Institut für Informatik
der Technischen Universität München

Region & Gateway Mapping: Acquiring Structured and Object-Oriented Representations of Indoor Environments

Dissertation

Derik Schröter

Institut für Informatik
der Technischen Universität München

Region & Gateway Mapping: Acquiring Structured and Object-Oriented Representations of Indoor Environments

Derik Schröter

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Alfons Kemper, Ph.D.
Prüfer der Dissertation:	1.	Univ.-Prof. Dr. Bernd Radig
	2.	Univ.-Prof. Dr.-Ing. Darius Burschka
	3.	Univ.-Prof. Dr. Wolfram Burgard, Albert-Ludwigs-Universität Freiburg

Die Dissertation wurde am 10.11.2005 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 29.03.2006 angenommen.

Abstract

State-of-the-art robot mapping approaches are capable of acquiring impressively accurate 2D and 3D models of their environments. To the best of our knowledge, few of them represent structure or acquire models of task-relevant objects. In this work, a new approach to mapping of indoor environments is presented, in which the environment structure in terms of regions and gateways is automatically extracted, while the robot explores. Objects, both in 2D and 3D, are modeled explicitly in those maps and allow for robust localization. We refer to those maps as structured object-oriented environment representations or **R***egion & * **G***ateway Maps (RG Maps).* The process of building such maps is called **R***egion & * **G***ateway Mapping (RG Mapping).*

RG Maps and RG Mapping make several contributions to the field of map building of indoor environments. First, RG Mapping automatically recovers the structure of large classes of indoor environments and represents them explicitly. Therefore, novel algorithms have been developed for the detection and recognition of gateways, i.e. transitions between regions. Second, it detects rectangular 2D/3D objects using laser range as well as image data that are used for gateway detection and localization. Third, the semantic description, i.e. annotation of regions and objects, is obtained from human-machine interaction in the context of task assignment. Fourth, a compact Region & Gateway Graph is easily extracted from RG Maps, and it is used for efficient path planning on the global scale. It allows reasoning about the feasibility of a given path and learning of the properties of path segments while the robot moves through the environment. Fifth, due to the clustering of metric data into regions, region-based localization and path planning only need to consider the respective region data. Thus, the complexity of the data association problem for localization and the search space for metric path planning is substantially reduced.

The RG Mapping and Navigation System has been fully implemented as a distributed (module-based) system, and runs in real-time on a real robot. Due to its architecture, the system can be easily ported to different platforms, provided that similar sensor data is available. In order to support reliable exploration of and navigation within RG Maps, the presented system features a novel approach to collision avoidance and low-level control. The low-level control allows for very precise and fast pursuing of short trajectory segments, which can be changed at any time. The collision avoidance generates trajectory segments based on the interpretation of the current sensor data and short-distance targets from the path planning process. As a result, the navigation behaviour of the robot is reproducible and predictable, which is a very desirable feature for high-level planning.

Acknowledgements

The opportunity to conduct research in robotics and vision in the Intelligent Autonomous Systems Group at the Technische Universität München turned out to be an exciting and pleasant experience. This has been effected mainly by the friendly atmosphere and the scientific spirit in the department. Needless to mention, the work presented in this thesis has benefited greatly from the input and support of numerous people inside and outside of the group, and therefore I want to express my deepest gratitude to everyone involved.

I want to thank Prof. Bernd Radig for the confidence and trust he placed in me, by facilitating my research position and financial support. After defining the general direction for my research, he always encouraged me to formulate my own ideas and agenda and gave me the motivation and freedom to shape the project as it progressed. Without the unlimited support of Prof. Michael Beetz I might be sitting here facing only a bunch of good ideas. In uncounted fruitful and sometimes very lively discussions he helped me to form my research, and there was always time for friendly and benevolent motivations. Most of all his many comments on my papers, presentations and finally my thesis have taught me how to make my writing tell a story and how ideas can form a research project. I have learned not just to explain, but to convince. Thank you, Michael.

Thanks to Prof. Darius Burschka for serving as a member of my dissertation committee, as well as Prof. Wolfram Burgard who has been a source of inspiration for both my work and my writing. In particular, I want to thank him for reviewing my thesis and the time we shared in several discussions about related subjects. Likewise, my thanks go to Dirk Hähnel, Steffen Gutmann, Patrik Beeson, Michael Thielscher and all the others I was glad to meet at Conferences, Workshops and other occasions. Thanks for your critical reflections and helpful comments. I want to express special gratitude to Maria Fox and Boris Flach not only for technical discussions but also for long conversations about life and research in general and the good red wine that went along with it. They contributed more than they possibly know to my motivation and well-being.

I did not mean to forget about my past and present colleagues and students in the department. After all, they have been a tremendous help and support in the everyday work. First and foremost I want to thank Freek Stulp and Andreas Hofhauser for critically reflecting on my ideas and constantly encouraging me to continue (and to finally finish). They have become good friends to me, and I sure like to remember the time Freek and me played in that great rock band (www.theunknownhost.com). Andreas was among the students that contributed essentially to the implementation of the work presented here, others are Jan Bandouch, Stefan Plafka, Thomas Weber, Stefan Hinterstoisser and Andreas Iizuka. Their help has been very much appreciated. Further thanks go to Kajetan Berlinger and

Heiko Gottschling who seem to know everything about programming, scripting and useful tools. Thanks also go to Robert Hanek, Thorsten Schmitt, Sebastian Buck and Christoph Hansen for help and support throughout my first years as a research assistant. No system runs without administration - special thanks to Oliver Bösl, Quirin Lohr, Thomas Engel and Christoph Scheich for their patience and support regarding all matters of computer hard- and software.

Finally, I want to thank Freek Stulp (again), Lars Witta, Frank Althof and Radu Bogdan Rusu for reviewing my thesis and giving improving comments. And last but not least I want to express my very sincere gratitude to my friends and my sister, the ones that make life worthwhile beyond work and research. Special thanks to Dominick, Bianca, Andreas K., Joe, Janine, Andreas H., Freek, Radu, Lars, Eric, Jochen, Franz, Vincent and Rayk.

There are things known and things unknown
and in between are The Doors.

Jim Morrison

Contents

Chapter 1

Introduction

Designing mobile autonomous and intelligent systems has been a research objective for over thirty years. But since even longer, people have been carrying and expressing ideas about intelligent machines that have an understanding of their environment and that are able to execute a given task reliably, efficiently and accurately. The respective research fields investigate the problems of perception, the control of dynamics (mobile robot, manipulator), the generation of plans for complex tasks (navigate to pick up an object, parallel user requests), the immediate response to unforeseen situations, as well as safety awareness towards humans, technical equipment and the environment in general. The resulting systems were always meant to support humans in the industrial fabrication of a variety of products (machinery, electronic devices, furniture etc.), in the working environment (delivery or carriage tasks), and in daily life, in particular assisting elderly or handicapped people.

As a consequence, automation was successfully applied to many different fields in the industry in the last century, even though the systems have not been very adaptive. They were restrictively programmed to perform considerably simple tasks as fast and as accurate as possible, without noticing too much in their periphery. But in the last two decades, the research focus moved more and more towards the development of intelligent systems that can autonomously adapt to different tasks and respond deliberately to unexpected situations. Most of the applications in industry are related to immobile manipulators that pick up, place and paint parts, drill holes, grind, saw etc. In automated warehouses mobile vehicles drive autonomously along rails or magnetic leads and perform delivery tasks. In the end of the nineties, mobile robots have guided visitors autonomously through crowded museums [110, 14, 108] and today similar technology can be bought or rent[1]. Additionally, autonomous and intelligent systems have proven to be applicable and successful in space missions [6, 86, 119].

[1]see for example http://www.bluebotics.com

In order to navigate reliably through an environment and plan the assigned tasks properly, an autonomous mobile platform needs to have a model of its surroundings and relevant or useful objects within it. In robotics acquiring this knowledge is commonly referred to as map building and navigation. Different sensors like odometry, sonar, laser range finder, cameras and others are used to capture the characteristics of the environment. Appropriate algorithms have to deal with the problem of inherent sensor noise and ambiguities, i.e. a high degree of uncertainty. Research topics in mobile robotics also seek to answer the questions of localization with respect to a given map, path planning, collision avoidance and control of robot dynamics.

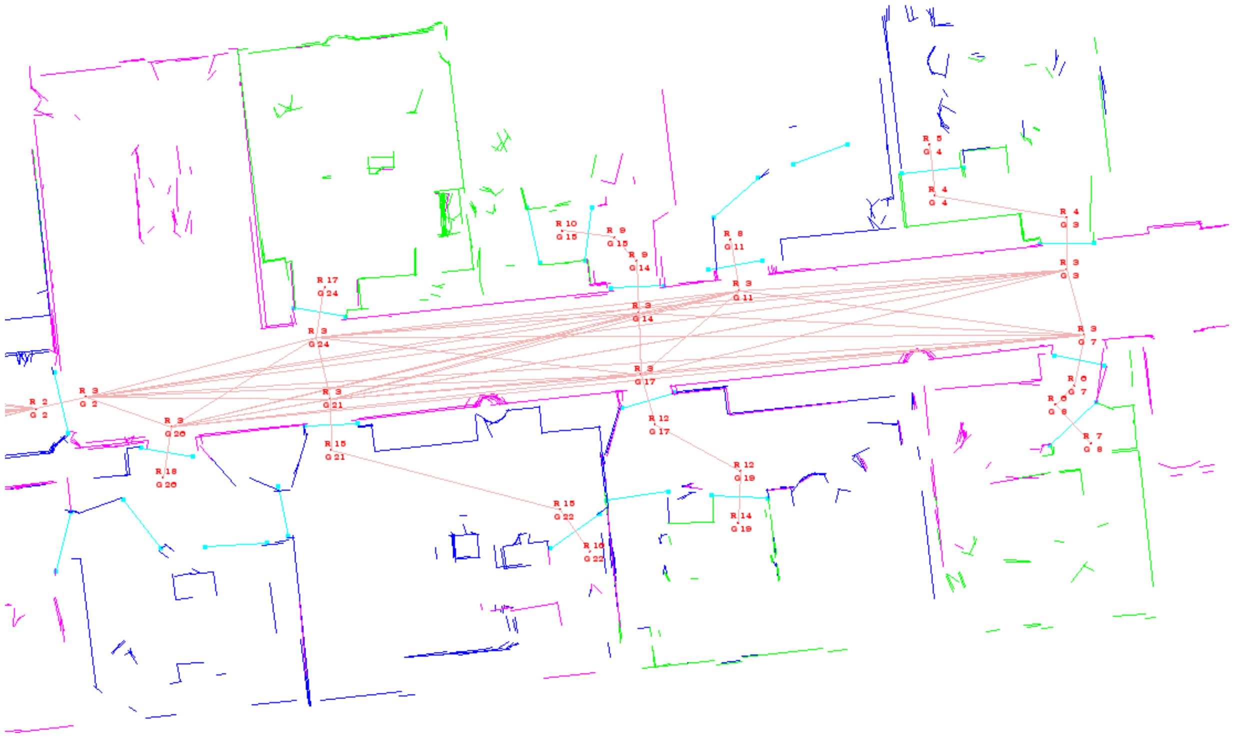


Figure 1.1: Detailed section of the RG Map of our department floor at TUM IX (Garching, Munich), see also Figure 1.2

The mapping and navigation system presented in this work focuses on building structured maps that contain 2D/3D objects. Figure 1.2 (upper right part) shows a robot that drives along a hallway and keeps track of its position despite severe odometry errors. To generate the complete map of our department floor the robot traveled about 350 meters and collected 1248 laser scans. The mapping process works on-line, that is, the system detects doors, hallway turns and crossings and upon traversing them, it starts mapping the new region (office, hallway, lobby). Thus, the sensor data obtained in an environment is continuously clustered according to the observed regions. A rough estimate of the region description is generated while the robot explores a region. Upon leaving that region, the acquired sensor data is interpreted to build a more compact and accurate representation.

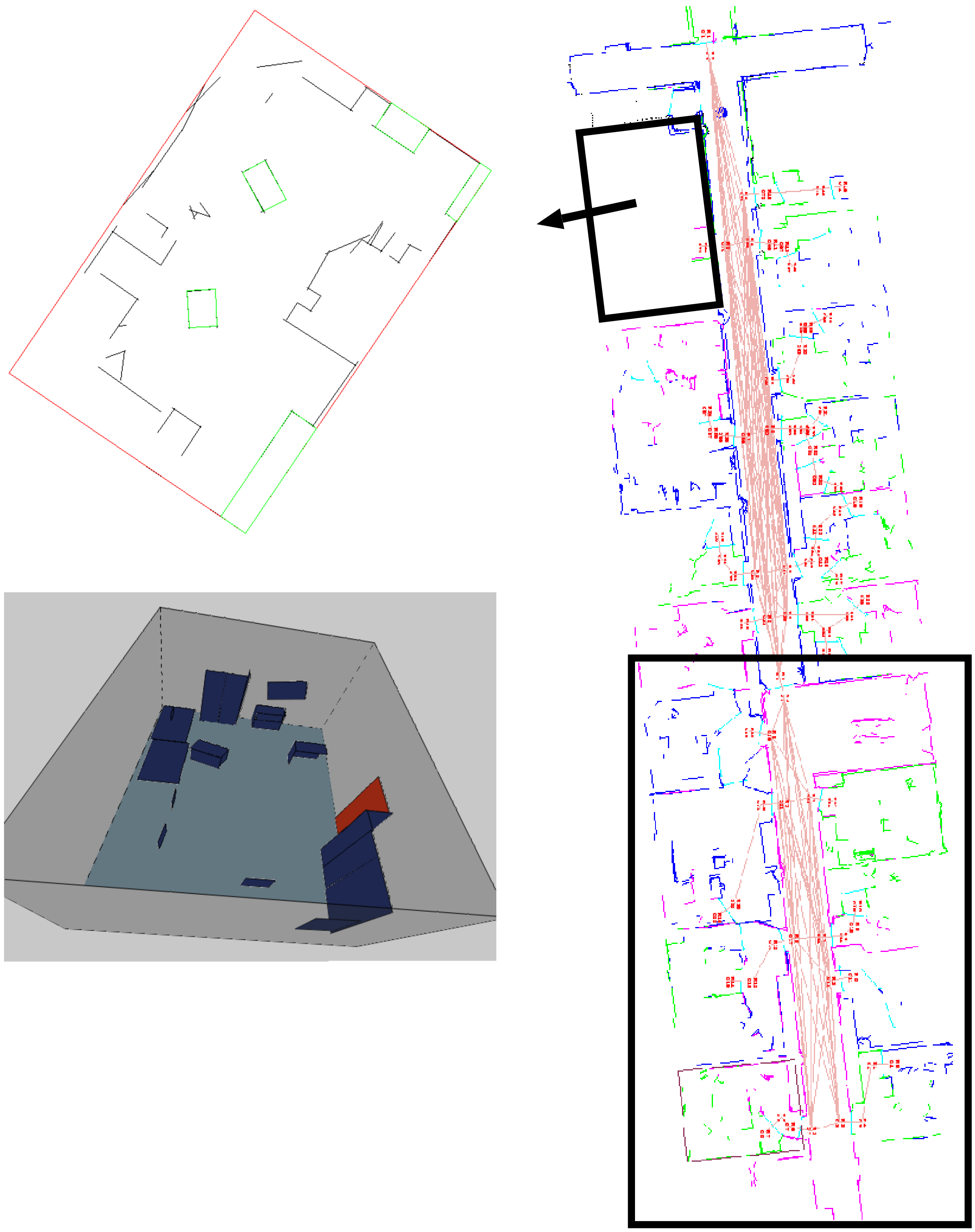


Figure 1.2: RG map of our department floor at TUM IX (Garching, Munich). Raw data: 1248 laser scans, i.e. over 200 000 point measurements. The left images show the region description in 2D and 3D. The lower part marked with a rectangle is enlarged in Figure 1.1.

Such a description, among others, comprises a 2D line segment map and 2D/3D objects for localization and path planning, as well as a list of adjacent gateways. Regions and 3D objects can be used for either referencing them explicitly, e.g. "Go to the closet.", or interaction, e.g. "Fetch the book from the table.".

Figure 1.2 (left) shows how an office environment is represented as a *Region & Gateway Map*. The lower section is depicted in more detail in Figure 1.1. The map comprises objects that describe the nineteen offices and two hallways by means of 41 regions and 58 gateways. Thereby nineteen gateways have either not yet been traversed or belong to the same region, i.e. they are only useful for localization. Since vision was not used in the depicted experiment, the system detected not only doors as gateways but narrow passages in general. The representation of an office contains only the information gathered within that office, and the doors describe possible transitions between those entities. We refer to the before mentioned entities as *Regions* and to the transitions as *Gateways*. Together they form a **R***egion &* **G***ateway Map* (RG Map). The process of building such maps is called **R***egion &* **G***ateway Mapping* (RG Mapping).

In the context of navigation, RG Maps have several advantages compared to state-of-the-art approaches such as pure metrical [107] or topological maps [22, 121, 94]. This is partly due to the fact that RG Maps present a mixture of both, but in detail differ considerably from classic hybrid approaches [109, 113, 112]. First, a compact *Region & Gateway Graph* can be easily extracted from RG Maps, see Figure 1.1. It is used to plan a path on region and gateway level. In the description, costs for gateway or region traversal are explicitly represented, and thus, can be easily adapted, i.e. automatically learned when navigating through the environment. Fast graph search algorithms allow for real-time reasoning on the level of regions and gateways. Second, global localization can be performed based on regions, instead of using the latest observation together with all acquired metric information. That means, when the system does not know in which region it is located, it explores the current region, i.e. builds a region description, generates a compact feature vector and performs region recognition on that level. After the localization has converged to a single region, it only needs to determine the relation of the current observation to the respective region data. Thus, the dimension of the data association problem is strongly reduced, because only a subset of the map data has to be considered. Thereby, localization gives additional proof for gateway traversal by projecting particles into adjacent regions, when the robot approaches a gateway.

RG Maps are applicable to a wide range of indoor environments, because buildings, in particular office buildings, share salient design characteristics in order to be functional, well-priced and efficient. Such environments are structured into rooms that can be entered through doorways and are connected by hallways. Examples include but are not necessarily limited to hospitals, office and apartment buildings as well as homes for elderly or handicapped people. Usually such environments contain tables, shelves, closets, door frames, posters etc., which can explicitly be represented in RG Maps.

RG Mapping implicates additional contributions to the field of map building for indoor environments. First, it automatically recovers the structure of large classes of indoor environments and represents them explicitly. As a result, the complexity of data interpretation is significantly reduced, and high-level planning or human operators can explicitly refer to places in the environment. Novel algorithms have been developed for the detection and recognition of gateways, i.e. transitions between regions. Second, RG Mapping detects 2D/3D objects using laser range as well as image data, see Figure 1.3. The focus lies on a more general segmentation instead of model-based recognition. For now, the object recognition is restricted to arbitrary rectangular objects in 2D/3D. It is important to note that the system has no further apriori knowledge about pose, size or appearance of those objects. The assignment of objects to regions is done automatically, along with the structuring process. Third, semantic descriptions are obtained from human-machine interaction in the context of task assignment. That means, if a human operator specifies a region or object that has no name assigned to it, the system initiates a dialog to extend the semantic description of the map, i.e. requesting additional information from the user.

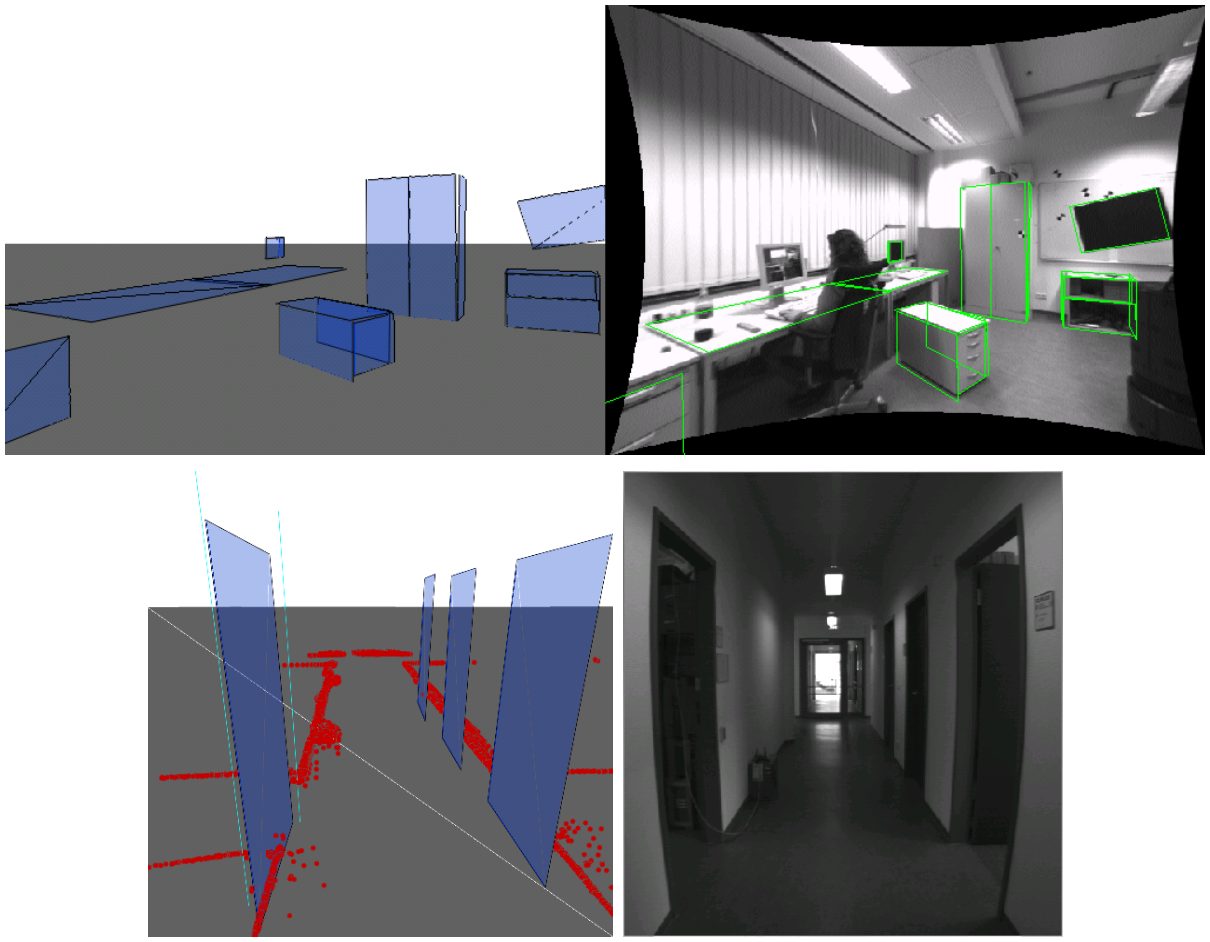


Figure 1.3: Reconstructed 3D objects in an office environment

Finally, the presented mapping and navigation system has two other important aspects, which are intended to support reliable exploration of, and navigation within RG Maps. It is important to note that these contributions are not only applicable to a system based on RG Maps, but their properties are particularly well suited for directed and intelligent exploration in the context of structured maps. First, contrary to common approaches for collision avoidance, the low-level control of the robot movement and the reasoning about collision free paths are strictly separated. The low-level control allows for very precise and fast pursuing of short trajectory segments, which can be changed at any time. The collision avoidance generates trajectory segments based on the interpretation of the current sensor data and short-distance targets from the path planning process. Additionally, the execution of low-level control is continuously monitored and, if necessary, adapted with respect to dynamic changes in the environment. The second aspect is the fact that the presented work has been fully implemented by means of a distributed (module-based) system, and runs in real-time on a real robot. Due to its architecture, the system can be easily ported to different platforms, provided that similar sensor data is available. Also, it facilitates simple incorporation of new algorithms for object or gateway recognition, localization, path planning etc.

As mentioned before, gateway detection/recognition is a crucial precondition in structured mapping, because it defines subsets of sensor data by means of regions. Thus, the main problem in RG Mapping is to reliably recognize gateways, despite changing illumination, partial occlusion and dynamic objects. This raises the question of how to define classes of gateways, and which features to use to describe those classes. But it also asks for fast and robust recognition algorithms. Such algorithms should be as generic as possible, because gateways may appear different in different environments. Also, the framework for the gateway detection has to be extendable to support the incorporation of new or improved recognition approaches. Finally, such algorithms have to deal with noisy data, false positives and apriori knowledge in case a place has been visited before.

On the other hand, a compact and appropriate description of regions based on the sensor data gathered within that region is necessary to guarantee granularity of the given information, unique identification of regions and scalability of the approach. RG Mapping integrates data from a laser range finder and cameras, which poses the problem of data fusion. Appropriate algorithms have to balance accuracy versus compactness along with reproducibility and robustness. The representation of a region must allow for robust localization and path planning, but at the same time should be compact in the sense of the amount of data that has to be stored. In addition, as for gateway detection, the generation of region descriptions includes the problem of noisy data, and furthermore data from dynamic objects. The latter addresses the problem of updating the region representation upon repeated observation.

The next chapter introduces and defines *Region & Gateway Maps* and discusses the steps necessary to build such maps. It describes the concepts of regions and gateways, and how they form RG Maps. In this context, we also explain the notion of the *Region & Gateway Graph*, how it is generated from RG Maps and how it is used for path planning. The last section in Chapter 2 gives an overview of different mapping methods and discusses the relation of *Region & Gateway Maps/Mapping* to those approaches.

Chapter 3 focuses on the generation of object models based on laser and vision data. Thereby, we restrict ourselves to rectangular 2D/3D objects, which present a prominent subclass of objects in indoor environments. It addresses the problem of 2D line segment grouping and presents an approach that is well suited for 2D laser data as well as for 2D line segments generated from image data. The latter is a precondition for our fast and accurate 3D reconstruction algorithm. The results from both laser and vision are utilized to align objects/observations in 3D with the respective 2D observations from a laser scanner. Finally, we present experimental results for the 2D segmentation and 3D reconstruction based on real-world scenarios.

Algorithms for the detection of gateways are described in Chapter 4. First, we present a novel approach for laser based recognition and classification of crossings and turns. It introduces a feature language well suited to describe the properties of such gateways, and a probabilistic recognition based on Hidden Markov Models. The second contribution in this chapter is the recognition of doors based on laser and vision. Thereby, we combine the results of Chapter 3, namely the 3D reconstruction of arbitrary rectangular 3D objects, with a simple detection algorithm that is based on laser range scans. We will present results for both approaches, and finally discuss and summarize our work on gateway detection.

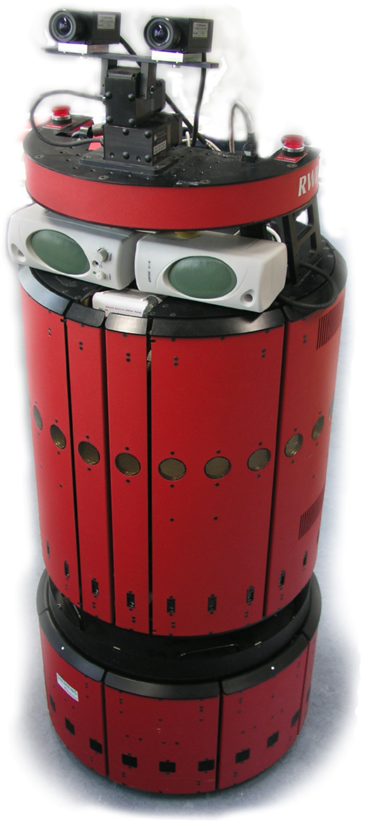


Figure 1.4: B21r robot

The before mentioned algorithms for object recognition and gateway detection along with the principles of RG Maps have been implemented in a module based framework - the *TumBot Mapping & Navigation System* on a real robot (B21r)[2], depicted in Figure 1.4. Chapter 5 gives an overview of the overall system and describes the functionality of the different modules, including path planning, localization, collision avoidance and robot motion control. In particular, path planning and localization have been developed with special focus on RG Maps. It will be shown that both of these problems can be handled efficiently and that they explicitly gain in transparency and usability due to the properties and advantages of RG Maps. In the last section of this chapter we present a novel approach to collision avoidance. It utilizes trajectory segments, laser range data and stereo vision to plan collision-free paths. In this context we will also discuss the related work.

[2]Product from iRobot Corp., formerly known as Real World Interfaces

In Chapter 6 we present experiments conducted with the overall system both based on external data sets and using a B21r robot. We will show that RG Mapping is both well suited and indeed capable of acquiring structured and object-oriented representations of large indoor environments. An important part of our evaluation is the discussion of structural consistency in RG Maps and the limitations of the presented mapping system.

The last chapter summarizes this work its contributions and results, along with a discussion of future work.

Chapter 2

The Principles of Region & Gateway Maps

In order to generate an appropriate path to a given target position a mobile robot needs to localize itself within its environment. Since the goal is to reach that target safely and in a reasonable amount of time, the robot is expected to reason about the feasibility of different solutions. When a plan has been chosen and a certain path is followed, the robot must recognize whether that path is blocked, e.g. because a door is closed or a chair is in the way, and should then react deliberately, based on the given environment representation. Therefore, an autonomous system needs information to identify objects that are used for localization along the path, e.g. door frames, hallway turns and crossings. This includes objects that have a descriptive meaning with respect to the topology of the environment and eventually the feasibility of a given path, but also objects that contain useful information like room numbers on door plates or signposts on a wall. In addition, the robot needs to have knowledge about movable objects like office chairs, small boxes etc. that can eventually be displaced if they block a given path. This leads to the following questions:

- What additional information, besides 2D metric descriptions, would enhance the ability of mobile robots to successfully navigate through an environment and fulfill complex tasks?
- How can such maps be generated automatically by a mobile robot?
  What are the key problems that have to be solved?

We propose *Region & Gateway Maps (RG Maps)* along with *Region & Gateway Mapping (RG Mapping)* as novel answers to these questions. *RG Maps* explicitly refer to regions, gateways and 2D/3D objects to give a structured object-oriented representation. *RG Mapping* aims at the autonomous acquisition of structured models of indoor environments, in particular office-, hospital- and museum-like environments.

In this chapter, we will first describe the general principles of *RG Maps* in Section 2.1, along with the definition of regions, gateways and the *RG Graph.* Section 2.2 formalizes the process of *RG Mapping.* The last section in this chapter gives a comprehensive overview of existing mapping approaches with a focus on indoor environments. We will also discuss how *RG Maps* and *RG Mapping* relate to existing approaches.

2.1 Region & Gateway Maps

Region & Gateway Maps (RG Maps) are described using a set of *Regions* and a set of *Gateways.* In general, the term *Region* refers to a contiguous area which is self-contained in the sense that it can only be entered or left through certain transitions (gateways). *Gateways* can be understood as the structuring elements in this mapping approach. They represent identifiable transitions between regions. "identifiable" means that they can be detected in the first place and recognized later on. More formally, an *RG Map* is represented by a tuple $\langle \mathbf{R}, \mathbf{G} \rangle$, where R denotes a set of *Regions* and G is a set of *Gateways* that represent possible transitions between regions. Each region contains references to the adjacent gateways, and each gateway "knows" to which regions it is linked to. Thus, *RG Maps* are in general of topological nature, but they differ from classical topological maps in that they provide metrical information for all places that have been visited/explored, not only for certain places (nodes), see also Section 2.3. Moreover, within a region, task-relevant objects are explicitly represented. That means, a *Region* is a tuple $\langle \mathbf{M}, \mathbf{O}, \mathbf{P}, \mathbf{G_{Ref}} \rangle$, where M denotes some metrical description, O is a set of object hypotheses, P refers to a set of characteristic properties, and G_{Ref} contains references to the adjacent gateways. Accordingly, *Gateways* are tuples $\langle \mathbf{M}, \mathbf{P}, \mathbf{R_{Ref}} \rangle$, with M and P as for regions, and R_{Ref} holding the references to the adjacent regions. The single most important property of a *Gateway* is its class label, defining how many regions it connects and in which fashion.

2.1.1 Region

In this section we will discuss the term *Region* with respect to indoor environments. It has already been mentioned that regions represent contiguous, bounded and self-contained areas. That means that all the data or observations collected within a region are only accessible/"visible" within that region. In other words, all information gathered within a region is stored in an encapsulated region object with a unique identification number and eventually a name tag. Hence, the data acquired in a large scale indoor environment is automatically clustered into separated units, i.e. regions. The data is then utilized to generate a detailed region description, which allows for accurate region-based localization and path planning. Region-based sensor data interpretation can achieve more accurate metric descriptions because it prevents the merging of observations that cannot result from the same object, such as distance measurements of opposite sides of a wall. Furthermore, by providing object hypotheses, regions support the semantic annotation of task-relevant objects, e.g. by human-machine interaction, and thus semantic high-level planning. Characteristic properties of regions are utilized to globally distinguish, i.e. recognize, regions. The different aspects for the use of regions in the context of *RG Mapping* are described in detail in Chapter 5.

We will now focus on the description of regions in the context of indoor environments, where a region can be an office, a hallway, an entrance hall or a room in an apartment. It can be entered or left through different kinds of doors, wide-narrow transitions, hallway crossings or turns. In the practical system, we use laser range and vision sensors. Therefore, regions are metrically described using a set of 2D line segments and 2D/3D object hypotheses. Each region uses its own coordinate system. We decided to employ a 2D line segment map because it is more compact than occupancy grids [85, 34] or a list of laser scans [44, 76], but still holds enough information for localization and path planning. As mentioned above, regions contain sets of object hypotheses, where we restrict ourselves to object classes in 3D that are rectangular, such as pictures or doors, and objects that can be represented by bounding cuboids, such as desks. In addition, rectangular 2D object hypotheses are generated from the 2D line segments. The respective algorithms are outlined in detail in the next chapter. Other characteristic properties of regions are:

- 2D bounding rectangle (bounding box), based on the metrical description
- main axes/directions of the 2D line segments
- 2D freespace, explicitly represents the obstacle free area within a region
- measure of accuracy and completeness for the metric description

From the 2D freespace feature further information like size or center of gravity with respect to the bounding box is calculated and used for global localization, i.e. region recognition. It is also utilized in region based localization, see Section 5.3. The measure of accuracy and completeness trigger the exploration process. That means, they describe how well the metric description fits to the acquired sensor data and whether or not there are still unexplored areas in that region.

2.1.2 Gateway

Gateways are the most important component of *Region & Gateway Maps*. They represent transitions between regions, and hence, they are the structuring element in the process of building such maps. A number of researchers including Kortenkamp [62], Youngblood [122] and Chown [23, 22, 24] have proposed gateways as first class objects in map representations:

“In buildings these are typically doorways... Therefore a gateway occurs where there is at least a partial visual separation between two neighboring areas and the gateway itself is a visual opening to a previously obscured area. At such a place one has the option of entering the new area or staying in the previous area.”, Chown et. al. [22, page 32].

“... they provide a natural way to divide an environment up into smaller, more manageable, pieces. Gateways can be thought of as occurring at the transition between regions. Eventually gateways can be used to organize these regions in a more abstract, hierarchical structure...”, Chown [23, page 10].

The main problem is to detect gateways based on sensor data, i.e. to decide whether some gateway is present or not. But it is also very important that gateways can be recognized, when they are observed again, and that there exists a method to determine whether or not a gateway has been traversed. To summarize, gateways form perceptually recognizable places in the environment that often open new views that could not be observed before traversal.

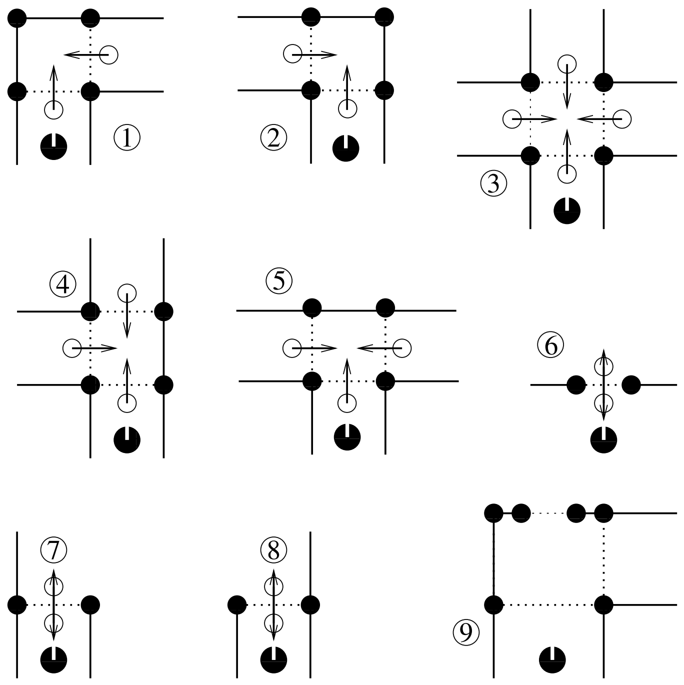


Figure 2.1: Examples for gateways: *Right/Left-Turn* (1,2), *X-Crossing* (3), *LeftT- and T-Junction* (4,5), *Narrow Passage* (6), Right/Left Opening (7,8), Combination of Gateways (9); (• gateway point, ◦ crossing point, ← traversal direction, ... region border)

In indoor environments, several types of gateways can be distinguished, e.g. *X-Crossings, LT/T/RT-Junctions* and *L/R-Turns* in hallways, *Narrow Passages* like doors and changes from a rather narrow hallway into an open room, e.g. a lobby. The different types of gateways are depicted in Figure 2.1. Gateways are specified by a class label, some metric description, adjacent regions and characteristic properties, such as gateway points, crossing points and traversal directions. The latter two are used to detect when a gateway is entered and left. Gateway points are generated from the observations taken by laser range and vision sensors. They support recognition and tracking of gateways. In Chapter 4 we will present approaches for the detection of different gateway classes in structured indoor environments, and Chapter 5 shows their application in our mapping and navigation system.

2.1.3 Region & Gateway Graph

The purpose of the *Region & Gateway Graph (RG Graph)* is to provide a compact graph representation for an RG Map that allows for fast, adaptive and intelligent path planning on the level of regions and gateways. Strictly speaking, the RG Map itself already presents a graph, where nodes refer to regions and edges to gateways or the other way around. For two reasons these interpretations are not very useful for graph based path planning. First, there could be more than one direct connection between two nodes, which would complicate the search. For example, two regions could be connected by more than one gateway, e.g. a large room and a hallway that share two doors. Second, if a gateway or a region refers to a node, it would be difficult to assign costs to it, and consider them in the planning algorithm. But we certainly want to incorporate costs for gateway or region traversal into path planning. Based on the definition of gateways, in particular the property we call crossing points, we can define a very suitable graph representation, i.e. the RG Graph. Each node refers to a crossing point, and is annotated with the identification number (ID) of the gateway and region that the crossing point belongs to. As a consequence, each edge in the graph describes the costs between two crossing points, which refers to either a gateway or a region traversal. Furthermore, it can be easily determined whether an edge presents a route through a region, i.e. both nodes have the same region id, or through a gateway, where both nodes have the same gateway id. Figure 2.2 shows an example of an RG Graph.

Since crossing points are generated during gateway detection and stored in the RG Map, the proposed graph representation can be easily obtained. To build the graph, an arbitrarily chosen crossing point presents the first node in the graph. Adjacent crossing points are then iteratively added. If a new region is added, we only need to add the respective node to the graph representation. As a result, each node in the RG Graph has only one or no connection to any other node in the graph. And each edge describes a meaningful route either through a region or a gateway. In order to apply the A* search algorithm to the RG Graph, we need to define a cost heuristic. Because the coordinate transformations between regions are stored in the gateway objects, it is easy to transform all crossing points into a global frame of reference while adding them to the graph. Thus, our initial heuristic uses the Euclidean distance between crossing points. When moving through a region, these costs can be adapted, e.g. by estimating the real distances and accounting for the time needed. For gateways, one could observe how often it was traversable, e.g. the door was open, and use this information to adapt the costs and thus bias the path

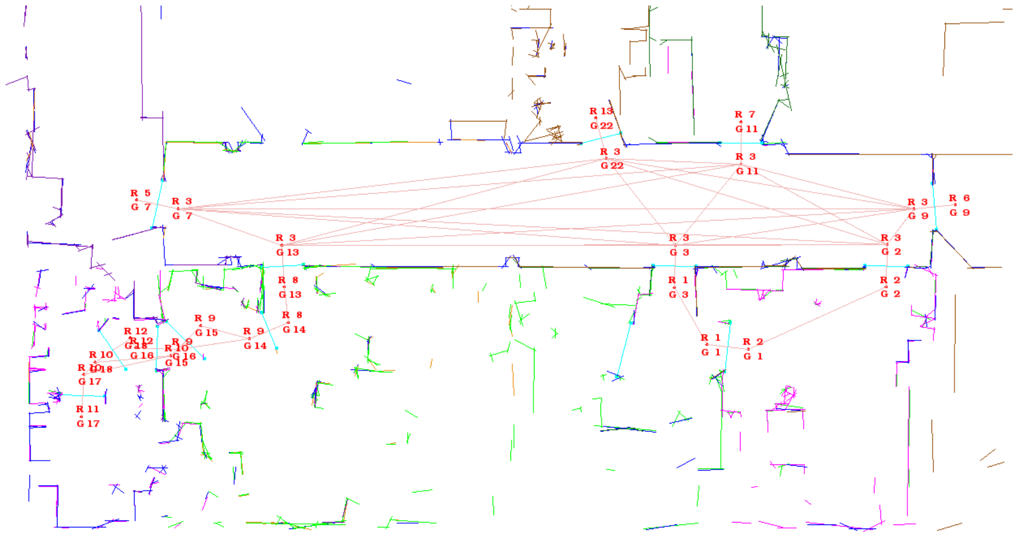


Figure 2.2: Region & Gateway Graph: red lines depict edges, end points refer to crossing points annotated with region and gateway ID.

planning. We will come back to that point in the context of the path planning module presented in Chapter 5.4.

2.2 The Process of Region & Gateway Mapping

Having introduced *Regions*, *Gateways* and the general concepts of *RG Maps*, this section describes how the mapping process works and how it is integrated within a navigation system. A very general formalization of the *Region & Gateway Mapping* task is: Given methods to robustly/reliably detect and recognize gateways, algorithms for data interpretation by means of metrical mapping and object recognition, and methods to robustly/reliably recognize regions, compute the environment representation that best explains the observations. For the mapping approach we are going to present, this formalization can be subdivided into three tasks as illustrated in Figure 2.3. This is the formal basis of the implemented mapping and navigation system, we will present in Chapter 5 and 6.

Region & Gateway Mapping

1. Generate a topology based on the observation of gateways
2. Calculate a description for each region utilizing all observations/measurements collected within that region
3. Calculate a description for each gateway utilizing all observations/measurements collected within and around that gateway

Figure 2.3: Simple overview of the process of RG Mapping

The general description of *RG mapping* poses several questions. How can we detect gateways based on laser range data and 3D features from stereo vision? How must the path planning and localization be adapted to work for those structured maps? The reconstruction of objects from laser and image data is described in the next chapter. Chapter 4 presents solutions to the gateway detection problem, and Chapter 5 discusses the overall navigation system, including localization and path planning. For now we assume that modules, which handle those problems are available.

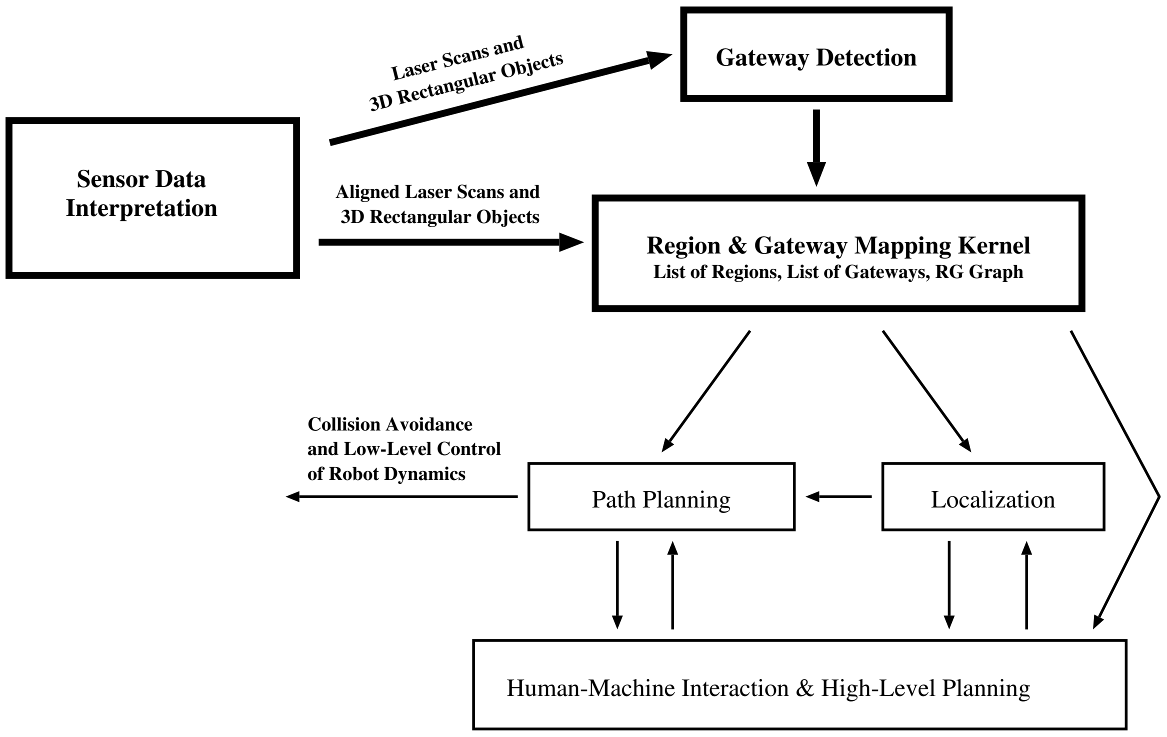


Figure 2.4: Overview of the Region & Gateway Mapping system

That leaves the question of how the structuring process works, and how the modules interact? Figure 2.4 gives a schematic overview of the *Region & Gateway Mapping* system. The central mapping process runs in the *Region & Gateway Mapping kernel (RGM Kernel).* The task can be formalized as follows. Given a set of aligned laser scans, object and gateway hypotheses, consistently structure the environment representation and calculate compact descriptions for each region and gateway. This problem can be solved online as long as gateways can robustly be detected based on the present data, and data interpretation (laser, vision) can be carried out in real-time. The mapping process can be divided into the following steps. After entering a new region the *RGM Kernel* stacks all the incoming data, gradually adding new scans to the current 2D line segment map and properly aligning object hypotheses to that map. At this point the information of the current region is already available to the *Path Planning* and *Localization Modules*, although it may be incomplete. When the region is left, the *Gateway Detection* triggers the *RGM Kernel* that in turn evaluates which scans and objects belong to the previous region. The respective data is then coherently interpreted so as to give the best description for that region. If the region has not been visited before, it is added to the *RG Map*. Otherwise the former description is updated based on the newly acquired data.

RG Maps and *RG Mapping* make several important technical contributions to existing mapping techniques, in particular to Gutmann's scan matching [44]. First, *RG Mapping* infers compact yet accurate 2D line segment-based geometric descriptions. Because these 2D line maps describe areas as a small number of line segments they yield smaller search spaces for generating object hypotheses and localization. Second, *RG Maps* capture topological structures that are typically found in indoor environments such as office buildings and apartments. Third, *RG Maps* store mission relevant objects in the environment. As a result, the complexity of path planning and localization is significantly reduced, which will be discussed in detail in Chapter 5. Furthermore, *RG Maps* offer an intuitive representation of indoor environments, which eases semantic annotation of regions and objects by human-machine interaction, as well as high-level planning.

Table 2.1 on page 17 summarizes all features of regions and gateways. Whereas "in general" refers to the more general description of features, "in particular" points out what has been implemented already or is currently worked on in the context of our mapping and navigation system.

2.3 Related Work on Map Building

The research of map building for mobile robots is commonly divided into approaches for indoor and outdoor environments [28, 107]. In the following we will focus on indoor environments, because it is the scenario we have chosen to show the applicability of our approach. Algorithms in this class of mapping problems differ in the degree they capture structure, objects and accurate metrical description. In the literature they are referred to as metrical, topological and hybrid approaches.

Metrical and topological mapping are somewhat contrary in their weaknesses and strengths. The former provides rich metric description of the environment, but lacks structure. Such representations often involve huge amounts of data to be maintained and considered, when extending or utilizing the map, in particular for large-scale environments. Because the data describes a global map, it is rather difficult to generate more compact descriptions that can still be easily updated or extended in the event of new observations. On the other hand, topological maps contain only distinctive places and a description of how the robot can move from one to the next. The necessary amount of data depends on the choice of such places, but is in general much smaller compared to metrical maps. The problem is to resolve ambiguities between distinctive places, which is more often rather difficult, due to the fact that there is no metric information available between those places. That means the robot can only localize itself at distinctive places. Hybrid approaches intend to combine the properties of both. They do so by extracting the topology of the generated metric map

Region		**Gateway**	
in general	in particular	in general	in particular
Metric 2D/3D data for sensor based localization	2D line segment map	Characteristic features for detection, recognition and traversal observation	gateway points, crossing points plus traversal directions
2D/3D Object hypotheses	2D/3D rectangles, 2D circle	Class Label	X-Crossing, *L/R-Turn*, *LT/T/RT-Junction*, *Narrow Passage*, (*Wide/Narrow-Transition)*
Feature Vector for recognition and classification	bounding rectangle, 2D freespace and its 1st/2nd order moments, main axes	Adjacent regions	list of references to neighboring regions
Characteristic measures	accuracy, completeness	Metric 2D/3D data	not yet supported
Connected Gateways	list of references to adjacent gateways		
List of Sub-Regions	not yet supported		

Table 2.1: Summary of the Features for *Regions* and *Gateways*

by means of Voronoi graphs [109, 112]. Such a topological description allows for fast path planning, but is not necessarily related to the structure, in the way a human would perceive it. Table 2.2 gives a brief comparison of the three basic classes of mapping algorithms, with respect to their characteristics described above. It is important to note that RG Maps do not represent a class on its own, but technically belong to hybrid approaches. We mention it explicitly because it differs from classical hybrid mapping in the way it handles objects and environment structure. Although, topological and hybrid approaches capture some structure of the environment, it strongly depends on the method for finding distinctive places in order to be useful for human-machine interaction or high-level planning. We will come back to that point in Subsection 2.3.2.

To each of the basic mapping classes for indoor environments belongs a huge number of different approaches. For the sake of completeness we will briefly summarize the research on metrical mapping in the next subsection. In Subsection 2.3.2 we will discuss approaches that deal with environment structure, i.e. topological and hybrid mapping. In particular, it will be discussed that RG Mapping is closely related to the *Spatial Semantic Hierarchy (SSH)* by Kuipers and Byun [66, 69] and Yeap's theory of *Absolute Spatial Representations (ASR)* [121], but differs in some aspects.

Properties/ Approach	**represents structure**	**represents objects**	**accurate metrical description**	**has applicable implementations for large scale environments**
Metrical Maps	no	no	yes, globally	yes
Topological Maps	yes	no	only for distinctive places	no
Classical Hybrid Maps	yes	no	yes	yes
RG Maps (hybrid approach)	yes	yes	yes	yes

Table 2.2: Summary of Mapping Approaches

2.3.1 Metric Maps

Most of the successful mapping approaches, e.g. [108, 14, 13, 110, 83, 46, 32, 44, 43], measure the environment outline metrically by means of some sensor, e.g. sonar, laser range or cameras. A global metrical map is built in an exploration phase, and afterwards the robot navigates through the environment. It is assumed that the environment is more or less static and that therefore measurements are similar when visiting known places. As a consequence, such places can be recognized, i.e. the platform can autonomously and continuously localize itself with respect to the given map. Few mapping approaches explicitly deal with dynamic environments, either by estimating and suppressing implausible, unreliable measurements using the expectation-maximization (EM) algorithm [48] or by modeling objects in 2D [7, 1]. Since they need to iterate over all obtained measurements to generate a global map, they can not incrementally build and update maps while the robot explores its environment. Also, it is not addressed how the information about objects is integrated into a consistent and useful map representation. Other approaches estimate the position and displacement of dynamic objects, i.e. perform object/people tracking [47], and use this information in the mapping process.

There exist a great number of approaches for building detailed 3D environment models using cameras or laser [84, 2, 123, 55, 74, 45, 99]. In [28] Kak gives a recent and comprehensive survey of vision-based navigation and map building for mobile robots. It shows that, based on appropriate 3D models, mobile robots can perform localization solely based on vision [63]. Position tracking without prior knowledge using cameras [65] will soon be an adequate alternative to laser based approaches. Several researchers build terrain maps from stereo vision [51, 53, 72, 57], which we believe will play a key role in mapping of unstructured outdoor environments. Nevertheless, to the best of our knowledge, almost all practical mapping and navigation systems to date that autonomously map large and cyclic indoor environments and robustly navigate through them, rely on two-dimensional representations based on laser or sonar sensors.

Methods for metrical mapping can be roughly characterized by the representation they use [20, 34, 73, 76], and either they work incrementally [42, 44, 83, 46] or globally [76]. According to Thrun, virtually all algorithms for metrical mapping are based on probabilistic methods. He gives an extensive and comprehensive survey in [107], along with a discussion of their differences, weaknesses and strengths. Of particular interest are the algorithms that map and localize concurrently, because they enable mapping and navigation systems that can perform robust life-long learning. They are referred to as *simultaneous localization and mapping (SLAM)* or *concurrent mapping and localization (CML)*, and to date a number of solutions exist [83, 46, 32, 44, 29, 26, 30, 19, 41, 111].

Region & Gateway Mapping relies in its core on a fast, robust and accurate SLAM algorithm. It is important to note that literally any incremental SLAM algorithm could be utilized within the RG Mapping framework. Since occupancy grid maps [85, 34] present a discretization of the environment, they are not well suited for generating compact yet accurate geometric descriptions. Nevertheless, they could be used as such without further processing or compression. Instead, we decided to use an approach that provides the raw sensor readings, after incremental pose estimation. We found Gutmann's solution to SLAM [42, 44] to be fast, considerably robust and accurate, and easy to integrate. A thorough comparison of SLAM approaches and a discussion of its foundations is beyond the scope of this work, because RG Mapping works on top of incremental SLAM, rather than presenting a contribution to SLAM on its own. RG Mapping aims at real-time interpretation of the acquired sensor data in order to generate a structured representation with explicit reference to 2D/3D objects. In this sense, RG Mapping can be seen as a layer on top of classical metrical SLAM approaches, with the important difference that the acquired data is automatically clustered. There is no global frame of reference. Thus, the generation of compact metric representations is greatly simplified, because only small parts of the data, namely for regions, must be considered. Consequently, inaccuracies or errors in the geometric description of a region have only impact on the localization performance within that region. Such errors are mainly caused by odometric measurements. Furthermore, once such deficiencies are detected, the robot simply gathers new data within the respective region, i.e. explores, and builds a new region representation.

2.3.2 Topological and Hybrid Maps

Another class of mapping approaches captures the topology of the environment in terms of distinctive places and connections between those places by means of motor commands, paths or directions. The idea is based on cognitive maps [68], which are inspired by the biological and psychological analogue, i.e. by animals and humans [77, 90, 100]. Classical approaches build a topology of places, where each place contains information that supports its identification upon later traversal. That means, the robot can localize with respect to those places. Such a map can be compactly represented by a graph, where the nodes refer to places in the environment, and the edges describe how the platform can get from one node to the next. The representation used for place recognition mainly depends on the sensor data. Often it is based on sonar or laser, thus two-dimensional metric descriptions are applied, as discussed in Subsection 2.3.1. Vision based systems mostly use feature representations, i.e. vertical lines [61, 16] or points [64].

One of the major problems is to decide when to add a new place to the topology, or in other words, how to characterize or classify such places. Chown and Kortenkamp [22] suggested gateways to present such places in structured environments, that is, places that naturally subdivide the environment, see also Subsection 2.1.2. In '93 Kortenkamp presented an implementation of PLAN [62] that he called RPLAN, which used sonar to detect gateways and visual cues to describe the gateways [61]. Impressive as it was fifteen years ago, the system could not deal with strongly cluttered or dynamic environments. This is because it uses a complex state automaton to detect openings and closings in the sonar readings, and that automaton is based on the assumption that the robot follows a wall. Nevertheless, it showed that topological maps based on the notion of gateways can be autonomously built by a mobile robot and used for localization.

Kuipers has been one of the first to suggest topological maps for environment representation in the context of mobile robots [68]. Later he and Byun [66, 69] formalized the topological mapping problem with their theory of the *Spatial Semantic Hierarchy (SSH)* inspired by the properties of human cognitive maps [77]. The SSH forms a hierarchy of representations, which consist of several levels, both quantitative and qualitative. The sensor level abstracts over the sensory system of the mobile robot and provides quantitative measurements, e.g. from camera, laser range finder or sonar sensors. Control laws are generated in the control level. They describe how the robot should move along a given path between two or more nodes. The causal level abstracts from the control and the sensor level by building a discrete model, which is described in terms of sensory views, actions, and the causal relations among them. For example, wall-following or more general trajectory-following define discrete actions, which are related to certain sensory views, whereas the continuous control is hidden in the control level. Places, paths, regions and their topological relations are maintained at the topological level. It presents a discrete model of the environment that describes which place is linked to which other place, and the path between those places. Finally, the metric level supports the generation of a global metric map within a single frame of reference. Kuipers states that this level "may be useful but is seldom essential" [69, page 198].

For RG Maps, from our point of view, the sensor abstraction is directly linked into the region processing, which implicates detailed and accurate metric descriptions for regions. Under these circumstances, control laws for reaching a certain point within a region are generated on the fly and do not need to be stored, see also Section 5.5. Therefore, RG Maps do not contain a control level. By the same rationale it does not seem to be necessary to store causal descriptions for the robot's movements. Instead, the robot chooses the appropriate trajectories or control laws based on the current observations and the planned path. Thereby, it operates within the frame of a region and considers only the respective data. We agree that the global metric map is not needed for the robots performance in that case, but possibly for the human machine interaction. As a consequence, global metric maps are not explicitly generated and maintained within RG Maps, but they can be extracted easily on demand.

RG Maps contain a very strong topological level, which supports features similar to those suggested in the SSH. RG Maps represent the environment in terms of connected areas and distinctive places between those areas, i.e. the gateways. But instead of storing paths or actions that describe how to traverse the distance between two gateways, a compact metric description is utilized, to determine the necessary actions while the robot moves. Each of the entities of an RG Map contains topological attributes by means of connectivity and reachability. This information can be directly retrieved from the references to adjacent regions or gateways. Visibility of metric data is implicitly modeled, because regions and gateways are encapsulated objects, in that they comprise only data, acquired in the context of those objects. Furthermore, the RG Graph that can be easily extracted from an RG Map, describes environment properties in terms of traversability, which allows for fast, adaptive and reactive path planning.

The notion of regions as an important component of topological and cognitive maps has also been addressed by Yeap [121] in terms of *Absolute Spatial Representations (ASR).* He argues that it is important "to compute and recognize local environments" and "that identification of exits is the cornerstone to computing a description of the local space" [121, page 265 and 273]. Local environments are presented by ASRs, which are two-dimensional representations based on line segments. They are built and extended gradually while the robot moves around. Their interconnections follow from the order they have been observed. Exits are defined by the shortest edge that covers an occluded area. In this context "occluded" does not necessarily mean that the space behind that edge is not directly observable, but the robot has to cross it in order to get there. Because this definition seems to be focused on two-dimensional observations, it is unlikely to result in a meaningful representation in highly cluttered regions. It is not clear, if Yeap's *Computational Theory of Cognitive Maps* can be easily extended to account for three-dimensional observations. Reviewing RG Maps with respect to Yeap's nomenclature suggests that gateways correspond to exits, and regions to ASRs. Thus, the main ideas seem to be very similar, but RG Maps also differ in at least two aspects. First, the separation of the environment is based on distinctive places that are supposed to be meaningful with respect to the environment. Namely, RG Maps implement the gateway paradigm as suggested by Chown [22, 24], which is more closely related to distinctive places in the sense of the SSH than to the exits in Yeap's theory. Second, based on this structuring process all data acquired within a region is utilized to generate compact region descriptions, including three-dimensional features and/or objects. While the role of regions with respect to RG Maps is comparable to the notion of ASRs, they differ in the way they are built and interpreted.

Recently, Beeson et al. [4, 67, 3] have presented algorithms and experiments for the determination of distinctive places in the framework of the *Spatial Semantic Hierarchy (SSH)*. They define distinctiveness measures based on local metric maps by means of the extended Voronoi graph. The approach seems very promising, and it would be interesting to investigate how his method could be combined with the search for meaningful distinctive places in RG Mapping, i.e. gateways as defined in Subsection 2.1.2, and how it performs. For now, further investigations are beyond the scope of this work.

We mentioned in the introduction of this section that some approaches extract the topology from metrical maps after a global metric map has been built [109]. Since these approaches present a mixture of metrical and topological maps, they are referred to as hybrid mapping. The main purpose of the topological description is to speed up path planning by means of a fast graph search algorithm. Therefore, the graph is often not related to the structure of the environment, as a human would perceive it, and can thus hardly be used for high-level planning or human-machine interaction. Also, these approaches do not exploit an important advantage of topological mapping, that is, the decreased influence of odometric errors on the map building process. We have outlined before that in RG Maps odometric errors have only impact within a region, i.e. locally, and can thus be compensated for more easily. In the worst case, the respective region must be explored again, but not the whole environment.

2.3.3 Discussion

Building environment representations for mobile autonomous systems is a very active and challenging research field. A huge amount of approaches exist which build impressively accurate two-dimensional global metric maps while the robot moves through the environment (SLAM/CLM problem). But for complex tasks involving object manipulation, human-machine interaction or representation/interpretation of dynamic environments they hardly seem to be sufficient. Different theories tackle the problem of representation from a cognitive point of view by means of topological mapping, and algorithms to account for some of the before mentioned scenarios, e.g. people tracking in dynamic environments. Only few of the topological approaches seem to be able to reliably map large scale environments and localize the robot over long periods of time and traveled distance. As a consequence, there have been several suggestions to close the gap between those two main classes of mapping approaches, and the work presented here falls into this category. We combine the gateway paradigm, the notion of local spatial representations, topological structure, objects and metric representation in a single framework that we call Region & Gateway Mapping.

Chapter 3

Acquiring Models of Task-Relevant 2D/3D Objects

In the last chapter, Region & Gateway Maps have been introduced as a novel representation for structured indoor environments. It has been pointed out that the region description does not only comprise metric information from the laser scans (2D line segment map), but also explicit hypotheses of task-relevant 2D/3D objects. In this context task-relevant means that the 2D objects give hints for possible 3D objects, so they support the map building process. But 2D/3D objects can also be used for feature based localization. Furthermore, those objects can be referred to by a human operator, e.g. a door or a desk.

For a better explanation let us consider the following example. The outline of a desk that can be observed from all sides by a laser range finder is found to be a rectangle in 2D. Then the mapping process segments this object and adds it to the region description. Later on a human operator gives the robot the task to pick up a book from Frank's desk, which is the object the robot just added to the map. But at this point the robot has no name for the 2D rectangle (or 3D cube), hence it asks and in turn annotates the respective object. Next time the robot knows which object in the map refers to Frank's desk. Thus, task-relevant objects denote useful information for exploration behaviour, gateway detection (doors), human-machine interaction and high level planning. Additionally, 2D objects give an initialization for 3D reconstruction.

The first step in either laser or vision-based reconstruction is segmentation by means of extracting lines from the sensor data. In fact, for RG Maps, the 2D data is already present in form of a 2D line segment map. For image-based reconstruction, we need to segment lines in the image, which is described briefly in Section 3.3.1. The next step is to group those lines to form either 2D rectangles (in case of the laser data) or 2D quadrangles (in case of image data). The 2D grouping problem can be solved in a similar fashion for both cases. In Section 3.1 we propose an algorithm which is considerably robust and fast, and gives a plausible set of rectangle/quadrangle hypotheses. As a result we obtain 2D rectangular objects from 2D line segment maps (section 3.2), and 2D quadrangles (hypotheses for 3D rectangles) from images. The latter are used for reconstruction of rectangular 3D objects, which is described in detail in Section 3.3. The chapter concludes with a discussion and summary.

3.1 2D Line Grouping to Quadrangular Objects

In this section we present an algorithm for grouping 2D line segments with respect to quadrangle hypotheses. The most straightforward approach would be to iterate over all combinations of four lines (out of all N given lines), hence considering $\binom{N}{4}$ configurations. It is clear that this implies a very high computational cost. The approach can be subdivided by first considering only two lines. Certain constraints for these lines must be defined and fulfilled. On the next level the third line is considered for valid two line configurations and so on. Thus, the search space is decreased in each step, which results in a faster generation of hypotheses. Defining these constraints for two, three and four line configurations is a difficult task, and chances are high that some valid quadrangle hypotheses would not be considered. Also, the rate of false positives can be quite high, when at the same time the rules are supposed to be rather general in order not to miss hypotheses.

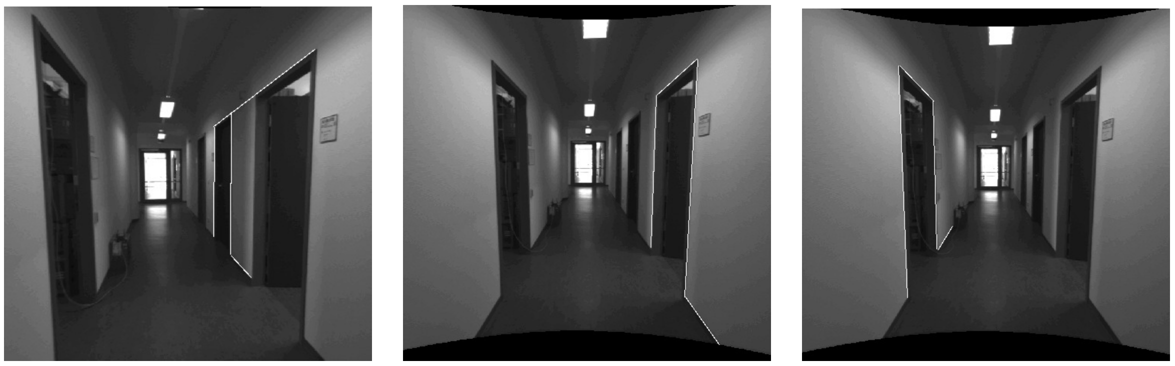


Figure 3.1: Examples for none-restrictive 2D line grouping of line segments.

For these reasons and to further speed up the grouping process, we suggest a two stage approach. In the first step all two line configurations are considered, i.e. $\binom{N}{2}$, whereas the constraints are fairly simple, i.e. angle and distance. In the second step, we consider all combinations of those "corners". We distinguish the restrictive (for rectangles in 2D) and the non-restrictive case (for quadrangles in images). That means for 2D rectangles we assume all pairs of line segments (here (1,2), (2,3), (3,4), (4,1)) to be considerably close, i.e. $distance < DN_{max}$. In images it often happens that parts of lines are missing, in particular the lower edge of a door when it is open. Nevertheless, this edge can be "found"

1. For all pairs of line segments l_i and l_j:
   - $\alpha_{min} < min\{\ angle(l_i, l_j)\ \} < \alpha_{max}$ **AND** $distance(l_i, l_j) < DN_{max}$
     $\Rightarrow$ set of valid pairs lp_{ij}
2. For all pairs of pairs of line segments lp_{ij}, lp_{kl}:
   Assuming the first pair lp_{ij} refers to the first and second line segment, and lp_{kl} refers to the third and fourth, there are only two possible valid configuration - $quad(l_i, l_j, l_k, l_l)$ and $quad(l_i, l_j, l_l, l_k)$
   - Calculate distances d_{nm} for $(l_i, l_k), (l_i, l_l), (l_j, l_k), (l_j, l_l)$
   - Compare those distances to DN_{max}
     $\Rightarrow$ $b_{ik}, b_{il}, b_{jk}, b_{jl}$, where b_{nm} can be either true or false,
     depending on whether the condition $(d_{nm} < DN_{max})$ is fulfilled or not
   - Invalid configurations:
     ($b_{ik} = true$ **AND** $b_{jk} = true$); ($b_{il} = true$ **AND** $b_{jl} = true$);
     ($b_{ik} = true$ **AND** $b_{il} = true$); ($b_{jk} = true$ **AND** $b_{jl} = true$);
     ($b_{ik} = false$ **AND** $b_{jk} = false$ **AND** $b_{il} = false$ **AND** $b_{jl} = false$)
   - Valid configurations:
     A) Non-restrictive case:
     $\rightarrow$ l_i is direct neighbour of l_k and/or l_j is direct neighbour of l_l
     ($b_{ik} = true$ **OR** $b_{jl} = true$)
     AND ($\alpha_{min} < min\{\ angle(l_i, l_k)\ \} < \alpha_{max}$)
     AND ($\alpha_{min} < min\{\ angle(l_j, l_l)\ \} < \alpha_{max}$)
     AND ($distance(l_i, l_l) > DO_{min}$)
     AND ($distance(l_j, l_k) > DO_{min}$)
     $\rightarrow$ l_i is direct neighbour of l_l and/or l_j is direct neighbour of l_k
     ($b_{il} = true$ **OR** $b_{jk} = true$)
     AND ($\alpha_{min} < min\{\ angle(l_i, l_l)\ \} < \alpha_{max}$)
     AND ($\alpha_{min} < min\{\ angle(l_j, l_k)\ \} < \alpha_{max}$)
     AND ($distance(l_i, l_k) > DO_{min}$)
     AND ($distance(l_j, l_l) > DO_{min}$)
     B) Restrictive case:
     Similar to the non-restrictive case, but:
     Replace **OR** by **AND** in the first condition,
     i.e. if l_i is direct neighbour of l_k than l_j must be direct neighbour
     of l_l in terms of the distance criterion

Figure 3.2: Grouping of 2D line segments to 2D quadrangles

by extending the wall-floor transition if it is detectable. That means for images, we want to consider configurations as depicted in Figure 3.1. Only four parameters must be specified for this algorithm, and they have a clear geometric motivation:

- $\alpha_{min},\ \alpha_{max}$ - minimum/maximum angle between two neighbouring line segments
- DN_{max} - maximum distance between two neighbouring line segments
- DO_{min} - minimum distance between opposite line segments

The function $min\{\ angle(l_i, l_j)\ \}$ returns the smaller of the two intersection angles between the lines defined by l_i and l_j. The smallest distance between two line segments ($distance(l_i, l_j)$) is zero if a real intersection exists. Otherwise it is the minimum of all distances of the end points and virtual intersections of the line segments with respect to each other or each others end points. The algorithm is summarized in Figure 3.2.

The next sections shows how the presented 2D line grouping algorithm can be used to generate 2D rectangular objects from 2D line segment maps. In Section 3.3.1 the same algorithm is applied to obtain 2D quadrangle hypotheses for the reconstruction of rectangular 3D objects.

3.1.1 Merging 2D Line Segments

We have seen that the presented 2D line grouping algorithm scales strongly with the overall number of line segments N, and of course with the number of valid quadrangles in the given array of line segments. By merging the line segments beforehand, we can decrease the number of line segments to be considered, and hence speed up the grouping process. We will briefly outline how this merging of line segments is performed. The goal is to combine one or more line segments into a larger line segment, if and only if:

- The line segments lie approximately on the same line (angle and distance).
- The line segments overlap or the smallest distance between their endpoints is below a certain threshold.

In the first step the 2D line segment array is sorted according to the angle to the x-axis, whereas the angle itself is only calculated once per line segment. Because of the sorting, we do not compare each line with all others, but instead operate in a window (down- and upwards in the array) around the current line segment given by a maximum deviation of the angle. This way the algorithm runs approximately in $O(N)$, where N is the number of line segments. If the angle between two line segments is smaller than a threshold α_{max}, we consider the distance D_L as follows. The endpoints of each line segment are orthogonally projected onto the other line segment, and the distance d_i of an endpoint P_i to its projection P_i' is computed. The distance D_L between two line segments is then given by the averaged sum over those four distances, see Figure 3.3.

$$D_L = \sum_{i=1}^{4} d_i$$

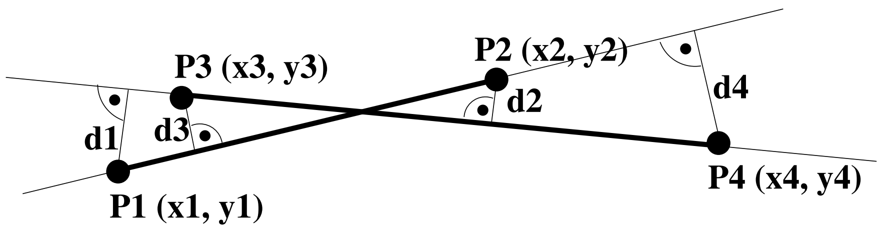


Figure 3.3: Merging of two line segments

The line segments are not considered further if $D_L > D_L^{max}$. Otherwise, the minimum distance D_{gap} between the endpoints along the common line is calculated. The lines are merged if they do either overlap or D_{gap} is below a threshold D_{gap}^{max}. Thereby the new line segment is defined by the endpoints of the given line segments, which have the largest distance. Altogether, three parameters have to be specified for the presented merging algorithm:

- α_{max} - maximum angle between two line segments
- D_L^{max} - maximum (overall) distance between two line segments
- D_{gap}^{max} - maximum gap between two line segments,
  that are considered to lie on the same line

The unit of the distance parameters is millimeter or pixel, depending on whether we consider laser range or image data. An example is depicted in Figure 3.4.

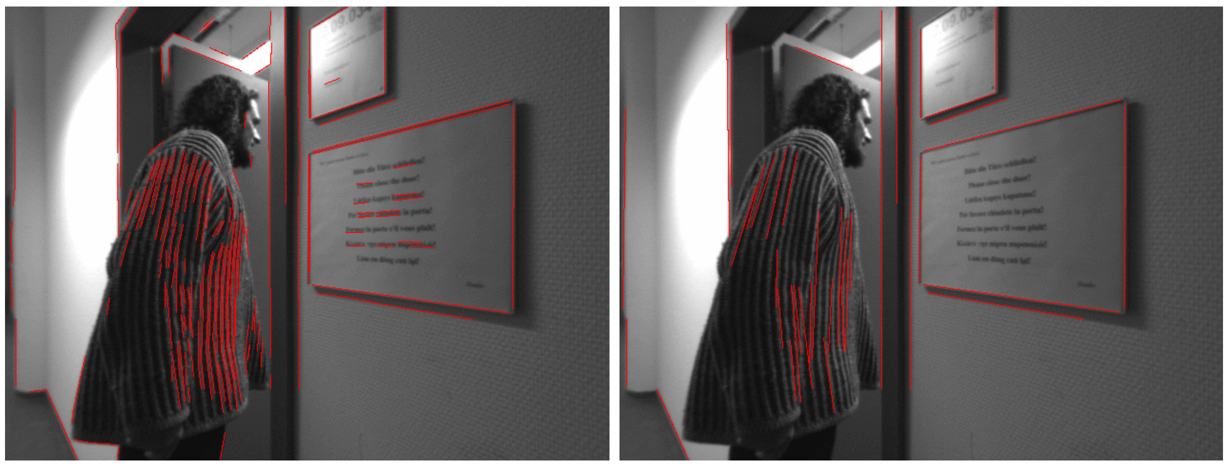


Figure 3.4: Example for line merging: **left:** line segments generated by edge detection, e.g. with Sobel or Canny edge detectors; **right:** Resulting line segments after merging

3.2 Generating Rectangular 2D Objects

From Chapter 2 we know that after a region has been fully explored one component of its description is the 2D line segment map. To generate 2D object hypotheses, the algorithm described in Section 3.1 is applied to this map. As a preprocessing step the given line segments are merged according to Subsection 3.1.1. Since laser distance measurements are very accurate, the parameters have been chosen very restrictive, i.e.:

$$\alpha_{max} = 3 \ degree; \quad D_L^{max} = 20 \ millimeter; \quad D_{gap}^{max} = 150 \ millimeter$$

For the 2D line grouping we applied the restrictive case and found the following parameters to give good results: $\alpha_{min} = 80 \ degree; \quad \alpha_{max} = 100 \ degree;$

$$DN_{max} = 200 \ millimeter; \quad DO_{min} = 200 \ millimeter$$

As described before, DO_{min} mainly defines the minimum size of the objects to be considered, here the opposite sides should be at least 200 millimeters apart. By increasing DN_{max}, e.g. to 800 millimeters, more hypotheses are generated. When deleting all hypotheses that are contained in others or strongly overlap the resulting segmentation denotes larger regions of interest, see Figure 3.5 (left).

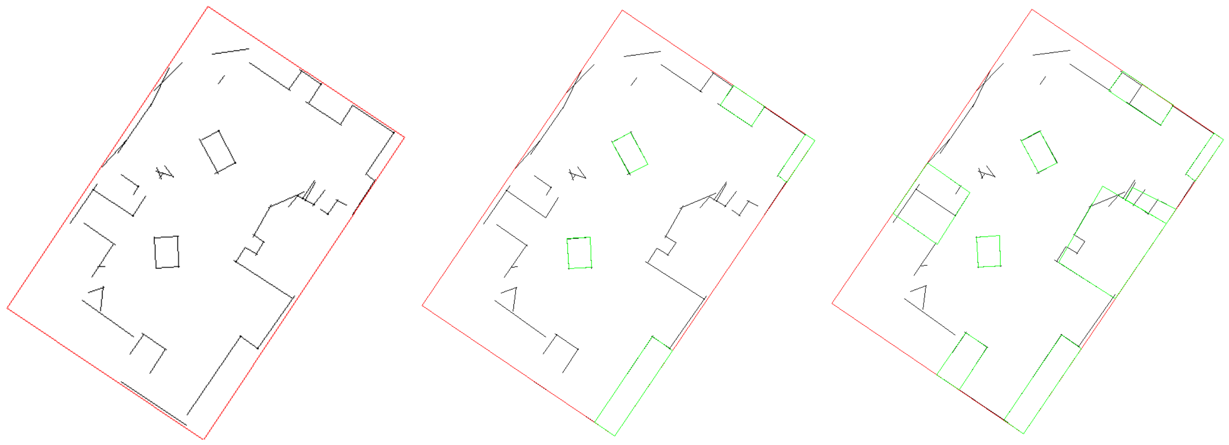


Figure 3.5: RG Map and the odometry-corrected raw data

3.3 Reconstruction of Rectangular 3D Objects

Model based object recognition, localization and tracking has been a field of extensive research for over thirty years. Many approaches are based on the object's appearance in the image by means of 2D shape [79, 116, 5], texture, e.g. using color histograms [105, 104] or image signatures like the parametric eigenspace [10], just to name a few. Ray and Weiss presumed a 3D wire frame model and used invariants to match the model with an edge image [118]. Similar methods are used in the context of object tracking. It is commonly assumed that the model and an approximate initial position are given. Special effort has been put into face detection because of its importance in human machine interaction as well as in surveillance and rescue scenarios. There exist fast algorithms that initialize and track faces in image sequences [117, 96, 89].

Other researchers use point features like the Harris corner detector [49] or SIFT by Lowe [75, 98] to learn object classes. They extract those features from images, cluster them and decide which features are important for a given unknown class of objects. It is assumed that in the set of training images the most or only stable image content is of the class to be learned. In addition, the configuration of those point features is learned once it has been decided which features are meaningful [36]. Although these approaches manage to automatically detect objects of a pre-learned class in an image, it is difficult to incorporate this information into a robot map. This is due to the fact that point features (and their texture environment) change strongly with lighting conditions and view point, and little is learned about the general outlines of those object classes.

In this section we present an algorithm for automatically acquiring models of arbitrary and unknown rectangular 3D objects based on stereo vision. The algorithm makes little assumptions on the object class in the first place, besides that the objects share some general geometric outline, e.g. rectangular or cubic in 3D. In many indoor environments such as office buildings, hospitals or museums, rectangular objects play key roles in building structured maps, localization and human-machine interaction. Examples for such objects and their representation in maps are depicted in Figure 3.6. In particular, in the next chapter we use the rectangular shape of a door frame for the robust detection of doors. Also, many other objects, such as pieces of furniture, e.g. closets, shelves etc., are composed of rectangular objects. Once the object has been reconstructed, one could apply methods from above to draw conclusions about its class, e.g. door, poster, table etc.. The main advantage is that the description of objects in terms of geometric primitives, like 3D rectangles or 3D cubicles, is very compact and holds enough information for navigation.

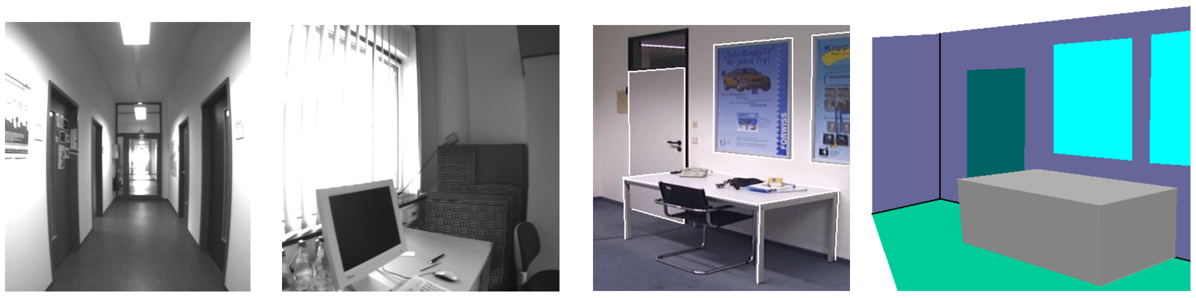


Figure 3.6: **left images:** Many of the task-relevant objects in office environments have rectangular shape, e.g. doors, monitors, shelfs etc.. **right images:** An illustrative view of the visual recognition of rectangular objects in images and how they are stored in maps.

The acquisition of object models is performed as follows. First, object hypotheses, i.e. quadrangles in $\mathcal{R}^2$ that may be projections of rectangular objects in $\mathcal{R}^3$, are extracted from a single image. Then, the plane normal n_Π is calculated based on projective geometry. Given a depth measurement for at least one corner point, all other corner points of the rectangular object can be reconstructed in $\mathcal{R}^3$ by ray-plane-intersection. This yields the description of an object in the camera coordinate system (CCS). Finally, region-based (or global) registration, e.g. scan matching based on laser range data [44], is applied to transform those object representations into a region based (or global) coordinate frame.

In the ideal case, the presented algorithm provides perfect reconstruction. In this case "ideal" means that no spatial discretization by the sensor and exact depth measurements for single points are assumed. Both are obviously not the case in a real world application, but practical experiments have shown that the approach is robust against such error sources, i.e. discretization or quantization, and erroneous measurements. Furthermore, most false hypotheses are automatically rejected in the step of ray-plane intersection.

Our method advances the state of the art in object acquisition and map building in the following ways. It uses methods from stereo image processing and projective geometry to detect 3D rectangular objects and autonomously acquire 3D models for them. Rectangular and cubic objects, which are composed of rectangles, are probably the single most important object classes in many indoor environments. The method is shown to be impressively accurate and to work for a wide range of distances and viewing angles.

The remainder of this section is organized as follows. Subsection 3.3.1 gives a short description of how we generate quadrangle hypotheses from the images. The method for estimating the plane normal of the 3D rectangle and the calculation of the 3D corner points is described in Subsections 3.3.2 and 3.3.3, respectively. Finally, we evaluate the presented algorithm (Subsections 3.3.4) and show how the generated objects can be integrated into a global or region-based metric map (Subsections 3.3.5).

3.3.1 Generating Object Hypotheses in the Image Plane

Pixels in the image that represent edges are detected by an edge filter, e.g. Sobel, Canny [18], Deriche [27] etc.. The pixels are then concatenated to line segments. The resulting set of line segments is compressed applying the merging algorithm described in Subsection 3.1.1. Thereby, we have choosen the following parameters:

$$\alpha_{max} = 1\ degree; \quad D_L^{max} = 2\ pixel; \quad D_{gap}^{max} = 70\ pixel$$

Additionally, small single line segments ($< L_{max}$) are rejected. Based on this set of M line segments, hypotheses are generated by grouping according to Section 3.1. For images, we use the non-restrictive case with the following parameters:

$$\alpha_{min} = 20\ degree; \quad \alpha_{max} = 160\ degree;$$
$$DN_{max} = 25\ pixel; \quad DO_{min} = 15\ pixel$$

For two reasons, the line merging is reversed after grouping:

1. Merging as presented in Section 3.1 introduces small inaccuracies in the measured line segments.
2. Measurements from images are already erroneous, due to the discretization in the sensor.

In order to achieve the best accuracy in the reconstruction, it should be based on the real image measurements. Therefore, after grouping, several hypotheses are extracted from one according to the line segments that originally contributed to a merged line segment. Some of which can be immediately disregarded by topology/plausibility checks. As a result we obtain sets of four line segments, which describe the quadrangle hypotheses.

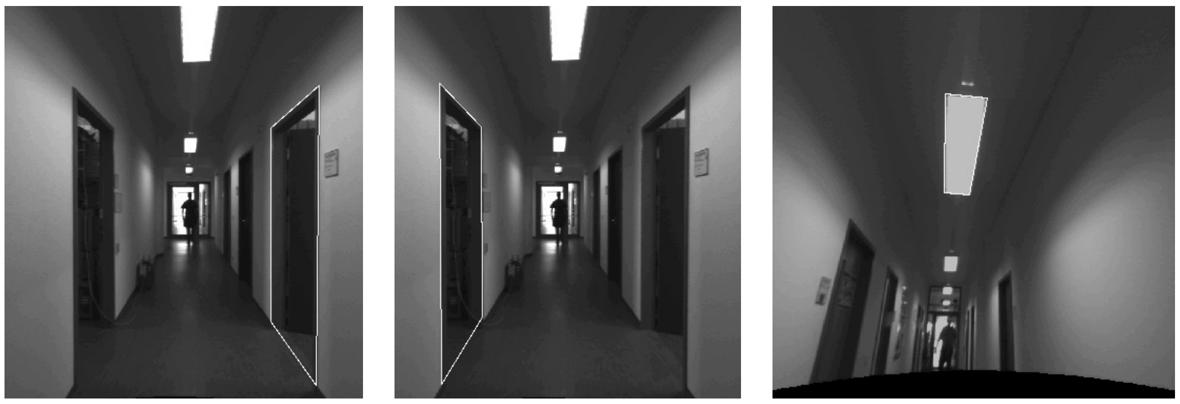


Figure 3.7: Examples for quadrangular object hypotheses in the image plane

3.3.2 Estimating the Plane Normal

Assuming a rectangular object, which lies in the plane Π, we estimate the plane normal n_Π by means of projective geometry. In the projective space $\mathcal{P}^n$, points P^P and lines L^P are specified in homogeneous coordinates (of dimension $n+1$), in particular for $P^P, L^P \,\epsilon\, \mathcal{P}^2$:

$$P^P = (p_1, p_2, p_3) \;\; and \;\; L^P = (l_1, l_2, l_3)$$

$$\text{where } P^R \,\epsilon\, \mathcal{R}^2 = (x, y) = f(P^P) = (\tfrac{p_1}{p_3}, \tfrac{p_2}{p_3})$$

The cross product of two points in $\mathcal{P}^2$ defines the line in $\mathcal{P}^2$, which connects those two points. And similarly, the cross product of two lines in $\mathcal{P}^2$ defines the intersection point of those two lines in $\mathcal{P}^2$. Considering two lines which are parallel in $\mathcal{R}^3$ with their projection on the image plane being L_1^P and L_2^P. The intersection of L_1^P and L_2^P is called vanishing point P_v^P.

$$P_v^P = L_1^P \;\times\; L_2^P \tag{3.1}$$

Taking all possible sets of parallel lines in $\mathcal{R}^3$, which lie in the same plane (including rotation, translation and scaling within the plane), results in a set of vanishing points in $\mathcal{P}^2$, which lie all on the same line in $\mathcal{P}^2$. This is the vanishing line L_v^P. For each plane in $\mathcal{R}^3$ (and its parallels), there exists exactly one vanishing line in $\mathcal{P}^2$, and the corresponding relationship can be utilized to calculate the plane normal in $\mathcal{R}^3$ (see equation 3.2). For more detail on projective geometry and the point-line-dualism in $\mathcal{P}^2$ refer to the well-written book by Hartley and Zissermann [50].

Given a calibrated camera with camera matrix C, the plane normal n_Π in $\mathcal{R}^3$ is calculated from the vanishing line by:

$$n_\Pi = C^T \cdot L_v^P \tag{3.2}$$

The camera parameters are determined in advance by calibration procedures. Hence, radial distortion can be removed from the images. We assume the principal point (C_x, C_y) to lie in the center of the camera coordinate system (CCS). That means, it is necessary, to perform a translation by (C_x, C_y) on the point features extracted from the image, prior to calculating the plane normal. The following camera matrix C results from those considerations and is used to determine the plane normal:

$$C = \begin{pmatrix} \frac{f}{s_x} & 0.0 & 0.0 \\ 0.0 & \frac{f}{s_y} & 0.0 \\ 0.0 & 0.0 & 1.0 \end{pmatrix}$$

Where f denotes the focal length and (s_x, s_y) refer to width and height of a pixel, respectively. Note that (C_x, C_y) must also be known.

The algorithm can be summarized as follows:

1. Transforming the given lines from $\mathcal{R}^2$ to $\mathcal{P}^2$.
   A line L_i^R is described by two points $(P_k^R, P_l^R) \in \mathcal{R}^2$.
   We transform the single points from $\mathcal{R}^2$ into $\mathcal{P}^2$
   and determine the line in $\mathcal{P}^2$ with the cross product:
   $$P^P = (p_1, p_2, p_3) = f(P^R, C_x, C_y) = (\ x - C_x,\ y - C_y,\ 1.0\ )$$
   $$L_i^P = P_k^P \times P_l^P$$

2. Calculate the vanishing points P_{1v}^P, P_{2v}^P for the two sets of opposite lines with equation 3.1.

3. Calculate the vanishing line $L_v^P = P_{1v}^P \times P_{2v}^P$

4. Calculate the plane normal with equation 3.2

3.3.3 Ray-Plane Intersection

Considering a pinhole camera model, all rays intersect in the camera center P^C, which has been chosen to be the origin of the CCS. The original point P_i in $\mathcal{R}^3$ and the projected point P_i' both lie on this ray and the following relation ship holds:

$$\overline{P^C P_i} = P^C + \lambda_i \cdot \overline{P^C P_i'} \quad i \,\epsilon\, (1...4)$$

$$P_i' = (x_i', y_i', f) \,\epsilon\, \mathcal{R}^3, \text{ f is the focal length}$$

and since $P^C = (0,0,0) \,\epsilon\, \mathcal{R}^3$:

$$P_i = \lambda_i \cdot P_i' \quad i \,\epsilon\, (1...4) \tag{3.3}$$

On the other hand, a vector v_1 can be constructed, which is orthogonal to n_Π, i.e.:

$$n_\Pi \cdot v_1 = n_x \cdot v_{1_x} + n_y \cdot v_{1_y} + n_z \cdot v_{1_z} = 0$$

And from there, vector v_2 can be determined according to:

$$v_2 \;=\; n_\Pi \;\times\; v_1$$

Given one corner point P^{cr} of the rectangular object in $\mathcal{R}^3$ we can derive a plane equation

$$P_i = P^{cr} + \mu_i^1 \cdot v_1 + \mu_i^2 \cdot v_2 \quad i \,\epsilon\, (1...4), \quad \mu_i^1, \mu_i^2 \,\epsilon\, \mathcal{R} \tag{3.4}$$

All other 3D points can then be calculated by ray-plane intersection, i.e. solving the system of linear equations (3.3, 3.4) for the points P_i in $\mathcal{R}^3$, i.e. set equation 3.3 = 3.4 and solve for λ_i. In fact it is not neccesary to compute μ_i^1 and μ_i^2, except for performing tests on the results for debugging purposes. Due to erroneous measurements for the corner points in 3D, they must not exactly lie on the respective ray $\overline{P^C P_i'}$ from the camera center. Therefore, all given 3D corner points are first orthogonally projected onto the ray $\overline{P^C P_i'}$. This gives a very efficient method to fuse the results for different corner points, which straight-forward would be the average over the corner points of the reconstructed rectangles:

- Take one of the given corner points P_1^{cr} to be the base point P_{B1}^{cr}.
- Using the ray-plane intersection by means of equations 3.3 and 3.4 to calculate P_{Bi}^{cr} based on another given corner point P_i^{cr} $(i \neq 1)$.
- Average over the different corner points for P_1^{cr}: $\quad P_{base}^{cr} = \frac{1}{N}\sum_{i=1}^{N} P_{Bi}^{cr} \quad 1 < N < 5$ and calculate the other corner points P_i^{cr} $(i = 2,3,4)$ from P_{base}^{cr} $(= P_1^{cr})$

Note, the points P_i^R $(i = 1..4) \,\epsilon\, \mathcal{R}^2$ from the image must first be transformed into the CCS. Thereby, the z-coordinate is the focal length.

It is also possible to choose an arbitrary distance for one corner point and determine all 3D points by means of equation 3.3 and 3.4. If the considered 2D quadrangle is in fact a projection of a 3D rectangle, the reconstruction is correct up to a scaling factor. That means, in general it is possible to verify or reject the hypothesis without any depth measurements.

3.3.4 Evaluation

The presented reconstruction algorithm is exact, if we neglect discretization in the image sensor and inaccuracies of depth measurements. That means, given a projected 3D rectangle, i.e. a quadrangle in 2D, the plane normal can be exactly calculated. Given the ray-plane intersection, the 3D rectangle can be determined up to scaling and translation along the optical axis. Both factors are directly correlated. Adding the 3D coordinates of one corner point, defines the rectangular object uniquely.

In this subsection, we will empirically evaluate the accuracy and robustness of the model acquisition of rectangular objects from stereo. We have stated in the introduction of this section that for non-discrete sensors with accurate 3D measurements, our method is exact. In this section we will vary the discretization resolution and apply the reconstruction method to images captured by our B21 robot in order to evaluate the method under realistic conditions and in real settings.

To do so, we will first evaluate the influence of pixel discretization on reconstruction robustness and accuracy based on simulation (section 3.3.4.1). The use of simulation enables us to make more controlled experiments, in which we can vary the discretization resolution of non-discrete image sensors (which cannot be done with the real imaging devices) and access ground truth data easily. We will then assess the impact of inaccuracies in 3D measurements based on results that are obtained in experiments with our B21 robot (3.3.4.2).

3.3.4.1 Influence of spatial discretization by the image sensor

The experimental setting for assessing the impact of discretization in the image sensor on the reconstruction results is as follows. We first generate a virtual image sensor for the specified resolution of the image sensor discretization. We then systematically generate sets of rectangular objects in 3D space, project them into 2D space in order to produce the input data for our virtual sensor. Our 3D reconstruction method is then applied to the output of the virtual sensor and then compared to the real 3D position of the rectangular object. We classify the outcome into three categories: first, the reconstruction failed; second, the reconstruction succeeded but was inaccurate; and third, the reconstruction was accurate.

To give an intuition about which 3D poses can be accurately reconstructed and which ones not, we have visualized both sets in Figure 3.8. In this particular experiment, we have placed a 3D rectangle at z = 1 meter and the camera at z = - 3 meter. Then the rectangle was rotated stepwise in 3D by 36 degree around the z-, x- and y-axis, which gives us a total of 1000 3D poses of the object. The parameters of the virtual camera are set to 766×580 pixels resolution, sx≈sy= $8.3 \cdot 10^{-3}$ meter, C_x=391.685, C_y=294.079 and focal length of 0.00369 meter, which are the estimated parameters of the cameras on the robot. The accuracy is defined as the length of the error vector between the reconstructed and the original 3D point.

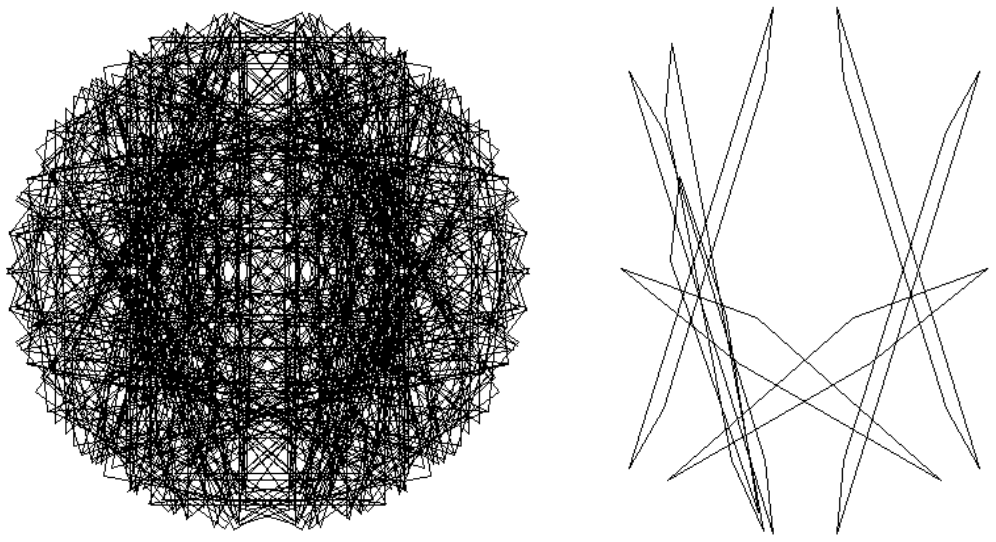


Figure 3.8: Reconstructible (left) and non-reconstructible (right) projections due to spatial discretization in the image sensor

Our results show that despite the inaccuracies caused by discretization, the algorithm still gives very good result. From some projections reconstruction is not possible. But those cases can clearly be classified, as can be seen later. Figure 3.8 (left) shows the set of projections that can accurately be reconstructed. These are 968 of the 1000 poses that are reconstructed with an error of less than 10 centimeters. In the majority of cases the error is substantially smaller. Figure 3.8 (right) shows the 32 projections which cannot be reconstructed. It can be seen that these 2D quadrangles are characterized by very large (more than 150 degree) and very small internal angles. We have also performed this experiment with smaller angle step sizes in order not to miss any other special cases. But there are none. In fact, the percentage of non-reconstructible projections is always about eight percent of the whole considered set. This is due to the before mentioned special cases of quadrangles. Furthermore, we found that projections with bad reconstruction results are close to the configuration where the reconstruction failed.

To get additional evidence for these findings, we have learned rules for predicting whether or not a reconstruction of a rectangular object will be accurate or at least possible. We have trained a decision tree with the angles of 64000 projected objects and whether or not the reconstruction has been successful. The angle α for each of these projections was determined as follows:

$$\alpha \;=\; \textstyle\sum_{i=1}^{4} |\alpha_i \;-\; \frac{\pi}{2}|$$

Where α_i denote the internal angles of the quadrangle in $\mathcal{R}^2$. We take the deviation of the internal angles to 90 degree as criterion, because we assume from the experiments above that reconstruction problems result from very large and very small internal angles. The learning algorithm came up with compact rules that predict when the reconstruction will fail and when it will succeed. As a result, two rules have been learned:

```
        Rule 1:
                sum_angle <= 284.865
                ->  class true  [98.2%] (53796 used samples)
        Rule 3:
                sum_angle > 314.908
                ->  class false  [79.2%] (3992 used samples)
```

For interpretation purposes we also consider the average deviation: $\alpha_{mean} = \frac{\alpha}{4}$
Rule 1 says that if α is smaller than 284 degrees the prediction will succeed. This rule has captured more than 80% of the cases and a prediction accuracy of 98%. That means, we have a high probability that the reconstruction will succeed for $\alpha_{mean} < 71\ degree$, hence for $19\ degree < \alpha_i < 161\ degree$.

The second rule that predicts failure applies if α is larger than 314 degrees with an expected prediction accuracy of about 80%. With the same rationale, we can conclude that $\alpha_i > 168\ degree\ and\ \alpha_i < 12\ degree$ causes critical cases. We conducted the same experiment for projections with good and bad reconstruction results, where bad refers to errors larger then 5 centimeters. Thereby the cases where the reconstruction would fail have been neglected. In this case, more than two rules have been learned, but the conclusions are the same. In a certain range of angle α_i, we have a high probability that the reconstruction error is smaller than 5 centimeters, and outside of this range the error increases. That means, by limiting the internal angles for the quadrangle hypotheses to be considered, we can not only guarantee with high probability that the reconstruction will succeed, but also that the error does not exceed a certain upper bound.

We conclude from these experiments that the presented algorithm can robustly deal with effects of pixel discretization, and rectangles can be very accurately reconstructed, as long as the internal angles of the projection, i.e. the 2D-quadrangle, lie in a certain range around 90 degree ($26\ degree < \alpha_i < 154\ degree$).

3.3.4.2 Influence of inaccurate 3D-measurements

We have studied the influence of errors in depth measurements with real world image data. Depth measurements have been generated with the Small Vision System from SRI [59]. As stated above inaccuracies from 3D measurements can be evaluated by means of scaling errors. We have chosen the width and height of the rectangular objects as the criterion for quality measures. In all figures the depicted white quadrangles are backward projections of the reconstructed 3D rectangles into the image. It can be seen that these back projections fit very well to the image data. The question is, how well the reconstructed rectangle describes the actual rectangular object.

Very prominent rectangular objects in a variety of indoor environments are door frames. All door frames depicted here (fig. 3.9) are 210 centimeter high and 101 centimeter wide. Smaller 3D rectangles include for example posters or white boards (fig. 3.10, left) as well as computer monitors (fig. 3.10, right) in office environments. The noisy depth measurements are depicted on the right in all those figures. Table 3.1 summarizes the reconstruction results for the presented examples.

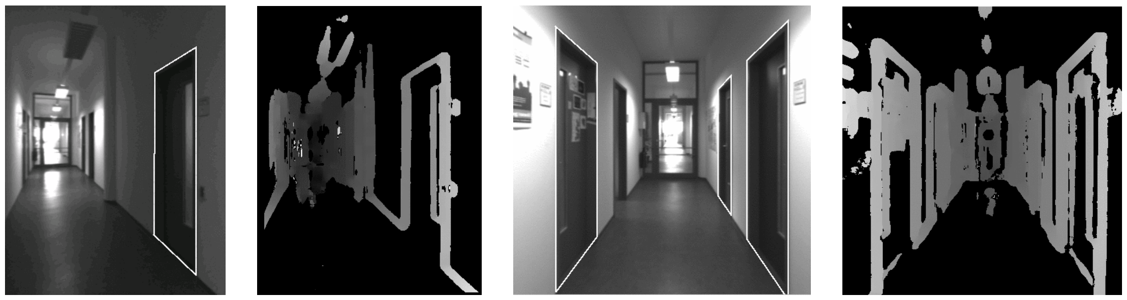


Figure 3.9: Door frame experiment 1 and 2

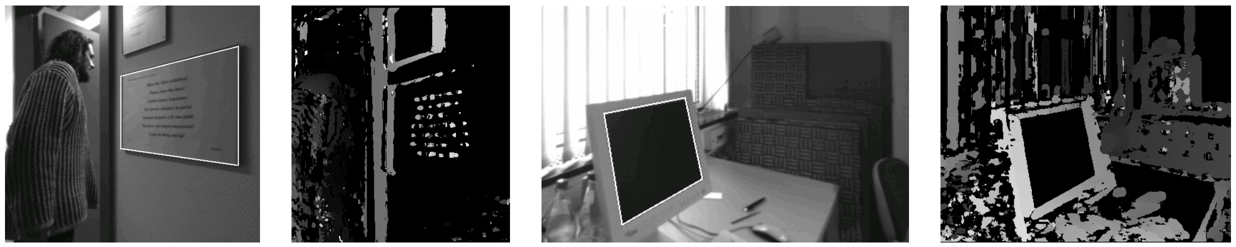


Figure 3.10: White board in a hallway (left), Monitor in office environment (right)

experiment	original size[1]	reconstructed size[1]
Door 1	210x101	207.2x104.3
Door 2, LeftFront	210x101	208.5x105.3
Door 2, RightFront	210x101	207.9x105.6
Door 2, RightBack	210x101	192.0x107.8
White board	21.4x30.2	21.1x30.9
Monitor	27.0x33.8	27.5x34.04

Table 3.1: Summary of reconstruction results.

[1] The values for original and reconstructed size denote height×width in centimeters.

It can be seen that the reconstruction for different scenarios and rectangular objects is very accurate. This leads to the conclusion that the presented approach is very well suited for automatic generation of representations for 3D rectangular objects in indoor environments. The accuracy increases if more than one 3D corner point can be measured and is given to the reconstruction algorithm.

3.3.5 Integrating 3D Objects into a Common Frame of Reference

The reconstruction algorithm presented in this section generates 3D object hypotheses which are described metrically with regard to the robot coordinate system. Since the robot moves around in the environment, objects are reconstructed from different positions. Thus, in order to determine a consistent region description, the 3D objects have to be aligned with the 2D line segment map generated from the laser data, which corresponds to a transformation into a common frame of reference, either global or region based. This is done using the corrected odometry measurements generated by laser-based metric SLAM [44]. Given an object in robot coordinates the idea is to obtain a pair of poses, i.e. raw and corrected odometry, that define a transformation between the two coordinate systems. This transformation is commonly assumed to be locally linear and constant and can than be applied to the given object.

Even though the incremental scan matching corrects the odometry readings this approach turned out to be nontrivial because the mapping algorithm that was applied optimizes the position for a chain of observations. That means for the current pose it converges after several new poses and scans have been incorporated. The general problem is that in fact the transformation between the raw and corrected odometry coordinate system is nonlinear and not constant due to the arbitrary odometry error. Additionally, the odometry correction is naturally performed in discrete steps. That means, scans are matched for discrete positions having a certain difference in position and orientation. Given an observation from stereo or laser and the respective odometry pose, we first search for the closest odometry pose available from the scan mapping process. Defining "closest" in terms of Euclidean distance might introduce arbitrary errors because the resulting pose might refer to an earlier visit of the same place. Therefore the search is further specified by a scan ID. That is, each scan has a unique ID. When objects are reconstructed from image data, this ID is retrieved and stored with the objects.

To increase the accuracy of this transformation, laser scans could be integrated more often within the scan matching process, i.e. at shorter distance or angle intervals. Obviously, this leads to more data storage and slower matching performance. Another possibility is to acquire a scan whenever an object has been reconstructed and to use this scan along with scan matching to determine the correct transformation. We used both those ideas in our experiments. In praxis, however, this is not suitable due to the mentioned drawbacks. Here the transformation of objects and the fusion of observations must be delayed until, and/or repeated when reliable pose estimates are available.

3.4 Summary

In this chapter we first presented an algorithm that generates 2D quadrangle hypotheses from a set of 2D line segments. We showed its use for finding objects of rectangular shape in 2D line segment maps generated from laser range data. For vision based reconstruction we applied the grouping algorithm to line segment sets obtained from image segmentation in order to find quadrangle hypotheses. These hypotheses present the input to a new algorithm that automatically acquires models of arbitrary and unknown rectangular 3D objects. We showed that the resulting models are very accurate and that the presented method is robust with respect to the discretization of the image sensor and erroneous depth measurements. In addition, we can build predictive models for the accuracy and success of applying the method to a given group of lines which can be used in autonomous exploration. That means, if the robot detects that a particular reconstruction is not promising it can infer a position from which the reconstruction of the hypothesized object is likely to succeed.

The reconstruction results can be additionally verified by projecting the reconstructed rectangle into the image of the second camera. Thereby, the observations (line segments) of that camera are compared with the projected (reconstructed) 3D rectangle to either reject or accept the hypotheses. Also, it can be expected that tracking those 3D object hypotheses over time and merging the corresponding observation will lead to a further increase of robustness and accuracy.

Chapter 4

Gateway Detection

From Chapter 2 we know, that gateways form perceptually recognizable, characteristic transitions between two or more adjacent regions. They can be traversed in any direction and are the only possibility to pass from one region into another. It is a key problem of the presented mapping algorithm to detect and recognize gateways, when the robot approaches or traverses them. This is not a drawback in itself, but a natural necessity for the generation of structured maps. Nevertheless, it is a difficult task, and to date there exist only few approaches which tackle it [62, 4, 3].

To illustrate the importance of gateway detection in the context of RG Mapping, lets consider the following two scenarios:

Exploration. During exploration, a map is built and continuously updated by the mapping process. That means the Region & Gateway Map is automatically structured based on the gateways detected in the sensor data. The actual class of the gateway is not important in the first step, but the fact that some kind of gateway might be present is essential for the structuring process. This is because it is more difficult to split a region than to merge two.

Navigation. For navigation through explored areas, path planning and localization are based upon the present region data. After gateway traversal, the system has to switch to the new region data. If a gateway is not detected, the system may fail to perform this context switch which in turn may cause the loss of global localization.

In this chapter we present and evaluate algorithms to detect and recognize two main classes of gateways. The first comprises crossing gateways in Section 4.1, which refer to hallway crossing and junctions like *L/R-Turns* and *T-Junctions*. The second deals with *Narrow Passages*, in particular doors of different form, e.g. width, double doors etc., see Section 4.2. Whereas crossing gateways are detected and classified solely based on laser range data, doors are found using both laser and vision data.

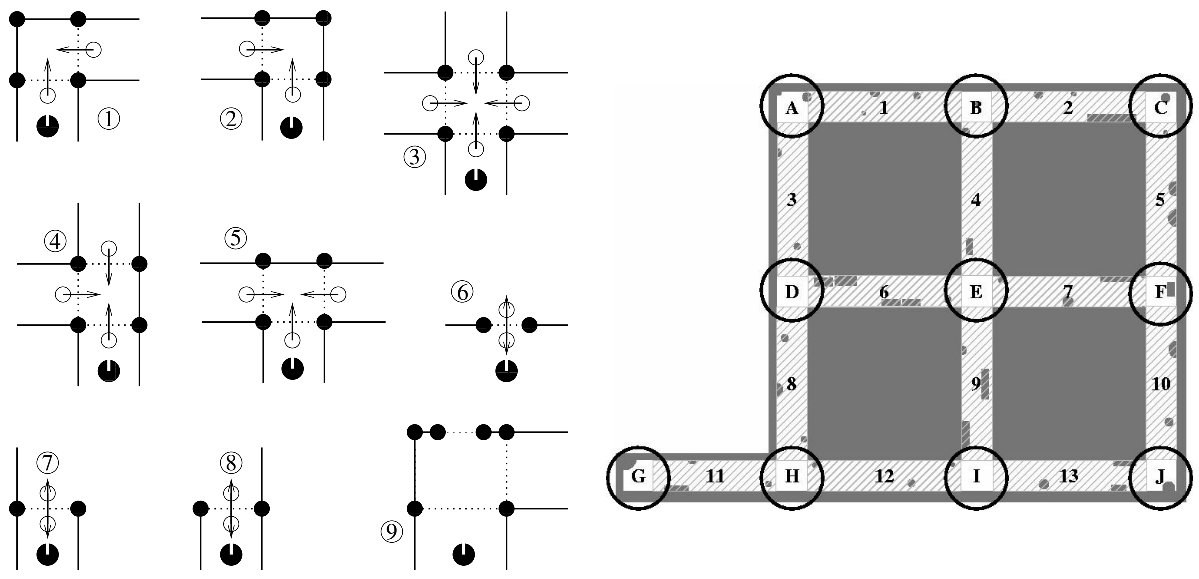


Figure 4.1: :**Left**: Classes of Gateways - **R**_ight/_**L**_eft-Turn_ (1,2), _X-Crossing_ (3), **L**_eftT/T/_**R**_ightT-Junction_ (4,5), _Narrow Passage_ (6), Right/Left Opening (7,8), Combination of Gateways (9); (• gateway point, ∘ crossing point, ← traversal direction, **...** region border); **Right**: Example environment - letters denote gateways, numbers denote regions

4.1 Laser-Based Detection of Turns & Crossings

We will investigate the gateway recognition problem by first introducing _virtual line models_ (VLMs) as an appropriate feature language for the detection and classification of crossing gateways. It is described in detail how VLM hypotheses can be computed from laser scan data. Next, we introduce different measures for VLMs constructed from laser scans that enable us to assess the similarity of the perceived VLM hypotheses and the different gateway classes. Based on these measurements we present different classification methods that we empirically evaluate and compare, see Subsection 4.1.4.

The computational problem of recognizing and classifying crossing gateways can be formulated as follows: Given a single scan or a sequence of scans provided by a laser range finder and a set of gateway models, autonomously detect and classify crossing gateways. We will solve the gateway recognition problem in a computational process that executes a sequence of three steps:

1. Generating hypotheses for virtual line models (VLMs), Subsection 4.1.1

2. Determining weights according to general and specific gateway models,
   Subsection 4.1.2

3. Using the generated observation vectors and sequences for classification,
   Subsection 4.1.3

4.1.1 Generating Hypotheses for Virtual Line Models (VLMs)

In order to represent gateway hypotheses we propose *virtual line models* (VLMs) as an appropriate feature language. VLMs are based on the assumption that environments are rectangular and hallways have approximately the same width. The VLM consists of a left, right and front virtual line as well as a hidden virtual line, see Figure 4.2. To generate VLM hypotheses, we first extract low-level features, i.e. virtual lines and depth singularities from the line segment and point scan, respectively. In the next step, those virtual lines are grouped to form hypotheses with respect to the VLM as depicted in Figure 4.2. To generate estimates for the hidden line we use the depth singularities and prior knowledge about the hallway width. An extension of this approach could take additional point features into account, e.g. corners, and intersections.

4.1.1.1 Low-level Feature Extraction

In the first processing step the algorithm generates a line segment scan L_{LS} from the point scan L_P by means of linear regression according to [42]. Additionally, depth singularities are determined based on the point scan.

Virtual Lines. Line segments from L_{LS} which lie approximately on the same line are grouped and represented by that line, also referred to as virtual line, see Figure 4.2 (left and middle).

Depth Singularities. This point feature is extracted from L_P and denotes discontinuities in the distance measurements of a laser scan, see Figure 4.2 (right). The parameter Δd_{min} indicates the minimum distance difference of two consecutive distance measurements to represent a depth singularity. $P_i \in L_P$ is the point where the distance measurement d_i ends. P^i_{ds} are the points at the depth singularities.

$$P^i_{ds} = \{P_i \ : \ (d_i < d_{i\pm1}) \ \wedge \ (|d_i - d_{i\pm1}| > \Delta d_{min})\}$$

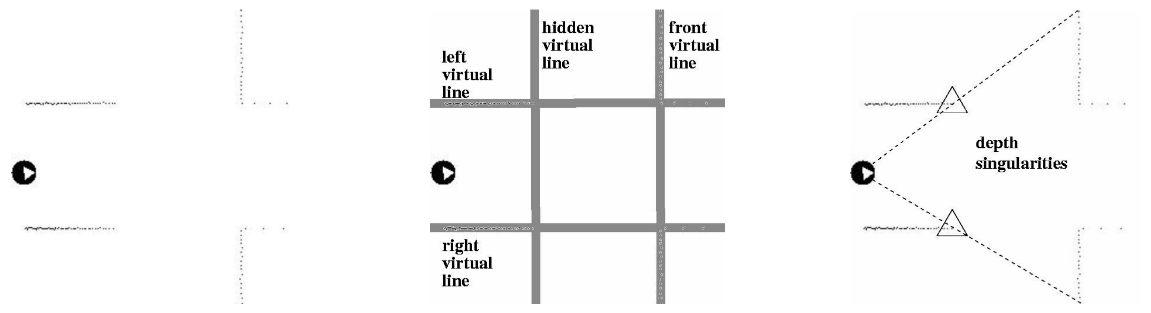


Figure 4.2: Point scan of an *X-Crossing* (left); Virtual Line Model for Crossing Gateways, i.e. *X-Crossing, L/R-Turn, T-Junction* (middle); Depth singularities in a laser scan (right)

4.1.1.2 Virtual Line Grouping

Based on the virtual lines and depth singularities we generate hypotheses for VLMs, which indicate that a crossing gateway of some kind may be present. To generate candidates for the virtual left and right line, we search for parallels among the virtual lines, where the robot is in between. Virtual front lines intersect a pair of parallels approximately in a right angle and in front of the robot. Finally, we estimate the virtual hidden lines. Therefore, we consider depth singularities that are close to the virtual left or right line. The hidden line is constructed such that it is parallel to the virtual front line and intersects with the given depth singularity. To deal with situations where no valid depth singularities are present in the point scan, e.g. due to clutter, we add hypotheses where the estimation of the hidden line is solely based on the environment assumptions. That means, the distance to the virtual front line is given by the hallway width. This may lead to identical VLMs, which are deleted before further processing. As a result, we obtain a set of annotated virtual line quadruples, which represent hypotheses for VLMs. The annotation of the virtual lines refers to the VLM in terms of virtual left, right, front and hidden line, see Figure 4.2 (middle). The gateway points are defined by the intersections of those virtual lines.

4.1.2 Evaluating the VLM Hypotheses

In the previous subsection we introduced the virtual line model (VLM) as a feature language for the recognition of crossing gateways, and described how VLM hypotheses can be generated from laser scans. Given such hypotheses and the raw sensor data, the task is to assess the similarity of the perceived VLM and the different gateway classes. We propose the following two measures. The first reflects the general model quality with respect to rectangularity and hallway width, the second accounts for the match with a specific gateway class. As a result we obtain an observation vector for each VLM hypothesis. Additionally, we track VLM hypotheses over consecutive measurements while the robot is moving towards the gateway to generate observation sequences.

4.1.2.1 Measures with respect to the General Model

To quantify the similarity of the generated VLM hypotheses with a given VLM, we compute measures, which scale with the deviation of certain properties from the model, namely the deviation from the expected hallway width and rectangularity. Considering the deviation for a distance measurement between two gateway points, small inaccuracies mainly arise from imperfect measurements whereas inaccurate or false hypotheses occur due to clutter (people, flower pots). The same holds for the inner angles of the convex quadrangle given by the VLM hypotheses.

Distance Measure. The expected hallway width d_{hw} has been manually measured. Deviations from this value are weighted according to:

$$w^i_{distance} = 1 - \sqrt{\frac{|d_i - d_{hw}|}{d_{hw}}}$$

$$W_d = 0.25 \cdot \sum_{i=1}^{4} w^i_{distance}$$

Whereas d_i is the Euclidean distance between two neighboring gateway points and W_d denotes the averaged distance weight.

Rectangularity Measure. The rectangularity criterion refers to the inner angles α_i ($i = 1...4$) of the convex quadrangle, given by the VLM. We define the rectangularity by the deviation of the inner angles α_i from $\frac{\pi}{2}$.

$$W_r = 1 - \frac{1}{2\pi} \cdot \sum_{i=1}^{4} |\alpha_i - \frac{\pi}{2}|$$

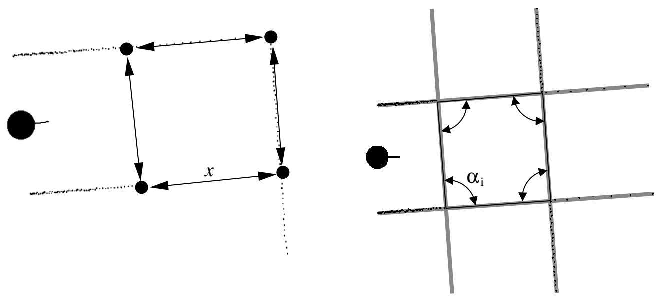


Figure 4.3: Distance measurement (left) and rectangularity (right)

4.1.2.2 Measures with respect to a Specific Gateway Class

As we have seen in Subsection 4.1.1 the VLM is a general model that does not explicitly account for specific classes of crossing gateways. In this section we introduce measures, which support the discrimination between different gateways. Thereby, it is important to ensure comparability of those measures.

Freespace Measure. Considering the VLM as depicted in Figure 4.2 (middle), we define three pairs of gateway points, namely on the virtual left, right and front line. According to those pairs of gateway points, we divide the sensor data into three sectors, Figure 4.4. Each sector S_i comprises N_{S_i} measurements. Based on those definitions we propose the freespace measure (FSM) as a quantity for the match of a hypothesis to the sensor data. In each of the three sectors the sensor measurements should either be close to a given line (*On Line FSM*) or should cross a given line (*Over Line FSM*). The gateway class determines which of the two FSM variants applies to a certain sector. For example, considering an *L-Turn* the measurements in the front sector are expected to match the virtual front line (*On Line FSM*). Whereas for an *X-Crossing*, measurements in the same sector are expected to cross the virtual front line (*Over Line FSM*), see Figure 4.4.

On Line FSM. P_i denotes a laser measurement from the point scan L_P and $d(P_i)$ is the respective distance measurement. We compute a point P_i^{vl} on the considered virtual line and its distance to the robot $d(P_i^{vl})$, whereas P_i and P_i^{vl} lie on the same ray from the robot. Then we count all measurements for which the difference of $d(P_i)$ and $d(P_i^{vl})$ lies between a given lower and upper threshold. Finally, we normalize this on-line-count (C_{ol}) with the overall number of measurements in the sector:

$$W_{on\ line}^{S_i} = \frac{C_{ol}^{S_i}}{N_{S_i}}$$

Over Line FSM. This measure only applies to the front sector. We construct a line l_{par} parallel to the virtual front line and set back by a given distance. Then we count all measurements which intersect l_{par}, and normalize the resulting over-line-count. Analogous to the *On Line FSM* we get $W_{over\ line}^{S_i}$, see also Figure 4.4.

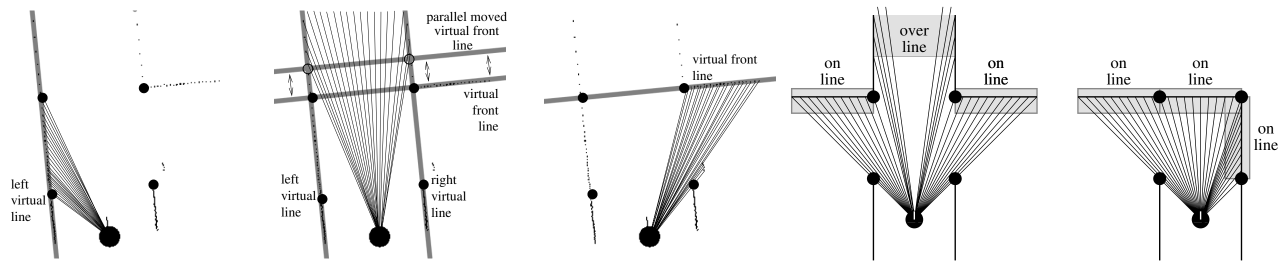


Figure 4.4: **From Left to Right:** Freespace measure (FSM) for different cases - *On Line FSM* (1,3), *Over Line FSM* (2), (• gateway point, — laser scan measurement, - - line for free space evaluation); FSM configurations for *X-Crossing* (4) and *L-Turn* (5)

4.1.2.3 Generating Gateway Weights and Observation Sequences

Utilizing the proposed measurements, we define weights for each VLM hypothesis with respect to a certain gateway class GW:

$$W(GW, VLM) = \frac{f_{vlm}}{2} \cdot (W_d(VLM) + W_r(VLM)) + \frac{f_{gw}}{3} \cdot \sum_{i=1}^{3} W_{FSM}^{S_i}(GW)$$

Whereas f_{vlm} and f_{gw} denote weighting factors for the general and gateway specific measurements, respectively. As a result we obtain an observation vector for each VLM hypothesis, where the entries quantify the similarity of the hypothesis to a specific gateway class. In most practical cases the mobile platform approaches the gateway area. Thus, we observe the same VLM hypotheses from different positions, where the distance to the gateway is continuously decreasing. The VLM hypotheses tracking is based on the gateway points and Euclidean distances. If all gateway points of two VLM hypotheses have an approximate match, they are considered to be identical. Based on this tracking, we obtain sequences of observation vectors. A sequence starts when the hypothesis is first observed and the distance falls below a threshold. It is finished or corrupted when it is either lost or the robot enters the gateway.

4.1.3 Classification

In the previous sections we described a sensor model based on laser range data, which provides observation sequences for gateways in hallway environments, i.e. *X-Crossings, T/LT/RT-Junctions* and *L/R-Turns.* We added two special cases, namely *Deadend* and *Hallway*, to counterbalance situations where an VLM hypothesis may be present, but does not refer to any of the considered gateway classes. As a result we obtained sequences of observation vectors, which are based on the same VLM hypothesis, and contain measurements which have been obtained from different positions. In fact, we assume that the mobile platform is steadily approaching the gateway area, hence the distance is continuously decreasing. The start of the sequences is defined by the first appearance of a certain VLM hypothesis and a maximum distance. The sequence is finished when the robot is closer to that VLM hypotheses than a minimum distance. It is important to note that there are no sequences for approaching a *Hallway*, since this is an explicit counterbalance measurement. For *Deadend* however, VLM hypotheses can be generated, hence we obtain observation sequences. The task at hand is to classify those sequences with regard to a certain gateway class.

Several approaches have been investigated and will be outlined in the following subsections. First, we consider classification methods solely based on the observation vector closest to the gateway area (subsection 4.1.3.1). We extend those classifiers to account for the whole sequence by calculating the weighted average over all observation vectors in a sequence (subsection 4.1.3.2). Finally, we show how Hidden Markov Models (HMMs) can be applied to generate discrete probability distributions over the types of sequences (subsection 4.1.3.3).

4.1.3.1 Single Observation based Classification

Observations close to a gateway imply a more complete coverage of the gateway area by the sensors, hence they are in general the most informative. The first classification method we present ignores the fact that observation sequences have been obtained and considers only the observation vector closest to the gateway area. That means, the single observation classifier (SOC) determines the gateway at hand by selecting the maximum weight in the observation vector. This approach demonstrates the discrimination power of the freespace measurements and the resulting weights. It is, however, very sensitive to sensor noise, occlusions and dynamic changes in the environment. As a matter of fact, in dynamic environments, the last observation is not necessarily the best. For example, when considering *L/R-Turns*, the robot might cut the corner and still observe the gateway. Since the laser sensor does not fully cover the gateway, the classification of the generated hypothesis is likely to fail, see Figure 4.5. Also, since the *Hallway* detector gives large weights in particular for *X-Crossings* and *LT/RT-Junctions*, because there is open space to the front, these weights had to be ignored. As a consequence the SOC makes no use of this counterbalance class, i.e. if an observation vector is present the classifier always decides for one of the gateway classes.

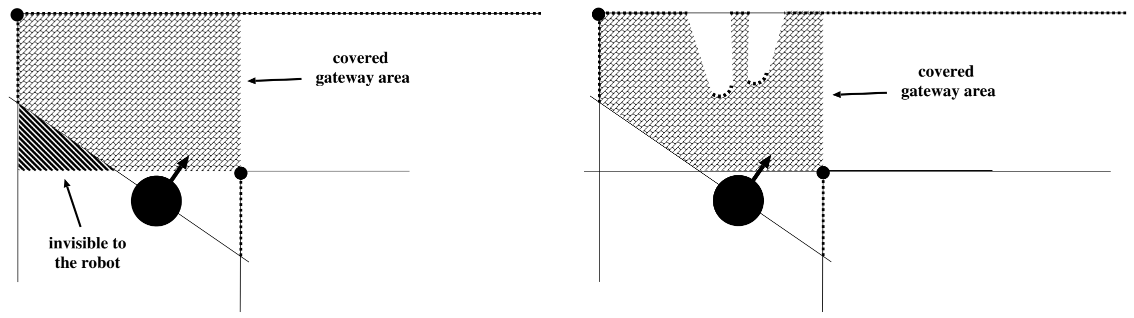


Figure 4.5: Dynamic changes that decrease the number of observable features can introduce arbitrary errors in the classification if only a single view is used.

The approach can be extended by considering special characteristics of the maximum, e.g. distance to the second or third maximum. In this case heuristics would have to be found and applied. We did not further investigate such extensions because we believe that it is clear that the SOC is not sufficient for the given task, see also Subsection 4.1.4.

4.1.3.2 Averaged Sequence based Classification

In order to increase the robustness in case of clutter or ambiguities it seems necessary to evaluate the whole observation sequence. A simple solution is the fusion of consecutive measurements by calculating a weighted average over the observation sequence, where the weights are inversely proportional to the distance. Afterwards, the maximum weight is used to determine the gateway at hand, analogous to the SOC. Whereas the average sequence based classification (ASC) considers the complete observation sequence, it does not fully exploit probabilistic properties of observations and temporal relations between them. Additionally, choosing the weighting function is not an easy task, and finally, *Hallway* measurements had to be ignored by the same rational as for the SOC.

4.1.3.3 Classification based on Hidden Markov Models (HMMs)

A more promising approach to gateway classification is the use of Hidden Markov Models (HMMs). They provide mechanisms to model temporal structures in sequences, by the use of probabilistic observation and state transition models. A detailed description of the theory can be found in [93]. In the next paragraphs we briefly outline the steps necessary to use HMMs in the context of gateway detection based on the introduced sensor model.

Clustering the Data. Since our sensor model provides continuous measurements we use HMMs with continuous outputs, i.e. a sensor model based on mixture of Gaussians. To deal with the implicated complexity of such HMMs we explicitly cluster the data using the k-means-algorithm [78, 33]. The clusters are represented by Gaussian distributions and used to built the observation model (mean and covariance matrices), and to define the structure of the HMM.

We expect the data to represent clusters for different distance intervals, because the coverage of the gateway area by the sensor differs for different positions. The start values for the k-means algorithm are computed accordingly. Theoretically, these values could be chosen arbitrarily, however, in the presented case, the clustering performs better and more robust when they are drawn from the data. Therefore, the training data is first sorted according to distance. The data refers to all observation vectors from all sequences with regard to a certain gateway class. Then we slide a window over the data and calculate k mean vectors. The size of the window is given by k, the number of observation vectors and a predefined window overlap. The resulting mean vectors are fed into the k-means algorithm.

The distance assumption is verified by the fact that the mean vectors are only altered slightly by the k-means clustering. To get a further intuition we labeled each observation vector according to distance intervals: I_1, I_2, I_3. A distance of zero meter refers to the center of the gateway. That means the closest observations we get are half of the hallway width, e.g. 1 meter for 2 meter hallways. After clustering, we sorted the clusters according to the intervals, and counted how much of the prelabeled data has been assigned to which cluster. In Table 4.1 it can be seen that all of the clusters contain a reasonable amount of samples (over all sum), and that clusters are built according to distance intervals (max/min distance). They contain either observations from disjunctive or slightly overlapping distance intervals or represent different distributions for approximately the same interval. Those findings are very important for the choice of the HMM structure and initialization, but they also allow for interpretation of the learned model.

Initializing the Hidden Markov Model. Based on the clustering and the knowledge of the data, the initial observation model and thereby the structure of the HMM is generated automatically. All clusters C_k that cover approximately the same distance interval are assigned to the same HMM state S_i. More precisely, the covariance matrix and the mean of each C_k add a dimension to the observation model of S_i. New states are added to the HMM, for any new distance interval. Considering Table 4.1, we obtain an HMM with five

cluster id	1	2	3	4	5	6	7	8
$I_1 = 7m...4m$	475	508	685	795	2	1	0	0
$I_2 = 4m...2m$	0	0	0	16	277	252	1143	44
$I_3 = 2m...$	0	0	0	0	0	0	32	437
over all sum	475	508	685	811	279	253	1175	481
mean distance	6112	5796.4	5770.2	4455	3796.8	3538.3	2706.2	1755.5
max distance	6872.2	6956.1	6976.3	4968.7	4009.4	4103.6	3591.7	2310.3
min distance	4854.5	4960.8	4878	3954.8	3438.6	3037.8	1775.9	1427.9

Table 4.1: Clustering for *T-Junction* data, columns refer to different clusters, rows depict cluster properties; all distance measurements in millimeter

states, where $[C_1, C_2, C_3]$ present the first state, C_4 the second, $[C_5, C_6]$ the third, C_7 and C_8 the fourth and fifth, respectively. The mixture matrix M_{mix} is initialized uniformly, the dimension is given by the number of states Q and the maximum number of mixture components M. The states are arranged to form a left-right HMM, and according to the sequences, left refers to large and right to small distances. Although, in a left-right HMM consequently all entries below the diagonal of the transition matrix T are zero, we initialized the full matrix with 1/Q. The motivation was to verify that the EM algorithm for learning the HMM parameters, would converge to a left-right HMM, when using the described initialization and observation sequences. The same holds for the prior. We expect the probability that a sequence starts in a certain state, to decrease from left to right. The parameters of the HMM are denoted with λ, according to [93]. Figure 4.6 shows a left-right model, where the arrows denote the possible transitions from state S_i to S_j with probabilities p_{ij}.

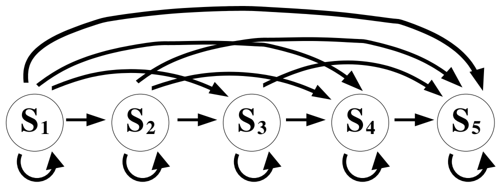


Figure 4.6: Graph of Left-Right HMM

Learning and Evaluation of the Hidden Markov Model. We fix the observation model obtained from clustering and use expectation-maximization (EM) learning to determine appropriate values for T, M_{mix} and the state prior, according to [93]. Since the observation space given by our sensor model is filled very sparsely and the covariances of the data are all considerably small, we encountered problems of overfitting. That means, observation probabilities tend to zero and cause numerical instabilities. To anticipate those problems, we add noise to the clustering data, to artificially spread the distributions.

The task to find the best HMM is to optimize the learning with regard to the number of clusters and the noise to be added. Too many clusters cause some clusters to not cover a sufficient number of samples, and too few reduce the discrimination power. On the other hand, too much noise reduces the discrimination power but increases the generality of the model. None or little noise results in over-selective HMMs. By now we semiautomatically search for an optimal solution. Figure 4.7 shows the graph of the HMM and its transition matrix that were learned for the case presented in Table 4.1. As expected we obtained a left-right model (no backward transitions), and most states are only connected with the next state.

$$\begin{tikzcd}
\mathbf{S_1} \arrow[r] \arrow[loop below] & \mathbf{S_2} \arrow[r] \arrow[loop below] \arrow[rr, bend left] & \mathbf{S_3} \arrow[r] \arrow[loop below] & \mathbf{S_4} \arrow[r] \arrow[loop below] & \mathbf{S_5} \arrow[loop below]
\end{tikzcd}$$

$$\begin{pmatrix} 0.8 & 0.2 & 0 & 0 & 0 \\ 0 & 0.58 & 0.417 & 0.003 & 0 \\ 0 & 0 & 0.37 & 0.63 & 0 \\ 0 & 0 & 0 & 0.68 & 0.32 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure 4.7: Learned HMM for case presented in Table 4.1; Also given is the transition matrix T, presenting the respective transition probabilities p_{ij}.

4.1.4 Experimental Results

In this section, we will empirically evaluate the proposed approaches. To acquire a sufficient amount of data for different hallway environments we used a simulator (RHINO Navigation Software [15]) which provides laser measurements, based on the sensor model of the real SICK LMS200 laser range finder. Also, we annotated the maps, in order to automatically label the recorded observation sequences. As a result we obtained about 200˙000 observation vectors for eight different environments, which adds up to approximately 20˙000 observation sequences (divided in training and test data). The environments differ in the amount of clutter that is present, and the width of hallways (2, 2.5 and 3 meter). The environment depicted in Figure 4.1 is referred to as *2m uncluttered*. In Figure 4.8 the same environment is depicted for the slightly and strongly cluttered case.

It can be seen from Tab. 4.2 that in some cases the recognition rate for the single observation classifier is very high, but in particular for *L/RTurn* it is rather poor. This is due to the fact that the last observation is not necessarily the best, e.g. when the robot is cutting the edge in a left or right turn. The classifier “last but one” in Tab. 4.2 works like the SOC but considers the observation before the last. It improves the classification for some classes, but for others, like *T-Junction*, it slightly degrades. This implies that it is difficult to determine which single observation should be used for SOC. Furthermore, the classification is slightly worse when clutter is present, due to ambiguous measurements. For the averaged sequence classifier (ASC) the classification is strongly dependent on the weighting function, the

Gateway Class	*X-Crossing*	*T-Junction*	*LT-Junction*	*RT-Junction*	*L-Turn*	*R-Turn*
2m uncluttered	581	390	200	198	99	106
ASC	86.8%	96.9%	98%	99.5%	14.1%	41.5%
SOC - last obs	86%	100%	95.5%	97.5%	50.5%	100%
SOC - last but one	86.8%	100%	99%	99.5%	100%	100%
2m cluttered	148	169	70	42	30	38
ASC	83.7%	39.6%	90%	61.9%	6.7%	2.6%
SOC - last obs	95.6%	93.5%	77.2%	57.2%	40%	94.7%
SOC - last but one	98.2%	90.5%	97.2%	59.5%	46.7%	97.4%

Table 4.2: Classification results for the SOC (last observation, and last but one) and the averaged sequence classifier (ASC); *DeadEnds* have 100% recognition rate for all cases.

more we rely on the closer measurements the better. Whereas the classification results are comparable to the SOC, we gain a little more robustness due to the averaging. The classification results can be improved when the weights for *Hallway/Deadend* are ignored, but this way it is difficult to evaluate ambiguous situations.

The EM learning converged to left-right HMMs with expected apriori probabilities for all types of sequences and training data from one or more environments. When we train the HMM for a single environment only, the classification rate is 100 percent for the respective test data, which shows the validity of the HMM approach. Since it is difficult to determine the optimal HMM for a certain gateway type and data from different environments, we did not yet obtain an optimal set of HMMs to handle all classes with satisfying discrimination power. But we give examples for pairwise classification in Tab. 4.3. The experiments have been performed on the same test data used for the SOC/ASC evaluation. It can be seen that the hallway width does not influence the discrimination power, but as for the SOC/ASC, the recognition rate decreases in the presence of clutter. Besides the difficulty of finding the optimal HMMs, the presented approach seems to be very promising in the context of the automatic generation of structured robot maps. The advantage is that the resulting HMM classifier provides probabilities for observation sequences with regard to the different gateways. Thus, it is possible to globally fuse the results of different observation sequences formally, using Bayes filter.

environment	2m uncluttered	2m cluttered	2.5m cluttered	3m uncluttered
lturn/rturn	100/100%	100/100%	96.3/100%	100/100%
lturn/tcrossing	100/100%	100/93.5%	98.75/93.5%	100/100%
rturn/tcrossing	100/100%	100/92.9%	100/23%	100/100%
lturn/xcrossing	100/100%	100/100%	99.4/100%	100/100%

Table 4.3: HMM based classification; given data of two gateways to the respective HMMs

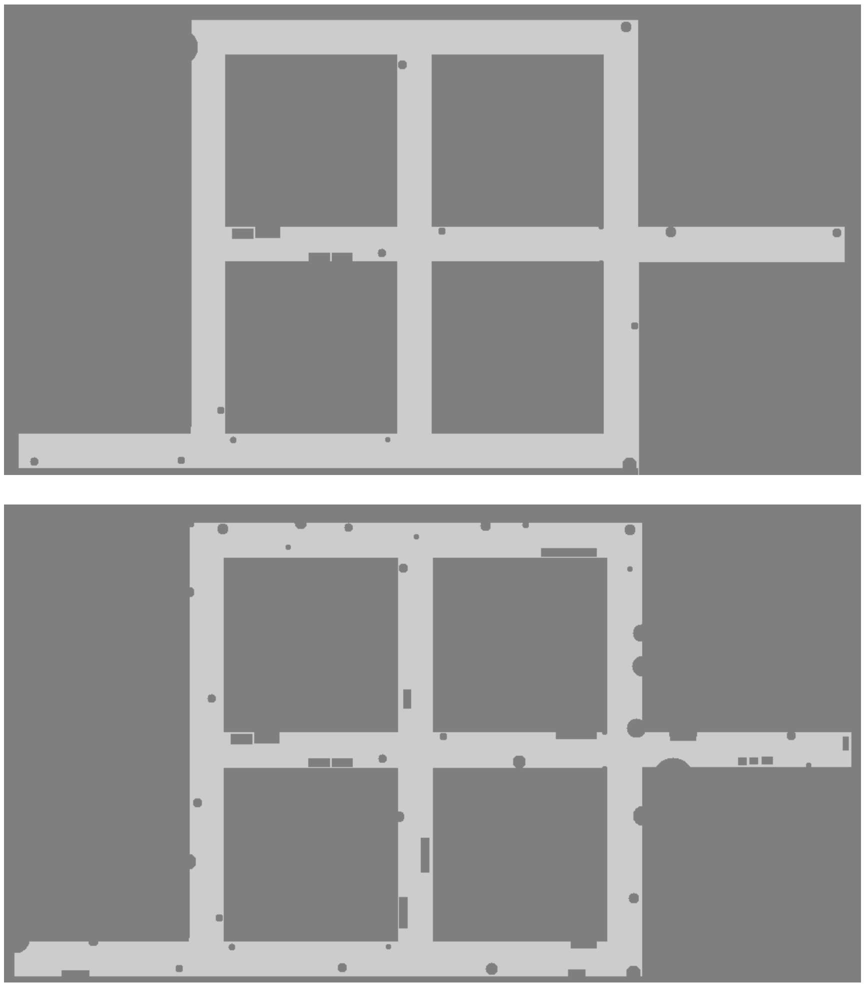


Figure 4.8: Slightly cluttered (top) and strongly cluttered (bottom) hallway environments

4.2 Detecting Doors using Laser and Vision

In Chapter 3 it has been shown how 3D rectangular objects can be efficiently and accurately reconstructed using stereo vision. In this section we are particularly interested in door frames to extend a simple laser range based detector for *Narrow Passages* to a robust method for the detection and recognition of doors. In Subsection 4.2.1 we present the detection of *Narrow Passages* based on distance measurements and discuss its limits with respect to door detection. The fusion of these detection results with observations of appropriate 3D rectangles is described in detail in Subsection 4.2.2.

4.2.1 Laser Based Detection of Narrow Passages

Figure 4.9 shows the raw laser readings as acquired by a laser range finder. In the middle of the 2D point scan we can recognize an opening, which in this case is a door. The left and right margin of the opening are marked with black circles. According to Chapter 2, those marks are called gateway points. They define the pose of the gateway. The first step is to detect those points or possible candidates within the 2D point scan:

1. Divide the field of view of the scanner into a left and right range. The middle is defined by the viewing direction α_m, hence the left and right range by $\alpha_m - \Delta\alpha < \alpha < \alpha_m$ and $\alpha_m < \alpha < \alpha_m + \Delta\alpha$, respectively.
2. Find the closest distance measurement within the left and right range, and calculate the distance D_{lr} for the respective 2D points P_{left} and P_{right}. α_{NP} is the enclosed angle, see also Figure 4.9. Due to the construction, the 2D line segment given by the two closest points has an intersection P_{is} with the line of sight of the scanner. The pose of the scanner is (P_{sc}, α_m), and $D_{NP} = ||\overline{P_{is}P_{sc}}||_2^2$ is the distance of the scanner/robot to the narrow passage.

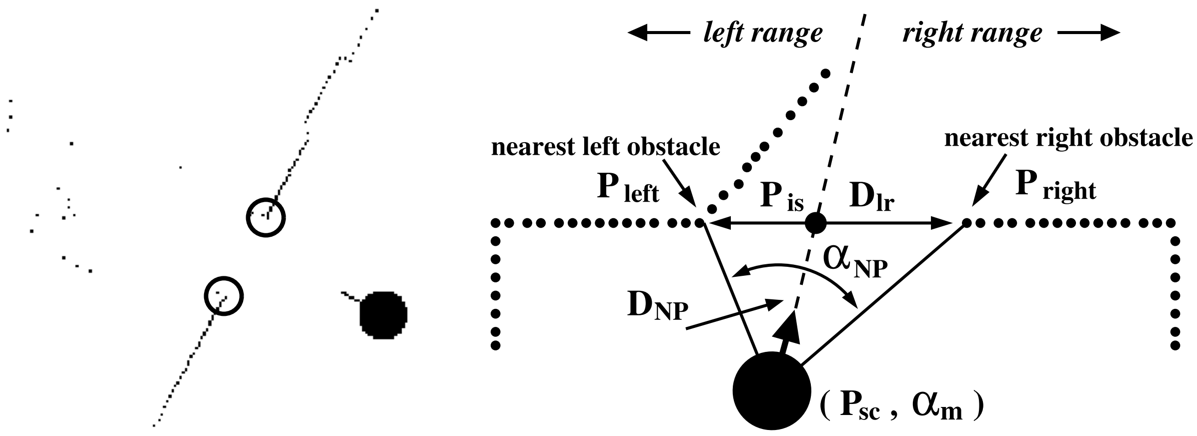


Figure 4.9: **left:** Narrow passage in a laser scan, **right:** The model used to generate hypotheses for narrow passages from laser data.

The second step is to evaluate the calculated features with respect to the robot's ability to traverse narrow passages. That means, the robot can only move through narrow openings if they are larger than a minimum distance D_{lr}^{min}. A rough estimate for $D_{lr}^{min}/2$ is given by the sum of the robot's radius plus a security distance. The actual value is determined by the collision avoidance that is applied.

1. $D_{NP} < D_{NP}^{max}$, since chances for ambiguities are high, the observation is irrelevant if the scanner/robot is further away than a given distance.
2. $D_{lr}^{min} < D_{lr}$, i.e. the width of the opening should be large enough to allow the robot to safely traverse it.

Further considerations analyze the measurements between P_{left} and P_{right} to assess if the given hypothesis offers new views after traversal. Basically, this idea is also used in frontier based exploration [120]. The question is, if there are hidden frontiers or visible openings of reasonable size behind the narrow passage. The algorithm iterates from P_{left} towards P_{right} and vice versa as long as the distance between consecutive measurements is below D_{lr}^{min} and the distance to the opposite gateway point is still large enough ($> D_{lr}^{min}$) for the robot to traverse it. This procedure either declares the considered passage as being valid or invalid, and eventually results in additional candidates for P_{left} and/or P_{right} as depicted in Figure 4.10. If the observation is declared to be invalid, it is ignored, see Figure 4.11 for examples. This way, most observations that do not seem to offer the possibility for the robot to enter a new region are neglected.

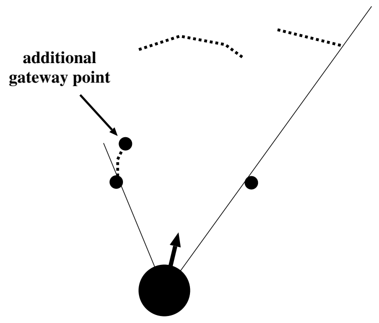


Figure 4.10: Additional gateway point that defines another hypothesis

If only laser data is used to detect narrow passages the aperture angle α_{NP} and the gateway width D_{lr} are assumed to lie in a certain range. The range is specified by minimum and maximum thresholds which are derived from the expected door width for a given environment.

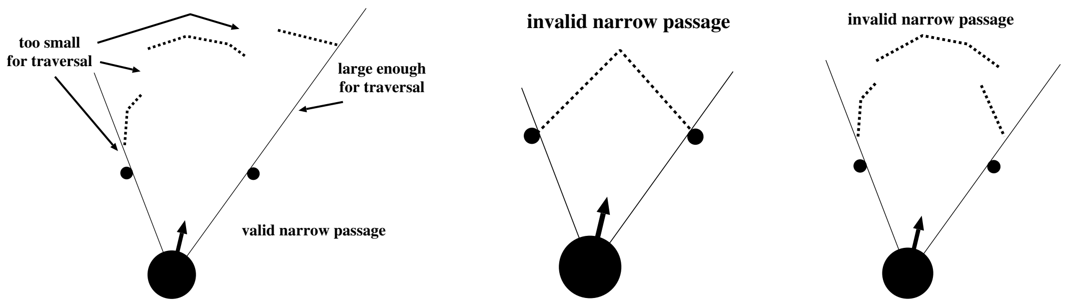


Figure 4.11: Valid and invalid hypotheses for narrow passages with respect to open space behind the gateway line

Different hypotheses for P_{left} and P_{right} lead to different pairs of gateway points, where the pair with minimum distance is chosen. Nevertheless, it is clear that using only 2D laser range data can hardly account for all possible configurations. This is partly due to to the fact that only a single laser scan is considered, but also because of the limitations of 2D perception.

4.2.2 From Narrow Passages to Doors using Stereo Vision

We have seen in the previous section that openings or narrow passages can be detected using 2D laser range data, but the methods are not suitable to differentiate a door from a passage between a table and a closet. Therefore, observations from a stereo camera rig are used to increase the reliability of that decision. That means, first 3D rectangular objects are reconstructed from the images. Then candidates, which may present door frames, are chosen and correlated with the laser observation. Appropriate rectangles are determined according to size, i.e. height and width including tolerance intervals, and orientation, i.e. the normal vector should have a very small z-component. The left and right vertical line are projected into 2D and transformed towards the position where the laser scan has been obtained, and finally compared to the 2D measurements for the gateway points in terms of Euclidean distance. If a hypothesis attained from the laser range data is supported by a 3D rectangle, the observation is considered to represent a door. The described algorithm assumes that the pose of the camera with respect to the robot coordinate system has been determined by means of external calibration. We discussed the inherent problems in the alignment of 3D objects with respect to a common frame of reference in Chapter 3.3.5.

4.3 Summary

In this chapter we presented methods to detect and classify different classes of gateways. For turns and crossing we proposed a new algorithm that uses Hidden Markov Models to classify sequences of observations. The observation models for each state are represented by Mixtures of Gaussian which are automatically determined based on k-means clustering. Compared with simple heuristic classification the new algorithm shows superior performance in both generalization and classification results. To detect doors we developed a method based on 2D laser data and the 3D reconstruction of door frames. The experimental evaluation and discussion of these methods along with our mapping system will be given in Chapter 6.

Chapter 5

TumBot Mapping & Navigation System Based on RG Maps

In this chapter we will present a module based implementation of Region & Gateway Mapping along with localization, path planning, collision avoidance and robot motion control. We believe that consequent modularity by means of parallel and encapsulated processes is an important precondition for a complex mapping and navigation system to run robustly on a real robot. In principle, any module could encounter a situation that it cannot handle, either because of a hardware failure or because of some software bug. In other words, no software can be guaranteed to run error-free for ever. But it is relatively seldom that all components get stuck at the same time. Thus, several watchdog processes can observe the performance of working modules, and restart them when necessary, without influencing modules that wait for incoming data. Any module of the TumBot Mapping & Navigation System can be restarted at any time independent of the others. And considering the worst case, i.e. some module crashes unexpectedly, it would neither harm nor influence any other module. This is an important and desirable feature, because it does not only ease the development of complex systems but also allows the system to analyze itself and properly react to error situations. Furthermore, from the software development point of view such an architecture supports simple integration/replacement of different projects/modules and concurrent development. Therefore, two requirements must be met:

1. The module based framework must be generic and transparent.
2. The interfaces for the communication between modules must be well-defined

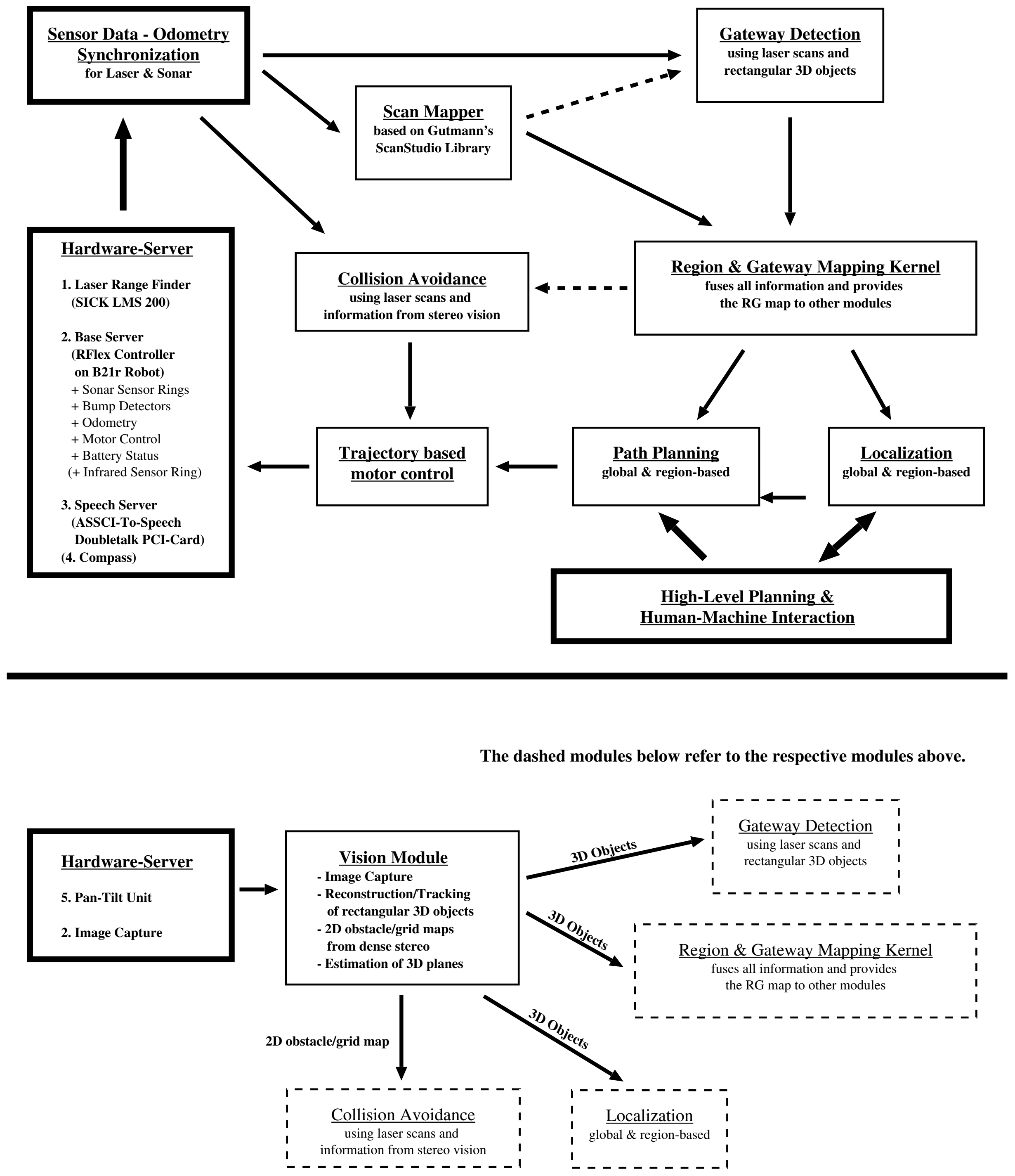
We will discuss the principles of our modular architecture in detail in Section 5.1. Currently, we run the system on a B21r mobile robot from iRobot Corporation. It is equipped with a 180 degree laser range finder (SICK LMS 220) that is facing horizontally to the front at a height of about forty centimeters. A stereo camera rig is mounted on top of the robot, comprising two parallel cameras (Leutron Vision) and a pan-tilt unit (Directed Perception

Inc.). The cameras are located approximately 125 centimeters above the ground plane. The robot features a four-wheel-synchro drive, which has small odometric errors, when employed in indoor environments. Additionally, the robot provides sonar, infrared and bumper sensors as well as a compass, power control and two Dual-Pentium III processors with a clock rate of 800 MHz. For visualization and user interaction we use a laptop that is attached on top of the robot. The platform is connected to an external network through a 54 MBit/sec wireless LAN adaptor. Due to its four batteries the robot can be operated for about five to six hours without recharge.

Figure 5.1 depicts the overview of the mapping and navigation system. The sensor data is first synchronized with odometry readings in order to properly fuse different measurements (laser, vision) within one map. The *Gateway Detection Module* uses either original or aligned laser scans to recognize gateways. Therefore, laser scan packets are fed into the *ScanMapper Module* that aligns the scans using Gutmann and Konolige's Local Registration/Global Correlation (LRGC) algorithm [44]. These scans are also utilized by the *RG Mapping Module* to generate a compact map of 2D line segments. The maps are used for localization, path planning and to calculate 2D object hypotheses (rectangle, circle) for the region description. The *High-Level-Planning Module* controls the behavior of the robot, i.e. it controls the exploration process, allows for user interaction and generates plans to fulfill subsequent tasks or parallel user requests. Based on the input from those modules, the *Path Planning Module* generates an appropriate list of intermediate and final targets that is then sent to the *Collision Avoidance Module (CAM)*. The CAM steers the robot using the trajectory based motor control.

Until now, little has been said about the lower part of Figure 5.1. It is an illustration of our vision module and its interactions. As for laser and sonar, after images have been captured, they are synchronized with odometry. Our system comprises a stand alone *Image Capture Module (ICM)* that can also be started within another module, here the *Vision Module*. The rationale behind this is to prevent the huge amount of image data to be transferred between processes. As soon as the necessary bus capacity becomes available, the ICM is started as a single process. The same holds for the different image processing tasks, as long as it is meaningful. For example, the generation of 2D obstacle maps from dense stereo has only little interaction with the 3D reconstruction, thus they could and should be run as parallel processes. For now, parallelization within the *Vision Module* is implemented on the level of threads (according to the POSIX standard).

As outlined in Chapter 2, the structuring process of RG Mapping is encapsulated in the *RG Mapping Kernel*. This module processes the results from gateway detection, scan alignment and object recognition to build and provide a consistent RG Map, i.e. accurate region and gateway descriptions as well as the RG Graph. Section 5.2 gives a more detailed description, along with the *Gateway Detection*, *Vision* and *ScanMapper Modules*. Localization and path planning based on RG Maps are described in Section 5.3 and 5.4, respectively. Two additional modules handle the low-level control of the mobile platform,



Sensor Data - Odometry Synchronization
for Laser & Sonar
Gateway Detection
using laser scans and rectangular 3D objects
Scan Mapper
based on Gutmann's ScanStudio Library
Hardware-Server
1. Laser Range Finder (SICK LMS 200)
2. Base Server (RFlex Controller on B21r Robot)
+ Sonar Sensor Rings
+ Bump Detectors
+ Odometry
+ Motor Control
+ Battery Status
(+ Infrared Sensor Ring)
3. Speech Server (ASSCI-To-Speech Doubletalk PCI-Card)
(4. Compass)
Collision Avoidance
using laser scans and information from stereo vision
Region & Gateway Mapping Kernel
fuses all information and provides the RG map to other modules
Trajectory based motor control
Path Planning
global & region-based
Localization
global & region-based
High-Level Planning & Human-Machine Interaction
The dashed modules below refer to the respective modules above.
Hardware-Server
5. Pan-Tilt Unit
2. Image Capture
Vision Module
- Image Capture
- Reconstruction/Tracking of rectangular 3D objects
- 2D obstacle/grid maps from dense stereo
- Estimation of 3D planes
3D Objects
3D Objects
3D Objects
2D obstacle/grid map
Gateway Detection
using laser scans and rectangular 3D objects
Region & Gateway Mapping Kernel
fuses all information and provides the RG map to other modules
Collision Avoidance
using laser scans and information from stereo vision
Localization
global & region-based

Figure 5.1: Overview of the TumBot Mapping & Navigation System based on Region & Gateway Maps. Depicted is the module based interaction for laser range data (top) and image-based data (bottom). The view has been divided for the sake of clarity.

namely the robot dynamics and collision avoidance. In fact, contrary to all approaches we know, our mapping and navigation system features a collision avoidance that is completely separated from motor control. This has several advantages. First, because the robot is steered by a feedback control that is activated about twenty times per second, trajectory following is very accurate, predictable and reproducible. Depending on the accuracy of the odometry, this is true even for longer distances. Second, reasoning about collision free paths is done on trajectory level, i.e. explicitly in Euclidean space that allows for smooth control of the robot and strongly reduces the possibilities of deadlocks and local minima. Third, apriori knowledge about the environment can be easily incorporated. We will present our approach to collision avoidance and motor control in more detail in Section 5.5.

5.1 The TumBot Module Architecture

There probably exist almost as many different approaches to software architecture for robot systems as there are robotic research groups. One of the first, as far as we know, mapping and navigation software used by several different groups around the world was developed at Bonn University/Germany (starting from some CMU sources). According, to the name of their robot (RHINO), we call it *Rhino Mapping & Navigation Software* [13, 110]. It features occupancy grid based mapping and localization (Markov and Monte Carlo/particle filter), path planning, 2D simulation and collision avoidance as well as modules that control the hardware, perform people tracking and many more. Parts of this software have been commercialized together with iRobot Corporation (formerly known as Real World Interfaces). The interprocess communication was based on the TCX Module Architecture, developed by Christopher Fedor [35] in 1990. Later in 1994, Simmons developed the Inter Process Communication Package (IPC), which is for example applied in the *Carnegie Mellon Robot Navigation Toolkit (CARMEN)*[1]. The ideas of TCX and ICP are based on the *Task Control Architecture (TCA)* by Reid Simmons [101][2].

The Rhino Software supports distributed processes and a central module (TCX-server) which organizes the initialization of the communication between modules. After that, the modules can exchange messages, provided that an appropriate message handler exists. However, the serious development of the Rhino Software stopped somewhere in the last years, and the CARMEN-Project can be considered a follow-up, providing modules for base and sensor control, obstacle avoidance, localization, path planning, people-tracking, and mapping. Another robotics software platform that recently became more popular is the Player/Stage project hosted by SourceForge [3]. For now, its main purpose is to provide a generic interface to the robots hardware that can be easily replaced by a 2D multi-robot (Stage) or a 3D simulator (Gazebo). Nevertheless, there already exist a couple of

[1]http://www.cs.cmu.edu/∼carmen
[2]http://www.cs.cmu.edu/afs/cs/project/TCA/www/TCA-history.html
[3]http://playerstage.sourceforge.net

modules/drivers, e.g. for collision avoidance, localization, path planning, speech synthesis and blob tracking.

All the frameworks mentioned so far (and there exist a lot more) use different interface architectures (also called communication protocols) based on TCP/IP. A whole different set of software architectures is based on the *Common Object Request Broker Architecture (CORBA)*, which is first and foremost a language and platform independent specification for interprocess communication[4]. A process only needs to know the interface to another that is described by means of the *Interface Description Language (IDL)*. The two processes can run on different operating systems and can be programmed in different languages. But no CORBA implementation itself provides a convenient module based framework. Consequently, lots of researchers implemented their ideas of such a framework like Miro [115], SmartSoft [17], DARA [31], DyKnow [52] etc., just to name a few. iRobot Corporation that has been mentioned in the beginning, changed from the TCX Architecture to a CORBA based module architecture they call Mobility. We started to use Mobility for two reasons. First, it provided hardware server for our robot, so we could operate the robot immediately. Second, it was relatively easy to understand and to use, and it was not particularly specialized on robotics. It could be used for any project that involves interprocess communication. Nevertheless, we incorporated some extensions, with focus on reusability of the implemented algorithms and transparency for someone who works with a module that he/she did not develop.

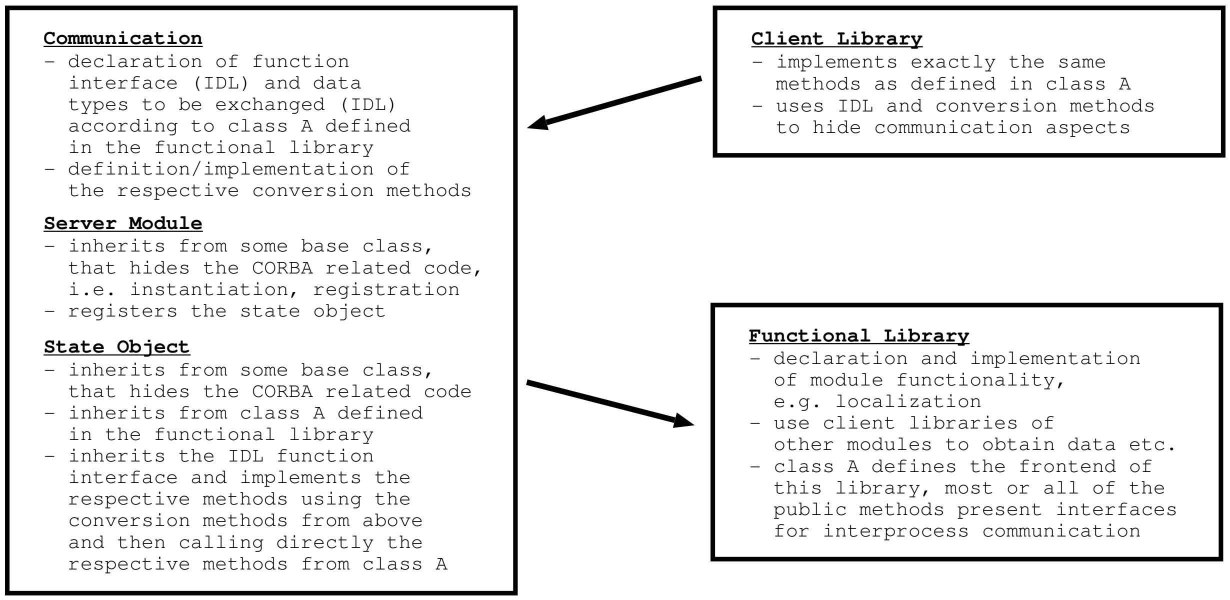

Communication
- declaration of function interface (IDL) and data types to be exchanged (IDL) according to class A defined in the functional library
- definition/implementation of the respective conversion methods
Server Module
- inherits from some base class, that hides the CORBA related code, i.e. instantiation, registration
- registers the state object
State Object
- inherits from some base class, that hides the CORBA related code
- inherits from class A defined in the functional library
- inherits the IDL function interface and implements the respective methods using the conversion methods from above and then calling directly the respective methods from class A
Client Library
- implements exactly the same methods as defined in class A
- uses IDL and conversion methods to hide communication aspects
Functional Library
- declaration and implementation of module functionality, e.g. localization
- use client libraries of other modules to obtain data etc.
- class A defines the frontend of this library, most or all of the public methods present interfaces for interprocess communication

Figure 5.2: TumBot Module Architecture, (for now) using the CORBA based Mobility framework, arrows depict dependencies

[4]For detailed information about CORBA, please visit http://www.omg.org

As a result, the functionality of all modules in the TumBot System is completely separated from the communication. That means, Mobility could be replaced by any other preferably CORBA based module framework in a reasonable amount of time. This is true even for module interconnections, because a client library (for each module) hides all communication details, i.e how to connect to another server, how to call methods using CORBA/Mobility etc. Therefore, each object, i.e. class or structure that is supposed to be exchanged between modules/processes, needs an additional definition in IDL and conversion methods that convert from one to the other representation. Even though the conversion of data types costs time, we found that on state-of-the-art computers, this time is neglectable, compared to the transparency it provides. Server and clients simply incorporate these methods and do not add any other functionality but interprocess communication to the system. Accordingly, the implementation of server modules and client libraries is fairly simple, see also Figure 5.2. We believe that it is possible to automate most of this process, but further investigation is beyond the scope of this work. The TumBot System is solely implemented in C++.

5.2 RG Mapping Modules

Several modules in the *TumBot Mapping & Navigation System* contribute to the RG Mapping process, namely the *Gateway Detection*, the *ScanMapper* and the *Vision Module* as well as the *RG Mapping Kernel*. The purpose of this section is to describe briefly the interactions between these modules and their functionality in order to built RG Maps.

As presented in Chapter 4, the *Gateway Detection Module (GDM)* utilizes distance measurements from laser range data and rectangular 3D objects to detect gateways. The detection and classification of crossing gateways (turns and junctions) was implemented and tested using Matlab, and to this date has not been fully integrated into the TumBot framework. Therefore, we concentrate on the detection of narrow passages/doors using range and vision data. The GDM continuously acquires raw laser readings and analyzes them according to Subsection 4.2.1. Additionally, it requests rectangular 3D objects from the vision module that fulfill several characteristics with respect to doors. That means, the GDM considers only 3D rectangles with a certain width and height. More importantly, the plane normal is supposed to possess a very small z-component, because the orientation of doors is usually vertical, i.e. orthogonal to the ground plane which in our case is the z-plane. The corner points of the 3D rectangle are than projected into the plane of the laser readings. A hypothesis for a narrow passage from the laser based detection is defined by the two gateway points. If the distance between those points and the projected corner points from the rectangle falls below a given threshold, the hypothesis is assumed to present a door. The GDM stacks this information in a buffer that is processed by the *RG Mapping Kernel*. Thus, the GDM provides a single interface to access the buffer of gateway hypotheses.

The *ScanMapper Module (SMM)* is based on the ScanStudio library from Steffen Gutmann. Because the scan matching results are fused with odometry using a Kalman filter, certain dynamic constraints have to be fulfilled. These constraints are defined by the dynamic model that the Kalman filter applies. In practice that means, scans are expected to be acquired at certain distance or angle intervals. Therefore, the SMM buffers the incoming laser scans and filters them according to the before mentioned restrictions. These scans are than aligned using Gutmann's scan matching. The SMM provides full access to the resulting scans that are corrected with respect to their position to compensate odometric errors. Apart from the range readings, the scans comprise a unique ID and the original odometry pose. This way they can be related to scans that have been processed by the GDM. The interface of the SMM supports the request of single scans or a range of scans by means of their IDs.

In Chapter 3 we described the generation of models for task-relevant 2D/3D objects. One such task in RG Mapping is the detection of doors or door-like gateways. Therefore, the *Vision Module (VM)* captures images from the framegrabber, reconstructs rectangular 3D objects and provides the results in an output buffer. The VM supports an interface to request this information. Furthermore, the VM generates 2D obstacle maps from dense stereo and estimates 3D planes. The former is utilized in collision avoidance, i.e. it is fused with observations from the laser range finder. Since 2D laser range data naturally misses a number of obstacles like table tops, staircases etc., we use vision to complete the perception of the environment, in order to increase the robustness of collision avoidance. 3D plane estimates are integrated into the region description. They are also used to check the plausibility of reconstructed rectangular 3D objects. Figure 5.3 depicts the integration of the VM into the TumBot system. For reasons of clarity, the figure focuses on the vision part and the interaction of those modules with other TumBot components. As outlined in the introduction of this chapter, parallelization within the VM is realized on thread level. The only reason to do so is the performance gain, due to the reduced amount of data that must be transfered between the different vision tasks.

Finally, the instance where all the provided information is interpreted and merged is the *Region & Gateway Mapping Kernel (RGMK)*. It continuously requests new scans from the *ScanMapper Module* and 3D objects from the *Vision Module* in order to generate a compact description for the current region. This preliminary region map is used by localization, path planning and high-level planning, e.g. to create an exploration strategy. Additionally, in a parallel execution procedure, the RGMK waits for new information from the *Gateway Detection Module*, i.e. gateway hypotheses. If a new gateway has been detected and traversed, the RGMK finalizes the interpretation of the previous region data, i.e. the respective range of scans and the list of 3D objects. It builds a compact map of 2D line segments from the laser readings and aligns the 3D objects with respect to this map. In addition, it calculates a bounding rectangle for the scans belonging to that region, whereas only measurements inside the region are considered. That means, laser beams that cross a gateway are filtered out. The freespace map is a binary occupancy grid map, where

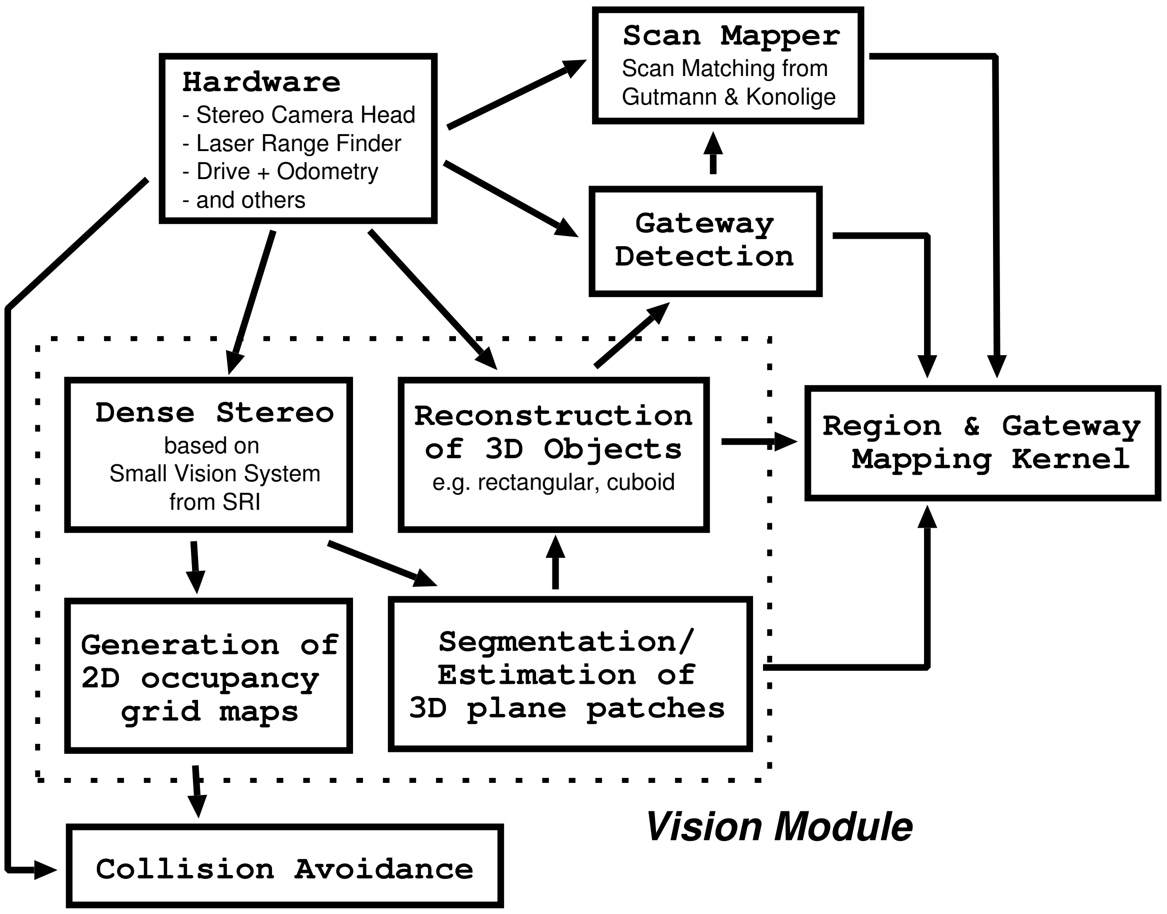

Scan Mapper
Scan Matching from Gutmann & Konolige
Hardware
- Stereo Camera Head
- Laser Range Finder
- Drive + Odometry
- and others
Gateway Detection
Dense Stereo
based on Small Vision System from SRI
Reconstruction of 3D Objects
e.g. rectangular, cuboid
Region & Gateway Mapping Kernel
Generation of 2D occupancy grid maps
Segmentation/ Estimation of 3D plane patches
Vision Module
Collision Avoidance

Figure 5.3: Vision components within the TumBot framework

for each cell only one bit is stored, indicating whether the cell is considered to be free or occupied. This should not be confused with a probabilistic occupancy grid map, where each cell is holds a probability value for its occupancy. In fact, a freespace map for a 30 square meter region with a resolution of 15 centimeter in x- and y-direction requires only about 200 byte. For comparison, a probabilistic grid map that uses floating point values for the probabilities, would need more than 5300 bytes. We use the freespace map in the region based localization to ensure that new particles are only generated for plausible locations, i.e. places which are not occupied. Furthermore, it allows to quickly check if a given particle lies in the current region.

Generating 2D line segment maps from laser data. Laser range scans acquired in a single region are considered for the region's 2D line segment map. First, they are matched consecutively with respect to each other by applying the ScanStudio-Library[5] from Steffen Gutmann, for details refer to Gutmann's work [42, 44]. After each new scan has been successfully matched, the 2D line segment map is updated. Therefore, the scans

[5]The software was used with kind permission of Steffen Gutmann.

are converted to line scans and then the line segments are merged based on position and angle. As a result the region keeps a compact 2D line segment map, which is used for localization. Considering Figure 5.4, for a fully explored region, described by 30 scans each containing about 150 to 180 measurements, the data adds up to about 4500 points. Most of those points refer to the same features in the environment, since the scans do strongly overlap. The respective 2D line segment map comprises roughly 40 line segments, which corresponds to 80 points. Thus, we have a compression factor of about 50. Even though the line segments represent an abstraction of the points, and in some cases may not exactly approximate the laser readings, experiments showed that the maps hold enough information for robust localization [97]. Note that a 2D line segment map contains lines that are outside a region's bounding box. This is because these lines can be seen through the gateways, and they are necessary for robot self-localization.

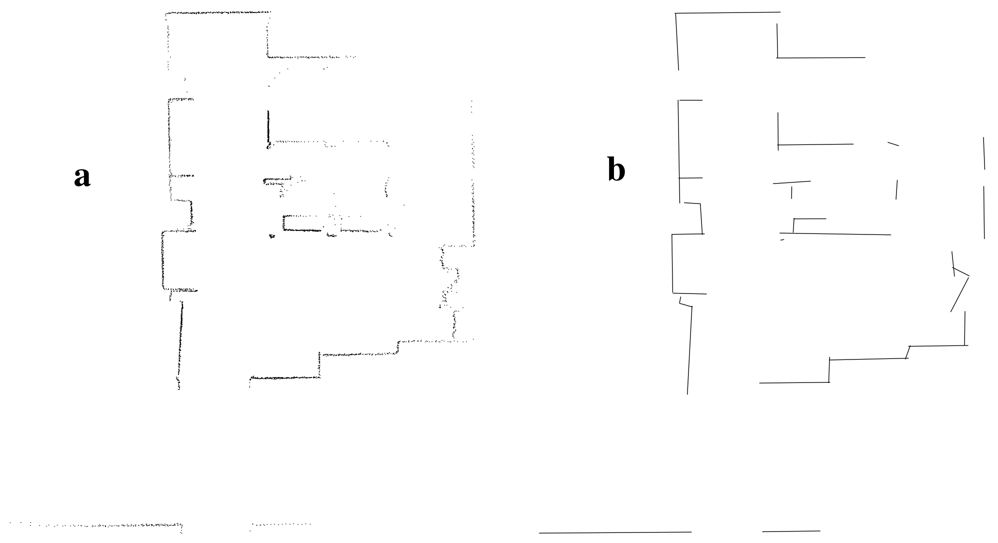


Figure 5.4: Generating 2D line segments maps from laser scan data, a - 5042 scan points from 30 scans, b - after line merging, 38 line segments

5.3 Localization with RG Maps

Given a 2D or 3D metric description of the environment and sensor data that is reflected in the representation, a mobile system can estimate its position with respect to the given map. This process is called localization and it has been thoroughly studied over the last decades. For the two-dimensional case based on an accurate global and static map and range data this problem is more or less solved [38, 40, 43]. Even for 3D representations there exist a number of approaches [64], though limited processing power restricts their wide application. Another problem in the context of camera based localization, is to deal with changing illumination. And a more general and unsolved question is, how can accurate, complete, metric 3D models be acquired and updated.

The region based localization in the TumBot system is mainly based on 2D maps and laser range data using a particle filter. For details about this localization method the interested reader may refer to [40]. In this section we want to describe the differences of localization within the context of *Region & Gateway Maps*. Let us first assume that the region in which the robot is located is known. The *Localization Module (LM)* requests the data for this region from the *RG Mapping Kernel*, including a list of adjacent gateways. The LM also requests and stacks the data for the neighboring regions, if they exist. The motivation is that when the robot approaches the region boundary, particles beyond gateways are projected into the respective known neighboring regions and evaluated there. Thus, the evaluation is always based on the data of the correct region (if it already exists). Also, this procedure gives additional proof whether a gateway has been traversed or not, because in each step the localization estimates the most probable position within the particle cloud and considers this to be the real pose. For now we simply consider the particle with maximum probability to represent the current pose. If this estimated pose lies at a reasonable distance behind the gateway, with respect to the current region, the localization assumes that the gateway has been passed.

For global localization within RG Maps it is necessary to be able to recognize regions. Different features from the region description like size, number of connected gateways, binary gridmap for freespace etc. have to be considered in order to solve this problem. In addition, an action history can be utilized to not consider all possible regions but only the most likely according to the path the robot has taken. However, in the current version of the presented mapping system global localization is not implemented.

5.4 Path Planning with RG Maps

Path planning within the *TumBot Mapping and Navigation System* is divided into two parts, i.e. global and region-based planning. The former is based on the *Region & Gateway Graph (RG Graph)*. It applies A* search to find the optimal path according to the costs along the edges. The result is a list of crossing points, which in pairs belong to either the same region or the same gateway. As discussed in Section 2.1.3, the edges in the RG Graph refer to traversals through gateways or regions. The cost for an edge is initialized with the Euclidean distance of the adjacent crossing points (nodes). During navigation the robot measures the real distances and time and changes the costs accordingly. Additionally, traversability over time is a very useful property for gateways, in particular for doors, because they can be open or closed on repeated observations. For now, we apply a simple but efficient mechanism, that is, we decrease or increase the cost of a door-edge by adding a fixed, possibly negative, length to the Euclidean distance, whenever the door could or could not be traversed, respectively. The lower bound for the costs is given by the real distance between the adjacent crossing points. When incorporating hallway gateways, this lower bound for doors can be increased in general, so that the resulting path prefers hallway crossings and turns to doors. Technically, the search algorithm determines the "shortest" path, because it only considers distances. But, due to the manipulation of those distances, the result is not necessarily the shortest path with respect to the real environment. It rather presents a path with high probability for success and moderate navigation complexity. The latter also implies, that even though the resulting path might not be the shortest in the Euclidean sense, it can still be traversed considerably fast. That means, the path is optimized with respect to feasibility and travel time. Due to the inherent heuristics of this approach, it is not possible to compare this solution to a global optimum, whereas it is also not clear how to determine that global optimum. The approach always returns a path that is consistent with the map representation and valid in the sense that it has been traversed before, an example is depicted in Figure 5.5.

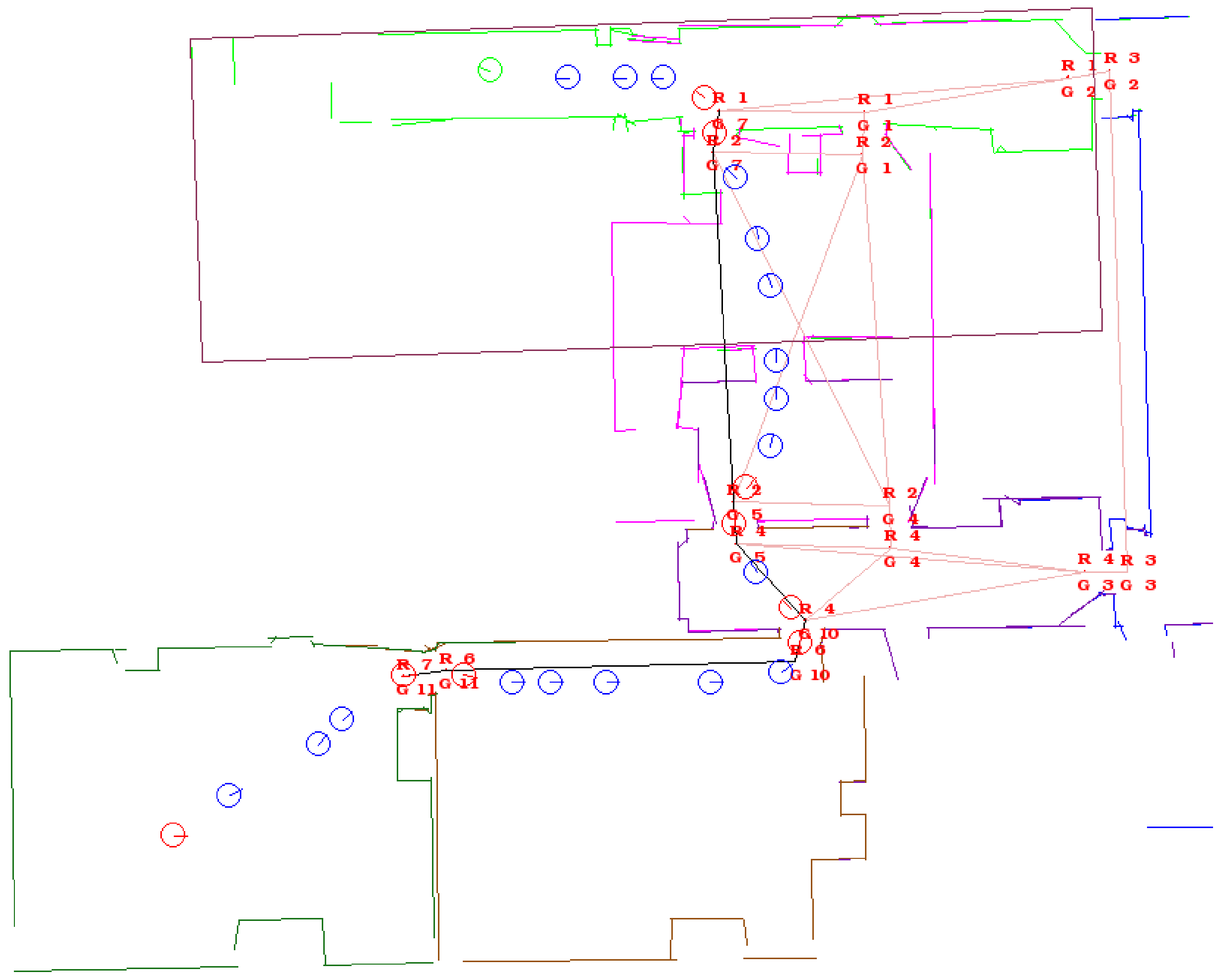


Figure 5.5: Example for global path planning based on the Region & Gateway Graph

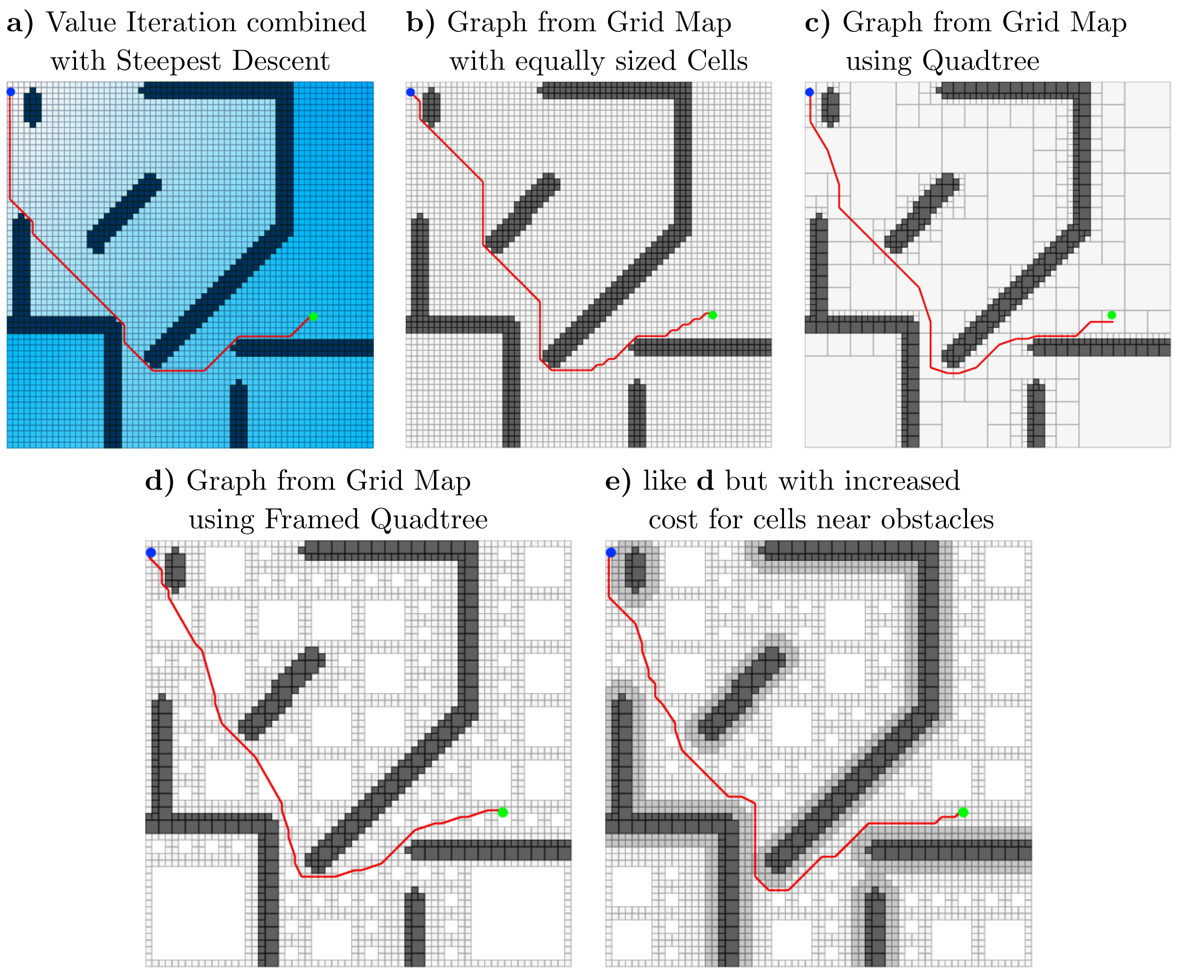

a) Value Iteration combined with Steepest Descent
b) Graph from Grid Map with equally sized Cells
c) Graph from Grid Map using Quadtree
d) Graph from Grid Map using Framed Quadtree
e) like d but with increased cost for cells near obstacles

Figure 5.6: Comparison of different methods for metric path planning. The path search in **b** to **e** is performed using A* search in the generated graph. In our system we decided to apply the approach shown in **e**.

The task of the region based path planning is to consider the metric representation of the environment, i.e. of a region. We apply a graph based approach, where the graph is built from a grid representation of the region data, namely the map of 2D line segments. The line segments are first rendered into a traditional occupancy grid using the 2D Bresenham algorithm [11]. Then a quadtree structure is extracted from the grid [88], which strongly diminishes the necessary number of grid cells, and thus, the size of the graph. The resulting path seemed to be suboptimal, when cells of small resolution adjoin large quads. This is because the nodes represent the center of cells. That means, if the quad is large, it causes the distance to its center to be large as well, and thus, if the shortest path would only lead through a part of that quad the distance will still be based on the distance to its center. Figure 5.7 illustrates this deficiency. Therefore, we added a frame of the smallest resolution to each quad. This representation is known as framed quadtrees [106, 21], and it allows to traverse large quads in all directions, compare Figure 5.6. A final optimization step smoothes the path. First, the start and target positions are adapted, because they do not necessarily lie in the center of a cell. Second, intermediate path points are eliminated, if they would lead to a longer path and if the alternative path is collision free, see Figure 5.8 (left). Due to the graph representation, it is easy to consider navigation constraints, e.g. the robot should avoid moving too closely along walls. It is integrated by means of increased costs for the respective nodes. As a result, the generated path "stays" away from walls and obstacles, i.e. it exploits freespace where possible as can be seen in Figure 5.6. Additionally, the presented approach naturally favors the traversal of wider passages towards narrow ones, see Figure 5.8. For a detailed comparison an evaluation of the investigated methods see also [91].

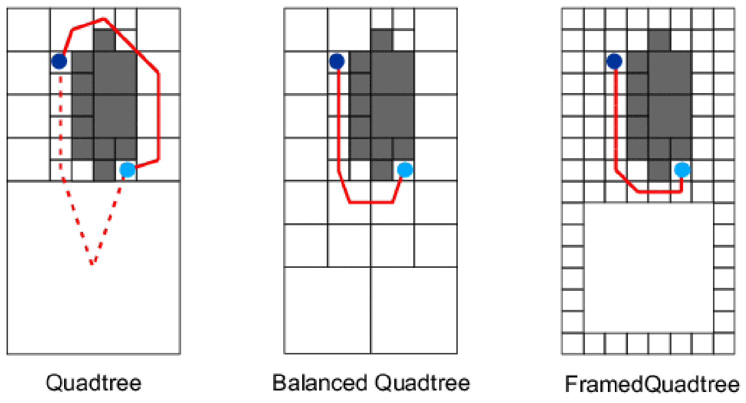


Figure 5.7: **left:** If the Euclidean shortest path would traverse only small parts of a large quad, this path can not be found in the graph. This is because nodes in the graph are represented by the cell center and thus the ideal shortest path is not contained in the graph. **middle:** Constraining the difference in size for adjacent cells (balanced quadtrees) improves the situation presented here, but this can not be guaranteed for all cases. **right:** Generating the search graph from a framed quadtree representation always returns the correct shortest path.

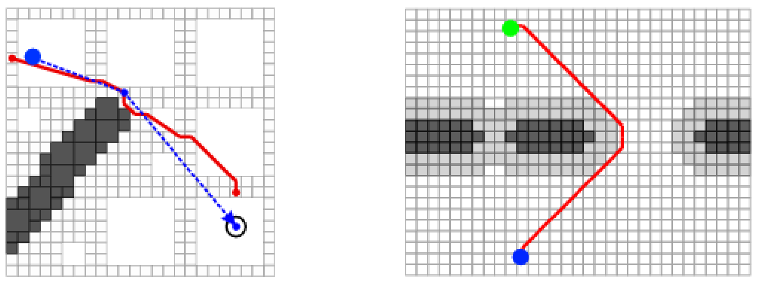


Figure 5.8: **left:** Path optimization **right:** Increasing the cost of cells near obstacles leads to paths that "stay" away from obstacles and prefer wider passages to narrow ones.

5.5 Collision Avoidance & Robot Motion Control

Given intermediate target poses and a final destination pose from a path planning module the task of collision avoidance is to navigate collision-free along the given path. This includes the reasoning about obstacle free space as well as the generation of motor commands, i.e. in most cases translational and rotational velocities. Existing approaches for collision avoidance consider different sensors like bumpers, sonar, infrared, laser range finders and vision. They also differ in the way they represent the environment, e.g. Vector Field Histogram [9, 114], Velocity Space [37], Nearness Diagram [81, 82] etc. But from our viewpoint the main difference lies in the strategy which is applied to generate appropriate motor commands, i.e. whether it is locally-reactive or foresightedly-planning. In the next subsections we will explain these two strategies and give references to respective approaches. In Subsection 5.5.3 we briefly present our new approach to collision avoidance.

5.5.1 Locally-Reactive Collision Avoidance

Reactive approaches consider the current dynamic state, the general direction of the next intermediate target and the sensor data to generate the immediate next motor command. The dynamic state is generally described by a position and an orientation of the robot as well as its current velocities. That means, reactive approaches "plan" only one step ahead and they integrate the reasoning about obstacle free space and control considerations within one iteration. The goal is to move the robot approximately in the direction of the next intermediate target without causing collisions. As the name implies these approaches react reasonably fast to changes in dynamic environments and can result in smooth driving behaviour. But mostly, several parameters have to be adjusted, which are not always intuitive and often differ for different mobile platforms. Also, reactive strategies run the risk of getting trapped in local minima and their behaviour is neither predictable nor reproducible. A detailed discussion of existing methods is beyond the scope of this work. Common approaches include the *Potential field methods* [8], *Interpreted Polar Scan* [9, 114, 81, 82] and the *Dynamic Window Approach* [37, 39, 102, 12]. The latter has been validated in impressing experiments, e.g. on the mobile robots RHINO and Minerva [13, 14, 108]. The *Nearness Diagram Method* [82] offers a rule-based and simple to implement alternative.

5.5.2 Planning Collision Avoidance

On the other hand, several approaches generate not only the next motor command but plan forward in time to the next intermediate target pose or further. This results in a sequence of future dynamic states that specify the navigation of the robot. The motivation is that for example service robots often operate in slowly changing environments. That means, movable objects like chairs, boxes or handcarts may change their position but they are usually fixed when a mobile robot passes them. The problem is that these changes are not always represented in the environment map and thus the planned path may intersect with obstacles. While little has been written about the feasibility of such approaches in highly dynamic environments involving groups of human, foresightedly-planning strategies for collision avoidance offer several advantages. The behaviour of the platform is to a certain extent predictable and reproducible, and local minima can be actively avoided.

Latombe distinguishes different path planning algorithms according to their environment representation, i.e. roadmap, potential field and cellular decomposition [71] and suggests continuous fast replanning. Because repeatedly generating the complete path, while the robot moves, implies a huge amount of redundant calculation, the idea is to construct the plan only once and then reactively adapt it during traversal. This might be problematic if an obstacle moves slowly towards the path, and thereby "pushes" it away until the free corridor, e.g. between the obstacle and the wall, is completely blocked. In this case, the obstacle should be passed on the opposite side, which means that a new plan should be generated. Choosing between adaption and replanning seems to be difficult. An example for a "plan once and then adapt" approach is the *Elastic Bands Method* [92], and Konolige [60] presented a new fast approach for the "always replan"-strategy. Another interesting approach is to search in the full state space for an optimal trajectory [103]. To achieve real-time feasibility the resulting huge search space is expanded only in parts and an efficient A^* heuristic search accelerates the computation of the path. Even though the solution has been shown to be reasonably close to the full search and is optimal with regard to time, the approach relies on a global map and robust localization. Thus, inaccuracies in localization could induce suboptimal or wrong trajectories, and the applicability during exploration even in static environments is difficult.

5.5.3 Trajectory-Based Collision Avoidance

To robustly navigate our mobile robot and to better exploit the features of RG Maps we developed a novel approach to collision avoidance that falls into the category of foresightedly-planning strategies. It abstracts from the dynamic states by means of simple trajectory segments and completely separates the reasoning from the control task. That means, we apply a Lyapunov feedback controller [54] that reliably follows given trajectories by controlling translational and rotational velocities about twenty times per second. The respective module is called *Low-Level-Drive (LLD)*. Extensive experiments showed that the LLD guarantees very precise and reproducible traversal of given trajectories. To simplify the trajectory based collision avoidance we implemented a library of geometric primitives like conic, elliptic and straight segments, arcs of circles, clothoids and some combinations of those. If possible, these trajectory segments are directly determined given start and end poses. They are geometrically described by a few parameters which prevents the storage and transmission of large amounts of data. Note that the representation is irrelevant to the *Collision Avoidance Module (CAM)*. Applicability of certain trajectory segments is determined from the changes of the robots orientation and velocities.

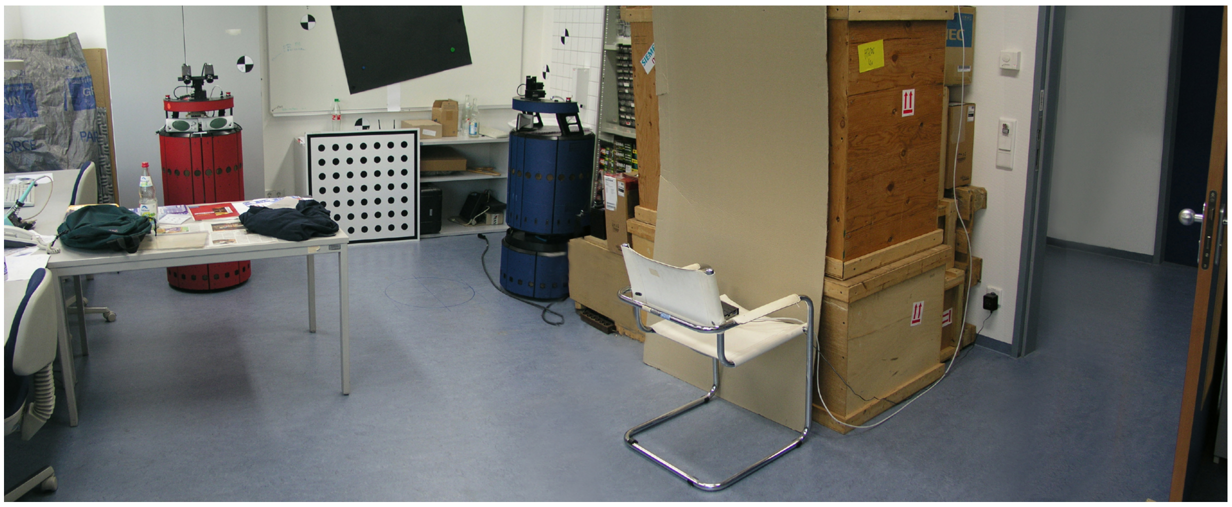
The task of the CAM is to generate consecutive trajectory segments given the target poses from the path planning, the current dynamic state, constraints on the robots dynamics and the sensor data. The distance between intermediate target poses lies between 0.5 and 2 meters. The resulting trajectories are send to the LLD and its execution is continuously observed. If a collision is detected the respective trajectory segments are replanned in time. Even though the actual implementation level is rather prototypical the experimental results are very promising. First, the prediction for the time needed to traverse a chain of trajectory segments is very accurate. Assuming the environment is correctly represented this optimal time estimate can be calculated very fast for the overall path. Hence, it can be used to evaluate different possible paths or even transportation tasks on a higher level. Second, since the pursuing of trajectory segments is very precise, the length of the overall path can also be predicted and used for path evaluation. In particular, this feature supports very accurate navigation, e.g. for docking maneuvers or exploration based on best viewpoint algorithm [25, 80, 95, 58]. Finally, the presented approach to collision avoidance provides accurate reproducibility, which is very useful for real world experiments.

Additionally we applied stereo vision to augment the laser based perception of the CAM. In order to efficiently calculate depth information from two images the cameras have been calibrated. Calibration and stereo correlation are mainly based on commercially available software packages, i.e. Halcon [87, 70] from MvTec[6] and the Small Vision System [59, 56] distributed by Videre Design[7]. As a result we obtain a 3D point cloud in camera coordinates. External calibration allows to transform those points into the robot coordinate system. The points are then rendered into a 3D occupancy grid map to explicitly model free space and filter outliers. The observations from stereo and laser are merged only in the respective grid layer, i.e. at the height of the laser. Thereby sensor fusion is performed per grid cell using a mechanism similar to the Kalman filter. That is, the occupancy value of the grid is assumed to present the mean value and the variance refers to the reliability of the different sensors. The fusion per grid cell is a weighted mean according to the sensors reliability. After this step the 3D grid is projected into 2D along the columns by multiplying the respective occupancy probabilities. The resulting grid map is used to reason about collision-free trajectories. Figure 5.9 depicts a scenario where two obstacles are in the robots path which are not properly represented in the laser measurements, i.e. a table and a chair. Shown is an image of the environment and the traveled path. In Figure 5.10 the stereo data and the respective occupancy grid maps show the improved representation. The images beneath the disparity images refer to the gridmaps from the laser range finder, simple 2D projection of 3D Data, projection of 3D Data using 3D occupancy gridmaps and the result of the sensor fusion (from left to right). It can be seen that free space is properly

[6]http://www.mvtec.com

[7]http://www.videredesign.com/products.htm

modeled using 3D occupancy gridmaps and the resulting map accurately represents the objects not visible to the laser.



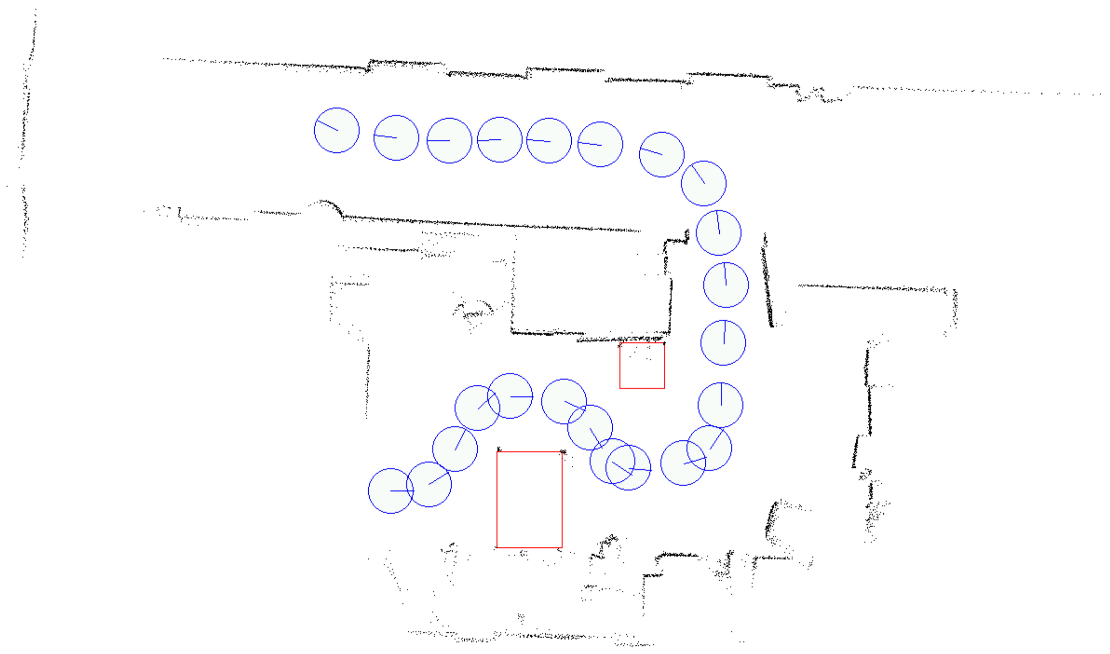


Figure 5.9: Collision avoidance scenario in our robot laboratory. The lower image presents the traveled path from topview along with the odometry corrected laser measurements. The red rectangles depict the objects that are not visible to the laser range finder.

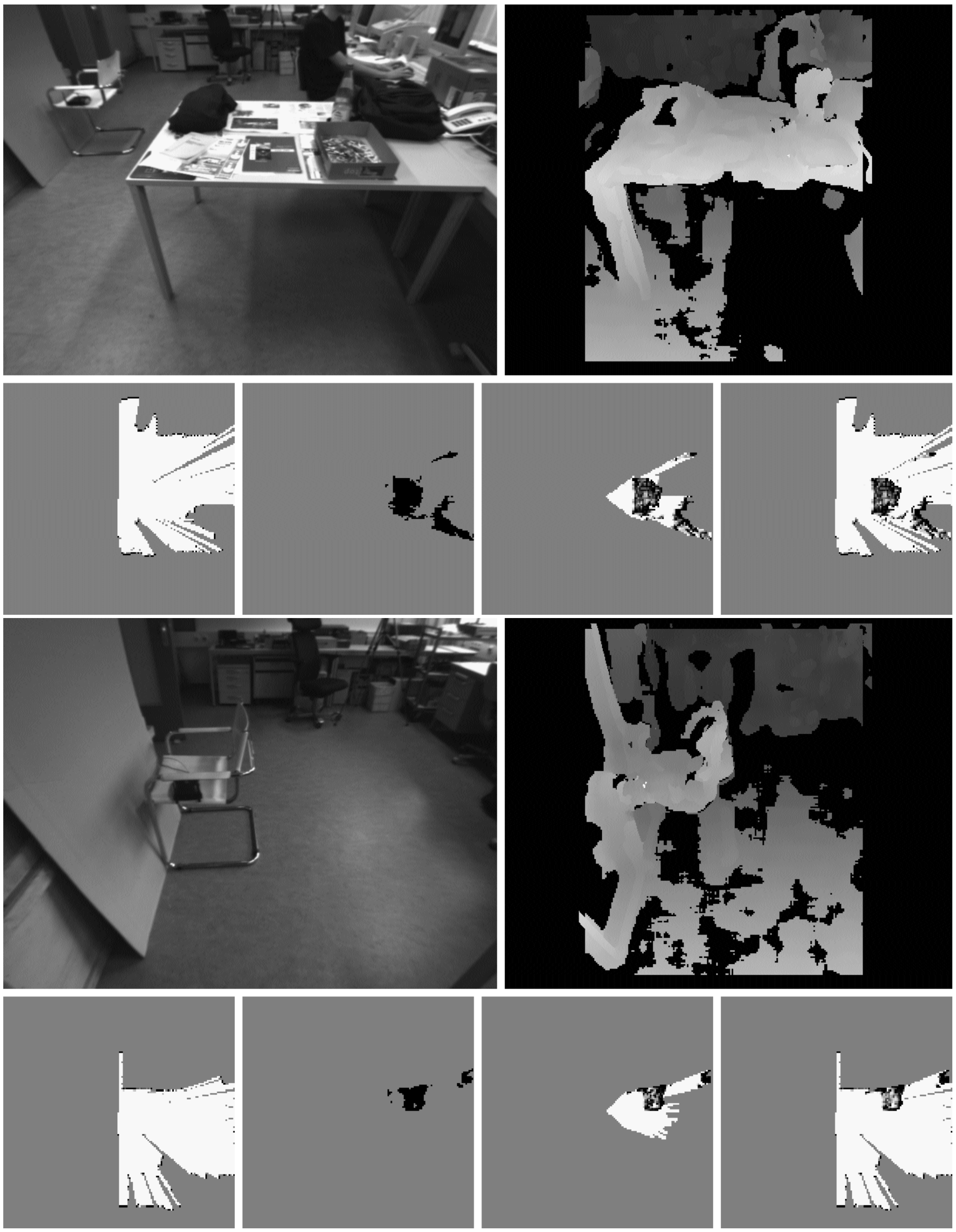


Figure 5.10: image and 3D data for the table (top) and chair (bottom) along with gridmaps from laser, simple projection, projection using 3D gridmaps and the resulting map.

Chapter 6

Experimental Results

In this chapter we evaluate the *TumBot Mapping & Navigation System's* applicability to indoor environments. The following sections illustrate that **R***egion &* **G***ateway Mapping* is both well suited and capable of acquiring structured and object-oriented representations of large indoor environments. Due to the incremental SLAM inherent in the map building process, the *TumBot Mapping & Navigation System* generates very accurate metric region descriptions. It also appropriately closes loops in the environment and adds 3D object hypotheses during map building.

The *TumBot Mapping & Navigation System* has been tested using our B21r robot but also a variety of data sets that stem from different robots and environments. Whenever possible we will give explicit references as to who provided the data, where it has been recorded, and what kind of robot has been used. However, most of those data sets were obtained from *Radish: The Robotics Data Set Repository*[1] and the website of Cyrill Stachniss[2]. Assuming a navigational speed of 0.3 to 0.6 meter/second, all processing is carried out in real-time. With respect to the data sets, that means, maps could have been built while the robots move around. In fact, in all experiments we run the whole mapping part of the system as described in the previous chapters only that acquiring data from a laser scanner is replaced by reading it from a file.

In Section 6.1 we evaluate the structuring capabilities of our mapping system when it uses only 2D laser range data. We discuss different scenarios with emphasis on the consistency of the generated maps and the performance of gateway detection. An experimental evaluation of the vision part of our mapping system in conjunction with laser-based odometry correction is given in Section 6.2.

[1]http://radish.sourceforge.net/index.php

[2]http://www.informatik.uni-freiburg.de/∼stachnis/research/rbpfmapper

6.1 RG Mapping based on Laser Range Data

It has been described in Chapter 4 and 5 that the presented mapping and navigation system uses laser range data as its primary source to generate structured maps. In addition, vision is utilized to reconstruct 3D objects within regions and to robustly detect doors. In this section, we evaluate the system's structuring capabilities when only laser data is provided. The following aspects will be addressed:

- Structural consistency of Region & Gateway Maps
- Detection performance with respect to doors, i.e. does the system detect all doors and how many of the detected gateways do probably not correspond to doors

With consistency of RG maps we first of all mean that the generated description complies to the definition of regions and gateways as given in Chapter 2. That is, regions are contiguous areas which are connected by gateways and the relationship is uniquely defined. Obviously, consistency depends strongly on the observation of gateways not as much in which order but whether or not they are detected in the first place and than visited again. We will discuss this in more detail in Subsection 6.1.1.

As mentioned before, the pure metric descriptions that were generated are very accurate and geometrically consistent due to the underlying laser-based incremental map building (SLAM). However, for some data sets the deployed SLAM method was not able to built consistent maps. A discussion of the reasons would mean to analyze the metric SLAM problem itself which is beyond the scope of this work. Several other approaches, including FastSLAM, have been proposed that overcome the limitations of the SLAM method we applied. A brief overview has been given in Chapter 2.3.

Because of the limitations of 2D perception, it can be expected that "false" gateways, in particular narrow passages between tables and closets, will appear in the maps. A quantitative evaluation will be presented in Subsection 6.1.3.

6.1.1 Structural consistency of Region & Gateway Maps

To better illustrate which situations must be considered to avoid the generation of inconsistent RG Maps we will discuss three simple example scenarios along with the experiments based on real world data. The example environments comprise a circular hallway which is divided by different numbers of gateways. The first contains only one gateway and after passing it the representation consists of two regions (R1,R2) which are connected by the given gateway (G1), Figure 6.1 (left). Assuming the robot keeps going it will at some point in the future detect this inconsistency. That is, it arrives at the gateway and determines that it should be in R1 in contrast to its belief of being in R2, Figure 6.1 (middle). The conclusion here is that R1 and R2 represent the same region, thus they are merged. The final representation is depicted in Figure 6.1 (right), and will not change further over time.

Depending on the direction in which the robot follows the circle, i.e. clockwise or counter-clockwise, the resulting single region could also be R2, but obviously this does not change the structural description. We refer to gateways, which are connected to the same region on both sides, as being "neutralized", because they do not contribute to the navigation behaviour. However, they can still be used for localization.

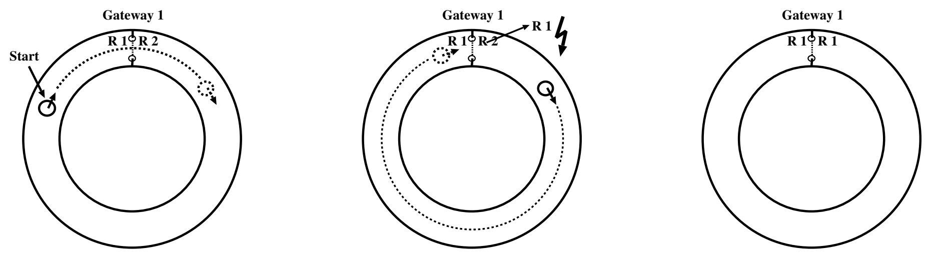


Figure 6.1: Circular hallway with one gateway

Figure 6.2 illustrates that the number of gateways within the circular hallway has no impact on the described mechanism as long as they are detected in the first place. The initial inconsistency is solved upon observing another known gateway within the same region. A real-world example is shown in Figure 6.3. Gateways are depicted by cyan lines. Neutralized gateways are not annotated as can be seen in the hallway on the left side. This particular gateway has been detected on the first traversal of the hallways. During the second loop the inconsistency was automatically detected and resolved. After this the robot enters different regions upon detecting the respective gateways and structures the environment representation accordingly. The separate 2D line segment maps for each region are shown in different colors.

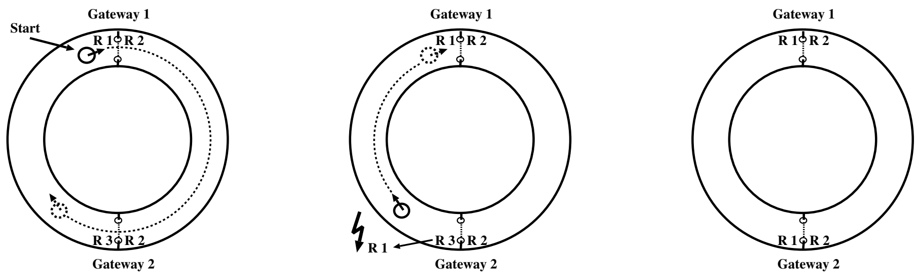


Figure 6.2: Circular hallway with two gateways

First the robot entered region R3 through gateway G4 and left through gateway G5. The mapping system consequently assumed that a new region was entered when traversing G5. Thus R3 was connected with R1 and R4. Later in exploration it became apparent that R1 and R4 describe the same region and thus they have been merged, leaving the correct description as depicted in Figure 6.3. Adjacent to the hallway in the lower half of the map are two gateways G9 and G10 leading into two different regions R8 and R9. From the map it seems that these regions are actually identical but the robot did not fully explore

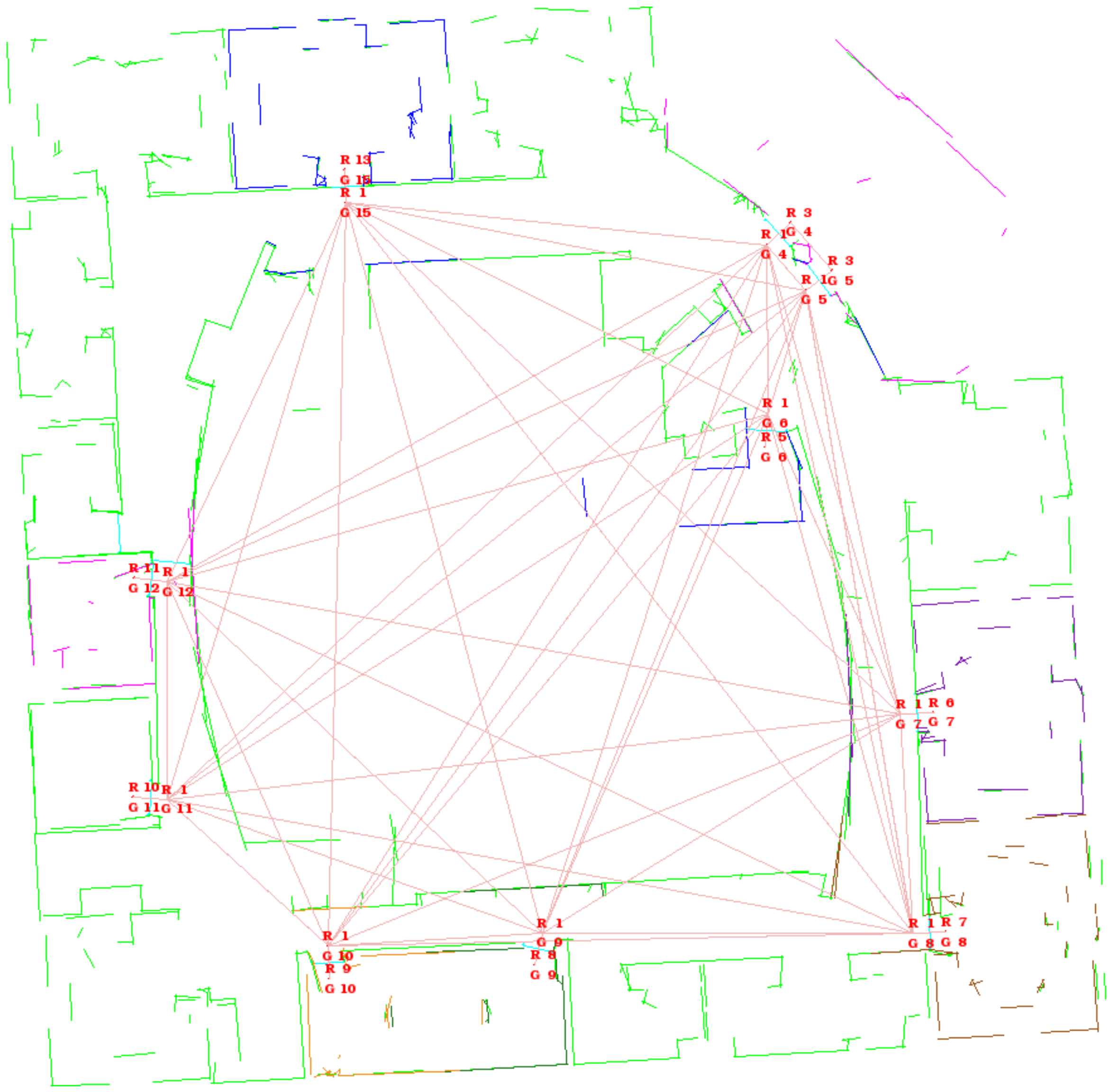


Figure 6.3: Interior of the Intel Research Lab in Seattle. The robot did not yet enter the part in the middle. Thanks go to Dirk Hähnel and Dieter Fox for providing this data set.

either of those regions, hence it could not detect this connection. Nevertheless, from the structural point of view the map is consistent given the observations and can be used for navigation. For example, if the robot was to travel from a position in R8 to a position in R9 it would move through G9, then along the hallway belonging to R1, and then into R9 through G10. Given the fact that these two regions are the same, e.g. by human-machine interaction, the mapping system could easily merge their representation and represent R8 and R9 by a single region.

We have discussed the rationale for the neutralized gateway within the hallway on the left. Close to this there is another that leads into a region that is now part of region R1. Here the gateway has not been detected when the robot first entered, but when it left. Considering Figure 6.4 the problem becomes more transparent. The example comprises three gateways one of which is not detected upon first traversal (left column). The three rows of Figure 6.4 show different scenarios where this gateway is later detected and initially leads to a false connection. Upon detection of another of the known gateways this temporary inconsistency is resolved and results in similar structural representations as depicted in the right column.

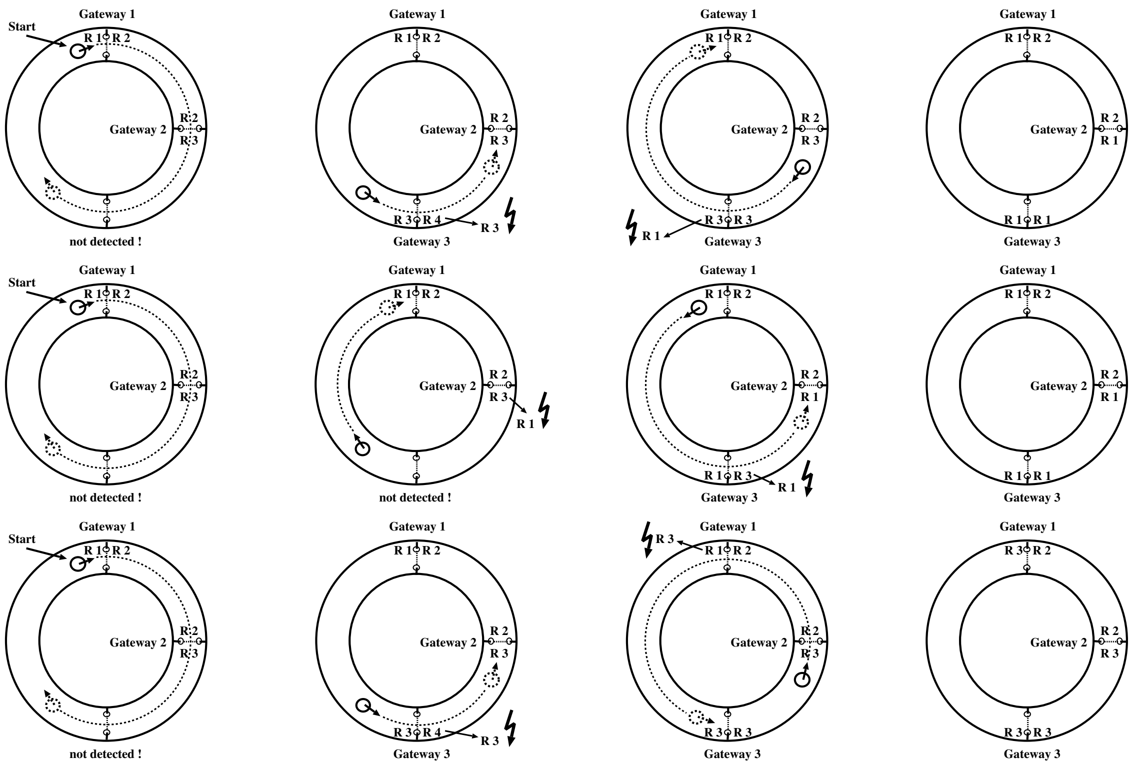


Figure 6.4: The problem of failing to detect a gateway upon first traversal

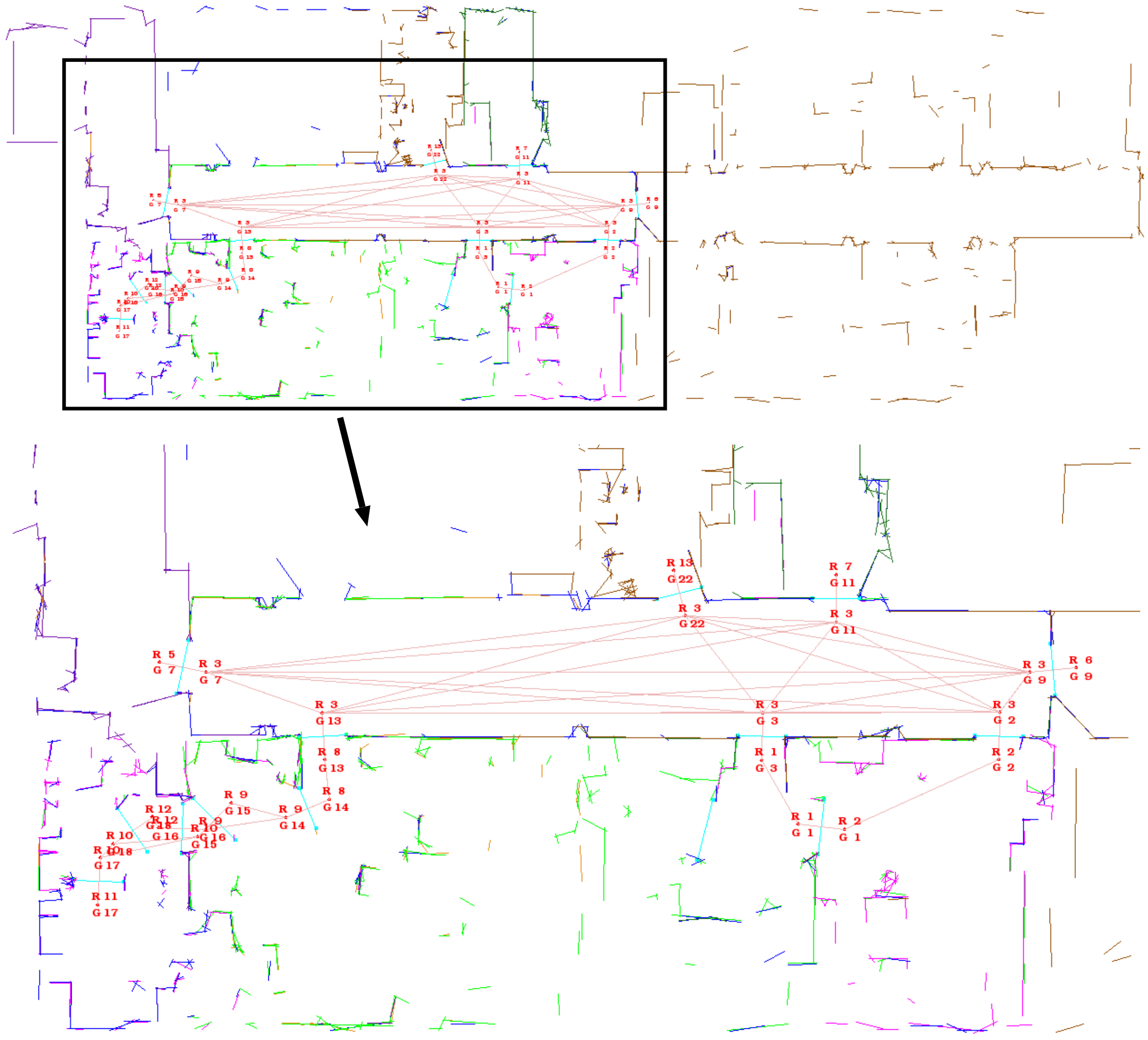


Figure 6.5: University Freiburg, Building 079. Thanks go to Cyrill Stachnis for providing this data set.

Another example is shown in Figure 6.5. A loop has been correctly closed in the lower right part and except from the area in the lower left part all observed gateways have been detected upon first traversal. Thus, for the larger part of the map the structural description is consistent. However, having a close look at regions R8 to R12 in Figure 6.6 it can be seen that the map contains one inconsistency. Several factors contribute to this situation. First, gateway G16 has been observed before traversing G15, thus it was labeled to connect R9 and R10. Then, gateway G18 was not detected upon first traversal and gateway G17 was correctly associated with R10 and R11. On return G18 was detected belonging to R10 and another new region R12, which lead to the correction of G15 that now connects R12 and R9. The problem is that G15 and G16 are too close to each other to disambiguate the situation. Clearly one of the two (G15 or G16) should be neutralized, preferably G15. Performing a global check for consistency on the Region & Gateway Graph is likely to solve that problem. We discussed this example to show that detecting gateways upon first traversal is crucial for the presented mapping approach to work robustly. Although for some of those situations inconsistencies can be resolved, problems may arise when gateways are too close to each other, i.e. in strongly cluttered narrow environments. Also, passing more than one gateway without detecting them is likely to introduce inconsistencies into the structural description.

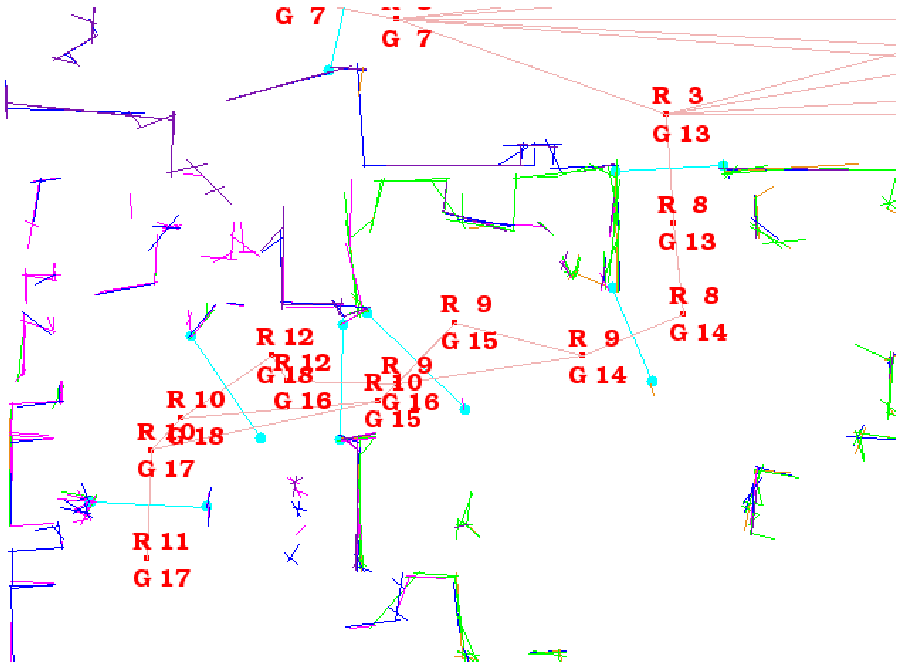


Figure 6.6: Enlarged section of the Freiburg data set shown in Figure 6.5

Earlier in this chapter, we presented results for a part of the Intel data set. Figure 6.7 shows how the whole environment is represented as a RG Map. In the center it has a narrow long hallway which causes numerous gateways to be detected when using only the provided laser data. Nevertheless, a closer look at this part in Figure 6.8 reveals that even though the area is oversegmented the structural description is indeed consistent. The only remaining problem is that region R1 and region R20 are actually the same. This connection could again not be detected because the robot did not visit another known gateway in R1 after entering R20. Instead it turns around and moves back through the narrow hallway towards its final destination in the upper part of R1 close to the gateway G15. Moving around in R1 the robot eventually approaches gateways that belong to region R20, namely G27 or G28, and then R1 and R20 would be merged.

Figure 6.10 shows the consistent RG Map of our department floor (Munich, Orleanstr.). The length of one hallway is approximately 50 meters. The robot traveled about 817 meters, collected 2638 scans and built the map while it was moving. We suppressed the RG Graph in this figure for visibility reasons. Additional examples of consistent RG Maps that were automatically generated from other data sets are depicted in Figure 6.11, 6.12, and 6.13. Figure 6.9 shows a 3D view of the largest region for the Intel Lab data set. Wall hypotheses have been generated from the 2D line segment map of that region by selecting line segments whose length exceeds a certain threshold.

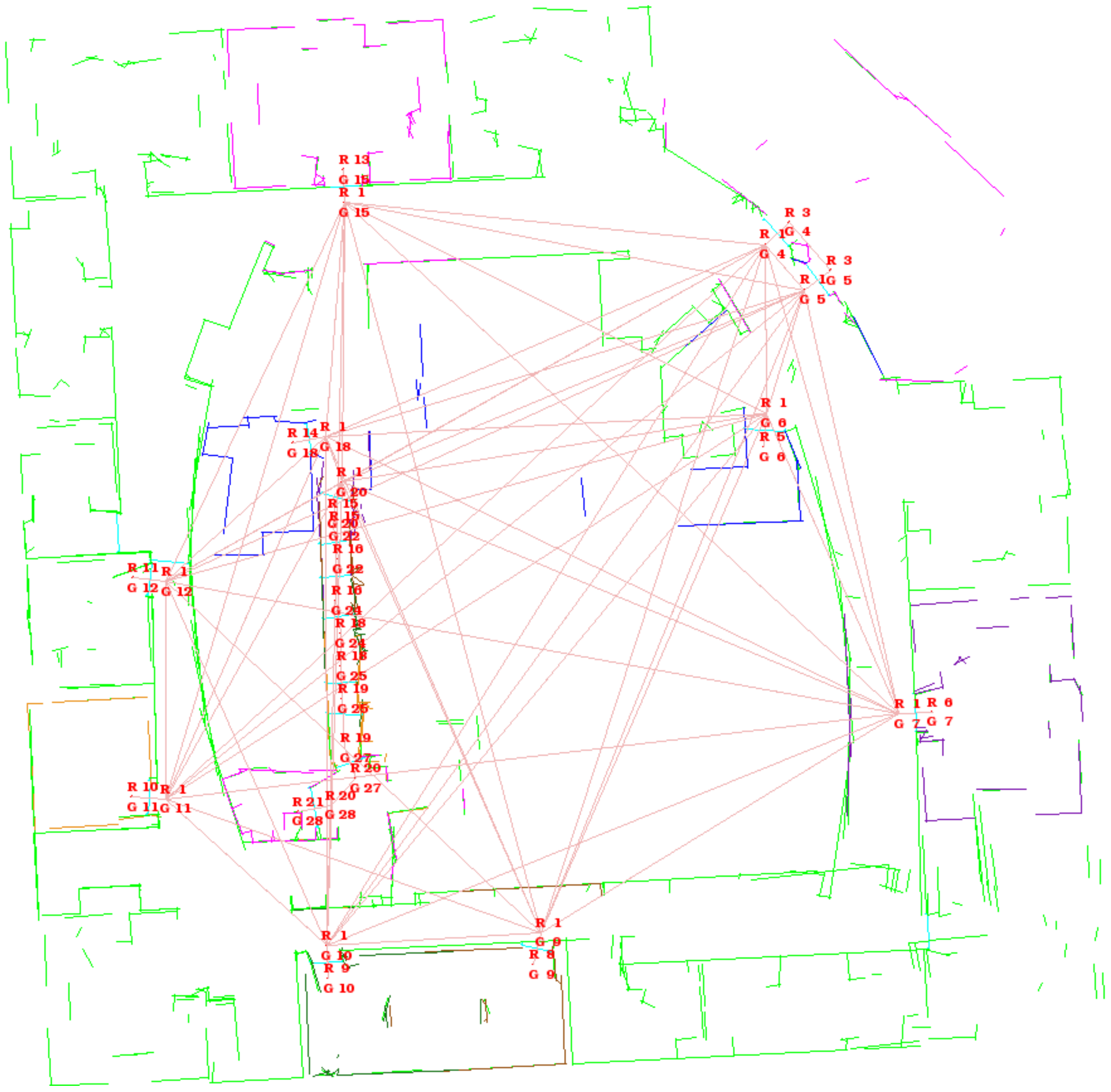


Figure 6.7: Complete RG Map of the Intel Laboratories (Seattle)

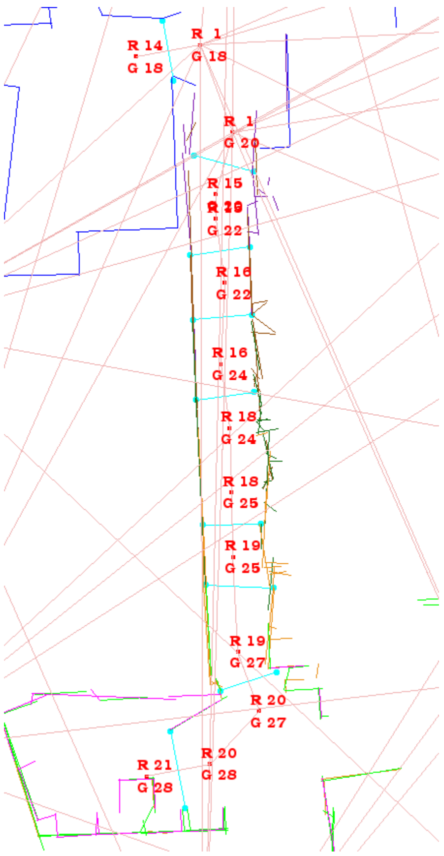


Figure 6.8: Narrow hallway in the Intel Laboratories (Seattle)

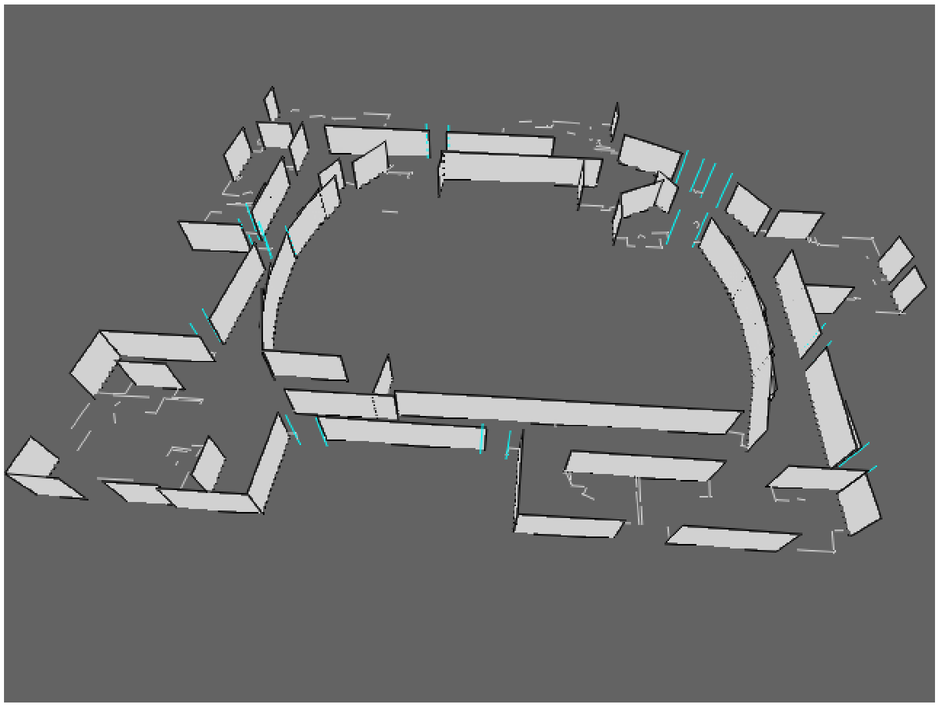


Figure 6.9: 3D view of the Intel Laboratories for region R1, according to Figure 6.7

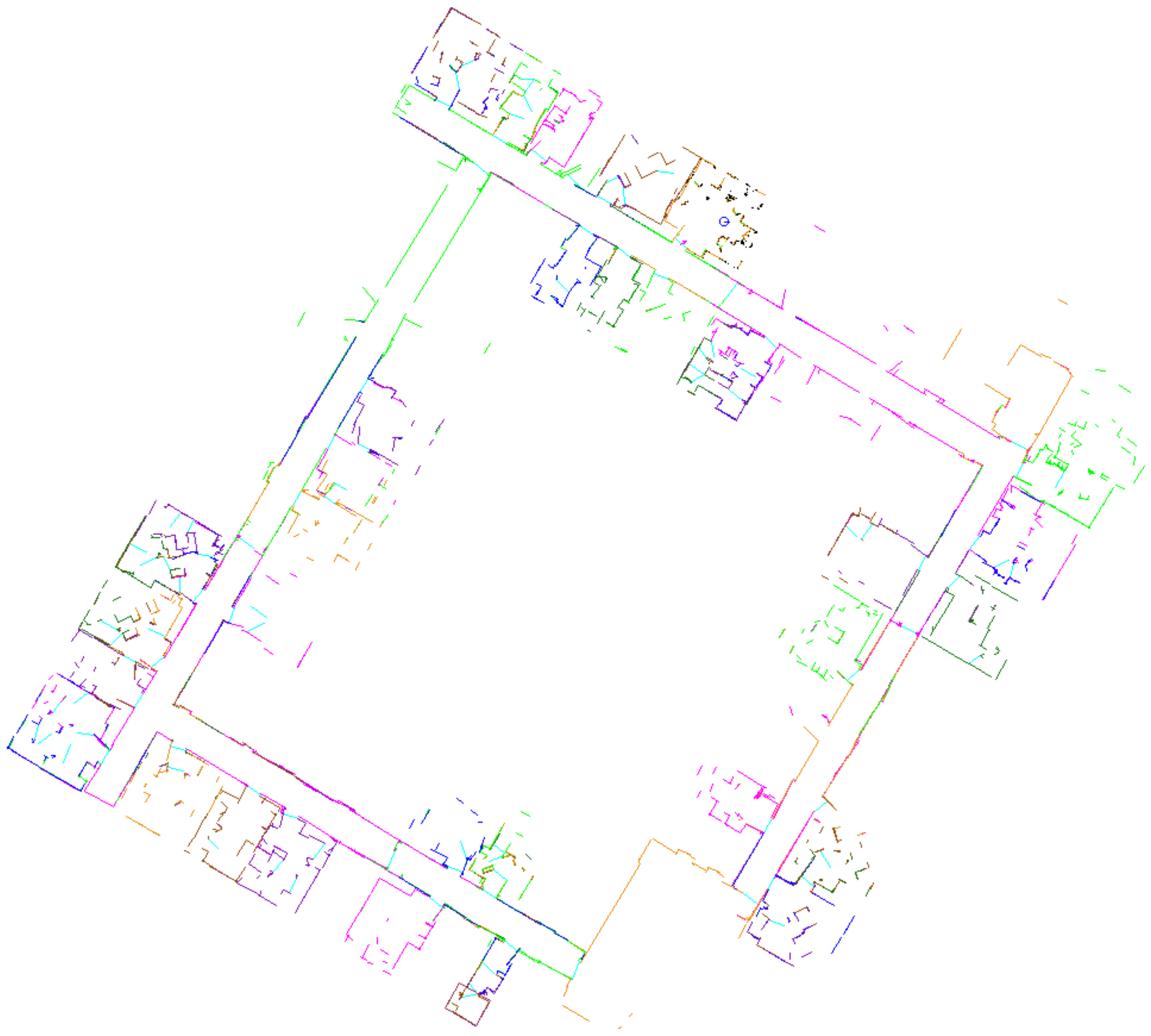


Figure 6.10: Department floor of Informatik IX, TU Munich (Orleanstr.). The data set comprises 2638 scans. The robot traveled a total of about 817 meter.

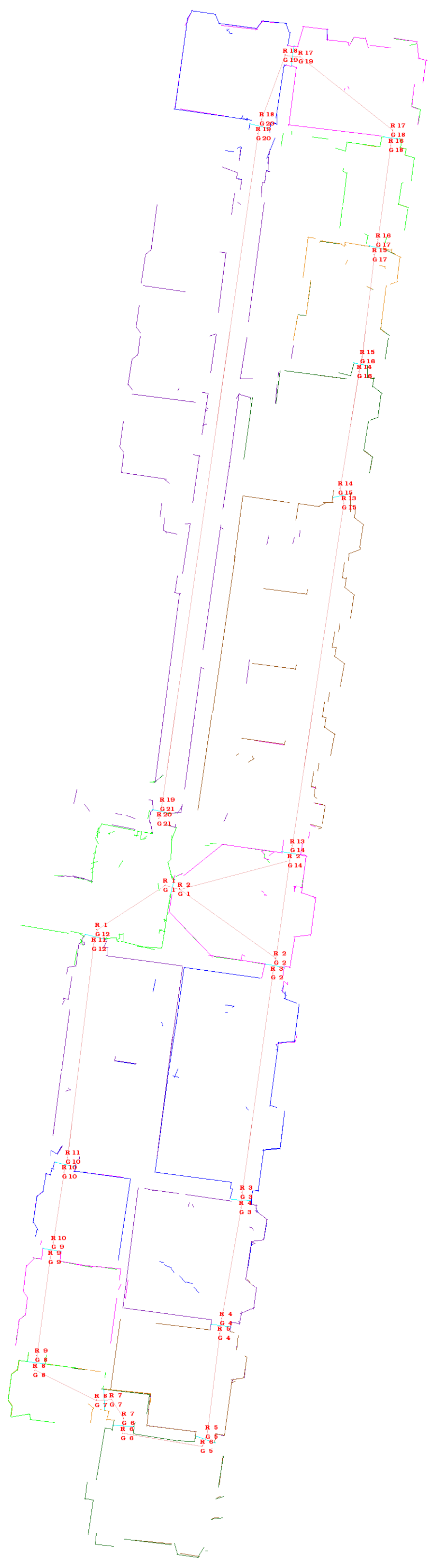


Figure 6.11: Castello di Belgioioso. Thanks go to Dirk Hähnel for providing this data set.

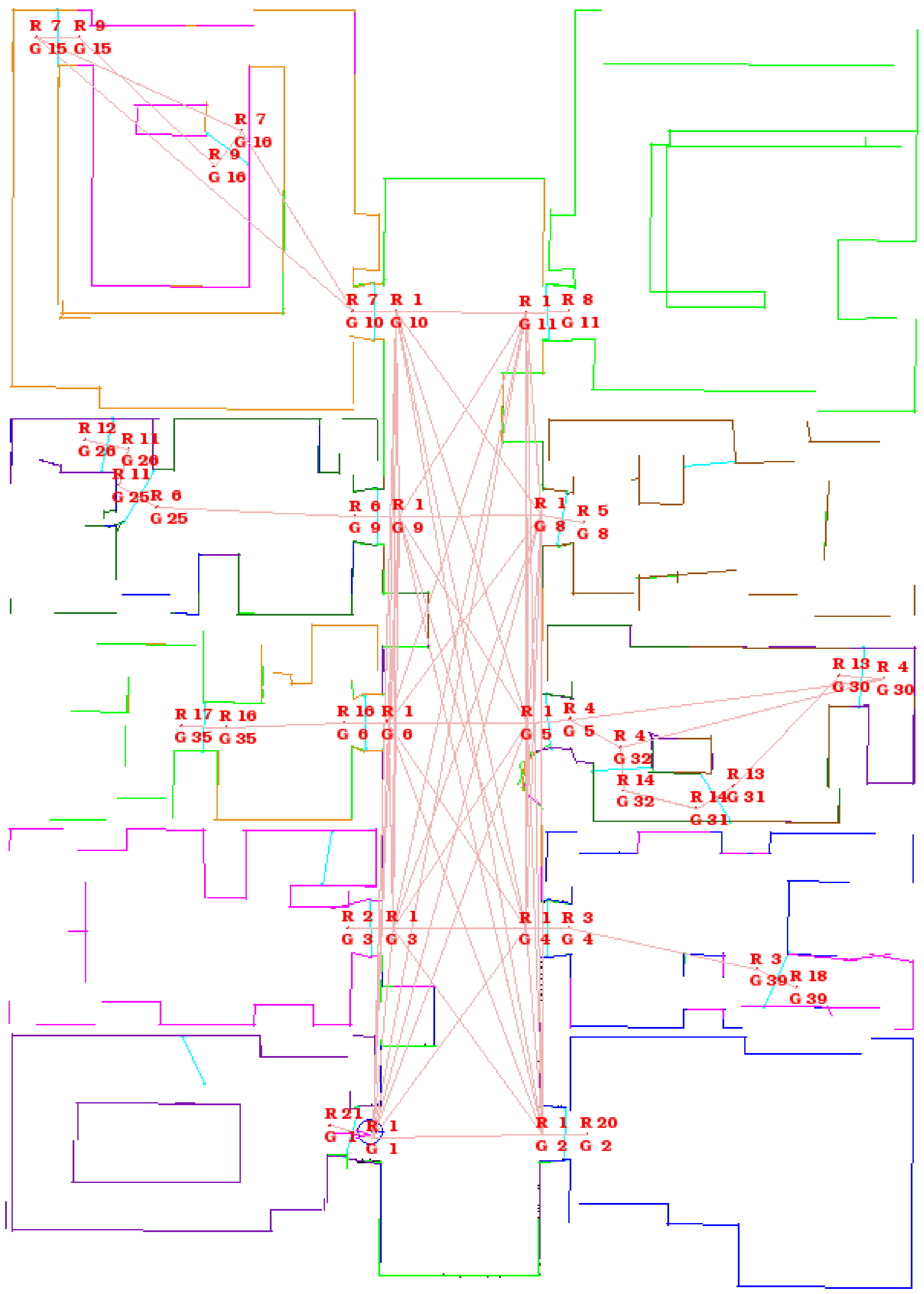


Figure 6.12: This data set has been generated using the RHINO Simulator and a CAD model of the Interiors of Informatik III at Bonn University.

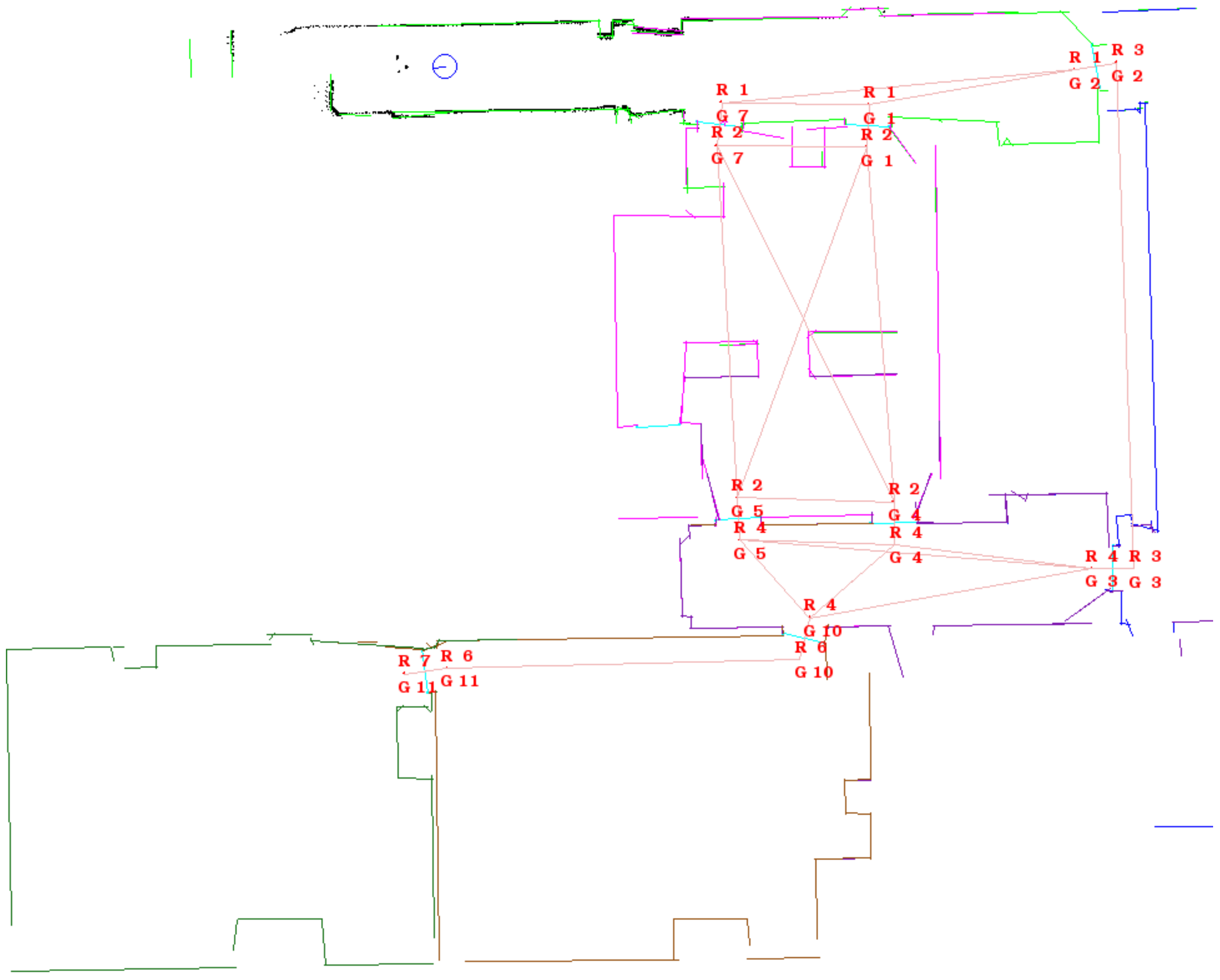


Figure 6.13: Fort AP Hill. Thanks go to Andrew Howard for providing this data set.

6.1.2 Closing loops in RG Mapping

In the previous subsection we briefly mentioned the problem of loop closing and that it is implicitly handled by resolving inconsistent observations. All experiments shown there were based on external data sets. In this subsection we present an experiment carried out with a B21r robot in our department. The robot traveled about 75 meters in our department floor, entering and leaving seven different regions: part of hallway - R1, part of another hallway plus printer lobby - R6 and five offices - R2, R3, R4, R5, R7.

The upper scene in Figure 6.14 shows the robot after it consecutively traversed all regions. Two gateways have not yet been observed or traversed, and both will lead to cycles in the environment. In the lower scene the robot added the new gateway G9, and falsely assumes that it has entered a previously unknown region (R8). This false assumption is solved upon observing a known gateway (G1). As a consequence region R1 and R8 are merged which means the loop is closed, as depicted in the left scene of Figure 6.15. Then, the gateway G13 is detected and generates a similar situation, right scene in Figure 6.15. After that the structural representation is correct and will not change on future visits. The final *RG Map* is shown in Figure 6.16 together with the odometry corrected raw data.

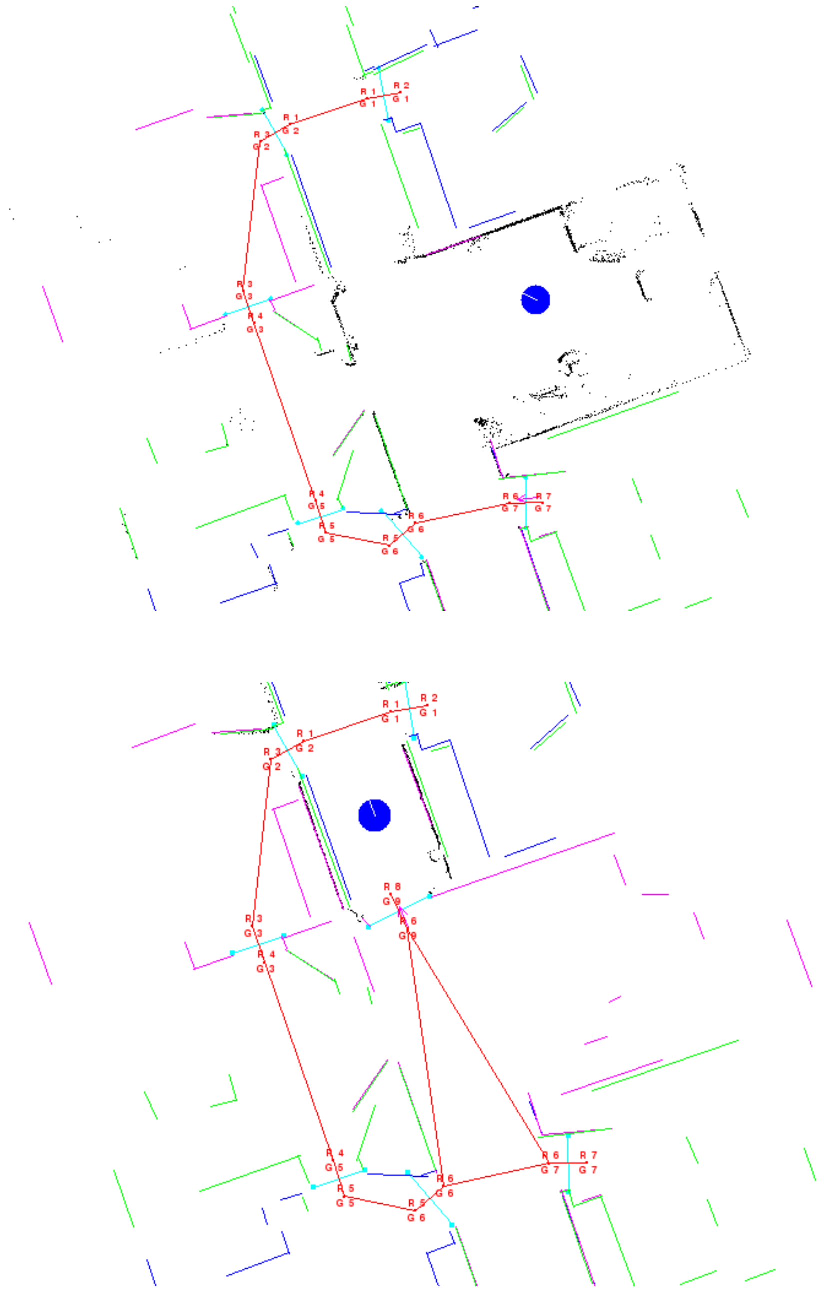


Figure 6.14: Closing loops in RG Mapping (part 1)

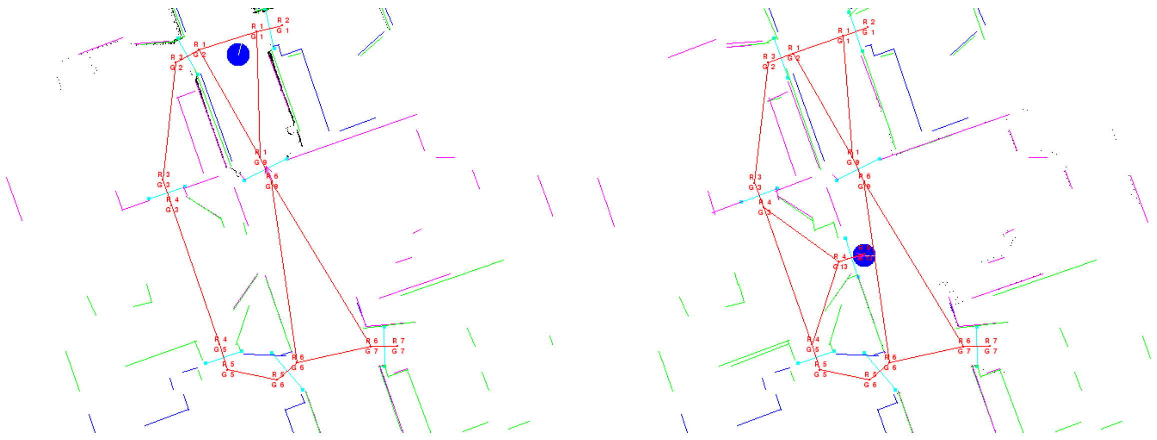


Figure 6.15: Closing loops in RG Mapping (part 2)

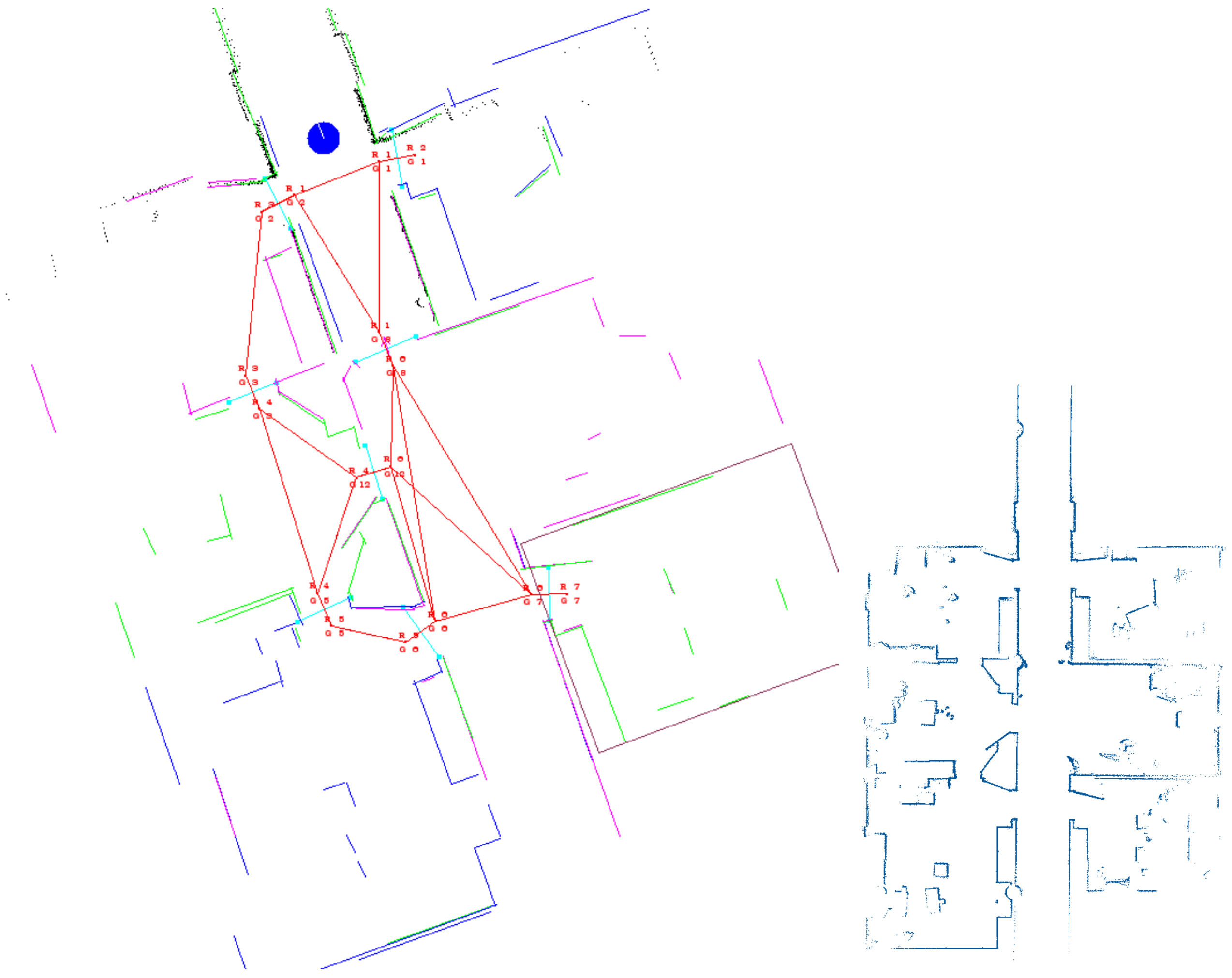


Figure 6.16: RG Map and the odometry-corrected raw data

6.1.3 Evaluation of the Detection of Narrow Passages

When using external data sets to generate 2D environment representations, even for humans it is not always clear which narrow passages refer to doors and which do not. In these cases we consider the gateway not to present a door, thus obtaining a rather pessimistic estimate of the false-positive rate. Only one gateway that was classified as a door was not correctly detected, thus, it is not represented in the respective map. Gateways in the map that have not yet been visited, e.g. a door the robot passed by but did not approach, are not considered.

environment	overall number of gateways	number of doors	number of neutralized or not yet traversed gateways	number of false-positives
Intel Lab (Seattle)	21	8	5	8
Fort AP Hill	8	8	0	0
Castello di Belgioioso	19	17	0	2
Dep. Informatik IX TUM, Garching	58	20	18	20
Dep. Informatik IX TUM, Orleanstr	100	41	25	34
Part of Building 079 Freiburg University	14	8	1	5
Data Set fig. 6.16 TU Garching	8	8	0	0
Dep. Informatik III Bonn University	22	10	3	9

Table 6.1: Gateway detection results for different indoor environments

Table 6.1 summarizes the gateway detection results. The number of neutralized gateways shows that in most experiments only few narrow passages have not been detected upon first traversal. Considering the respective RG Maps, almost all of these gateways are located within cluttered regions, i.e. offices. That means, they do not corrupt the general structure that refers to offices, rooms, hallways and doors. Clearly, the mapping process generated a considerable amount of false-positives. This shows that using only 2D laser data for the structuring process is not sufficient. For path planning, however, any structure that consistently represents the environment is advantageous, even though for humans the representation might not seem to be very intuitive. On the other hand, given the Region & Gateway representation, structural corrections are easy to accomplish by a simple human-machine interface, e.g. mouse click to choose regions that are afterwards automatically merged. Substantially better results can be achieved using vision-based detection of door frames, as described in Chapter 3, because it allows to resolve ambiguous observations from the laser-based detection.

6.2 RG Mapping using Laser and Vision

In Chapter 4 we described an algorithm for door detection based on laser range data and vision. The laser data is used to obtain hypotheses for doors, i.e. narrow passages. When a narrow passage is detected it triggers a search for 3D rectangles within the region description that match the given dimensions of a door frame. Note that the laser-based detection does not trigger the reconstruction process because when a narrow passage is detected the robot is already close to the door. At this point the door frame is not or only partially visible, thus, it can not be reconstructed. Instead, the *Vision Module* reconstructs 3D rectangles continually and independent of the laser-based detector. These rectangles are stored in the region description. Figure 6.17 shows two typical scenes where the robot moved along a hallway and reconstructed the door frames. In the lower image it can be seen that the reconstructed door frames match very well with the laser data (red). The vertical lines (cyan) on the left depict the narrow passage as it was detected using laser range data. The two observations together clearly lead to a more robust detection of doors.

We have shown that using 3D vision enables more elaborate detection of doors, but it also results in maps that contain substantially more information in terms of 3D object hypotheses. In Chapter 3 we have shown that certain prominent object classes, e.g. 3D rectangular objects, can be very accurately reconstructed using image segmentation, stereo vision and projective geometry. Aligning all objects within a common frame of reference is accomplished using laser-based odometry correction within regions, as described in Chapter 3.3.5. Figure 6.18 shows an example for a region description as it was built by the *TumBot Mapping & Navigation System* running on our B21r robot. The only information that has been added manually is the height of the walls. All objects and the position of the walls are automatically generated from the laser and vision data acquired within that region. It can be seen that the door is correctly detected (orange) and the alignment of 3D objects, i.e. table tops, monitor, closets, poster, shelf etc., is very accurate. The position of the walls has been determined from the bounding rectangle of the 2D line segment map.

Figure 6.19 gives additional prove for those findings. 3D wireframe models (red for the door and green else) of reconstructed objects are projected into the camera image for different robot poses and camera orientation. The models fit very well to the image data indicating the accuracy of the 3D reconstruction algorithm. The left part of the two upper images shows the generated 3D model as seen from the robot's view point.

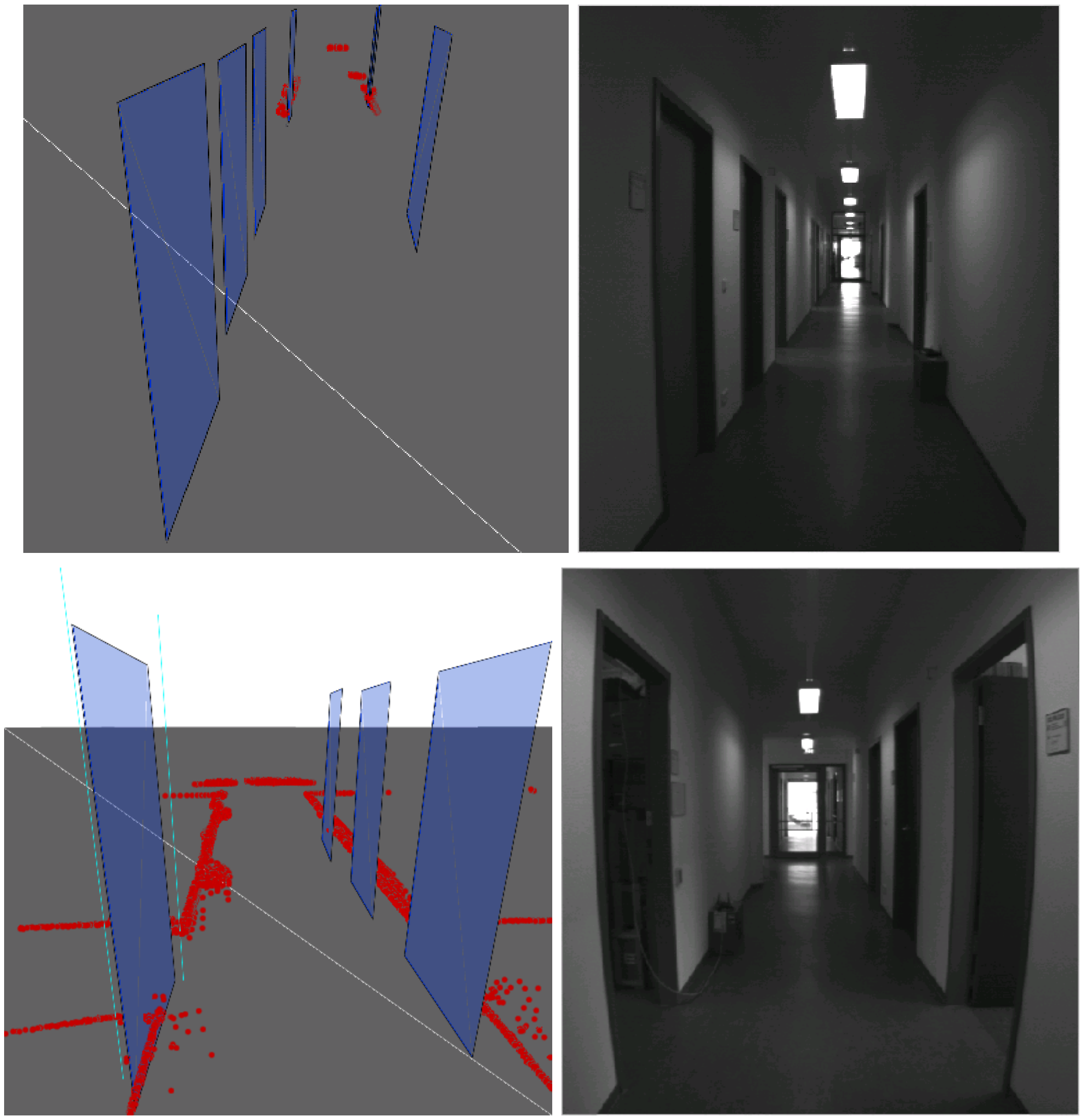


Figure 6.17: Doors detected in hallway scenarios

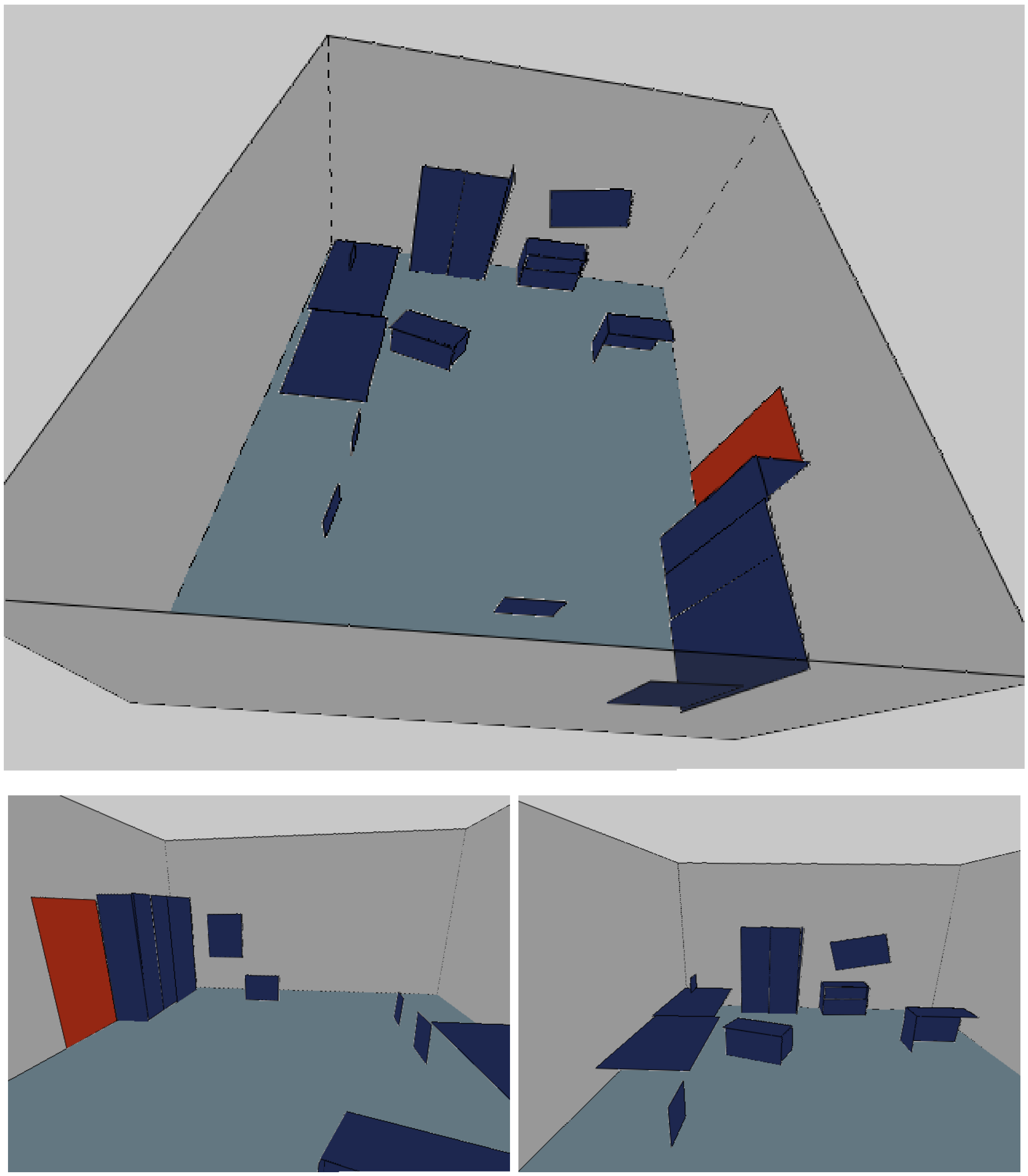


Figure 6.18: 3D view of a region showing the reconstructed rectangular 3D objects, the door (orange) and the walls. Except for the height of the walls the description was fully generated using only the sensor data associated with that region.

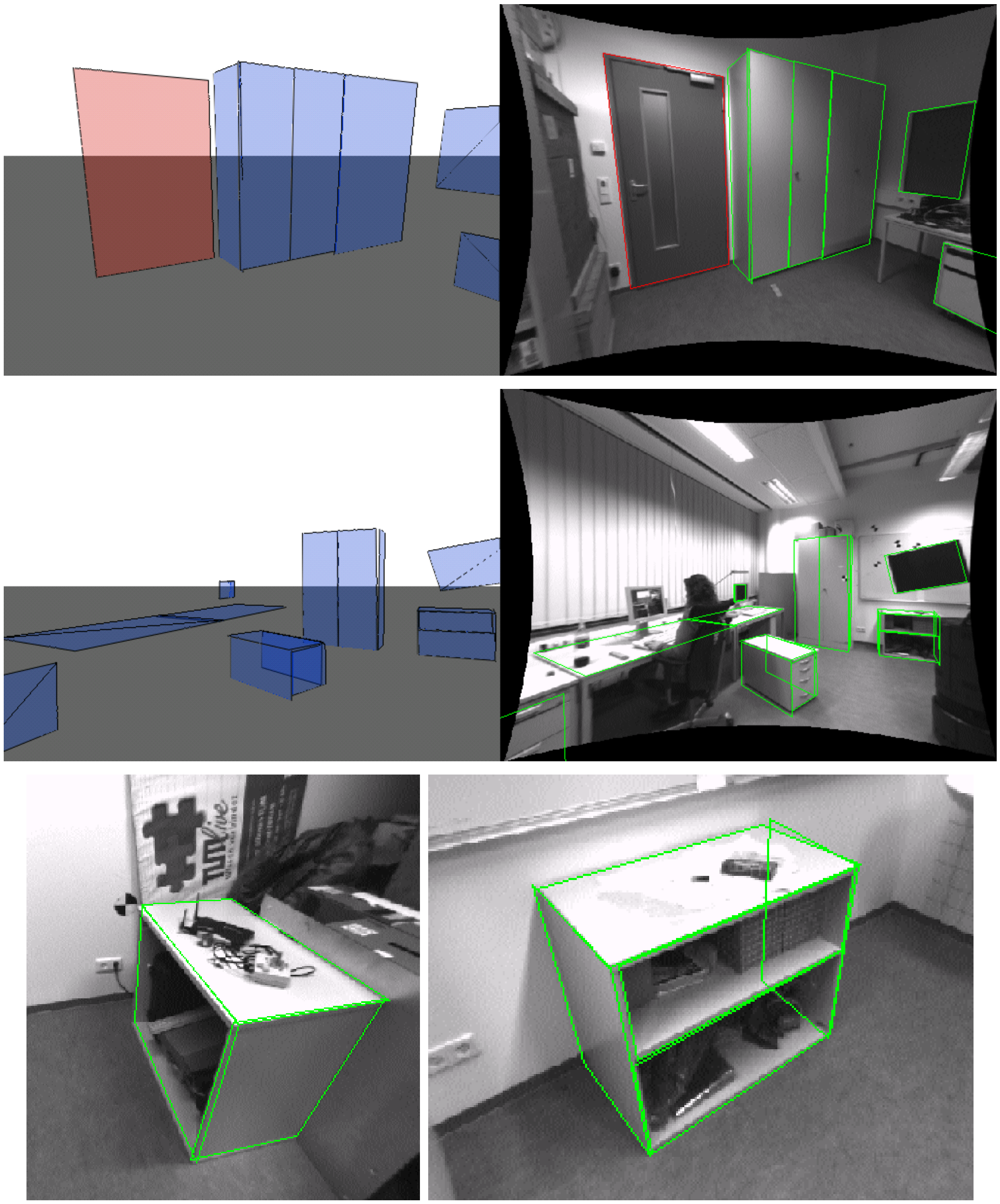


Figure 6.19: Different views from the robot's perspective. Overlayed in the images are the contours of reconstructed objects. The two upper left images are generated from the 3D model using the camera positions determined by the robot.

Chapter 7

Conclusion

In this thesis, we presented *Region & Gateway Mapping* as a novel approach to map building. It automatically generates *Region & Gateway Maps*, which are structured representations of indoor environments, using 2D laser range data and stereo vision. RG Maps extend the state-of-the-art, in particular metric maps, by providing a compact description of the environment's topology and explicit references to 2D/3D objects within the environment.

In contrast to topological approaches, RG Maps provide metric information for all places that have been visited by means of a region description. Using these structuring mechanisms the complexity of localization and path planning is substantially reduced because only a subset of the map's data has to be considered, namely the description of the respective region. Global path planning utilizes the *Region & Gateway Graph* that is easily generated from a given RG Map. The edges in the graph represent path segments for gateway and region traversals together with their respective properties, such as the path length, an estimation of the required time, and the probability of the segment being blocked. Thus, global path planning based on RG Maps simplifies the computational problem of finding paths for reliable and efficient navigation.

In the context of our mapping system, we developed an algorithm for automatically acquiring models of rectangular 3D objects based on visual information. The algorithm uses image segmentation, stereo vision and projective geometry, and makes little assumptions on the object class, besides that the objects share some general geometric outline. For a variety of different scenes and objects in office environments we have shown that our method generates very accurate models. In addition, we described a method to determine the probability of reconstruction success.

Successful RG Mapping requires the robot to reliably and accurately detect and recognize gateways. To this end, we proposed a new algorithm for the detection and classification of hallway crossings and turns using Hidden Markov Models. The observation models for each state are represented by mixtures of Gaussians which are automatically determined based on k-means clustering. Extensive evaluation based on 20000 observation sequences (about 200000 observations) validated the applicability of this approach. Our gateway classification method outperforms heuristic methods, and generalizes well with regard to different hallway width, the degree of clutter and other parameters of the environment. In addition, we developed a method for detecting doors based on 2D laser data and the 3D reconstruction of door frames. This method combines laser-based detection of narrow passages and our 3D reconstruction algorithm. Our experiments with a number of different data sets showed that using only laser does not enable the robot to reliably distinguish a door from a narrow passage between a wall and a compact table. Thus, the combination of laser range sensing and vision has proven itself both to be necessary as well as sufficient for reliable doorway recognition.

Finally, based on RG Mapping we developed a distributed mapping and navigation system that we evaluated in a number of experiments using our B21r robot and a variety of data sets. We demonstrated that RG Mapping is both well suited and capable of acquiring structured and object-oriented representations of large indoor environments. The system also features a novel approach to collision avoidance that uses trajectory segments and stereo vision.

The approach and ideas presented in this thesis pose a number of interesting and challenging future research objectives. First of all, active exploration would substantially improve the robustness of structured mapping. In the context of RG Maps, each region should first be completely explored before traversing any new gateway. Gateways that have not yet been traversed can be stored on a stack and then used to direct the exploration behaviour. Also, it seems advantageous to visit these gateways in a breadth first manner to prevent ambiguities in loop closing. New methods for gateway detection play an important role in this context. Another subject of research is the exploitation of region recognition. Being able to robustly recognize regions would allow for a new SLAM approach based on regions instead of all observations. The robot would explore the environment region-by-region and after each completion of a new region, it determines the probability of having observed this region before. These likelyhoods can then be used in a probabilistic framework to determine the topological structure, in particular to close loops correctly.

Bibliography

[1] D. Avots, E. Lim, R. Thibaux, and S. Thrun. A Probabilistic Technique for Simultaneous Localization and Door State Estimation with Mobile Robots in Dynamic Environments. In *Proc. of the IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2002.

[2] N. Ayache and O. Faugeras. Maintaining Representations of the Environment of a Mobile Robot. *Int. Journal of Robotics and Automation*, 5(6):804–819, 1989.

[3] P. Beeson, N.K. Jong, and B. Kuipers. Towards Autonomous Topological Place Detection using the Extended Voronoi Graph. In *Proc. of IEEE International Conf. on Robotics and Automation (ICRA), New Orleans/USA*, 2005.

[4] P. Beeson, M. MacMahon, J. Modayil, J. Provost, F. Savelli, and B. Kuipers. Exploiting Local Perceptual Models for Topological Map-Building. *IJCAI Workshop RUR-03*, 2003.

[5] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 24(4):509–522, April 2002.

[6] D.E. Bernard, G.A. Dorais, C. Fry, E.B. Gamble Jr., B. Kanefsky, J. Kurien, N. Millar, W. Muscettola, P.P. Nayak, B. Pell, K. Rajan, N. Rouquette, B. Smith, and B.C. Williams. Design of the Remote Agent Experiment for Spacecraft Autonomy. In *Proc. of IEEE Aerospace Conf.*, 1998.

[7] R. Biswas, B. Limketkai, S. Sanner, and S. Thrun. Towards Object Mapping in Dynamic Environments with Mobile Robots. In *Proc of the IEEE International Conf. on Intelligent Robots and Systems (IROS)*, 2002.

[8] J. Borenstein and Y. Koren. Real-Time Obstacle Avoidance for Fast Mobile Robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):1179–1187, - 1989.

[9] J. Borenstein and Y. Koren. The Vector Field Histogram - Fast Obstacle Avoidance for Mobile Robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, 1991.

[10] H. Borotschnig, L. Paletta, M. Prantl, and A. Pinz. Appearance-based Active Object Recognition. *Image and Vision Computing*, 18(9):715–728, 1999.

[11] J.E. Bresenham. Pixel-Processing Fundamentals. *CGA*, 16(1):74–82, Januar 1996.

[12] O. Brock and O. Khatib. High-speed Navigation using the Global Dynamic Window Approach. In *Proc. of IEEE International Conf. on Robotics and Automation (ICRA)*, 1999.

[13] J. Buhmann, W. Burgard, A.B. Cremers, D. Fox, T. Hofmann, F. Schneider, J. Strikos, and S. Thrun. The Mobile Robot RHINO. *AI Magazine*, 16(2):31–38, 1995.

[14] W. Burgard, A.B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Experiences with an Interactive Museum Tour-Guide Robot. *Artificial Intelligence*, 114(1-2):3–55, 1999.

[15] W. Burgard, A.B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(1-2), 2000.

[16] D. Burschka, C. Eberst, and C. Robl. Vision Based Model Generation for Indoor Environments. In *Proc. of the IEEE International Conf. on Robotics and Automation (ICRA)*, pages 1940–1945, 1997.

[17] C. C. Schlegel and R. Wörz. The Software Framework SmartSoft for Implementing Sensorimotor Systems. In *Proc. of IEEE/RSJ International Conf. on Intelligent Robots and Systems (IROS)*, pages 1610–1616, 1999.

[18] J. Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 9(6):679–698, 1986.

[19] J.A. Castellanos, J.M.M. Montiel, and J. Neira. The SPmap: A Probabilistic Framework for Simultaneous Localization and Map Building. In *IEEE Transactions on Robotics and Automation*, volume 15(5), pages 948–953, 1999.

[20] R. Chatila and J.-P. Laumond. Position Referencing and Consistent World Modeling for Mobile Robots. In *Proc. of IEEE International Conf. on Robotics and Automation (ICRA)*, 1985.

[21] D.Z. Chen, R.J. Szczerba, and J.J. Uhran Jr. A Framed-Quadtree Approach for Determining Euclidean Shortest Paths in a 2-D Environment. *IEEE Transactions on Robotics and Automation*, 13(5):668–681, 1997.

[22] E. Chown, S. Kaplan, and D. Kortenkamp. Prototypes, Location and Associative Networks (PLAN): Towards a Unified Theory of Cognitive Mapping. *Cognitive Science*, 19(1):1–52, 1995.

[23] Eric Chown. Making predictions in an uncertain world: Environmental structure and congnitive maps. *Adaptive Behaviour*, 7(1):1–17, 1999.

[24] Eric Chown. Gateways: An Approach to parsing Spatial Domains. In *ICML 2000 Workshop on Machine Learning of Spatial Knowledge*, 2000.

[25] C.I. Connolly. The Determination of Next Best Views. In *Proc. of the IEEE International Conf. on Robotics and Automation (ICRA)*, volume 2, pages 432– 435, 1985.

[26] M. Csorba. *Simultaneous Localisation and Map Building.* PhD thesis, University of Oxford, 1997.

[27] R. Deriche. Using Canny's criteria to derive a recursively implemented optimal edge detector. *International Journal of Computer Vision*, 1(2):167–187, 1987.

[28] G.N. DeSouza and A.C. Kak. Vision for Mobile Robot Navigation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 24, Feb. 2002.

[29] G. Dissanayake, H.F. Durrant-Whyte, and T. Bailey. A Computationally Efficient Solution to the Simultaneous Localisation and Map Building (SLAM) Problem. In *Working notes of ICRA Workshop W4: Mobile Robot Navigation and Mapping*, 2000.

[30] G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba. A Solution to the Simultaneous Localisation and Map Building (SLAM) Problem. *IEEE Transactions of Robotics and Automation*, 2001.

[31] P. Doherty, P. Haslum, F. Heintz, T. Merz, T. Persson, and B. Wingman. A Distributed Architecture for Intelligent Unmanned Aerial Vehicle Experimentation. In *Proc. of International Symposium on Distributed Autonomous Robotic Systems (DARS)*, pages 1610–1616, 2004.

[32] T. Duckett, S. Marsland, and J. Shapiro. Learning Globally Consistent Maps by Relaxation. In *Proc. of the IEEE International Conf. on Robotics and Automation (ICRA)*, 2000.

[33] R. O. Duda, P. E. Hart, and D. G Stork. *Pattern Classification.* New York: John-Wiley & Sons, Inc., second edition., 2001.

[34] A. Elfes. *Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation.* PhD thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, 1989.

[35] Christopher Fedor. TCX, an Interprocess Communication System for Building Robotic Architectures. Technical report, Carnegie Mellon University, 1994.

[36] R. Fergus, P. Perona, and Zisserman A. Object Class Recognition by Unsupervised Scale-Invariant Learning. In *Proc. of IEEE International Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 2, page 264, 2003.

[37] D. Fox, W. Burgard, and S. Thrun. The Dynamic Window Approach to Collision Avoidance. *IEEE Robotics & Automation Magazine*, 4(1), 1997.

[38] D. Fox, W. Burgard, and S. Thrun. Markov Localization for Mobile Robots in Dynamic Environments. *Journal of Artificial Intelligence Research*, 11, 1999.

[39] D. Fox, W. Burgard, S. Thrun, and A. B. Cremers. A Hybrid Collision Avoidance Method for Mobile Robots. In *Proc. of IEEE International Conf. on Robotics and Automation (ICRA)*, pages 1238–1243, 1998.

[40] D. Fox, S. Thrun, F. Dellart, and W. Burdard. *Sequential Monte Carlo Methods in Practice*, chapter Particle Filters for Mobile Robot Localization. Springer Verlag, New York, 2000.

[41] J. Guivant and N. Nebot. Optimization of the Simultaneous Localization and Map Building Algorithm for Real-Time Implementation. In *IEEE Transaction of Robotic and Automation*, 2001.

[42] J.-S. Gutmann. Robuste Navigation autonomer mobiler Systeme (in German). Akademische Verlagsgesellschaft Aka, Berlin, 2000. Doctoral Thesis *University of Freiburg*.

[43] J.-S. Gutmann, W. Burgard, D. Fox, and K. Konolige. An Experimental Comparison of Localization Methods. In *Proc. of the IEEE/RSJ International Conf. on Intelligent Robots and Systems (IROS)*, 1998.

[44] J.-S. Gutmann and K. Konolige. Incremental Mapping of Large Cyclic Environments. In *Proc. of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, 2000.

[45] D. Hähnel, W. Burgard, and S. Thrun. Learning Compact 3D Models of Indoor and Outdoor Environments with a Mobile Robot. In *The fourth European workshop on advanced mobile robots (EUROBOT)*, 2001.

[46] D. Hähnel, D. Fox, W. Burgard, and S. Thrun. A highly efficient FastSLAM algorithm for generating cyclic maps of large-scale environments from raw laser range measurements. In *Proc. of IEEE IROS, Las Vegas/USA*, 2003.

[47] D. Hähnel, D. Schulz, and W. Burgard. Map Building with Mobile Robots in Populated Environments. In *Proceeding of International Conf. on Intelligent Robots and Systems (IROS)*, 2002.

[48] D. Hähnel, R. Triebel, W. Burgard, and S. Thrun. Map Building with Mobile Robots in Dynamic Environments. In *Proceeding of the IEEE International Conf. on Robotics and Automation (ICRA)*, 2003.

[49] C.G. Harris and M. Stephens. A Combined Corner and Edge Detector. In *Proc. of the 4th Alvey Vision Conf.*, pages 147–151, 1988.

[50] R.I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521623049, 2000.

[51] M. Hebert, T. Kanade, and I. Kweon. 3-D Vision Techniques for Autonomous Vehicles. Technical Report CMU-RI-TR-88-12, Carnegie Mellon University, 1998.

[52] F. Heintz and P. Doherty. DyKnow: An Approach to Middleware for Knowledge Processing. *Journal of Intelligent & Fuzzy Systems*, 15(1), 2004.

[53] D. Huber, O. Carmichael, and M. M. Hebert. 3D map reconstruction from range data. In *Proc. of the IEEE International Conf. on Robotics and Automation (ICRA)*, volume 1, pages 891–897, April 2000.

[54] Andreas Iizuka. Trajectory Based Control in the Context of Mobile Robots. Master's thesis, Technische Universität München, Faculty of Computer Science, Chair IX, 2004.

[55] L. Iocchi, K. Konolige, and M. Bajracharya. Visually Realistic Mapping of a Planar Environment with Stereo. In *International Symposium on Experimental Robotics (ISER)*, 2000.

[56] Luca Iocchi and Kurt Konolige. A Multiresolution Stereo Vision System for Mobile Robots. Technical report, Universita di Roma, 1998.

[57] I-K. Jung and S. Lacroix. High Resolution Terrain Mapping using Low Altitude Aerial Stereo Imagery. In *Proc. of International Conf. on Computer Vision (ICCV)*, 2003.

[58] K. Klein and V. Sequeira. View Planning for the 3D Modelling of Real World Scenes. In *Proc. of the IEEE International Conf. on Intelligent Robots and Systems (IROS)*, volume 2, pages 943–948, 2000.

[59] Kurt Konolige. Small Vision Systems: Hardware and Implementation. In *Proc. to 8th International Symposium on Robotics Research, Hayama, Japan*, 1997.

[60] Kurt Konolige. A Gradient Method for Realtime Robot Control. In *Proc. of IEEE/RSJ International Conf. on Intelligent Robots and Systems (IROS)*, 2000.

[61] D. Kortenkamp and T. Weymouth. Topological Mapping for Mobile Robots using a Combination of Sonar and Vision Sensing. In *Proc. of the 12th National Conf. on Artificial Intelligence (AAAI)*, 1994.

[62] David Kortenkamp. *Cognitive Maps for Mobile Robots: A Representation for Mapping and Navigation*. PhD thesis, University of Michigan, 1993.

[63] A. Kosaka and A.C. Kak. Fast Vision-Guided Mobile Robot Navigation Using Model-Based Reasoning and Prediction of Uncertainties. *Computer Vision, Graphics, and Image Processing Image Understanding*, 56(3):271–329, 1992.

[64] J. Kosecka and F. Li. Vision Based Topological Markov Localization. In *Proc. of the IEEE International Conf. on Robotics and Automation (ICRA)*, 2004.

[65] J. Kosecka, F. Li, and X. Yang. Global Localization and Relative Positioning based on Scale-invariant Keypoints. *Robotics and Autonomous Systems*, 2005.

[66] B. Kuipers and Y.-T. Byun. A Robot Exploration and Mapping Strategy based on a Semantic Hierachy of Spatial Representations. *Journal of Robotics and Autonomous Systems*, 8:47–63, 1991.

[67] B. Kuipers, J. Modayil, P. Beeson, M. MacMahon, and F. Savelli. Local Metrical and Global Topological Maps in the Hybrid Spatial Semantic Hierarchy. In *Proc. of IEEE International Conf. on Robotics and Automation (ICRA), New Orleans/USA*, 2004.

[68] Benjamin Kuipers. Modeling Spatial Knowledge. *Cognitive Science*, 2:129–153, 1978.

[69] Benjamin Kuipers. The Spatial Semantic Hierarchy. *Artificial Intelligence*, 119:191–233, 2000.

[70] Stefan Lanser. *Modellbasierte Lokalisation gestützt auf monokulare Videobilder*. PhD thesis, Technische Universität München, 1997.

[71] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.

[72] J. Leal, S. Scheding, and G. Dissanayake. 3D Terrain Mapping: A Stochastic Approach. In *Proc. of the Australian Conference on Robotics and Automation (ACRA)*, pages 135–140, 2001.

[73] J.J. Leonard and H.F. Durrant-Whyte. Dynamic Map Building for an Autonomous Mobile Robot. *International Journal of Robotics Research, 11(4):89-96*, 1992.

[74] Y. Liu, R. Emery, D. Chakrabarti, W. Burgard, and S. Thrun. Using EM to learn 3D Models with Mobile Robots. In *Proc. of the International Conf. on Machine Learning (ICML)*, 2001.

[75] D.G. Lowe. Object recognition from local scale-invariant features. In *Proc. of IEEE International Conf. on Computer Vision (ICCV)*, pages 1150–1157, 1999.

[76] F. Lu and E. Milios. Globally Consistent Range Scan Alignment for Environment Mapping. In *Autonomous Robots*, volume 4, pages 333–349, 1997.

[77] Kevin Lynch. *The Image of the City.* Technology Press, Cambridge, MA, 1960.

[78] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1:281-297, Berkeley, USA, 1967. University of California Press.

[79] Ravi Malladi, James A. Sethian, and Baba C. Vemuri. Shape Modeling with Front Propagation: A Level Set Approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 17(2):158–175, 1995.

[80] J. Maver and R. Bajcsy. Occlusions as a Guide for Planning the Next View. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 15(5):417–433, 1993.

[81] J. Minguez and L. Montano. Nearness Diagram Navigation (ND): A New Real Time Collision Avoidance Approach for Holonomic and nonholonomic Mobile Robots. In *Proc. of IEEE/RSJ International Conf. on Intelligent Robots and Systems (IROS)*, 2000.

[82] J. Minguez, L. Montano, and J. Santos-Victor. Reactive Navigation for Nonholonomic Robots using the Ego Kinematic Space. In *Proc. of IEEE International Conf. on Robotics and Automation (ICRA)*, 2002.

[83] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem. In *Proc. of the National Conf. on Artificial Intelligence (AAAI)*, Edmonton, Canada, 2002.

[84] H.P. Moravec. The Stanford Cart and the CMU Rover. *Proc. IEEE*, 71(7):872–884, 1983.

[85] H.P. Moravec. Sensor Fusion in Certainty Grids for Mobile Robots. *AI Magazine*, 9(2):61–74, 1988.

[86] N. Muscettola, P.P. Nayak, B. Pell, and B.C. Williams. Remote Agent: To Boldly Go Where No AI System Has Gone Before. *Journal of Artificial Intelligence*, 103(1-2):5–48, August 1998.

[87] MVTec Software GmbH, München, Germany. *Halcon/C++ – Reference Manual, Version 7.0*, December 2003.

[88] H. Noborio, T. Naniwa, and S. Arimoto. A Quadtree-based Path-Planning Algorithm for a Mobile Robot. *Journal of Robotic Systems*, 7(4):555–574, 1990.

[89] A. Pentland, B. Moghaddam, and T. Starner. View-based and Modular Eigenspaces for Face Recognition. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 1994.

[90] J. Piaget and B. Inhelder. The Child's Conception of Space. New York: Norton, 1967.

[91] Stefan Plafka. Region-Based Path Planning for RG Maps using framed Quad-Trees. Technical report, Technische Universität München, Faculty of Computer Science, Chair IX, September 2004.

[92] S. Quinlan and O. Khatib. Elastic Bands: Connecting Path Planning and Control. In *Proc. of IEEE International Conf. on Robotics and Automation (ICRA)*, pages 802–807, 1993.

[93] L. R. Rabiner and B. H. Juang. An Introduction to Hidden Markov Models. *IEEE ASSP Magazine*, pages 4–15, January 1986.

[94] E. Remolina and B. Kuipers. Towards a General Theory of Topological Maps. *Artificial Intelligence*, 152:47–104, 2004.

[95] D. Roberts and A. Marshall. Viewpoint selection for complete surface coverage of three dimensional objects. In *Proc.of the British Machine Vision Conference (BMVC)*, pages 749–750, 1998.

[96] S. Romdhani, P. Torr, B. Schölkopf, and A. Blake. Computationally Efficient Face Detection. In *Proc. of IEEE International Conf. on Computer Vision (ICCV)*, 2001.

[97] D. Schröter, M. Beetz, and J.-S. Gutmann. RG Mapping: Learning Compact and Structured 2D Line Maps of Indoor Environments. In *Proc. of 11th IEEE ROMAN Conf., Berlin/Germany*, 2002.

[98] S. Se, D.G. Lowe, and J. Little. Vision-based mobile robot localization and mapping using scale-invariant features. In *Proc. of IEEE International Conf. on Robotics and Automation (ICRA)*, pages 2051–58, 2001.

[99] V. Sequeira, Kia Ng, E. Wolfart, J.G.M Goncalves, and D. Hogg. Automated 3D reconstruction of interiors with multiple scan-views. *Journal of Photogrammetry and Remote Sensing (ISPRS)*, 54:1–22, 1999.

[100] A.W. Siegel and S.H. White. The Development of Spatial Representations of Large-Scale Environments. In H.W. Reese, editor, *Advances in Child Development and Behavior*. Academic Press, 1975.

[101] Reid Simmons. Structured Control for Autonomous Robots. *IEEE Transactions on Robotics and Automation*, 10(1):34–43, February 1994.

[102] Reid Simmons. The curvature velocity method for local obstacle avoidance. In *Proc. of IEEE International Conf. on Robotics and Automation (ICRA)*, 1996.

[103] C. Stachniss and W. Burgard. An Integrated Approach to Goal-directed Obstacle Avoidance under Dynamic Constraints for Dynamic Environments. In *Proc. of IEEE/RSJ International Conf. on Intelligent Robots and Systems (IROS)*, 2002.

[104] M. Stricker and M. Swain. The Capacity of Color Histogram Indexing. In *Proc. of IEEE International Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 704–708, 1994.

[105] M. J. Swain and D. H. Ballard. Color Indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[106] D. S. Tan, J. T. Herro, and R. J Szczerba. Simulation of Euclidean Shortest Path Planning Algorithms Based on the Framed-quadtree Data Structure. Technical Report 95-26, University of Notre Dame CSE, 1995.

[107] S. Thrun. Robotic Mapping: A Survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium.* Morgan Kaufmann, 2002.

[108] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, S. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. Probabilistic Algorithms and the Interactive Museum Tour-Guide Robot Minerva. *International Journal of Robotics Research*, 19(11):972–999, 2000.

[109] S. Thrun and A. Bücken. Integrating Grid-Based and Topological Maps for Mobile Robot Navigation. In *Proc. of the 13th National Conf. on Artificial Intelligence (AAAI)*, 1996.

[110] S. Thrun, B. Bücken, W. Burgard, D. Fox, T. Fröhlinghaus, D. Henning, T. Hofmann, M. Krell, and T. Schmidt. Map Learning and High-Speed Navigation in RHINO. In D. Kortenkamp, R.P. Bonasso, and R. Murphy, editors, *AI-based Mobile Robots: Case Studies of Successful Robot Systems.* MIT Press, 1998.

[111] S. Thrun, D. Fox, and W. Burgard. A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots. In *Machine Learning*, volume 31, pages 29–53, 1998.

[112] S. Thrun, J.-S. Gutmann, D. Fox, W. Burgard, and B. Kuipers. Integrating Topological and Metric Maps for Mobile Robot Navigation: A Statistical Approach. In *Proc. of the 15th National Conf. on Artificial Intelligence (AAAI)*, 1998.

[113] N. Tomatis, I. Nourbakhsh, and R. Siegwart. Simultaneous Localization and Map Building: A Global Topological Model with Local Metric Maps. In *Proc. of the IEEE/RSJ International Conf. on Intelligent Robots and Systems (IROS)*, 2001.

[114] I. Ulrich and J. Borenstein. VFH*: Local Obstacle Avoidance with Look-Ahead Verification. In *Proc. of IEEE International Conf. on Robotics and Automation (ICRA)*, pages 2505–2511, 2000.

[115] H. Utz, S. Sablatnög, S. Enderle, and G.K. Kraetzschmar. Miro - Middleware for Mobile Robot Applications. *IEEE Transactions on Robotics and Automation, Special Issue on Object-Oriented Distributed Control Architectures*, 18(4):493–497, August 2002.

[116] B. Vijayakumar, D. Kriegman, and J. Ponce. Invariant-Based Recognition of Complex Curved 3D Objects from Image Contours. In *International Conf. on Computer Vision (ICCV)*, pages 508–514, 1995.

[117] P. Viola and M. Jones. Robust Real-time Object Detection. *International Journal of Computer Vision*, 2002.

[118] I. Weiss and M. Ray. Model-based recognition of 3D objects from single images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 23(2):116–128, February 2001.

[119] B.C. Williams, M. Ingham, S.H. Chung, and P.H. Elliott. Model-based Programming of Intelligent Embedded Systems and Robotic Space Explorers. *Proc. of the IEEE: Special Issue on Modeling and Design of Embedded Software (invited paper)*, 9(1):212–237, January 2003.

[120] Brian Yamauchi. A Frontier-Based Approach for Autonomous Exploration. In *Proc. of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, 1997.

[121] W.K. Yeap and M.E. Jefferies. Computing a Representation of the Local Environment. *Artificial Intelligence*, 107(2):265–301, 1999.

[122] G.M. Youngblood, L.B. Holder, and D.J. Cook. A Framework for Autonomous Mobile Robot Exploration and Map Learning through the use of Place-Centric Occupancy Grids. In *Proc. of the Machine Learning Workshop on Learning From Spatial Information*, 2000.

[123] Z. Zhang and O. Faugeras. A 3D World Model Builder with a Mobile Robot. *Robotics Research*, 11(4):269–285, 1992.