this reason, we search for the pose $q \in G_{ref}$ with the smallest distance to the pose estimate x computed via MCL:

$$q = \underset{p \in V_{ref}}{\operatorname{argmin}} |x - p|. \tag{4.11}$$

After determining q, we compute a constraint between the graphs G_{ref} and G_i as the

relative rigid body transform between both poses

$$c = q\, p_k^{-1}. \tag{4.12}$$

In principle, one could now add the inter-graph constraints to the pose graphs and connect the independent graphs. However, the procedure we just described is likely to generate outliers since structurally similar areas may exist in different parts of the building. Consider, for example, two corridors with an identical layout that are located on the same floor of a building. When two robots map this building and each of the robots maps one of the two corridors, a false constraint between both corridors may be generated. Adding such false constraints to the SLAM optimization procedure that computes the global map is likely to lead to a catastrophic failure (Roberts et al., 2011). In contrast to that, it is typically not critical to omit a correctly identified constraint. Therefore, we perform two tests to reduce the risk of adding wrong constraints.

The first test considers the sensor measurement likelihoods obtained when evaluating the sensor model. Let us assume that the Monte Carlo localization approach uses N particles to estimate the pose distribution given in Equation (4.11). When the filter converges to a pose estimate in Algorithm 3, we compute the average observation likelihood $\hat{l}$ of measurement z_d over all N particles as

$$\hat{l} = \frac{1}{N} \sum_{j=1}^{N} P(z_d \mid x^{(j)}, m_{ref}), \tag{4.13}$$

where $x^{(j)}$ denotes the pose hypothesis of particle j.

High values of $\hat{l}$ indicate that the measurement can be explained well by the map m_{ref} while low values indicate that the particles may have converged to a position that does not correspond to a likely pose in the map. Thus, we use a heuristic threshold criterion and accept the corresponding constraint as an alignment hypothesis if $\hat{l} > l^{min}$, otherwise we reject it as an outlier. The threshold l^{min} can be learned by evaluating the sensor model for localization while building a grid map from the current pose graph G_i. If $\hat{l}$ exceeds the threshold, the alignment hypothesis c is added to a candidate set $\mathcal{C}_i$. Then, the filter is reset and the procedure is restarted.

While a correct localization will result in high observation likelihoods, the converse is not true in all cases. Pose estimates that lead to a high observation likelihood do not necessarily correspond to a correct association of map positions. Just by chance, two structurally similar areas may exist in the same building and a false constraint would possibly be generated between them. To remove such outliers from the set of alignment hypotheses, we apply the random sample consensus (RANSAC) algorithm (Fischler and Bolles, 1981) on the candidate sets $\mathcal{C}_i$. RANSAC is an iterative algorithm that estimates parameters of a model from a set of observed data that contains outliers. In our system,

the model corresponds to an alignment of pose graphs which can be described by a planar translation and rotation around the vertical axis. Using the RANSAC notation, the alignment hypotheses in C_i serve as observations of this alignment. RANSAC will then determine the maximum set of consistent inter-graph constraints.

Once a consistent set of inter-graph constraints is found, we generate a joint SLAM graph from the individual SLAM graphs and the constraints. This joint graph serves as input to the SLAM back-end which then optimizes the overall model. Note that the minimal amount of inter-graph constraints that is necessary to correctly map the environment depends on the input data. Under the assumption that the individual graphs are consistent, a single constraint would be sufficient to align two graphs and RANSAC would not have to be used. In practice, however, measurement noise and odometry errors lead to erroneous local constraints in SLAM graphs. For this reason, our current implementation requires at least three constraints to be found for each graph.

Inter-graph constraints do not only align individual pose graphs but they also mitigate estimation errors in graphs and lead to a globally more consistent model. Intuitively, when the same area is mapped correctly in one graph but erroneous in another, inter-graph constraints will improve the overall model. We will evaluate this effect in our experiments in Section 4.5.5.

4.4. Front-End for Multi-Robot SLAM based on Landmark Maps

In the previous section, we presented a multi-robot SLAM front-end based on grid maps. In this section, we present an algorithm that is based on landmark maps. We define a landmark as a physical object that can be detected using a sensor on a mobile robot. We assume that the sensor provides range and bearing measurements relative to the current pose of the robot and that a planar 2D representation is sufficient to map the landmarks. Real-world examples of such features include tree trunks, lamp posts, doors and windows, corners of walls, road signs – in short a landmark can be any object that is detected reliably and that describes a position in the environment.

The single robot SLAM front-end in the landmark-based approach is similar to the one presented in Section 4.3. We generate nodes in the pose graph from the trajectory of the robot. Range and bearing measurements of landmarks are stored in measurement vectors z_i associated with the poses p_i. Whenever the same landmark is detected from two different poses, a constraint is computed between those poses and added to the graph as an edge. This step involves a data association problem: We need to identify previously measured landmarks. When landmarks can be uniquely identified, for example when they carry unique markers, this problem can be solved trivially. In our approach, we do not require unique identifiers but we assume that a good initial guess of the position of

a landmark can be computed from the pose graph and that it is sufficient to search for matching landmarks in a neighborhood around the initial pose. This assumption holds in many realistic scenarios, such as our real-world experiments presented in Section 4.6.

4.4.1. Triangle Map Matching

In a distributed system, one pose graph with landmark measurements is created for every robot. To connect multiple SLAM graphs, we associate landmarks detections stored in one graph with landmark measurements in another graph. We assume that the relative transform between the graphs is initially unknown, therefore we cannot associate landmarks based on pose information alone. Instead, we generate landmark maps from each graph and analyze the relative position of the landmarks in the maps. To this end, we compute a triangulation of landmarks and then identify matching triangles based on features that are invariant to Euclidean transformations.

In general, a unique transformation between two planar maps m_i and m_j can be computed from two point correspondences (Booij and Zivkovic, 2009). A naive approach would therefore randomly choose, without replacement, two landmarks from m_i and two landmarks from m_j, compute the corresponding transformation and verify it, for example, by counting the number of matched landmarks after applying the transformation. Unfortunately, this approach is intractable even for moderately sized landmark maps. If we assume that both maps consist of N landmarks, then there are N^2 pairs of landmarks in each map and a total of N^4 possible matchings to verify.

To avoid the combinatorial complexity associated with the naive solution, we compute a set of geometric features from the landmark maps. These features allow for the efficient generation of correspondences by guiding the search for matches. Instead of matching landmark maps directly, our approach computes a Delaunay triangulation of the landmark positions in a map. Given a set of points in the plane, this algorithm determines a triangulation such that no point in the set is inside the circumcircle of any triangle. The Delaunay triangulation is known to be unique if the points are in *general position*, that is, no more than three points lie on a circle. The triangulation can be computed in $O(n \log n)$ (Lee and Schachter, 1980).

Our approach is summarized in Algorithm 4. In general, the algorithm matches every pair of two pose graphs, but for the sake of clarity, let us assume in the following description that there are exactly two graphs G_a and G_b. We first compute their landmark maps and then obtain two sets of triangles T_a and T_b using the Delaunay triangulation. A set of features is computed for each triangle to guide the search for corresponding triangles. Since we want to recover an Euclidean transformation between both maps, these features need to be invariant to Euclidean motion. We compute the sum of triangle edge lengths and the area of the triangles. For each triangle t_j with edge lengths a_j, b_j, c_j we compute

Algorithm 4 Constraint Generation Using Triangulation

Input: $\{G_1, \cdots, G_n\}$, independent graphs
Output: G, connected graph

// pre-processing step
for all $G_i \in \{G_1, \cdots, G_n\}$ **do**
  $m_i \leftarrow \text{computeLandmarkMap}(G_i)$
  $T_i = \{t_{i,j}\} \leftarrow \text{DelaunayTriangulation}(m_i)$
  // compute triangle features
  for all triangles $t_{i,j} \in T_i$ **do**
    // $a_{i,j}, b_{i,j}, c_{i,j}$ are the edge lengths of $t_{i,j}$
    $f^1_{i,j} \leftarrow a_{i,j} + b_{i,j} + c_{i,j}$
    $f^2_{i,j} \leftarrow \sqrt{s(s-a_{i,j})(s-b_{i,j})(s-c_{i,j})},\ s = \frac{1}{2} f^1_{i,j}$
  end for
end for
// graph matching step
for all map pairs $(T_i, T_j),\ i, j \in \{1, \cdots, n\},\ i > j$ **do**
  for all triangle pairs $(t_{i,k}, t_{j,l}),\ t_{i,k} \in T_i,\ t_{j,l} \in T_j$ **do**
    $s = \prod_{m \in \{1,2\}} \exp((f^m_{i,k} - f^m_{j,l})^2)$
    if $s < \tau$ **then**
      $c \leftarrow \text{computeConstraint}(t_{i,k}, t_{j,l})$
      $\mathcal{C}_{i,j} = \mathcal{C}_{i,j} \cup \{c\}$
    end if
  end for
  $\mathcal{C}_{i,j} \leftarrow \text{RANSAC}(\mathcal{C}_{i,j})$
end for
$G \leftarrow \text{mergeGraphs}(\{G_1, \cdots, G_n\}, \{\mathcal{C}_{i,j}\})$

those features as:

$$f^1_j = a_j + b_j + c_j \tag{4.14}$$

$$f^2_j = \sqrt{s(s-a_j)(s-b_j)(s-c_j)},\ s = \frac{1}{2} f^1_j \tag{4.15}$$

Both features are invariant to translation and rotation. The set of correspondences $\mathcal{C}_{a,b}$ between T_a and T_b is defined as

$$\mathcal{C}_{a,b} = \{(t_a, t_b) \mid t_a \in T_a, t_b \in T_b, s(t_a, t_b) < \tau\}, \tag{4.16}$$

where τ is a threshold on the (dis)similarity of two triangles that is given by

$$S(t_a, t_b) = \prod_i \exp((f^i_a - f^i_b)^2). \tag{4.17}$$

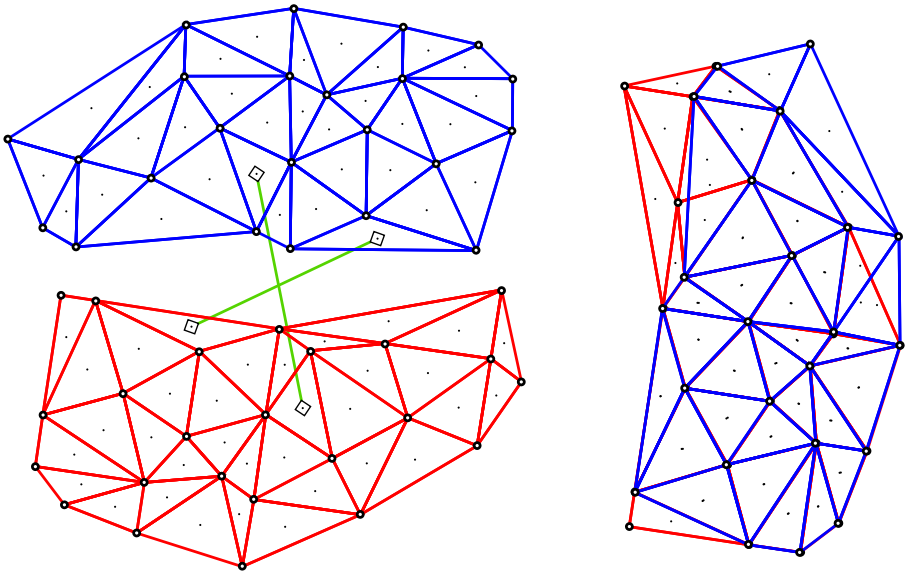


Figure 4.2: Matching of landmark maps. Left: two sets of landmarks (white dots) are visualized. Their Delaunay triangulations are shown as blue and red lines while the triangle centers are depicted by black dots. Matching constraints between two of the triangles, shown in green, define an alignment of the landmark maps. Right: correctly aligned maps.

Since the ordering of the triangle vertices is unknown in general, we use only the center points of the triangles to compute the transformation and hence need to find at least two triangle correspondences. An illustration of the matching process can be seen in Figure 4.2.

The set of correspondences $\mathcal{C}_{a,b}$ will in general contain outliers since the geometric features described above allow for some ambiguities. For this reason, we apply RANSAC on $\mathcal{C}_{a,b}$ to eliminate outliers. RANSAC is an iterative algorithm that estimates parameters of a model from a set of correspondences. In our system, the model corresponds to the transformation $t_{a,b}$ between the graphs G_a and G_b which can be described by a planar translation and rotation. Following the standard RANSAC algorithm, we determine $t_{a,b}$ by repeatedly sampling two correspondences from $\mathcal{C}_{a,b}$ and returning the model that maximizes the number of matched landmarks.

The complexity of the matching algorithm is dominated by the triangulation, the search for matching triangles, and the RANSAC verification step. Let n be the number of landmarks in a map, for simplicity we assume that every map contains an identical number of landmarks. As stated above, the Delaunay triangulation has a complexity of $O(n \log n)$. It can be shown, that any triangulation of n points results in less than $2(n-1)$ triangles, so the number of triangles can be asymptotically approximated by n. To align n triangles of one map to n triangles in another map, we search for matching features in Equation (4.16). The naive implementation given in Algorithm 4 takes $O(n^2)$ comparisons. In practice, however, the search can be implemented using efficient data structures such as

hash tables or search trees resulting in an asymptotic average complexity of $O(n)$ for hash tables and $O(n \log n)$ in the case of search trees. The complexity of RANSAC depends on the cost of verifying a model and the number of iterations. To verify a transformation between two maps, we need to transform every landmark position in one map and count the number of matched landmarks in the other map. This step therefore has a computational complexity of $O(n)$. The number of RANSAC iterations can be determined based on the outlier probability and is a constant small value. The overall average asymptotic complexity of matching two maps is therefore $O(n \log n)$.

4.5. Evaluation of Grid-Based SLAM Front-End

We evaluated both approaches presented in the previous sections using several real world datasets as well as simulated data. The experiments are designed to show that the proposed approaches are able to generate inter-floor constraints and in this way enable consistent mapping in a distributed system.

The grid map based approach was evaluated in the context of multi-floor building mapping. Here, one or more robots map each floor of a building independently and the relative transformation between individual floor maps is assumed to be unknown. We assume that individual floors of a building show structural similarities. Most real-world buildings meet this requirement since floors share structural elements such as stairways, elevators, or corridors. We apply the data association technique we introduced in Section 4.3 to align maps of several floors. Our method generates constraints between the floors and these constraints are then used to estimate a consistent model of the complete building. Furthermore, we also evaluated whether the overall mapping error is reduced and to what extent mapping errors in one of the floors can be corrected using the maps of other floors. All real world datasets were recorded using a Pioneer2 robot equipped with a SICK LMS 291 laser range finder.

4.5.1. Mapping a Typical Office Building

In this experiment, we mapped an office building that consists of four floors with a similar layout. The dataset was recorded in building 106 on the computer science campus in Freiburg which is depicted in Figure 4.3. The floors differ mainly in the area around the staircase (left side of the floor maps) but are virtually identical in the area around the elevator (right side of the maps). Smaller differences are introduced by open doors and furniture.

We applied Algorithm 3 to generate constraints between the four initially unconnected pose graphs. The joint building graph was then optimized using the SLAM back-end presented in Section 4.2.1. From the corrected SLAM graph, we generated grid maps of

Figure 4.3: Building 106 on the Freiburg computer science campus.

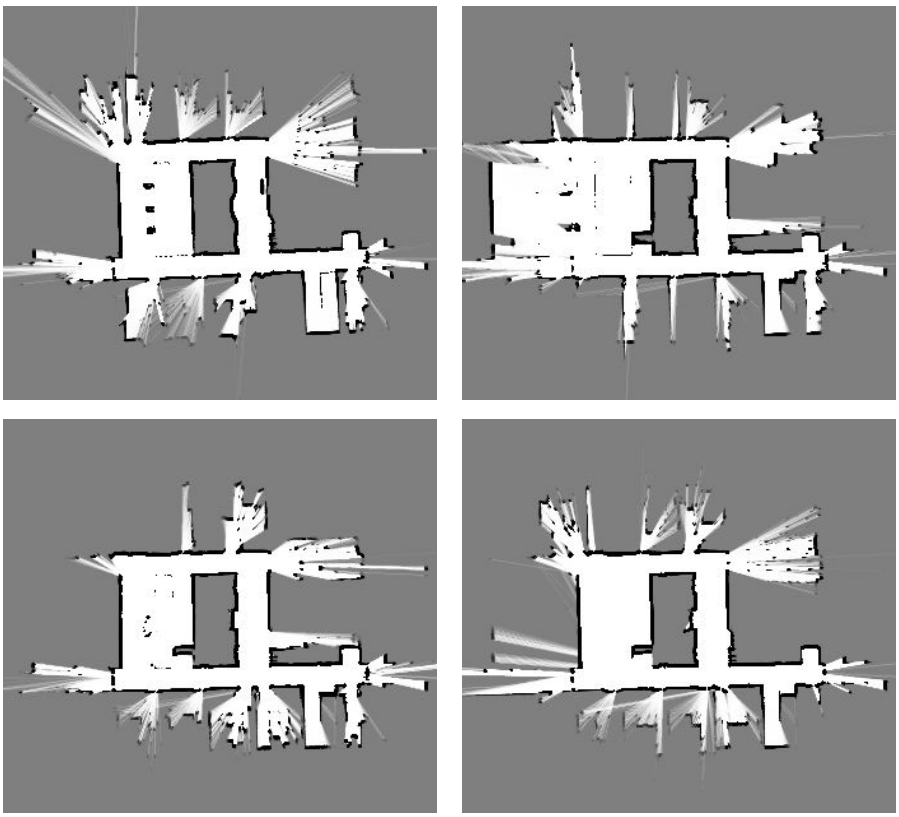


Figure 4.4: Grid maps of each floor of building 106. The maps were aligned based on inter-graph constraints that were computed by our approach.

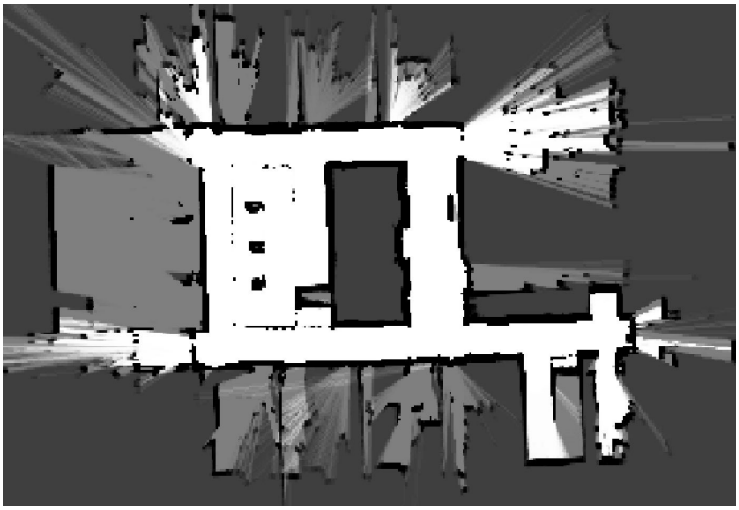


Figure 4.5: Overlay of the four floor maps of building 106 after alignment.

each floor, the result can be seen in Figure 4.4. By visual inspection of the overlay of all four maps given in Figure 4.5 there is no apparent alignment error.

4.5.2. Mapping Long Corridors

A more challenging dataset was recorded in building 051 on the computer science campus in Freiburg. Each of the three floors of this building essentially consists of two long corridors meeting at an elevator in the middle of the building. With all office doors closed the elevator area and a staircase on either end of the building are the main structural features. Using the proposed approach, constraints were found in those salient areas and a correct alignment was obtained. In Figure 4.6 two of the three aligned floors are depicted in a three-dimensional illustration.

4.5.3. Building with Few Structural Similarities

To explore the limits of our approach, we mapped a building whose floors have very few visible similarities. Building 101 on the computer science campus in Freiburg features a modern architecture with large glass constructions, see Figure 4.7. Its three floors do not share a lot of structural features, the only exception being an elevator shaft and a short corridor leading to the restrooms.

The MCL-based approach successfully generated inter-floor constraints in the similar areas around the elevator but failed to generate further constraints in other parts of the building. For this reason, the alignment achieved for this dataset is not as good as in the previous experiments. A small but noticeable rotational error remains and can be seen in the overlay of two of the resulting floor maps depicted in Figure 4.8.

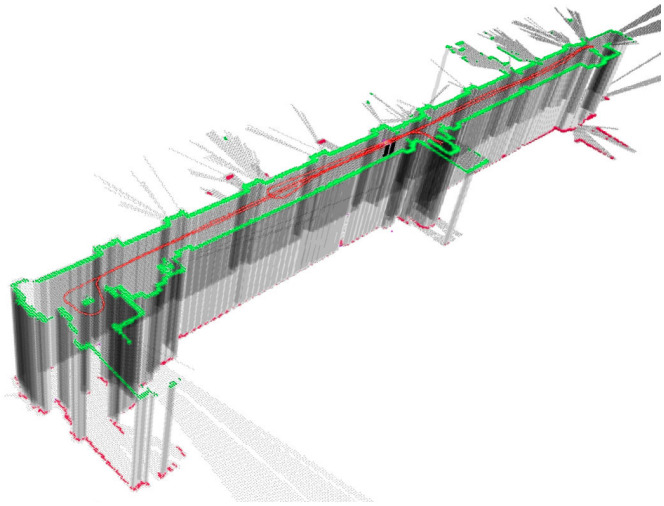


Figure 4.6: 3D visualization of two correctly aligned floors of building 051. To illustrate the alignment, occupied cells found in both floor maps have been connected by lines.

4.5.4. Quantitative Evaluation Using a Simulated Building with Ten Floors

To quantitatively evaluate the alignment obtained using our approach, we created a virtual dataset consisting of ten floors taken from a real world dataset. To this end, we extended the dataset of building 106 recorded in the first experiment. The top three floors were duplicated twice while the first floor was used once only. A Gaussian rotational error was added to each floor graph to simulate pose uncertainty across floors. The standard deviation of 0.2 rad relates to the rotational error observed in experiments in which a robot entered an elevator, turned on the spot, and left the elevator again. The graphs of the individual floors without connecting constraints as well as the merged and corrected graph generated by our approach are shown in Figure 4.9.

To analyze the alignment error, we manually selected the first floor as the reference floor and aligned each of the following floor graphs against it using our approach. By choosing the reference floor in this way, an unrealistic overfitting is avoided since matchings between virtual, duplicated, floors are not considered. The graph optimization was then carried out with and without considering the inter-floor constraints generated by our system. We manually determined the correct alignment of the floors and computed the deviation of this to the alignment obtained by the SLAM approaches. The experiment was repeated nine times.

Figure Figure 4.10 depicts a statistical analysis of the resulting alignment error. From the average error and the confidence intervals it can be seen that using inter-floor constraints significantly reduces the alignment error of the floors and therefore improves the overall model of the building.

Figure 4.7: Building 101 on the Freiburg computer science campus.

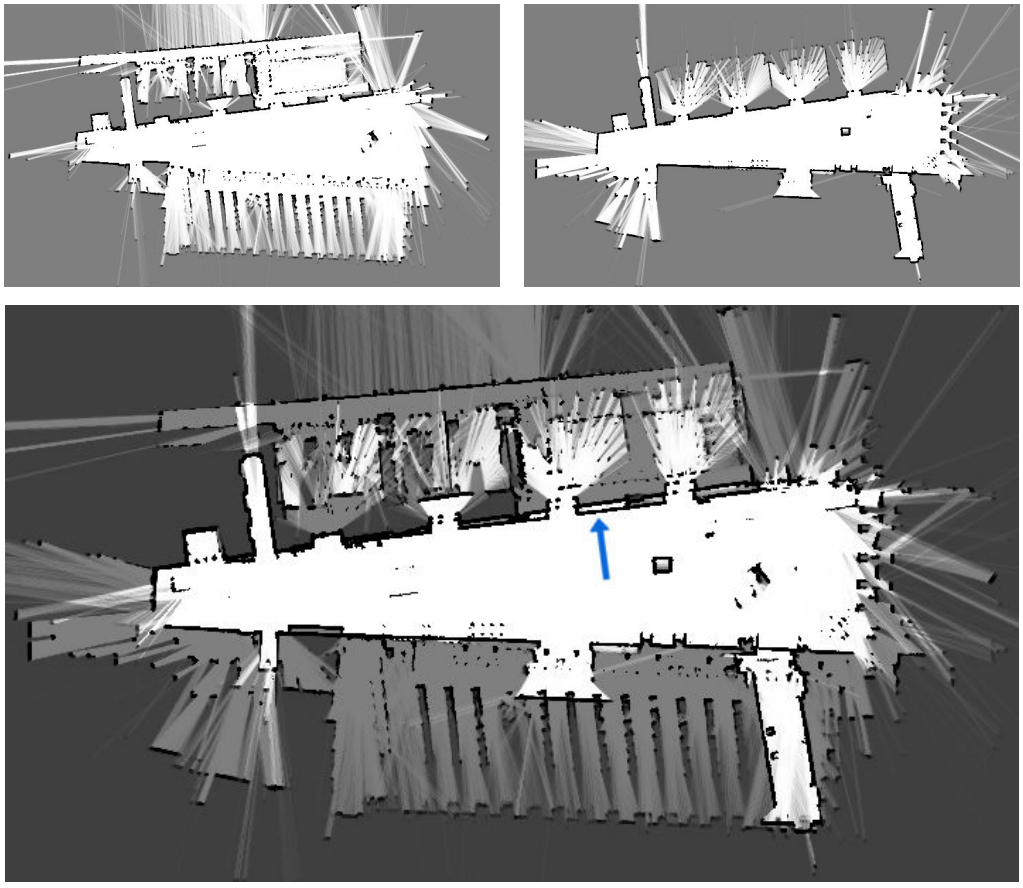


Figure 4.8: Result of our SLAM approach applied to the building 101 dataset. Top: two floor maps of building 101 with few similarities. Bottom: overlay of the floors with alignment found by our approach. The blue arrow indicates a minor inconsistency.

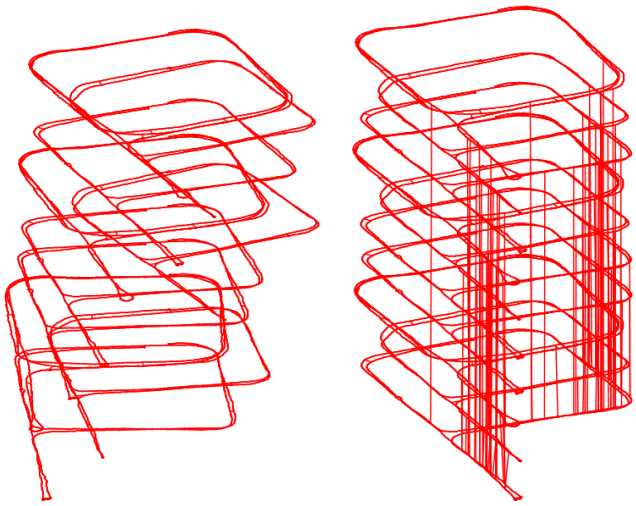


Figure 4.9: SLAM graphs of a simulated building with ten floors. Left: Floor graphs before alignment. Right: A merged graph has been computed and each of the floor graphs has been correctly aligned against the reference floor.

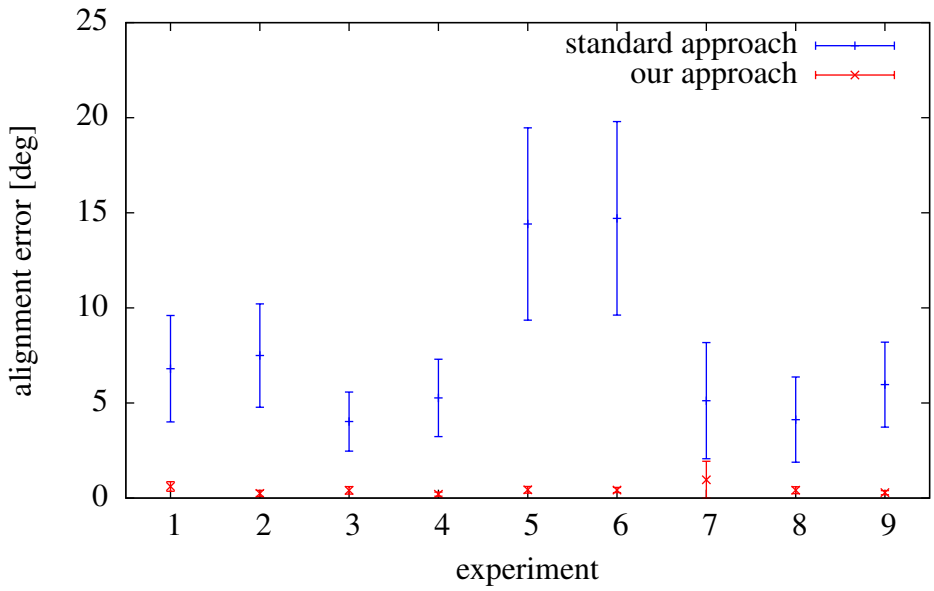


Figure 4.10: Average alignment error with confidence intervals according to a t-test with 95% confidence computed for 9 different experiments aligning 10 floors each.

4.5.5. Correction of Systematic Mapping Errors

In this experiment, we evaluate to which extent our approach is able to correct systematic sensor errors in one of the pose graphs based on information in the remaining graphs. Such errors may, for example, be induced by lighting conditions in the case of vision sensors or by long, featureless corridors in the case of laser range finders. To evoke such an error, we simulated an environment based on the map of building 051 that consists of two different corridors with a length of 130 m as depicted in Figure 4.11. Floor A features a similar geometrical structure as the original building but was extended by copying parts of the corridors. Floor B is identical to floor A with the exception that it does not contain any significant structure within the corridors. However, both floors still share the unmodified elevator area and the two staircases that are present in the original map.

The standard SLAM front-end was able to generate a consistent map of floor A. The featureless corridor of floor B, however, posed a problem to the scan matcher. The maximum measurement range of 30 m was not sufficient to measure structurally salient regions or the limiting walls at the ends of the corridor at all times. For this reason, the corridor of floor B was shortened by the scan matching procedure leading to a shortened corridor in the resulting map, see Figure 4.11 c. This is a typical effect of short range proximity sensors (Stachniss et al., 2008). We measured the error in the corridor length over ten runs of the experiment. On average, the shortening amounted to 14.2% of the original length, corresponding to a length of 18.5 m.

Our MCL-based approach was able to find inter-graph constraints at the structurally identical staircases at both ends of the simulated floors. Using those constraints and the floor graphs of floor A and floor B, our approach was able to generate a merged graph of both floors. The resulting map of floor B can be seen in Figure 4.11 d. The average residual error in length was reduced to 5.8%, corresponding to 7.5 m. This improvement is significant according to a t-test performed with a confidence level of 5%.

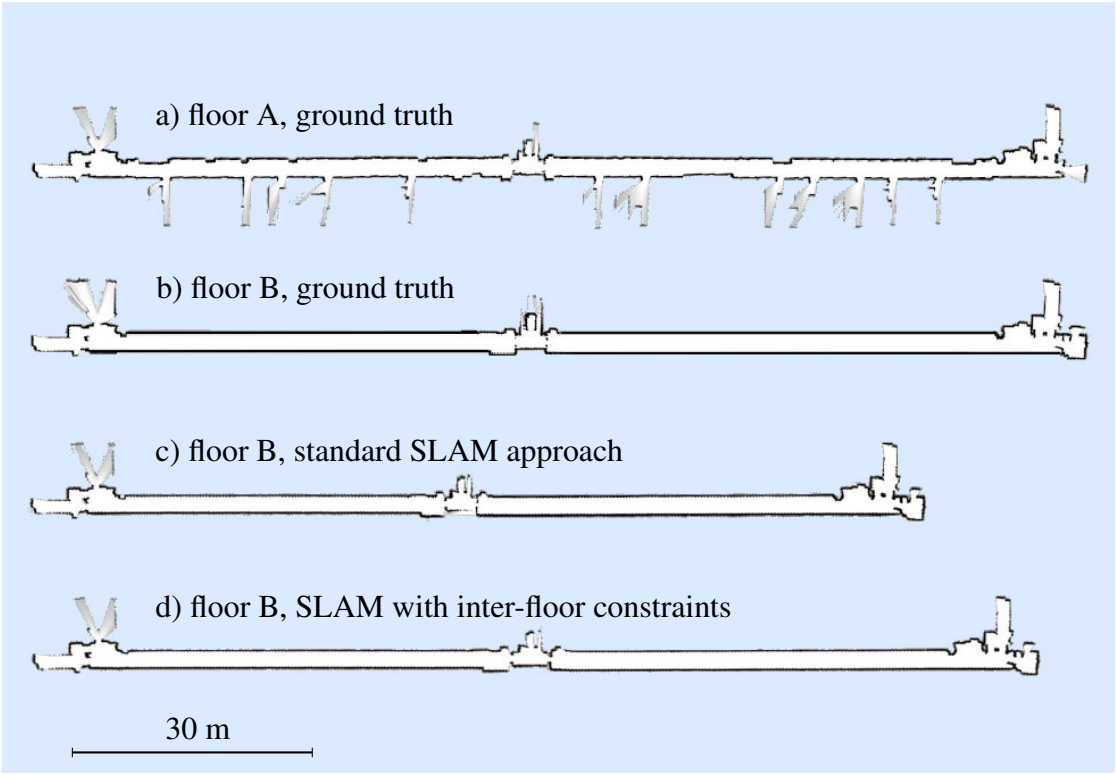


Figure 4.11: Simulated building with long corridors. a: Simulated floor A, b: simulated floor B with few structural features, c: floor B mapped using the standard approach without inter-floor constraints. There is an average error in length of 14.2%. d: Floor B after applying our approach. The average remaining error in length is reduced to 5.8%.

4.6. Evaluation of Landmark-Based SLAM Front-End

We evaluated our landmark-based multi-robot approach using a large simulated dataset and a dataset created by three real robots. The experiments were designed to demonstrate that the approach is able to create constraints between multiple pose graphs in realistic scenarios. We furthermore evaluated to what extent the map matching is influenced by the overlap of the individual maps.

4.6.1. Real World Experiments

To evaluate our approach under realistic conditions, we performed experiments using a team of three real robots that can be seen in Figure 4.12. The team consisted of an ActiveMedia Pioneer2, Pioneer2 AT and a PowerBot, each equipped with a SICK LMS 291 laser range finder. The experiment was conducted in the parking lot of the computer science campus in Freiburg. The robots were manually steered through the environment. Feature measurements, consisting of distances and bearings, came from a pole detector and were filtered to remove false detections, such as the legs of the operators. Since the parking lot does not contain a sufficient amount of detectable features, a number of artificial landmarks (poles) have been placed there additionally. We used poles with a diameter of 15 cm and a height of about 100 cm. We conducted two runs in which the robots were steered for about 20 min. An overview of the parking lot and the landmarks as well as the trajectories of the robots during the first run is given in Figure 4.13. In the figure, the locally consistent trajectories returned by single-robot SLAM were manually aligned using our background knowledge about the setting.

To estimate a consistent model that considers all robot trajectories, we applied the DDF-SAM approach by Cunningham *et al.* (2010) to perform local SLAM and graph optimization. To optimize the individual pose graphs, this approach uses an improved version of iSAM (Kaess et al., 2008) that enables real-time performance (Kaess et al., 2010). The global pose graph was optimized using the Georgia Tech Smoothing and Mapping library (Georgia Tech Research Corporation, 2010). Constraints between pose graphs were generated using the triangulation-based map matching approach presented in Section 4.4 where the triangulation was computed using the Triangle library (Shewchuk, 1996).

The result of the real-world experiments are shown in Figure 4.14. Depicted are the merged maps of both runs in a globally consistent frame. No ground-truth information was available in the experiments, but from visual inspection, the global landmark positions are consistent with the landmarks in the environment. The alignment computed by our approach cannot be differentiated from the manual alignment shown in Figure 4.13.

Figure 4.12: Robots and one of the landmarks used in the experiments

4.6.2. Simulated Victoria Park

To quantitatively evaluate our landmark-based map matching approach, we simulated a large outdoor environment based on the popular Sydney Victoria Park dataset recorded by Guivant (2001). This dataset consists of laser-range data and vehicle odometry, recorded in a park with sparse tree coverage and spans across an area of roughly 300 m × 300 m. To evaluate our multi-robot SLAM approach, we recorded simulated trajectories and corresponding laser range measurements in a simulation of this environment. The simulation is based on the map estimate of Kaess *et al.* (2008) that consists of the positions of 140 individual tree trunks. From the trunk positions we generated a grid map with a resolution of 0.1 m and simulated a robot that traversed this map using the CARMEN Robot Navigation Toolkit (Montemerlo et al., 2002). Twenty trajectories were recorded starting from different positions in the map and then traversing the environment in a round trip. Sensor and odometry noise was not considered in the simulation, the only source of error was a slight synchronization offset of trajectory and sensor readings. In this way, we were able to evaluate our map matching technique independently from influences of single-robot SLAM. An overview of the simulated data is shown in Figure 4.15.

We considered all unique pairs of the 20 trajectories, so a total of 190 pairs were evaluated. Each experiment was repeated three times since the RANSAC approach includes random sampling. From the simulated laser measurements, we extracted range and bearing landmark detections and we estimated the position of landmarks using Kalman filtering. For each pair of trajectories, we generated landmark maps in intervals of 5 s simulated time. We then applied our landmark-based map matching approach to compute a relative transformation between the maps. In those cases where one trajectory was longer than the other, the final map of the shorter trajectory was used multiple times.

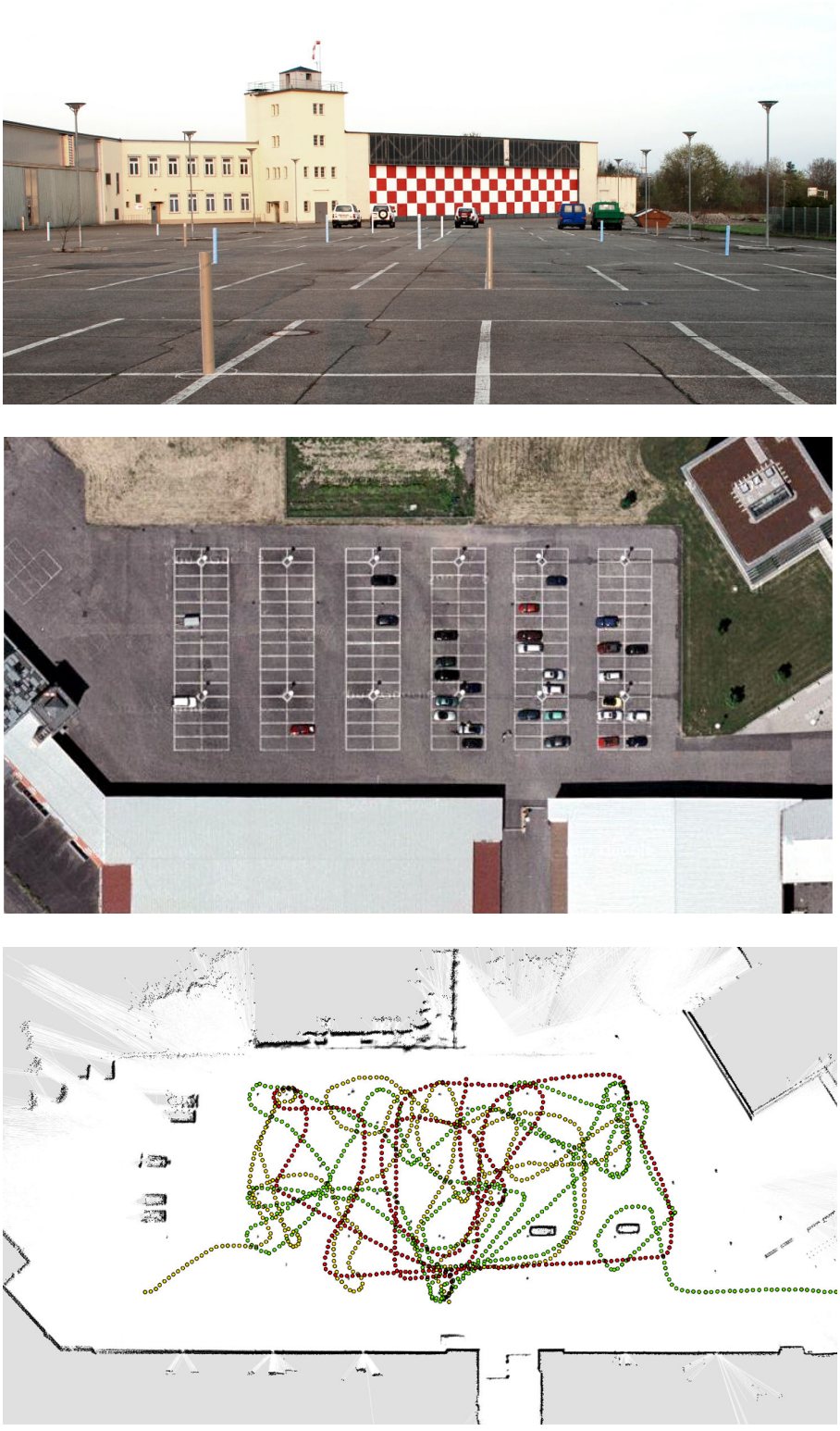


Figure 4.13: Parking lot with landmarks (top), an aerial view of the parking lot (middle), and grid map of the environment with overlaid trajectories (bottom left). Aerial image courtesy of Google Maps, Copyright 2008, DigitalGlobe.

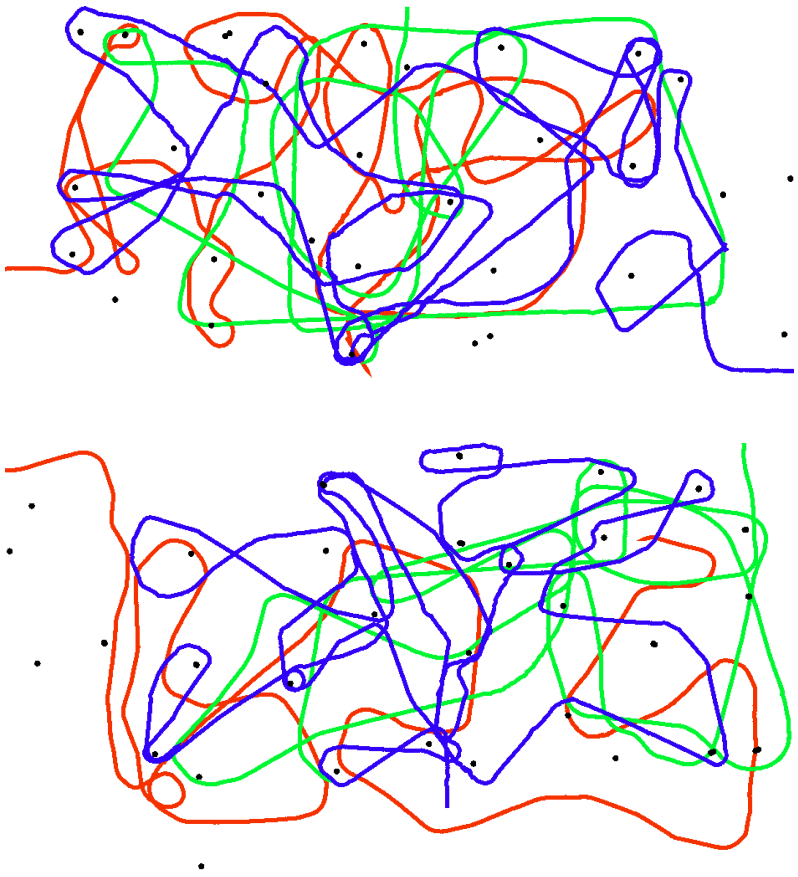


Figure 4.14: Result of parking lot experiment. Shown are the optimized trajectories of the three robots (colored red, green, and blue) and the landmark positions (black). The upper figure shows the results of the first run while the lower figure visualizes the result of the second run.

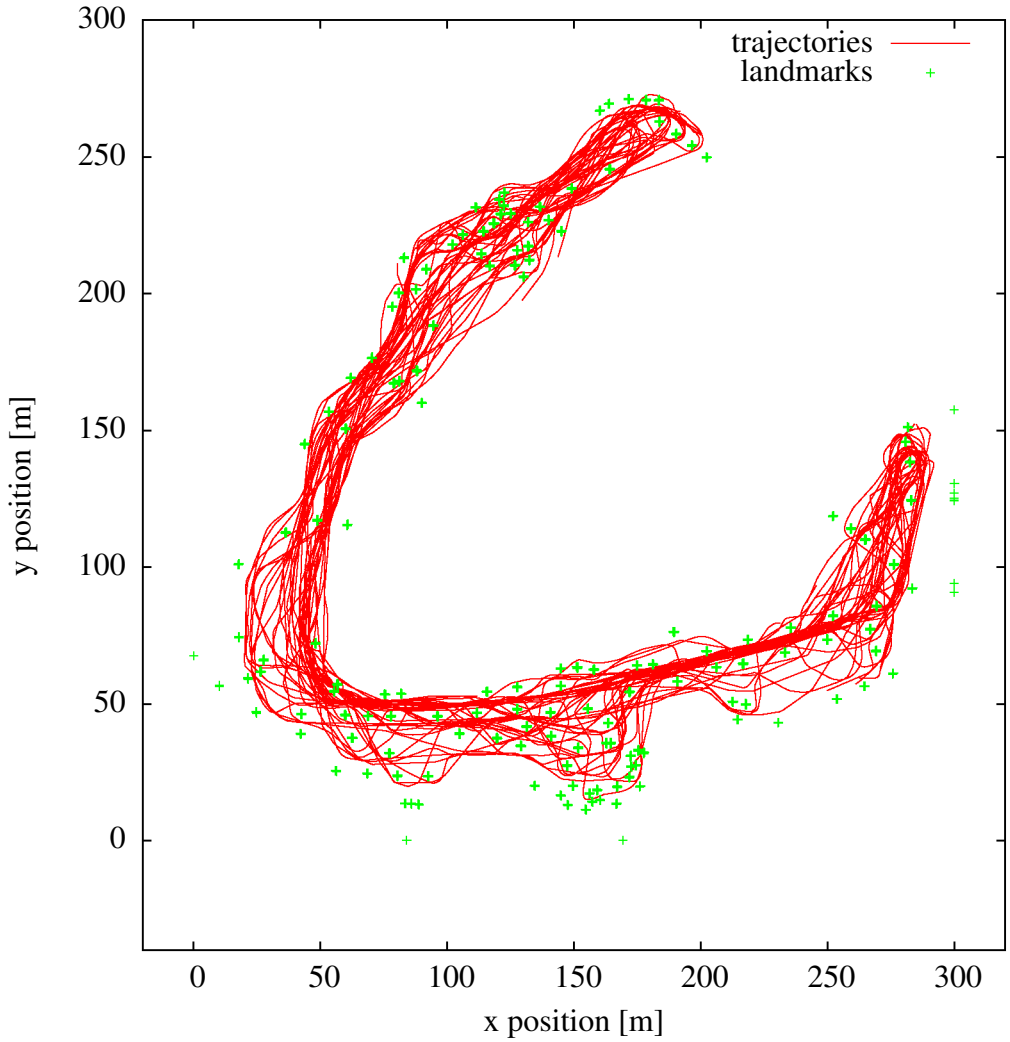


Figure 4.15: Simulated trajectories in the Victoria Park landmark dataset. 20 different robot trajectories (red) and corresponding landmark measurements (green) are shown.

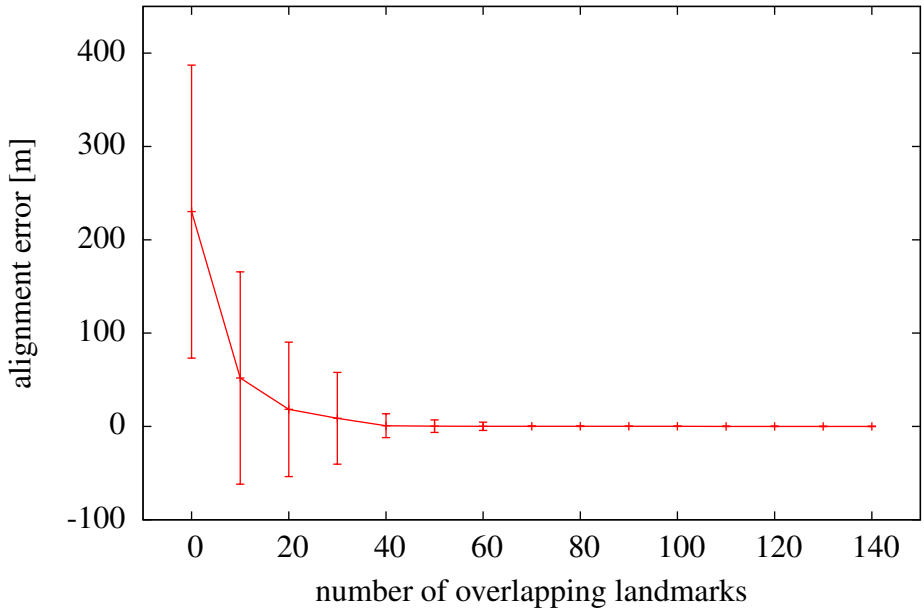


Figure 4.16: Average translational error of pairwise map alignments. Error bars correspond to the standard deviation.

As a measure of matching quality, we determined the error in distance and in rotation. Furthermore, we computed the overlap of the maps in terms of the absolute number of overlapping landmarks. The overlap was determined based on the absolute position of landmark measurements that were available in the simulation.

A total of 100.608 matching trials were performed. Of these trials, 83.828 pairs of maps generated the minimum number of two consistent triangle correspondences and were considered in the following evaluation. The results are given in Figure 4.16 and Figure 4.17. The plots show the matching error in relation to the map overlap. For the sake of clarity, we computed the mean error and standard deviation for ranges of overlap values. It can be seen, that our approach was able to correctly align maps when more than 50 landmarks were shared between both maps, which corresponds to less than half the total number of landmarks. In fact, the average translational error for an overlap of 50 to 59 landmarks is 0.3 m which is the width of three grid cells in the simulated map, or 0.1% of the map width. When all 140 landmarks are used to compute the alignment, the average error drops to 0.04 m. The rotational component of the alignment can be determined up to an average error of 0.04 rad with as few as 30 to 39 overlapping landmarks and is practically error free when the overlap is bigger than 50 landmarks.

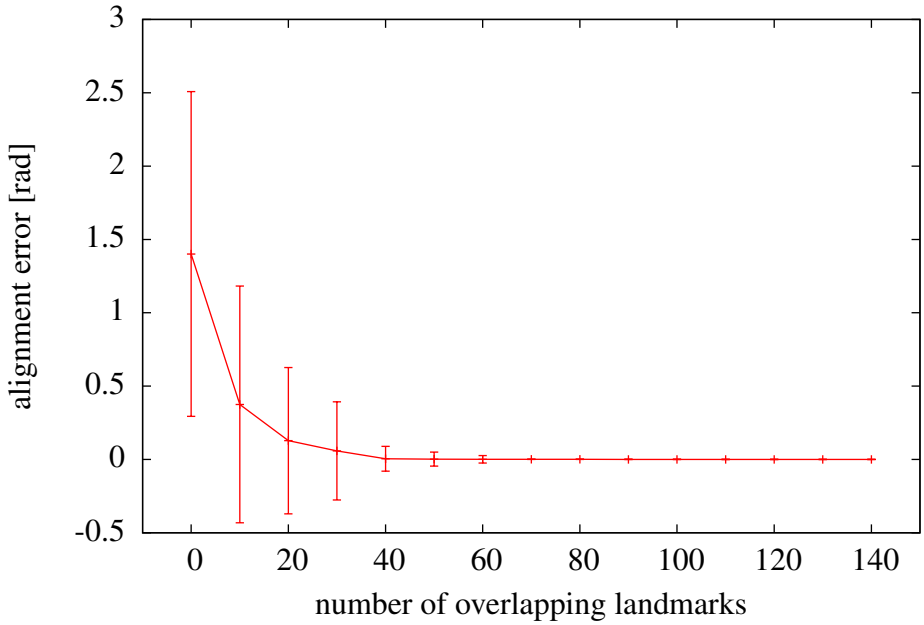


Figure 4.17: Average rotational error of pairwise map alignments. Error bars correspond to the standard deviation.

4.7. Related Work

A large variety of SLAM approaches is available to the robotics community. Common techniques apply extended and unscented Kalman filters (Julier et al., 1995; Leonard and Durrant-Whyte, 1991), sparse extended information filters (Eustice et al., 2005), particle filters (Montemerlo and Thrun, 2003), and graph-based, least square error minimization approaches (Lu and Milios, 1997; Frese et al., 2005; Olson et al., 2006; Grisetti et al., 2007c).

The graph-based formulation of SLAM has been introduced by Lu and Milios (1997). In recent years, a variety of algorithms have been proposed to efficiently solve constraint networks in the context of SLAM. Frese *et al.* presented the treemap approach which employs multilevel relaxation (Frese, 2006) similar to Paskin's TJTF method (Paskin, 2003). A stochastic gradient descent method is proposed by Olson *et al.* (2006) which has been further extended by Grisetti *et al.* (2007a) using an efficient tree parameterization.

The problem of generating constraints between pose graphs is closely related to the problem of computing an alignment of several maps. To align grid maps, Bosse and Zlot (2008) proposed a map matching approach that is based on histograms. They maintain a graph of local grid maps and compute histograms by projecting the 2D maps onto three one-dimensional values. By correlating these histogram features, matches between submaps are identified. A similar approach was presented by Carpin (2008). In contrast to the approach by Bosse and Zlot, grid maps are interpreted as binary images and then aligned by computing their Hough spectra. These spectra express the directions of

lines in the binary image and are used to determine a relative matching orientation. The displacement is then computed using projection histograms.

The alignment of landmark maps was first investigated by Dedeoglu and Sukhatme (2000). Their approach assumes that landmarks carry position and heading information. They evaluate all possible pairs of landmarks within one semantic class and in cases where all landmarks fall within the same class, every pair of landmarks has to be evaluated. Huang and Beevers (2005) extended this approach by considering the structure a graph-based landmark map. Starting from landmark-to-landmark matches, they grow the match along edges in the map. The authors state that the runtime of their approach is $O(n^2 \log n)$.

Thrun and Liu (2005) determine triplets of adjacent landmarks and then use relative distances and angles within these triangles to find matches. Map transformation hypotheses are verified based on negative information and the likelihood of a merged map. The matching technique is not described in detail, the authors mention that hill-climbing is applied to select correspondences from the set of hypotheses and that a Bayesian MAP estimator is used to merge maps. The authors assume that the approach has an asymptotic complexity of $O(n \log n)$. This approach is similar to the landmark-based matching approach presented in this chapter. Our algorithm, however, is described in detail and we provide a complexity analysis that shows that our approach has a complexity of $O(n \log n)$. Furthermore, we apply the Delaunay triangulation that is unique when all landmarks are in general position. The Delaunay triangulation was previously used by Ogawa (1986) to match sets of points in images. The approach assumes that each point carries a label that assigns it to a class. In contrast to this, our approach is based solely on the position of landmarks in the plane and does not assume any further knowledge or labels.

When multiple robots explore the same environment and are able to communicate, one can assume that the robots meet and recognize each other. Several approaches have been proposed based on this assumption. Howard *et al.* (2005) present a graph-based SLAM approach that is able to merge the maps of several exploring robots once they meet and recognize each other. The approach presented by Zhou *et al.* (2006) relies on the stronger assumption that robots can compute a relative transform from robot-to-robot measurements. Landmark-based maps of the robots are then matched using a nearest neighbor search. A multi-robot SLAM approach was also presented by Ko *et al.* (2003). The relative positions of the robots are initially unknown and estimated by localizing each robot in the maps of the other robots. Hypotheses are actively verified by employing a rendezvous strategy. This approach is similar to the grid map based approach presented in this chapter in that it uses a localization method to generate constraints. We do not, however, assume the robots are able to communicate and mutually recognize each other.

So far, there exist only very few methods that explicitly model the relation between separate floors of a building. One such approach has been presented by Iocchi *et al.* (2007).

A set of separate 2D floor maps is used to model the environment. In contrast to the approach presented in this chapter, the floor maps need to be generated by a single robot which is able to derive the floor alignment by employing a precise visual odometry method. Consequently, this approach cannot be applied in a multi-robot scenario and will fail to provide a correct alignment if floor maps are not mapped consecutively or if the transition of floors is not visually salient. General graph optimization frameworks have also been successfully applied to datasets containing simulated multi-level buildings (Frese and Schroder, 2006; Grisetti et al., 2007a). Note, however, that in these datasets constraints between floors have not been generated from sensor measurements but have been generated from the simulated ground truth.

4.8. Discussion

In this section, we presented an approach to simultaneous localization and mapping (SLAM) with multiple robots. Most existing approaches focus on efficient solutions to SLAM with single robots. To extend these approaches to multi-robot systems we align the maps and trajectories of the individual robots. We make use of a graphical formulation of the SLAM problem, that maintains pose graphs for each robot. To connect these initially independent graphs, we solve a data association problem: Areas that have been covered by multiple robots are identified based on the observations of the robots. From these associations, we extract alignment constraints between the trajectories of the robots. The resulting global graph is then optimized to estimate a consistent model.

We presented two methods to solve the data association problem in distributed systems. The first approach is based on grid maps while the second approach uses landmark maps. Our grid-based approach applies Monte Carlo localization to perform a global pose estimation. By localizing one robot in the map of another robot, it associates positions in multiple grid maps. The system was evaluated in the context of mapping multi-floor building, where each floor was mapped independently. Our approach successfully generated constraints between individual floors and in this way generated consistent multi-floor maps. Our evaluation shows that maps generated my our method are more accurate than those obtained with a standard SLAM approach. Additionally, our experiments showed that adding constraints between overlapping SLAM graphs can improve the overall accuracy of the model. Inter-graph constraints allow to mitigate errors that are present in one graph using information from other graphs that correctly model the corresponding part of the environment.

Our algorithm to align landmark maps computes a Delaunay triangulation of the maps and then searches for similar triangles using geometric features. Matching triangles that are found in pairs of overlapping maps are then used to compute relative transformations. The algorithm has an average computational complexity of $O(n \log n)$ and is among the

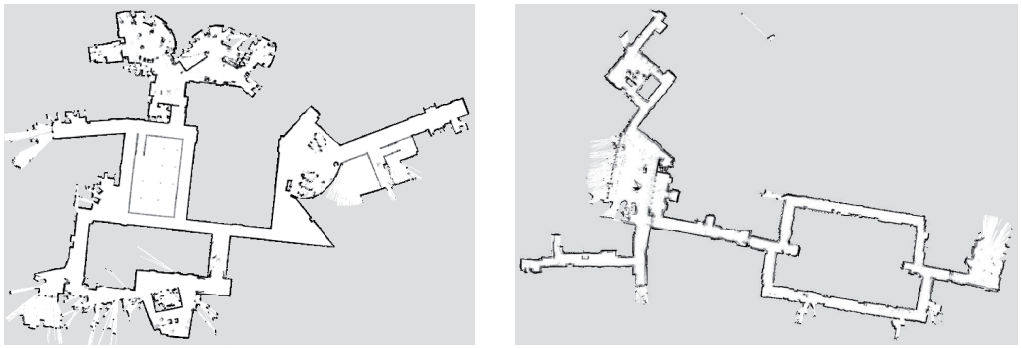


Figure 4.18: Maps of the Stata Center building. Left: Third floor, Right: Eighth floor. Without further background knowledge it is unclear how to align the maps.

fastest approaches that solve this particular problem. We evaluated our approach using several real world datasets as well as simulated data. The results show that constraints between SLAM graphs can be computed reliably and that consistent global models are estimated by the approach.

4.8.1. Limitations of Multi-Floor Mapping

As we illustrated in the experimental section, the application of our multi-robot system to multi-floor mapping was successful in typical real world buildings. Our approach was able to align floors even if they share only few common structures. Buildings with strong symmetries, however, may pose a problem to the described approach. In such cases, the global localization method will most likely converge to a multi-modal distribution and thus no constraints can be generated.

Our approach will furthermore not lead to improvements when a building is mapped whose floors do not share any similarities. These building occur rarely in real world but do exist. One such building is the Stata Center at MIT, a dataset is available from the Radish repository (Howard and Roy, 2003). Here, our approach was unable to find constraints between the third and the eighth floor, see Figure 4.18. This comes as no surprise, since even a manual alignment of the floor maps is nearly impossible without background knowledge.

Chapter 5
Efficient 3D Mapping

5.1. Introduction

When robots navigate autonomously, they need a model of the environment that defines regions which are safe to traverse. In Chapter 2 and Chapter 3, we assumed that the environment can be sufficiently modeled using planar, two-dimensional maps. This assumption is valid for many navigation tasks in buildings and other man-made structures. There are, however, several robotic applications that require a three-dimensional model of the environment. Three-dimensional models are relevant in many airborne, underwater, or extra-terrestrial missions and may also be necessary in domestic scenarios, for mobile manipulation tasks, or for navigation in multi-leveled environments.

In this chapter, we will assume that 3D maps are created using the sensors of a robot in combination with appropriate state estimation techniques. In general, one could create a 3D map manually, for example, using CAD software. But this manual approach is time-consuming and the model has to be created by an expert that has sufficient knowledge about the environment. Acquiring the 3D model using a mobile robot offers two distinct advantages: First, robots can be deployed without prior knowledge of the environment as long as they are able to navigate safely while taking sensor measurements. In hostile or remote environments there often will not be sufficient prior knowledge to create maps manually, examples include disaster scenarios, underwater operation, and extra-terrestrial exploration. The second advantage of creating maps using robots is that the model can then be created with the same type of sensor in the same setup that the robot is later using during navigation. This will ensure that the work space of the robot is modeled well so that all relevant structures in the environment are represented in the map, for instance, during localization.

The 3D representation we present in this chapter is designed to meet the following three requirements: It is a probabilistic representation, it models free and unmapped

areas, and it is efficient with respect to runtime and memory requirements. We will now discuss these three requirements in detail.

Probabilistic representation: To create 3D maps, mobile robots sense the environment by taking 3D range measurements. Such measurements are afflicted with uncertainty: Typically the error in range measurements is in the order of centimeters and there may also be more severe errors that are caused by reflections or dynamic obstacles. When the task is to create an accurate model from such noisy measurements, the underlying uncertainty has to be taken into account probabilistically. Multiple uncertain measurements can then be fused into a robust estimate of the true state of the environment. Another important aspect is that a probabilistic sensor fusion method allows for the integration of data from multiple sensors and of multiple robots.

Representation of unmapped areas: In autonomous navigation tasks, a robot can plan collision-free paths only for those areas that have been covered by sensor measurements. Unmapped areas, in contrast, need to be avoided and for this reason the map has to represent such areas explicitly. Furthermore, the knowledge about unmapped areas is essential during exploration. When maps are created autonomously, the robot has to plan its actions so that measurements are taken in previously unmapped areas. When unmapped areas are represented in the map, exploration targets can be determined, for instance, using a 3D variant of the frontiers approach which we discussed in Section 2.2.

Efficiency: The map is a central component of any autonomous system because it is used during action planning and execution. For this reason, access to the map needs to be efficient. It needs to be efficient with respect to access times but also with respect to memory consumption. From a practical point of view, memory consumption often is the major bottleneck in 3D mapping systems. For this reason, we require that the model is compact in memory so that large environments can be mapped, a robot can keep the model in its main memory, and it can be transmitted between multiple robots.

Several approaches have been proposed to model 3D environments. For comparison, we applied four different approaches to represent an example data set, a visualization of the results is given in Figure 5.1. The 3D measurements are represented using point clouds, elevation maps, multi-level surface maps as proposed by Triebel *et al.* (2006), and the representation presented in this chapter. None of the previous approaches fulfills all of the requirements we set out above. Point clouds store large amounts of measurement points and hence are not memory-efficient. Point clouds furthermore do not allow to differentiate between obstacle-free and unmapped areas and they provide no means of fusing multiple measurements probabilistically. Elevation maps and multi-level surface maps are efficient but do not represent unmapped areas either. Most importantly, these approaches cannot represent arbitrary 3D environments, such as the branches of the tree in the example.

In this chapter, we present a mapping approach based on octrees that combines the advantages of previous approaches to meet the requirements specified above. To fuse

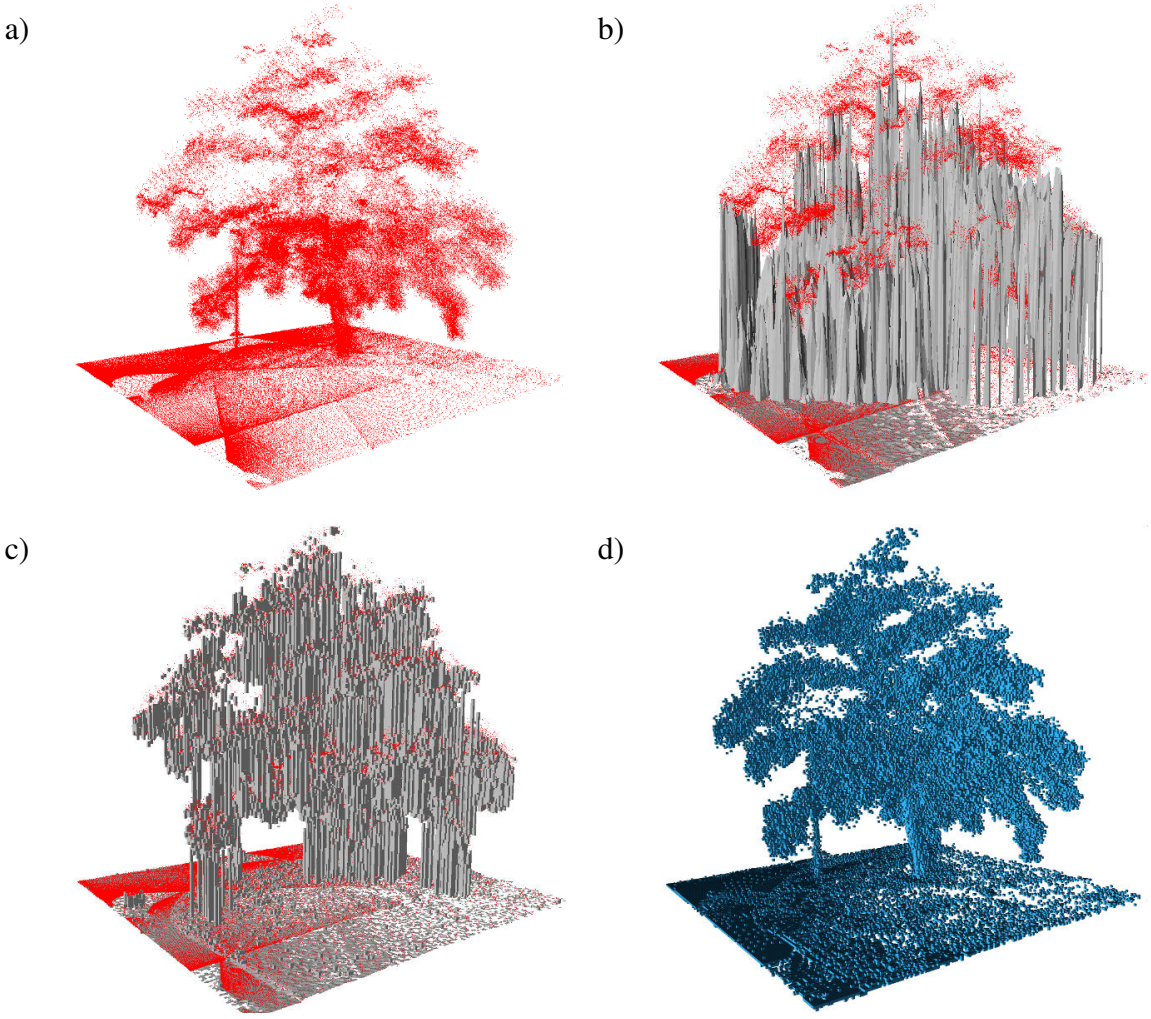


Figure 5.1: Measurements of a tree modeled using different representations. Visualized are a: Point clouds, b: an elevation map superimposed on point clouds, c: a multi-level surface map superimposed on point clouds, d: our approach.

measurements, we apply probabilistic state estimation techniques similar to those used in 3D occupancy grid mapping. In this way, we model obstacles as well as obstacle-free volumes in the environment and we therefore also represent unmapped areas. Our approach stores map data in an octree that keeps the memory consumption at a minimum. As a key contribution of our approach, we furthermore introduce a lossless compression method that reduces the memory requirement by locally combining coherent map volumes. We implemented our approach and thoroughly evaluated it on various simulated and real datasets of both indoor and large-scale outdoor environments. As we will show in Section 5.5, our approach meets the requirements set out above and outperforms previous approaches.

In Section 5.3, we present an extension to our mapping approach that exploits hierarchical dependencies in the environment. Most mapping approaches are based on the assumption that the environment can be mapped using a single global map. Such monolithic maps are, however, unable to represent movable objects and they cannot adapt mapping parameters to the local structure. To address these challenges, our hierarchical approach maintains a collection of submaps in a tree-structure, where each node represents a subspace of the environment. The subdivision applied in our system is based on a spatial relation that is defined by the user. In contrast to a monolithic map, our hierarchical method is able to represent movable objects and it can adapt mapping parameters locally. To evaluate our extended mapping approach, we applied it to the scenario of mobile manipulation, where individual objects need to be modeled precisely.

In Chapter 2 and Chapter 3 we presented techniques to coordinate teams of robots that create maps of an environment autonomously. A central aspect of these strategies was how to choose from a number of possible exploratory actions. In Section 5.4, we present an approach that acquires 3D models autonomously by generating and evaluating 3D sensing actions. It takes into account the expected information of potential future observations. This estimation makes use of the information about unmapped areas that is represented in our 3D map structure. The proposed exploration system was evaluated using our hierarchical 3D mapping approach in experiments conducted with a simulated mobile manipulation robot.

In our experiments, we generate 3D maps from range measurements. Sensor modalities for acquiring such measurements include laser range finders and stereo cameras. Throughout this chapter, we assume that measurements are taken from known sensor origins. Estimating the poses of the sensor using appropriate 3D SLAM methods is not addressed by our mapping technique.

This chapter is organized as follows. In the next section, we present our map technique including its underlying data structure, the sensor fusion approach we apply, and techniques to reduce its complexity. We extend the basic map structure to a hierarchical model in Section 5.3 and present an implementation for mobile manipulation. An approach to mapping 3D environments autonomously is presented in Section 5.4. In Sec-

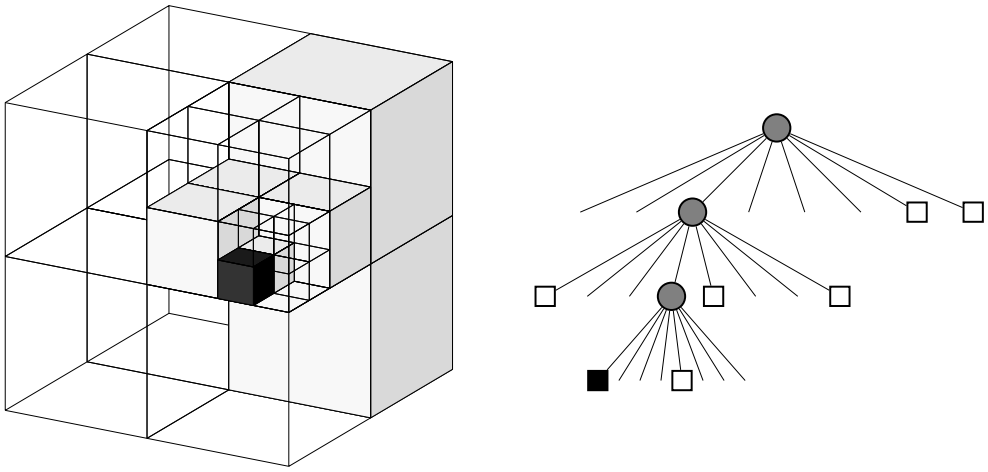


Figure 5.2: Illustration of the octree data-structure. Left: Example of an octree storing free voxels (shaded white) and occupied voxels (black). Right: The corresponding tree representation.

tion 5.5, we then evaluate our approach in different scenarios including large-scale outdoor maps, modeling of indoor environments, mobile manipulation, and 3D exploration. Finally, we provide a detailed discussion of related work in Section 5.6.

5.2. The OctoMap Mapping Framework

Our approach models obstacles as well as free space. To achieve this task efficiently, it discretizes the environment by dividing it into equally-sized cubic volumes called voxels. Our approach then estimates the probability of each voxel to be occupied by an obstacle. By integrating multiple measurements probabilistically, sensor noise is taken into account and data from multiple sources can be fused. Our approach maintains data in a tree-based representation that allocates memory only for those parts of the model that carry information about the environment. Furthermore, a lossless compression method ensures the compactness of the resulting models. In the following, we describe the data structure, the sensor fusion approach and techniques to reduce the memory complexity of our representation.

5.2.1. The Octree Data Structure

To store information about obstacles and free space in the environment, we use octrees (Meagher, 1982; Wilhelms and Van Gelder, 1992). An octree is a tree-based data structure that represents the 3D space by recursively subdividing the corresponding volume into eight sub-volumes until a given minimum cube size is reached. Each volume created in this process is represented as a node in the octree and each inner node in the tree has at most eight children. The relation between the volumes and the tree is illus-

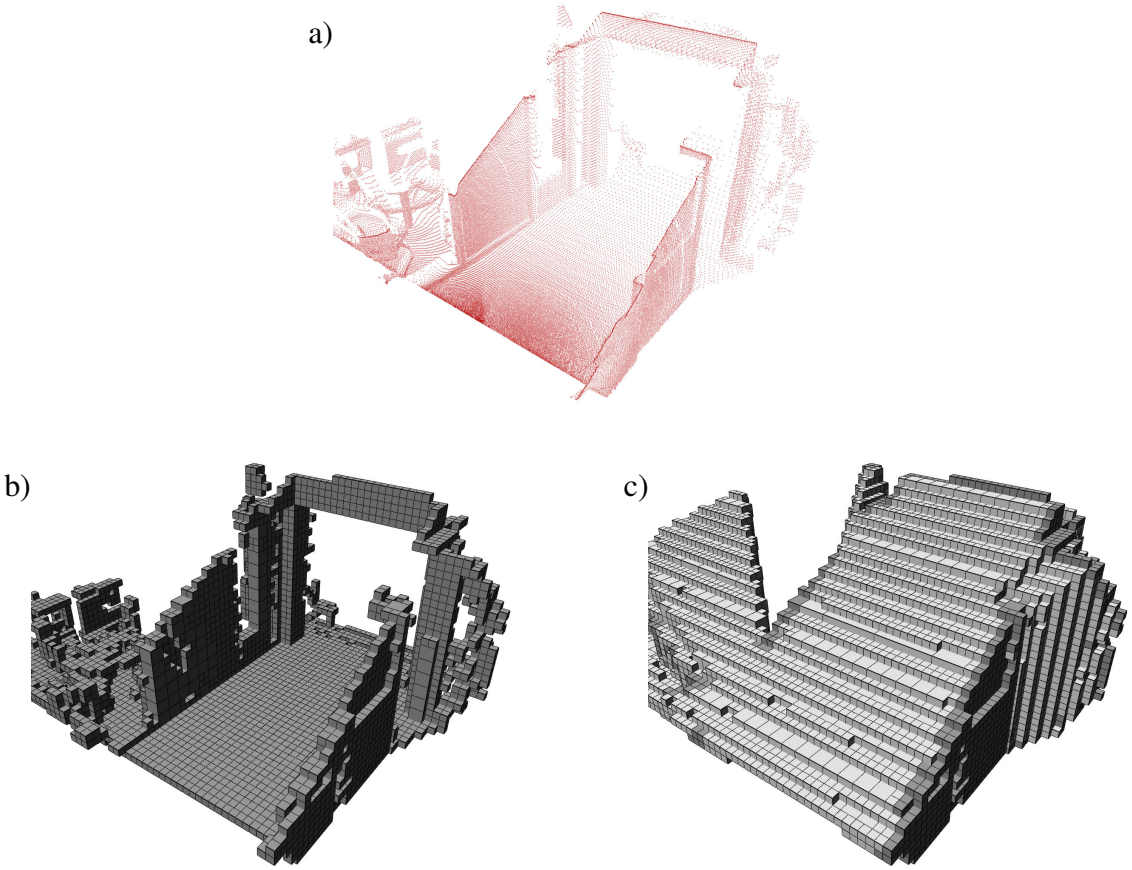


Figure 5.3: An octree generated from example data. a: Point cloud recorded in a corridor with a tilting laser scanner. b: Octree generated from the data, showing occupied voxels only. c: Visualization of the octree showing occupied voxels (dark) and free voxels (white).

trated in Figure 5.2. When used for 3D modeling, each node in the octree represents one voxel. The size of the voxels that corresponds to the leaf nodes of the tree determines the resolution of the 3D map.

In its basic form, octrees model a Boolean property and in the context of robotic mapping, this usually is the occupancy of a volume. If a certain volume is measured as occupied, the corresponding node in the octree is initialized. Any uninitialized node, however, can be free or unknown. To resolve this ambiguity, free cells can be explicitly represented by nodes marked as *free* and sub-volumes which are not initialized then implicitly model *unknown* areas.

The use of such Boolean occupancy states allows for a compact representation of the octree: If all children of an inner node are occupied leaf nodes, or all are free leaf nodes, then the information encoded by the child nodes can be stored in the parent node instead. The children of the node can then be removed. Pruning the tree in this way avoids storing redundant information and leads to a substantial reduction in the number of nodes that need to be maintained. Figure 5.3 shows a visualization of a 3D laser scan and the octree

created from this measurement. It can be seen that a number of free nodes have been combined into bigger voxels. This is the result of the compression mechanism we just described.

When maps are created using mobile robots, one has to cope with sensor noise and temporarily or permanently changing environments. A discrete occupancy state is then insufficient to fuse multiple measurements. Instead, occupancy has to be modeled probabilistically, that is, by estimating the probability of each voxel to be occupied. In the next section, we present the state estimation technique applied in our approach.

5.2.2. Probabilistic Sensor Fusion

We have already discussed occupancy grid mapping in the context of two-dimensional maps in Section 2.2. This approach can be extended to three-dimensional environments in a straightforward way: Instead of estimating the state of cells in a 2D grid map, we estimate the state of voxels in a 3D map. Recall that the probability of a cell to contain an obstacle is estimated using a binary random variable. In our approach, we integrate sensor readings by applying occupancy grid mapping as introduced by Moravec and Elfes (1985) and we adopt the common assumption that all voxel are independent of each other. Analogous to Moravec (1988), we derive a recursive update rule for the probability $P(c \mid z_{1:t})$ of a voxel c to contain an obstacle given all previous sensor measurements $z_{1:t}$. First, we apply Bayes' rule and obtain

$$P(c \mid z_{1:t}) = \frac{P(z_t \mid c, z_{1:t-1}) P(c \mid z_{1:t-1})}{P(z_t \mid z_{1:t-1})}. \tag{5.1}$$

We then compute the ratio

$$\frac{P(c \mid z_{1:t})}{P(\neg c \mid z_{1:t})} = \frac{P(z_t \mid c, z_{1:t-1})}{P(z_t \mid \neg c, z_{1:t-1})} \frac{P(c \mid z_{1:t-1})}{P(\neg c \mid z_{1:t-1})}. \tag{5.2}$$

Similarly, we obtain

$$\frac{P(c \mid z_t)}{P(\neg c \mid z_t)} = \frac{P(z_t \mid c)}{P(z_t \mid \neg c)} \frac{P(c)}{P(\neg c)},$$

which can be transformed into

$$\frac{P(z_t \mid c)}{P(z_t \mid \neg c)} = \frac{P(c \mid z_t)}{P(\neg c \mid z_t)} \frac{P(\neg c)}{P(c)}. \tag{5.3}$$

Applying the Markov assumption that the current observation is independent of previous observations given we know that a voxel is occupied gives

$$P(z_t \mid c, z_{1:t-1}) = P(z_t \mid c) \tag{5.4}$$

and utilizing the fact that $P(\neg c) = 1 - P(c)$, we obtain

$$\frac{P(c \mid z_{1:t})}{1 - P(c \mid z_{1:t})} = \frac{P(c \mid z_t)}{1 - P(c \mid z_t)} \frac{P(c \mid z_{1:t-1})}{1 - P(c \mid z_{1:t-1})} \frac{1 - P(c)}{P(c)}. \tag{5.5}$$

Finally, this equation can be transformed into the following recursive update formula:

$$P(c \mid z_{1:t}) = \left[1 + \frac{1 - P(c \mid z_t)}{P(c \mid z_t)} \frac{1 - P(c \mid z_{1:t-1})}{P(c \mid z_{1:t-1})} \frac{P(c)}{1 - P(c)} \right]^{-1} \tag{5.6}$$

This update formula depends on the current measurement z_t, a prior probability $P(c)$, and the previous estimate $P(c \mid z_{1:t-1})$. The term $P(c \mid z_t)$ denotes the probability of voxel c to be occupied given the measurement z_t. This value is specific to the sensor that generated z_t and we provide details on the sensor model we used in our experiments in Section 5.5.1.

Under the common assumption of a uniform prior probability, $P(c) = 0.5$, and by using the logOdds notation, Equation (5.6) can be rewritten as

$$\text{logOdds}(c \mid z_{1:t}) = \text{logOdds}(c \mid z_{1:t-1}) + \text{logOdds}(c \mid z_t) \tag{5.7}$$

$$\text{with} \quad \text{logOdds}(c) = \log \left[\frac{P(c)}{1 - P(c)} \right] \tag{5.8}$$

This formulation of the update rule allows for faster updates since multiplications are replaced by additions. In case of pre-computed sensor models, the logarithms do not have to be computed during the update step. Note that logOdds values can be converted into probabilities and vice versa and we therefore store this value for each voxel instead of the occupancy probability.

When a 3D map is used for navigation, a threshold on the occupancy probability $P(c \mid z_{1:t})$ is often applied. A voxel is considered to be *occupied* when the threshold is reached and is assumed to be *free* otherwise, thereby defining two discrete states. From Equation (5.7) it is evident that to change the state of a voxel we need to integrate as many observations as have been integrated to define its current state. In other words, if a voxel was observed free for n times, then it has to be observed occupied at least n times before it is considered occupied according to the threshold (assuming that free and occupied measurements are equally likely in the sensor model). While this property is desirable in static environments, a mobile robot is often faced with temporary or permanent changes in the environment and the map has to adapt to these changes quickly. To ensure this adaptability, Yguel *et al.* (2007a) proposed a clamping update policy that defines an upper and lower bound on the occupancy estimate. Instead of using Equation (5.7)

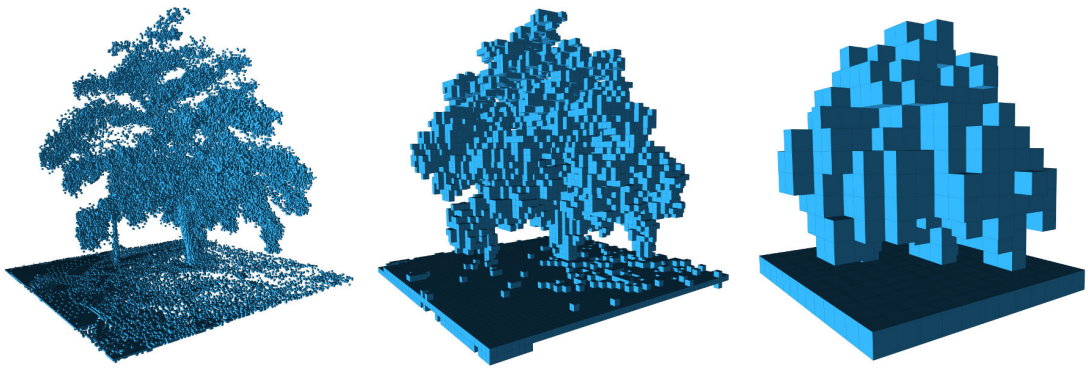


Figure 5.4: Multi-resolution queries of an octree. By limiting the depth of a query, multiple resolutions of the same map can be obtained. Shown is a visualization of voxels with an occupancy probability higher than 0.5 at a resolution of 0.08 m, 0.32 m, and 1.28 m (left to right).

directly, occupancy estimates are updated according to

$$\begin{aligned} \text{logOdds}(c \mid z_{1:t}) = & \\ & \max\left(\min\left(\text{logOdds}\left(c \mid z_{1:t-1}\right) + \text{logOdds}\left(c \mid z_t\right), l_{\max}\right), l_{\min}\right), \end{aligned} \tag{5.9}$$

where $l_{\min}$ and $l_{\max}$ denote the lower and upper bound on the logOdds value. Intuitively, this modified update formula fixes the number of updates that are needed to change the state of a voxel. Applying the clamping update policy in our approach leads to two advantages: we ensure that the confidence in the map remains bounded and as a consequence the model can adapt to changes in the environment quickly. Furthermore, we are able to compress neighboring voxels that reach the upper or lower bound. As we will discuss in Section 5.2.4, this leads to a considerable reduction in the number of voxels that have to be maintained.

5.2.3. Multi-Resolution Queries

When measurements are integrated into our map structure, probabilistic updates are performed only for the leaf nodes in the octree. But since an octree is a hierarchical data structure, we can make use of the inner nodes in the tree to enable multi-resolution queries. Observe that we yield a coarser subdivision of the 3D space when the tree is traversed only up to a given depth that is not the depth of the leaf nodes. Each inner node spans the volume that its eight children occupy, so to determine the occupancy probability of an inner node, we have to aggregate the probabilities of its children. Several strategies could be pursued to determine the occupancy probability of a node c given its eight sub-volumes c_i (Kraetzschmar et al., 2004). Depending on the application at hand,

either the average occupancy

$$\bar{l}(c) = \frac{1}{8} \sum_{i=1}^{8} \text{logOdds}(c_i) \tag{5.10}$$

or the maximum occupancy

$$\hat{l}(c) = \max_i \text{logOdds}(c_i) \tag{5.11}$$

can be used, where $logOdds(c)$ returns the current logOdds occupancy value of a node c. Using the maximum child occupancy to update inner nodes can be regarded a conservative strategy which is well suited for robot navigation. By assuming that a volume is occupied if any part of it has been measured occupied, collision-free paths can be planned and for this reason the maximum occupancy update is used in our system. Note that in an even more conservative setting, $logOdds(c)$ can be defined to return a positive occupancy probability for *unknown* cells as well. An example of an octree queried for occupied voxels at several resolutions is shown in Figure 5.4.

5.2.4. Compact Map Representation

We will now discuss techniques to compress octree-based 3D maps, how to implement the data structure efficiently, and how to generate compact map-files.

Lossless Map Compression

In Section 5.2.1, we explained how tree pruning can reduce the amount of redundant information in octrees with discrete occupancy states in which a voxel can be either *occupied* or *free*. The same technique can also be applied in maps that use probabilistic occupancy estimates. In general, however, one cannot expect the occupancy probability of neighboring nodes to be identical, even if both voxel are occupied by the same physical obstacle. Sensor noise and discretization errors can lead to different probabilities and therefore interfere with compression schemes that rely on identical node information. A possible solution to this problem is to apply a threshold on the voxel probability, for example 0.5, and in this way generate a discrete state estimation as suggested by Fairfield *et al.* (2007). This, however, is not a lossless compression since the individual probability estimates cannot be recovered after the tree has been pruned.

In our approach, we achieve a lossless map compression by applying the clamping update policy given in Equation (5.9). Whenever the logOdds value of a voxel reaches either the lower bound $l_{\min}$ or the upper bound $l_{\max}$, we consider the node *stable* in our approach. Intuitively, stable nodes have been measured free or occupied with high confidence. In a static environment, all voxels will converge to a stable state after a sufficient

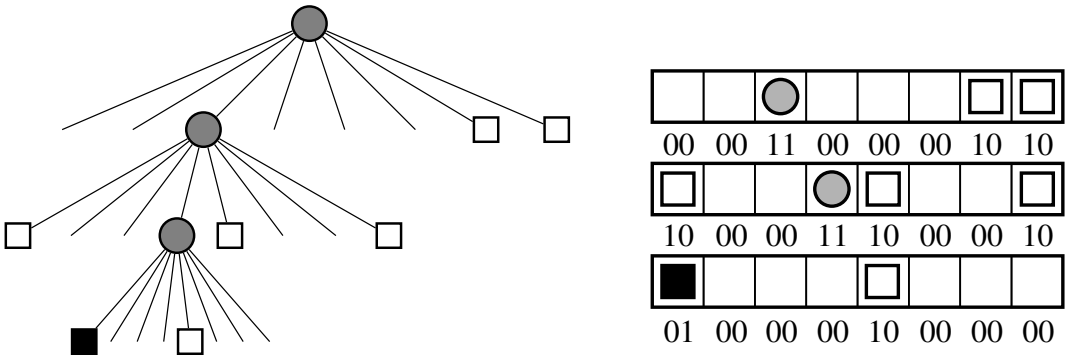


Figure 5.5: Example of the bit stream encoding of an octree (compare Figure 5.2). The octree structure shown on the left can be stored using only 48 bits, 2 bits for each child of a node, as shown on the right. Here, each row corresponds to one inner node, starting at the root node.

number of measurements have been integrated. In our experiments, for example, five agreeing measurements are sufficient to render an unknown voxel into a stable voxel. If all children of an inner tree node are stable leaf nodes with the same occupancy state, then the children can be pruned. Should future measurements be integrated that contradict the state of the corresponding inner node, then its children are regenerated and updated accordingly. Applying this compression does not lead to a loss of information in the probabilistic model. It does, however, lead to a considerable reduction in the memory consumption, in our experiments we observed an improvement of up to 44%.

Memory-Efficient Implementation

Each node in an octree needs to maintain an ordered list of its eight children and this can be achieved naively by storing eight pointers per node. The memory required for the pointers, $8\,\text{bit} \times 32\,\text{bit} = 256\,\text{bit}$ per node on a 32 bit architecture, will lead to a significant memory overhead (Wilhelms and Van Gelder, 1992). With an implementation trick, however, one can overcome this problem. In our implementation, we use one pointer per node which points to an array of eight pointers. This array is only allocated if children need to be initialized. The majority of nodes in an octree are leaf nodes that do not have children. Assuming that the logOdds value is stored as a 32 bit floating point value, each leaf occupies 64 bit compared to 288 bit in the naive implementation, thus leading to a more compact representation.

Compact Map Files

Many robotic applications require maps to be stored in files. This includes cases where a map is generated during a training phase and later is used by mobile robots during path planning and localization. Another use case is a multi-robot system where maps are

exchanged between robots. In either case, a compact representation is required in order to minimize the consumption of disk space or communication bandwidth.

A straightforward encoding is achieved by storing the occupancy probability of each node as a 32 bit floating point number along with a vector of eight bits that specifies which of the node's children has been initialized. Starting from the root node of the tree, a depth-first search is then used to encode the octree structure and all occupancy probabilities.

More compact files can be generated whenever a maximum likelihood estimate of the map is sufficient for the task at hand and the individual probability estimates can be thresholded. This is the case for most navigation systems as they do not add information to the map once it has been generated. Octree maps can then be encoded as a compact bit stream by representing each node using the discrete occupancy states of its children. Beginning at the root node, each child that is not a leaf node is recursively added to the bit stream in a depth-first search. We do not encode leaf nodes explicitly because they can be reconstructed from their occupancy state that is stored with the parent node. As motivated above, unmapped volumes can be of special interest in robotic systems and for this reason, we explicitly differentiate between free and unknown areas. In our encoding, each inner node is represented by 16 bits in the map file, two bits per child to encode one of four possible states: children can be free, occupied, unknown, or they can be inner nodes. Our encoding is considerably more compact than the straightforward encoding that stores all probability estimates and takes a total of 40 bits per node. Figure 5.5 illustrates our bit-stream encoding: Each row represents one node and the upper row corresponding to the root node. The lower row only contains leafs so no further nodes are added. In our experiments, file sizes never exceeded 2 MB, even in the case of a large outdoor environments with a size of 292 m $\times$ 167 m $\times$ 28 m (see Figure 5.15 on page 129).

5.3. Hierarchies of 3D Maps

In the following, we present an extension to our octree-based mapping approach that exploits hierarchical dependencies in 3D environments. A collection of submaps is maintained in a tree-structure, where each node represents a subspace of the environment. The subdivision applied in our system is based on a user-defined spatial relation. Figure 5.6 gives an illustration of a hierarchy that is based on the relation of objects and their supporting planes.

Compared to a single, monolithic map of the environment our hierarchical approach exhibits a number of advantages: First, each submap is maintained independently and mapping parameters such as the resolution can be adapted for each submap. Second, submaps can be manipulated independently. For example, one of the submaps represent-

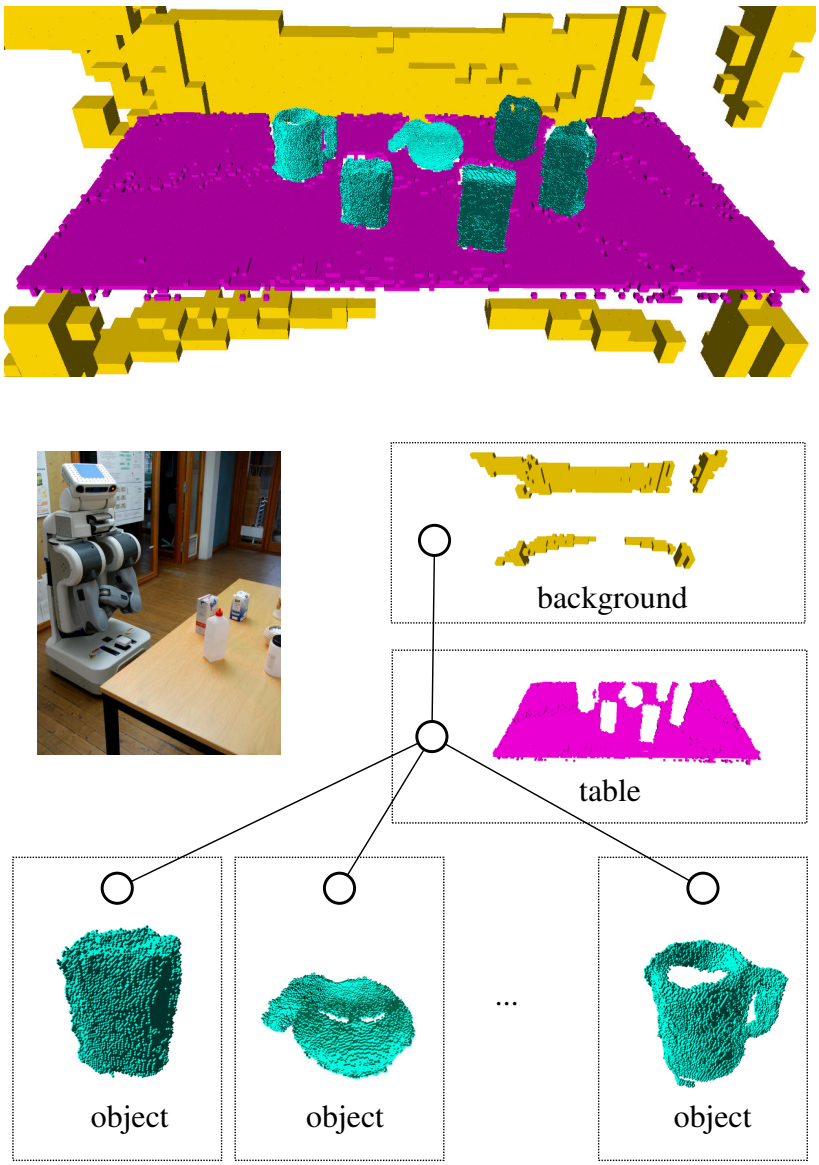


Figure 5.6: Illustration of a map hierarchy based on supporting planes. Top: Visualization of a hierarchical model of the table top scene shown in the photo. The objects (green) and the table (magenta) are represented in individual submaps. Bottom: illustration of the map hierarchy. Each map node is represented in a separate octree map.

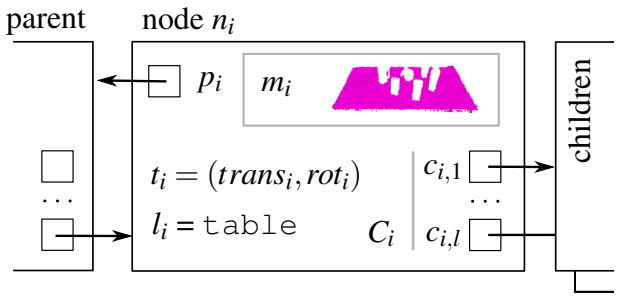


Figure 5.7: Illustration of a node in our map hierarchy. Each node consists of a separate 3D map and the hierarchy is defined via parent and child pointers.

ing an individual object can be moved while the rest remains static. Third, hierarchical dependencies of submaps can be encoded in the hierarchy. For example, all objects on a table can be associated to this table and if the table is moved then the objects are moved along with it.

In the following, we will introduce the proposed hierarchical data structure, explain how to generate a hierarchy of maps based on a spatial relation, and show how the representation can be updated. We will then present a specific implementation of a map hierarchy for table top manipulation.

5.3.1. Definition of the Map Hierarchy

We define a map hierarchy as a tree of map nodes, where each node maintains a separate 3D map that is oriented in space. Formally, we define a hierarchy M as a set of nodes n_i

$$M = \{n_1, \ldots, n_k\}, \tag{5.12}$$

where k is the number of map nodes in the tree. A node n_i is defined as

$$n_i = (m_i, p_i, C_i, t_i, l_i), \tag{5.13}$$

see Figure 5.7 for an illustration. Each node stores a 3D map m_i and the tree structure is defined via a reference to the node's parent node p_i and a set of references to child nodes

$$C_i = \{c_{i,1}, \ldots, c_{i,l}\}. \tag{5.14}$$

To be able to represent objects that move in the environment, we interpret each node's maps m_i with respect to a reference frame t_i. The 6D transform t_i denotes the relative

transform from the parent's reference frame to the node's local reference frame. We recover the global reference frame t_i^{global} of node n_i by traversing the tree. Starting at node n_i, we follow the parent references upwards to the root of the tree. This leads to the following recursive formulation

$$t_i^{global} = \begin{cases} t_i & n_i \text{ is root node} \\ t_{p_i}^{global} \circ t_i & \text{else} \end{cases}, \tag{5.15}$$

where $t_{p_i}^{global}$ denotes the global reference frame of n_i's parent p_i and $\circ$ is the 6D composition operator.

By defining parent and child pointers and relative transforms we define the hierarchy of 3D maps. In the node definition given above, we include an additional semantic label l_i. This label stores semantic information that is associated with a map node, for example, an object class identifier. It can be used, for instance, to facilitate data association or to adapt certain map properties. To determine the semantic label, we assume that a labeling function $L(z)$ exists that returns a label for a given sensor measurement z. The labeling function can be implemented by applying object detection algorithms, for example the approach of Ruhnke *et al.* (2009), or scene analysis methods, for example the approach presented by Nüchter *et al.* (2008). In Section 5.3.4, we will give an example of a labeling function which is based on plane-detection.

5.3.2. Construction via Spatial Relations

In this section, we describe how to generate a map hierarchy from 3D measurements. We assume that 3D measurements are given as a pair (z, o_z) consisting of a cloud of endpoints z and the corresponding sensor origin o_z. We assume that techniques exist to analyze point cloud measurements for meaningful structures. More specifically, we assume that there exists a method $S(z)$ which computes a segmentation of a point cloud measurement z so that

$$S(z) = \{z_1, \ldots, z_m\},\ z = \bigcup_i z_i, \tag{5.16}$$

where each segment z_i corresponds to a structure such as surface planes, geometric primitives, or point clusters.

The construction of the map hierarchy is based on a spatial relation that is provided by the user. This relation $r(n_i, n_j) \subseteq M \times M$ determines whether node n_i is at a higher level in the hierarchy than n_j. Real-world examples of such relations include the relations *supports* or *contains*.

Whenever a new node n_{new} is added to the hierarchy M, the relation r is evaluated for

each existing node and a set $\mathcal{A}$ of candidate ancestors is generated

$$\mathcal{A} = \{n \in M \mid (n, n_{new}) \in r\} \tag{5.17}$$

In general, the candidate set $\mathcal{A}$ can consist of more than one candidate. For example, a cup standing on a table is *supported by* the table but also by the floor that the table is standing on. For this reason, n_{new} is added to the hierarchy as a child of the deepest candidate node a^* in the hierarchy

$$a^* = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \operatorname{depth}(a), \tag{5.18}$$

where $depth(a)$ denotes the depth of a in the tree.

More than one relation may be used to define the hierarchy as long as they are mutually exclusive, that is, no more than one relation contains the tuple (n, n_{new}) for a given node n. In this case, all relations $r \in R$ in a set of relations are evaluated in Equation (5.17). Using more than one relation allows the hierarchy to model, for example, that objects are *supported by* a table and that this table is *contained in* a specific room. Note that to ensure mutual exclusivity an object that is *supported by* a table cannot be *contained in* the table at the same time.

There are certain situations which cannot be modeled using a tree structure, for example, in the case of a *supported by*-hierarchy objects cannot rest on two tables simultaneously. However, this restriction allows us to update the reference frames of map nodes recursively. In the example, when an object is located on a table and the table's reference frame is updated because the table was moved in the environment, then the object's reference frame is updated along with the table's.

5.3.3. Update of the Hierarchy

An existing map hierarchy can be updated in three ways. First, new 3D measurements can be integrated into the node maps. Second, the position or orientation of a node can be updated based on a perceived movement. Third, objects may be removed from the scene.

To integrate a measurement into an existing hierarchy, we first need to determine the nodes that have to be updated. We assume that an association function f_A exists. Given a point cloud measurement z, this function returns the lowest node $n \in M$ in the tree that z falls into or the empty set if no such node can be found. Given this association function and a segmentation function S, the model update follows three steps:

1. Segment the point cloud $\{z_i\} = S(z)$

2. Associate each segment z_i to an existing map node using f_A. Multiple segments can be associated to the same map node.

3. Update node maps determined in previous step using the corresponding segments and the sensor origin o_z. Create a new map nodes for segments which could not be associated to a node.

Our data structure is able to model changes that result from object movements in the environment. Such movements can be perceived using object detection methods, for example, based on viewpoint feature histograms as proposed by Rusu *et al.* (2010). To integrate a perceived relative 6D-transformation t, that encodes a movement and a change of orientation of a submap n_i, the corresponding node transform is updated according to: $t_i \leftarrow t_i \circ t$.

In general, there are also unobserved transformations of the environment and objects can be removed from the environment entirely. To cope with such cases, we estimate the probability of existence for each node n. The probability $P(n \mid z_{1:t})$, where $z_{1:t}$ denotes the set of all measurements up to time t, is estimated using a binary Bayes filter similar to Equation (5.6):

$$P(n \mid z_{1:t}) = \left[1 + \frac{1 - P(n \mid z_t)}{P(n \mid z_t)} \frac{1 - P(n \mid z_{1:t-1})}{P(n \mid z_{1:t-1})} \frac{P(n)}{1 - P(n)}\right]^{-1} \tag{5.19}$$

The inverse sensor model $P(n \mid z_t)$ and the prior probability $P(n)$ both are specific to the sensor and the segmentation used for mapping. In our implementation, $P(n \mid z_t)$ confirms the node's existence if measurements of z_t are integrated into the node's map and it opposes the existence if no measurements are integrated although the node is within the sensor's field of view.

5.3.4. 3D Models for Tabletop Manipulation

In this section, we present a specific implementation of a map hierarchy H' along with a segmentation S', labeling function L', relation r', and association function f'_A. We apply our approach to the task of representing an indoor environment for tabletop manipulation. Our model is designed to represent the entire working space of the robot, at the same time it represents objects at a fine resolution to facilitate, for example, grasp computation. In our implementation of H', we differentiate between objects, tables and the remaining structures, such as the floor and the walls, which are part of the scene background, see Figure 5.6 for an illustration. We make use of two central properties of the proposed hierarchical map structure: First, objects are modeled separately so that they can be updated and manipulated independently. Second, we adapt mapping parameters locally, in the case of H', the mapping resolution is adapted to the semantic class of a map node.

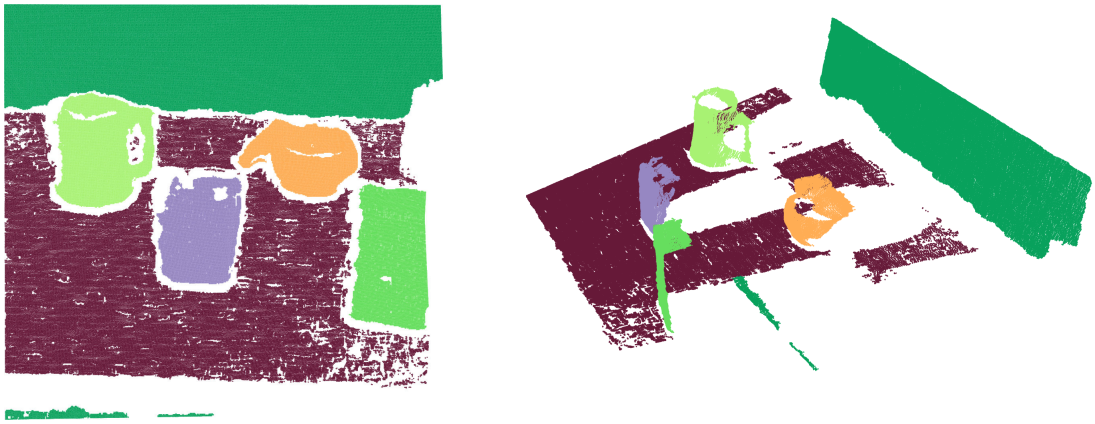


Figure 5.8: Example of a segmented point cloud taken with a stereo camera. A table top scene is segmented and segments are visualized in different colors. Left: Scene as seen by the cameras. Right: Scene shown from the side.

Construction of the Hierarchy

The map hierarchy H' is based on supporting planes, that is, we follow the common assumption that objects rest on flat tabletops. To identify supporting structures and to compute a segmentation of measurements we analyze surface elements in the 3D point clouds. We first locate horizontal planes at approximate table height, between 0.5 m and 1.0 m. This is done by fitting planes using the random sample consensus (RANSAC) algorithm (Fischler and Bolles, 1981). The sets of table inliers are stored in segments $z_{table,1}$ to $z_{table,n}$.

The planes are then used to identify object points in the measurement point cloud. Objects are defined as measurements above a detected table plane. To segment those points into individual object measurements $z_{obj,1}$ to $z_{obj,m}$, the object points are clustered based on a threshold on the Euclidean point distance, in our experiments we use 0.01 m. All points that are neither table inliers nor belong to an object cluster, are stored in a background segment z_{bg}. The result of this geometric analysis defines the segmentation function $S'(z)$:

$$S'(z) = \{z_{bg},\ z_{table,1},\ \ldots,\ z_{table,n},\ z_{obj,1},\ \ldots,\ z_{obj,m}\}. \tag{5.20}$$

An example of a segmented point cloud can be seen in Figure 5.8. Based on the result of $S'(z)$ we define the labeling function $L'(z)$ as:

$$L'(z) = \begin{cases} \texttt{table} & z \in \{z_{table,1}, \ldots, z_{table,n}\} \\ \texttt{object} & z \in \{z_{obj,1}, \ldots, z_{obj,m}\} \\ \texttt{background} & \text{else} \end{cases}. \tag{5.21}$$

Finally, the spatial relation $r'(n_i, n_j)$ is defined based on the node labels:

$$\begin{aligned} r' = \{\, (n_i, n_j) \mid n_i, n_j \in M',\, & (l_i = \texttt{table} \wedge l_j = \texttt{object}) \vee \\ & (l_i = \texttt{background} \wedge l_j = \texttt{table}) \vee \\ & (l_i = \texttt{background} \wedge l_j = \texttt{object}) \,\}, \end{aligned} \tag{5.22}$$

where l_i and l_j denote the labels of nodes n_i and n_j.

Node Maps

We maintain node maps using the OctoMap mapping framework introduced in Section 5.2 and each 3D map is represented using a separate octree. All map updates are local with respect to the map node's reference frame. When a new node n_i is added to the hierarchy, we use the first point cloud measurement that is integrated into the node to determine its reference frame t_i relative to the parent node p_i. The translational component is computed from the centroid of the measurement. While this constant offset does not have an influence on the distribution of voxels, the orientation of the node map has to be chosen carefully. It determines the orientation of the map voxels and in our experiments we observed that smooth models are generated when the voxel orientation follows the dominant surface normals of the object. For this reason, we use the principal component analysis on the distribution of measurement points to determine the node orientation. Although this cannot guarantee an optimal orientation of the reference frame with respect to all future measurements that are integrated into n_i, it typically results in smoother object surfaces than choosing a predefined orientation.

To integrate a measurement into a node map, we first transform it into the reference frame of the node map. Let (z, o_z) be a measurement and n_i the map node to be updated. The transform of the measurement and the corresponding origin are given by

$$(z', o'_z) = ((t_i^{global})^{-1}(z), (t_i^{global})^{-1}(o_z)), \tag{5.23}$$

where $(t_i^{global})^{-1}$ denotes the inverse global transform of n_i. Map m_i is then updated using (z', o'_z). To efficiently determine those voxels that need to be updated, we perform a ray-casting operation from o'_z to each measurement endpoint in z'. Then, we update volumes along the beam using occupancy grid mapping as described in Section 5.2.2. For the sake of efficiency, we update only those freespace voxels along the beam that fall within the oriented bounding box of m_i.

An important advantage of the proposed map hierarchy is the ability to adapt the mapping resolution based on the semantic class. In the context of mobile manipulation, for instance, objects usually need to be modeled at very fine resolutions (e.g., millimeters) while in many applications, a table top can be modeled at a coarser resolution (e.g., cen-

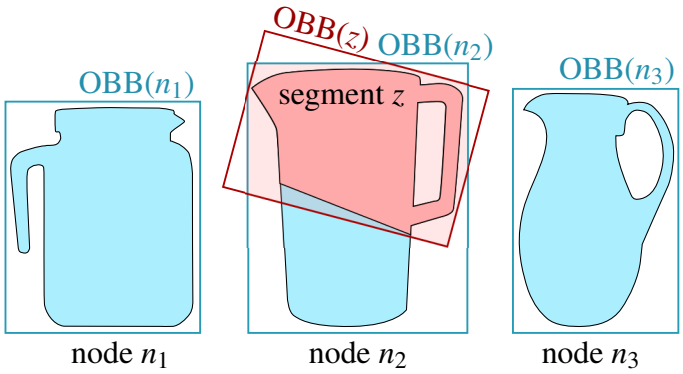


Figure 5.9: 2D illustration of data association based on oriented bounding boxes (OBBs). The OBB of a measurement segment z (red) is tested for intersection with the OBBs of existing map nodes n_i (blue).

timeters) and walls and the floor can be modeled even coarser. We applied such a multi-resolution approach in our experiments and we will show that it resulted in a significant reduction in both memory consumption and computation time.

Node Association

To implement an association module for table top manipulation f'_A, we use oriented bounding boxes (OBBs) and the node labels stored in the map hierarchy. Given a point cloud segment z, we first use its semantic label $L'(z)$ to find a set of existing map nodes $\mathcal{N}$ with the same label. Then we determine the OBB of z by performing principle component analysis and we compute the OBB of each node $n_i \in \mathcal{N}$. This can be done efficiently using the reference frame t_i and the extend of the node map m_i. To find the node that the measurement z falls into, we test the bounding box of z for intersection with the bounding boxes of the map nodes in $\mathcal{N}$. This test can be performed efficiently using, for example, Gottschalk's algorithm (Gottschalk, 2000). A 2D illustration of the bounding box test is given in Figure 5.9. If the objects on the table are well separated and each object is represented by one node in the hierarchy, we will find at most one intersection. In cases, where the the OBB of z intersects the OBBs of multiple map nodes, f'_A returns the map node whose centroid has the smallest Euclidean distance to the centroid of z.

5.4. Autonomous Model Acquisition

In the previous sections, we explained how to generate 3D models from *given* sensor data. In this section, we consider the problem of learning models autonomously. Our exploration system is an information-driven approach, it takes into account the expected information of potential, future observations. Our approach can be seen as an extension of the exploration system of Stachniss *et al.* (2005) towards 3D environments. Both a single

Algorithm 5 Autonomous Model Acquisition.

$M_0 \leftarrow \emptyset, t \leftarrow 0$
$z_0 \leftarrow$ getMeasurement()
$M_1 \leftarrow$ updateModel(M_0, z_0)
repeat
  $V \leftarrow$ sampleViewpoints(M_t) *// generate viewpoints*
  for all $v \in \mathcal{V}$ **do**
    $c_v \leftarrow$ estimateCost(M_t, x_t, v) *// compute cost*
    $u(v) \leftarrow \alpha\, I(M_t; v) - (1-\alpha)\, c_v$ *// trade of information gain and cost*
  end for
  $v^* \leftarrow \text{argmax}_{v \in \mathcal{V}}\, u(v)$
  if $u(v^*) > \tau$ **then**
    moveBase(v^*)
    $z_t \leftarrow$ getMeasurement()
    $M_{t+1} \leftarrow$ updateModel(M_t, z_t)
  end if
  $t \leftarrow t+1$
until $u(v^*) \leq \tau$

3D map and a hierarchy of maps can be created using our approach. In the experiments, we generated hierarchical models as introduced in Section 5.5.7.

We assume that the robot consists of a movable platform that is equipped with a 3D range sensor. The key idea of our exploration approach is to sample potential target locations in the environment and to evaluate their utility. For each target location, we estimate the cost of taking a measurement at that location and we estimate the expected information gain, which we define as the expected reduction of uncertainty in the map. The algorithm then selects the target location which provides the best trade-off between cost and expected information gain. The algorithm for this procedure is given as pseudo code in Algorithm 5. We will discuss the overall exploration algorithm first and then apply it to the problem of tabletop exploration in the following section.

The exploration is initialized by taking a 3D measurement at the starting location of the robot. From this measurement, we generate an initial model of the world M_1. After initialization, we execute an iterative process of generating, evaluating, and selecting possible targets. From the freespace encoded in the map M_t, *sampleViewpoints*(M_t) samples a set of target locations $\mathcal{V}$ that can be reached by the robot. Next, the potential view points are evaluated with respect to their cost and the expected reduction of uncertainty in M_t. While the cost of reaching a location depends on the current pose of the robot x_t, the reduction in uncertainty is governed by the potential measurements that we expect to obtain. The computation of this value is discussed in the following section. For each

potential view point $v \in \mathcal{V}$, we then determine a utility value that is defined as

$$u(v) = \alpha\, I(M_t; v) - (1-\alpha)\ \text{estimateCost}(M_t, x_t, v), \tag{5.24}$$

where $I(M_t; v)$ is the expected information gain depending on the current 3D map M_t, *estimateCost*(M_t, x_t, v) estimates the corresponding exploration cost, and α is a constant factor to trade off cost against gain which is determined heuristically.

Finally, we select the best next viewpoint v^* based on the utility of the sampled viewpoints:

$$v^* = \underset{v \in \mathcal{V}}{\text{argmax}}\ u(v). \tag{5.25}$$

The robot then moves to v^*, a 3D measurement z_t is taken and integrated into the 3D model. This loop is repeated as long as there are unexplored targets and their utility exceeds a user-defined threshold τ.

Estimated Information Gain

The expected information gain is defined as the expected reduction of the entropy H in the map and depends on potential observations that the robot would take at a given target location. We refer to (MacKay, 2003) for a detailed description of information-theoretic concepts. In general, the entropy of a discrete random variable X with possible outcomes $\{x_1, \ldots, x_n\}$ is defined as

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log P(x_i), \tag{5.26}$$

where $P(x_i)$ denotes the probability of outcome x_i. In the case of grid maps, we consider binary random variables that can have two outcomes: the corresponding grid cell is either *occupied* or *free*. Our sensor fusion approach provides us with an estimate $P(c)$ of a cell to be occupied. The entropy of a single grid cell c is therefore given by

$$H(c) = -P(c) \log P(c) + (1 - P(c)) \log(1 - P(c)) \tag{5.27}$$

and the entropy of a grid map M, consisting of n independent cells c_i, can be computed as

$$H(M) = -\sum_{i=1}^{n} \left[P(c_i) \log P(c_i) + (1 - P(c_i)) \log(1 - P(c_i)) \right]. \tag{5.28}$$

To compute the expected information gain of a target location in our exploration approach, we consider the entropy reduction $\Delta H(M_t; z)$ in the current model M_t caused by receiving

a specific measurement z which is defined as

$$\Delta H(M_t; z) = H(M_t) - H(M_t \mid z). \tag{5.29}$$

Since the measurement that will be obtained at the target location is not known beforehand, one has to integrate over all possible measurements. The expected information gain of a target location v is thus given by

$$I(M_t; v) = \int_z P(z \mid M_t, v)\, \Delta H(M_t; z)\, dz. \tag{5.30}$$

Integrating over all possible observations is infeasible but we can approximate Equation (5.30) by making the following assumptions. We can substantially simplify the computation by assuming that all measurements pass through free space cells in M_t, measure an obstacle when reaching an occupied cell, and are reflected with a uniform probability when traversing unmapped cells. We furthermore assume that only previously unknown cells contribute to a change of entropy. This is clearly not the case but the uncertainty reduction of such cells given a measurement z typically dominates $\Delta H(M_t; z)$. This can be seen from the definition of a cell's entropy in Equation (5.27). Observe that the entropy of a cell c reaches its maximum at $P(c) = 0.5$, that is, when the state of the cell is unknown.

Under these two assumptions, we can efficiently approximate Equation (5.30) so that it can be computed online—which is important for exploration tasks. We determine $\Delta H(M_t; z)$ directly by performing ray-casting operations at the potential view point. Given the assumptions mentioned above, the reduction in entropy only depends on the number of unknown cell encountered along the rays and on the change in entropy that is caused by measuring an unknown cell. We can derive this quantity directly from the sensor model.

Tabletop Exploration

We applied the approach presented above to the task of exploring a table top. More specifically, the task is to generate a hierarchical 3D model as defined in Section 5.3.4. We assume that the environment contains at least one table with some objects to be explored. We furthermore assume that a table can be detected in an initial measurement and that all relevant objects can be sufficiently measured by moving the robot around the table.

We follow the approach given in Algorithm 5. The viewpoint sampling method is summarized in Algorithm 6. From the initial model, we select the closest unexplored table as our area of interest. To generate exploration targets around the selected table, we first estimate the traversable area. Since the robot's base moves on the ground only, we

Algorithm 6 Tabletop Exploration Viewpoint Sampling

Input: M_t, current 3D map
r, sampling radius
Output: $\mathcal{V}$, set of viewpoints

$table \leftarrow$ findNearestUnexploredTable(M_t)
$m_{trav} \leftarrow$ computeTraversable(M_t)
$\mathcal{V} \leftarrow$ sampleTableViewpoints$(m_{trav}, table, r)$

project all obstacles from the 3D model onto a 2D traversability grid map m_{trav}. This is similar to the approach presented by Strand *et al.* (2008). We then sample a set $\mathcal{V}$ of robot poses from the traversable area around the selected table in a given radius. The radius is specific to the environment and the sensor, in our experiments we used a radius of 2 m.

To compute the cost of exploring a target we estimate the travel time from the robot's current pose to the target location. This can be done efficiently using the traversability map m_{trav} and a path-planning algorithm such as A^*.

5.5. Evaluation

The mapping approach presented in this chapter was evaluated using several real world datasets as well as simulated ones. The experiments were designed to verify that the proposed representation meets the requirements formulated in the introduction, more specifically, we demonstrate that our approach is able to adequately model various types of environments and that mapping is efficient. Furthermore, we evaluated our hierarchical mapping framework with respect to memory consumption, runtime, and mapping results in the presence of changes in the environment. In a simulated tabletop environment we evaluated the exploration system introduced in Section 5.4

5.5.1. Sensor Model for Laser Range Data

The datasets have been acquired using laser range finders such as the SICK LMS or the Hokuyo 30LX and we used stereo cameras with projected texture in the tabletop experiments. Our 3D map representation can be used with any kind of distance sensor, as long as an inverse sensor model can be learned. We employ a beam-based inverse sensor model which assumes that endpoints of a measurement correspond to obstacle surfaces and that the line of sight between sensor origin end endpoint does not contain any obstacles. The occupancy probability of all volumes is initialized to the uniform prior of $P(c) = 0.5$. To efficiently determine the map cells which need to be updated, a ray-casting operation is performed that determines voxels along a beam from the sensor

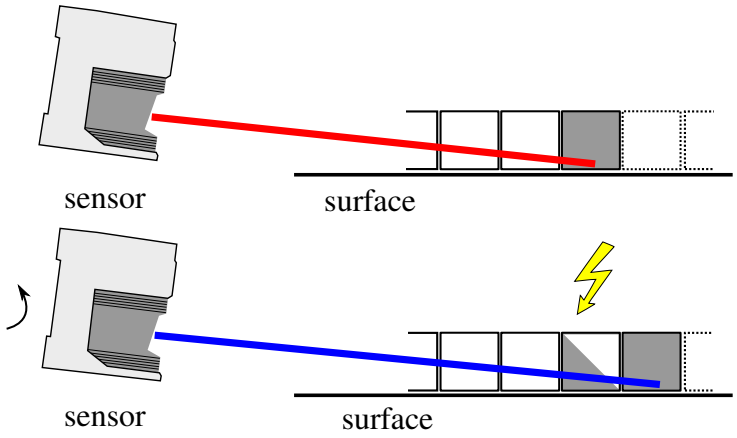


Figure 5.10: A laser scanner sweeps over a flat surface at a shallow angle by rotating. A cell measured occupied in the first scan (top) is updated as free in the following scan (bottom) after the sensor rotated. Occupied cells are visualized as gray boxes, free cells are visualized in white, measurement beams are shown in red and blue.

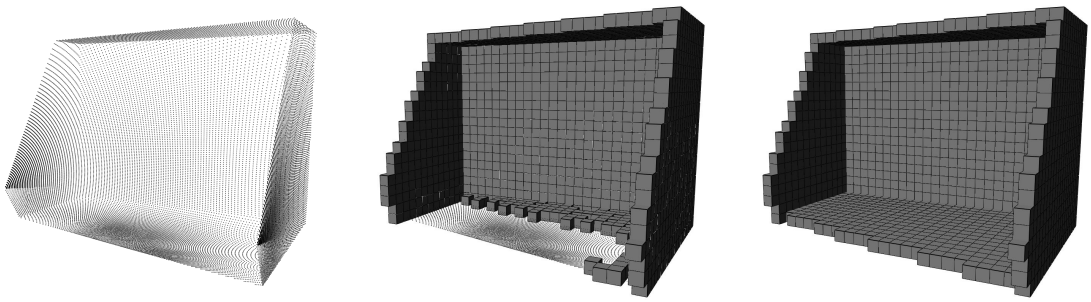


Figure 5.11: A simulated noise-free 3D laser scan (left) is integrated into our 3D map structure. Sensor sweeps at shallow angles lead to undesired discretization effects (center). By updating each volume at most once, the map correctly represents the environment (right). For clarity, only occupied cells and laser endpoints are shown.

origin to the measured endpoint. For efficiency, we use a 3D variant of the Bresenham algorithm to approximate the beam (Amanatides and Woo, 1987). Volumes along the beam are updated as described in Section 5.2.2 using the following inverse sensor model:

$$\mathrm{L}(c \mid z_t) = \begin{cases} l_{\mathrm{occ}} & \text{if beam is reflected within volume} \\ l_{\mathrm{free}} & \text{if beam traversed volume} \end{cases} \tag{5.31}$$

Throughout our experiments, we used logOdds values of $l_{\mathrm{occ}} = 0.85$ and $l_{\mathrm{free}} = -0.4$, corresponding to probabilities of 0.7 and 0.4 for occupied and free volumes, respectively. The clamping thresholds are set to $l_{\mathrm{min}} = -2$ and $l_{\mathrm{max}} = 3.5$, corresponding to the probabilities of 0.12 and 0.97. By lowering these thresholds, a stronger compression can be achieved but this obviously trades off map confidence for compactness.

Discretization effects of the ray-casting operation can lead to undesired results when using a sweeping laser range finder. During a sensor sweep over flat surfaces at shal-

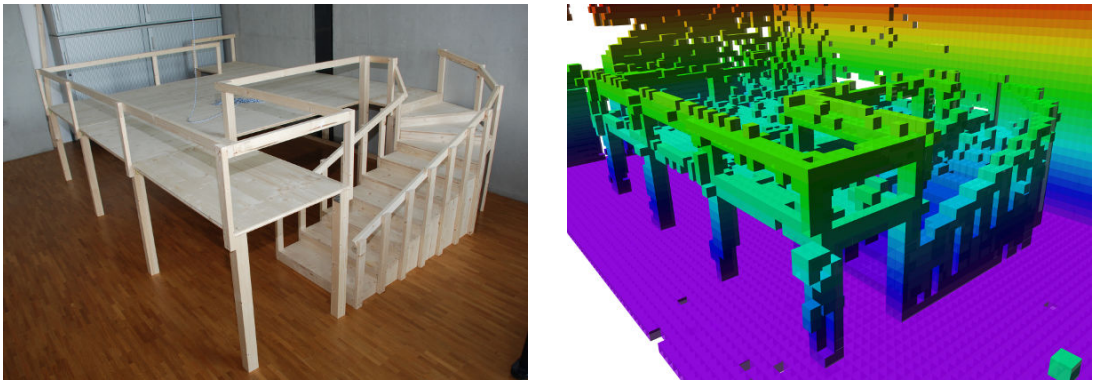


Figure 5.12: A small-scale indoor environment with two floors connected by a staircase. Left: Photo of the environment. Right: Visualization of a 3D map computed from sensor data.

low angles, volumes measured occupied in one 2D scan may be marked as free in the ray-casting of following scans. This effect is illustrated in Figure 5.10. Such undesired updates usually creates holes in the modeled surface, as shown in the example in Figure 5.11. To overcome this problem, we treat each sensor sweep as a single 3D measurement in our mapping approach. We update each voxel in the map at most once per 3D scan. Since measurements of laser scanners usually result from reflections at obstacle surfaces we make sure that the voxels corresponding to endpoints are updated as occupied. More precisely, whenever a voxel is updated as occupied according to Equation (5.31) it is not updated as free in the same measurement update of the map. By updating the map in this way, the described effect can be prevented.

5.5.2. 3D Models from Real Sensor Data

In this experiment, we demonstrate the ability of our approach to model environments using real sensor data. A variety of different datasets is used.

Two indoor datasets were recorded using a Pioneer2 AT platform equipped with a SICK LMS 291 laser range finder on a pan-tilt unit, the robot is shown in Figure 6.13 on page 165. Odometry errors were reduced using 3D scan matching. The first dataset was recorded in a small-scale indoor environment designed as a test environment for humanoid robots, see Figure 5.12. The environment features a staircase and two different levels. The data set consists of eleven 3D measurements recorded at different poses. Considerable interpolation noise of the laser scanner at sharp edges exists in the individual scans. The second dataset was recorded in building 079 at the Freiburg campus. This building features a long corridor that leads to a number of office rooms, a 2D map of the building is shown in Figure 2.7 on page 25. The robot traversed the length of the corridor three times and the resulting dataset consists of 66 scans. A further indoor data set

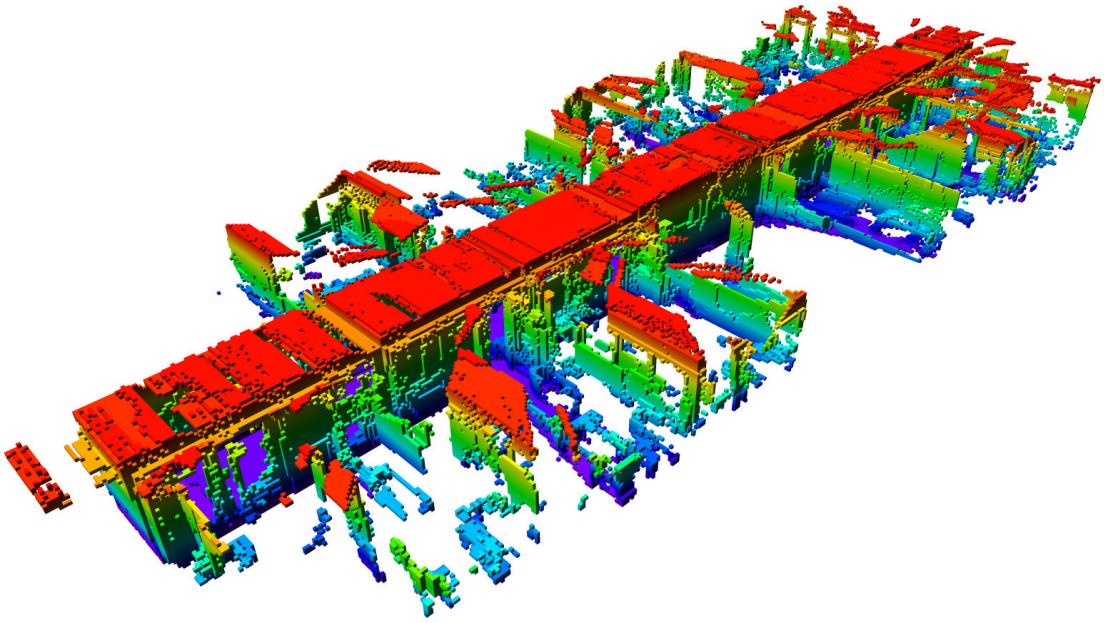


Figure 5.13: 3D map of the corridor of building 079 on the Freiburg campus, as seen from the top. A robot traversed the corridor of the building several times to collect 3D sensor data. Rooms have been partially observed through open doors and glass panes. Mapped area: 43.8 m × 18.2 m × 3.3 m.

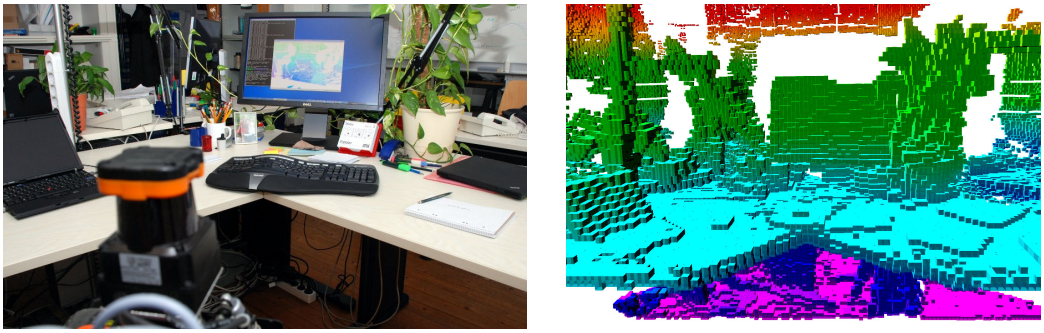


Figure 5.14: A tabletop (left) and a visualization of its 3D model (right).

Table 5.1: Mapping experiments

Experiment	Dataset	Mapped Area [m^3]	Resolution [m]
A	Humanoid environment	$3.5 \times 5.2 \times 1.7$	0.05
B	Freiburg building 079	$43.8 \times 18.2 \times 3.3$	0.05
C			0.1
D	Freiburg campus	$292 \times 167 \times 28$	0.20
E			0.80
F	New College (Epoch C)	$250 \times 161 \times 33$	0.20
G			0.80

was recorded using a Hokuyo 30LX laser range finder on a pan-tilt unit, see Figure 5.14. Here, the environment consists of a tabletop with several objects.

A fairly large outdoor dataset was recorded at the computer science campus in Freiburg (Steder and Kümmerle, 2010). It consists of 81 dense 3D scans and covers an area of 292 m $\times$ 167 m. In addition, we used laser range data of the New College data set (Smith et al., 2009). This data was recorded in a large-scale outdoor environment with two laser scanners sweeping to the left and right side of the robot as it advanced. For this dataset, an optimized estimate of the robot's trajectory generated by visual odometry was used (Sibley et al., 2009).

A visualization of the resulting models is shown in Figure 5.12 to Figure 5.15. Note that the free space is not visualized in the figures but is represented in the model. It can be seen, that our approach is able to model detailed indoor scenes as well as large outdoor environments. The sensor fusion approach presented in this chapter successfully integrated data from multiple measurements and sensor noise is handled probabilistically. Interpolation effects, which are especially pronounced in data of the SICK LMS 291, are mitigated by the probabilistic estimation: Veil points that occur in freespace do not lead to obstacle cells in the final 3D model.

5.5.3. Memory Consumption

In this experiment, we evaluated the memory consumption of our approach. Several datasets were processed at various mapping resolutions. An overview of the experiments is given in Table 5.1. We compared the memory usage of our representation to the amount of memory that a minimal 3D grid would require if it was stored linearly in memory. To evaluate the lossless compression method introduced in Section 5.2.4, we processed each dataset with and without map compression. Each map is furthermore written to disk using the binary format described in Section 5.2.4. The resulting file size is then compared to the size of files generated using a straightforward encoding that stores all voxel probabilities.

The results of the evaluation are given in Table 5.2. It can be seen that the compressed

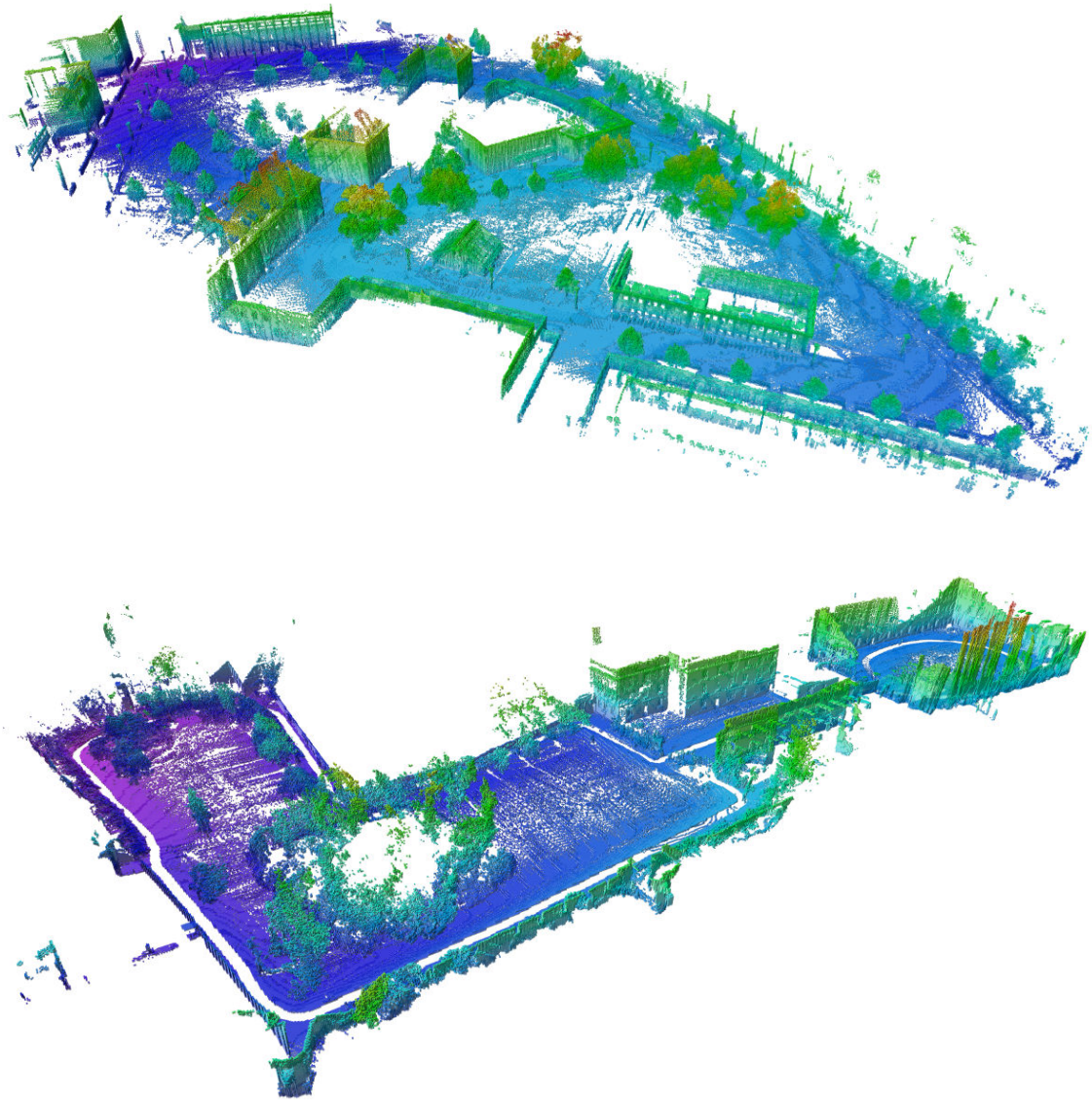


Figure 5.15: Visualization of octree maps of two outdoor environments at resolution of 0.2 m. Top: *Freiburg campus* (mapped area: 292 m × 167 m × 28 m), bottom: *New College dataset* (mapped area: 250 m × 161 m × 33 m). For clarity, only occupied volumes are shown with height visualized by a color coding.

Table 5.2: Memory consumption

Experiment	**Memory consumption [MB]**			**File size [MB]**	
	Full grid	No compression	Lossless compression	All data	Binary
A	1.03	1.91	1.38	0.54	0.02
B	80.54	73.64	41.70	15.80	0.67
C	10.42	10.90	7.25	2.71	0.14
D	654.42	188.09	130.39	49.75	2.00
E	10.96	4.56	4.13	1.53	0.08
F	637.48	91.43	50.70	18.71	0.99
G	10.21	2.35	1.81	0.64	0.05

octree data structure is significantly more compact than a 3D grid. High compression ratios can be achieved especially in large outdoor environments. Here, pruning will merge considerable amounts of free space volumes. On the other hand, the map structure is also able to model detailed indoor environments with moderate memory requirements. In very confined spaces, such as the environment we mapped in experiment A, an optimally aligned 3D grid may take less memory than an uncompressed octree. Note however, that the size of the environment has to be known in advance to achieve this result. In practice, this will rarely be the case and a larger 3D grid would therefore be initialized during mapping. Our approach, on the other hand, initializes only those voxels that are used during mapping and does not require the extend of the environment to be known. Furthermore, the difference in memory requirement is diminished as soon as compression techniques are used.

We analyzed the relation between memory usage and mapping resolution at the example of the Freiburg campus dataset. The memory requirement of maps at several different resolutions is shown in Figure 5.16 (top). Please note that a logarithmic scaling is used in the plot. It can be seen that memory usage increases exponentially with the tree resolution. This relation is expected and, in fact, any grid-based 3D map will exhibit the same property: If the minimal voxel size is halved, eight cubes are necessary to map the same volume that was previously modeled by one voxel.

For the 079 corridor dataset we analyzed the evolution of memory consumption as the map was created, see Figure 5.16 (bottom). The robot explored new areas up to measurement number 22 and then again from measurement number 39 to 44. In the remaining time, previously mapped areas were revisited and memory usage remained nearly constant during this time. A slight increase can still be noticed which is due to new information gathered by scanning from different viewpoints.

In Table 5.2, we additionally compare the file sizes of our maximum likelihood bitstream encoding, denoted as “Binary”, to the file sizes of the full probability encoding (“All data”). Map files generated using the proposed bitstream encoding are comparably small. For example, the 3D model of the Freiburg campus dataset, including free and unknown areas, requires 2 MB while its visualization as a 2D image file, shown in Figure 5.15, uses 816 kB. Note that map files can be compressed even further by using standard file compression methods.

5.5.4. Runtimes

In this experiment, we analyzed the time required to integrate range data into our 3D map structure. We processed the Freiburg building 079 dataset with a maximum sensor range of 10 m and the Freiburg campus dataset at maximum ranges of 50 m and 10 m. We measured the average time to insert 100,000 beams on an Intel® Core™ i7 950 3.07 GHz. The results for several mapping resolutions are given in Figure 5.17.

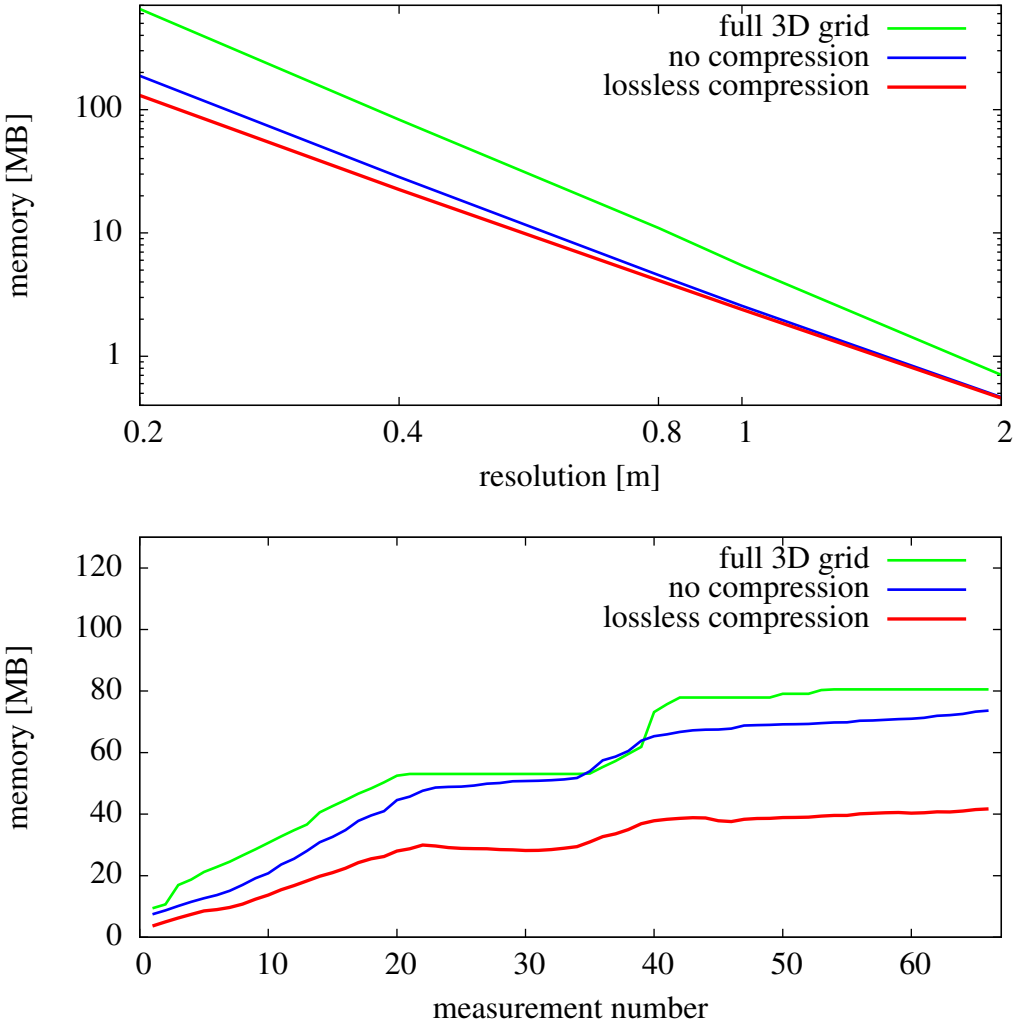


Figure 5.16: Top: relation of resolution and memory usage shown for the Freiburg campus dataset. Note that a logarithmic scaling is used. Bottom: Evolution of the memory usage while mapping the FR-079 corridor dataset at a resolution of 0.05 m.

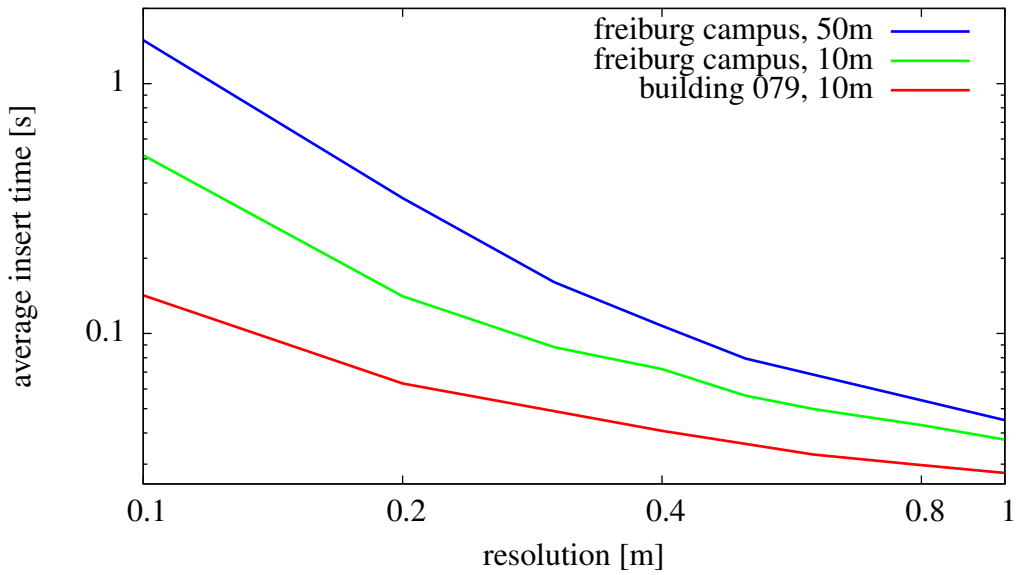


Figure 5.17: Average time to insert 100,000 data points of the Freiburg campus dataset with a maximum sensor range of 50 m, of the same dataset with a maximum range of 10 m, and of data from the building 079 dataset at a maximum range of 10 m. Note that a logarithmic scale is used.

The results show, that the insert time depends not only on the number of endpoints in a measurement, but also on the mapping resolution and on the range of the measurements. This is a consequence of the beam-based sensor model that we apply in the update: the longer the beam, the more cells need to be updated along the beam. The same is true for the map resolution: the higher the resolution, the more cells are hit along a measurement beam.

In our experiments, a single 3D scan usually consist of about 90,000 measurements and the time to acquire the data using a laser range finger on a pan-tilt unit is about 6 s. Typically, such a scan can be integrated into the map in less than one second, in indoor environments the update time will often be in the order of 0.1 s, for example for a maximum beam length of 10 m and at a mapping resolution of 0.1 m. Even with long measurement beams and high map resolutions, updating the map will not take longer than a few seconds on a standard desktop computer. For this reason, data can be integrated online and our approach is well suited for collision avoidance and autonomous exploration.

5.5.5. Hierarchical 3D Maps of Tabletops

In the next set of experiments, we evaluated the hierarchical map structure introduced in Section 5.3. The experiments are designed to show that maintaining a hierarchy of maps is more efficient than modeling the environment in a single global map. Furthermore, we show that our map hierarchy adapts to changes in the environment.

Figure 5.18: Experimental setup. A PR2 robot measures objects on a table using its stereo camera.

The first experiment compares the hierarchical map to a global monolithic map in the context of object modeling. The goal is to model objects on a table at a high resolution. We used a Willow Garage PR2 robot to acquire 3D measurements, the robot is equipped with a stereo camera and a texture projector to improve stereo quality. The experimental setup can be seen in Figure 5.18. Please note that the presented approach is not limited to stereo images and could be applied using any 3D range sensor such as a tilting laser scanner or similar devices.

Two sets of objects, shown in Figure 5.19, were scanned by sweeping the stereo camera across the table top scene taking 20 overlapping dense stereo images each. The objects depicted in Figure 5.19 b are considerably larger than those shown in Figure 5.19 a. The measurements were integrated into a 3D model as introduced in Section 5.3.4. In the hierarchical map, the resolution was adapted using the semantic labels of the map nodes. In all experiments, map nodes with the label `table` and `background` were mapped at a resolution of 10 mm and 50 mm respectively, these values are commonly used in navigation. Nodes with the label `object` were mapped at various different resolutions of 1 mm, 2 mm, 4 mm, 8 mm, and 16 mm. A visualization of the resulting models at an object resolution of 2 mm can be seen in Figure 5.19.

For comparison, all measurements were also integrated into a single monolithic octree map, using the approach presented in Section 5.2. In this case, segmentation was ignored and the map resolution was set to the resolution used for the `object` class.

Each mapping experiment was repeated five times on an Intel® Core™ i7-based desktop computer. A comparison of memory consumption and overall runtime can be seen in Figure 5.20 and Figure 5.21. Please note that a logarithmic scale is used in the plots showing the memory consumption. A mapping resolution of 1 mm could not be evaluated

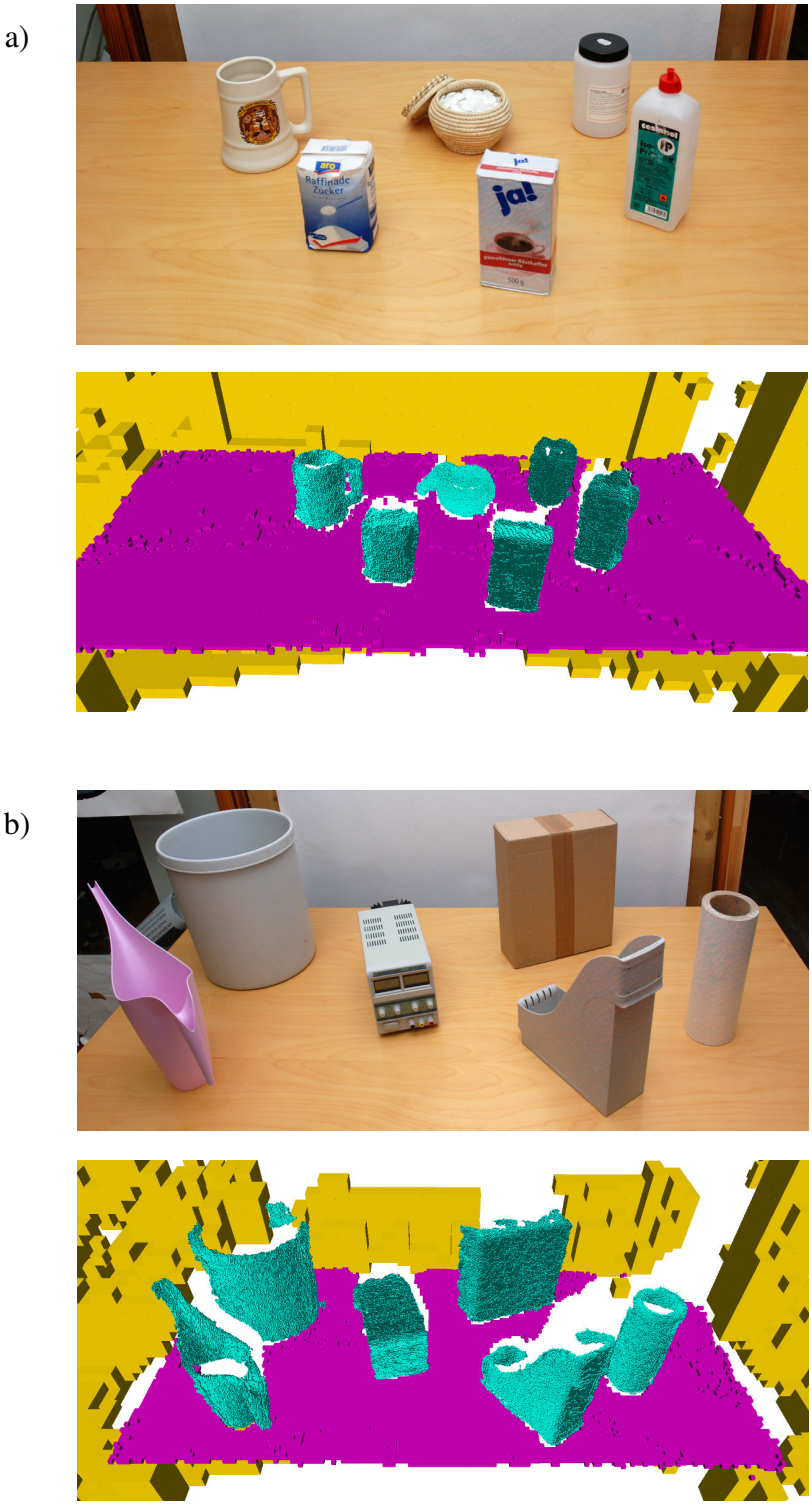


Figure 5.19: Photos of two tabletop scenes and visualizations of the resulting models. a: the *small objects* dataset. b: the *big objects* dataset. In the visualizations, objects are shown in cyan, the table is displayed in magenta and yellow voxels represent the background. An object mapping resolution of 2 mm is shown. Voxels with an occupancy probability of less than 0.5 are considered freespace and are not visualized.

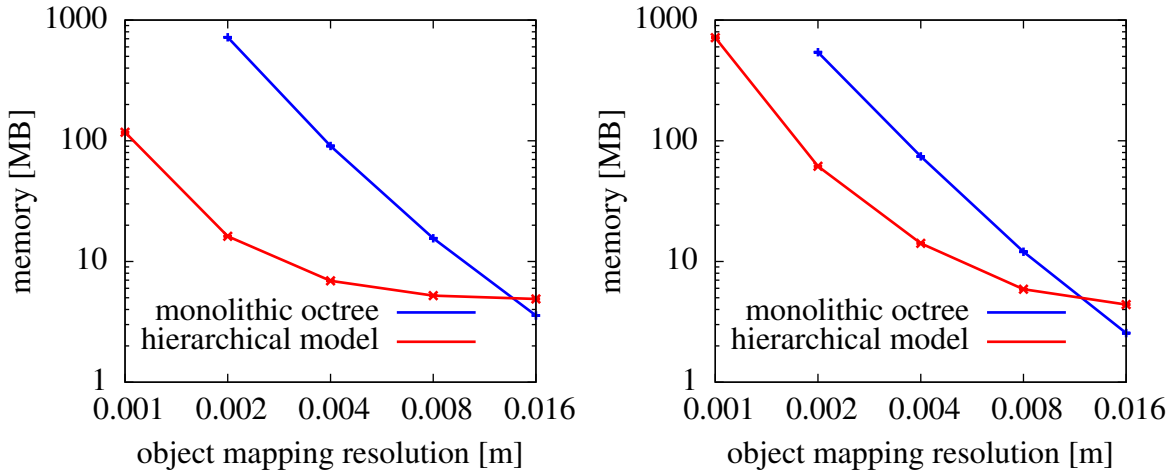


Figure 5.20: Memory usage of hierarchical tabletop models. Left: models of the *small objects* dataset, right: results using the *big objects* dataset. Note that a logarithmic scale is used in the plots. At an object resolution of 16 mm the monolithic map consumes slightly less memory since our implementation of the hierarchical map represents the table class at a fixed resolution of 10 mm.

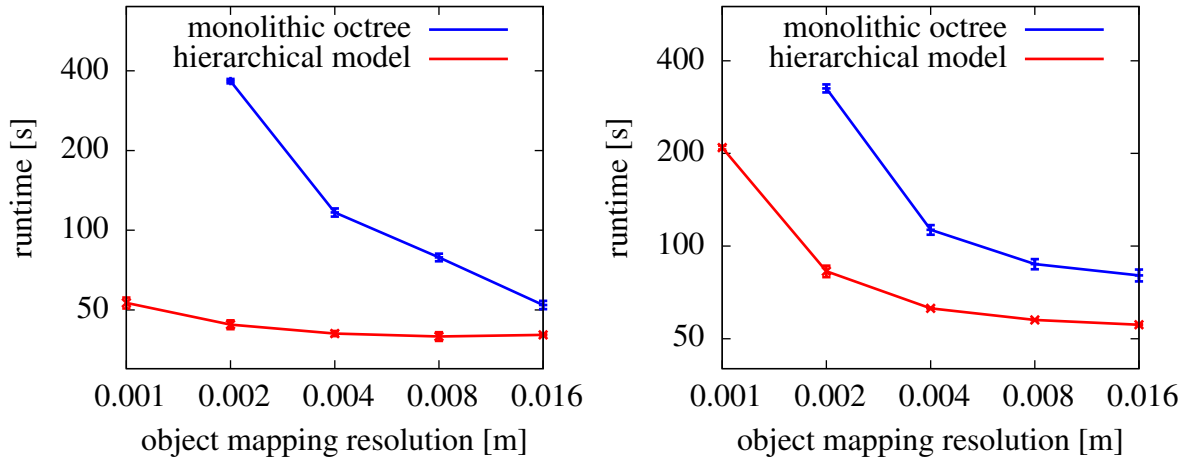


Figure 5.21: Runtime to integrate all measurements. Left: results using the *small objects* dataset, right: results using the *big objects* dataset.

using monolithic maps since they would have required in the order of 10 GB of memory which exceeded the capability of our computers. From the results it can be seen that the hierarchical model consumes significantly less memory and is updated significantly faster. At fine resolutions the monolithic map is about one order of magnitude bigger and it takes about one order of magnitude longer to compute the model compared to our hierarchical map.

This increase in efficiency is a direct consequence of the local adaptation of mapping parameters, in this case the mapping resolution. As we have shown in the experiment in Section 5.5.3 and Section 5.5.4, the memory and runtime requirements of a 3D map depend on the mapping resolution. While the size of the objects and therefore the mapped volume has a notable influence both on runtime and memory consumption, the resolution is dominates the complexity of the models. By adapting the resolution of each submap in the hierarchy, so that only the objects on top of the table are mapped at the target resolution, we can significantly increase the overall efficiency of the mapping approach.

5.5.6. Modeling Non-Static Environments

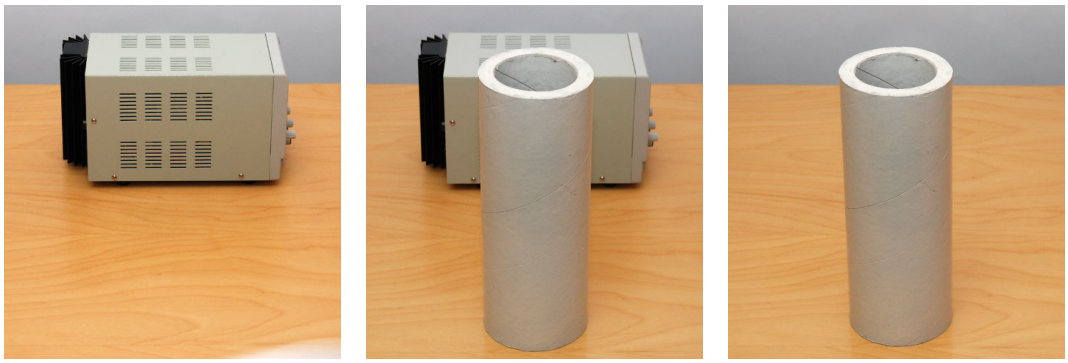
As in the previous experiment, the task is to model objects on a table. Unlike the previous environment, however, the environment was not static but objects are added and removed from the scene. This experiment is designed to show that our representation is able to adapt to changes in the environment. We applied our hierarchical mapping approach to generate a 3D map and compared it to the results obtained using a single map.

Measurements of two objects were taken in the sequence illustrated in Figure 5.22. The first object, a power supply, is measured and integrated into the map. Then, the second object, a tube, is placed in the scene so that it partially occludes the first object and then the scene is measured again. Finally, six measurements were taken after the power supply was removed from the scene. All measurements are integrated into a hierarchical model where the objects, the table and the background were mapped at a resolution of 2 mm, 10 mm, and 50 mm respectively. For comparison, the measurements were also integrated into a monolithic map at a resolution of 2 mm.

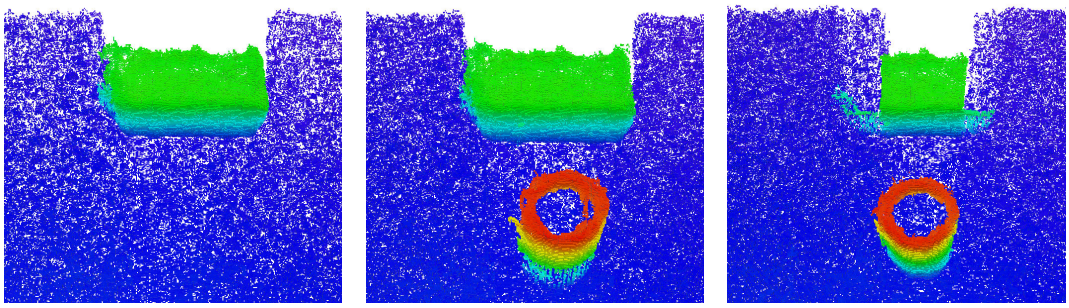
From the visualization of the resulting models shown in Figure 5.22, it can be seen that the monolithic map was unable to fully adapt to the changes in the environment. Both mapping approaches correctly modeled the environment as long as objects were added to the scene. However, the monolithic map was unable to update the cells representing the power supply once they were occluded by the tube. Since in this model all voxel occupancy probabilities are estimated independently, occluded cells remained occupied after the power supply was removed from the scene.

The hierarchical model, in contrast, was able to represent the environment correctly. The use of submaps in our model allowed us to remove the corresponding map node completely. By maintaining a probability of existence for each node according to Equa-

a) Photos of scene



b) Measurements integrated into monolithic map



c) Measurements integrated into hierarchical map

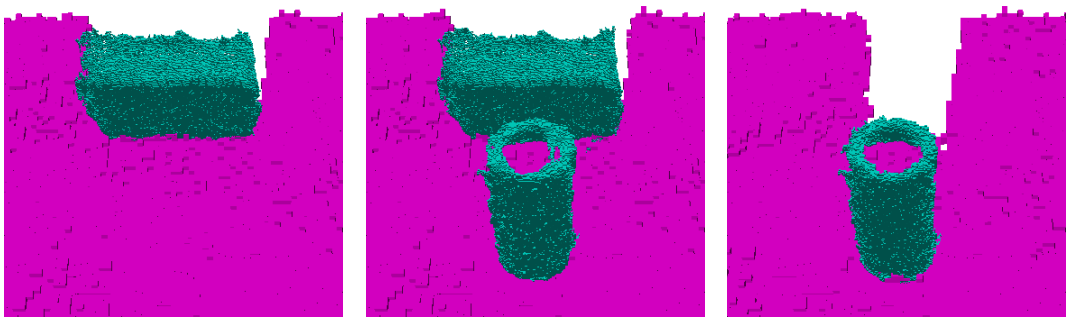


Figure 5.22: Experiment to evaluate mapping in non-static environments. a: Scenes measured in experiment 5.5.6, from left to right. b: Visualization of the mapping results obtained using a monolithic model at a resolution of 2 mm. Colors correspond to voxel height. c: Visualization of a hierarchical model of the scene with an object resolution of 2 mm.

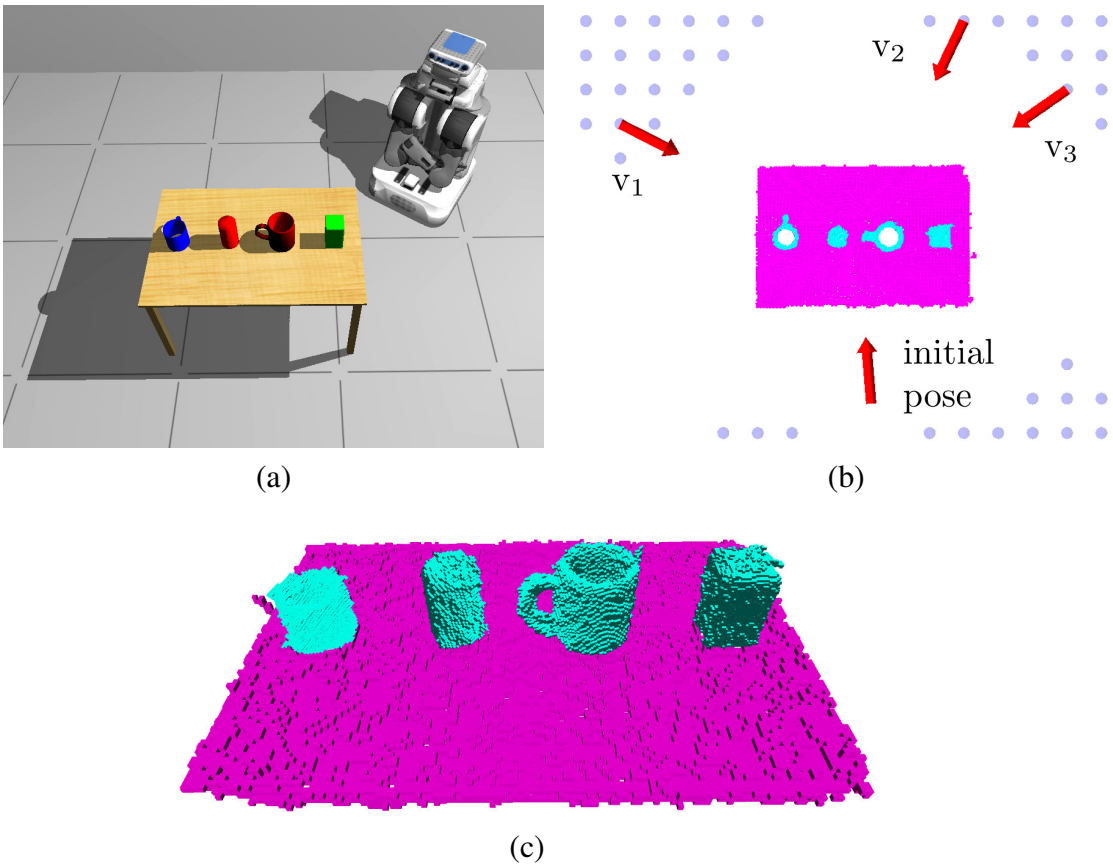


Figure 5.23: Tabletop exploration experiment. a) Simulated 3D environment. b) View points generated during exploration (discs) and chosen sensing locations (arrows). c) The final model generated autonomously. Objects are modeled at a resolution of 5 mm, the table top is modeled at 10 mm. Background voxels and free space voxels are omitted in the visualization in the visualization for clarity.

tion (5.19), we model the dependency of individual map cells within a map node probabilistically. This can be seen as a "common fate" assumption: Although the occupancy of each map cell is estimated individually, all cells are associated to the same map node. After the existence probability of the map node representing the power supply fell below a threshold of 0.5, it was removed from the model. This experiment shows that in non-static environments it can be an advantage to use a map structure that encodes hierarchical dependencies.

5.5.7. Object Exploration

To illustrate the tabletop exploration algorithm introduced in Section 5.4, we simulated a PR2 robot using the Gazebo 3D simulator. The environment contained a table with a number of objects, it is shown in Figure 5.23 a. During exploration, the robot maintained a hierarchical 3D map to represent the objects, the table, and the background structures.

An exemplary run of the system is depicted in Figure 5.23 b. After an initial measure-

ment, the table plane was detected. Then, potential viewpoints were generated from a traversability map and a series of view points v_i were chosen based on our information-based evaluation method. The resulting model is visualized in Figure 5.23 c. The robot successfully mapped the objects on the table. The only remaining unmapped space is located within the simulated cups and this volume could not be measured by moving the robot due to its sensor setup.

5.6. Related Work

Three-dimensional models of the environment are a key prerequisite for many robotic systems and consequently they have been the subject of research for more than two decades.

A popular approach to 3D mapping is to discretize the environment using a grid of cubic volumes. The state of these volumes is then estimated based on sensor measurements, Roth-Tabak and Jain (1989) as well as Moravec (1996) presented early works using such a representation. A major drawback of rigid grids is their large memory requirement. The grid map needs to be initialized so that it is at least as big as the bounding box of the mapped area, regardless of the actual distribution of map cells in the volume. The memory requirement of the grid becomes prohibitive in large-scale outdoor scenarios or when fine resolutions are needed. Furthermore, the extent of the mapped area needs to be known beforehand. In exploration tasks, for example, such knowledge cannot be assumed.

A discretization of the environment can be avoided by storing 3D range measurements directly. The occupied space in the environment is then modeled by the 3D point clouds returned by range sensors such as laser range finders or stereo cameras. This point cloud approach has been used in several 3D SLAM systems such as those presented by Cole *et al.* (2006) and Nüchter *et al.* (2007). The drawbacks of this method are that neither free space nor unknown areas are modeled and that sensor noise and dynamic objects cannot be dealt with directly. Thus, point clouds are only suitable for high precision sensors, in static environments, and when unknown areas do not need to be represented. Furthermore, the memory consumption of this representation increases with the number of measurements which is problematic as there is no upper bound.

If certain assumptions about the mapped area can be made, 2.5D maps are sufficient to model the environment. Typically, a 2D grid is used to store the measured height for each cell. In its most basic form, this results in an elevation map where the map stores exactly one height value per cell (Hebert et al., 1989). One approach in which such maps have been demonstrated to be sufficient is the outdoor terrain navigation method described by Hadsell *et al.* (2009). Whenever there is a single surface that the robot uses for navigation, an elevation map is sufficient to model the environment, since overhanging obstacles that are higher than the vehicle, such as trees, bridges or underpasses, can be safely ignored.

The strict assumption of a single surface can be relaxed by allowing multiple surfaces per cell, as proposed by Triebel *et al.* (2006), or by using classes of cells which correspond to different types of structure as presented by Gutmann *et al.* (2008). A general drawback of most 2.5D maps is the fact that they cannot store free or unknown areas in a volumetric way. A related approach was proposed by Ryde and Hu (2010). They store a list of occupied voxels for each cell in a 2D grid. Although this representation is volumetric it does not differentiate between free and unknown volumes either.

Octrees have been used in several previous approaches. They avoid one of the main shortcomings of grid structures: They delay the initialization of map volumes until measurements need to be integrated. In this way, the extent of the mapped environment does not need to be known beforehand and the map only contains volumes that have been measured. The use of octrees for mapping was originally proposed by Meagher *et al.* (1982). Early works mainly focused on modeling one Boolean property (Wilhelms and Van Gelder, 1992). Payeur *et al.* (1997) used octrees to adapt occupancy grid mapping from 2D to 3D and thereby introduced a probabilistic way of modeling occupied and free space. A similar approach was used by Fournier *et al.* (2007) and Pathak *et al.* (2007). In contrast to the approach presented in this chapter, however, the authors did not explicitly address the problems of map compression or bounded confidence in the map. An octree-based 3D map representation was also proposed by Fairfield *et al.* (2007). Their map structure called *Deferred Reference Counting Octree* is designed to allow for efficient map updates, especially in the context of particle filter SLAM. In contrast to our approach, lossless compression of trees is not described. Instead, a lossy maximum-likelihood compression is performed periodically. Furthermore, the problem of overconfident maps and multi-resolution queries are not addressed.

Yguel *et al.* (2007b) presented a 3D map based on Haar wavelets. This representation also allows for multi-resolution queries and it is probabilistic. However, the authors did not evaluate applications to 3D modeling in-depth. In their evaluation, unknown areas are not modeled and only a single simulated 3D dataset is used. Whether this map structure is as memory-efficient as octrees is hard to assess without further analysis.

Surfels (Habbecke and Kobbelt, 2007) have recently been used successfully in the context of object mapping (Weise et al., 2009; Krainin et al., 2010) but they only represent the surface of obstacles. Free space or unknown volumes cannot be represented and for this reason they are based on strong assumptions about the corresponding environment. In mobile manipulation scenarios, for example, knowledge about the free space is essential for safe navigation.

A number of previous approaches use a hierarchy or collection of maps instead of one global map. Popular methods represent objects based on collections of 2D grid maps (Anguelov et al., 2002; Galindo et al., 2005). Petrovskaya *et al.* (2007) represent movable objects in a 2D map at a high resolution while the background is represented at a coarse resolution. Douillard *et al.* (2010) combine a coarse elevation map for background

structures with object voxel maps at a higher resolution to perform 3D segmentation. In contrast to our hierarchical approach, they do not organize submaps in a hierarchy and they do not integrate multiple measurements into the model.

There exist several methods that consider semantic information in the context of 3D mapping. Nüchter *et al.* segment and classify 3D point clouds to improve scan registration (2006) and to detect objects (2008). Rusu *et al.* (2008) analyze segmented point clouds to derive functional properties of the environment in a household context. These approaches, however, are based on a point cloud representation, which has the disadvantages discussed above.

Our mapping approach assumes known sensor origins. To determine the pose of a 3D sensor, several techniques have been proposed. Typically, such methods align 3D measurements, detect loop closures, and perform global optimization of the sensor poses. To align 3D laser data, the iterated closest point (ICP) algorithm is a popular method and is the basis of the approaches of Cole *et al.* (2006) and Nüchter *et al.* (2007). Evaluations have shown that if precise 3D sensors are used then this techniques is able to determine sensor poses up to precision of a few millimeters (Müller et al., 2006). To align images of monocular cameras, stereo cameras, or depth cameras, features such as the Scale Invariant Feature Transform (SIFT) (Lowe, 2004) and its variants are usually extracted from the images. Matching features are then aligned over consecutive frames. An approach to extract 3D information from monocular cameras was presented by Davison *et al.* (2003). A 3D SLAM approach that uses SIFT to align 3D measurements of depth cameras was presented by Henry *et al.* (2010).

Several previous approaches addressed the problem of creating 3D maps autonomously. As one of the first, Whaite and Ferrie (1997) have presented an approach to map objects in 3D by reducing the uncertainty of the map. In contrast to the method presented in this chapter, they maintain a parametric representation of object surfaces. An approach to create 3D maps of buildings was presented by Surmann *et al.* (2003). To determine sensing locations they apply a variant of the frontiers approach in several horizontal slices of the 3D map. Similar to our approach, they evaluate viewpoints based on the expected information gain. However, they use 2D ray-casting to estimate this quantity while our approach uses 3D ray-casting in a volumetric representation. Joho *et al.* (2007) introduced a similar system that performs 3D ray-casting but is based on a 2.5D representation. Determining the next best viewpoint is related to the problem of active object recognition. Here, a number of known objects have to be localized in the environment. An overview of solutions to this problem was given by Dutta *et al.* (2004).

5.7. Discussion

In this chapter, we presented a technique to model 3D environments efficiently. Three-dimensional maps are relevant for several robotic tasks including navigation, exploration, and mobile manipulation. Our mapping approach meets three goals: It models the environment probabilistically, it represents unmapped areas, and it is efficient both in runtime and in its memory requirement. We use probabilistic occupancy estimation to integrate sensor measurements and to represent free space as well as unknown areas. Map data is maintained in octrees which facilitates multi-resolution map queries and leads to a compact memory representation. As as key contribution, the proposed approach uses a bounded per-volume confidence. In this way, the map can adapt to changing environments quickly and the bounded confidence allows for a lossless compression scheme that substantially reduces memory usage. In our evaluation, our compression method reduced the memory requirement by up to 45% compared to an uncompressed map. In outdoor scenarios, a 3D occupancy grid map took more than ten times the amount of memory that was required by our approach.

Furthermore, we extended our mapping approach to a hierarchy of octree maps. Most previous approaches maintain a single map that encodes the complete environment. Our hierarchical approach, in contrast, maintains a tree of independent 3D submaps that is based on a user-defined spatial relation. In this way, the hierarchy is able to encode dependencies in the environment, for example, objects that are supported by planes. Compared to a single global map, our hierarchy of maps offers two main advantages. First, the mapping parameters such as the resolution can be adapted for each submap. Second, submaps can be manipulated independently. For example, one can represent movable objects in submaps and then transform these object maps independently from the scene background. In addition to the formal description of the map hierarchy, we presented a specific implementation for tabletop manipulation tasks. In this setting, objects on a table are mapped at very fine resolutions while the table and background structures were mapped at lower resolutions. This approach led to models that were about an order of magnitude more compact than a single map that represents the complete scene.

Finally, we presented an information-driven exploration algorithm that generates 3D models autonomously. To determine and evaluate potential sensing locations, the approach exploits the knowledge about unmapped areas that is encoded in our 3D representation. Our approach iteratively selects measurement locations to reduce the uncertainty in the model. This application demonstrates that our mapping approach is well suited for autonomous mapping systems.

We evaluated our mapping approach using both real-world and simulated data. The results demonstrate that our approach is able to model the environment in an accurate and efficient way. Our evaluation also shows that in a tabletop scenario a hierarchy of maps consumes significantly less memory than a single 3D map and that the hierarchy

can also be updated significantly faster. Moreover, we showed that the hierarchy is able to adapt to changing environments by removing outdated map nodes from the hierarchy. In a simulated experiment we furthermore showed that a map hierarchy can be generated autonomously using our information-driven exploration approach.

5.7.1. Open Source Implementation

The approach presented in Section 5.2 has been made available as a self-contained C++ library under the BSD open source license. The library, called OctoMap, can be obtained from `http://octomap.sf.net`. It received a considerable uptake from the scientific community and has been used in a variety of applications, among others these include 6D localization (Hornung et al., 2010), autonomous navigation with air vehicles (Heng et al., 2011; Müller et al., 2011), autonomous navigation with humanoid robots (Oßwald et al., 2011), 3D exploration (Shade and Newman, 2011; Dornhege and Kleiner, 2011), 3D SLAM (Hertzberg et al., 2011), 3D arm navigation (Ciocarlie et al., 2010), semantic mapping (Blodow et al., 2011), and it has been used as a benchmark to develop further 3D mapping techniques (Dryanovski et al., 2010; Stoyanov et al., 2011; Huhle et al., 2011).

Chapter 6
Identifying Vegetation from Laser Data

6.1. Introduction

In Chapter 2, we presented a technique to coordinate teams of robots based on a segmentation of the environment. We mentioned that a segmentation can also be obtained by considering traversability information: Areas that are traversable only by certain types of robots form segments in outdoor environments. Such a segmentation can then be used to coordinate heterogeneous teams of exploring robots. In this chapter, we will present an approach to reliably detect vegetation from laser data.

Most vehicles that navigate outdoors, such as transportation vehicles, autonomous wheelchairs, surveillance robots, or autonomous cars, have been designed to drive on streets and paved paths. Often such vehicles rely on curbs or similar geometric features to detect the boundary of the drivable surface. In case these features are not available or the vehicle fails to detect them, it runs into the risk of leaving the road. In many urban scenarios, this leads to a situation where the vehicle traverses areas that contain vegetation such as grass. Traversing vegetation increases the risk that the vehicle gets stuck or that the wheel slippage is increased which leads to errors in the location estimation process. Both of these cases pose a danger to the safety of the system.

This danger can be greatly reduced using methods to classify the surface terrain. By predicting the terrain type, these methods enable the robot to detect drivable areas reliably and therefore to plan safe trajectories. Terrain classification is a well studied research area in the field of outdoor navigation. The goal of these approaches is to detect the type of terrain using the sensors of the robot. Previous approaches used laser range scanners to analyze the roughness of the environment (Macedo et al., 2001; Hebert and

Vandapel, 2003; Wolf et al., 2005; Lalonde et al., 2006) or a combination of cameras and laser scanners (Manduchi et al., 2003; Wellington et al., 2006; Bradley et al., 2007; Douillard et al., 2008). Additionally, several authors have investigated vibration-based classification (Sadhukhan et al., 2004; Brooks et al., 2005; DuPont et al., 2005; Weiss et al., 2006). These previous approaches, however, suffer from a number of drawbacks. Camera-based systems, for instance, are sensitive to changes in the lighting conditions and, in particular, cannot be used in the dark. Vibration-based system can reliably detect the type of terrain that is currently being traversed but they cannot predict the terrain type in front of the robot. Previous laser-based systems are able to detect vegetation such as trees and bushes but fail to reliably detect flat vegetation such as freshly mowed lawns. In this chapter, we introduce a novel laser-based terrain classification approach that is especially suited for detecting low vegetation.

Most outdoor robots are equipped with laser range finders. When laser range finders measure an obstacle, they return the distance to this obstacle. Low vegetation poses a challenge to terrain classification methods that depend solely on these measurements. The variance in the measured distances is often small and thus it is hard to differentiate flat vegetation from other flat surfaces. However, most sensors also report the reflectivity of the object surface. To measure surfaces, laser scanners emit near-infrared light. The surface reflectivity reported by the sensors thus also expresses reflectivity in the near-IR spectrum. From satellite image analysis it is known that near-IR light is strongly reflected by chlorophyll, which is found in living plants (Myneni et al., 1995). In our approach, we exploit this effect to detect vegetation in laser scans of the robot's surroundings. Figure 6.1 shows an example of the classification result using only distances measurements (middle) and using our approach based on laser reflectivity (bottom).

The contribution of this chapter is an approach to reliably detect flat vegetation. We formulate the detection of vegetation as a classification problem. As inputs to this classification problem we use individual laser measurements consisting of laser reflectivity, the measured distances, and the incidence angles. In contrast to previous approaches, we take into account the dependency of the distance and the incidence angle on the laser remission value. We apply a supervised learning method to train a classifier from example measurements. We also provide a way to gather such training data in a self-supervised way and thus eliminate the need for manual labeling of training examples.

In the evaluation, we show that our approach can be used to accurately map vegetated areas and do so with a higher accuracy than standard techniques which are based solely on range values. In our mapping experiments, we evaluate two popular sensor setups: rotating laser scanners capturing 3D point clouds and laser scanners mounted at a fixed, downward-looking angle. We furthermore demonstrate that our laser-based classifier can be trained in a self-supervised way using a vibration-based classification approach to label training data. Additionally, we present experiments which show that by applying

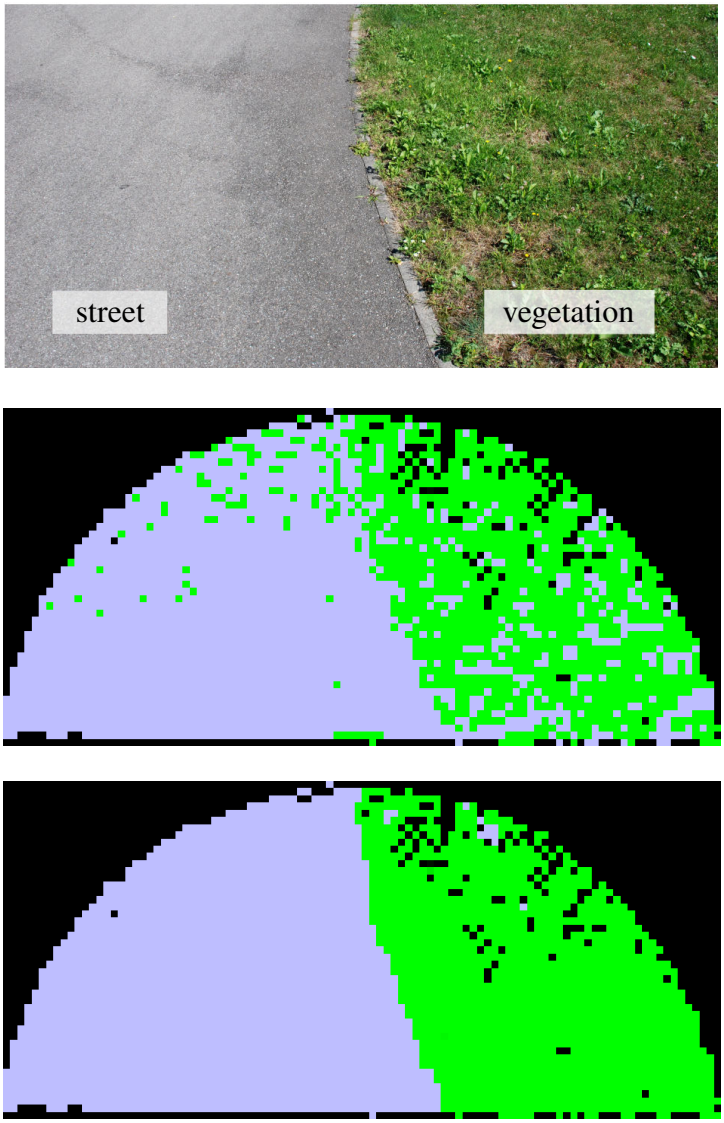


Figure 6.1: Example of a street and an area containing grass (top) and typical classification results obtained using an approach based on range differences (middle) and our approach based on laser reflectivity (bottom). Shown is a bird's eye view of a 3D scan of the area depicted at the top with a maximum range of 5 m. Whereas points classified as street are depicted in blue, points corresponding to vegetation are shown in green.

our approach robots that have not been designed to drive on vegetation are able to safely navigate in structured outdoor environments such as parks or campus sites.

This chapter is organized as follows. After describing support vector machines, which we use for classification, we present our approach to terrain classification using laser reflectivity. In Section 6.4, we describe how training data can be gathered autonomously using the output of a vibration-based terrain classifier. Section 6.5 describes the mapping approach we use to integrate multiple measurements of the same region. We provide a comparison of different sensors in Section 6.6. An alternative classification method for resource-constrained systems is presented in Section 6.8. In Section 6.9, we present the experimental results obtained with real robots navigating through an university campus and neighboring areas. Finally, related work in the field of terrain classification is discussed in Section 6.10.

6.2. Support Vector Machines

To classify sensor measurements, we use a support vector machine (SVM). SVMs are widely used for a broad range of regression and classification problems. We define a classification problem as the task of assigning one of two possible discrete class labels to an input $x \in \mathbb{R}^n$ using a decision function d:

$$y = d(x),\ y \in \{-1,+1\} \tag{6.1}$$

The SVM is a supervised method that learns a decision function based on a labeled set of training data $\mathcal{D} = \{(x_i, y_i)\}$, where a target label $y_i \in \{-1,+1\}$ is given for each example input x_i. Given this training data, the SVM determines a hyperplane defined by

$$f(x) \quad = \quad \langle w,x\rangle + b = 0, \tag{6.2}$$

where w is the normal vector of the plane and b denotes the offset of the hyperplane from the origin. To compute the class label y of a new input vector x, the SVM then uses the following decision function:

$$f(x) \quad = \quad \langle w,x\rangle + b \tag{6.3}$$

$$y \quad = \quad \mathrm{sgn}(f(x)) \tag{6.4}$$

An illustration of a separating hyperplane is given in Figure 6.2. To determine a hyperplane that separates the training data $\mathcal{D}$, we have to choose the parameters w and b such

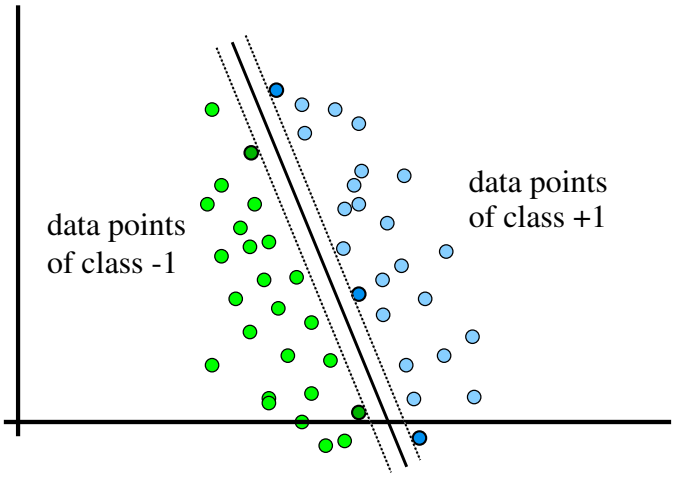


Figure 6.2: Data of two classes, shown as green and blue circles, are separated by a hyperplane. The hyperplane is depicted as a solid line and the dashed line illustrate its margin to the data. Support vectors are shown as darker circles on the margin.

that

$$\langle w, x_i \rangle + b \geq +1, \; \forall (x_i, y_i) \in \mathcal{D}, \; y_i = +1 \tag{6.5}$$

$$\langle w, x_i \rangle + b \leq -1, \; \forall (x_i, y_i) \in \mathcal{D}, \; y_i = -1, \tag{6.6}$$

which can be rewritten as

$$y_i \; \langle w, x_i \rangle + b \leq +1, \; \forall (x_i, y_i) \in \mathcal{D}. \tag{6.7}$$

If we assume that the training data is linearly separable, then there is an infinite number of parameters w and b that satisfy this constraint. The SVM computes the hyperplane that best separates the training data by maximizing the margin of the hyperplane. The margin is defined as the minimum distance to the hyperplane of any training input. The distance of an input x_i is given by

$$\frac{y_t \langle w, x_i \rangle + b}{||w||}. \tag{6.8}$$

Note that minimizing $||w||$ in the above formula maximizes the distance. For this reason, the optimal separating hyperplane that maximizes the margin can be constructed by solving the constraint optimization problem defined by

$$\min \frac{1}{2} ||w||^2 \tag{6.9}$$

$$\text{subject to } \; y_i \; (\langle w, x_i \rangle + b) \geq 1, \; \forall (x_i, y_i) \in \mathcal{D}.$$

This can be alternatively expressed using Lagrange multipliers α_i as

$$L_p = \frac{1}{2}||w||^2 - \sum_i \alpha_i [y_i \left(\langle w, x_i \rangle + b\right) - 1], \; \alpha_i \geq 0, \tag{6.10}$$

where we minimize the system with respect to w and b and maximize with respect to the α_i. There exist efficient methods to solve this problem using quadratic programming. The solution can be expressed as a weighted sum of the training vectors:

$$w = \sum_i \alpha_i y_i x_i \tag{6.11}$$

It can be observed that only a small number of the α_i are non-zero and that therefore only a small number of the x_i are relevant to the solution. These inputs x_i are called the support vectors. They lie on the margin of the separating hyperplane and satisfy

$$y_i \left(\langle w, x_i \rangle + b\right) = 1. \tag{6.12}$$

Finally, the offset of the hyperplane can be determined using any of the support vectors as:

$$b = y_i - \langle w, x_i \rangle. \tag{6.13}$$

So far, we assumed that the input data can be separated linearly. To separate classes that cannot be separated linearly, the SVM applies the so-called kernel trick: The training data is first mapped into a higher-dimensional feature space using a kernel function. A separating hyperplane is then constructed in this features space which yields a nonlinear decision boundary in the input space. In practice, the Gaussian radial basis function (RBF) is often used as a kernel function. It is given by

$$k(x, x') = e^{\frac{-(x-x')^2}{2l^2}}, \tag{6.14}$$

where l is the so-called length-scale parameter and defines the global smoothness of the function.

There exist variations of the basic SVM-formulation that are robust against occasional mislabeled training points. Among those, C-SVM is a popular method. It defines a soft margin, that is, an additional parameter commonly denoted as C adjusts the trade-off between maximizing the margin and minimizing the training error. A good overview of machine learning techniques, including support vector machines and their variations, can be found in (Schölkopf and Smola, 2002; Alpaydin, 2004; Theodoridis and Koutroumbas, 2006).

6.2.1. Class Probabilities

The decision function of support vector machines, as given in Equation (6.4), yields a class label y which is not a probability: y is either -1 or 1. In many cases, however, it is desirable to know the uncertainty associated with the output of a classifier. One of these cases is the terrain classification approach discussed in this chapter, where sensor measurements are uncertain and this uncertainty is propagated to the classification results. Knowing the uncertainty associated with classification results enables us to use probabilistic model estimation methods as we will discuss in Section 6.5.

Several methods have been proposed to map the output of SVMs to posterior class probabilities (Platt, 1999; Vapnik, 1998). Among those, Platt's method (Platt, 1999) is a popular approach. It fits the posterior using a parametric model. The model is given by

$$P(y = 1 \mid f) = (1 + \exp(Af + B))^{-1}, \tag{6.15}$$

where f is the output of the function given in Equation (6.3). The parameters A and B are determined using maximum likelihood estimation from a training set $\{(f_i, y_i)\}$ consisting of outputs f_i and the corresponding class labels y_i. For details on the optimization step, we refer the reader to the work by Platt (1999).

6.3. Terrain Classification Using Remission Values

The goal of our approach is to precisely identify those parts of the environment that contain vegetation. It enables a robot to avoid vegetation by providing a classification of terrain types in the surrounding of the robot. This prediction of terrain type around the robot is achieved by measuring the surface using a forward-looking or rotating laser scanner. Typical laser range fingers, such as the SICK LMS scanners or the Hokuyo UTM, emit light at a wavelength of 870 nm to 915 nm, which is within the near-infrared range. Near-infrared light is strongly reflected by chlorophyll, which is found in living plants (Myneni et al., 1995). We exploit this effect to identify vegetation in measurements of the robot's surrounding. Compared to an approach based purely on the geometric scan point distribution, a far better classification accuracy can be achieved when we consider the reflectivity of laser measurements. We will provide a comparison of both approaches in Section 6.9.1.

Laser scanners do not provide a measurement of the surface reflectivity directly. Instead, they return a so called remission value that corresponds to the power received by the sensor relative to the power transmitted by the laser. However, the remission value returned by the scanner does not only depend on the material of the measured surface, but also on the distance at which it is hit. This can be seen from Figure 6.3, which shows the relation between the measured range and the remission value for both vegeta-

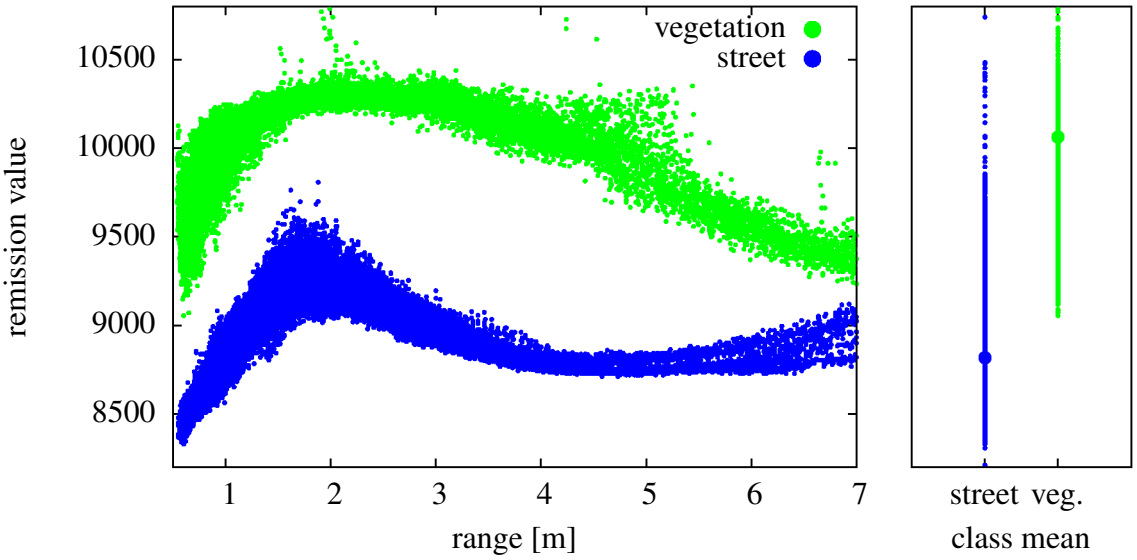


Figure 6.3: Left: typical remission measurements of a SICK LMS 291-S05 for street (blue) and vegetation class (green). Right: mean remission values for both classes.

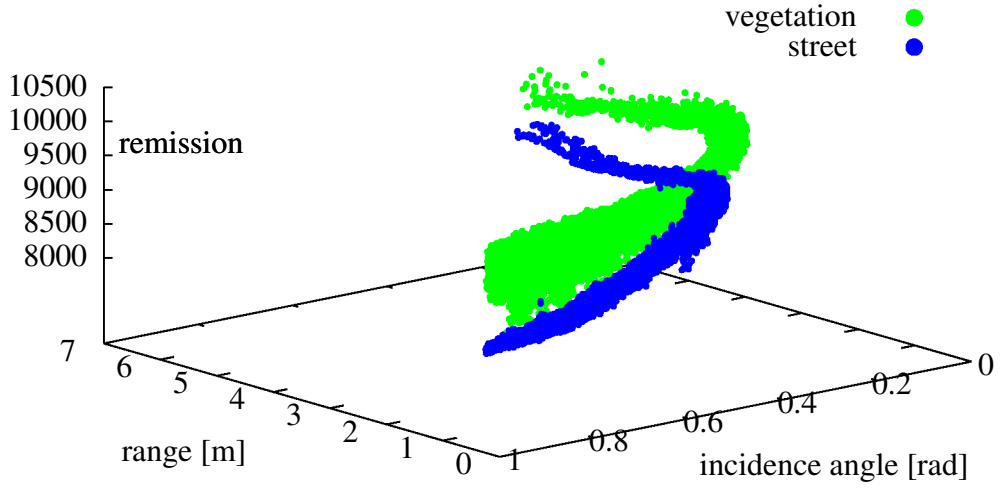


Figure 6.4: Visualization of sensor data in the 3D feature space. The plot shows data recorded using a SICK LMS 291-S05.

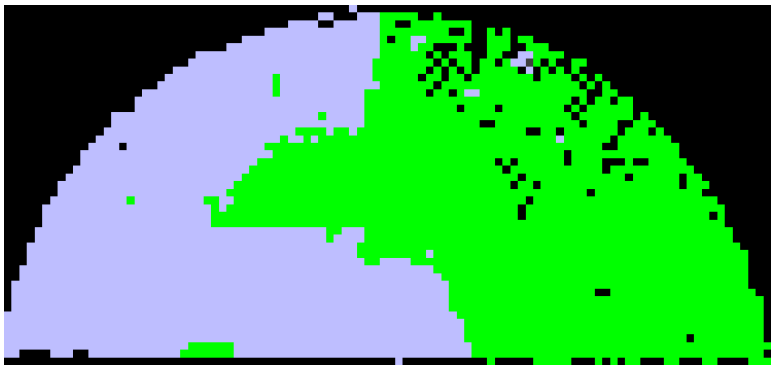


Figure 6.5: Classification of the scene depicted in Figure 6.1 using a naive classifier that uses the mean remission value (see Figure 6.3, right) to distinguish both classes.

tion and non-vegetation. There is an additional dependency of the remission value on the angle of incidence (Baribeau et al., 1992). This can be seen in the visualization of the three-dimensional feature space consisting of remission value, measured distance, and incidence angle which is shown in Figure 6.4.

While the range and remission of each laser measurement is accessible directly, the incidence angles need to be estimated. This estimation, however, can be done in a straightforward way: Given a 3D scan, one can group neighboring endpoints and compute the eigenvalues of the covariance matrix obtained from these points. The direction of the eigenvector that corresponds to the smallest eigenvalue is a good estimate of the surface normal. From the surface normal, the known mounting height of the sensor, as well as the angle of the laser beam relative to the robot, the incidence angle can be estimated.

The importance of considering all three features in the classification process is illustrated by the following experiment. We implemented a naive classifier, where measurements are classified based solely on the remission value. We computed the average remission value over all measured ranges and angles for both classes in a set of example measurements (see Figure 6.3). To classify new measurements, we compute the distance of the corresponding remission value to those averages and chose the class with minimal distance. The results obtained by this approach is visualized in Figure 6.5. Here, the scene from Figure 6.1 was classified using only the mean remission value for both classes ignoring the range and incidence angle. It can be seen that this approach generates a large number of false positive vegetation detections. These misclassifications occur especially in the near range of up to 3 m that is critical for robot navigation.

It is worth mentioning that the measured difference in reflectivity between vegetation and other material is not due to the lighter appearance of grass compared to, for example, an asphalt street. In Figure 6.6, a SICK LMS 151 sensor is used to compare remission data of an asphalt street and of grass to those of a surface covered with light tiles. These tiles exhibit a visual brightness similar to grass. It can be seen that the measured infrared reflectivity of the tiles is indeed higher than those of the asphalt street but it is

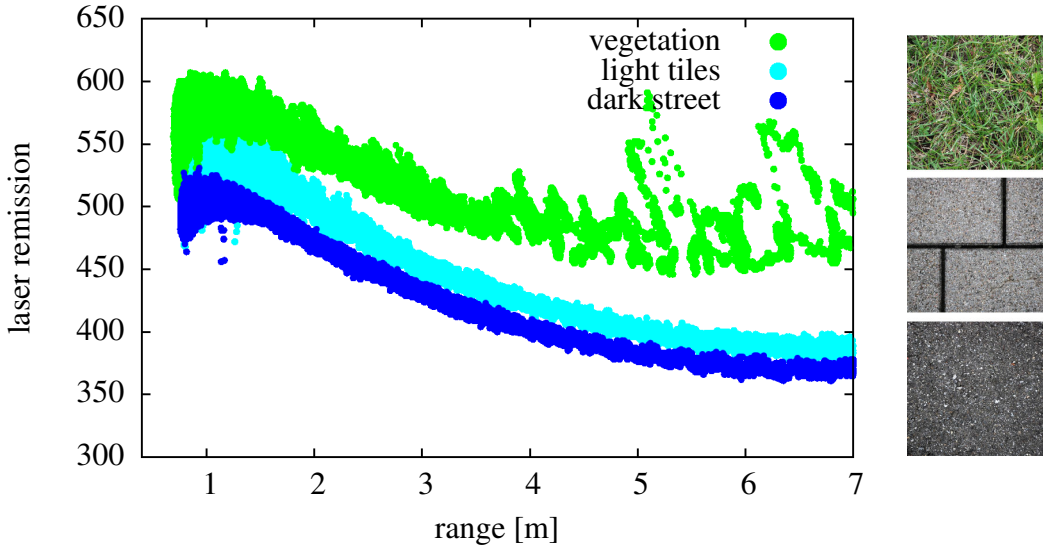


Figure 6.6: Left: remission values of a SICK LMS 151 measuring vegetation, light tiles, and an asphalt street. Right: examples of the measured surfaces.

still considerably lower than measurements of vegetation. For this reason it is possible to differentiate these kinds of surface from vegetation using our approach.

6.3.1. Robust Terrain Classification Using an SVM

We detect flat vegetation in laser measurements using a support vector machine. Each measurement consists of the measured range, an estimated incidence angle, and the measured surface reflectivity. These three features define the feature space of the classification problem. We apply a C-SVM using an RBF-kernel as introduced in Section 6.2.

The input to the learning algorithm is a labeled set of laser measurements. Each data point consists of a range measurement, incidence angle, and a remission value. By determining the separating hyperplane between both classes in the training set, we implicitly model the reflectivity characteristic for the street and vegetation class. There are two parameters of the SVM that have to be chosen appropriately, the length-scale parameter l of the kernel as well as the soft-margin parameter C. To optimize the parameters, we perform a systematic grid search on the grid defined by $\log_2 l \in \{-15,...,3\}$ and $\log_2 C \in \{-5,...,15\}$. The training is done offline and the optimal parameters are determined using 5-fold cross validation. Once the classifier has been trained, the model can be used on any robot which is equipped with the same sensor in a similar configuration. More specifically, the laser should be mounted so that it measures the surface at angles and distances similar to those observed during the training phase.

6.4. Training Data from Vibration-Based Terrain Classification

The SVM-classifier relies on a labeled set of example measurements during training. Our approach avoids tedious hand-labeling of dense three-dimensional data by using a self-supervised training method. To generate training data for our laser-based classifier, we utilize the output of a second, reliable and pre-trained classifier. This second classifier uses a different sensor: It measures the vibration induced in the body of the robot while it is traversing the surfaces. In this way, the terrain type directly underneath the robot is determined. As the robot is moving through the environment, it generates a 3D model using the method described in Section 6.5. To acquire the training set, each individual 3D scan point in the measurements of the robot is stored along with the corresponding remission value, incidence angle, and range value. During the acquisition of the training set the robot is manually steered so that it passes previously scanned surfaces. The previous measurements associated with these surfaces are then labeled according to the output of the vibration-based classifier, which is described below. Once enough laser data are labeled in this way, they are then used to train the support vector machine of the laser-based classifier.

6.4.1. Terrain Classification Based on the Vibration of the Vehicle

We just described how a vibration-based classifier can be used to label training data for the laser-based classification system. We will now describe this classifier in more detail. Different types of terrain cause different recognizable patterns of vibration. Vibrations can be measured, for example, by an inertial measurement unit (IMU) and can then be used to classify the terrain the robot currently traverses. Note that such vibration-based classifiers do not allow the prediction of terrain classes in areas which have not been traversed yet. Thus, it is useful for training, but cannot substitute the laser-based classifier presented in this chapter as it cannot predict the terrain type in the area in front of the robot.

In other works (Sadhukhan et al., 2004; Brooks et al., 2005; DuPont et al., 2005; Weiss et al., 2006), the terrain type of up to seven classes have been classified successfully. In our system, however, we only need to differentiate between vegetation and streets (non-vegetation). In our experiments, we use an XSens MTi to measure the acceleration along the vertical axis of the robot. To analyze the spectrum of vibration frequencies, we apply the fast Fourier transform to the raw acceleration data. A discussion of the fast Fourier transform can be found in (Cochran et al., 1967). In our algorithm, a 128-point fast Fourier transform is applied to capture the frequency spectrum of up to 25 Hz. In Figure 6.7, we show an example of the characteristic frequency spectrum induced by grass (left) and asphalt (right).

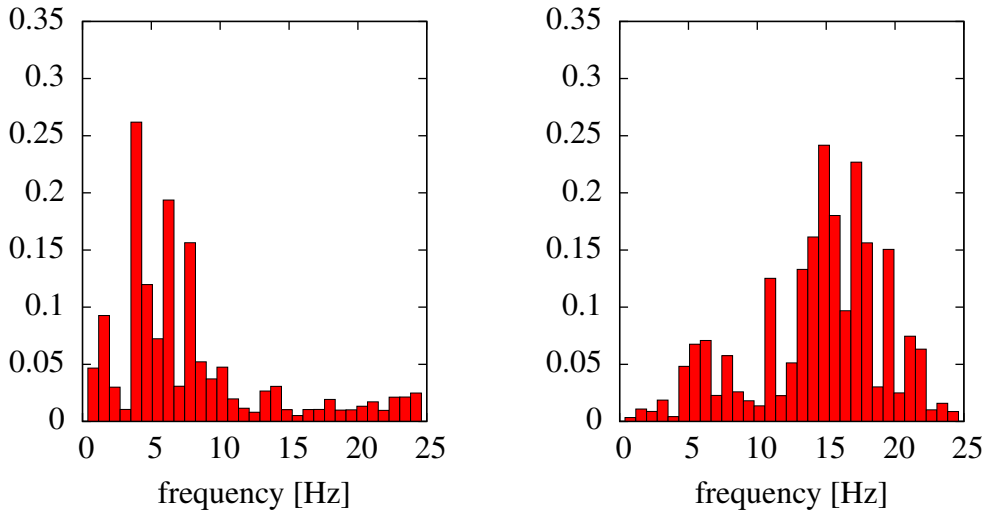


Figure 6.7: Examples of the characteristic frequency spectrum induced by grass (left) and asphalt (right).

The frequency spectrum depends on the terrain type but also on the driving speed of the robot. To account for this dependency, it has been suggested to train several classifiers at different speeds (Weiss et al., 2007). In our system, however, we decided to instead treat the forward velocity as a training feature. In addition to this, we also consider the rotational velocity to account for vibrations that result from the skid-steering of the Pioneer2 AT robot, which we used in our experiments.

We use a C-SVM with an RBF-kernel to classify the acceleration and velocity data. Our feature vector x consists of the first 32 Fourier magnitudes $|F_m|$, the mean forward velocity $\bar{v}_t$, and the mean rotational velocity $\bar{v}_r$ of the robot over the sample period

$$|F_m| = \sqrt{(Re\,F_m)^2 + (Im\,F_m)^2},\ m \in \{0,...,31\} \tag{6.16}$$

$$x = \{|F_0|,...,|F_{31}|,\bar{v}_t,\bar{r}_t\}, \tag{6.17}$$

where F_m denotes the m-th Fourier component. Optimization of the SVM parameters l and C was done using grid search and 5-fold cross validation in the same way that we described in Section 6.3.1.

In the experiments that will be presented in Section 6.9.2, the classifier was trained by recording short tracks of about 50 m at varying speeds both on streets and on vegetation. We achieved an almost perfect classification accuracy with this vibration-based approach. Uncertain classification results are returned mostly at the boundary between different terrains, for example, when the robot leaves a road and then enters an area containing vegetation. In this example, part of the robot is in fact already traversing vegetation while the other part is still situated on the street. This boundary can, however, be detected in a straightforward analysis of the sequence of measured terrain classes and we chose to

ignore such uncertain measurements when generating training data for the laser-based classification.

6.5. Mapping Vegetation

The laser-based classification approach presented in the previous section predicts the terrain type of individual laser measurements. We will now describe how these measurements can be integrated into one consistent model of the environment. To create a three-dimensional map of the environment that also stores terrain classes, we apply the approach presented in Chapter 5. We use a 3D occupancy grid map that discretizes the environment using a regular grid. Each cell in this 3D grid is called a voxel and we estimate the probability of each voxel being occupied by an obstacle. In the following, we extend the basic grid map to store, in addition to the occupancy, the probability of the voxels to contain vegetation.

In general, voxels will be observed multiple times and we need to probabilistically combine results from several measurements. Let $P(\mathrm{v}^i)$ denote the probability of voxel i to contain vegetation and let $z_{1:t}$ be the laser measurements taken at time 1 to t. We derive a recursive update rule for $P(\mathrm{v}^i \mid z_{1:t})$ analogous to the sensor fusion method introduced in Section 5.2.2 on page 107 that is based on occupancy grid mapping (Moravec and Elfes, 1985). The vegetation estimate in each voxel is updated according to:

$$P(\mathrm{v}^i \mid z_{1:t}) = \left[1 + \frac{1 - P(\mathrm{v}^i \mid z_t)}{P(\mathrm{v}^i \mid z_t)} \frac{1 - P(\mathrm{v}^i \mid z_{1:t-1})}{P(\mathrm{v}^i \mid z_{1:t-1})} \frac{P(\mathrm{v}^i)}{1 - P(\mathrm{v}^i)}\right]^{-1} \tag{6.18}$$

To perform the update step, one needs to provide an inverse sensor model $P(\mathrm{v}^i \mid z)$. This, however, can directly be obtained from the estimated class probabilities of the classifier. During the training phase, we apply Platt's method as discussed in Section 6.2.1. The posterior class probabilities computed using Equation (6.15) constitute an inverse sensor model and in this way the uncertainty in classification is explicitly taken into account.

The prior probability of a voxel to contain vegetation $P(\mathrm{v}^i)$ clearly depends on the environment that the robot is operating in. In our system, it was set to $P(\mathrm{v}^i) = 0.5$ to not prefer any class. This choice seems reasonable in an environment that contains as much vegetation as non-vegetation, such as the one depicted in Figure 6.15, but can be adapted if this is needed.

In Figure 6.8, a visualization of a three-dimensional model can be seen that was estimated using the approach we just described. To generate this model, we used data recorded in one of our experiments (see Section 6.9.2). The figure also shows a 3D model that was generated from range measurements only without performing any classification of terrain. Both models capture the geometric structure of the environment but

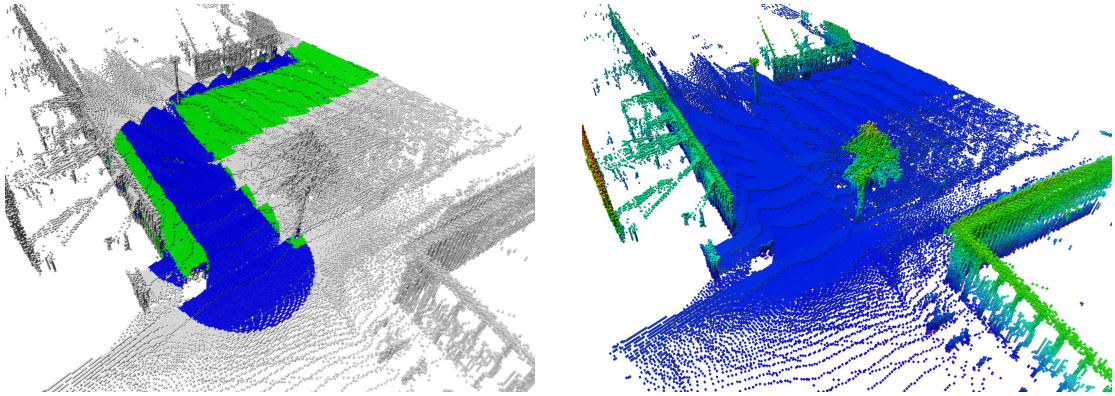


Figure 6.8: Visualization of a 3D model. Left: the output of our classification approach is shown in green (vegetation) and blue (non-vegetation). We classified scans up to a maximum range of 5 m. Right: a 3D model from the same data that does not consider the classification results, colors correspond to a height encoding. Both models have a resolution of 0.1 m.

only the model that contains information about vegetation allows to avoid such surfaces efficiently.

6.6. Comparison of Different Sensors

Most laser range finders that are used on mobile robots are able to provide range and remission information and all of these sensors measure surfaces using near-infrared light. Thus our approach can be applied to data from any such sensor as long as it provides remission values. We evaluated three different laser scanners: the SICK LMS 291-S05, the SICK LMS 151, and the Hokuyo UTM 30LX. To gather example data under comparable conditions, we mounted the sensors on a mobile robot at a fixed angle of 70° relative to the ground and at a height of approximately 0.6 m. The sensors and their setup are shown in Figure 6.9. We recorded example data by moving the robot base forward so that the sensors were swept over the surface. Two data sets were gathered for each sensor: one set containing example measurements of an asphalt street and another set that contained measurements of grass.

The example data are depicted in the plots in Figure 6.10. For the sake of clarity, we omit the estimated incidence angles in the plots and show the ranges and reflectivities only. It can be seen that all sensors report a considerably higher reflectivity for the vegetation samples than for the asphalt samples. The LMS 291 offers the best separation of the classes, especially in the near range, followed by the LMS 151. The two classes overlap considerably in the measurements of the Hokuyo UTM in the near range of up to approximately one meter. From the examples shown above, one can see that all three sensors can be used for vegetation detection but the classification accuracy may vary. The example data of the SICK LMS 151 and the Hokuyo UTM show that it may be challenging to detect vegetation in the near range of up to 1 m but that these sensors provide well-separated measurements beyond this distance. Both sensors are, however, smaller than the SICK LMS 291 so the decision which sensor to use on a given system will be a trade-off between precision in the near range and sensor size.

6.7. Correction of Remission Values

The classification problem described in Section 6.3 would be simplified substantially, if the remission values reported by the sensor could be corrected with respect to the measurement range and the incidence angle. For some laser range finders, the true reflectivity can indeed be computed from the remission value. According to Höfle and Pfeifer (2007), the relationship between the remission ρ_m reported by an airborne laser scanner and the true reflectivity ρ of the measured surface can be modeled as

$$\rho_m \propto \frac{\rho}{R^2}\, \eta_{atm}\, \cos\alpha\, C, \tag{6.19}$$

where R denotes the measured range, η_{atm} is an atmospheric attenuation coefficient, α is the incidence angle and C is a factor that depends on the sensor.

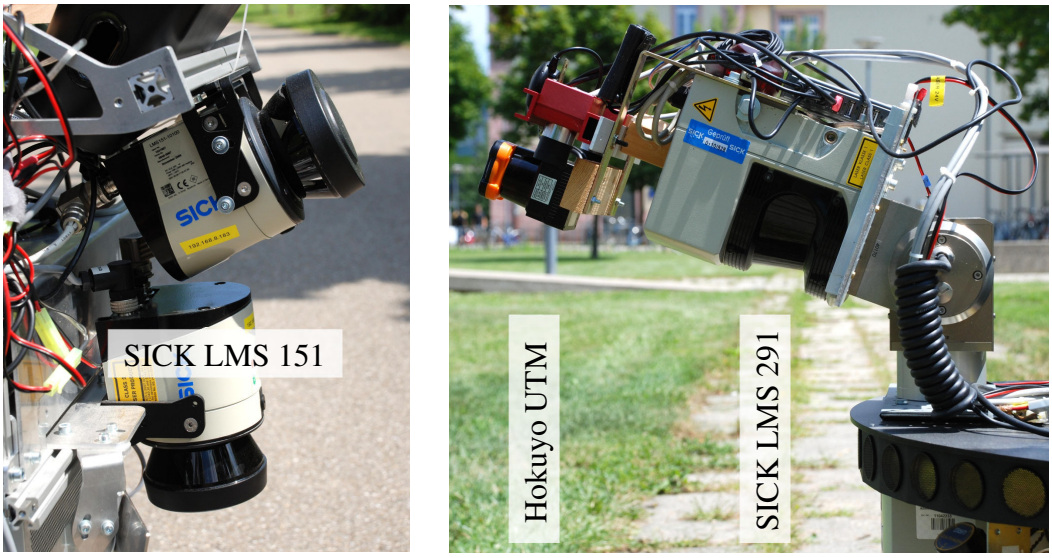


Figure 6.9: Sensors used in the evaluation.

It is important to note that this model assumes a constant emission power and a linear mapping of incoming power to recorded remission values. We applied the proposed correction to the data of the SICK LMS 291 shown in Figure 6.10 (top). The result can be seen in Figure 6.11. Here, we set $\eta_{atm} = 1$ and $C = 1$ since atmospheric effects can be neglected for small ranges (Höfle and Pfeifer, 2007), and C is a constant factor changing only the slope of the correction. Unfortunately, the result still depends non-linearly on the measured range. A likely explanation of this result seems to be that the sensor does not map the received power linearly to the output value. Unfortunately, such details of the internal computations of the sensor are not documented. In the next section we will present an alternative method to reduce the complexity of the classification problem using dimensionality reduction.

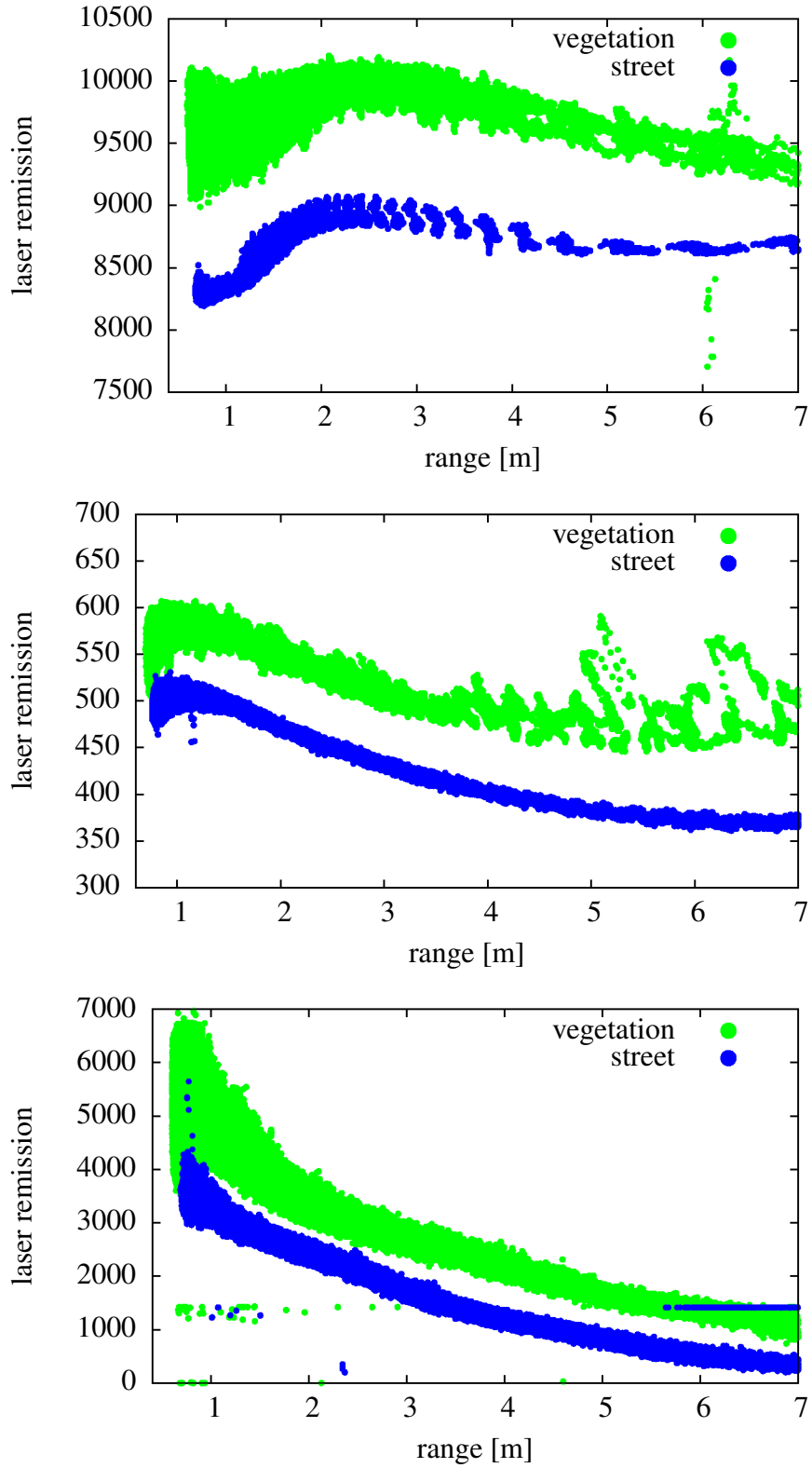


Figure 6.10: Comparison of remission values of a SICK LMS 291-S05 (top), SICK LMS 151 (middle), and a Hokuyo UTM 30LX (bottom). Please note that these are projections of the full three-dimensional data sets since incidence angles are not shown.

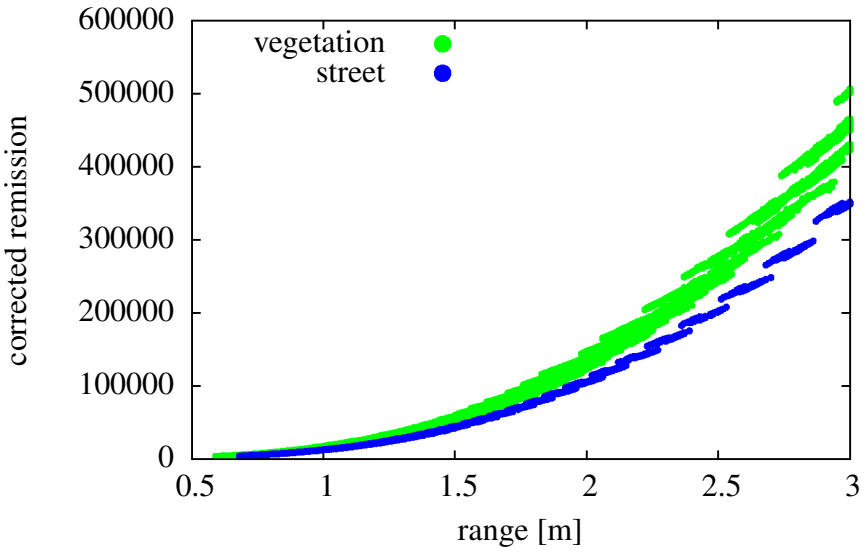


Figure 6.11: Data of the SICK LMS 291 corrected using Equation (6.19).

6.8. Classification for Resource Constrained Systems

On systems with limited computational resources, such as embedded systems, applying our online classification approach can be challenging. To process every laser beam online, assuming a sensor update frequency of 36 Hz and 180 laser beams per measurement, we need to perform 6,480 classifications per second. In other words, each individual classification can take no longer than 0.15 ms. On systems with strictly limited resources, the SVM classifier can be computationally too demanding to be used online.

The computational complexity of the SVM classification is governed by the evaluation of the kernel and the potentially high number of support vectors to appropriately separate the classes. Intuitively, the SVM will use more support vectors when the two classes cannot be well separated linearly. To reduce the complexity of the classification problem in those cases where computational resources are limited, we apply linear discriminant analysis (LDA) (Alpaydin, 2004). We use this method to project the three-dimensional feature space (remission, range, and incidence angle) down to one dimension such that the two classes are best separated. As a further reduction in the computational complexity, we can then apply an efficient linear classifier instead of an SVM to separate the classes. The individual steps are explained in the remainder of this section.

6.8.1. Linear Discriminant Analysis

Linear discriminant analysis (LDA) is a supervised dimensionality reduction technique. In contrast to the similar principal component analysis, LDA considers the labels of the input data and seeks to estimate a reduced feature space so that the classes are best separated. An illustration of a reduction from $\mathbb{R}^2$ to $\mathbb{R}$ is shown in Figure 6.12. Reducing the

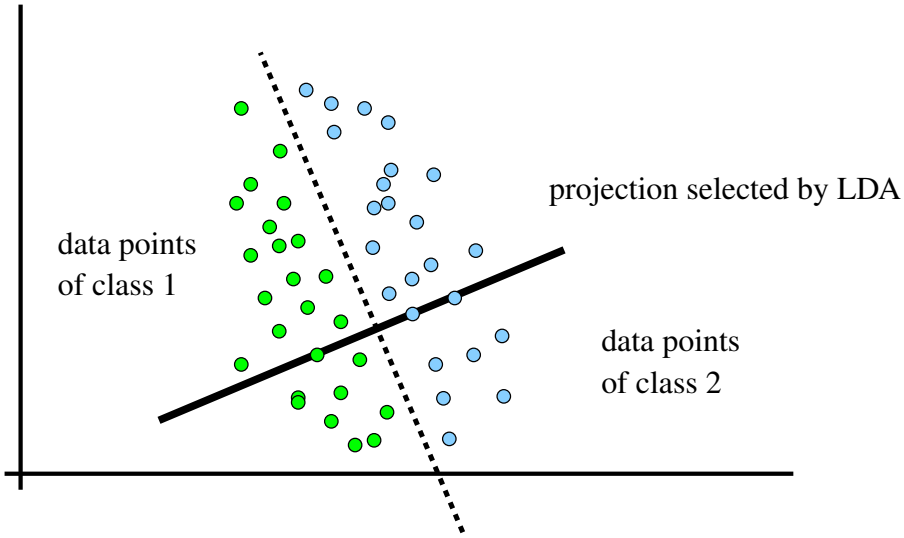


Figure 6.12: Reduction from $\mathbb{R}^2$ to $\mathbb{R}$ using linear discriminant analysis (LDA). LDA seeks to separate the two classes (illustrated by green and blue) as well as possible.

space to a one-dimensional one then allows to separate the classes using, for instance, a naive Bayesian classifier.

We will now describe how LDA computes a reduced feature space from given training data. Let K be the number of classes C_k in the training data and x_i the d-dimensional training inputs. Let furthermore t be the dimensionality of the target space. In our vegetation classification problem, for example, we have $K = 2$ and $d = 3$ and in our experiments we found a one-dimensional target space ($t = 1$) to be sufficient. The objective of LDA is then to find a $d \times t$ matrix W so that $v_i = W^T x_i$ with $v_i \in \mathbb{R}^t$ and so that the classes C_k are separated best in terms of distances between the v_i.

Let $r_{k,i}$ be an indicator variable with $r_{k,i} = 1$ if $x_i \in C_k$ and $r_{k,i} = 0$ otherwise. Let furthermore μ_k be the mean of the d-dimensional vectors $x_i \in C_k$. Then, the so-called scatter matrix of class C_k is defined as

$$S_k = \sum_i r_{k,i} \, (x_i - \mu_k)(x_i - \mu_k)^T, \tag{6.20}$$

and the total within-scatter matrix is defined as

$$\begin{aligned} S_W &= \sum_{k=1}^{K} S_k \\ &= \sum_{k=1}^{K} \sum_i r_{k,i} \, (x_i - \mu_k)(x_i - \mu_k)^T. \end{aligned} \tag{6.21}$$

The between-class scatter matrix is defined as

$$S_B = \sum_{k=1}^{K} \left(\sum_i r_{k,i} \right) (\mu_k - \mu)(\mu_k - \mu)^T, \tag{6.22}$$

$$\text{with } \mu = \frac{1}{K} \sum_{k=1}^{K} \mu_k. \tag{6.23}$$

Let us now consider the scatter matrices after projection using W. The within-scatter matrix after projection is $W^T S_W W$, and the between-class scatter matrix accordingly is $W^T S_B W$, both are $t \times t$ dimensional. The goal is to determine W in a way that the projected means of the classes $W^T \mu_k$ are as far apart from each other as possible while the spread of their individual projected class samples is small. The determinant of a scatter matrix characterizes the spread and it is computed as the product of the eigenvalues specifying the variance along the eigenvectors. Thus we aim at finding the matrix W that maximizes

$$J(W) = \frac{|W^T S_B W|}{|W^T S_W W|}. \tag{6.24}$$

The largest eigenvectors of $S_W^{-1} S_B$ are the solution to this problem. We refer the reader to Alpaydin (2004) for more details on LDA.

6.8.2. Classification Using Linear Discriminate Analysis

Given the projection computed by applying LDA to our training data, we are able to reduce the feature space to one dimension. This enables us to use an efficient linear classifier to compute class probabilities for laser measurements. For each class C_k, our method fits a Gaussian distribution $p(x \mid C_k)$ using maximum likelihood estimation. Posterior probabilities are obtained by applying Bayes' rule

$$p(C_k \mid x) = \frac{p(x \mid C_k) p(C_k)}{\sum_i p(x \mid C_i) p(C_i)}, \tag{6.25}$$

where $p(C_i)$ is the class prior for class C_i. Since in our case we have two classes and we assume a uniform prior, this simplifies to

$$p(C_k \mid x) = \frac{p(x \mid C_k)}{p(x \mid C_1) + p(x \mid C_2)}, \tag{6.26}$$

where $k \in \{1, 2\}$ refers to the classes "vegetation" and "street".

Note that this classification approach is significantly faster than the SVM, but it is also less powerful. Our evaluation in Section 6.9.6 shows that the resulting error depends on the type of sensor and the measurement range. In our experiments, we observed that

the precision dropped from 99% down to 96% in the worst case compared to the SVM-based classifier. However, the LDA-based system was around two orders of magnitude faster than the SVM. For this reason, there clearly is a trade-off between precision and classification runtime.

6.9. Evaluation

Our approach has been implemented and evaluated in several experiments. The experiments are designed to demonstrate that our approach improves navigation in structured outdoor environments by enabling robots to reliably detect vegetation.

We used three different robot systems in our experiments (see Figure 6.13). The self-supervised learning approach is evaluated using an ActivMedia Pioneer 2 AT, which is able to traverse low vegetation. It carries an XSens MTi IMU to measure vibrations while it traverses the terrain. For mapping large environments and for an autonomous driving experiment, we use an ActivMedia Powerbot platform. This robot cannot safely traverse grass since its castor wheels would sink into the softer ground due to its weight, thus blocking the robot. Both robots are equipped with SICK LMS S291-S05 laser scanners on pan-tilt units. Three-dimensional scans are gathered by tilting the laser scanner from 50° upwards to 30° downwards. To perform navigation experiments and to evaluate the SICK LMS 151 sensor, we used the EUROPA platform depicted in Figure 6.13 (right). Just as the Powerbot, this platform cannot safely traverse vegetation. Furthermore, the sensor is mounted at a fixed angle on this platform.

In the evaluation we limit the classification to laser measurements with a range smaller than 5.0 m. The approach itself is not limited in range as it explicitly models the dependency of remission values on the range. However, long range data are too sparse both to reliably detect drivable surfaces and to gather training data. The reason for this is the

Figure 6.13: Robots used in our experiments. Left: An an ActivMedia PowerBot (yellow) and an ActivMedia Pioneer 2 AT (red), both equipped with SICK LMS 291 laser range finders. Right: The EUROPA platform equipped with a SICK LMS 151 sensor.

coarse angular resolution of the laser scanners. The LMS 291, for example, has an angular resolution of 1° and at 5 m distance two adjacent measurements are already set 87 mm apart. Additionally, the low mounting height of the sensor relative to the ground, which is approximately 0.5 m-0.6 m, results in very flat incidence angles at longer distances and this leads to a weak remission response of the sensors at far ranges.

Throughout this work, we used the *C*-SVM implementation of LibSVM (Chang and Lin, 2001), Fourier analysis was performed using FFTW3 (Frigo and Johnson, 2005).

6.9.1. Comparison to Vegetation Detection Using Range Differences

In a first experiment, we evaluated an approach that detects vegetation based purely on the range differences of neighboring laser measurements. It is similar to the algorithm proposed by Wolf *et al.* (2005) but extends it to three-dimensional data. We implemented the method for the purpose of comparing its results to our proposed classification approach.

To identify vegetation in the surrounding of the robot, three-dimensional data are acquired by gathering a sweep of 2D scans. This can be achieved by tilting the laser using a pan-tilt unit or by moving the robot base in those cases where the laser is mounted at a fixed angle. Typical laser scanners with a field of view of 180° return 181 or 361 range values per sensor measurement, depending on the angular resolution of the sensor. From the range and angle measurements of each laser beam we compute the individual 2D endpoints $p_i = \langle x_i, y_i \rangle$. A local feature d_i is then determined for every scan point x_i depending on its neighboring scan point x_{i-1} as

$$d_i = x_i - x_{i-1}. \tag{6.27}$$

This feature captures the local roughness of the terrain. To cope with flat but tilted surfaces, we classify scans based on the absolute difference in d_i between adjacent range measurements, as suggested by Manduchi *et al.* (2003)

$$f_i = |d_i - d_{i-1}|. \tag{6.28}$$

Since the density of the 3D data varies from near to far scans, we also consider the measured range in the classification step. To predict the terrain, we trained an SVM based on these two features, the roughness represented by f_i and the range of the measurements.

In our experiments, we achieve a classification accuracy of about 75% using the described method. An example of the classification results can be seen in Figure 6.1. Similar results have been reported by Bradley *et al.* (2007). As we will show in the remainder of this section, our approach offers a substantially higher accuracy using the same type of sensor.

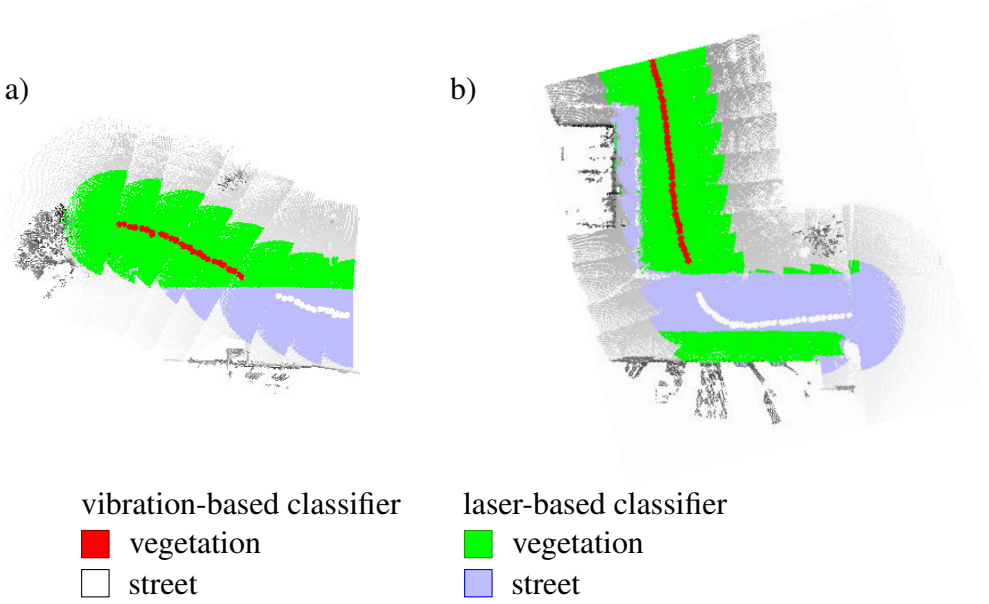


Figure 6.14: Visualization of the data we used to evaluate the laser-based vegetation detection approach. Left: the training set which was used to train the classifier. Right: the test set recorded at a different location. The output of the vibration-based classifier we used to label the data sets is shown in red/white and the classification result of the laser-based approach after training is shown in green/blue.

6.9.2. Vegetation Detection Based on Laser Remission

To evaluate our laser-based approach as introduced in Section 6.3, we recorded two sets of data, a training set and a testing set. Both data sets were gathered by manually steering the Pioneer 2 AT robot through an outdoor environment consisting of streets and areas covered with grass. Both data sets contain sample measurements of vegetation and non-vegetation surfaces. The location of both robot trajectories on the Freiburg computer science campus can be seen in Figure 6.15. To correct odometry errors of the robot during data acquisition, we employed a 2D SLAM approach (Stachniss and Grisetti, 2007) in combination with 3D scan-matching. We recorded 3D scans approximately every 4 m and stored range, estimated incidence angle, and remission values of every surface measurement. While the robot was traversing the terrain, its IMU sensor measured the vibration of the robot's body. These vibration measurements were classified using the approach described in Section 6.4.1 and were used to label the laser data. The data recorded at the border region between streets and vegetation were ignored since the precise location of the border cannot be determined using the vibration sensor. The training set is visualized in Figure 6.14 a. It consists of 19,989 vegetation and 11,248 street sample measurements. Based on this data, the laser-based SVM-classifier, as described in Section 6.3.1, was trained.

We recorded separate test data at a different location to evaluate the precision of the

Table 6.1: Confusion matrix for Experiment 6.9.2 (number of data points)

	vegetation	street
vegetation	36,300	138
street	4	28,745

classifier. This data is visualized in Figure 6.14 b. The test set contains 36,304 vegetation and 28,883 street measurements. Again, the labeling of the data was carried out using the vibration-based classifier. The previously trained laser-classifier reached a recall of 99.6 % and a precision of 99.9 % on the test data. The confusion matrix is given in Table 6.1. Note that false positive classifications, that is, the classifier predicted the terrain to contain vegetation when it was in fact a street, occurred only 4 out of 28,749 times. This result implies that a robot which is navigating on a street will not take unnecessary detours to avoid erroneous detections of vegetation. In sum, our results show that our approach is able to detect vegetation reliably.

6.9.3. 3D Mapping

In the previous experiment we evaluated the classification of single laser measurements. In contrast to this, we now present an experiment that uses the mapping approach presented in Section 6.5 to detect vegetation in a large environment. In this experiment, we recorded data using the Powerbot robot at the computer science campus of the University of Freiburg. The laser range finder of the robot was tilted to a fixed angle of 20° downwards. The robot was then steered manually on the streets of the campus. In this way, a fairly large area was covered in less than 15 minutes. The length of the trajectory is 490 m.

The laser-based approach presented in this chapter was used to detect vegetation in a three-dimensional model of the environment. To properly integrate multiple measurements, we used the mapping approach described in Section 6.5 where the cell size in the model is 0.1 m. Compared to the previous experiment, we decided to use a significantly different sensor setup on the Powerbot platform and we were therefore not able to use the SVM-model generated for the Pioneer robot. Instead, we recorded a training set of 12,153 grass and 10,448 street samples by placing the robot in front of flat areas containing only street and only vegetation. This method is only applicable if such training data can be gathered and thus should be considered inferior to the self-supervised approach described in Section 6.4.

The results of the mapping experiment can be seen in Figure 6.16. In comparison to the aerial view of the campus site shown in Figure 6.15, the mapping result achieved by our approach is highly accurate. Even small amounts of vegetation, for example between tiles on a path, can be identified. To quantitatively evaluate the accuracy of the resulting map, we used the aerial view and our own knowledge of the environment to manually mark

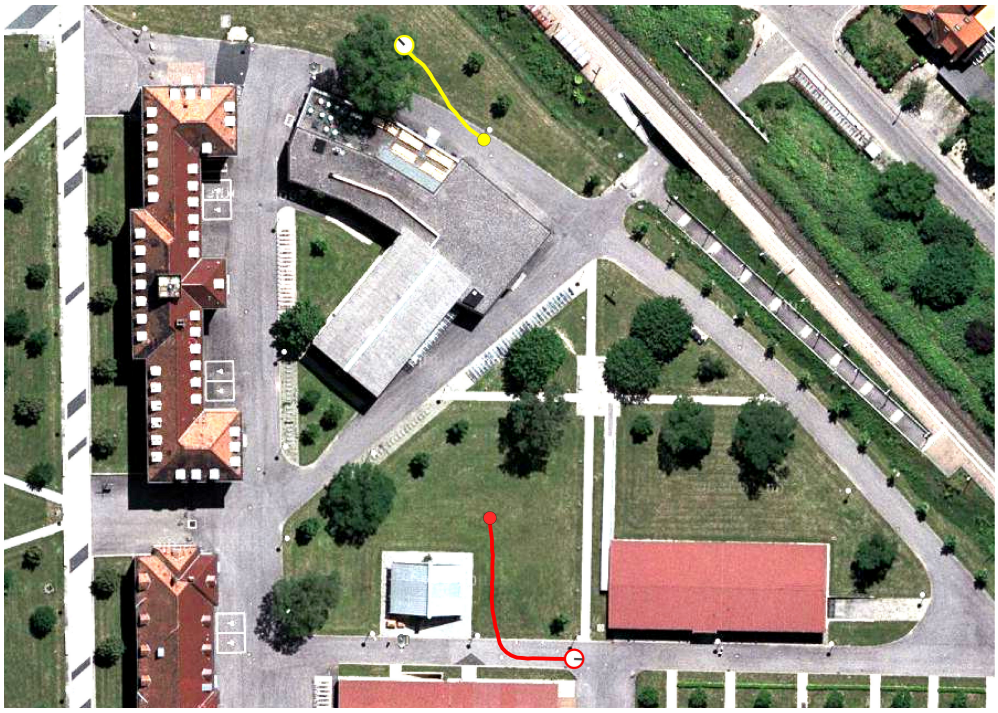


Figure 6.15: Aerial view of the computer science campus in Freiburg. Estimated robot trajectories are shown for the training (top, yellow) and the test set (bottom, red). Aerial image courtesy of Google Maps, Copyright 2008, DigitalGlobe.

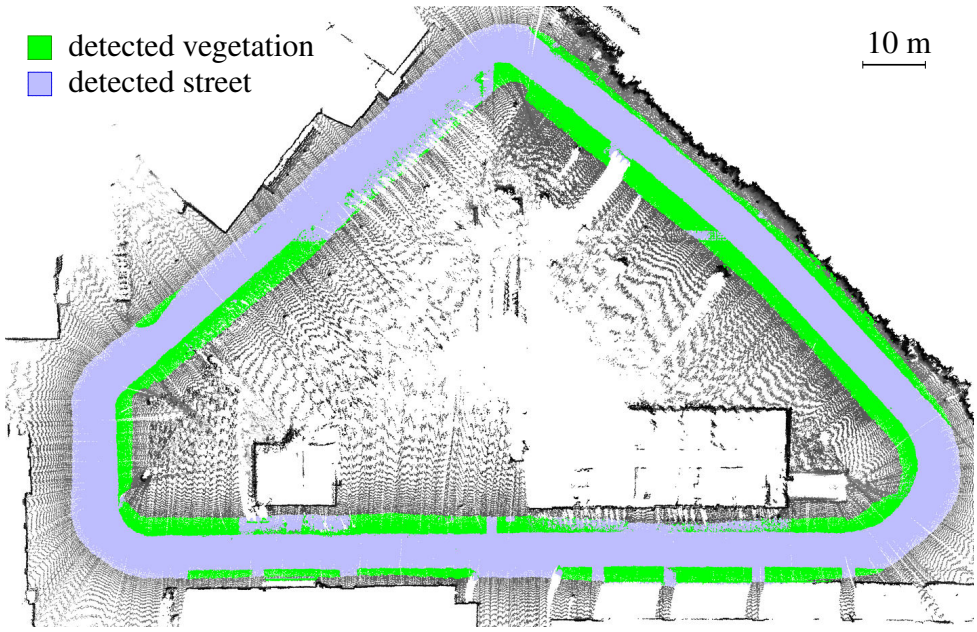


Figure 6.16: Mapping a large outdoor environment. The laser was tilted at a fixed angle of 20 degrees while the robot was moving. The figure shows a 2D projection of the 3D map.

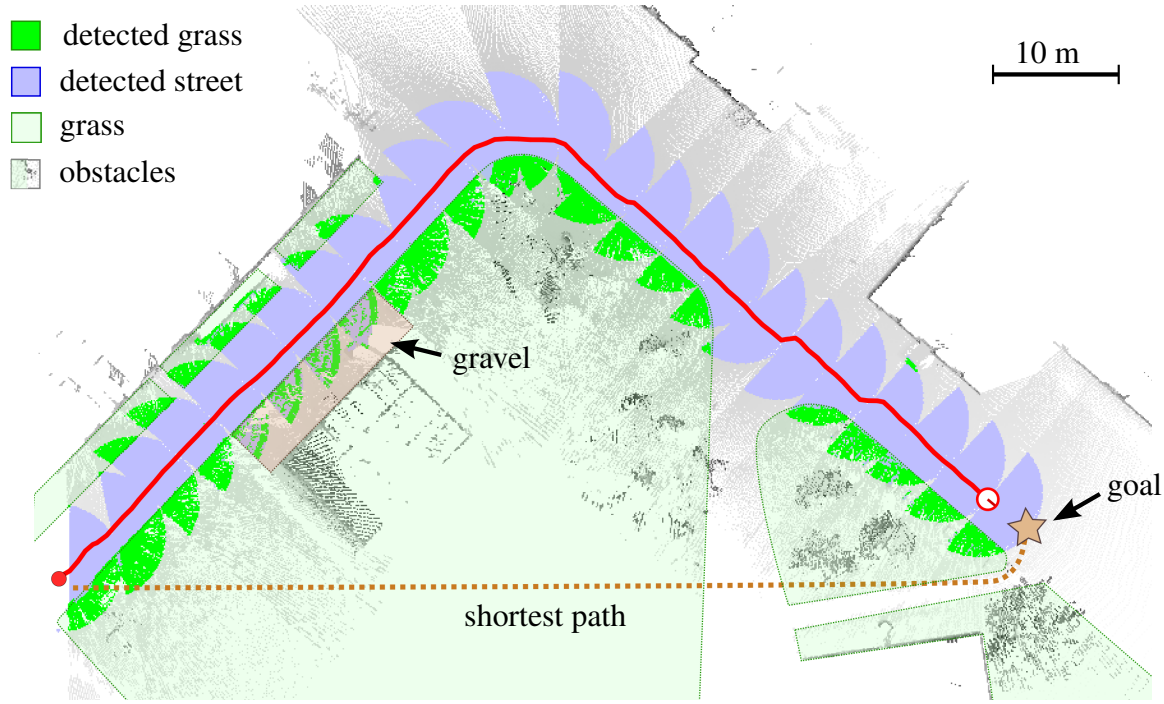


Figure 6.17: The task of the robot was to reach the goal without traversing vegetation. No prior map was given to the robot. The robot reliably avoided the vegetated areas by using our vegetation classifier and traveled over the paved streets to reach the goal. The output of the classifier is shown in dark green (vegetation) and blue (street) while we highlighted the remaining areas that contain vegetation in light green. The trajectory of the robot during the experiment is depicted in red.

wrongly classified cells. Of a total of 271,638 cells (75,622 vegetation and 196,016 street measurements), we found 547 false positives and 194 false negatives. This corresponds to a precision of 99.23 % and a recall of 99.74 %. This result shows that our vegetation classifier in combination with our mapping approach is able to identify vegetation in large environments.

6.9.4. Autonomous Navigation Using a 3D Scanner

As mentioned above, the Powerbot robot cannot safely traverse vegetation. To show that our approach improves the navigation capabilities of this robot in environments that contain vegetation, we conducted the experiment illustrated in Figure 6.17. Here, the robot was told to autonomously navigate to a goal position in a distance of 80 m and to avoid vegetation while it was driving there. Without knowledge about the specific terrain, the shortest obstacle-free path would have led the robot across a large area containing grass. Since the robot did not have a map, it explored the environment in the process of reducing its distance to the goal location. It created a map of the environment and detected vegetation using our approach. The resulting trajectory is shown in Figure 6.17. The robot chose a safe trajectory along the street by considering the classification results

in the path costs computed by its navigation system. It successfully avoided areas that contain vegetation and an area that contains loose gravel which is overgrown by grass was also avoided.

6.9.5. Autonomous Navigation Using a Fixed-Angle Sensor

We conducted an extended autonomous navigation experiment to show that our system is able to guide autonomous vehicles during a navigation task in large-scale environments. In this experiment, we used a custom-made platform depicted in Figure 6.13 (right). It was designed for urban navigation within the project EUROPA and we will refer to this robot as the EUROPA platform. It is equipped with a SICK LMS 151 that is mounted at a fixed angle and additionally uses a horizontally mounted range finder to detect obstacles (the sensors are shown in Figure 6.9). The robot was steered manually along a trajectory of approximately 7,500 m across the Freiburg computer science campus and neighboring areas to obtain a map of the environment. The mapped area includes the computer science campus, urban areas, and the park of the University Medical Center Freiburg. In addition to this, a set of example measurements of vegetation and non-vegetation was collected to train our laser-based classifier. Since the EUROPA platform is not able to safely traverse grass, we recorded this training set by placing the robot in front of grassland and alongside a street. We then trained the classifier and applied it to classify the previously collected data. Our approach accurately identified vegetation, and a visualization of the resulting map can be seen in Figure 6.18.

After the map was created using our approach, we carried out an autonomous navigation experiment that is summarized in the same figure. The task of the robot was to navigate from a given start to a given goal location using the previously created map. During this task the robot used its horizontal laser scanner to localize itself. The trajectory that the robot followed is shown in red. At the beginning of the experiment, the robot was located on the computer science campus in the top left of the map. The target position was located in the park area in the lower right part of the map and the shortest path would have led across large areas covered with vegetation. By taking into account the classification result of our approach, however, the robot was able to plan a feasible and safe path towards the goal location. The robot then autonomously traveled a total distance of approximately 1,120 m and successfully reached the goal location.

6.9.6. Resource-Friendly Classification with Linear Discriminant Analysis

In this section, we compare the SVM-based classifier introduced in Section 6.3 and the linear classifier using LDA for dimensionality reduction as introduced in Section 6.8. We compared both approaches with respect to runtime and classification accuracy. For

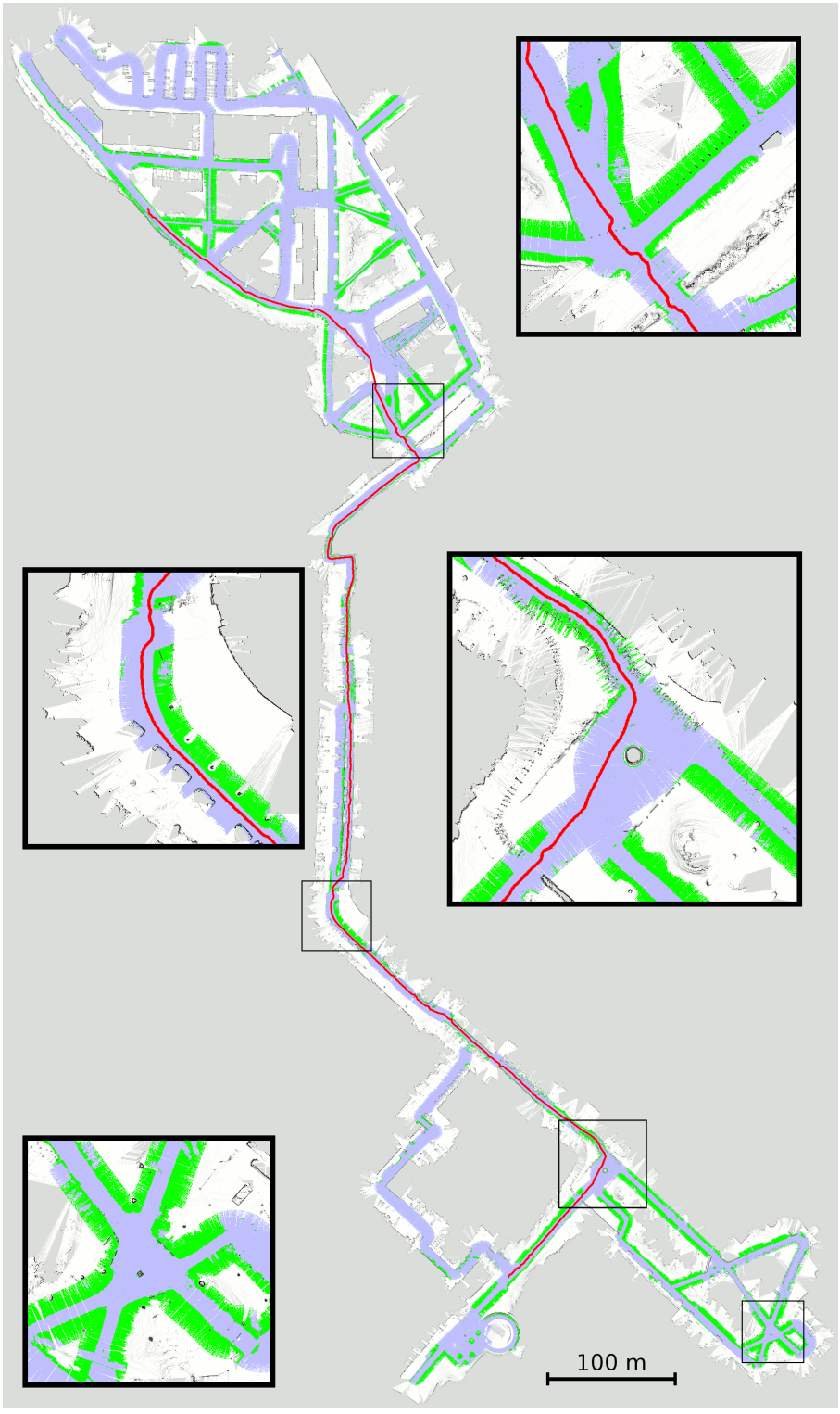


Figure 6.18: Visualization of a map of a large-scale environment and the result of our vegetation detection algorithm. Areas classified as vegetation are shown in green while drivable surfaces are shown in blue. Magnified views of selected areas in the environment are shown in rectangles. Using this map, the EUROPA robot autonomously navigated along the path shown in red. The robot traveled a distance of approximately 1,120 m.

Table 6.2: Comparison between the SVM-based and the LDA-based classifiers

	precision (SVM)	recall (SVM)	precision (LDA)	recall (LDA)
LMS 151	99.7%	100%	99.1%	99.8%
LMS 291	99.9%	100%	99.4%	100%
UTM	99.9%	99.9%	96.3%	99.8%

evaluation, we used labeled data that we recorded with three different laser scanners: a SICK LMS 291-S05, a SICK LMS 151, and a Hokuyo UTM 30LX. We divided each dataset into a training dataset containing 2,000 randomly sampled data points and a test dataset containing all the remaining data points.

First, we compared the runtimes of the SVM-classifier and the LDA-based approach. In our experiments on a regular desktop computer, the LDA-based approach was around 400 times faster than classification using the support vector machine. As expected, the LDA-based approach is computationally less demanding and thus is useful for resource-constraint systems or for robots that run a fairly large number of processes.

Next, we compared the classification accuracy of both approaches. The results are depicted in Table 6.2. On the SICK LMS 151 data set the LDA approach had a recall of 99.1% and a precision of 99.8% whereas the SVM-based classifier had a recall of 99.7% and a precision of 100.0%. For the SICK LMS 291 dataset, the LDA-based approach yielded a recall of 99.4% and a precision of 100.0% whereas the use of the SVM resulted in a recall of 99.9% and in a precision of 100.0%. When applied to the Hokuyo UTM 30LX dataset, the LDA-based approach reached a recall of 96.3% and a precision of 99.8% while the SVM-based classifier gave a recall of 99.9% and a precision of 99.9%.

The results show that the classification of both classifiers is highly accurate. However, in all our experiments the SVM-based approach outperforms the LDA-based method. When analyzing the error, we observed that most of the misclassification of the LDA-based classifier occur at measurements in a range of up to 2 m. In autonomous systems, however, close-range terrain predictions are critical to an efficient navigation: To ensure the safety of the vehicle, the path planner has to avoid traversing any vegetation. In this way, false terrain predictions can lead to detours and unplanned halts. Therefore, if efficiency and safety of the system are key requirements, the SVM-based system should be preferred in online navigation tasks.

6.10. Related Work

There exist several approaches for detecting vegetation from laser measurements using geometric features. Wolf *et al.* (2005) present an approach that analyzes scans from a tilted laser scanner to differentiate navigable (e.g., street) from non-navigable (e.g., grass) terrain. They use hidden Markov models and their main feature for classification is the

variance in height measurements. In contrast to the approach presented in this chapter, they do not address the problem of classifying 3D point clouds in the surrounding of the robot. Other approaches analyze the distribution of 3D endpoints in a sequence of scans (Macedo et al., 2001; Hebert and Vandapel, 2003; Lalonde et al., 2006). However, flat vegetation, such as a freshly mowed lawn, cannot be reliably detected using this feature alone.

It seems intuitive to use color cameras to detect vegetation in the environment. Manduchi (1999), for example, uses a combination of color and texture features to detect grass in camera images. The main drawback of using cameras is that they are sensitive to lighting conditions. Shadows, for instance, can have a strong influence on the appearance of vegetation. A more robust classifier can be derived using the Normalized Difference Vegetation Index (NDVI). This value is based on the difference between red and near-infrared light reflectance and is a strong indicator for the presence of vegetation, see (Myneni et al., 1995) for a discussion on this topic. On a mobile robot, the NDVI can be determined using a calibrated combination of a regular and an infrared camera (Bradley et al., 2007) or a single camera that uses a filter mask to capture both frequency bands at different positions on the same sensor (Zhao et al., 2010). Unfortunately, such a sensor setup is rarely available on a mobile robot and, being a passive sensor, still depends on ambient light.

A combination of camera and laser measurements has been used to detect vegetation in several approaches (Manduchi et al., 2003; Wellington et al., 2006; Bradley et al., 2007; Douillard et al., 2008). In a combined system, Wellington *et al.* (2006) use the remission value of a laser scanner as a classification feature in addition to density features and camera images. However, they do not take into account the dependency of remission values on the measured range and incidence angle. Probably due to this fact, they found the feature to be only "moderately informative". In contrast to that, we showed in our work that remission is highly informative if it is considered jointly with the measured range and the incidence angle of the individual laser beams. The approach that is closest to our approach has been proposed by Bradley *et al.* (2007). Vegetation is detected using a combination of laser range measurements, regular and near-infrared cameras. Vegetation is recognized using the NDVI computed from both camera images. 3D laser measurements of the environment are projected into the camera images. A classifier is then trained using the vegetation feature and features from the distribution of 3D endpoints. According to the authors, the approach yields a classification accuracy of up to 95%. This approach, however, requires sophisticated camera equipment. In contrast to combined systems that make use of laser measurements in addition to camera systems, our approach uses a laser scanner as the sole sensor. Using our approach, we achieved classification results with an accuracy of over 99% in all our experiments. Furthermore, our system is independent of lighting conditions and can be used on a variety of existing robot systems that are often already equipped with a laser range finder.

Terrain types have also been classified using vibration sensors (Sadhukhan et al., 2004; Brooks et al., 2005; DuPont et al., 2005; Weiss et al., 2006). In these approaches, a robot traverses the terrain and the induced vibration is measured using accelerometers. The measurements are usually analyzed in the Fourier domain. Sadhukhan *et al.* (2004) presented an approach based on neural networks. A similar approach was presented by DuPont *et al.* (2005). Brooks and Iagenemma (2005) use a combination of principal component analysis and linear discriminant analysis to classify terrain. The approach presented by Weiss *et al.* (2006; 2007) uses support vector machines to classify vibration patterns. Vibration-based approaches typically offer highly accurate classification results. The drawback of such methods is, however, that only the terrain underneath the robot can be classified. It is not possible to predict the terrain type in the surrounding of the robot. Thus, this information cannot be integrated directly into the path planning system of a mobile robot.

Self-supervised learning has previously been used by Dahlkamp *et al.* (2006) in a vision-based road detection system. Here, laser measurements are used to identify nearby traversable surfaces. This local information is then used to label camera image patches in order to train a classifier that is able to predict traversability in the far range. In our approach, we adopt the idea of self-supervision to generate labeled training data. We apply a vibration-based classifier to label laser measurements recorded by the robot. This labeled dataset is then used to train a laser-based vegetation classifier.

6.11. Conclusion

In this chapter, we presented a novel approach to vegetation detection that uses the remission values of a laser scanner. Laser remission values depend on the surface reflectivity as well as on the distance and the angle of incidence. This dependency has not been considered by previous methods that detect vegetation. Our approach trains a classifier based on individual laser measurements consisting of the remission value, the distance to the surface, and the angle of incidence. Our method is able to distinguish vegetation from drivable surfaces such as streets. To predict the terrain type, we use a support vector machine (SVM). We avoid the need to label training data manually by labeling sets of example measurements in a self-supervised fashion by means of a pre-trained vibration-based terrain classifier. In addition to the system based on support vector machines, we presented a classification approach based on linear discriminant analysis (LDA). This system is around 400 times faster than the SVM-based approach and is especially designed for robots with limited computational resources. Both approaches have been implemented and evaluated in various real-world experiments. Our experiments show that the SVM-based approach is able to accurately detect low, grass-like vegetation with an accuracy of close to 100%. The classification method based on LDA achieves accurate

predictions but in our experiments they were up to 4% worse than results of the SVM-based approach. In further experiments, we demonstrated that autonomous robots are able to navigate efficiently and safely in structured outdoor environments using our terrain classification method.

Chapter 7
Discussion

In this thesis, we presented several innovative techniques that enable teams of robots to explore and map environments more efficiently. When teams of robots explore an environment, there are two fundamental challenges that need to be addressed. The first challenge is the coordination of the team, in other words, assigning actions to robots so that the efficiency of the team is maximized. The second challenge is model estimation, the task of generating a map of the environment.

7.1. Contributions

With respect to robot coordination, we identified two open research questions: How can semantic knowledge about the environment be utilized to improve multi-robot systems? And how can heterogeneous teams of robots be coordinated efficiently? In the first part of this thesis, we presented novel coordination approaches that address these questions.

In Chapter 2, we considered the question of how a team of exploring robots can make use of information about the structure of a building. Knowledge about structurally important regions, such as rooms and corridors has not been considered by previous coordination approaches. In typical environments, however, multiple exploration targets are often generated in the same room or corridor. This led to cases in which several robots explored the same room or corridor. We presented a coordination technique that makes use of structural knowledge to assign robots to separate regions of the environment. Our approach partitions the map of a partially explored building into regions such as rooms. It then clusters exploration targets in the regions and assigns robots to those regions instead of assigning them to exploration targets directly. Applying this coordination strategy leads to a balanced distribution of robots in the building. In our experiments, we could show that our approach outperforms an approach that does not consider the structure of the building.

In Chapter 3, we presented an approach to coordinate heterogeneous teams of robots in which the robots are able to perform actions that go beyond navigation. For instance, robots might be equipped with manipulators to open doors, or they could be able to deploy other robots. Such symbolic actions stand in contrast to navigation actions that move robots to a given goal position. We presented a technique to coordinate teams of robots that integrates a symbolic planning system and a robotic path planner. This combination allows our system to consider planning problems that include symbolic actions. To coordinate a team of robots, we generate a symbolic description of the current state of the system and of the goal state. These descriptions serve as input to the symbolic planner. To solve the coordination problem, the symbolic planner uses the path planner to efficiently estimate travel costs for the robots. In contrast to cost-based coordination approaches, our system is able to explicitly plan for the execution of symbolic actions. Still, the use of action costs that are determined by the robotic path planner allows us to generate time-efficient solutions. In this way, our approach combines the strength of cost-based coordination approaches with the flexibility of symbolic planning systems. We applied our approach to two settings. First, we considered teams of exploring robots that were able to deploy and pick up smaller robots. Second, we simulated a disaster scenario in which the task was to clear blockades and to perform pre-defined actions at certain critical locations in the environment. Our coordination approach successfully coordinated the robots in both scenarios and outperformed heuristic extensions of numeric coordination approaches.

In the second part of this thesis, we addressed open questions in the field of map estimation. Specifically, we presented solutions to the problem of combining maps of multiple robots into one joint map, we presented an approach to efficiently map 3D environments, and we introduced a robust technique to identify vegetation in outdoor environments.

In Chapter 4, we presented an approach to simultaneous localization and mapping (SLAM) with multiple robots. The task of mapping the environment with a team of robots can be rephrased as a divide-and-conquer problem. In the first step (divide), the mapping problem is distributed among the robots and each robot solves a so-called local mapping problem. In the second step (conquer), the team combines the local results into a joint solution. If no global estimates of the robot positions are available, the conquer step entails a data association problem. The team has to identify locations that occur in more than one of the local maps. We presented two methods to solve the data association problem in distributed systems. The first approach computes associations between grid maps while the second approach associates parts of landmark maps. Our grid-based approach applies Monte Carlo localization and performs global pose estimation. By localizing one robot in the map of another robot, it associates positions in multiple grid maps. The second approach aligns maps of landmark positions. It computes a Delaunay triangulation of the maps and then searches for similar triangles using geometric features. Matching triangles are then used to compute relative transformations between

pairs of overlapping maps. The algorithm has an average computational complexity of $O(n \log n)$ and is among the fastest approaches that solve this particular problem. The results of our evaluation showed that alignments between individual robot maps can be determined robustly and that consistent global models are estimated by our approach. We successfully applied the grid-based approach to the problem of mapping multi-floor buildings, where each floor was mapped independently. We computed relative transformations between the individual floors of a building and in this way generated multi-floor maps that were more accurate than those obtained with a standard SLAM approach. The experiments also showed that using constraints computed from map associations can improve the overall accuracy of the model. Inter-graph constraints allow to mitigate errors that are present in one graph using information from other graphs that correctly model the corresponding part of the environment.

An efficient method to generate three-dimensional maps was presented in Chapter 5. While techniques exist to estimate large 2D maps, there is no effective extension of these methods to the case of 3D mapping. For example, 2D grid maps are well suited to map large planar environments but the large memory requirement of 3D grid maps limits its use in real-world applications. We presented an efficient approach to generate 3D maps which are compact, probabilistic, and which can be used during 3D exploration. We make use of probabilistic state estimation techniques and efficient data structures to achieve these goals. Our approach is able to integrate data from multiple robots or sensors. As one of the key contributions, we presented a lossless map compression method to keep 3D models compact. The maps we generated in our evaluation were as small as 8% of the size of a 3D occupancy grid. We furthermore addressed the question of how semantic information can be incorporated into 3D maps. We presented an approach that uses knowledge about supporting planes to construct a hierarchy of 3D maps. Our approach maintains a tree of independent 3D submaps that is based on a user-defined spatial relation. In this way, the hierarchy is able to encode dependencies in the environment, for example, objects that are supported by planes. Compared to a single global map, our hierarchy of maps offers two main advantages. First, the mapping parameters such as the resolution can be adapted for each submap. Second, submaps can be manipulated independently. For example, one can represent movable objects in submaps and then transform these object maps independently from the scene background. Our evaluation showed that in a tabletop scenario a hierarchy of maps consumes about one order of magnitude less memory than a single 3D map and that the hierarchy can also be updated significantly faster. Moreover, we showed that the hierarchy is able to adapt to changing environments. In addition to that, we presented an information-driven approach to autonomously generate hierarchical maps of tabletops.

In Chapter 6, we present an approach to detect vegetation. In structured outdoor environments, such as parks or campus sites, large areas are often covered with grass. Our approach reliably detects flat vegetation from remission values of laser scanners. To pre-

dict the terrain type, it uses a support vector machine. The input to this classifier are individual laser measurements consisting of the remission value, the distance to the surface, and the angle of incidence. In contrast to previous approaches, we take into account the dependency of the distance and the incidence angle on the laser remission value. We presented a self-supervised training technique that uses a pre-trained vibration-based terrain classifier to generate training data for the laser-based classifier. In our evaluation, we showed that our approach is highly reliable and achieved an accuracy of more than 99%. We applied our classifier to improve traversability estimates in structured outdoor environments. In our experiments, we showed that vehicles that have not been designed to drive on vegetation are able to safely navigate in such environments using our approach.

All of our approaches have been evaluated extensively and all approaches have been tested successfully in real world experiments. We coordinated a team of real robots that explored an office building using the approach introduced in Chapter 2. To coordinate a team of real robots during a task that involved symbolic actions, specifically removing an obstacle from a passage, we applied the approach presented in Chapter 3. The data association modules introduced in Chapter 4 were applied to the problem of mapping real-world buildings and to the problem of mapping a large outdoor environment that contained sparse features. Our efficient 3D mapping approach, introduced in Chapter 5, was used to model a number of environments using data collected with real robots. Finally, we conducted extensive real-world experiments to evaluate our approach to detect vegetation. Our experiments show that the approaches presented in this thesis not only advance the theoretical state-of-the-art but that they can be applied under realistic conditions.

7.2. Future Work

The approaches described in this thesis allow teams of robots to map environments more efficiently. However, multi-robot systems continue to offer challenges and research opportunities. In the following, we will describe possible future research directions.

7.2.1. Multi-Robot Coordination

In Chapter 2, we improved robot coordination by analyzing the structure of indoor environments. In future work, additional sources of semantic knowledge could be considered. Robots could, for instance, infer the typical use of rooms by detecting objects. This would allow robots to learn distributions of the expected exploration area in such a room. For example, a restroom (detected from a towel rack) is usually small while a warehouse (detected from a storage shelve) can be large.

Our segmentation-based coordination approach has been evaluated with teams of two to eight robots. If teams were considerably larger, for example, if they included hundreds

of robots, then there would often be more robots than reachable rooms in a building. In such cases, one would have to employ further strategies to distribute robots in the building. Future work could therefore develop strategies to distribute large teams so that the overall efficiency is maximized.

When teams of robots explore an environment, they usually apply a multi-robot SLAM approach to generate a joint map. If no global positioning information is available, the maps of the individual robots need to overlap to solve the data association problem we discussed in Chapter 4. This poses a challenge to the coordination approach since it involves a trade off between exploring new areas and ensuring an overlap between robot maps. It would be interesting to evaluated whether introducing corresponding symbolic actions would lead to an advantage over purely numeric approaches.

Furthermore, it would be interesting to consider heterogeneous teams of exploring robots that consist of ground robots, flying robots, or robotic boats. One would have to answer questions such as: How do exploration targets between those groups of robots relate? Which are the relevant costs and utilities of the robots and how can they be compared? How can a joint map be estimated in such teams?

In the context of disaster recovery, future work should address mixed teams of humans and robots. A coordination approach such as the one proposed in Chapter 3 could provide assistance to the humans while ensuring autonomous operation of the robots. This setting would most likely also include feedback from the human agents and humans would be able to override robot assignments based on their assessment of the situation. The challenges of mixed teams will most likely center around communication: How can information from the coordination system be accessed by the human teammates? How can humans efficiently alter assignments, set out orders to the robots, or introduce new target locations?

7.2.2. Model Estimation

In this thesis, we presented techniques to associate maps of several robots to generate a joint map estimate. These techniques are not specific to the multi-robot case we evaluated. They could also be employed by a single robot to identify previously visited locations during SLAM.

We evaluated our data association methods in the context of 2D mapping with several robots. An interesting future research question would be how to extend our approaches to the case of 3D mapping. For example, it might be possible to apply global localization techniques to associate 3D maps. Landmark- or feature-based approaches could also be applied in 3D, for instance, following up on 3D place recognition systems such as the approach introduced by Steder *et al.* (2011).

We presented an efficient method for 3D mapping. However, our method requires state-of-the-art CPUs to update maps with high-resolution sensors in real-time. In future

work, it would be interesting to explore the possibilities of multi-core systems. In our proposed hierarchy of 3D maps, each map can be updated independently which enables parallel updates. Furthermore, recent generations of GPUs offer exciting opportunities in the case of read-only queries of 3D maps. Especially 3D localization systems could profit from the possiblilty of performing hundreds of ray-casting operations in parallel.

3D segmentation techniques that are based on the distance of objects surfaces are prone to errors when objects are in contact with each other. For example, a cup on a saucer would be treated as the same object instead of two separate objects. An interesting extension for table-top mapping systems would be to verify the 3D segmentation by manipulating objects. For example, a robot that is equipped with a manipulator could grasp the cup and in this way realize that it is a different object than the saucer.

To detect vegetation, we classify laser remission measurements. This exploits the strong difference in reflectivity between vegetation and non-vegetation. However, other materials also differ in their reflectivity. While the difference may not be as pronounced as in the case of vegetation, it might be possible to differentiate other classes. Metal, for instance, strongly reflects near-infrared light and so do certain types of paint. Detecting such materials would provide valuable information for mapping, object detection, or data association techniques.

List of Figures

List of Tables

List of Algorithms

Bibliography

E. Alpaydin. *Introduction To Machine Learning*. MIT Press, 2004.

J. Amanatides and A. Woo. A fast voxel traversal algorithm for ray tracing. In *Proc. of Eurographics*, Amsterdam, The Netherlands, August 1987.

D. Anguelov, R. Biswas, D. Koller, B. Limketkai, S. Sanner, and S. Thrun. Learning hierarchical object maps of non-stationary environments with mobile robots. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI)*, Edmonton, Canada, 2002.

R. Baribeau, M. Rioux, and G. Godin. Color reflectance modeling using a polychromatic laser range sensor. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):263–269, 1992. ISSN 0162-8828.

P. Beeson, N.K. Jong, and B. Kuipers. Towards autonomous topological place detection using the extended voronoi graph. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2005.

M. Berhault, H. Huang, P. Keskinocak, S. Koenig, W. Elmaghraby, P. Griffin, and A. Kleywegt. Robot exploration with combinatorial auctions. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 1957–1962, 2003.

N. Blodow, L.C. Goron, Z.C. Marton, D. Pangercic, T. Ruhr, M. Tenorth, and M. Beetz. Autonomous semantic mapping for robots performing everyday manipulation tasks in kitchen environments. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 4263–4270, 2011.

B. Bonet and H. Geffner. Planning as heuristic search. *Artificial Intelligence*, 129(1): 5–33, 2001.

O. Booij and Z. Zivkovic. The planar two point algorithm. *Technical report of the Intelligent Systems Laboratory Amsterdam*, IAS-UVA-09-05, 2009.

M. Bosse and R. Zlot. Map matching and data association for large-scale two-dimensional laser scan-based slam. *International Journal of Robotics Research*, 27 (6):667–691, 2008.

D. Bradley, R. Unnikrishnan, and J. Bagnell. Vegetation detection for driving in complex environments. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2007.

C.A. Brooks, K. Iagnemma, and S. Dubowsky. Vibration-based terrain analysis for mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 3415–3420, 2005.

E. Brunskill, T. Kollar, and N. Roy. Topological mapping using spectral clustering and classification. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, October 2007.

W. Burgard, M. Moors, C. Stachniss, and F. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–378, 2005.

S. Cambon, R. Alami, and F. Gravot. A hybrid approach to intricate motion, manipulation and task planning. *International Journal of Robotics Research*, 28(1):104–126, 2009.

Y.U. Cao, A.S. Fukunaga, and A.B. Khang. Cooperative mobile robotics: Antecedents and directions. *Journal of Autonomous Robots*, 4(1):7–27, 1997.

S. Carpin. Fast and accurate map merging for multi-robot systems. *Journal of Autonomous Robots*, 25(3):305–316, 2008.

C-C. Chang and C-J. Lin. LIBSVM: a library for support vector machines. http://www.csie.ntu.edu.tw/c̃jlin/libsvm, 2001.

Y. Chen, B. W. Wah, and C.-W. Hsu. Temporal planning using subgoal partitioning and resolution in SGPlan. *Journal of Artificial Intelligence Research*, 26:323–369, 2006.

H. Choset and J. Burdick. Sensor based planning, part i: The generalized voronoi graph. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Nagoya, Japan, 1995.

H. Choset, , and Burdick J. Sensor-based exploration: The hierarchical generalized voronoi graph. *International Journal of Robotics Research*, 19(2), 2000.

M. Ciocarlie, K. Hsiao, E.G. Jones, S. Chitta, R.B. Rusu, and I.A. Sucan. Towards reliable grasping and manipulation in household environments. In *Proc. of RSS 2010 Workshop on Strategies and Evaluation for Mobile Manipulation in Household Environments*, 2010.

W. Cochran, J. Cooley, D. Favin, H. Helms, R. Kaenel, W. Lang, Jr. Maling, G., D. Nelson, C. Rader, and P. Welch. What is the fast fourier transform? *IEEE Transactions on Audio and Electroacoustics*, 15(2):45 – 55, jun 1967.

D.M. Cole and P.M. Newman. Using laser range data for 3D SLAM in outdoor environments. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 1556–1563, 2006.

F. Cordes, I. Ahrns, S. Bartsch, T. Birnschein, A. Dettmann, S. Estable, S. Haase, J. Hilljegerdes, D. Koebel, S. Planthaber, et al. Lunares: lunar crater exploration with heterogeneous multi robot systems. *Intelligent Service Robotics*, pages 1–29, 2011.

A. Cunningham, M. Paluri, and F. Dellaert. Ddf-sam: Fully distributed slam using constrained factor graphs. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 3025–3030, 2010.

H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G. Bradski. Self-supervised monocular road detection in desert terrain. In *Proc. of Robotics: Science and Systems (RSS)*, Philadelphia, USA, 2006.

A. Davison. Real-time simultaneous localization and mapping with a single camera. In *Proc. of European Conf. on Computer Vision (ECCV)*, 2003.

G. Dedeoglu and G.S. Sukhatme. Landmark-based matching algorithm for cooperative mapping by autonomous robots. In *Distributed autonomous robotic systems 4*, 2000.

F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Leuven, Belgium, 1998.

F. Dellaert, T. Balch, M. Kaess, R. Ravichandran, F. Alegre, M. Berhault, R. McGuire, E. Merrill, L. Moshkina, and D. Walker. The Georgia Tech yellow jackets: A marsupial team for urban search and rescue. In *AAAI Mobile Robot Competition Workshop*, 2002.

C. Dornhege and A. Kleiner. A frontier-void-based approach for autonomous exploration in 3d. In *IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, 2011.

C. Dornhege, P. Eyerich, T. Keller, S. Trüg, M. Brenner, and B. Nebel. Semantic attachments for domain-independent planning systems. In *Proc. of the Int. Conf. on Automated Planning and Scheduling (ICAPS)*, pages 114–121, 2009.

B. Douillard, D. Fox, and F. Ramos. Laser and vision based outdoor object mapping. In *Proc. of Robotics: Science and Systems (RSS)*, Zurich, Switzerland, 2008.

B. Douillard, J. Underwood, N. Melkumyan, S. Singh, S. Vasudevan, C. Brunner, and A. Quadros. Hybrid elevation maps: 3D surface models for segmentation. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

A. Drenner and N. Papanikolopoulos. A Framework for Large-Scale Multi-Robot Teams. *Modeling and Control of Complex Systems*, page 297, 2007.

I. Dryanovski, W. Morris, and J. Xiao. Multi-volume occupancy grids: An efficient probabilistic 3d mapping model for micro aerial vehicles. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 1553–1559, 2010.

T. Duckett, S. Marsland, and J. Shapiro. Learning globally consistent maps by relaxation. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, volume 4, pages 3841–3846, 2000.

G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. Robotic exploration as graph construction. *IEEE Transactions on Robotics and Automation*, 7(6):859–865, 1991.

G. Dudek, M.. Jenkin, E. Milios, and D. Wilkes. A taxonomy for multi-agent robotics. *Journal of Autonomous Robots*, 3(4):375–397, 1996.

E.M. DuPont, R.G. Roberts, C.A. Moore, M.F. Selekwa, and E.G. Collins. Online terrain classification for mobile robots. In *Proc. of the Int. Mechanical Engineering Congress and Exposition Conference (IMECE)*, Orlando, USA, 2005.

S. Dutta Roy, S. Chaudhury, and S. Banerjee. Active recognition through next view planning: a survey. *Pattern Recognition*, 37(3):429–446, 2004.

T. Edlinger and E. von Puttkamer. Exploration of an indoor-environment by an autonomous mobile robot. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 1278 –1284, sep 1994.

European Space Agency ESA. ESA's lunar robotics challenge website. http://www.esa.int/esaCP/SEMGAASHKHF_index_0.html, 2008.

R. Eustice, H. Singh, and J.J. Leonard. Exactly sparse delayed-state filters. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2428–2435, Barcelona, Spain, 2005.

P. Eyerich, R. Mattmüller, and G. Röger. Using the context-enhanced additive heuristic for temporal and numeric planning. In *Proc. of the Int. Conf. on Automated Planning and Scheduling (ICAPS)*, pages 130–137, 2009.

N. Fairfield, G.A. Kantor, and D. Wettergreen. Real-time SLAM with octree evidence grids for exploration in underwater tunnels. *Journal of Field Robotics*, 2007.

M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Graphics and Image Processing*, 1981.

J. Fournier, B. Ricard, and D. Laurendeau. Mapping and exploration of complex environments using persistent 3D model. In *Proc. of the Conf. on Computer and Robot Vision*, pages 403–410, 2007.

M. Fox and D. Long. PDDL2.1: an extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research (JAIR)*, 20(1):61–124, 2003.

U. Frese. Treemap: An $o(logn)$ algorithm for indoor simultaneous localization and mapping. *Journal of Autonomous Robots*, 21(2):103–122, 2006.

U. Frese and L. Schroder. Closing a million-landmarks loop. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 5032–5039, 2006.

U. Frese, P. Larsson, and T. Duckett. A multilevel relaxation algorithm for simultaneous localisation and mapping. *IEEE Transactions on Robotics*, 21(2):1–12, 2005.

S. Friedman, H. Pasula, and D. Fox. Voronoi random fields: Extracting topological structure of indoor environments via place labeling. In Manuela M. Veloso, editor, *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 2109–2114, 2007.

Matteo Frigo and Steven G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005.

C. Galindo, A. Saffiotti, S. Coradeschi, P. Buschka, JA Fernández-Madrigal, and J. González. Multi-hierarchical semantic maps for mobile robotics. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2005.

Georgia Tech Research Corporation. Georgia tech smoothing and mapping library (gtsam). https://collab.cc.gatech.edu/borg/gtsam/, 2010.

A. Gerevini, A. Saetti, and I. Serina. An approach to efficient planning with numerical fluents and multi-criteria plan quality. *Artificial Intelligence.*, 172(8-9):899–944, 2008.

B.P. Gerkey and M.J. Matarić. Sold!: Auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation*, 18(5):758– 768, 2002.

B.P. Gerkey and M.J. Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *International Journal of Robotics Research*, 23(9):939, 2004.

S. Gottschalk. *Collision Queries using Oriented Bounding Boxes*. PhD thesis, The University of North Carolina, 2000.

R. Grabowski, L.E. Navarro-Serment, C.J.J. Paredis, and P.K. Khosla. Heterogeneous teams of modular robots for mapping and exploration. *Journal of Autonomous Robots*, 8(3):293–308, 2000.

G. Grisetti, S. Grzonka, C. Stachniss, P. Pfaff, and W. Burgard. Efficient estimation of accurate maximum likelihood maps in 3D. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007a.

G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46, 2007b.

G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard. A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In *Proc. of Robotics: Science and Systems (RSS)*, Atlanta, GA, USA, 2007c.

G. Grisetti, C. Stachniss, and W. Burgard. Non-linear constraint network optimization for efficient map learning. *IEEE Transactions on Intelligent Transportation Systems*, 10(3), 2009.

J.E. Guivant and E.M. Nebot. Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *IEEE Transactions on Robotics and Automation*, 17(3):242–257, 2001.

J.-S. Gutmann, M. Fukuchi, and M. Fujita. 3D perception and environment map generation for humanoid robot navigation. *International Journal of Robotics Research*, 27 (10):1117–1134, 2008.

D. Guzzoni, A. Cheyer, L. Julia, and K. Konolige. Many robots make short work. *AI Magazine*, 18(1):55–64, 1997.

M. Habbecke and L. Kobbelt. A surface-growing approach to multi-view stereo reconstruction. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2007.

R. Hadsell, J. Andrew Bagnell, and M. Hebert. Accurate rough terrain estimation with space-carving kernels. In *Proc. of Robotics: Science and Systems (RSS)*, 2009.

D. Hähnel. *Mapping with mobile robots*. PhD thesis, Universitätsbibliothek Freiburg, 2005.

M. Hebert and N. Vandapel. Terrain classification techniques from ladar data for autonomous navigation. In *Proc. of the Collaborative Technology Alliances conference*, College Park, MD., 2003.

M. Hebert, C. Caillas, E. Krotkov, I. S. Kweon, and T. Kanade. Terrain mapping for a roving planetary explorer. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, volume 2, pages 997–1002, May 1989.

M. Helmert. Decidability and undecidability results for planning with numerical state variables. In *Proc. of the Int. Conf. on Automated Planning and Scheduling (ICAPS)*, pages 44–53, 2002.

M. Helmert. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

M. Helmert and H. Geffner. Unifying the causal graph and additive heuristics. In *Proc. of the Int. Conf. on Automated Planning and Scheduling (ICAPS)*, volume 8, 2008.

L. Heng, L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys. Autonomous obstacle avoidance and maneuvering on a vision-guided mav using on-board processing. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2472–2477, 2011.

P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments. In *Proc. of the Int. Symposium on Experimental Robotics (ISER)*, 2010.

C. Hertzberg, R. Wagner, O. Birbach, T. Hammer, and U. Frese. Experiences in building a visual slam system from open source components. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2644–2651, 2011.

J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

B. Höfle and N. Pfeifer. Correction of laser scanning intensity data: Data and model-driven approaches. *ISPRS Journal of Photogrammetry and Remote Sensing*, 62(6): 415–433, 2007.

D. Holz, N. Basilico, F. Amigoni, and S. Behnke. A comparative evaluation of exploration strategies and heuristics to improve them. In *Proc. of the European Conf. on Mobile Robots (ECMR)*, Oerebro, Sweden, September 2011.

J. Hopcroft and R. Tarjan. Algorithm 447: efficient algorithms for graph manipulation. *Communications of the ACM*, 16(6):372–378, 1973.

Armin Hornung, Kai M. Wurm, and Maren Bennewitz. Humanoid robot localization in complex indoor environments. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, 2010.

A. Howard and N. Roy. The robotics data set repository (Radish), 2003. http://radish.sourceforge.net/.

A. Howard, M.J. Matarić, and S. Sukhatme. An incremental deployment algorithm for mobile robot teams. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2849–2854, Lausanne, Switzerland, 2002.

Andrew Howard. Multi-robot simultaneous localization and mapping using particle filters. In *Robotics: Science and Systems*, pages 201–208, Cambridge, MA, USA, 2005.

Henry Hsu and Peter A. Lachenbruch. *Paired t Test*. John Wiley & Sons, Inc., 2007. ISBN 9780471462422.

W.H. Huang and K.R. Beevers. Topological map merging. *International Journal of Robotics Research*, 24(8):601–613, 2005.

B. Huhle, T. Schairer, A. Schilling, and W. Strasser. Optimal alignment of 3d data for spatial discretization. In *Proc. of the 2011 Canadian Conference on Computer and Robot Vision*, CRV '11, pages 355–362, 2011.

L. Iocchi, D. Nardi, M. Piaggio, and A. Sgorbissa. Distributed coordination in heterogeneous multi-robot systems. *Journal of Autonomous Robots*, 15(2):155–168, 2003.

L. Iocchi, S. Pellegrini, and G. D. Tipaldi. Building multi-level planar maps integrating LRF, stereo vision and IMU sensors. In *Proc. of IEEE International Workshop on Safety, Security and Rescue Robotics (SSRR'07)*, Rome, Italy, 2007.

D. Joho, C. Stachniss, P. Pfaff, and W. Burgard. Autonomous exploration for 3d map learning. *Autonome Mobile Systeme 2007*, pages 22–28, 2007.

E.G. Jones, M.B. Dias, and A. Stentz. Time-extended multi-robot coordination for domains with intra-path constraints. *Autonomous robots*, 30(1):41–56, 2011.

S. Julier, J. Uhlmann, and H. Durrant-Whyte. A new approach for filtering nonlinear systems. In *Proc. of the American Control Conference*, pages 1628–1632, Seattle, WA, USA, 1995.

E. Kadioglu and N. Papanikolopoulos. A method for transporting a team of miniature robots. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2297–2302, 2003.

L.P. Kaelbling and T. Lozano-Perez. Hierarchical task and motion planning in the now. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2011.

M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, Dec 2008.

M. Kaess, V. Ila, R. Roberts, and F. Dellaert. The Bayes tree: An algorithmic foundation for probabilistic robot mapping. In *Intl. Workshop on the Algorithmic Foundations of Robotics*, Dec 2010.

J. Kiener and O. Von Stryk. Towards cooperation of heterogeneous, autonomous robots: A case study of humanoid and wheeled robots. *Journal of Robotics & Autonomous Systems*, 58(7):921–929, 2010.

J. Ko, B. Stewart, D. Fox, K. Konolige, and B. Limketkai. A practical, decision-theoretic approach to multi-robot mapping and exploration. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 3232–3238, Las Vegas, NV, USA, 2003.

S. Koenig, B. Szymanski, and Y. Liu. Efficient and inefficient ant coverage methods. *Annals of Mathematics and Artificial Intelligence*, 31:41–76, 2001.

M. Koes, I. Nourbakhsh, and K. Sycara. Heterogeneous multirobot coordination with spatial and temporal constraints. In *Proc. of the National Conf. on Artificial Intelligence*, page 1292, 2005.

J.R. Kok, M.T.J. Spaan, and N. Vlassis. Non-communicative multi-robot coordination in dynamic environments. *Journal of Robotics & Autonomous Systems*, 50(2-3):99–114, 2005.

K. Konolige. Large-scale map-making. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, pages 457–463, 2004.

G.K. Kraetzschmar, G.P. Gassull, and K. Uhl. Probabilistic quadtrees for variable-resolution mapping of large environments. In M. I. Ribeiro and Santos J. Victor, editors, *Proc. of the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles*, Lisbon, Portugal, July 2004.

M. Krainin, P. Henry, X. Ren, and D. Fox. Manipulator and object tracking for in hand model acquisition. In *Proc. of the Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation at the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2010.

H.W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1):83–97, 1955.

B. Kuipers and Y.-T. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics & Autonomous Systems*, 8: 47–63, 1991.

J-F. Lalonde, N. Vandapel, D. Huber, and M. Hebert. Natural terrain classification using three-dimensional ladar data for ground robot mobility. *Journal of Field Robotics*, 23 (10):839 – 861, 2006.

D.T. Lee and B.J. Schachter. Two algorithms for constructing a delaunay triangulation. *International Journal of Parallel Programming*, 9(3):219–242, 1980.

J.J. Leonard and H.F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(4):376–382, 1991.

D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Journal of Autonomous Robots*, 4:333–349, 1997.

J. Macedo, R. Manduchi, and L. Matthies. Ladar-based discrimination of grass from obstacles for autonomous navigation. In *Proc. of the Int. Symposium on Experimental Robotics (ISER)*, London, UK, 2001.

D.J.C. MacKay. *Information theory, inference, and learning algorithms*. Cambridge Univ Press, 2003.

R. Manduchi. Bayesian fusion of color and texture segmentations. In *Proc. of the Int. Conf. on Computer Vision (ICCV)*, page 956, 1999.

R. Manduchi, A. Castano, A. Talukder, and L.Matthies. Obstacle detection and terrain classification for autonomous off-road navigation. *Journal of Autonomous Robots*, pages 81–102, 2003.

D. Meagher. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19(2):129–147, 1982.

A. Meijster, J.B.T.M. Roerdink, and W.H. Hesselink. *Mathematical Morphology and its Applications to Image and Signal Processing*, chapter A General Algorithm for Computing Distance Transforms in Linear Time, pages 331–340. Kluwer Academic Publishers, 2000.

M. Montemerlo and S. Thrun. Simultaneous localization and mapping with unknown data association using FastSLAM. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 1985–1991, Taipei, Taiwan, 2003.

M. Montemerlo, N. Roy, S. Thrun, D. Hähnel, C. Stachniss, and J. Glover. Carmen robot navigation toolkit. http://carmen.sourceforge.net, 2002.

H. Moravec. Robot spatial perception by stereoscopic vision and 3D evidence grids. Technical Report CMU-RI-TR-96-34, Robotics Institute, Pittsburgh, PA, September 1996.

H.P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, pages 61–74, 1988.

H.P. Moravec and A.E. Elfes. High resolution maps from wide angle sonar. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 116–121, St. Louis, MO, USA, 1985.

J. Müller, N. Kohler, and W. Burgard. Autonomous miniature blimp navigation with online motion planning and re-planning. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

M. Müller, H. Surmann, K. Pervolz, and S. May. The accuracy of 6d slam using the ais 3d laser scanner. In *IEEE Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems*, 2006.

R.R. Murphy, M. Ausmus, M. Bugajska, T. Ellis, T. Johnson, N. Kelley, J. Kiefer, and L. Pollock. Marsupial-like mobile robot societies. In *Proc. of the Annual Conf. on Autonomous Agents*, page 365, 1999.

R.B. Myneni, F.G. Hall, P.J. Sellers, and A.L. Marshak. The interpretation of spectral vegetation indexes. *IEEE Transactions on Geoscience and Remote Sensing*, 33(2): 481–486, 1995.

C.W. Niblack, D.W. Capson, and P.B. Gibbons. Generating skeletons and centerlines from the medial axis transform. *Proc. of the Int. Conf. on Pattern Recognition*, i:881–885, 1990.

A. Nüchter and J. Hertzberg. Towards semantic maps for mobile robots. *Journal of Robotics & Autonomous Systems*, 56(11):915–926, 2008.

A. Nüchter, O. Wulf, K. Lingemann, J. Hertzberg, B. Wagner, and H. Surmann. 3D mapping with semantic knowledge. *RoboCup 2005: Robot Soccer World Cup IX*, pages 335–346, 2006.

A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. 6d slam—3d mapping outdoor environments. *Journal of Field Robotics*, 24(8-9):699–722, 2007.

H. Ogawa. Labeled point pattern matching by delaunay triangulation and maximal cliques. *Pattern Recognition*, 19:35–40, January 1986. ISSN 0031-3203.

E. Olson, J. Leonard, and S. Teller. Fast iterative optimization of pose graphs with poor initial estimates. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2262–2269, 2006.

S. Oßwald, A. Gorog, A. Hornung, and M. Bennewitz. Autonomous climbing of spiral staircases with humanoids. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 4844–4849, 2011.

M.A. Paskin. Thin junction tree filters for simultaneous localization and mapping. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 1157–1164, Acapulco, Mexico, 2003.

K. Pathak, A. Birk, J. Poppinga, and S. Schwertfeger. 3D forward sensor modeling and application to occupancy grid based sensor fusion. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2059–2064, 2007.

P. Payeur, P. Hebert, D. Laurendeau, and C.M. Gosselin. Probabilistic octree modeling of a 3-d dynamic environment. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1997.

A. Petrovskaya and A.Y. Ng. Probabilistic mobile manipulation in dynamic environments, with application to opening doors. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, 2007.

J.C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74, 1999.

S. Richter and M. Westphal. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research (JAIR)*, 39:127–177, 2010.

R. Roberts, S.N. Sinha, R. Szeliski, and D. Steedly. Structure from motion for scenes with large duplicate structures. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3137–3144, 2011.

Y. Roth-Tabak and R. Jain. Building an environment model using depth information. *Computer*, 22(6):85–90, June 1989.

M. Ruhnke, B. Steder, G. Grisetti, and W Burgard. Unsupervised learning of 3d object models from partial views. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2009.

S.J. Russell and P. Norvig. *Artificial intelligence: a modern approach*. Prentice hall, 2010.

R.B. Rusu, Z.C. Marton, N. Blodow, M. Dolha, and M. Beetz. Towards 3D Point cloud based object maps for household environments. *Journal of Robotics & Autonomous Systems*, 56(11):927–941, 2008.

R.B. Rusu, G. Bradski, R. Thibaux, and J. Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

J. Ryde and H. Hu. 3D mapping with multi-resolution occupied voxel lists. *Journal of Autonomous Robots*, pages 1–17, 2010.

D. Sadhukhan, C. Moore, and E. Collins. Terrain estimation using internal sensors. In *Proc. of the IASTED Int. Conf. on Robotics and Applications*, Honolulu, Hawaii, USA, 2004.

B Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, 2002.

R. Shade and P. Newman. Choosing where to go: Complete 3d exploration with stereo. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2806–2811, 2011.

Jonathan Richard Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In Ming C. Lin and Dinesh Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, May 1996.

G. Sibley, C. Mei, I. Reid, and P. Newman. Adaptive relative bundle adjustment. In *Proc. of Robotics: Science and Systems (RSS)*, 2009.

K. Singh and K. Fujimura. Map making by cooperating mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 254–259, Atlanta, GA, USA, 1993a.

Karansher Singh and Kikuo Fujimura. Map making by cooperating mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 254–259, Atlanta, GA, USA, 1993b.

M. Smith, I. Baldwin, W. Churchill, R. Paul, and P. Newman. The new college vision and laser data set. *International Journal of Robotics Research*, 28(5):595–599, May 2009.

C. Stachniss. *Exploration and Mapping with Mobile Robots*. PhD thesis, University of Freiburg, Department of Computer Science, April 2006.

C. Stachniss. *Robotic Mapping and Exploration*, volume 55 of *STAR Springer tracts in advanced robotics*. Springer, 2009.

C. Stachniss and G. Grisetti. GMapping project at OpenSLAM.org. http://openslam.org, 2007.

C. Stachniss, G. Grisetti, and W. Burgard. Information gain-based exploration using rao-blackwellized particle filters. In *Proc. of Robotics: Science and Systems (RSS)*, pages 65–72, Cambridge, MA, USA, 2005.

C. Stachniss, O. Martínez-Mozos, and W. Burgard. Speeding-up multi-robot exploration by considering semantic place information. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 1692–1697, Orlando, FL, USA, 2006.

C. Stachniss, M. Bennewitz, G. Grisetti, S. Behnke, and W. Burgard. How to learn accurate grid maps with a humanoid. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2008.

B. Steder and R. Kümmerle. Freiburg campus 360 degree 3d scans. http://ais.informatik.uni-freiburg.de/projects/datasets/fr360/, 2010.

B. Steder, M. Ruhnke, S. Grzonka, and W. Burgard. Place recognition in 3D scans using a combination of bag of words and point feature based relative pose estimation. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

T. Stoyanov, M. Magnusson, H. Almqvist, and A.J. Lilienthal. On the accuracy of the 3d normal distributions transform as a tool for spatial representation. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 4080–4085, 2011.

M. Strand and R. Dillmann. Using an attributed 2D-grid for next-best-view planning on 3D environment data for an autonomous robot. In *Int. Conf. on Information and Automation (ICIA)*, 2008.

H. Surmann, A. Nüchter, and J. Hertzberg. An autonomous mobile robot with a 3d laser range finder for 3d exploration and digitalization of indoor environments. *Robotics and Autonomous Systems*, 45(3):181–198, 2003.

Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition, Third Edition*. Academic Press, 3 edition, March 2006.

S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

S. Thrun and Y. Liu. Multi-robot slam with sparse extended information filers. *International Journal of Robotics Research*, pages 254–266, 2005.

S. Thrun, Y. Liu, D. Koller, A.Y. Ng, Z. Ghahramani, and H. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *International Journal of Robotics Research*, 23(7/8):693–716, 2004.

S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

Sebastian Thrun. Exploration and model building in mobile robot domains. In *Proc. of the ICNN-93*, pages 175–180, San Francisco, CA, March 1993. IEEE Neural Network Council.

R. Triebel, P. Pfaff, and W. Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

V.N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

T. Weise, T. Wismer, B. Leibe, and L. Van Gool. In-hand scanning with online loop closure. In *ICCV Workshops*, 2009.

C. Weiss, H. Frohlich, and A. Zell. Vibration-based terrain classification using support vector machines. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

C. Weiss, N. Fechner, M. Stark, and A. Zell. Comparison of different approaches to vibration-based terrain classification. In *Proc. of the European Conf. on Mobile Robots (ECMR)*, Freiburg, Germany, 2007.

C. Wellington, A. Courville, and A. Stentz. A generative model of terrain for autonomous navigation in vegetation. *International Journal of Robotics Research*, 25(12):1287 – 1304, 2006.

P. Whaite and F.P. Ferrie. Autonomous exploration: Driven by uncertainty. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):193–205, 1997.

J. Wilhelms and A. Van Gelder. Octrees for faster isosurface generation. *ACM Transactions on Graphics (TOG)*, 11(3):201–227, 1992.

D.F. Wolf, G. Sukhatme, D. Fox, and W. Burgard. Autonomous terrain mapping and classification using hidden markov models. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2005.

B. Yamauchi. A frontier based approach for autonomous exploration. In *Proc. of the IEEE Int. Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, pages 146–151, 1997.

B. Yamauchi. Frontier-based exploration using multiple robots. In *Proc. of the Int. Conf. on Autonomous Agents*, pages 47–53, May 1998.

M. Yguel, O. Aycard, and C. Laugier. Update policy of dense maps: Efficient algorithms and sparse representation. In *Proc. of the Int. Conf. on Field and Service Robots (FSR)*, volume 42, pages 23–33, 2007a.

M. Yguel, C. Tay Meng Keat, C. Braillon, C. Laugier, and O. Aycard. Dense mapping for range sensors: Efficient algorithms and sparse representations. In *Proc. of Robotics: Science and Systems (RSS)*, June 2007b.

B. Zhao, L. Tian, and T. Ahamed. Real-time ndvi measurement using a low-cost panchromatic sensor for a mobile robot platform. *Environment Control in Biology*, 48(2):73–79, 2010.

X.S. Zhou and S.I. Roumeliotis. Multi-robot slam with unknown initial correspondence: The robot rendezvous case. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 1785–1792, 2006.

Z. Zivkovic, B. Bakker, and B. Kröse. Hierarchical map building and planning based on graph partitioning. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 803–809, 2006.

R. Zlot, A.T. Stenz, M.B. Dias, and S. Thayer. Multi-robot exploration controlled by a market economy. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2002.