Hierarchical Voronoi Graphs

Jan Oliver Wallgrün

Hierarchical Voronoi Graphs

Spatial Representation and Reasoning for Mobile Robots

Springer

Dr.-Ing. Jan Oliver Wallgrün
Cognitive Systems Group
Department of Mathematics and Informatics
University of Bremen
P.O. Box 330 440
28334 Bremen
Germany
wallgruen@informatik.uni-bremen.de

This thesis was accepted as doctoral dissertation by the Department of Mathematics and Informatics, University of Bremen, under the title “Hierarchical Route Graph Representations for Mobile Robots Based on Generalized Voronoi Graphs”. Based on this work the author was granted the academic degree Dr.-Ing.

Date of oral examination: 19 August 2008

Reviewers:

Prof. Christian Freksa, Ph.D. (University of Bremen, Germany)
Prof. Benjamin Kuipers, Ph.D. (University of Texas at Austin, USA)

ISBN 978-3-642-10302-5 e-ISBN 978-3-642-10345-2
DOI 10.1007/978-3-642-10345-2
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2009941853

ACM Computing Classification (1998): I.2, J.2.

Cover design: KuenkelLopka GmbH

Printed on acid-free paper

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

Foreword

What is space? Is there space when there are objects to occupy it or is there space only when there are no objects to occupy it? Can there be space without objects? These are old philosophical questions that concern the ontology of space in the philosophical sense of ‘ontology’ – *what is the nature of space?*

Cognitive science in general and artificial intelligence in particular are less concerned with the nature of things than with their mental conceptualizations. In spatial cognition research we address questions like *What do we know about space? How is space represented? What are the representational entities? What are the representational structures?* Answers to these questions are described in what is called ontologies in artificial intelligence. Different tasks require different knowledge, and different representations of knowledge facilitate different ways of solving problems. In this book, Jan Oliver Wallgrün develops and investigates representational structures to support tasks of autonomous mobile robots, from the acquisition of knowledge to the use of this knowledge for navigation.

The research presented is concerned with the robot mapping problem, the problem of building a spatial representation of an environment that is perceived by sensors that only provide incomplete and uncertain information; this information usually needs to be related to other imprecise or uncertain information. The routes a robot can take can be abstractly described in terms of graphs where alternative routes are represented by alternative branches in these route graphs. The proposed hierarchical Voronoi graph representation provides a conceptualization of space excellently suited for navigation, high-level spatial reasoning about environments, and agent-agent communication about navigable space. The problem is that the Voronoi representations are rather sensitive to measurement errors; therefore unique representations cannot easily be derived in practice. In this book, Jan Oliver Wallgrün presents a number of significant contributions to the state of the art in robot mapping by developing techniques for generating stable Voronoi representations from real sensor data.

Robot mapping and the construction of Voronoi graphs can also be seen as a spatial reasoning problem in which local spatial knowledge has to be consistently integrated into global knowledge. On an abstract qualitative level, these kinds of spatial reasoning problems are studied in the area of qualitative spatial reasoning. Jan Oliver Wallgrün successfully brings together these two distinct research areas; this results in a multi-hypothesis tracking framework for topological mapping. From the perspective of qualitative spatial reasoning, the work describes an instructive application domain for comparing spatial calculi. It also provides valuable results regarding the applicability of direction information and existing reasoning techniques. The research presented in this book raises significant challenges for the development of spatial calculi and spatial reasoning systems that have to be addressed by future theoretical research.

Bremen,
July, 2009

Christian Freksa

Preface

Learning an internal spatial model of an initially unknown environment is considered to be one of the fundamental capabilities for an autonomous spatial agent. It is noticeable that—in contrast to what is known about mental spatial representations of humans—most map learning approaches in robotics employ sensor-near representation formats in which the environment is described by providing precise locations of environmental features using a single and absolute frame of reference. The question of how the spatial properties of the environment should best be represented in order to support fundamental tasks like navigation and communication about space is frequently not addressed at all. Instead, the focus is on the problem of how to build up a spatial model from uncertain sensor data for a representation approach which is assumed as given.

One goal of the work described in this book is to take a more general view on the robot mapping problem, explicitly distinguishing between the spatial representation perspective and the uncertainty handling perspective: The spatial representation perspective is concerned with what kind of spatial information should be represented and how this information should be represented in the model in order to adequately support a broad range of spatial competences. The uncertainty handling perspective addresses the question of how to deal with the inherent uncertainty that makes learning of an environmental model such a challenging problem.

The main contributions of this work are made with respect to one particular spatial representation approach, in which the environment is modeled as a hierarchical route graph based on the generalized Voronoi diagram. From the spatial representation perspective, this approach is particularly well suited as the core representation for environments with a clear route structure, such as most indoor environments. The main challenge with regard to this kind of representation and the focus of this book is the development of techniques that allow the robust construction of the spatial model under uncertainty.

One underlying thesis of this work is that the combination of rather abstract representations—like the route graphs considered here—with proven uncertainty handling methods is a promising direction of research. It has the potential of leading to approaches which are at the same time robust and well-suited to realize high-level spatial cognitive abilities. Another concern of this book is to investigate the application of constraint satisfaction techniques stemming from research on qualitative spatial reasoning for consistently integrating local observations into survey knowledge in the context of robot mapping.

The techniques developed in this book are concerned with three different aspects of the model acquisition process: First, the problem of constructing the proposed hierarchical representation from noisy 2D range data is considered, assuming that the correct data association between perceived elements and the corresponding elements in the robot's internal model is given. The main results are a measure to assess the relevance and stability of Voronoi nodes and the methods to automatically build up the hierarchy based on this measure.

Second, the data association problem is considered with the goal of achieving reliable identification of Voronoi nodes on the local level. A matching approach for Voronoi graphs is developed which takes into account variations caused by sensor limitations and allows the incorporation of geometric constraints.

Third, to deal with uncertainty on the global level, mapping is formulated as the problem of finding a minimal route graph model that explains the history of observations and actions, an approach which has been proposed by Kuipers. The developed solution consists of a best-first branch and bound search through the tree of possible associations of nodes resulting in a simultaneous tracking of multiple map hypotheses. The focus here is on investigating the adequacy of qualitative direction constraints and the planarity constraint to significantly reduce the search space.

Besides the individual techniques developed in this work, which can be combined in multiple ways, an overall mapping system is presented that is able to construct the Voronoi-based hierarchical route graph representation directly from range data. Although the described methods have been developed with regard to this particular representation, contributions like the data association approach and the results concerning global mapping using qualitative spatial constraints are more generally applicable and provide insights that are also relevant outside the robot mapping domain.

Acknowledgements

Writing this book would not have been possible without the support and input from numerous people I had the pleasure of meeting over the last few years during my time as a doctoral candidate. I am deeply grateful for all the assistance I received.

First and foremost, I would like to thank my advisor, Christian Freksa, for his continuous encouragement and guidance. I appreciate that he gave me the freedom to explore and pursue my own ideas. At the same time, he always provided valuable advice and feedback when I needed it and has been tremendously supportive. I am sincerely grateful for the great time I had in his group and the inspiring interdisciplinary research environment he has created.

I also want to express my gratitude to Benjamin Kuipers for his willingness to review my thesis and for taking interest in my research. His pioneering work and in particular his course at the ISCSI in Bad Zwischenahn in 2003 had a major impact on this work.

I also owe many thanks to my colleagues and friends at the Cognitive Systems group at the Universität Bremen for countless discussions and helpful suggestions with regard to my work and for generally creating a very comfortable and entertaining working environment. In particular, I would like to thank Diedrich Wolter for being my first address to discuss new ideas and for providing invaluable advice. Special thanks also go to the other members of the R3-[Q-Shape] project, Frank Dylla, Lutz Frommberger, and Reinhard Moratz. The collective project work has been highly enjoyable and had a huge impact on my own research.

Several people directly contributed to this work and deserve my gratitude: Cyrill Stachniss provided the FastSLAM implementations used for the evaluation part of this work. I also utilized two exploration data sets made publically available by Nick Roy and Dirk Hähnel. Martin Held's software Vroni was used in the software implementations of some of the ideas described here. Stefan Dehm, Lutz Frommberger, Kai-Florian Richter, Christoph Sippel, Holger Schultheis, and Thorsten Timm all helped to improve this text by providing feedback on earlier versions and by proofreading it.

Financial support of my research by the German Research Foundation (DFG) within the scope of the Spatial Inference project of the SPP Raumkognition and the R3-[Q-Shape] project of the SFB/TR 8 Spatial Cognition is gratefully acknowledged.

Finally, I want to express my gratitude to my family for their constant encouragement, love, and support, especially during the last months of intensive writing. This book is dedicated to you.

Bremen, Jan Oliver Wallgrün
July, 2009

Contents

Notation

2^M	Power set of set M.
$A \rightleftharpoons B$	Edge connecting A and B in a route graph model.
$E(G)$	Set of edges of graph G.
$G(M)$	Route graph model of a map hypothesis M.
$L(J)$	Set of hallway identifiers of junction observation J.
$M^2(d, m)$	Mahalanobis distance between feature d and feature m.
$R(J)$	Spatial description of junction observation J.
$R_i^{[v]}$	Accessible region corresponding to edge $e_i^{[v]}$ of node v.
$V(G)$	Set of nodes of graph G.
$\text{argmin}_{x \in M} f(x)$	Argument x from set M that minimizes $f(x)$ (arbitrarily chosen when multiple such elements are contained in M) or—when explicitly stated—the set of all x from M that minimize $f(x)$.
$\text{child}_i v$	ith child node of node v in an AGVG subtree.
$\deg(v)$	Degree of node v.
$\mathcal{A}_i^{[v]}$	Set of nodes contained in accessible region $R_i^{[v]}$ of node v.
$\mathcal{HT}_{\mathcal{H}}$	Set of hallway traversals contained in history $\mathcal{H}$.
$\mathcal{JO}_{\mathcal{H}}$	Set of junction observations contained in history $\mathcal{H}$.
$\max M$	Maximum from set M.
$\max_{x \in M} f(x)$	Maximum from $\{f(x) \mid x \in M\}$.
$\text{lb}(R_i^{[v]})$	If true, the rsm value of $R_i^{[v]}$ is a lower bound estimate.
$\text{lb}(v)$	If true, the vnrm value of v is a lower bound estimate.
$\text{neighbor}_i(v)$	ith neighbor node v in an AGVG.

$\mathrm{rsm}(R_i^{[v]})$	Region significance value of accessible region $R_i^{[v]}$.
$\mathrm{connects}(e)$	Yields the 2-element set of nodes incident to edge e.
$\mathrm{cyclic}(e)$	If true, edge e is part of a cycle in the graph.
$\mathrm{mj}(J)$	Function that maps junction observation J to a node in a map hypothesis.
$\mathrm{ml}(l)$	Function that maps leaving hallway identifier l to an edge in a map hypothesis.
$\mathrm{other}(e, v)$	Node incident to edge e that is not v, or v if e is a loop.
$\triangle_v^{(u)}$	Subtree with v as root and ancestor u.
$\triangle_w^{u,v}$	mm-subtree.
$\triangle_v^u$	m-subtree.
$\mathrm{vnrm}(v)$	Voronoi node relevance value of Voronoi node v.
$d(v_i, v_j)$	Shortest path distance between nodes v_i and v_j in a graph.
$d \rightsquigarrow m$	Feature d is matched with feature m.
$d_{\text{succ}}(x, y)$	Distance between two objects x and y with regard to a cyclic order specified by successor function succ.
$e_i^{[v]}$	ith edge of node v (the edges are assumed to be numbered in accordance with the cyclic edge order specified for v).
$gp_i^{[v]}$	ith generating point of Voronoi node v.
$i \oplus j$	Index of jth successor of the element with index i in a cyclically ordered set of objects.
$l(e)$	Length of edge e.
$r(v)$	Radius of maximal inscribed circle of Voronoi node v.
$s(J)$	Successor function specifying the cyclic order of leaving hallways for junction observation J.
$\|M\|$	Cardinality of set M.
$\|\|p - q\|\|$	Euclidean distance between p and q (length of vector $p-q$).

Abbreviations

AGVG	Annotated generalized Voronoi graph (Sect. 3.3).
EGVG	Embedded generalized Voronoi graph (Sect. 3.2).
EKF	Extended Kalman filter (Sect. 2.4.1.2).
GVD	Generalized Voronoi diagram (Sect. 3.1).
GVG	Generalized Voronoi graph (Sect. 3.2).
HAGVG	Hierarchical annotated generalized Voronoi graph (Sect. 3.4).
ICNN	Individual compatibility nearest neighbor (Sect. 5.1.2).
QSR	Qualitative spatial reasoning.
RGE	Route graph edge (Sect. 6.1).
RGN	Route graph node (Sect. 6.1).
rsm	Region significance measure (Sect. 4.1).
SLAM	Simultaneous localization and mapping.
SSH	Spatial semantic hierarchy.
VD	Voronoi diagram (Sect. 3.1).
vnrm	Voronoi node relevance measure (Sect. 4.1).

List of Figures

List of Tables

Chapter 1
Introduction

Artificial spatial agents, such as the mobile robots we will be concerned with in this book, have a great deal in common with their natural counterparts, animals and humans. They are all embodied physical systems situated in the real world, important characteristics regarded by many as a prerequisite for genuine intelligence (Pfeifer & Bongard, 2007; Varela et al., 1992). They share the ability to perceive their environment and extract spatial information from their perceptions. They store spatial information over time and this information affects their future decisions and actions. And they are able to affect the state of their environment by their actions. An important part of acting in space involves moving to other parts of the environment outside of the agent's current sensory scope and navigation between known places. Hence, a spatial agent benefits from the ability to integrate local observations and to derive spatial relationships on a larger scale.

The details of how spatial information is extracted, stored, and processed in humans and animals are still largely unclear and subject of ongoing research across many disciplines. In this book, we follow the tradition of Braitenberg's "law of uphill analysis and downhill invention" (Braitenberg, 1984) in the sense that we attempt to design artificial agents that demonstrate a certain set of competences, possibly drawing inspiration from empirical studies about how these competences are achieved by humans or animals. The results, positive and negative ones, can then be used to draw conclusions about spatial information processing in natural agents.

The central object of attention of this book is the *cognitive map* (O'Keefe & Nadel, 1978; Tolman, 1948) or *spatial model* of an agent. We use these terms in a broad sense for the entire collection of long-term spatial knowledge held by an agent. The set of competences considered comprises the learning of a cognitive map based on observations of the environment (referred to as *mapping*) and its utilization for *navigation* (including localization, path planning, etc.), autonomous *exploration* of a previously unknown environment, and *communication* about space with other agents. The main problem we are concerned with is the *robot mapping problem* of autonomously constructing and maintaining a suitable spatial model, a problem which has sparked off a vast amount of research over the last few decades but still provides many challenges.

1.1 The Robot Mapping Problem

As Trullier et al. (1997) point out, animals and humans employ a diverse set of techniques to achieve successful navigation ranging from basic capabilities like aligning towards and approaching a visible target to advanced abilities like route planning and taking shortcuts. Advanced navigational capabilities like route planning are not conceivable without some kind of spatial representation that describes relations between distinct places known to the agent and that can serve as a basis for planning and decision making. However, this still leaves a wide area of possibilities about what kind of information is actually memorized, how it is stored and organized, and how these knowledge structures can be constructed from perceptions collected over time.

In the AI and robotics community, a common approach is to provide a mobile robot with a set of basic reactive procedures (e.g., collision avoidance, motion behaviors) and to model the cognitive map as a declarative knowledge representation implemented by a particular data structure on which algorithms for incorporating new observations, for localization, and for planning operate. Often this knowledge representation describing the robot's belief about the state of the environment is simply referred to as its *map*.

What makes designing a cognitive map data structure for a mobile robot together with algorithms to learn and maintain this representation a hard problem is the fact that all the agent's knowledge about the world in general and the spatial aspects of the environment in particular is inherently uncertain.[1] Only a small part of the environment is perceivable by the robot at any given time; whatever happens in other parts cannot be predicted with certainty. In addition, only some properties of the world are measurable by the robot at all, and these measurements are error-prone and noisy. The same holds for the actuation, which leads to uncertain results of performing a particular action. Finally, even if sensors and actuators would be perfect, the fact that every somewhat interesting section of the world can only be modelled efficiently by making use of approximations inescapably leads to uncertainty about the state of the world.

One important consequence of the inherent uncertainty is that it is usually not sufficient to model the cognitive map as a set of facts about the environment that represents the agent's current belief. Rather it is necessary to consider alternative models, and the robot might have different degrees of belief in the different alternatives. As a result, adequate methods to model and manage the uncertainty need to be conceived.

As we see, there exist two different perspectives to look at the robot mapping problem: (1) the spatial representation perspective asking what should be represented in the cognitive map and how it should be represented, and (2) the uncertainty handling perspective regarding the way uncertainty should be represented and handled in the system. Both aspects have to be considered in order to develop adequate solutions to the robot mapping problem.

[1]We here use the term *uncertainty* as a generic term for imperfect knowledge which does not allow us to tell whether a particular statement is true or not. It is, however, also often used in a more specific sense (see Smithson, 1989, for a taxonomy of different kinds of imperfect knowledge).

1.2 The Spatial Representation Perspective

A spatial representation is supposed to provide all the information that is required to reliably and efficiently perform the tasks at hand. Localization, for instance, necessitates the storage of information that allows the agent to recognize already visited places of the environment. This typically involves information about salient features and spatial relations holding between them. For path planning, the set of qualitatively distinct ways through the environment is needed together with information for deciding on a particular route. Communication with other agents (e.g., providing route directions) typically requires both, information for place recognition and route information.

From the point of view of a data structure, the cognitive map of an agent operating under time constraints should be organized in a way that allows performing the required operations (localization, incorporating new observations, route planning, etc.) as efficiently as possible. As there usually exist trade-offs between the efficiency of different operations, a reasonable demand is that the spatial model is optimized for best overall performance. In addition, in order to represent large environments, the cognitive mapping approach needs to scale well with the size of the represented environment in terms of efficiency of the operations and space consumption. Scalability demands a certain degree of sparseness and abstraction in the representation. Therefore, it is often a good approach to refrain from storing information that is directly perceivable when it is required or that is likely to change.

In the literature on mobile robot mapping, map construction is often studied as a problem per se, largely ignoring operations that should be realized based on the map. As a consequence, the proposed approaches are usually optimized for localization and updating the map, and involve simple sensor-near representations. While approaches like grid maps and geometric maps describe the boundaries of the free accessible space and thus allow for performing route planning (though with some computational effort), landmark-based maps consisting solely of positions of salient features do not allow the derivation of routes at all. On the other hand, approaches involving so-called topological maps are often directly based on the graph of distinct routes through the environment and are close to optimal with respect to the efficiency of route planning. However, they often lack the detailed geometric descriptions required for robust localization. Comparatively little research has been done on complex and heterogeneous forms of spatial representations.

1.3 The Uncertainty Handling Perspective

Uncertainty concerns both, the construction and maintenance of a model of the environment as well as the planning of actions and decision making based on this model. Whenever the robot performs an action, for instance locomotion, the uncertainty about the state of the world increases. New observations, on the other hand, can decrease the uncertainty when they include known features of the environment.

Various different ways of dealing with uncertainty in the spatial model have been employed in robot mapping systems. The simplest but also most error-prone way is to maintain a single model without representing the degree of belief in the relations and properties stored in the model at all. For every new observation the model is modified, resulting in a new world model. Approaches following this notion usually lack the ability to recover from errors made in earlier steps

Better results can be expected if for every relation or property in the cognitive map, the degree of belief in this fact is explicitly represented. The new world model is then the most likely one given the old model and the current observation. The predominant approach for describing and propagating belief values used in robotics is probability theory. However some approaches use other methods like Dempster-Shafer theory or fuzzy logic.

A different way to take uncertainty into account is to maintain a set of world models instead of just a single one. Two approaches can be found in the literature: First, a likelihood distribution over the space of all possible world models is maintained at any time. Every time an action is performed or a new observation becomes available, the likelihood distribution is updated accordingly. The likelihood distribution can usually only be represented approximately in these approaches. Second, a discrete set of hypotheses is tracked with or without belief values assigned to each hypothesis. Every time a hypothesis together with the current observation can be interpreted as two or more successive world models, all these interpretations are included in the set of overall hypotheses. The downside of approaches that track multiple hypotheses simultaneously, however, is increased computational costs.

1.4 Combining Representation and Uncertainty Handling

This work has to a large degree been motivated by the observation that of the many imaginable ways of combining spatial representation schemes with uncertainty handling mechanisms, only few combinations have been investigated extensively. The predominant approach in the last years has been to employ a plain global grid map or feature-based representation scheme combined with an uncertainty handling method that updates the likelihood distribution over all possible world models based on the recursive Bayes filter.

Within this class of approaches, much progress has been made through the development of sophisticated mathematical methods to represent and update the probability distributions. This raises the interesting question about whether the success of these approaches can be ascribed simply to their powerful uncertainty handling mechanisms rather than to the way they represent the spatial information.

More abstract representations like topological maps typically only employ very simple mechanisms for dealing with uncertainty, committing to a single hypothesis whenever incorporating a new observation. Therefore, these approaches currently do not achieve the same level of robustness and reliability. However, as pointed out earlier, more abstract representations have advantages when the focus is not on the mapping task alone, but on the entire bandwidth of spatial tasks, and generally they scale better to larger environments due to their compactness.

As a result, we think that more attention should be directed towards investigating ways to adapt the proven uncertainty handling methods for more abstract representation schemes with the goal of achieving a comparable level of robustness and reliability. In this book, we lay the groundwork to achieve this with regard to one particular spatial representation approach.

1.5 Route Graphs Based on Generalized Voronoi Diagrams

In this text, we will be concerned with one particular kind of spatial representation, namely the *route graph representation* (Werner et al., 2000), a special sort of topological map in which the environment is conceptualized as a network of distinct routes. and this network is abstracted into a graph structure. As we will see, this kind of representation is well suited to represent environments which possess a clear route structure. Hence, a typical scenario we consider in this book is that of an office-like indoor environment.

One means to derive the route structure of a planar environment from sensor data is the generalized Voronoi diagram. It can for instance be computed from a geometric 2D description extracted from range data as provided by a standard laser range finder. The Voronoi diagram can then be abstracted into a graph, the generalized Voronoi graph (see Fig. 1.1 for a first illustration of these concepts).

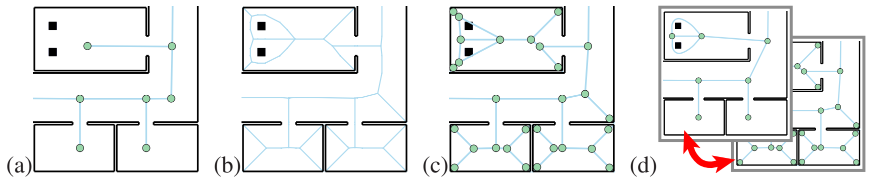


Figure 1.1: Illustration of the most important representational concepts we are concerned with in this work: (a) a general route graph model of a 2D environment, (b) the generalized Voronoi diagram, (c) the corresponding generalized Voronoi graph, and (d) a hierarchically organized route graph consisting of detailed layer and a coarser layer

Besides such plain Voronoi-based route graph models of an environment, we consider a hierarchical extension consisting of multiple route graph layers representing the environment at different levels of granularity. These hierarchical Voronoi graph representations allow performing important operations either on the most appropriate level of detail or in a hierarchical manner.

While from a spatial representation perspective Voronoi-based route graph representations and their hierarchical extensions have many advantages over the low-level representation approaches typically considered in today's mapping systems, the reliable acquisition of this kind of representation under uncertainty still poses an open problem. Most approaches involving a route graph representation only maintain a single model and tend to fail when a wrong decision is made in the construction process (e.g., when a loop is closed falsely).

It is our belief that to achieve a sufficient level of robustness of route graph model learning, techniques to deal with uncertainty at different levels and methods to recover from wrong decisions are required. In particular, the combination of route graph representations with established multi-hypothesis tracking methods is a promising direction of research.

1.6 Theses, Goals, and Contributions of This Book

The main goal of this work is to develop techniques for the Voronoi-based approach that make the combination with standard uncertainty handling methods possible. Furthermore, we want to demonstrate that this approach indeed leads to an improved robustness in the model acquisition process. The work is based on three main theses which have greatly influenced our approach to the robot mapping problem:

1. In order to realize high-level spatial cognitive abilities like spatial reasoning and communication about space on mobile robots, abstract levels of representation and complex organizational forms of spatial knowledge are required.

2. By combining abstract representation approaches with uncertainty handling and multi-hypothesis tracking methods, a similar level of robustness can be achieved as currently shown by the state-of-the-art approaches based on low-level sensor-near representations.

3. Relying on relations that can be more reliably observed reduces the problem of uncertainty handling and increases the scalability of the mapping approach.

The first two theses have already been touched upon previously. The third thesis expresses our believe that although most of the uncertainty handling methods have been developed for coordinate-based representations using a single absolute frame of reference, even better results can be achieved when combining them with representational approaches that are based on more abstract relations and employ multiple frames of reference. Relying on more abstract relations means that, on the one hand, the level of uncertainty in observations is decreased and, on the other hand, the size of the space of possible hypotheses about the state of the environment is reduced or even turned from a continuous space into a discrete one. Using finer relations only to describe local configurations of objects for which these relations can be determined quite reliably increases these effects while still providing enough detail, as, for instance, required for localization.

In the concrete work concerning our goal of providing techniques that allow robust model learning for our hierarchical Voronoi-based route graph representation, these notions are implemented in the following way: We only resort to detailed geometric information to specify the relative positions for local configurations of nodes. Mapping on the global level instead is based on abstract relations of connectivity and very coarse relations restricting the positions between the most relevant nodes. The high degree of reliability in the observations of these relations allows us to treat them as hard constraints and employ them to discard potential candidates when forming hypotheses at the local level (correctly identifying perceived nodes) or global level (determining the correct topology of the route graph).

In detail, the techniques for robust route graph model acquisition developed in this book tackle the problem at three levels.

First, at the level of reliable extraction and automatic abstraction we consider the problem of constructing the proposed hierarchical representation from 2D range data. The problem of finding the correct data association between perceived nodes or subgraphs and the corresponding entities in the robot's internal model is ignored in this part of the work. We develop techniques to assess the relevance and stability of nodes in a Voronoi graph derived from noisy sensor data, to autonomously create a hierarchy of coarser levels of representation, and to complement this representation when new information becomes available.

Second, on the level of local configurational knowledge we investigate the data association problem for Voronoi graphs, which requires the identification of corresponding elements in two Voronoi graphs. We extend existing data association approaches by including topological constraints and by incorporating the relevance of nodes into the matching process.

Third, at the global level we study the problem of map construction as the problem of abducting the simplest model that explains the history of observations, an approach that has been advocated by Kuipers (Kuipers et al., 2004; Remolina & Kuipers, 2004). The result is a multi-hypothesis tracking approach that computes a minimal consistent route graph model. We are especially interested in the question of the degree to which coarse directional information shrinks the search space of possible route graph hypotheses. In doing so, we apply techniques of spatial modeling and consistency checking developed in the area of qualitative spatial reasoning, and analyze the effects of qualitative direction and planarity constraints.

In the following, we briefly summarize the main contributions of this text.

- **Hierarchical annotated generalized Voronoi graph representation:** A hierarchically organized representation consisting of multiple linked Voronoi graph layers describing a planar environment at different levels of granularity and containing additional annotations to the graph structure.

- **Relevance and stability assessment of Voronoi nodes:** Measures and computation algorithms to assess the relevance of nodes in a Voronoi graph as well as their stability under noisy conditions.
- **Automatic generation of hierarchical Voronoi graph representations:** Methods that enable a robot to derive coarser Voronoi graphs and build up a complete hierarchical representation.
- **Data association methods for Voronoi graphs:** A matching algorithm for Voronoi graphs that incorporates different kinds of constraints to achieve reliable data association.
- **Multi-hypothesis approach for global mapping based on minimal model finding:** A best-first branch and bound search algorithm for minimal model computation using pruning based on planarity and consistency of qualitative direction information.
- **Empirical evaluation of the effects of qualitative direction constraints:** Systematic analysis of the effects of absolute and relative direction constraints as well as planarity constraints on the space of possible route graph hypotheses.
- **Overall Voronoi-based mapping system:** A complete mapping system for 2D range data that combines multi-hypothesis tracking and Voronoi-based route graph representations and merges the individual results of this work into an overall system.

1.7 Outline of This Book

The remainder of this text is structured as follows. In Chap. 2, we explore the robot mapping problem in detail and systematically review the relevant literature from the spatial representation perspective and the perspective of uncertainty handling and with regard to how both aspects have been combined. For this purpose, we introduce a two-level taxonomy of elementary spatial representations and distinguish several organization principles to combine them. The results of this analysis provide the motivation for the work presented in the later chapters.

Next, in Chap. 3 we turn towards the particular spatial representation approach we are concerned with in this work. We propose the hierarchical annotated Voronoi graph as a suitable representation approach for well-structured planar environments. We formally define the relevant concepts and structures and discuss advantages of the representation as well as challenges with regard to autonomously constructing and maintaining it under uncertainty.

In Chap. 4, we consider the problem of constructing the proposed hierarchical representation from 2D range data. We propose a relevance and stability measure for Voronoi nodes and describe the algorithms that allow us to reliably extract a Voronoi graph from noisy sensor data and autonomously create a hierarchy of coarser levels of representation.

Chapter 5 is concerned with the data association problem. We adapt and extend existing data association techniques for our needs and develop an approach to determine the optimal matching of two Voronoi graphs that enforces structural compatibility and geometric constraints.

Finally, we turn to the problem of global mapping in Chap. 6. Here we adopt Kuipers's framework of abducting the simplest model that explains a sequence of observations and actions. We present a branch and bound search algorithm through the interpretation tree, which effectively tracks multiple hypotheses about the topology of the environment. We also investigate the adequacy of qualitative spatial direction constraints and planarity constraints to efficiently prune the space of possible hypotheses and apply the developed theoretical framework to our Voronoi graph representation.

The experimental evaluation and quantitative analysis of the individual techniques developed in this book are presented in Chap. 7. We also explore several ways of combining these methods and present a complete mapping system.

Finally, in Chap. 8 we summarize and discuss the results achieved in this work and point out open questions and promising directions for future research.

As this work is based on techniques from two separate fields of AI research, namely the area of probabilistic robotics and the area of qualitative spatial reasoning, we have added brief overviews on the fundamentals of both areas in Appendix A and B, respectively.

Chapter 2
Robot Mapping

Robot mapping is concerned with developing techniques that enable a mobile robot to construct and maintain a model of its environment based on spatial information gathered over time. Typically, the spatial information stems from directly perceiving the environment through external sensors. In addition, internal sensors like odometry provide information about change of location within the environment. There are, however, many more ways of acquiring spatial information, including external representations such as floor plans, sketches, or written descriptions, as well as direct communication with other robots or with humans.

Most approaches to robot mapping are *incremental* in the sense that a new observation is used to adjust the current spatial model, leading to a new model. The observation is then discarded. The adjustment of the model when new information about the environment becomes available can be seen as a two-step process: First, in the *localization step* corresponding features contained in the new information and in the current model are identified (*data association*) and the robot's position within the model is updated based on the found correspondences (*position update*). Second, in the *map merging* step the spatial information in the model is complemented and updated based on the results of the localization step. The information flow of this general incremental mapping cycle is illustrated in Fig. 2.1. The current spatial model serves as input for the data association and map merging steps and is modified by the map merging step.

Depending on the nature and source of the new information, one of the two suboperations of localization, data association or position update, or the map merging step can be missing. For instance, the robot's internal odometry sensors provide information about the robot's position but not about environmental features. Thus, no data association and no map merging take place. Communicated information on the other hand may or may not provide any evidence about the robot's current position. In the first case, we will speak about *position-related communication*—an example could be information in the form of a you-are-here map—and the information flow would be similar to that of standard incremental mapping. An example of the second case,

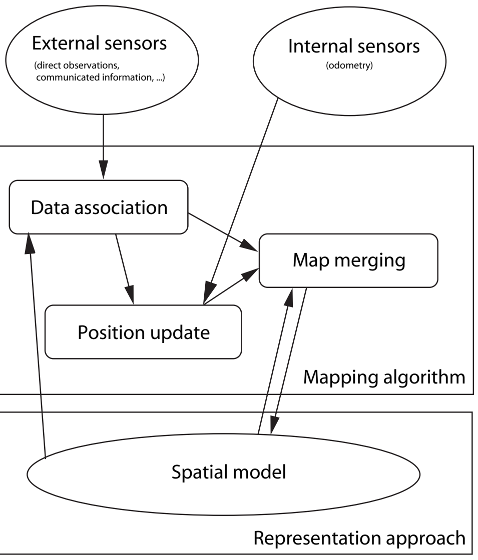


Figure 2.1: Incremental mapping information flow

which we will call *position-unrelated communication*, could be a floor plan which provides a lot of environmental information but no information about the robot's current position. In this case, the data association step would immediately be followed by the map merging step without a position update in between.

Robot mapping is a challenging problem because of the uncertainty inherent in the available spatial information and in the model itself, which always is an approximation of the real world. While it is comparatively easy to localize the robot given an accurate model of the environment (localization with an a priori map) or to generate an accurate model if the exact location is known for every perceived sensor measurement (mapping with known position), the combined problem is very hard because errors in the model and the localization do affect each other. As a consequence, the errors can grow without bounds. The robot mapping problem is therefore also referred to as the *simultaneous localization and mapping (SLAM) problem* (Leonard & Durrant-Whyte, 1991). It is often considered as a state estimation problem and tackled by sophisticated stochastic methods for dealing with the uncertainty, resulting in a research field commonly known as *probabilistic robotics* (Thrun, 2000; Thrun et al., 2005). We provide an overview of the main techniques developed in this field in Appendix A.

Comparing different approaches to the robot mapping problem on a theoretical level is a rather difficult endeavor. The main reason for this is the huge variety of ways in which space can be represented without clear criteria for what makes a good or even optimal representation. As argued in the introduction, over the last several years the robot mapping community has focused on approaching the robot mapping problem from the uncertainty handling perspective. Sophisticated techniques for maintaining probability distributions over high-dimensional state spaces have been developed. However, as a consequence the SLAM problem has mainly been reduced to finding algorithms that compute the most likely model from a sequence of actions and direct sensor observations for an arbitrarily predetermined representation approach.

For the most part this representation approach falls into one of three main classes: occupancy grids, geometric maps, or landmark-based maps. All three kinds of representations describe the environment by providing precise locations of features within a single global coordinate system. This raises the interesting question about why these approaches differ so much from what has been discovered about human cognitive maps (Golledge, 1999). For instance, it is known that human spatial knowledge is fragmented, incomplete, and distorted (Downs & Stea, 1973; Tversky, 1993) as well as hierarchically organized (Briggs, 1973; Hirtle & Jonides, 1985; McNamara, 1986).

In this book, we aim for a more general view on what makes a good or optimal robot mapping approach. Acquiring a spatial model is not regarded as a task per se, but considered in the context of a larger set of competences that are supposed to be realized with the help of the model. Hence, choosing a suitable representation approach becomes part of the problem, and different approaches will be assessed based on how well they support the operations required for different spatial tasks.

In the following, we will proceed by looking more closely at the most important competences that rely on a spatial model of the environment. Based on the set of tasks suggested there, we derive a set of criteria for comparing and evaluating different approaches to the robot mapping problem. Subsequently, we review the robot mapping literature with respect to the spatial representations employed as well as the uncertainty handling methods utilized, and make use of the identified evaluation criteria to compare the different approaches. As a result of this analysis, we will argue that certain promising directions of research have not yet been sufficiently explored. This insight serves as a motivation for the main part of this work, which pursues research in this direction.

In the remainder of this chapter, we will use the term *(spatial) model* or sometimes *map* for the entire spatial information a robot has stored about its environment at a given moment in time. Depending on the considered approach, this can include uncertainty information and different simultaneously considered alternatives. To talk about a single alternative, we will use the term *hypothesis*. *(Spatial) representation (approach)* will refer to the representation formalism used to formulate spatial knowledge in the model. We will further make the distinction between *basic representation approaches*—elementary homogeneous representations—and *organizational forms*—more complex structured representation formalisms that combine different basic representation approaches. Finally, *mapping approach* is used in general to refer to the combination of a spatial representation approach, uncertainty handling techniques, and algorithms employed in a particular implementation.

2.1 A Spatial Model for What?

The problem of acquiring and maintaining a long-term spatial model of an environment cannot be studied in isolation but has to be considered in the context of all the operations that are to be supported by the model. Although the concrete set of operations clearly depends on the specific area of application, we believe that it is reasonable to regard three tasks as fundamental for future mobile robot applications: the tasks of (1) navigation, (2) systematic exploration, and (3) communication about space. We will consider these tasks further below. Other forms of acting in space (e.g., manipulating a set of objects) while being based on spatial properties do not require a long-term model of the environment. Instead, they can be explained as being guided by immediate perception or based on a short-term local spatial model of the agent's proximity. Therefore, we will not consider such tasks in our discussion of environmental models here.

2.1.1 Navigation

Montello (2005, p. 258) defines navigation as a "coordinated and goal-directed movement of one's self (one's body) through the environment." The purpose of navigation is to move oneself in order to reach a particular (known) location. Levitt & Lawton (1990) list three fundamental questions that need to be answered in order to achieve successful navigation: "Where am I?", "Where are other places relative to me?", and "How do I get to other places from here?". Thus, it follows that a spatial representation needs to contain information that supports the ability to localize oneself with respect to environmental features, information about the locations of relevant places, and information that allows us to generate plans on how to reach these relevant places.

Research on animal and human navigation has identified a multitude of navigation behaviors and techniques that contribute to achieving overall successful navigation from low-level guidance-based behaviors over place recognition-triggered responses to higher-level topological and metrical navigation abilities (cf. Trullier et al., 1997). In work on human navigation it is common to distinguish between *locomotion* and *wayfinding* (Darken et al., 1999; Montello, 2005). Locomotion comprises sensor-based movement through the immediate surroundings while avoiding obstacles. Wayfinding comprises all the higher-level cognitive processes involved in navigation like localization and planning. While locomotion can be achieved reactively, wayfinding comprises those abilities that rely on spatial information stored in memory.

From the perspective of operations and spatial competences, we can thus distinguish three main subtasks under the term navigation: *localization*, *path planning*, and *locomotion*. Of these, only the first two require a global spatial model of the environment.

2.1.1.1 Localization

To determine one's location within a model of the environment can mean different things depending on the level of abstraction employed in the model. In an accurate geometric model, it would mean to determine the exact position and orientation of the agent within the global coordinate system used, usually referred to as the robot's *pose*. For a more abstract representation, it could mean, for instance, to establish that one is currently on the town hall square facing north.

In general, it gets easier to localize oneself when more information is included in the model. This is because more ambiguities arise when certain kind of information is not included or details are omitted, leading to increased *perceptual aliasing*. On the other hand, a large amount of detailed information may very well lead to unacceptable space consumption or processing times. However, it is also possible that information required for localization is not distributed homogeneously but mainly provided at certain *distinctive places* (Kuipers, 2000), while navigation between these distinct places is not based on location.

In robotics, three degrees of localization capabilities are distinguished: *Local localization* (or *position tracking*) refers to the ability of tracking your position correctly when your starting position is known. The problem of *global localization* means that the robot should be able to localize when starting at an unknown position. Finally, the term *kidnapped robot problem* (Engelson & McDermott, 1992) is used when the robot is supposed to recover from false localization: The robot wrongly assumes it is at a particular place but is able notice its error and subsequently determines its correct position.

2.1.1.2 Path Planning

Path planning is based on information about how places are connected with each other and about the possibilities to move from one place to another. Hence, it depends on the environment as well as the motion abilities of the robot. Possible connections can be either represented directly, in which case path planning becomes the problem of determining an optimal sequence of connections with respect to a given optimality criterion, or derived based on a description of the traversable space (the *free space*) and the agent's motion abilities.

The resulting plan is a description of how to get to the goal location. This can be specified in many different ways and at different levels of abstraction. It can, for instance, be a detailed sequence of motions to perform, a reactive behavior to execute, or an abstract route description containing references to places and environmental features. In all cases, the plan has to be in line with the motion and sensor abilities of the agent in order to be executable.

2.1.2 Systematic Exploration

Exploration is sometimes seen as a part of navigation, namely when an agent is supposed to find a certain goal location in an unfamiliar environment. This tasks requires a systematic exploration approach to be efficient. In robotics, however, exploration is typically not goal-directed in the sense of reaching a particular goal location. Rather, the purpose is to systematically cover all accessible parts of an environment, either to construct a spatial model for later use or to systematically search or cover the entire environment as in rescue or cleaning scenarios.

The most important aspect of exploration is to keep track of covered parts and parts of the environment that still need to be explored in order to attain complete coverage. In addition, it is often desirable to perform the exploration as efficiently as possible in terms of time or travel distance. In this case, information that supports the decision of which part to explore next needs to be provided by the representation as well.

2.1.3 Communication

Communication about space with other agents can take many forms. First, we can distinguish direct and indirect communication. In the first case, information is directly passed between the agents, typically in verbal form, possibly supported by gestures. In contrast, indirect communication makes use of external representations. Examples are sketches, all kinds of maps (such as topographical maps, you-are-here maps, or floor plans), and written descriptions. The external representation is usually either graphical or verbal.

In a second distinction, one could look at the purpose of the communicated information. Here we could distinguish communication with the goal of conveying information about the environment (like a city map or a verbal description of the city center) from communication with the goal of providing navigational information like route instructions supposed to guide you to a particular place.

Finally, as seen in the beginning of this chapter, communicated spatial information can be position-related or unrelated, depending on whether or not it is adapted to the receiving agent's current position. For example, a city map is position-unrelated as the agent first has to localize within the map in order to use it for navigation, while route directions ("take the third street on the right, then ...") are typically position-related based on the current location of the conversational partner.

In order to communicate with agents that have completely different sensor and motion capabilities, a suitable level of representation needs to be established on which spatial information can be exchanged. To facilitate this, an agent's spatial representation needs to bridge from low-level representations tuned for its own technical abilities to more abstract levels of representation that other types of agents can utilize (Wolter & Richter, 2004).

For communication with humans, a semantic level of description that makes use of relevant human spatial concepts needs to be part of the representation, and the representation needs to be able to mediate between this semantic information and the more sensor-related levels. Anchoring semantic information in the spatial representation has been investigated by Chatila & Laumond (1985) and Galindo et al. (2005).

2.2 Correctness, Consistency, and Criteria for Evaluating Spatial Representations

As mentioned previously, it is not an easy endeavor to compare existing mapping approaches in order to decide which one is most suitable under particular conditions. Assuming that at one point the robot has to make a decision for a single hypothesis, the ultimate goal of the involved mapping algorithms is to derive what we could intuitively call the correct model of the environment from the available information (sensor reading, actions performed, communicated information, background knowledge, etc.), the model that best describes the real environment within the approximation bounds predefined by the chosen representation approach.

However, taking into account the uncertainty of spatial information, we cannot expect that a robot mapping system always comes up with the correct solution. The available information may actually suggest a different state of the environment. Therefore, the goal can only be to compute the most likely or plausible model given the available evidence. Nevertheless, to evaluate the quality of a mapping approach, we would still be interested in its ability to determine the correct model given enough time to gather sufficient data.

Depending on the spatial representation approach chosen, it is common to replace correctness as introduced above by the slightly weaker criterion of the resulting model being *consistent* in the sense that certain crucial spatial properties are represented correctly while other less important ones may not be correct. For instance, again using the example of a geometric model, small discrepancies in the obstacle boundaries can be perfectly acceptable without diminishing the model's usability, while a model in which neighboring rooms overlap would be considered as inconsistent. In the same way, a graph-like representation for a street network with coordinates assigned to the nodes would be considered consistent as long as the graph correctly reflects the topology of the network even if the coordinates do not reproduce the positions of the junctions exactly. Usually, the criterion of consistency is only used intuitively in the robot mapping literature, without giving concrete conditions that a consistent model has to satisfy.

With regard to the spatial representations used in different mapping systems, we propose three general criteria to evaluate and compare different approaches: (1) extractability and maintainability, (2) information adequacy, and (3) efficiency and scalability. We are going to discuss these criteria in the following.

2.2.1 Extractability and Maintainability

It is crucial that a spatial model formulated within the chosen spatial representation approach can be constructed and maintained from the information available to the robot. This means we need to be able to formulate algorithms that take the input information and update the model accordingly. This requires managing the uncertainty in the input information.

Generally, the information required by approaches that model the environment in a low-level sensor-near way seems to be much easier to extract and maintain because no sophisticated processing of the sensor data is needed. The effects of imperfect sensors can be explicitly modeled statistically. On the other hand, more abstract representations may have the advantage that the modeled relations can be more reliably derived from the sensor data, e.g., we might be able to reliably tell that a certain object is between two other objects, while it is much harder to determine the exact shape and location of an object.

Extractability and maintainability of representation approaches can also vary significantly for different kinds of environments. For instance, an approach may only be suitable for well-structured indoor environments or rely on artificial unique landmarks. We will call representation approaches that can adequately represent arbitrary environments *universal*.

2.2.2 Information Adequacy

Similarly crucial is that the chosen representation approach provides all the information that is required for the operations that are supposed to work on the model. The information can be represented either explicitly, meaning that it can be retrieved without further computation, or implicitly, for instance, when the distance between two objects is computed via their coordinates (see Palmer, 1978, for a discussion of implicit and explicit representations). The costs for accessing information implicitly stored in the representation naturally can vary significantly.

Another aspect of information adequacy is that the level of detail is sufficient to support the regarded operations. However, demand for low computational costs and low space consumption (see below) warrants a certain pursuit of sparseness: A representation should not contain superfluous information, information that is required neither for any of the operations nor for maintaining the model itself, and required information should only be represented at a level of detail that is really needed.

2.2.3 Efficiency and Scalability

The ways in which a certain kind of spatial information can be represented are limitless. As a mobile robot typically is supposed to work in realtime, the operations should be as efficient as possible. Efficiency significantly depends on the way the information is represented. Typically, there are trade-offs involved in which one way

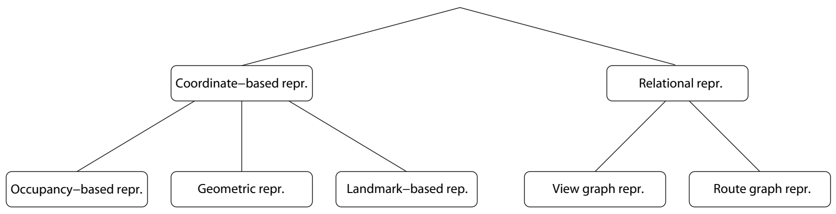


Figure 2.2: Taxonomy of spatial representation formalisms used for mobile robots

of representing things favors a certain operation while making other operations more expensive. A good spatial representation, therefore, would be one which optimizes the overall performance over all operations working on the spatial model, which is rather difficult to assess. However, looking at the complete set of operations, and not only at the efficiency of map construction and map maintenance, will give a much better picture.

In addition to the efficiency aspect, we will use the term *scalability* to discuss how well a representation approach scales with the size of the represented environment. This concerns efficiency of operations as well as space consumption.

2.3 Spatial Representation and Organization

In the following, we review the literature on robot mapping regarding the spatial representation approaches employed. We distinguish between different *basic spatial representation approaches*, which are the elementary representation formalisms to describe an environment in a homogeneous way, and different *organizational forms*, which describe different ways of combining basic spatial representation approaches to form more complex representation structures. Such representations are often referred to as *hybrid representations* (Buschka & Saffiotti, 2004).

2.3.1 Basic Spatial Representation Approaches

Basic spatial representation formalisms mainly differ in two ways: first, in the *basic entities* used to formulate knowledge about the environment, and second, in the way how configurations of the the basic entities are expressed in terms of *spatial relations* holding between the entities.[1] In order to structure this overview on current basic representation formalisms, we use the taxonomy depicted in Fig. 2.2.

At the top level of the taxonomy, approaches are classified based on the way the spatial relations between the basic entities are represented, leading to two main classes: *coordinate-based representations* (often broadly referred to as *metric representations*

[1]As we will see later in this chapter, spatial relations can sometimes be represented rather indirectly in the form of action sequences or motion behaviors.

in the literature) and *relational representations* (comprising, among others, approaches traditionally referred to as *topological maps*). The distinction made here is the following:

Definition 2.1 (Coordinate-based representation)**.** *Coordinate-based representations express spatial relations between basic entities implicitly by providing coordinates for each of the spatial objects within a single absolute coordinate system.*

Definition 2.2 (Relational representation)**.** *Relational representations express spatial relations between basic entities by explicitly stating that a certain relation holds between a certain set of objects.*

The consequences of these two fundamentally different ways to describe configurations will be discussed further below. On the second level of the taxonomy, a distinction is made based on the kind of basic spatial objects used, leading to three subclasses of coordinate-based representations (*occupancy-based representations*, *geometric representations*, and *landmark-based representations*), and two subclasses of relational representations (*view graph representations* and *route graph representations*).

2.3.2 Coordinate-Based Representations

The defining property of coordinate-based representations is that configurations between basic entities are described implicitly through coordinates within a single absolute coordinate system. As a consequence, it is not possible within these approaches to leave the relations between certain entities completely unspecified and thus distinguish between actually perceived relations and derived relations. This leads to the problem that coordinate information needs to be as exact as possible or else global inconsistencies may occur.

On the other hand, if many different spatial relations are required by the operations, coordinate-based representations offer a universal basis from which many different kinds of relations can be derived (e.g., distance, angles, adjacency).

The three most common kinds of basic entities used in coordinate-based representations are cells (used in occupancy-based representations), geometric objects, and landmarks extracted from the sensor information.

2.3.2.1 Occupancy-Based Representations

Occupancy-based representations represent occupied and free parts of space equitably by decomposing space into cells and storing for each cell whether it is (at least partially) occupied or (entirely) free. Typically, the decomposition is independent of the distribution of objects in space and uniform in the sense that all cells have the same shape and size. The dominant decomposition approach employed in robot mapping is the grid map, in which (in the 2D case) a square-shaped raster is used and which allows a simple mapping of locations in the world given in the form of coordinates in a global

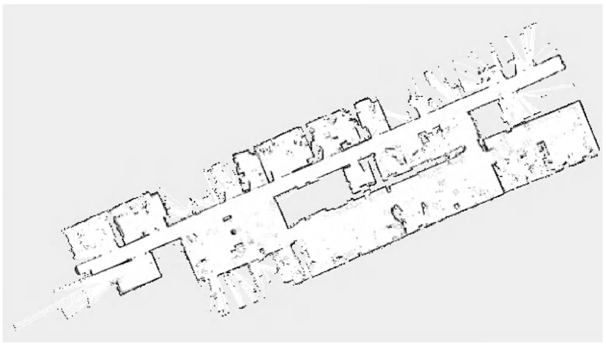


Figure 2.3: An occupancy grid representation. The cells' likelihood of being occupied is represented by their gray values, ranging from white (unoccupied) to black (occupied)

coordinate system to the indices of the corresponding cell in the grid and vice versa. Typically, instead of a binary grid a so-called *occupancy grid* is employed in which the uncertainty about the occupancy state of a cell is represented. In the majority of cases, a likelihood value $l \in [0, 1]$ is maintained for each cell. Figure 2.3 shows an occupancy grid of an indoor environment.

Moravec and Elfes (Elfes, 1989; Moravec & Elfes, 1985) were the first to propose the use of occupancy grid representations for mobile robot navigation and world modeling. They also formulated Bayesian update procedures based on a probabilistic sensor model. Since then, these techniques for learning occupancy grid representations have been refined and combined with advanced probabilistic uncertainty handling methods. Occupancy grids are now used in many state-of-the-art mapping systems (Grisetti et al., 2007a,b; Hähnel et al., 2003a; Thrun, 1998). The idea of occupancy grids has also been adopted to model 3D space (see, for instance, Moravec, 1996).

An extension of occupancy grids called *coverage maps* has been proposed by Stachniss & Burgard (2003a,b) together with suitable probabilistic map update methods. In coverage maps, the degree of coverage is represented as a probability distribution over the interval 0 (completely empty) to 1 (completely occupied) for each cell, resulting in a more accurate description of the environment.

Occupancy grids provide a detailed description of the environment in a sensor-near way, which does not require the extraction of higher entities from the sensor data. That makes it comparatively easy to model the propagation of uncertainty and to develop construction and maintenance methods because no explicit data association is required and incorporating observations (merging step) only involves updating the likelihood values of particular cells. In addition, an occupancy grid representation can slowly adapt to changes in moderately dynamic environments and is universal in the sense that any environment can be adequately modeled.

As an occupancy-based representation preserves most of the spatial information contained in the sensor data, it in principle provides all the information required for navigation and communication. In addition, the high level of detail allows for accurate localization. However, the low-level nature of the represented information without explicit modeling of obstacle boundaries makes many operations rather costly. Path planning can for instance be achieved by value iteration (Howard, 1960; Thrun et al., 1998a), but the search space is rather large compared to other representations. The absence of high-level information or objects complicates the annotation with or the derivation of semantic information as required for high-level reasoning or communication with humans.

With regard to systematic exploration, several techniques have been developed for occupancy-based representations ranging from simply counting the number of times a cell has been scanned, to the identification of so-called *frontiers* between the observed and unobserved areas Yamauchi (1997), to decision-theoretic approaches based on expected information gain (Bourgault et al., 2002).

The main drawback of occupancy-based representations is that they do not scale well to large environments. The high space consumption for larger environments resulting from the fact that the required space depends on the size of the environment and not on the complexity of the environment directly implies strongly increasing computational costs as well. To adequately capture the details in more complex areas a high cell resolution is required which is wasted in less complex areas. Techniques like quad- or octrees (Samet, 1988; Zelinsky, 1992) have been employed to reduce the space consumption problem but can also lead to increased computational costs.

2.3.2.2 Geometric Representations

Geometric representations use parameterized primitive geometric objects, i.e., points, lines, curves, planes, etc. For these objects, we will adopt the term *geom* here. A geometric representation basically consists of a list of geoms describing the boundaries of free space located in a single coordinate system. Most approaches used for mobile robots employ a single kind of geom. Figure 2.4 provides an example of a line-based 2D representation.

Chatila & Laumond (1985) describe an early geometric 2D representation used as part of the world model of their robot HILARE. The representation consists of a set of polygonal objects directly derived from sensor data.

In Crowley (1989), an early approach to construct a model consisting of line segments is described. The line segments are extracted from sonar range data while the robot moves around. Every time a line segment has been detected, it is matched to the model. If a suitable match is found, its parameters are updated accordingly. Otherwise a new line segment is added to the model. The approach is only employed to construct small local models of the robot's immediate surroundings though. In Tardós et al. (2002), techniques for computing a geometric map consisting of point objects (corners) and line objects (walls) are developed.

A lot of work from the area of scan matching has been concerned with computing

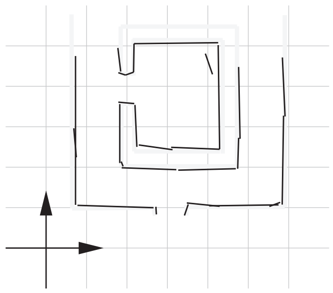


Figure 2.4: Example of a geometric representation: The boundaries of obstacles are represented by line segments

point set 2D representations from laser range data. For instance, Lu & Milios (1997) use a spring-based energy minimization approach to align the individual scans in order to derive a consistent representation. Several groups have developed similar techniques for constructing complete geometric 3D models. Nüchter et al. (2004, 2005) use scan matching of 3D scans together with a global relaxation method to compute a point set representation for complete six degrees of freedom robot motion. A data reduction technique is employed to decrease the number of points from each scan.

An approach for constructing a geometric 3D model consisting of planar surfaces is described in Hähnel et al. (2003b). The approach is based on scan matching to compute a raw 3D model and uses a local search procedure to generate a low-complexity plane model. A different approach described in Liu et al. (2001) focuses on extracting a compact 3D model from a given set of point measurements in 3D space.

Wolter, Latecki, and colleagues (Latecki et al., 2005a,b; Wolter et al., 2004) describe techniques based on shape matching to construct a geometric representation consisting of (generalized) polylines. They employ shape similarity measures to match polylines and complete scans and develop techniques to merge polylines when a new scan is incorporated into the map.

Like occupancy-based representations, geometric representations can represent arbitrary environments as long as geometric primitives are employed that allow us to approximate the shapes of object boundaries sufficiently well (e.g., points, lines, polylines). They are typically much more compact (depending on the geoms used), at least if a merging method is employed to join corresponding entities instead of blindly adding all perceived objects as new objects. However, developing adequate merging schemes is a hard problem and for numerous approaches no merging scheme is described.

As geometric representations describe the boundaries of free space, they are, on the one hand, well-suited for localization and, on the other hand, also allow for path planning. Path planning, however, usually requires the construction of a discrete search space from the representation in order to apply path planning techniques; this produces

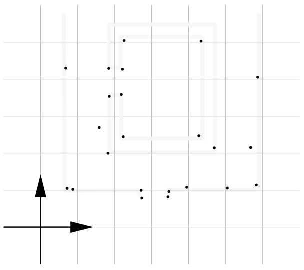


Figure 2.5: Landmark-based representation consisting of point landmarks referenced in a global coordinate system

additional computational costs. Typical approaches here are roadmap approaches, cell decomposition approaches, and potential field approaches (for an overview, see Latombe, 1991).

One disadvantage of geometric representations is that they do not lend themselves very well to systematic exploration: Keeping track of which parts of the environment have been covered requires that the perceived area be explicitly represented and that this description be continuously updated.

With regard to communication, approaches using more complex geometric primitives (e.g., polygons) are usually much better suited than those based simply on points or lines as they allow for describing complete objects which can then be used to attach semantic information. Overall, only certain types of communication can be directly achieved with geometric representations.

2.3.2.3 Landmark-Based Representations

Landmark-based representations represent the world as a set of salient objects (the *landmarks*) extracted from the sensor data. The positions of the landmarks are specified in a global coordinate system (see Fig. 2.5). Both, geometric and landmark-based representations have been subsumed under the term *feature-based representations*, mainly because they can be similarly stored as matrices and lend themselves to the same kind of uncertainty handling techniques.[2] However, from the spatial representation perspective, the distinction makes sense because geometric representations completely model the boundaries of free space, while landmark-based representations focus on specific objects useful for localization and orientation. Besides the positions of the landmarks, additional attributes can be used to discriminate similar landmarks in the data association step.

In the literature, we mainly find simple point-like landmarks that are easy to extract from camera data or range data, or artificial beacons often with a unique ID allowing for unambiguous identification.

[2]The overall approach is also often referred to as *stochastic maps*.

An example for using natural landmarks is the work by Guivant & Nebot (2001). They use tree trunks extracted from laser range data as landmarks to create a 2D map of the Victoria Park in Sydney. Their data set has been used as a benchmark for landmark-based mapping approaches by many other groups (e.g., Montemerlo & Thrun, 2003; Montemerlo et al., 2003; Nieto et al., 2003).

Neira & Tardós (2001) use vision to extract vertical edges corresponding to corners and wall or window frames and project them onto the ground plane to map an indoor environment. In Dissanayake et al. (2001), radar data is employed to map a combination of natural and artificial (radar reflectors) point-like landmarks. The quality of a landmark candidate is assessed by checking its behavior as a stationary point landmark; only stable candidates are incorporated.

Purely artificial landmarks are used, for instance, in Frese (2006b), where indistinguishable landmarks detectable by vision are placed on the floor along the walls. Identification of the landmarks here is based only on the relative positions, taking into account larger constellations of landmarks for global loop closing.

Landmark-based approaches have also been successfully employed in the underwater domain using artificial landmarks in the form of transponders as well as natural features extracted from sonar data (see, for instance, Newman & Leonard, 2003; Williams et al., 2001).

Landmark-based representations are rather compact (depending on the density of landmarks in the environment) and in general scale well to larger environments. They have been extensively used in SLAM approaches, mainly because they lend themselves very well to probabilistic uncertainty handling. Sophisticated techniques for maintaining these kinds of representations in the presence of uncertainty have been developed. Unfortunately, the developed approaches typically lack the universality of occupancy-based or geometric approaches as they are only applicable in environments that provide the required landmarks in sufficient density.

Moreover, landmark-based approaches do not represent the boundaries of free space. This makes them less suitable for path planning or systematic exploration. The only kinds of environments in which this is not problematic are open environments, in which the navigation between the landmarks is not much further restricted by obstacles other than the landmarks themselves. How well landmark-based representations support localization depends to a large degree on the ambiguity of landmarks and their density in the environment. For instance, if the environment is densely covered by landmarks which are easy to distinguish, the localization problem is simplified significantly.

Landmarks play an important role in human wayfinding (Denis, 1997; Lovelace et al., 1999; Sorrows & Hirtle, 1999). Thus, they are well suited to support communication with humans. For instance, they can be used for providing and processing route instructions if it is possible to enable the robot to recognize a set of landmarks similar to those that humans use.

Overall, landmark-based representations are in many cases not adequate as exclusive representations for a mobile robot, but they are well suited to be combined with other approaches.

2.3.3 Relational Representations

Relational representations explicitly enumerate relations that hold between objects. As a result, this allows us to express ignorance because the fact that a relation is not listed means that this relation may or may not hold. In addition, the relational approach allows us to keep track of which relations have been directly perceived and thus are reliable, and what can be deduced from these relations. Moreover, as relational representations usually deal with more abstract spatial relations, ensuring consistency is somewhat simplified. In some approaches the spatial relations are provided indirectly in an action-based form by specifying a sequence of movements or motion behaviors that will move the agent between the locations.

Most relational representations employed in current mobile robots are graph-based representations in which the nodes stand for the basic entities (e.g., views, places, objects) and the edges represent the relevant spatial relations (e.g., adjacency or connectivity). They are often referred to as *topological maps*. We distinguish two kinds of relational graph representations: *view graph representations* and *route graph representations*. In view graph representations the nodes are not directly derived from the environment but are more or less evenly distributed over the free space. Each node is characterized by the view that is available from this particular position. The placement of nodes is based on sufficient perceptual difference with adjacent nodes. In contrast, in route graph representations the nodes are directly induced by the environment. A route graph represents the environment as a network of distinct routes and the nodes stand for distinctive places or particular landmarks encountered along the routes. Often, the route graph representation directly reflects the topology of the free space.

In principle, relational representations may consist of purely propositional statements formulated in a logic-based language. But as graph-based implementations have significant computational advantages because of their analogical nature, this approach is rarely used. However, relational approaches containing graph representations have successfully been embedded into logical frameworks (see for instance Remolina & Kuipers, 2004).

2.3.3.1 View Graph Representations

In view graph representations the nodes are directly associated with the particular sensor input, called a *view*, available at a particular location. A link between two nodes accounts for the fact that both views have been seen in consecutive order, and thus the spatial adjacency of the corresponding locations. The link provides the information the robot requires to move between the two locations. Nodes have to be distributed densely enough so that reliable locomotion between them is possible based on the navigational capabilities of the robot. Construction of the view graph representation

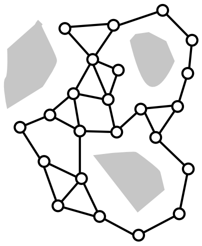


Figure 2.6: View graph representation embedded in the environment

thus means creating a network of nodes that covers the entire environment. Figure 2.6 depicts a view graph embedded within the environment it represents (nodes are located at the positions from which the views have been recorded).

Schölkopf and Mallot introduced the notion of a view graph (Schölkopf & Mallot, 1995) as a kind of minimal model that enables navigation and path planning. They describe a neural network approach for learning the view graph representation from a sequence of views and for using the view graph for navigation in a discrete maze-like world. In later work (Franz et al., 1998), this approach is extended to open environments and implemented on a real robot. The views are given by panorama images from a 360° camera and called *snapshots*. Navigation between views is achieved by a visual homing procedure. Adjacent nodes in the view graph have to be close enough together so that the individual "catchment areas" of the homing procedure overlap. Due to views being 360° images and treated independently of the robot's orientation, and the fact that only sufficiently distinctive views are stored in the graph, it is possible to associate views with particular locations and visualize the view graph as embedded in the environment. In Hübner & Mallot (2007), an extension of the view graph approach involving global position estimates for the nodes is described. Strictly speaking, this approach has to be classified as a hybrid approach combining coordinate-based and relational descriptions.

The ELDEN system described in Yamauchi & Beer (1996) uses a topological map representation that is adaptive, as the edges are annotated with confidence values. Thus, it can to a certain degree deal with changes in the environment such as moved objects, which cause changes in the topology of free space. The places are distributed throughout the environment based on distance from other already established places: If the robot is more than a certain distance away from the last visited place, a new place node is created. Nodes represent regions of space. They are annotated with the global coordinates of their centers and a local occupancy grid representing the geometry of the place. Matching of the grids is used for localization and for hill-climbing to the center of the region. The edges are annotated with the heading information that is updated each time the edge is traversed. The system relies on accurate dead reckoning and has only been evaluated for small environments.

Duckett & Nehmzow (1999a,b) report on experiments with a view graph repre-

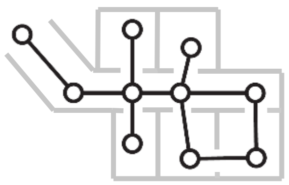


Figure 2.7: Route graph representation of an indoor environment

sentation that is very similar to that of Yamauchi and Beer. The system described in Duckett & Nehmzow (1999a) learns the topological map from sonar, infrared sensors, and compass readings. The nodes are annotated with local range data derived from the sonar measurements. Relative distance and direction information is stored at the edges. A neural network detects open areas and stores them as predicted places for further exploration. For localization, the approach requires global coordinate estimates, which are computed using a spring-based relaxation approach.

View graph approaches are representations very close to the actual sensorimotor experience of the agent but abstract the continuous world into a discrete representation. As such they are rather universally applicable as long as the sensor information is rich enough so that perceptual aliasing does not become a problem. Also, complementing the representation (merging step) is straightforward, and the data association problem is at least reduced.

View graphs provide the information for successful navigation, but the resulting paths tend to be suboptimal, especially when exact homing is required after every step. Their scalability depends on the density of nodes required. The downside of the representation is that almost no structural information about the environment is represented and that the model varies depending on the sensor and motion abilities of the agent and depending on the starting position. As a consequence, the representation is not very well suited for systematic exploration and communication.

2.3.3.2 Route Graph Representations

The route graph concept has been introduced in Werner et al. (2000) as a general model for environmental knowledge gained by integrating route information into survey knowledge by humans and animals, and also by artificial agents. We adopt it here for graph representations in which the nodes stand for distinctive places induced by the environment. The edges reflect distinctive paths connecting these places, allowing travel from one place to another. Figure 2.7, for instance, shows a route graph for an indoor environment in which the nodes correspond to rooms and junctions and the edges correspond to doorways and hallways. Route knowledge is generally assumed to play an important role in the development of human representations of large-scale space (Siegel & White, 1975).

The TOUR model by Kuipers (1978), which is proposed as a psychological model of human common-sense knowledge of large-scale space, describes a representation of a street network environment consisting of places, paths, and regions. It can be learned from abstract route descriptions consisting of sequences of turn and move actions. The place and path descriptions represent the environment as a route graph.

In Kuipers & Byun (1988), these ideas have been transferred to the domain of mobile robots operating in indoor environments. Reactive control procedures for hill-climbing to locally distinctive places (represented by the graph nodes) in the environment or for moving through the environment form the basis for abstracting the environment as a discrete graph structure. The edges stand for control procedures that need to be executed in order to move the robot between the nodes connected by the edge (e.g., following the midline of a corridor). The applicability of particular hill-climbing or control procedures is derived from the properties of the immediate surroundings of the robot. The approach explicitly allows for storing local (geo)metric information for the nodes and edges. An exploration agenda is kept listing nodes with directions that still need to be explored. Localization is based on matching the local description of places and employing a topological *rehearsal procedure* that validates the hypothesis that two places correspond by comparing the results of traveling to neighboring nodes.

These ideas have been further elaborated and refined within Kuipers's framework of the *Spatial Semantic Hierarchy* (SSH) (Kuipers, 2000; Kuipers & Byun, 1991; Kuipers & Levitt, 1988; Remolina & Kuipers, 2004). The SSH contains a topological level of representation that is derived from a sequence of views and actions. Recent implementations of the topological level of the SSH (Kuipers et al., 2004; Modayil et al., 2004) directly employ the Voronoi graphs (see below) of the environment and an extension for open spaces (Beeson et al., 2005) to derive and identify places.

Levitt & Lawton (1990) describe an approach to environmental modeling for open outdoor environments with landmarks. This approach contains a symbolic level of representation which can be interpreted as route graph representation. The landmarks create a natural partition of the environment into regions which can be identified based on the currently visible panorama of landmarks.[3] Adjacent regions share the same panorama with one exception: The orientation of the triangle formed by a random position in the region and the positions of two particular landmarks is reversed. An undirected graph that reflects the adjacency relation is used for qualitative path planning. Navigation between adjacent regions can be achieved by simply crossing the line connecting the two landmarks that make up the difference in the panorama.

Mataric (1992) proposes a distributed map representation that is completely integrated into a system based on the subsumption architecture (Brooks, 1986). The nodes correspond to wall or corridor landmarks observed while exploring the environment using some boundary-following behavior. Nodes experienced consecutively are linked. Localization and path planning are realized by spreading activation within the

[3]An error in this localization approach based on cyclic ordering of landmarks has been pointed out and corrected by Schlieder (1993).

network of nodes, but closing cycles requires at least coarse position estimates for the nodes.

In many approaches, the route network is extracted from the geometry of the environment. One way to achieve this is to follow the idea of retracting free space onto a set of one-dimensional curves called a *roadmap* (Latombe, 1991). One such retraction is provided by the generalized Voronoi diagram (GVD) (Lee & Drysdale III, 1981; Okabe et al., 2000) or the related idea of the medial axis transform (Blum, 1967).

Choset and colleagues (Choset & Nagatani, 2001; Nagatani et al., 1998) directly use the generalized Voronoi graph (GVG), the graph abstraction of the GVD that contains nodes for meet points and edges for the Voronoi curves, as the graph representation of the environment (see Chap. 3 for more details on GVDs and GVGs). The robot navigates and explores the environment using sensor-based control laws that allow for driving along the edges of the GVG. Nagatani & Choset (1999) propose a modified structure called the reduced GVG in which certain kinds of unstable nodes referred to as weak meet points are removed. However, the problem of instabilities is only reduced by this method, and more advanced techniques are needed, as we will discuss in Chap. 4. Choset and colleagues also describe an extension of the GVG into higher dimensional space (> 2 dimensions) called the hierarchical GVG, assuring that the crucial properties of the GVG (especially connectedness) are preserved (Choset & Burdick, 2000; Choset et al., 2000).

An alternative to the retraction approach has been described by Chatila & Laumond (Chatila & Laumond, 1985) and Thrun & colleagues (Thrun, 1998; Thrun et al., 1998a). In both approaches a route graph-like representation is derived from a global coordinate-based representation (a geometric one in the former case and a grid-based one in the latter) by first partitioning free space into regions and then capturing the adjacency relation of the regions in the graph structure.

Obviously, route graph representations are tailored to path planning. The structure directly provides a comparatively small search space that can be searched via standard graph searching techniques like Dijkstra's shortest path algorithm (Dijkstra, 1959) or A* (Hart et al., 1968). In contrast, localization is often mentioned as a problem of route graph representations. As theoretically investigated in Dudek et al. (1991) and Rekleitis et al. (1999), topological rehearsal alone is not sufficient; at least one marker is required to guarantee correct localization and map construction. However, in practice providing additional annotations that can be used to distinguish nodes and edges can improve localization within the graph structure significantly.

One advantage of typical route graph representations is that they directly facilitate systematic exploration. To cover the entire environment it is sufficient to trace all edges of the graph, which can be easily achieved by keeping an account of untraversed edges for each node. Furthermore, a representation of the route network of the environment which explicitly represents decision points is very useful for route-based communication (e.g., generating a route description for or interpreting a route description of a human). The compactness of the representation also makes it a good candidate when

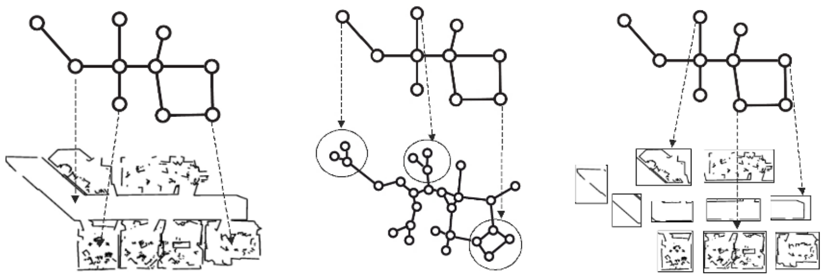


Figure 2.8: Different higher forms of organization: an overlay representation, a hierarchical representation, and a patchworks map

a complete map should be exchanged between multiple agents. In many approaches, places represented by nodes correspond to important concepts such as rooms, which makes it easy to anchor semantic information in the representation.

The main challenge for route graph-based representation approaches is to develop robust methods to construct and maintain the representation. Here, the higher level of abstraction becomes a problem as it is much harder to formulate suitable stochastic models, and the lack of precise localization raises additional problems.

Finally, for route graph representations to be applicable, the environment needs to possess a clear route structure. Open spaces or very unstructured environments do not lend themselves to route graph-based modeling. For structured environments, compactness and the fact that arbitrary additional information can be attached as annotations to the graph structure make route graph representations a promising approach.

2.3.4 Organizational Forms

The previous section provided an overview on basic representation formalisms used for environmental modeling, distinguishing five main classes which show different and often orthogonal strengths and weaknesses, as pointed out by Thrun (1998).

Consequently, people have combined different representation formalisms into more complex forms of organization. We will review these approaches in the following, distinguishing three main forms: *overlays*, *hierarchical representations*, and *patchworks*[4] (see Fig. 2.8). These main forms of combining basic representation approaches can themselves be combined in order to form even more complex forms of representation (cf. Sect. 2.3.4.5).

When reviewing different organizational forms in the following, our main focus is on which strengths and weaknesses are preserved and how the individual representation approaches cooperate. Buschka et al. (Buschka, 2005; Buschka & Saffiotti, 2004) distinguish two modes of cooperation between different representations in a

[4]A classification similar to ours is derived in Buschka (2005).

combined representation which we will adopt here: *Injection* allows us to perform a task within one basic representation based on information from another representation which would otherwise not have been possible to perform (e.g., localization information in a geometric representation is used to localize the robot within a route graph representation). *Synergy*, on the other hand, occurs when two basic representations can be used for the same task but the performance is increased by transferring information between the representations (e.g., localization in the route graph representation alone is possible but the result is improved by incorporating information from localization on the geometrical level).

2.3.4.1 Plain Representation

For completeness, we introduce the term *plain representation* for approaches that only use a single basic representation formalism to describe the environment. Most of the approaches listed above fall into this category, though some of them are actually parts of more complex organizations, as will be pointed out below.

2.3.4.2 Overlays

Overlay representations feature multiple layers of representation. Each layer employs its own representation formalisms and covers the entire environment. The layers are linked in a way that allows us to achieve injection or synergy effects.

Chatila & Laumond (1985) propose an overlay representation consisting of a geometric level and a route graph representation. The route graph representation is actually a hierarchy of graph representations. Its lowest level is directly derived from the geometric representation. Both layers evolve simultaneously while the robot explores the environment.

A similar example is the approach described by Thrun (1998), in which the fundamental representation is a classical occupancy grid used for mapping the environment. Once the environment has been mapped, a topological graph representation is derived from the grid, in which the nodes correspond to regions that form a partition of the free space. Thus, three layers are used in this approach: the occupancy grid representation, a geometric representation describing the regions, and a route graph representation.

Overlay representations typically allow for choosing the best layer of representation to perform a particular operation. This happens at the cost of additional computational efforts to maintain multiple representations and keep not only each layer consistent by itself but also the layers consistent with each other. Typical coordinate-based and graph-based overlays alleviate the problems of the relational representations by improving localization and maintaining the representation either through injection or synergy. However, the disadvantages of the coordinate-based approaches like high space consumption and computational costs typically remain because a global consistent coordinate-based representation still needs to be constructed. Overall, these kinds of overlay approaches tend to not scale well to larger environments.

2.3.4.3 Hierarchical Organization

Hierarchical representations, like overlays, consist of multiple layers, all covering the entire environment. The difference is that here all layers use the same representation formalisms but represent the environment at different levels of granularity. The lowest level provides a rather detailed image of the environment while the higher levels are much coarser or more abstract. In the literature, we can find examples of both, hierarchical coordinate-based representations and hierarchical relational representations.

Fernández & González (1997, 2001) define a general framework for a hierarchical graph-based representation of space called the *AH-graph* (annotated hierarchical graph). The AH-graph consists of multiple graph layers expressing structural information between objects or places. Nodes on higher levels correspond to subgraphs on the level below. Non-structural information like perceptual information about objects or locations or information about the type of structural information described by the edges is stored in form of annotations to the graph structure. Path planning within an AH-graph by hierarchically refining paths and conditions under which this approach yields optimal paths are discussed in Fernández & González (1998).

In Fernández & González (2001) and Fernández-Madrigal & González (2002) this framework is extended to a multi-hierarchical model in which multiple graph-based hierarchies are combined into a single description of the environment. The goal here is to allow for choosing the most appropriate hierarchy for a given task.

In Remolina et al. (1999), the AH-graph model is adopted to describe an extension of the topological level of the SSH, in which places are hierarchically organized into regions. Different approaches for grouping places are discussed, from fully automatic criteria to grouping based on user interaction.

The quad or octree representations mentioned in Sect. 2.3.2.1 can be seen as hierarchical occupancy-based representations of space. However, the tree data structure does not allow for operating exclusively on a chosen level of granularity, as information within one level is only linked via the higher levels. A layered approach as in the graph-based representations listed above would result in very high space requirements, making occupancy-based representations less applicable to hierarchical organization.

Hierarchical organization of information allows us to represent information at different levels of granularity and to either choose the appropriate level for a given task or perform tasks (e.g., path planning) hierarchically by switching to a finer or coarser level as required, increasing the overall efficiency. Especially with regard to communication between heterogeneous agents, this way of bridging from low-level individual levels of representations to more abstract and general levels of representation seems very suitable.

The downside of the hierarchical approach is that additional effort is required to maintain the connections between the layers and keep the individual levels of representation consistent with each other. Also, in most approaches constructing the hierarchy is at least partially based on user interaction. Methods to derive high levels of abstraction autonomously still need to be developed.

2.3.4.4 Patchworks

In patchwork representations, subregions of the environment are represented individually by so-called *local maps* using a single representation formalism, and each using its own frame of reference. The local maps are then related on a higher level to form a global representation. In the literature, the most common forms of patchwork representations are local coordinate-based maps related in a global graph-like relational representation. In addition, we can find approaches that employ local coordinate-based maps related in a global representation which is also coordinate-based.

Local Coordinate-Based and Global Relational Representation. Many authors have proposed representations consisting of local coordinate-based maps linked together on a global topological level. The local maps can be linked either to nodes or edges on the topological level. In some cases, the local maps completely cover the environment, while in others they only describe important areas.

Simhon & Dudek (1998), for instance, employ local geometric maps linked to the nodes of a topological graph representation and discuss how the local maps should be distributed based on a given task (e.g., navigation). The edges linking the nodes on the topological level correspond to control strategies for moving the robot between adjacent local maps.

Yeap & Jefferies (1999) describe a computational model of cognitive mapping in which local line-based geometric maps are constructed, bounded by so-called exits which are identified based on occlusion in the range data. The local maps are disjoint but cover the entire environment. On the topological level, adjacent local maps (those that share at least one exit) are connected by links and links can be enriched with different kinds of information about the connected exits in the local maps. Means for recognizing already visited local maps are discussed but the overall approach relies on a robust detection of the exits independently of the side from which they are approached. This can be hard to achieve in practice, especially in cluttered environments.

In Lisien et al. (2003), a patchwork representation called *Hierarchical Atlas* is proposed which combines local landmark-based maps with a global topological representation. The topological map consists of the reduced generalized Voronoi graph (Nagatani & Choset, 1999) which allows for path planning and systematic exploration. The local landmark representations are attached to the edges of the graph. They support localization within the topological map by disambiguating edges as well as precise localization within the region corresponding to an edge.

Local Coordinate-Based and Global Coordinate-Based Representation. Some mapping approaches describe or can at least be interpreted as using local coordinate-based representations located within a global coordinate system. For instance, scan matching approaches that store scans together with position and orientation estimates of the scans' origins fall into this class (Lu & Milios, 1997; Nüchter et al., 2004, 2005), although often enough the finally intended representation would merge all measurements into a single frame of reference.

Another approach is the work by Chatila & Laumond (1985), in which geometric representations of individual objects, each described within its own reference frame, are related in a global coordinate system. Furthermore, certain kinds of landmark-based mapping approaches employ submapping strategies in order to reduce the computational burden of uncertainty handling (cf. Sect. 2.4.1.2).

A general framework for patchwork approaches is the Atlas framework described in Bosse et al. (2003). The nodes in a topological map stand for local coordinate-based maps (e.g., landmark-based or geometric maps) and edges represent adjacency between the local maps. In addition, the edges represent the coordinate transformation between the local maps. Local maps and global relations between them are both modeled in a stochastic framework. Loop closing is performed based on matching local maps.

Overall, patchwork maps are mainly used to restrict the problem of increasing uncertainty to areas of manageable size, which are then modeled within their own frame of reference, and to reduce computational effort by using a kind of divide-and-conquer approach that avoids simultaneously tracking the relationships between all represented objects. However, the problem of dealing with unbounded uncertainty accumulation still arises when relating the local maps consistently on the global level (e.g., when correctly closing a loop in the global topological map). In many approaches, the space consumption is similar to that of the corresponding plain representation approach as the local maps cover the complete environment.

2.3.4.5 Combining Different Organizational Forms

Aguirre & González (2002) describe a good example of a representation in which different organizational forms are combined into a complex representation structure. The proposed representation can be seen as a patchwork representation of two components. The local maps of the patchwork are overlays of a geometric representation and an occupancy grid, both describing the same local area. The global representation is formed by a two-level hierarchy of graph-based representations. The local maps are attached to the lower level of the hierarchy, which describes the environment at a fine level of granularity, while the top level provides a coarser view of the environment that is used to generate high-level plans. The representation is embedded in a hybrid deliberative-reactive architecture, and the best available representation for the task is chosen based on context, explicitly permitting a certain level of representation not to be available for a certain area. The paper, however, focuses on the application of this complex organization for path planning and execution, and does not discuss global localization in detail. In addition, the approach depends highly on a reliable detection of important environmental features like doors and corridors.

A similar representation can be found in Galindo et al. (2005). A multi-level hierarchy of graph representations is used and local grid maps are attached to the nodes of the bottom level of the hierarchy. In addition, camera images of objects are attached to nodes corresponding to the regions in which the objects have been encountered, and the spatial hierarchy is linked to a semantic hierarchy. This work nicely demonstrates the suitability of this kind of representation to interface with conceptual knowledge, allowing for high-level symbolic reasoning and planning.

Furthermore, the SSH by Kuipers (Kuipers, 2000; Kuipers & Byun, 1991; Kuipers & Levitt, 1988) already mentioned several times in this chapter describes a complex form of organization of multiple spatial representations. As it has been developed as a model of human knowledge and experience of large-scale space, it contains elements like procedural competences and a memory of sensorimotor experiences that go beyond the declarative environmental models discussed here. Nevertheless, the SSH has been realized on mobile robots several times, and organizational forms like patchworks and overlays can be identified.

The SSH distinguishes five levels of spatial knowledge and multiple representations, each using its own ontology. The *sensor level* consists of sensorimotor control laws for leading the agent to locally distinctive states of the environment. At the *causal level*, the continuous experience of moving between distinctive states based on the control laws is abstracted as view-action-view triples. View here stands for the sensory input available at a distinctive state. The *topological level* consists of a graph model that best explains the experience of the causal level. The nodes correspond to places in the environment and can be (hierarchically) grouped into regions, while sequences of edges are grouped to form paths. The nodes may be linked to local coordinate-based maps describing the neighborhood of the place in a patchwork-like manner. The *metrical level* comprises an optional overlay of a coordinate-based representation constructed from the patchwork by merging the local maps into a global frame of reference.

The SSH has been completely axiomatized and inference rules for deriving knowledge from other levels have been formulated within suitable logical frameworks, e.g., for deriving the topological level from the causal level using non-monotonic reasoning (Remolina & Kuipers, 2004).

2.4 Uncertainty Handling Approaches

After looking at the different ways proposed to represent the environment, we will now regard the robot mapping problem from the second perspective mentioned, the perspective of handling uncertainty. It has frequently been pointed out that the sources of uncertainty are numerous: limitations of the sensors and noise in the measurements, imperfect actuation, general unpredictability of dynamic environments, and the fact that models of the environment have to be coarse approximations of the real world in order to be computationally feasible (Thrun, 2000; Thrun et al., 2005). This implies that a robot can never be certain about the current state of the environment.

In the following, we make two distinctions to classify uncertainty handling approaches appearing in the literature: The first distinction concerns the nature of the update process of the spatial representation. Most real-time mapping systems (often called *online approaches*) incorporate new observations *incrementally*: The current representation is updated, leading to a new representation; the decisions made will not be revised anymore in the future. Alternatively, sensor data is not processed sequentially but a (heuristic) search for an optimal solution is performed in which previous decisions can be retracted and the input data is processed in multiple passes. As we are mainly interested in online mapping, we will focus on incremental approaches and only briefly mention the most important multi-pass techniques.

Second, we distinguish approaches based on the number of hypotheses that are being tracked simultaneously in the mapping approach. The three options here are (1) a single hypothesis only, (2) multiple discrete hypotheses, and (3) keeping track of the complete state space at any moment.

2.4.1 Incremental Approaches

In the following review of incremental approaches we pay special attention to the question of which kinds of spatial representations have been combined with which uncertainty handling approach.

2.4.1.1 Single Hypothesis Approaches

Single hypothesis approaches maintain one hypothesis about the state of the environment. Whenever a new observation becomes available, the hypothesis is updated in the most plausible way. In probabilistic approaches, in which at least the uncertainty of some individual facts is represented (e.g., a probability distribution for the current pose of the robot), this means that the most likely model is constructed, and the approach is often referred to as *incremental maximum likelihood*.

Traditionally, relational representation approaches only maintain a single hypothesis about the state of the environment. We can find this approach both in work on view graph representations (Franz et al., 1998; Schölkopf & Mallot, 1995) and in work on route graph representations (Choset & Nagatani, 2001; Kuipers, 2000; Kuipers & Byun, 1991; Kuipers, 1978; Kuipers & Byun, 1988; Kuipers & Levitt, 1988; Nagatani et al., 1998). In all of these approaches, information is represented as facts without any kind of uncertainty assessment. As relational approaches typically focus on qualitative spatial relations that can be more reliably perceived, this is less problematic than for coordinate-based representations. Nevertheless, combining observation and model in the most plausible way can be a difficult task, and often the algorithms may fail to construct the correct model under certain conditions.

Some relational approaches do maintain uncertainty information for at least some individual facts contained in the model. Examples of this are the approach of Yamauchi & Beer (1996), in which confidence values for edges in the graph-based representations are employed; the approaches by Yamauchi & Beer (1996) and Duckett & Nehmzow (1999a,b), which both explicitly represent the uncertainty in the robot position; and the approach described in Hübner & Mallot (2007), which maintains position estimates for the nodes.

Coordinate-based approaches often also only maintain a single map hypothesis. Mostly, this holds for geometric representations in which new observations are matched to and merged into the current map (Crowley, 1989; Wolter et al., 2004). Again, many approaches here at least maintain a probability distribution over the robot's pose space, and sometimes also over individual object parameters.

The advantage of only maintaining a single hypothesis is the simplicity of the approach, resulting in much lower computational cost. The downside is a general brittleness of these approaches when the input data is ambiguous. As incremental approaches do not have the ability to recover from wrong decisions made in the past, a single error in the map update step is typically fatal and results in an unusable spatial model.

To deal more adequately with ambiguous input data, one resorts to approaches that try to maintain a discrete set of spatial models. However, before we discuss these approaches, we turn to the other extreme of approaches that keep track of the complete space of hypotheses by rigorously regarding the mapping problem as a probabilistic state estimation problem.

2.4.1.2 Complete-State-Space Approaches

The common approach taken in probabilistic mapping is to maintain a probability distribution over the space of all different hypotheses and to incrementally update it based on new observations. A prerequisite for this approach is that the perception and actuation processes can be adequately modeled stochastically. The theoretical basis for updating the probability distribution is then given by the recursive Bayes filter (see Appendix A for the technical details of probabilistic mapping).

In most realistic scenarios, however, it is infeasible to represent and compute the probability distribution exactly. Therefore, approximations have to be used. During the last two decades, very powerful approximation techniques have been developed or adopted from other application areas. Most of them fall into one of two main classes: *parametric filters* and *nonparametric filters* (cf. Thrun et al., 2005).

Parametric filters—based either on the Kalman (Kalman, 1960) or the Information filter (Mutambara, 1998)—use normal distributions as approximations and have been applied in combination with landmark-based representations and geometric representations. Smith & Cheeseman (1986) were the first to use the extended Kalman filter (EKF) to formulate a solution to the landmark-based SLAM problem. This work has spurred a huge amount of research on EKF-based SLAM (Castellanos et al., 1999; Dissanayake et al., 2001; Guivant & Nebot, 2001). Besides the general inability to adequately represent multimodal probability distributions, the main drawbacks or challenges of this approach are the quadratic space and time complexities: Incorporating a new observation requires $O(n^2)$ time and the covariance matrix which needs to be maintained has a size of $O(n^2)$ (where n is the number of landmarks). In addition, the approach relies on correct data association.

In the last few years, several methods to reduce the quadratic time complexity of the EKF and Information filter have been proposed (Frese & Hirzinger, 2001; Paskin, 2003; Thrun et al., 2002). Often these approaches employ some kind of submapping strategy involving a (hierarchical) decomposition of space into subregions, e.g., the Compressed EKF (Guivant & Nebot, 2003, 2001) and Treemap (Frese, 2006b; Frese & Schröder, 2006).

Nonparametric filters approximate probability distributions by a finite set of samples. The dominant approach in this class is the so-called Rao-Blackwellized particle filter (Doucet et al., 2000), in which the probability distribution is factorized and each particle stands for a particular robot trajectory and the map associated with this trajectory. The probability of each sample is given by its importance factor. Rao-Blackwellized particle filters have first been introduced for landmark-based representations under the name FastSLAM (Montemerlo & Thrun, 2003; Montemerlo et al., 2003). A result of the factorization is that the positions of the landmarks within the map can be tracked individually, each with its own extended Kalman filter.

In Hähnel et al. (2003a) the idea of Rao-Blackwellized particle filters is first used in connection with a grid map representation. Again, the particles stand for entire trajectories. The map corresponding to a particle is computed from the trajectory using standard occupancy grid mapping. Improvements of the original algorithms and approaches to reduce the so-called particle depletion problem that arises when the number of used particles is too small to adequately represent the probability distribution are described in Grisetti et al. (2007a,b). Eliazar & Parr (2003, 2004) describe a standard particle filter approach to construct occupancy grid representations.

The approach of maintaining a probability distribution over the complete state space has been very successful in achieving a high level of robustness under uncertain conditions. Most current state-of-the-art mapping approaches fall into this class. However, as mentioned, the approach is computationally demanding and the approximation techniques still tend to have limitations or are applicable only under particular conditions. In addition, while current approaches show that a rigorous probabilistic formulation is possible for representations that are still very close to the sensor data (grid maps, landmark maps, and to a certain degree geometric representations), it is an open problem how this approach could be adapted for more abstract forms of representation and organization.

2.4.1.3 Multi-hypothesis Approaches

Maintaining a discrete set of hypotheses offers a way to reduce the brittleness of many single hypothesis approaches. Multi-hypothesis approaches can better deal with ambiguities, although they carry additional computational costs as multiple spatial models have to be maintained. At the same time, it has been seen as a way to deal with one disadvantage of parametric complete-state-space approaches: their inability to adequately represent multimodal probability distributions. Moreover, particle filter-based approaches have a kind of multi-hypothesis flavor as each particle stands for one particular hypothesis. However, the fact that they form an approximate representation of the actual probability distribution over the complete state space explains why they belong to the previous class. Two important questions in the context of multi-hypothesis approaches are about when to instantiate a new hypothesis and when to discard a hypothesis that has become implausible.

In the context of Kalman filter-based approaches, multi-hypothesis tracking techniques have been developed to deal with the ambiguities arising in the data association step, which would result in a multimodal probability distribution. Cox & Leonard (1994) employ this idea—which originally was developed in work on object tracking (e.g. Reid, 1979)—for robot mapping using a geometric representation with features for walls and corners. The approach assumes that the poses of the robot are known. This allows the use of an independent Kalman filter for each geom. Whenever there exist multiple plausible associations for the current observation, which is determined using validation gating (cf. Sect. 5.1.2), a hypothesis is split into multiple new hypotheses. Probabilities are assigned to the hypotheses and they are maintained in a hypothesis tree with the current hypotheses forming the leaves. To deal with the problem of exponential growth, the authors consider multiple pruning strategies to remove branches of the tree based on the assigned probabilities.

In Smith & Leonard (1997), this approach is extended to deal with the full SLAM problem. Hypotheses for robot pose and feature map are tracked independently (ignoring correlations between the two robot and feature states).

While in the above approaches all hypotheses are constructed whenever a new observation becomes available, and hence all leaves are at the same level of the tree, (Hähnel et al., 2003) proposes a lazy search through the hypothesis tree. In the described approach, hypotheses are only updated when they have the potential to become more likely than the currently considered hypothesis. Thus, the leaves of the tree are at different levels depending on the number of observations that have already been incorporated in the corresponding hypothesis. The lazy search is possible since the log-likelihood values of the hypotheses decrease monotonically with the depth in the tree.

In a newer implementation of Kuipers's SSH (Kuipers et al., 2004), a topological equivalent to the previously described multi-hypothesis approaches for coordinate-based representations is presented. A tree of topological map hypotheses is maintained. Every time a new view-action-view triple is experienced, all hypotheses are updated accordingly. This can lead to multiple successor hypotheses. Hypotheses that violate the axioms of the topological level of the SSH are pruned. Active exploration is used until the simplest hypothesis with respect to the logical theory has been unambiguously identified. The experiments using an environment with nine places demonstrate that the high level of abstraction of the involved representation (nodes roughly correspond to junctions and edges correspond to hallways) together with the hard constraints for pruning allow for an exhaustive search through the tree of hypotheses. While it remains open how this general approach scales to more complex environments, an investigation of how the assumption that the graph hypotheses have to be embedded into the plane affects the size of the space of hypotheses is conducted in Savelli & Kuipers (2004).

In general, the multi-hypothesis approach seems to have great potential as a compromise for achieving a sufficient degree of robustness while avoiding the computational costs of complete-state-space approaches and still being rather universally applicable, as it does not assume known data associations. The main challenge is to find good strategies to exploit all available information to reduce the search space enough to avoid the exponential growth. It is especially promising for more abstract representations for which a complete probabilistic formulation might be hard to achieve but for which the combinatorial problem is significantly reduced.

2.4.2 Multi-pass Approaches

In multi-pass approaches the input data is processed several times, allowing for revising past location estimates or recovering from wrong data associations, a property that most incremental approaches are lacking. The most well-known family of multi-pass approaches is based on a statistical approach known as *expectation maximization* (EM) (Dempster et al., 1977). Thrun et al. (1998b) use this approach to build landmark-based maps (landmarks were simulated by button presses). The idea is to search for the most likely map given the input data by using hill-climbing in the likelihood space. The EM algorithm alternates two steps, the *expectation step* and the *maximization step*. In the expectation step, a sequence of pose estimates is determined based on the current most likely map (localization with a given map), while the maximization step computes the most likely map given the sequence of poses computed in the expectation step (mapping with given pose estimates). The result of the EM algorithm can then be used, for instance, to construct a grid map from the pose estimates and sonar measurements collected in addition to the landmark information. Although naturally the approach is only guaranteed to converge to a local maximum, the experiments indicate that it performs robustly in the presence of large odometry errors. A good initial map estimate improves performance of the algorithm significantly.

Burgard et al. (1999) improve this approach and replace the manually added landmarks by local grid maps. A modified maximization step is used in which deterministic annealing is utilized in order to reduce the chance of getting stuck in a local maximum because of a bad initial map. EM-based mapping has also been applied for the plane fitting subproblem of learning compact geometric 3D models from laser range data (Liu et al., 2001).

Other multi-pass mapping techniques have been developed in the context of scan matching. Lu & Milios (1997) describe the problem of constructing a map from a sequence of scans as a global optimization problem and compute a solution based on minimizing an energy function using a spring model. Similarly, in the work of Nüchter et al. on 6D SLAM (Nüchter et al., 2004, 2005) a global relaxation approach to diffuse the accumulated error over all scans is employed in which the scans are registered multiple times with respect to their neighboring scans.

2.5 Conclusions

We have looked at the multitude of mapping approaches described in the literature, first from the perspective of spatial representation and then from the perspective of uncertainty handling. For assessing different kinds of spatial representation, we referred to the criteria proposed in Sect. 2.2 (extractability and maintainability, information adequacy, efficiency and scalability) and the three main operations of navigation, systematic exploration, and communication (Sect. 2.1).

When looking at the spatial representations employed by the approaches listed in the previous section on uncertainty handling, several major clusters become immediately apparent. For all kinds of coordinate-based representations, ways to maintain multiple or even infinitely many hypotheses, as in the complete-state-space approaches, have been developed. In contrast, relational representations in almost all cases rely on the correctness of a single hypothesis and thus tend not to work reliably when the input data is ambiguous. So far, only very few attempts have been made to develop or adapt powerful uncertainty handling approaches for relational representations.

On the other hand, as we have seen when looking at representation approaches from the spatial representation perspective, relational approaches offer advantages over coordinate-based representations that make such techniques desirable. One approach to deal with this lack of reliable construction and maintenance methods is to aim for complex forms of organization in which a coordinate-based approach is used to derive a relational representation. However, as in the case of an overlay approach combining a global coordinate-based representation with a global graph representation, this approach tends to preserve too many of the disadvantages of the chosen representations. It also seems implausible from a cognitive perspective, as humans are known to develop knowledge about the geometric layout of an environment last, and this knowledge is usually systematically distorted (Siegel & White, 1975; Tversky, 1992).

We therefore argue that developing robust methods for directly constructing and maintaining representations that are relational at the core but can be enriched with other kinds of information will lead to robot mapping systems that scale much better to larger environments and will perform much better in the context of the considered broader set of tasks. To reach this robustness, investigating ways to combine relational representations with more sophisticated uncertainty handling techniques is a necessity. The fact that relational representations typically are based on relations that can be detected more reliably should in principle make this combination even more robust than in the case of representations based on information that can be observed only very unreliably.

In this work, we follow this direction of research by investigating the combination of a particular route graph representation with multi-hypothesis tracking and robust uncertainty handling techniques. Our focus lies in developing basic methods that make this combination possible. However, we also consider and compare three different mapping systems realized based on these techniques.

Chapter 3

Voronoi-Based Spatial Representations

In this chapter, we introduce the Voronoi-based route graph representation, which we will be concerned with in the remainder of this book, and its hierarchical extension. We will do this by going through a sequence of increasingly complex representations starting from the standard Voronoi diagram. Note that some of the names used for these structures are already in use in the robot mapping literature, but sometimes in a different sense. Especially, in contrast to the hierarchical generalized Voronoi graph of Choset et al. (Choset & Burdick, 2000; Choset et al., 2000), which is a generalization of Voronoi graphs to higher dimensions, our hierarchical Voronoi graphs are hierarchically organized representations of 2D space.

3.1 Voronoi Diagram and Generalized Voronoi Diagram

The *Voronoi diagram* (VD) and the *generalized Voronoi diagram* (GVD) are both geometric structures formed by sets of points in $\mathbb{R}^n$ (all the examples here use $\mathbb{R}^2$). A VD for a given set of point sites S as shown in Fig. 3.1(a) consists of the set of all points that have at least two minimally distant sites in terms of Euclidean distance. All points not part of the VD are closest to exactly one particular point site of S. The VD thus partitions the $\mathbb{R}^n$ into cells containing all the points closest to one of the sites.

Definition 3.1 (Voronoi diagram)**.** *Given a set S of different points in $\mathbb{R}^n$ and a function $md : \mathbb{R}^n \to 2^S$, $md(p) = \operatorname{argmin}_{q \in S} ||q - p||$ yielding the set of points from S with minimal distance to p, the Voronoi diagram $\mathcal{VD}_S$ of S is defined as*

$$\mathcal{VD}_S = \{ \, p \in \mathbb{R}^n \; | \; |md(p)| \geq 2 \, \} \tag{3.1}$$

The VD consists of line segments and half-lines, which are both formed of points with exactly two minimally distant points in S. These lines meet at so-called *meet points*, which have at least three minimally distant points in S. Often the notion of

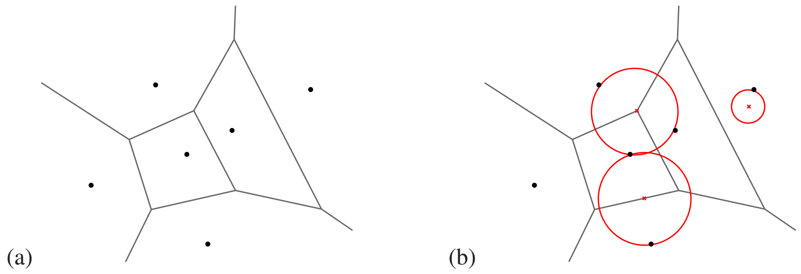


Figure 3.1: (a) The Voronoi diagram for a given set of point sites. It consists of line segments and half-lines that meet at so-called meet points. (b) The maximal inscribed circles for a meet point, a point on a line segment, and a point not belonging to the VD

a *maximal inscribed circle* is used to explain the concept of a Voronoi diagram (see Fig. 3.1(b)): The maximal inscribed circle of a point p is the circle centered on p with maximal radius that does not contain any of the points from S. For any point that belongs to the VD, the maximal inscribed circle touches at least two point sites. In the case of a meet point, the circle touches three or more point sites, while for any other point of the VD it touches exactly two.

The *generalized Voronoi diagram* (GVD) extends the idea of the VD from point sites to a set G of arbitrary geometric objects (e.g., lines, arcs). Accordingly, the GVD is the set of all points of $\mathbb{R}^n$ that have at least two minimally distant closest points belonging to different objects in G.

Definition 3.2 (Generalized Voronoi diagram)**.** *Given a set G of geometric objects in $\mathbb{R}^n$, a function dist : $\mathbb{R}^n \times G \to \mathbb{R}$ that yields the minimal distance between a point $p \in \mathbb{R}^n$ and an object $g \in G$, and a function md : $\mathbb{R}^n \to 2^G$, $md(p) = \operatorname{argmin}_{q \in G} dist(p, q)$ yielding the set of objects from G with minimal distance to p in terms of dist, the generalized Voronoi diagram $\mathcal{GVD}_G$ of G is defined as*

$$\mathcal{GVD}_G = \{\ p \in \mathbb{R}^n \ \mid\ |md(p)| \geq 2\ \} \tag{3.2}$$

In the remainder of this text, we will exclusively deal with GVDs for sets of line segments in 2D extracted from laser range data or with their discrete counterparts computed from grid maps. For dist(p, q) we take the function that yields the Euclidean distance to the closest point belonging to the line segment at hand. In all cases, the line segments form a simple polygon (potentially with polygonal holes) that describes the boundary of the free space. We will only consider that part of the GVD that is contained in free space and mask out the parts lying outside the polygon or inside the holes. As a result, the GVD will always be connected.

One example of a GVD is given in Fig. 3.2(a). It shows a polygonal 2D environment and the GVD computed from the line segments describing the object boundaries.

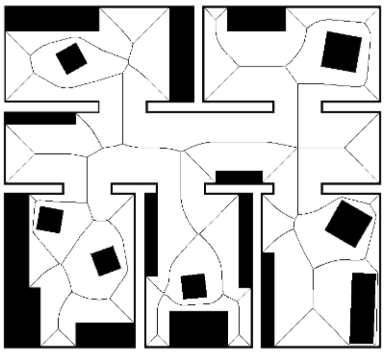
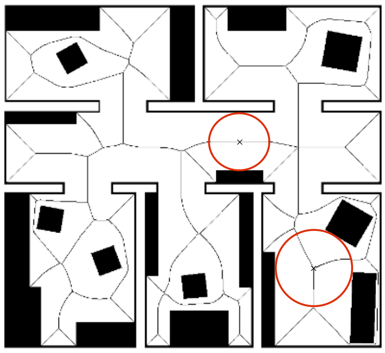
(a)

(b)


Figure 3.2: (a) The generalized Voronoi diagram (fine lines) derived from a polygonal description of a 2D environment. It consists of meet points connected by Voronoi curves. Some Voronoi curves end in concavities of the environment forming end points. (b) Maximal inscribed circles for a meet point and a point on a Voronoi curve

Instead of line segments and half-lines, the GVD consists of one-dimensional curve segments (either straight line segments or parabolic curve segments) that meet at the meet points. We will use the term *Voronoi curve* for such a curve segment. In addition to meet points, we have *end points* where a Voronoi curve terminates in a concavity of the environment.

The concept of the maximal inscribed circle also applies for GVDs, as shown in Fig. 3.2(b). We will call the points where the maximal inscribed circle of point p touches the boundaries of objects in the environment the *generating points* of p. The GVD can be computed in $O(n \log n)$ time, where n is either the number of line segments or the number of cells in a grid map representation (see Aurenhammer, 1991; Okabe et al., 2000).

3.2 Generalized Voronoi Graph and Embedded Generalized Voronoi Graph

A basic form of route graph representation can be achieved by employing the *generalized Voronoi graph* (GVG), which is the graph analogue of a GVD. It contains nodes, called *Voronoi nodes*, for every meet point and end point of the GVD. Two Voronoi nodes are connected by an edge if the corresponding meet or end points in the GVD are connected by a Voronoi curve. The GVG corresponding to the GVD of Fig. 3.2 is shown in Fig. 3.3. Please note that although the nodes are depicted at the positions of the corresponding meet and end points, no geometric information is included in the GVG.

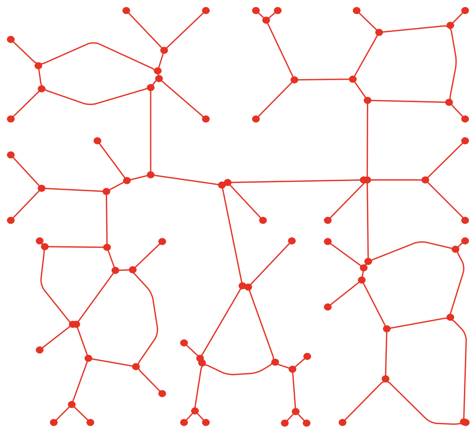


Figure 3.3: The generalized Voronoi graph corresponding to the GVD of Fig. 3.2. For visualization purposes, the nodes are placed at the positions of the corresponding meet points and the edges reflect the shapes of Voronoi curves. However, this information is not part of the GVG

Definition 3.3 (Generalized Voronoi graph)**.** *The generalized Voronoi graph of a GVD $\mathcal{D}$ is an undirected pseudograph $\mathcal{GVG}_{\mathcal{D}} = (V, E)$ in which V contains exactly one node for each meet point and end point of $\mathcal{D}$, E contains exactly one edge for each Voronoi curve in $\mathcal{D}$, and $e \in E$ is incident to $v, w \in V$ if its corresponding Voronoi curve connects the points corresponding to v and w in $\mathcal{D}$.*

The GVG has to be defined as a pseudograph because it may contain loops and parallel edges. For simplicity, we will just speak of graphs in the following, always referring to undirected pseudographs unless explicitly stating otherwise. We will use the notations $V(G)$ for the set of nodes of graph G and $E(G)$ for the set of edges of G. $\deg(v)$ stands for the degree of node v. We will also make use of the functions $\text{connects}(e)$, which yields the set of nodes connected by edge e, and $\text{other}(e, v)$, which yields the node from this set that is not v (or yields v in case e is a loop).

The GVG represents the connectivity of the GVD and, thus, also reflects the connectivity of free space. However, even if we assume a robot has the GVG of the environment given in advance, knows its starting position, and is able to reliably navigate between meet points along the GVD, it would turn out that the GVG is still rather inadequate as a route graph representation. The reason is that it is not possible to relate the leaving Voronoi curves perceived at a meet point to the edges in the graph. What is further required is a description of the *combinatorial embedding* of the GVG. A combinatorial embedding specifies for each node the cyclic order of edges incident to it in the graph. We will call a GVG with a combinatorial embedding an *embedded GVG* (EGVG). Given an EGVG, knowing along which edge an agent arrived at a node allows us to unambiguously map perceived Voronoi curves to edges and track the robot's way through the graph.

In the following definition, we will use the notion of successor functions to specify the cyclic order of edges incident to a node in an EGVG. A successor function maps each edge incident to a particular node v to the immediate successor edge in the counterclockwise order of incident edges of v.

Definition 3.4 (Embedded GVG)**.** *An embedded generalized Voronoi graph is a triple* (V, E, O) *where*

- (V, E) *is a GVG, and*
- $O = \{succ_{v_1}, succ_{v_2}, ..., succ_{v_n}\}$ *is a set of* $n{=}|V|$ *successor functions of which* $succ_{v_i}$ *specifies the counterclockwise cyclic order of edges incident to* v_i*.*

To refer to the edges incident to a node v in an EGVG, we will use the notation $e_i^{[v]}$ with $1 \leq i \leq \deg(v)$ for the ith edge of v. In doing so, we assume that the edges are numbered in accordance with the respective successor function of v (i.e., $e_{i+1}^{[v]} = \mathrm{succ}_v(e_i^{[v]})$), starting with an arbitrarily chosen edge $e_1^{[v]}$.

3.3 Annotated Generalized Voronoi Graphs

In principle, the EGVG provides all the information that is required for navigating through the environment. However, to achieve robust navigation in practice and to allow for planning and communication, the representation needs to be enriched with additional information. This additional information is stored in the form of annotations to the nodes and edges of the EGVG.

Examples of node annotations are

- local descriptions of the corresponding meet point within its own frame of reference, containing for instance
  - directions of leaving Voronoi curves
  - directions to adjacent Voronoi nodes
  - positions of generating points (equivalent to the radius of the maximal inscribed circle plus directions)
  - a local coordinate-based representation of the neighborhood of the node,
- global position estimates for the node.

Examples of typical edge annotations are

- distance between the connected nodes (in terms of linear distance, length of the Voronoi curve, or travel time),

- minimal clearance along the Voronoi curve,
- a local description of the shape of the Voronoi curve,
- a local coordinate-based representation of the area corresponding to the edge,
- explored/unexplored status information.

We will use the term *annotated generalized Voronoi graph* (AGVG) for all forms of graphs that result from an EGVG by labeling its nodes and edges with additional annotations. As an AGVG is not very useful in our context without its embedding, we formalize the concept as an extension of the EGVG but leave out "embedded" in the name and abbreviation. We formalize annotations as functions that map nodes and edges to values from the domains of the corresponding attributes.

Definition 3.5 (Annotated GVG)**.** *An annotated GVG is a 5-tuple* (V, E, O, A_V, A_E) *where*

- (V, E, O) *is an EGVG,*
- $A_V = \{a_1, a_2, ..., a_m\}$ *is a set of total functions* $a_i : V \rightarrow D_{V,i}$ *mapping each node to a value from the domain* $D_{V,i}$ *of the corresponding node attribute,*
- $A_E = \{b_1, b_2, ..., b_n\}$ *is a set of total functions* $b_j : E \rightarrow D_{E,j}$ *mapping each edge to a value from the domain* $D_{E,j}$ *of the corresponding edge attribute.*

Later in this text, we will, for instance, use the function $r : V \rightarrow \mathbb{R}_0^+$ for the node attribute that yields the radius of the maximal inscribed circle of a node.

3.4 Hierarchical Annotated Voronoi Graphs

A *hierarchical AGVG* (HAGVG) follows the notion of hierarchical representations as discussed in Sect. 2.3.4.3. It can be used to describe an environment at different levels of granularity. An HAGVG consists of multiple graph layers of increasing levels of abstraction. Neighboring layers are linked with each other, which allows switching to a coarser or finer level as the task at hand demands. The bottom layer of an HAGVG could for instance consist of the original AGVG derived from the environment. Every other layer in an HAGVG is derived from the layer below it by removing certain nodes and edges. Methods for automatically extracting the higher levels will be developed in Chap. 4.

In our approach, we allow nodes on a higher level to stand for node-bounded subgraphs at the level below, while edges may stand for edge-bounded subgraphs. The links between the neighboring layers are given by abstraction relations that associate nodes and edges from one layer with corresponding elements of the next highest layer. A general definition of the HAGVG is given below.

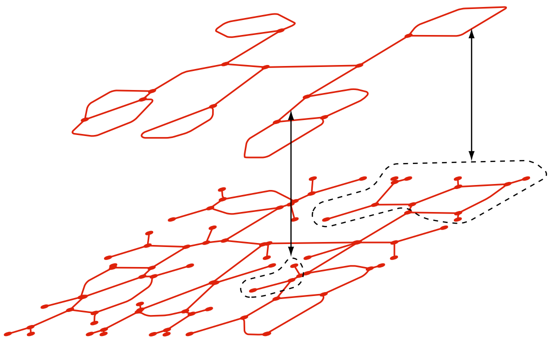


Figure 3.4: A hierarchical AGVG consisting of two layers: The original AGVG at the bottom and a coarser graph model at the top. Both layers are linked by an abstraction relation

Definition 3.6 (Hierarchical AGVG)**.** *A hierarchical AGVG is a pair* (L, A) *where*

- $L = \langle L_1, L_2, ..., L_n \rangle$ *is a list of AGVGs* $L_i = (V_i, E_i, A_{V,i}, A_{E,i})$ *(the layers) with* $V_{i+1} \subseteq V_i$*, and*
- $A = \langle \mathit{abstraction}_1, \mathit{abstraction}_2, ..., \mathit{abstraction}_{n-1} \rangle$ *is a list of abstraction relations* $\mathit{abstraction}_i \subseteq (V_i \cup E_i) \times (V_{i+1} \cup E_{i+1})$.

Figure 3.4 shows a two-layer HAGVG for the example environment used in this chapter. The bottom layer is the original AGVG from Fig. 3.3. The top layer is a coarser version in which several subgraphs of the bottom layer have been mapped to edges at the top layer, as indicated for two exemplary cases.

3.5 Partial and Local Voronoi Graphs

Our main goal in this work is to develop techniques to construct (H)AGVG representations incrementally from local observations. In contrast to approaches that extract a Voronoi representation from a complete geometric or grid map representation, the graph model grows incrementally while new nodes and edges extracted from new observations are incorporated. As a consequence, as long as the robot has not explored the entire environment, its (H)AGVG is actually only a *partial (H)AGVG*. A partial (H)AGVG is a substructure of the complete (H)AGVG. It contains edges which are marked as unexplored and end at nodes of degree 1. These node will be replaced once the corresponding edge gets explored further.

A particularly important concept for our incremental construction approach is the notion of a *local AGVG*. A local AGVG is an AGVG extracted from the robot's current perception or alternatively from a local map describing the robot's immediate environ-

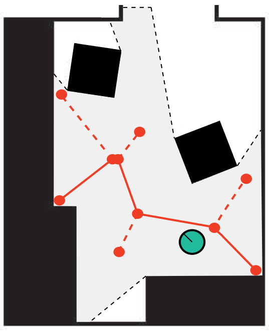
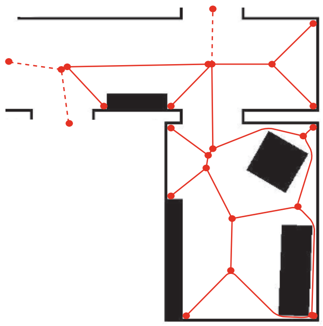
(a)  (b) 

Figure 3.5: Incomplete AGVG structures: (a) a local AGVG computed from the current visibility polygon, (b) a partial AGVG. Dashed lines represent unexplored edges

ment. It only contains nodes and edges which have to be part of the global AGVG of the environment.

We will only give a definition of the *local GVD* here. A local GVD can be abstracted by a *local (A)GVG* similarly to the way it is in the global case. We assume that the local information is given in form of a visibility polygon. A visibility polygon consists of two kinds of line segments: (1) actual obstacle boundaries and (2) line segments that are visibility boundaries. The visibility boundaries delimit areas of the environment that are occluded or are out of sensor range. Under this assumption, the local GVD is the set of all points that are part of the GVD of the visibility polygon and for which at least two of the minimally distant objects are real obstacle boundaries and not visibility boundaries. In the following definition, we assume that md(p), similarly to Definition 3.2, yields the set of minimal distant objects from the set of line segments forming the visibility polygon.

Definition 3.7 (Local GVD)**.** *Given a visibility polygon P, with B being the set of line segments of P that are proper obstacle boundaries, the local GVD $\mathcal{LGVD}_P$ of P is given by*

$$\mathcal{LGVD}_P = \{\ p \in \mathcal{GVD}_P \ \mid \ |md(p) \cap B| \geq 2\ \} \tag{3.3}$$

Local GVDs typically contain Voronoi curves that neither terminate at a meet point nor at an end point but somewhere within free space where their continuation cannot be determined because of the presence of visibility boundaries nearby. These Voronoi curves are the ones that will be represented as unexplored edges in the corresponding local AGVG. Figure 3.5 shows on the left a local AGVG for the visibility polygon shaded in gray. On the right, it shows a partial AGVG containing multiple unexplored edges. It has been constructed by merging multiple local AGVGs. Computing the local AGVG from a given visibility polygon by pruning parts from the complete AGVG of the visibility polygon has been addressed in Wallgrün (2002).

3.6 An Instance of the HAGVG

In the concrete instances of (H)AGVG representations investigated throughout the following chapters, the following annotations to the graph structure are employed:

1. Nodes are labeled with a *signature* (see Fig. 3.6(a)) that contains the distance to the generating points (which is the radius of the maximal inscribed circle) and the angles to the lines connecting the node with its generating points with respect to an arbitrarily chosen reference direction.

2. The approximate relative positions of nodes are represented by annotating the edges with the approximate distances between the connected nodes and by annotating the nodes with the approximate angles to the leaving edges, again with respect to the chosen reference direction (see Fig. 3.6(b)).

3. Every edge is annotated with a description of the shape of the Voronoi curve corresponding to it. The Voronoi curve may deviate significantly from the direct connection between the two nodes. The description simply consists of a sequence of intermediate points in the local reference frame of the edge. We will use the term *course* adopted from Krieg-Brückner et al. (2005) for this description. Moreover, the approximate travel distance for moving along the Voronoi curve is stored.

4. Additional edge attributes provide information about traversability of an edge (sufficient distance to obstacles) and whether the edge leads to still unexplored areas.

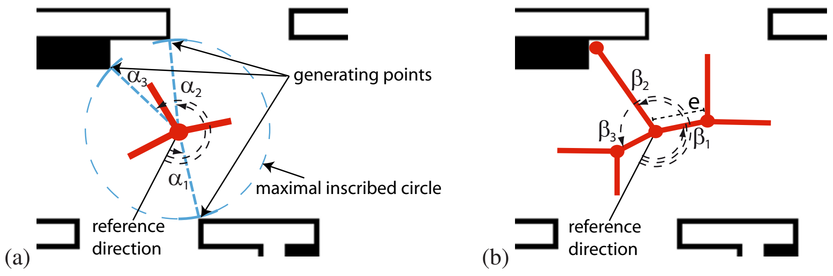


Figure 3.6: Different kinds of annotations to the AGVG-based route graph: (a) The node signature contains the distance of the generating points (radius of the maximally inscribed circle) and angles between the connections to the generating points. (b) Relative position information is given by the angles between leaving edges and the lengths of the edges

Our HAGVG representation does not contain any information about the geometry of the environment other than what is listed above. In particular, sensor information or local metric maps used to extract a local AGVG are not stored. As a result, the overall representation is relatively sparse. The signature information is helpful mainly for localization as it allows to compare two nodes based on the similarity of their signatures. The relative positional information provided by the approximate angles between leaving edges and their lengths allows us to geometrically match local configurations of nodes, which also supports localization (cf. Chap. 5). A benefit of using relative metric information is that the information can be globally inconsistent without diminishing its usability for navigation.

Path planning can be performed based on the estimated travel distance annotated to the edges using one of the standard graph search techniques for weighted graphs. The HAGVG can be employed for more efficient hierarchical path planning when the distances of the abstracting edges on higher levels correctly reflect the shortest path through the abstracted subgraph. The first step of the hierarchical search is to abstract start and goal nodes to nodes on the top level. A plan compiled on the top level would then correspond to macro operations like driving from one door to the next along a corridor. As this plan is not directly executable with the low-level navigation procedures of the robot, each edge has to be recursively refined until a plan at the detailed level of the original AGVG is reached.

The course information is mainly used for visualization, as movement along the Voronoi curve can also be achieved through reactive behaviors. Systematic exploration is supported by the corresponding edge status attributes, and different strategies can be employed to select where the exploration should be continued next.

In principle, much more information could be attached to the graph structure. One example would be information about landmarks, which could support the generation or execution of route instructions, but we do not explicitly cover communication in this work. Nevertheless, exchange of spatial information between heterogeneous agents can benefit from the compact high levels of abstraction provided by the top layer(s) of the HAGVG representation.

3.7 Stability Problems of Voronoi-Based Representations

One point of criticism raised against Voronoi-based representation approaches is that the underlying GVD is very unstable under noisy conditions. Figure 3.7, for instance, shows two GVGs computed from two laser scans recorded consecutively from exactly the same position. The underlying GVDs are computed by segmenting the scan into line segments using the *iterative end-point fits* method (Duda & Hart, 1973) and then computing the GVD from the resulting polygonal description. As we can see, some tiny concavities in the surrounding polygon stemming from noise will cause new end nodes and edges leading into these concavities. As a result, we see small structural deviations between the GVGs, for instance, in the top corner and on the right side. The

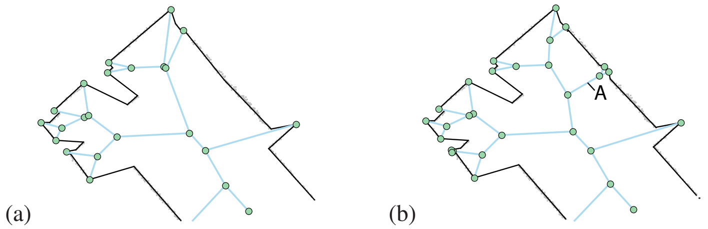


Figure 3.7: Structural deviations in the GVGs computed from two consecutive scans, caused by noise and different line segmentations

same problem arises when the GVD is computed from a grid map representation.

Structural variations like this can be problematic for robust construction, localization, and reliable reactive navigation between the nodes. One simple approach suggested to deal with this problem is to prune all end points together with their edges, as those are not relevant for navigation anyway (Nagatani & Choset, 1999). However, as the example shows, noise can introduce new branches consisting of more than a single edge. For instance, edge A in the example is not part of the left GVG and would not be removed by this simple approach.

A more adequate approach able to detect and remove unstable parts of a GVG will be developed in Chap. 4.

3.8 Strengths and Weaknesses of the Representation

Let us summarize why we believe the proposed hierarchical Voronoi-based route graph representation is a very good choice for the spatial representation of a mobile robot, and which problems and restrictions need to be addressed.

First of all, the choice of a route graph representation as the core has the following advantages:

- It is a rather compact representation focusing on relevant aspects of the environment and, hence, scales well to larger environments.
- It directly models topology of free space and thus allows for very efficient path planning.
- By directly modeling distinct routes through the environment, it becomes suitable for route-based communication.
- Metrical information stored in the form of annotations to the graph structure can be globally inconsistent without diminishing the representation's usability for navigation.

- It allows for anchoring other kinds of information (e.g., landmark information, sensor snapshots).

The downside of this choice is that the approach is restricted to clearly structured environments like indoor environments. Even there, depending on the employed sensors, open areas or clutter might cause problems. Moreover, localization and, as a consequence, construction and maintenance of the representation are complicated by the lack of detailed geometric information.

The Voronoi-based approach results in a few more benefits:

- The Voronoi diagram is a retraction of free space, which directly gives rise to systematic and easily realizable exploration algorithms.
- Stable meet points can also serve as landmarks, allowing a combination with stochastic mapping approaches (cf. Sect. 7.2).

One disadvantage of Voronoi-based route graphs in comparison with other kinds of route graph representations proposed in the literature is that the density of nodes is still comparatively high. This reduces the scalability and efficiency of operations like path planning to a certain degree. In addition, methods to work around the mentioned stability problems need to be developed. The chosen set of additional attributes stored in our representation, such as signature and coarse relative metric information, is intended to reduce perceptual aliasing and thus improve localization while still keeping the representation compact.

The hierarchical organization further improves certain properties of the representation approach. Especially, operations such as path planning become even more efficient. In addition, it improves the adequacy for communication tasks by bridging from sensor information to higher levels of abstraction. This comes at the cost of increased complexity for maintaining the representation, but we believe that this is unavoidable if the goal is to open new areas of applications, especially with respect to human-robot interaction.

While the representation has many advantages from the spatial representation perspective, the main challenge is to provide construction and maintenance procedures that show a robustness comparable to that of state-of-the-art mapping approaches.

To conclude this chapter, in which we described our proposed Voronoi-based route graph representation, we provide two examples of two-level HAGVG representations derived from real 2D sensor data, anticipating some of the results of later chapters. Figures 3.8 and 3.9 show the HAGVGs which have been constructed from exploration data sets of two different environments. The individual layers are shown separately and the links between the layers are omitted. Edges are drawn using the course information and thus are not simply straight connections. In the following chapters, we will develop and refine the techniques required to construct this kind of representation incrementally from local sensor information or small local metric maps, without the need for building a global metric map first.

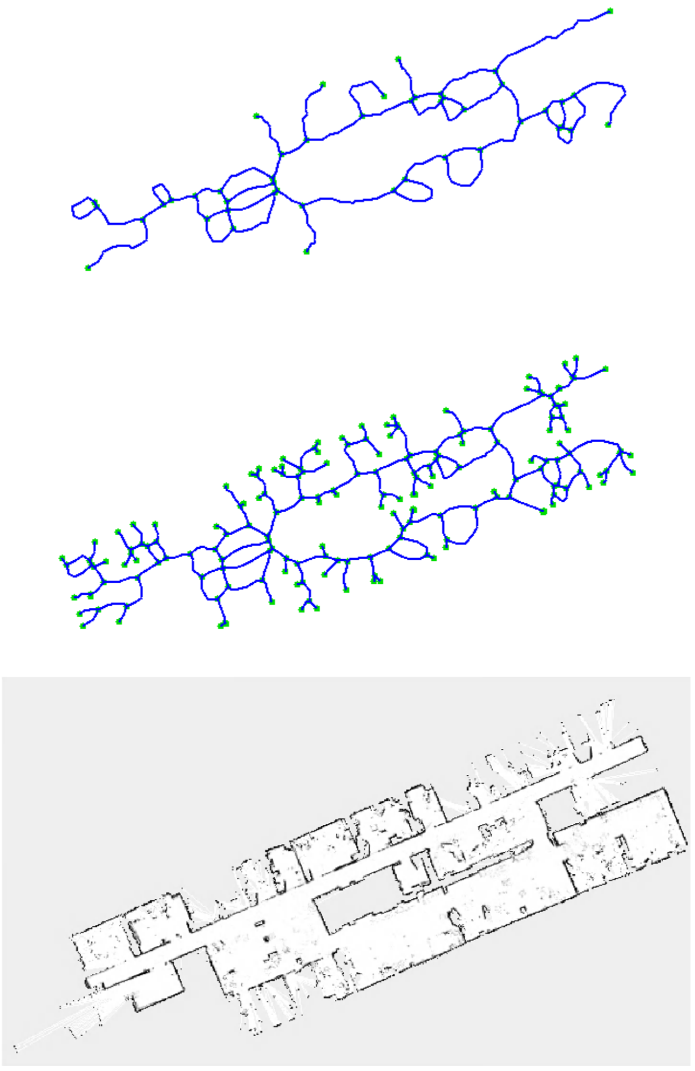


Figure 3.8: A two-level HAGVG representation consisting of the original AGVG (middle picture) and a coarser AGVG (top picture). The environment (bottom picture) shown as an occupancy grid is the third floor of the Cartesium, Universität Bremen

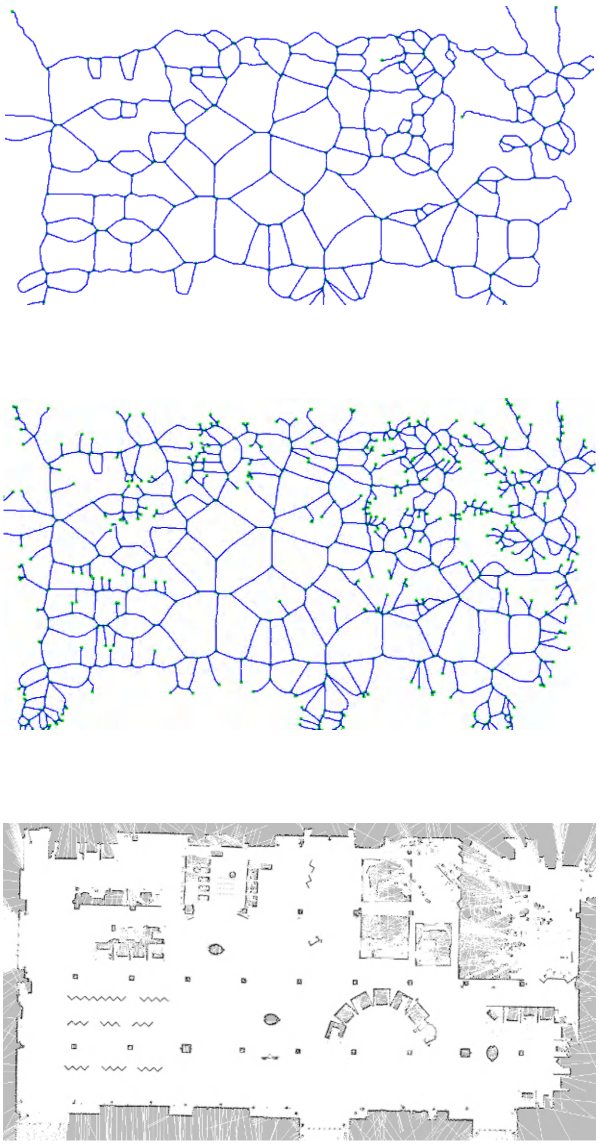


Figure 3.9: A second example of an HAGVG representation, this time constructed from a data set recorded in the Acapulco Convention Center (data set available at `http://radish.sourceforge.net/`, courtesy of N. Roy)

Chapter 4

Simplification and Hierarchical Voronoi Graph Construction

In the previous chapter, we proposed the HAGVG as an exhaustive environmental model for a mobile robot. As stated, our general aim is that the graph representation be learned incrementally from local information. Sensor information or locally computed metric representations used to compute a local AGVG will be discarded immediately.

In this chapter, we will start by focusing on the problem of automatically deriving coarser AGVG representations and constructing complete HAGVG representations from a given AGVG. For this purpose, we will introduce relevance measures for Voronoi nodes and the regions accessible via their edges. These measures serve two aims:

1. being able to deal with the stability problems described in Sect. 3.7 by identifying and removing unstable parts of an AGVG, and
2. assessing the significance of Voronoi nodes in an AGVG as a decision point for navigation with the goal of deciding which nodes should be retained on higher levels of abstraction.

As we will show, the two notions of stability and relevance for navigation are very much related and thus can be covered by the same measures. The process of constructing a coarser AGVG based on these measures will be referred to as *simplification*. Simplification can be used for deriving more abstract levels of representation as well as for removing unstable parts caused by noise. The work described in this chapter is related to work on shape representation and matching approaches developed in the vision community that also employ Voronoi diagrams or skeletons (e.g., Mayya & Rajan, 1996; Ogniewicz & Kübler, 1995; Siddiqi & Kimia, 1996). However, the basic conditions in these approaches are different as here a complete description of the shape's boundary can be used to simplify the Voronoi structures. In contrast, our incremental construction in which no global representation of the shape of the environment is

J.O. Wallgrün, *Hierarchical Voronoi Graphs: Spatial Representation and Reasoning for Mobile Robots*, DOI 10.1007/978-3-642-10345-2_4, © Springer-Verlag Berlin Heidelberg 2010

maintained requires a different approach, in which the simplification is purely based on the information stored in the AGVG.

4.1 Relevance Measures for Voronoi Nodes

In the following, we first assume that we have a complete AGVG with no unexplored edges. We will discuss the ramifications of allowing unexplored edges later in this chapter.

The relevance measures we are going to propose below are only concerned with inner nodes of the AGVG, which are those that correspond to meet points. Nodes corresponding to end points do not represent decision points but mark ends of routes. The foundation of the relevance measures are the following observations:

1. The relevance of a node depends on the properties of the regions accessible by its leaving edges.
2. For a node to constitute a relevant decision point, at least three of those regions need to be significant enough to be considered as alternative continuations when arriving at it.
3. The significance of a region is higher when it contains more distant goals.
4. The significance of a region is higher when it leads to areas secluded from the rest of the environment.

Figure 4.1 illustrates the idea of *accessible regions* of a node v. They emerge when the free space of the environment is split up by the virtual lines connecting v with its generating points. We will use the notation $R_i^{[v]}$ for the region that can be accessed from node v via edge $e_i^{[v]}$ without crossing one of these virtual connections. If multiple leaving edges of a node are part of a cycle in the AGVG like $e_2^{[A]}$ and $e_3^{[A]}$ in Fig. 4.1(a), their corresponding regions will be identical. The region $R_1^{[A]}$ of A in Fig. 4.1(a) only corresponds to a small niche and thus typically would not be considered as significant for navigation. This leaves the node with only two significant regions, meaning it is not really a relevant decision point. In contrast, the node representing the T-junction in Fig. 4.1(b) has three rather significant accessible regions, which means it is a true decision point.

As we assume that information about the environment stems from range sensors like laser range finders, the principal property for assessing the significance of an accessible region is its shape. However, no detailed description of the shape is available in the AGVG (e.g., no complete description of the boundaries of free space). Hence, we have to content ourselves with certain available indicators. The idea of distance and seclusion affecting the significance of a region proposed (points 3 and 4) is illustrated in Fig. 4.2. There we see three examples of niches in a corridor always corresponding

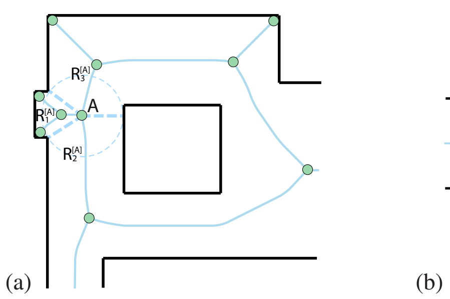


Figure 4.1: Two examples illustrating how the connections from a node to its generating points decompose free space into a set of regions

to region $R_1^{[A]}$. Intuitively, one would arguably choose the example in the middle as the one that fits the concept of a decision point best, while in the other two examples the niche and hence also Voronoi node A appear much more insignificant. The main difference between the first and second example is that one gets access to more distant areas of space as measured by the distance from the Voronoi node to the most distant node contained in the region (in this case the distance to either B or C).

However, as the comparison between the second and third example illustrates, distance alone is not sufficient as a relevance criterion. In both cases the distance to the most distant node is almost identical. The main difference is that in the third case almost the entire region is part of the corridor which is represented by the other two edges, while it is clearly secluded in the middle example. This aspect can be measured by looking at which part of the regions falls into the maximal inscribed circle of the node and which part lies outside.

As a result, we propose to measure the significance of an accessible region of a node v in an AGVG in terms of the length of the path to the most distant node belong-

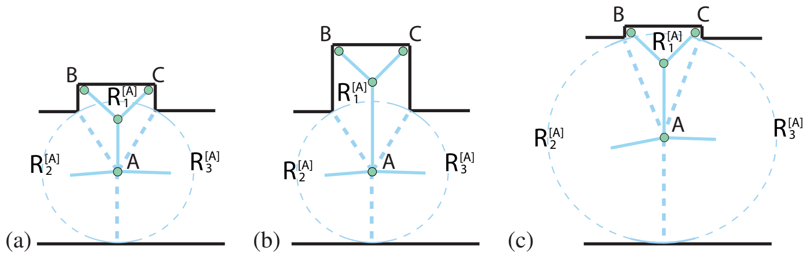


Figure 4.2: An example of how distance and seclusion influence the significance of an accessible region: The niche in (b) is clearly more significant than the examples in (a) and (c)

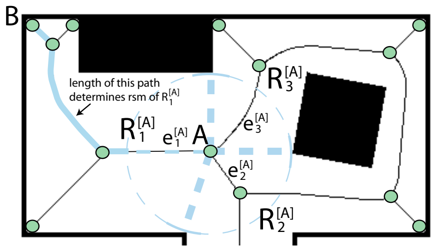


Figure 4.3: Computation of the rsm value of region $R_1^{[A]}$ to the left of node A: The length of the path to B lying within the maximal inscribed circle is subtracted from the length of the complete path to B, yielding the length of the heavy solid drawn part

ing to that region without counting that part of the path that lies inside the maximal inscribed circle of v. Assuming that the radius of the maximal inscribed circle of v is given by $r(v)$, $d(v_i, v_j)$ stands for the shortest path distance between nodes v_i and v_j in the AGVG, and the set $\mathcal{A}_i^{[v]}$ contains all nodes contained in $R_i^{[v]}$ (those nodes that can be reached from v via $e_i^{[v]}$ without passing v again), we define the *region significance measure* (rsm) as follows:

Definition 4.1 (Region significance measure)**.** *The region significance measure rsm assigns a significance value from* $\mathbb{R}_0^+$ *to each accessible region* $R_i^{[v]}$ *of a node* v*, given by*

$$rsm(R_i^{[v]}) = \begin{cases} \infty & \text{, if } e_i^{[v]} \text{ is part of a cycle} \\ \max\left\{\left(\max\limits_{w \in \mathcal{A}_i^{[v]}} d(v, w)\right) - r(v), 0\right\} & \text{, otherwise} \end{cases} \tag{4.1}$$

The first part of the definition assigns an rsm value of ∞ to accessible regions if the corresponding edge is part of a cycle. As a result, cyclic regions will be treated as maximally significant. Otherwise, the shortest paths to all the nodes belonging to the accessible region are considered to determine the length of the longest one. Finally, the radius of the maximal inscribed circle is subtracted from this path length. As a result, only the part of the path to the most distant node that lies outside the maximal inscribed circle is considered. Negative results for nodes located inside the maximal inscribed circle are turned into a relevance value of 0 by the outer maximum operation.

Figure 4.3 illustrates this approach: B is a most distant node of region $R_1^{[A]}$ in terms of shortest path length. The radius of the maximal inscribed circle is subtracted, effectively removing the dashed part of the path so that only the solid drawn part is taken into account.

Now that we have a measure to assess the significance of accessible regions purely based on the information contained in the AGVG, we can use the measure to determine

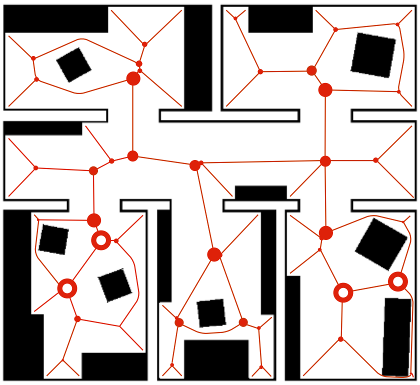


Figure 4.4: vnrm values assigned to the nodes of an AGVG depicted by the radii of the corresponding circles. Non-filled circles stand for nodes with a vnrm value of ∞

the relevance of a Voronoi node as a whole. As mentioned, to be considered relevant as a decision point for navigation, a Voronoi node needs to have at least three accessible regions of a certain significance. Two highly significant regions cannot make up for the insignificance of the third region. Also, a higher number of regions more insignificant than the third most significant region in the case of a node with degree > 3 will not increase the significance of the node. For instance, two small niches on opposing sides of the corridor will not result in a relevant decision point. Therefore, we define a *Voronoi node relevance measure* (vnrm) that assigns the significance value of the corresponding third most significant region to each inner node of an AGVG:

Definition 4.2 (Voronoi node relevance measure)**.** *The Voronoi node relevance measure vnrm* $: V \rightarrow \mathbb{R}_0^+$ *assigns each Voronoi node* v *with* $\deg(v) \geq 3$ *a relevance value given by*

$$\textit{vnrm}(v) = \max{}^3 RSM_v \tag{4.2}$$

where RSM_v *is the multiset* $\{\textit{rsm}(R_i^{[v]}) \mid 1 \leq i \leq \deg(v)\}$ *and* $\max^n$ *of a multiset* M *yields the* n*-highest value in M according to*

$$\max{}^n M = \begin{cases} \max\ M & , \textit{if}\ n = 1 \\ \max(M \ \setminus \max{}^{n-1} M) & , \textit{if}\ n > 1 \end{cases}$$

As a result of this definition, a node that has three or more edges which are part of cycles in the AGVG will be assigned a vnrm value of ∞. In Fig. 4.4, the individual vnrm values of the nodes in our exemplary environment are visualized by the radii of the corresponding circles. Nodes that have a vnrm value of ∞ are displayed by the non-filled circles. Important decision points, such as the junctions on the central corridor, receive high values while nodes caused by minor concavities in the obstacle boundaries have very low vnrm values.

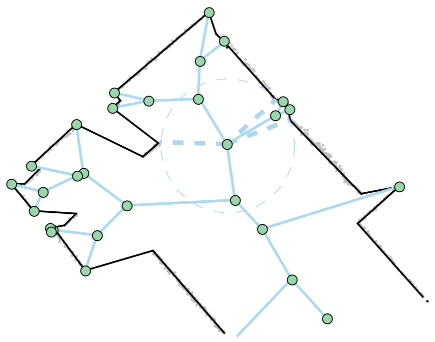


Figure 4.5: Nodes caused by noise in combination with line segmentation have very low vnrm values and, hence, can be filtered out by a low threshold value

Voronoi nodes that are the result of noise can be seen as an extreme case of nodes with an insignificant accessible region. As illustrated in Fig. 4.5, the region caused by noise on the right is almost completely contained in the maximal inscribed circle; this leads to a very low rsm value for that region. As a consequence, the relevance measure is well suited to completely identify the complete unstable subgraph. This is a considerable advantage over methods that only remove the last edge leading to the unstable end node.

4.2 Computation of Relevance Values

In this section, we present a basic algorithm for computing the rsm values for the accessible regions of Voronoi nodes in an AGVG and, derived from them, the vnrm values of the nodes themselves. We assume that we have a complete AGVG at hand. Extending the approach to deal with partial AGVGs containing unexplored edges as well as efficiency improvements and incremental versions of the basic algorithm will be discussed later in this section.

We start by describing the computation of the rsm values of the accessible regions of a single node v. Our algorithm is a modified version of Dijkstra's single source shortest path algorithm (Dijkstra, 1959), which determines the distance from a given start node to all other nodes in a weighted graph. Naturally, v is the source node and the length of the Voronoi curves annotated to the edges of the AGVG are taken as weights. The modifications we make are for detecting cyclic regions and for tracking which node has been reached via which leaving edge of the starting node v. A pseudocode version of the algorithm is given in Algorithm 1 and will be explained below. An example run is given in Fig. 4.6.

The algorithm uses a set of auxiliary variables that for simplicity are treated as global variables, although they are actually realized as attributes of the node and edge objects. These variables are

- rsm_i (initialized to 0), which contain the current estimates of the rsm values of the accessible regions $R_i^{[v]}$ and, thus, in the end contain the result of the algorithm,
- cyc_i (initialized to false), which are set to true when region $R_i^{[v]}$ is detected to be cyclic,
- d_{v_i}, storing the current shortest distance of v_i from v,
- m_{v_i} (initialized to 0), which are used to mark nodes based on which accessible region of v they belong to ($\mathrm{m}_{v_i} = 0$ means that this node has not been reached yet),
- closed_{v_i} (initialized to false), which, as in the standard Dijkstra algorithm, mark if a node has been closed, meaning its distance from v has been determined,
- a set L, which contains the nodes that have been reached and still need to be expanded.

Initialization In the initialization step of the algorithm the auxiliary variables are initialized as stated above. L is initially empty. For v, m_v is set to -1, which is a special mark only used for the start node, d_v is set to 0, and v is marked as closed. Then a single expansion step is performed for v by calling the subprocedure expand (to be explained in detail below) for each leaving edge of v. This expansion puts each node adjacent to v into the list L and marks it with the number i of the edge $e_i^{[v]}$ by which it has been reached. In addition, the distance values d_{v_i} are updated according to the length of the edge. The region numbers will be propagated through the graph, in addition to the minimal distances being computed in the main part of the algorithm. Figure 4.6(b) illustrates the state after the initialization step: v has been expanded. Its neighbors are now contained in L (depicted by the surrounding circles) and are marked with the region numbers 1 to 3. All nodes without a number still are marked as 0. The rsm_i variables for v are still 0.

Main Loop The main loop operates similarly to the standard Dijkstra algorithm: Node w with the minimal current distance d_w is taken from L and expanded. Neighboring nodes are labeled with the same region number as w and put into L. In addition, as the minimal distance of w has hereby been determined, we update the rsm_i variable for the corresponding region of v according to Eq. 4.1. Figure 4.6(c) and (d) show this step for nodes A and B, respectively, resulting in rsm values of 5 and 7 based on the length of the edges. After expanding B, L now contains five nodes: two for region $R_1^{[v]}$, one for $R_2^{[v]}$, and two for $R_3^{[v]}$.

Expand Procedure The expand procedure performs the expansion of one edge connecting the currently considered node x with another node y. There are three main cases that need to be distinguished:

1. $\mathrm{m}_y = 0$: This means node y has not been reached yet. In this case y will be marked as belonging to the same region as x (m_y is set to i); its distance d_y will be set to d_x plus the length $l(e)$ of the connecting edge e. Finally, y will be added to L.

2. $\mathrm{m}_y > 0$ and $i = \mathrm{m}_y$: y has already been reached and is labeled with the same region number as the one propagated from x. If the distance d_x plus the length of the connecting edge is smaller then the current distance d_y, we have discovered a new shortest path to y, and d_y is updated.

3. $m_y > 0$ and $i \neq m_y$: Again, y has already been reached, but this time is labeled with a different region number than the propagated one. This means a cycle involving two leaving edges of v has been detected. Accordingly, the cyc_i flags are set for both involved regions of v, signaling that no further expansion is required for these regions. Their rsm_i variables are set to ∞.

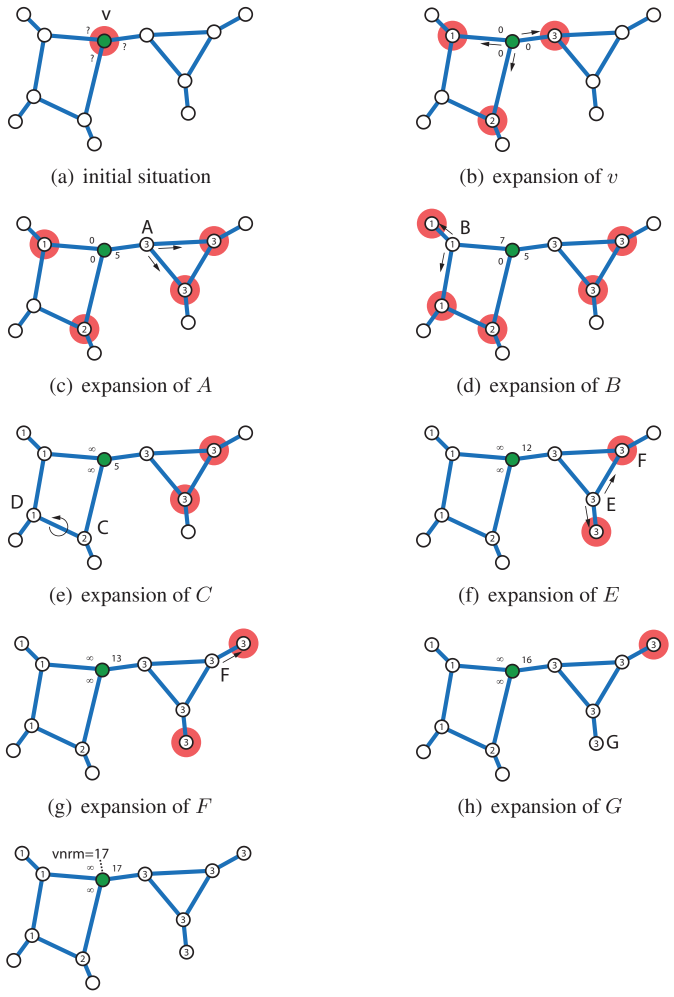
We have already seen the first case applied in Figs. 4.6(b)–(d). In Fig. 4.6(e), C is expanded and a cycle is detected (case 3) because the adjacent node D is labeled with the region number 1, not 2. As a result, rsm_1 and rsm_2 are set to ∞ and expansion stops in this part of the graph.

In Fig. 4.6(f), E is expanded, which leads to an updated rsm variable for region 3 because we have determined the shortest path to a new most distant node. Here, we encounter an instance of case 2 because E's neighbor F is labeled with the same region number. As the distance via E is larger than the already known distance for E, no distance update takes place.

Figures 4.6(g)–(i) show three more expansion steps in region 3. In the last two expansions, no more nodes are added to L; only rsm_3 is updated. Eventually, L is empty and the rsm_i variables contain the rsm values of all three regions. The third highest value (in this case 17) is the vnrm value of v.

In the actual implementation of the algorithm the list L is realized as a priority queue sorted by increasing distance d_{v_i}. As for the standard Dijkstra algorithm, the time complexity is $O(|E| + |V| \log |V|)$ because of the need to access nodes in order of increasing distance from the start node. The fact that an expansion is stopped when a region has been shown to be cyclic improves the efficiency in practice, especially in graphs with many cycles.

Computation of the rsm and vnrm values for all nodes in the graph changes the problem into an all-pair shortest path problem in which the shortest paths from every node to every other node have to be determined. No algorithm for this problem is known that has a better worst-case complexity than what we get when applying Dijkstra's single source shortest path algorithm for every node independently. This results



(a) initial situation

(b) expansion of v

(c) expansion of A

(d) expansion of B

(e) expansion of C

(f) expansion of E

(g) expansion of F

(h) expansion of G

(i) after the computation

Figure 4.6: Relevance value computation for node v performed by the basic algorithm

Algorithm 1 Basic relevance computation algorithm for a Voronoi node v

```
procedure computeRelevanceValues(Node $v$)
 1: rsm$_i \leftarrow 0$, for all $1 \leq i \leq \deg(v)$          // initialization
 2: cyc$_i \leftarrow$ false, for all $1 \leq i \leq \deg(v)$
 3: closed$_{v_i} \leftarrow$ false, for all $1 \leq i \leq |V|$
 4: m$_{v_i} \leftarrow 0$, for all $1 \leq i \leq |V|$
 5: $L \leftarrow \emptyset$
 6: m$_v \leftarrow -1$; d$_v \leftarrow 0$; closed$_v \leftarrow$ true
 7: for all $e_i^{[v]}, 1 \leq i \leq \deg(v)$ do
 8:     expand($e_i^{[v]}$,$v$,$i$)
 9: end for
10: while $|L| > 0$ do          // main loop
11:     $w \leftarrow \mathrm{argmin}_{l \in L}\, d_l$
12:     $L \leftarrow L \setminus w$
13:     if not closed$_{\mathrm{m}_w}$ and not cyc$_{\mathrm{m}_w}$ then
14:         closed$_w \leftarrow$ true
15:         rsm$_{\mathrm{m}_w} = \max\{\mathrm{d}_w - r(v), 0\}$
16:         for all $i, 1 \leq i \leq \deg(w)$ do
17:             expand($e_i^{[w]}$,$w$,m$_w$)
18:         end for
19:     end if
20: end while
21: vnrm$_v \leftarrow \max^3\{\mathrm{rsm}_i \mid 1 \leq i \leq \deg(v)\}$

procedure expand(Edge $e$, Node $x$, Integer $i$)
 1: $y \leftarrow$ other$(e, x)$
 2: if not closed$_y$ then
 3:     if m$_y \neq 0$ and $i \neq$ m$_y$ then
 4:         cyc$_i \leftarrow$ true; rsm$_i = \infty$
 5:         cyc$_{\mathrm{m}_y} \leftarrow$ true; rsm$_{\mathrm{m}_y} = \infty$
 6:     else if m$_y = 0$ or d$_x + l(e) <$ d$_y$ then
 7:         d$_y \leftarrow$ d$_x + l(e)$; m$_y \leftarrow i$
 8:         $L \leftarrow L \cup \{y\}$
 9:     end if
10: else if m$_x = -1$ then
11:     cyc$_i \leftarrow$ true; rsm$_i \leftarrow \infty$
12: end if
```

in an $O(|V|^2 \log |V| + |E||V|)$ time complexity. Accordingly, we execute our relevance computation method for each node with degree 3 or higher. The computed rsm and vnrm values are stored as annotations to the nodes and edges in the graph for later use. Later in this chapter we will present efficiency improvements of the algorithm that only compute rsm values as far as they are really required by the simplification algorithm described in the next section. In addition, we provide an incremental update procedure which avoids unnecessary recomputation of rsm values when an AGVG is constructed incrementally.

4.3 Voronoi Graph Simplification

After developing the techniques to assess the relevance or stability of a Voronoi node, we turn to the simplification algorithm that in its basic version takes an AGVG G and a relevance threshold θ and transforms G into a simplified AGVG G' in which less relevant parts have been removed.

The coarser AGVG G' computed by the simplification algorithm has the following properties:

1. G' is connected (assuming G was connected).
2. All inner nodes remaining in G' have a vnrm value $\geq \theta$.
3. All inner nodes remaining have an edge for each accessible region with rsm value $\geq \theta$.
4. G' contains at least two nodes (assuming G had at least two nodes) and no node has a degree of 2.
5. The result of the algorithm is uniquely determined assuming no two regions have the exact same rsm values.

A pseudocode version of the basic simplification algorithm is given in Algorithm 2. The algorithm starts with a list of all inner nodes of G. It iterates through the list, taking the node with the smallest vnrm value from the list and processing it, until all nodes have been processed or only two nodes remain in the graph. Processing a node consists of going through its accessible regions in order of increasing rsm values. Regions with an rsm value smaller than the given threshold θ are pruned until all such regions have been removed or the node has been reduced to degree 2. Pruning a region $R_i^{[v]}$ consists of removing all nodes and edges that are accessible from v via edge $e_i^{[v]}$. Since cyclic regions have an rsm value of ∞, the pruned region always is a separated subgraph. It may itself contain cycles though. All nodes pruned from G are also removed from the node list L. In case the pruning has indeed reduced v to degree 2, it is removed from G and the two adjacent nodes are connected by a new edge whose attributes are derived from the two replaced edges.

Algorithm 2 AGVG simplification algorithm

```
procedure simplify (AGVG G, $\theta$)
 1: $L \leftarrow \{v \in V(G) \mid \deg(v) \geq 3\}$
 2: while $|L| > 0$ and $|V(G)| \geq 2$ do
 3:     $v \leftarrow \mathrm{argmin}_{l \in L}\ \mathrm{vnrm}(l)$
 4:     $L \leftarrow L \setminus v$
 5:     repeat
 6:         $n \leftarrow \mathrm{argmin}_{1 \leq i \leq \deg(v)}\ \mathrm{rsm}(R_i^{[v]})$
 7:         $r \leftarrow \mathrm{rsm}(R_n^{[v]})$
 8:         if $r < \theta$ then
 9:             prune subgraph accessible via $e_i^{[v]}$
10:             remove all pruned nodes from $L$
11:         end if
12:     until $\deg(v) < 3$ or $r \geq \theta$
13:     if $\deg(v) = 2$ then
14:         remove $v$ together with its edges and connect nodes adjacent to $v$
15:     end if
16: end while
```

In Fig. 4.7, we apply the simplification algorithm to a part of the simple polygonal environment used throughout this work. Figure 4.7(a) shows the original AGVG with the rsm values annotated to the edges that correspond to the nodes' third most relevant region. Thus, the numbers also correspond to the nodes' vnrm values. As this particular graph does not contain nodes with degree > 3, they also represent all regions that are candidates for pruning.

The first pruning step is shown in Fig. 4.7(b): The node with lowest vnrm value is taken from the queue and the region with rsm value 7.4 is pruned. As a result the degree of the node is reduced to 2 and the node is removed together with its two edges and replaced by a new edge connecting its remaining two neighbors. Figure 4.7(c) shows the next pruning steps up to an rsm value of 30.7. Similarly, Figs. 4.7(d) and (e) show the results for threshold values of 60 and 100, respectively. Even for higher threshold values no further simplification would take place because the graph now only contains two nodes. In Figs. 3.8 and 3.9 of Chap. 3 we already saw some examples of applying this approach to real environments, and more will follow in Chap. 7.

Pruning subtrees is a linear operation, and instead of removing pruned nodes from L, we use an additional flag to mark nodes that have been pruned. These are then ignored when taken from the list. Hence, the dominating part is sorting the nodes in increasing order, and the algorithm has a worst-case time complexity of $O(n \log n)$, where n is the number of nodes in the AGVG.

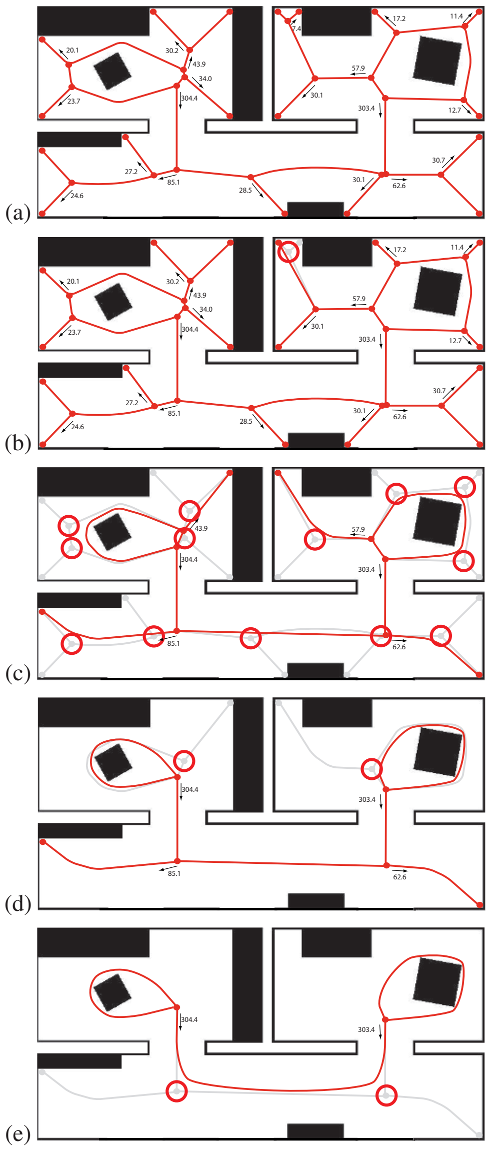


Figure 4.7: Steps of the simplification algorithm for a simple polygonal environment

4.4 HAGVG Construction

In the previous section, we described a basic version of the simplification algorithm that removes irrelevant parts of the original AGVG based on a given threshold value. In order to construct a hierarchical Voronoi graph representation, this algorithm has to be modified in two ways:

1. Instead of destructively modifying the given AGVG, it constructs a new coarser AGVG which is then added as a new layer.
2. In addition, it generates links between the given AGVG and the derived coarser AGVG.

Given an AGVG L_1 and a set of n threshold values $\theta_1, \theta_2, ..., \theta_n$, one can then construct an HAGVG with $n+1$ layers by iteratively applying the modified algorithm to L_i and θ_i for $1 \leq i \leq n$.

The links between two layers of an HAGVG realize the corresponding abstraction relations. Downward links allow changing to a finer level of granularity, while upward links can be used to change to a coarser level of representation. A node or edge on the higher level can correspond to a set of nodes and arcs on the lower level. In our implementation, the downward links always connect nodes and edges on the higher level to the same nodes and edges on the lower level. For the upward links, we have to distinguish two basic cases as shown in Fig. 4.8:

1. When region $R_i^{[v]}$ is pruned from a node v (Algorithm 2, line 9), all nodes and edges reachable via $e_i^{[v]}$ need to be linked to the node representing v on the higher level (Fig. 4.8(a)).
2. When a node v with degree 2 is removed (Algorithm 2, line 14), the node and its edges $e_1^{[v]}$ and $e_2^{[v]}$ need to be linked to the new edge constructed from $e_1^{[v]}$ and $e_2^{[v]}$ on the higher level (Fig. 4.8(b)).

As nodes with pruned regions can themselves be removed by further pruning, we first propagate information about removed parts before actually changing the upward links. The whole modified simplification algorithm performs the following steps:

1. Create a clone G' of the input AGVG G and link nodes and edges in G' with corresponding nodes and edges in G (downward links).
2. Attach a list $l_{v_i'}$ initialized as $\{v_i\}$ to each node v_i' in G'; do the same for all edges e_i' in G'.
3. Perform simplification on G':

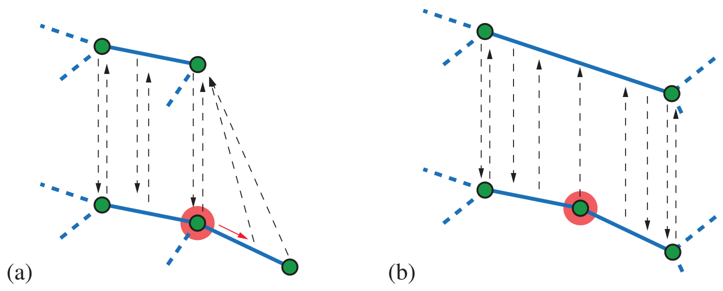


Figure 4.8: Two cases that need to be distinguished when constructing the links between two layers: (a) pruning of a region and (b) removal of a node with degree 2

- When pruning a region of v'_i (case 1), set $l_{v'_i}$ to the union of itself and all lists from the pruned nodes and edges.
- When removing a node v'_i of degree 2 (case 2) resulting in a new edge e'_j, set $l_{e'_j} = l_{v'_i} \cup l_{e_1^{[v'_i]}} \cup l_{e_2^{[v'_i]}}$.

4. For each remaining node or edge x in G', link all elements from G in l_x with x (upward links).

4.5 Admitting Incomplete Information

In the context of mobile robot mapping, most of the time no complete AGVG is available as we assumed in the previous sections. However, we still want to be able to compute coarser route graph layers from the partially constructed global AGVG that the robot has built up so far. In addition, hierarchical localization based on a matching scheme that utilizes the relevance values of the nodes in both the map and the local AGVG requires us to compute the relevance values for the inevitably incomplete local AGVGs.

In both cases, we still employ the same algorithm we applied in the case of complete information. All nodes of degree 1 that mark the end of a still unexplored edge are treated in the same way as the corner nodes. However, without knowing the complete AGVG, the values computed from an incomplete or local AGVG will often be only lower bound estimates of the actual relevance values. Every rsm value computed for a non-cyclic region in which the local AGVG has unexplored edges will be a lower bound of the real rsm value and will be marked as such by introducing a new boolean attribute lb$(R_i^{[v]})$.

For the vnrm value we can distinguish two cases:

1. The third most relevant region or an even less relevant region of the node has an rsm value which is a lower bound estimate. In this case, the vnrm value as given

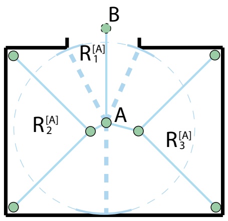


Figure 4.9: Computation of the relevance values for a local AGVG: While the rsm values for regions $R_2^{[A]}$ and $R_3^{[A]}$ of node A can be computed precisely, the value computed for $R_1^{[A]}$ only yields a lower bound on the actual rsm value because it is not known how the GVD continues behind B

by the rsm value of the third most relevant region is only a lower bound estimate as well.

2. If all rsm values from the third most relevant region to the least relevant one are known precisely, the vnrm value of the node is known precisely as well.

If the vnrm value of a node v is only a lower bound estimate, we record this fact using the additional node attribute $\mathrm{lb}(v)$. Both cases are illustrated in Fig. 4.9. It shows a local AGVG for a position in a room with an open door. Node B marks a point at which the course of the underlying GVD cannot be determined further because the GVD structure beyond B is unknown. As a result, the rsm value computed for region $R_1^{[A]}$ by treating B as a corner node is only a lower bound estimate. On the other hand, the rsm values for regions $R_2^{[A]}$ and $R_3^{[A]}$ can be computed exactly because they do not contain unexplored edges. The fact that these values are higher than the estimate for $R_1^{[A]}$ means that $\mathrm{vnrm}(A) = \mathrm{rsm}(R_1^{[A]})$ is also only a lower bound estimate. If either $\mathrm{rsm}(R_2^{[A]})$ or $\mathrm{rsm}(R_3^{[A]})$ would have been smaller than $\mathrm{rsm}(R_1^{[A]})$, $\mathrm{vnrm}(A)$ would have been known exactly because the third highest rsm value would have been known.

How nodes for which the vnrm value is a lower bound estimate should be treated depends on the task at hand. In our simplification algorithm, nodes with vnrm values marked as lower bounds are excluded from being considered for pruning, i.e., they are treated like nodes with a vnrm value of ∞. This approach is advantageous in this context because it still allows us to use the simplification to filter out unstable parts of an AGVG without accidentally removing important nodes which just happen to be close to a visibility boundary.

For our mapping approach, the existence of lower bound estimates of nodes in the partially constructed AGVG means that these need to be updated whenever new parts are added to a partially constructed AGVG. This will often allow us to replace the old estimates by better or even precise lower bound values. Incremental updating of the relevance values is further discussed in Sect. 4.7.

4.6 Improving the Efficiency of the Relevance Computation

As mentioned, computing the rsm and vnrm values for all nodes in an AGVG has an $O(|V|^2 \log |V| + |E||V|)$ worst-case time complexity. However, as we have also seen, determining the vnrm value of a node only requires the precise rsm values for the third most relevant regions and the even less relevant regions. Similarly, the simplification algorithm never operates on the two most relevant regions when it prunes an AGVG. Therefore, in this section and the next, we discuss improvements of the value computation based on two related ideas:

- As simplification and computation of vnrm values does not require the rsm values of the two most relevant regions for a given node, rsm value computation can be stopped as soon as it becomes clear which two regions are the most relevant ones.
- When AGVGs are constructed incrementally and the current AGVG is complemented by a new subgraph, it is not necessary to recompute relevance values for nodes for which only the two most relevant regions have changed.

As a result of improvements made based on these ideas, the relevance computation becomes very fast in practice. A quantitative analysis of this improvement is given in Chap. 7.

When looking at the basic relevance computation algorithm (Algorithm 1), one observation is that not only does each individual rsm$_i$ variable increase monotonically during the computation, but so does the sequence of updated rsm values computed in line 15 of Algorithm 1. This results from the fact that nodes are expanded in order of increasing distance from the start node. The only exception is when cyclic regions are detected and their rsm values are set to ∞.

As a consequence, assuming that no cyclic regions have been found so far and we have just expanded the last remaining node of a region, we only have to proceed if there are more than two regions which still have remaining unexpanded nodes. Otherwise, these other two regions have to be the most relevant ones (cyclic or not) and it is not necessary to compute their exact rsm values.

One additional complication arises from the fact that each detected cyclic region takes away one place for a region for which we do not need to compute the rsm value. When one cyclic region has been found, we only can have one additional open region. When two or more have been found, the values of all remaining regions have to be computed. Thus, given a variable openRegions that counts the number of regions for which the rsm value has not yet been determined and a variable cyclicRegions counting the number of regions that have been determined to be cyclic so far, the criterion for when computation can be stopped can be formulated as follows:

$$\text{openRegions} \overset{?}{=} \max\{2 - \text{cyclicRegions}, 0\} \tag{4.3}$$

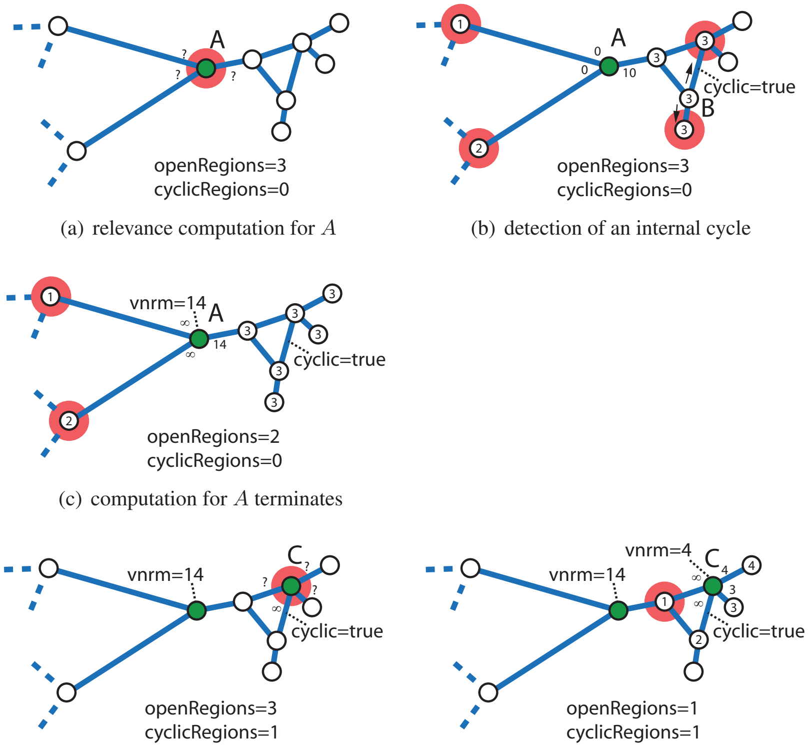
The improved relevance computation algorithm given in Algorithms 3 and 4 is based on this criterion. The openRegions variable is initialized with the degree of the start node, while the cyclicRegions variable is initialized with 0. They are continuously updated whenever cyclic regions are detected or non-cyclic regions are closed. Detecting closure of a region is realized by maintaining counter variables openNodes_i for each region. They count the number of nodes that have been marked for this region but have not been expanded yet.

The computation is terminated when the number of open regions becomes 0 or when the termination criterion (Eq. 4.3) is satisfied. As a side product of this approach, we can now directly determine the vnrm value of the node from the termination conditions. For instance, when we have just closed a non-cyclic region $R_i^{[v]}$ and the termination criterion is satisfied, it follows that $\text{vnrm}(v) = \text{rsm}_i$. When the termination criterion is met and open regions remain, we set their rsm value to ∞ to make sure that they are treated correctly by the simplification algorithm.

Besides these modifications, another efficiency improvement is made which is not reflected in the pseudocode: Whenever it becomes clear that an edge is part of a cycle in the graph, this fact is stored permanently as an attribute $\text{cyclic}(e)$ of this edge (in contrast to the cyc_i variables, which are reinitialized for each call of the value computation algorithm). When relevance computation for a node v starts and one of its leaving edges has the cyclic attribute set to true, the corresponding value can immediately be set to ∞ and no further expansion is needed for this edge.

In Fig. 4.10, we give an example of how this improved value computation works. At first, the value computation algorithm is applied to node A. The nodes belonging to regions 1 and 2 are far away from A; thus all expansion takes place in region 3. In Fig. 4.10(b), node B is expanded. An internal cycle within region 3 is detected and the cyclic attribute for the corresponding edge is set to true. The expansion proceeds until the last node of region 3 has been expanded in Fig. 4.10(c). This means that openNodes_3 has just become 0 and hence the number of open regions is reduced by 1. Now the termination criterion is checked and indeed, as we only have two more open regions and no cyclic region found so far, it is satisfied and the vnrm value of A is set to the value of rsm_3.

The next node for which the relevance computation algorithm is called is node C, which is of degree 4 (Fig. 4.10(d)). We already know that $e_2^{[C]}$ is part of a cycle and therefore rsm_2 is set to ∞. The number of open regions is decremented and the number of cyclic regions is incremented. In this situation, the termination criterion says that we need to compute the rsm values of two more regions. This happens for regions 3 and 4 (Fig. 4.10(e)). When closing region 4 the criterion is met and computation terminates setting vnrm_C to 4, which is the rsm value of region 4.



(a) relevance computation for A

(b) detection of an internal cycle

(c) computation for A terminates

(d) relevance computation for C

(e) computation for C terminates

Figure 4.10: Illustration of the improved relevance computation algorithm: (a)–(c) show the computation for node A, while (d) and (e) show the computation for node C, which reuses the information that one of its edges belongs to a cycle

Algorithm 3 Main procedure of the improved relevance computation algorithm

```
procedure computeRelevanceValues(Node $v$)
1: rsm$_i$ ← 0, for all $1 \le i \le \deg(v)$          // initialization
2: cyc$_i$ ← false, for all $1 \le i \le \deg(v)$
3: openNodes$_i$ ← 0, for all $1 \le i \le \deg(v)$
4: openRegions ← $\deg(v)$
5: cyclicRegions ← 0
6: closed$_{v_i}$ ← false, for all $1 \le i \le |V|$
7: m$_{v_i}$ ← 0, for all $1 \le i \le |V|$
8: $L \leftarrow \emptyset$
9: m$_v$ ← $-1$; d$_v$ ← 0; closed$_v$ ← true
10: for all $e_i^{[v]}, 1 \le i \le \deg(v)$ do
11:     expand($e_i^{[v]}$,$v$,$i$)
12: end for
13: while $|L| > 0$ and openRegions $> 0$ do          // main loop
14:     $w \leftarrow \mathrm{argmin}_{l \in L}\, \mathrm{d}_l$
15:     $L \leftarrow L \setminus w$
16:     if not closed$_w$ and not cyc$_{\mathrm{m}_w}$ then
17:         closed$_w$ ← true
18:         openNodes$_{\mathrm{m}_w}$ ← openNodes$_{\mathrm{m}_w}$ $- 1$
19:         rsm$_{m_w}$ $= \max\{\mathrm{d}_w - r(v), 0\}$
20:         $i \leftarrow 1$
21:         while $i \le \deg(w)$ and openRegions> 0 do
22:             expand($e_i^{[w]}$,$w$,m$_w$)
23:             $i \leftarrow i + 1$
24:         end while
25:         if not cyc$_{\mathrm{m}_w}$ and openNodes$_{\mathrm{m}_w}$ $= 0$ then
26:             openRegions ← openRegions $- 1$
27:             if openRegions $= \max\{2 -$ cyclicRegions$, 0\}$ then
28:                 vnrm$_v$ ← rsm$_i$
29:                 openRegions ← 0
30:                 for all $j, 1 \le j \le \deg(v)$ do
31:                     if not cyc$_j$ and openNodes$_j$ $> 0$ then rsm$_j$ ← $\infty$ end if
32:                 end for
33:             end if
34:         end if
35:     end if
36: end while
```

Algorithm 4 Expand procedure of the improved relevance computation algorithm

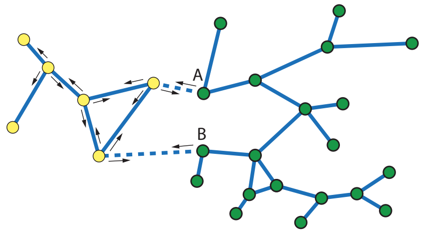
```
procedure expand(Edge $e$, Node $x$, Integer $i$)
 1: $y \leftarrow$ other$(e, x)$
 2: if not closed$_y$ then
 3:     if $m_y \neq 0$ and $i \neq$ m$_y$ then
 4:         if not cyc$_i$ then
 5:             cyc$_i \leftarrow$ true; rsm$_i = \infty$
 6:             openRegions $\leftarrow$ openRegions $- 1$
 7:             cyclicRegions $\leftarrow$ cyclicRegions $+ 1$
 8:         end if
 9:         if not cyc$_{\mathrm{m}_y}$ then
10:             cyc$_{\mathrm{m}_y} \leftarrow$ true; rsm$_{\mathrm{m}_y} = \infty$
11:             openRegions $\leftarrow$ openRegions $- 1$
12:             cyclicRegions $\leftarrow$ cyclicRegions $+ 1$
13:         end if
            if openRegions $= 0$ then vnrm$_v \leftarrow \infty$ end if
14:     else
15:         if m$_y = 0$ then openNodes$_i \leftarrow$ openNodes$_i + 1$ end if
16:         if m$_y = 0$ or d$_x + l(e) <$ d$_y$ then
17:             d$_y \leftarrow$ d$_x + l(e)$, m$_y \leftarrow i$
18:             m$_y \leftarrow i$
19:             $L \leftarrow L \cup \{y\}$
20:         end if
21:     end if
22: else if m$_x = -1$ then
23:     cyc$_i \leftarrow$ true; rsm$_i \leftarrow \infty$
24:     openRegions $\leftarrow$ openRegions $- 1$
25:     cyclicRegions $\leftarrow$ cyclicRegions $+ 1$
        if openRegions $= 0$ then vnrm$_v \leftarrow \infty$ end if
26: end if
```

4.7 Incremental Computation

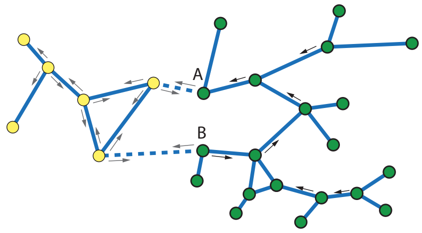
In the context of incremental mapping and HAGVG construction, further computation time can be saved by not recomputing all rsm and vnrm values when the current AGVG is complemented by new information. Our approach for appending a new AGVG H to our current AGVG G is the following:

1. For each node and each accessible region, variables are used to track whether the corresponding rsm and vnrm values need to be recomputed. For all nodes of H these are set to true; for all nodes of G they are initially set to false.
2. G and H are connected and nodes of G for which an unexplored edge has just been connected are stored in a list L.
3. For each node of L the regions corresponding to the now connected edges are marked for recomputation.
4. For each node v of L a depth-first search is started (not including the newly connected edges), and all non-cyclic regions encountered that have to contain v are marked for recomputation.
5. All nodes of G with degree ≥ 3 for which at least one of the regions marked for recomputation is not among the two most relevant regions are marked for recomputation.
6. The relevance computation algorithm is called for all nodes which are marked for recomputation.

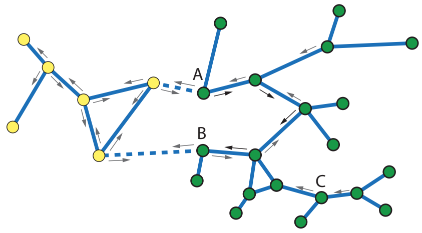
This basic approach is illustrated in Fig. 4.11: Figure 4.11(a) shows the two graphs (the dark nodes are the nodes of G, while the others belong to H) and the previously unexplored but now connected edges as dashed lines. The arrows mark regions which have been marked for recomputation. These are the regions from nodes in H and the regions corresponding to the just-connected edges in G. In Fig. 4.11(b), we see the result of propagating recomputation marks from node A. Doing the same for node B in Fig. 4.11(c) leads to more regions being marked for recomputation. At node C, the corresponding region is already marked for recomputation, at which point the depth-first search backtracks so that each region is only considered once. Hence, determining the nodes that require recomputation requires linear time. Finally, Fig. 4.11(d) shows the nodes which require recomputation: Node A, for instance, requires recomputation because (among other things) its third most relevant region has been extended. In contrast, for node B the two most relevant regions have changed and no recomputation is needed.



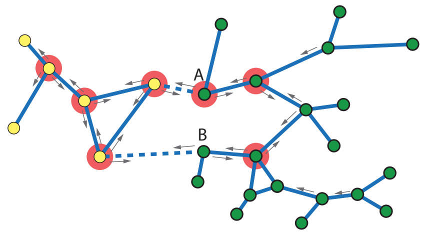
(a) connecting the graphs



(b) propagating recomputation marks for A



(c) propagating recomputation marks for B



(d) nodes that require recomputation

Figure 4.11: Incremental value computation scheme: Nodes that require recomputation are determined by looking at which regions would require recomputation

4.8 Application Scenarios

We close this chapter by briefly looking at possible application scenarios in which the techniques developed in this chapter could be employed. The intended application in our case, of course, is incremental construction and maintenance of an HAGVG as a spatial representation for a mobile robot. In addition, the approach can be employed to filter out unstable parts of an AGVG. Furthermore, as a side product the approach can be used to construct route graphs from vector data like floor plans.

4.8.1 Incremental HAGVG Construction

For constructing and maintaining an HAGVG representation the number of layers in the HAGVG and the corresponding threshold values are currently determined in advance. The bottom layer is maintained by appending newly perceived nodes and edges and by updating the relevance values as described in Sect. 4.7. The higher levels are currently recomputed completely when new information is complemented. So far, no need to increase the efficiency by developing an incremental procedure has been encountered in practice.

4.8.2 Removal of Unstable Parts

For removing unstable nodes and edges caused by noise, one can employ the simplification algorithm with a rather low threshold value and only keep the resulting AGVG. Figure 4.12 demonstrates this approach. It shows in Figs. 4.12(a) and 4.12(b) the AGVGs for the noise ratio of a real laser scanner used and for an unrealistically high noise ratio, respectively. As is clearly visible, the high noise ratio together with the segmentation of the range data results in a high number of additional nodes and edges in the AGVG. In both cases, the simplification resulted in the route graph shown in Fig. 4.12(c), with only slight variations in the exact positions of the nodes. Further testing by reducing the threshold value revealed that spurious nodes and edges caused by the noise can very reliably be filtered out by using a very small threshold value ($\theta < 100mm$), though, of course, nodes and edges caused by existing concavities of the same order of magnitude will be filtered out as well.

4.8.3 Automatic Route Graph Generation from Vector Data

Although this is not the intended area of application, our approach can be used to automatically generate route graphs from vector data, for instance, representing floor plans. However, it should be mentioned that the presence of a complete geometric description allows for formulating improved measures that take other properties of the regions into account. It also may allow the derivation of semantic information (e.g., a decomposition into particular rooms and corridors), leading to route graphs that are better suited for human usage.

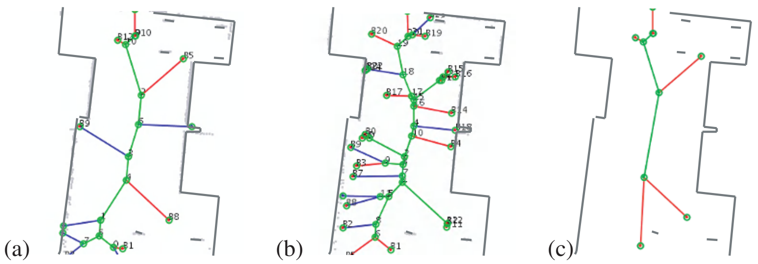


Figure 4.12: Simulation of route graph generation for different noise levels: (a) shows the AGVG constructed with a range sensor with low sensor noise and (b) with high. Identical coarse route graphs are computed from both AGVGs via simplification (c)

To illustrate the applicability of our approach we used the 2D model of a floor in the MZH building of the Universitä Bremen shown in Fig. 4.13. Computing the AGVG and applying the simplification algorithm afterwards resulted in the depicted route graph. This route graph could, for instance, serve as a starting point for generating route descriptions leading from a given location to a specific goal. Extracting or manually annotating additional semantic information (doors, hallways, etc.) should facilitate the automatic generation of route instructions, at least in a simple navigation system-like language. The number of nodes here has been reduced from 350 in the original AGVG to 73 in the simplified route graph, and the number of edges from 353 to 76.

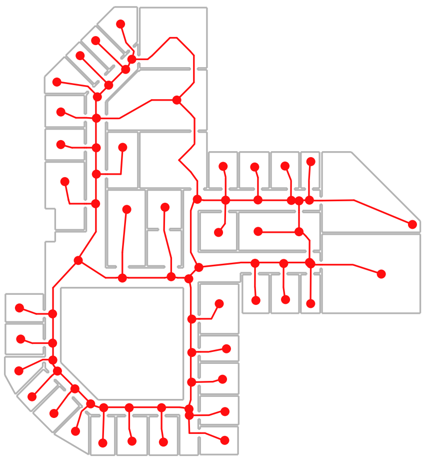


Figure 4.13: A coarse route graph computed by the simplification algorithm from a ground plan of a floor in the MZH building, Universität Bremen

In this chapter, we were concerned with the problem of extracting an AGVG representation from sensor data and of autonomously building up a complete HAGVG from it. With regard to adding new information to the HAGVG we made the assumption that the correct correspondences between nodes and edges are given. In the next chapter, we turn to the question of how to proceed when this is not the case.

Chapter 5

Voronoi Graph Matching for Data Association

In this chapter, we will be concerned with the problem of identifying correct correspondences between an AGVG extracted from local information about the robot's surroundings and the memorized map AGVG. Identifying the correct correspondences is a crucial step for complementing the map AGVG incrementally by inserting newly perceived parts from the local AGVGs and for localization in general. We develop a matching approach based on the notion of edit distance that compares the graph structures and the annotations of the AGVGs. Results from the previous chapter with regard to assessing the stability of Voronoi nodes are incorporated into the matching process in order to adequately deal with potential variations in the AGVGs.

5.1 The Data Association Problem

In the data association step of a robot mapping approach, features from the current local observation are identified with features in the agent's spatial model. Depending on the particular approach, the features can be single reflection points in the range data of a laser scanner, corners, line segments or more complex geometric structures, landmarks, or, as in our case, the nodes and edges making up an AGVG. The question always is, which feature from the local perception and which feature from the spatial model correspond to the same object in the physical world? The problem of determining the correct correspondences is referred to as the *data association problem* or *correspondence problem* in robotics and is considered as "arguably one of the most challenging problems in SLAM" (Bailey et al., 2006). Figure 5.1 illustrates the problem as it typically occurs in robot mapping: The features from the model have been mapped into observation space. Now the correct matching as indicated by the connecting lines has to be determined. In probabilistic approaches this is typically done by taking into account the uncertainty in the feature positions indicated by the ellipses in the figure.

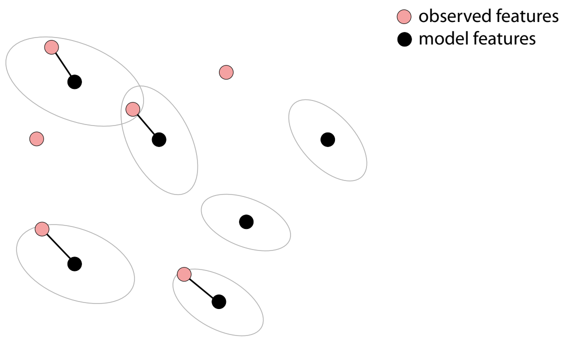


Figure 5.1: The data association problem: The currently observed features have to be assigned to corresponding features in the map. Some observed features might not be part of the map yet. Uncertainty in the positions of the map features is indicated by the ellipses

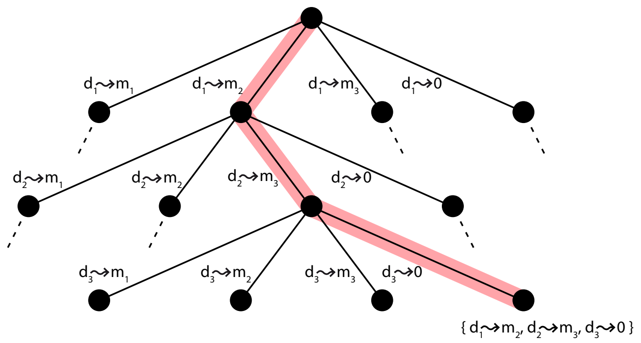
The individual demands on a data association algorithm depend on the overall mapping framework it is embedded in. For instance, multiple hypothesis approaches in which data association has to be performed for many different hypotheses have a bias towards computationally efficient solutions, while for single hypothesis approaches it is usually advisable to trade off efficiency for reliability. Other important aspects are the similarity distribution and density of features in the environment.

We proceed by formally describing the data association problem and reviewing the most common approaches in the context of mobile robot mapping.

5.1.1 Data Associations and the Interpretation Tree

For the following considerations, we assume that the local observation is given by a *data set* $D = \{d_1, d_2, ..., d_m\}$ containing m features. A second set of features, the *model set* $M = \{m_1, m_2, ..., m_n\}$, represents the features stored in our map or, alternatively, in a representation of the previous observation. Generally, a data association can be seen as a relation between the elements of D and those of M. However, in most mapping scenarios no general m-to-n mappings between the feature sets are possible. If one assumes that no perceived feature can correspond to multiple model features—as we will do in this work—a data association can be defined as a total function from the index set of D to the index set of M extended by 0, which will be used for features that are regarded as new and not yet contained in M:

Definition 5.1 (Data association)**.** *A data association between a given data set* $D = \{d_1, d_2, ..., d_m\}$ *and a given model set* $M = \{m_1, m_2, ..., m_n\}$ *is a total function* $da : \{1, 2, ..., m\} \rightarrow \{0, 1, 2, ..., n\}$ *that maps the indices of the features from* D *to those of features from* M *or to* 0*.* $da(i) = j$ *means that* d_i *is associated with* m_j *and* $da(i) = 0$ *means that* d_i *is not associated with any feature from* M*.*



$d_1 \rightsquigarrow m_1$
$d_1 \rightsquigarrow m_2$
$d_1 \rightsquigarrow m_3$
$d_1 \rightsquigarrow 0$
$d_2 \rightsquigarrow m_1$
$d_2 \rightsquigarrow m_2$
$d_2 \rightsquigarrow m_3$
$d_2 \rightsquigarrow 0$
$d_3 \rightsquigarrow m_1$
$d_3 \rightsquigarrow m_2$
$d_3 \rightsquigarrow m_3$
$d_3 \rightsquigarrow 0$
$\{d_1 \rightsquigarrow m_2, d_2 \rightsquigarrow m_3, d_3 \rightsquigarrow 0\}$

Figure 5.2: Part of the interpretation tree for a given data and model set, both consisting of three features. A path from the root to a leaf describes a particular data association

Alternatively to writing $da(i) = j$, we will use the notation $d_i \rightsquigarrow m_j$, and use $d_i \rightsquigarrow 0$ instead of $da(i) = 0$. An individual assignment $d_i \rightsquigarrow m_j$ will also be referred to as a *matching* or *pairing*. We will use the term *partial data association* for the case where not all elements from D have been assigned yet, so that da is a partial function.

The set of all possible data associations in a given situation can be represented as a so-called *interpretation tree* (Grimson, 1990), as depicted in Fig. 5.2. At each level of the tree, a particular element of D is associated either with an element of M or with 0. Thus, the branching factor at each node is $n + 1$ and the overall number of possible data associations is $(n + 1)^m$. The leaves of the interpretation tree represent complete data associations determined by the assignments made along the path from the root node to the leaf. Inner nodes represent partial data associations. A data association algorithm has to select one of these possibilities as the solution to the data association problem.

If we consider Voronoi nodes of AGVGs as the features making up the data and model sets, a further restriction is appropriate: As Voronoi nodes are point features that can be either perceived or not, but never be partially perceived (e.g., because of occlusion), no two perceived Voronoi nodes can correspond to the same one in the model set. This restriction has been termed the *all-different* constraint in the literature. Consequently, for a data association to satisfy the all-different constraint, the mapping to the elements of M has to be injective. Naturally, this restriction does not concern the elements that are mapped to 0.

Definition 5.2 (All-different constraint). *A data association da between a data set $D = \{d_1, d_2, ..., d_m\}$ and a model set $M = \{m_1, m_2, ..., m_n\}$ satisfies the all-different constraint if*

$$da(i) = k \wedge da(j) = k \implies i = j \vee k = 0 \tag{5.1}$$

holds.

Even with the all-different constraint in place, the number of possible data associations from which the most plausible one has to be determined is still $\sum_{i=1}^{m} \binom{m}{i}\binom{n}{i} i!$ (Wolter, 2008). Hence, additional information needs to be exploited in order to come up with feasible solutions to the data association problem. Examples of information that has been employed for this purpose are feature similarity, position estimates, co-visibility, and configurational knowledge. We provide a brief overview of the basic techniques relevant for this text in the next section.

5.1.2 Data Association Approaches

In order to determine the compatibility of an observed feature and a map feature in a non-stochastic approach when the position from which the observation was made is known, one would typically consider the Euclidean distance between the features after transferring both into the same coordinate system. In EKF-based stochastic mapping approaches (see Appendix A.2.2), in which the map consists of a mean vector for the robot pose and feature positions together with the corresponding covariance matrix, the analogous approach is to determine the compatibility by way of their Mahalanobis distance (Mahalanobis, 1936). The Mahalanobis distance is the point distance between two vectors scaled by their statistical variation.

Definition 5.3 (Mahalanobis distance)**.** *Assuming d_i and m_j stand for the respective means of a data feature and a map feature, μ and Σ are the mean vector and covariance matrix of the map, and the measurement model for feature m_j is given by $g_j(x)$ plus additive Gaussian noise with 0 mean and covariance matrix R_i, the (squared) Mahalanobis distance $M^2(d_i, m_j)$ of d_i and m_j is given by:*

$$M^2(d_i, m_j) = (d_i - g_j(\mu))^T \bar{\Sigma}_{ij}^{-1} (d_i - g_j(\mu)) \tag{5.2}$$

where

$$\bar{\Sigma}_{ij} = G_j \Sigma G_j^T + R_i$$

and

$$G = \left. \frac{\partial g_j}{\partial x} \right|_{\mu}$$

$\bar{\Sigma}_{ij}$ here is the covariance of the so-called innovation $d_i - g(\mu)$. $M^2(d_i, m_j)$ forms a chi-squared probability distribution. In the *validation gate test* (Bar-Shalom & Fortmann, 1988), a threshold χ^2 is used for the Mahalanobis distance and all associations with $M^2(d_i, m_j) \leq \chi^2$ are assessed as compatible. χ^2 is chosen so that a certain percentage of the probability mass lies between 0 and χ^2. The concrete value depends on the dimensionality of the feature vectors.

In Fig. 5.1, the validation gates for the map feature are visualized by the ellipses. An observation is compatible with a map feature if it falls into the corresponding ellipse. The figure also shows that data association using the validation gate is usually

Algorithm 5 The individual compatibility nearest neighbor algorithm

```
procedure ICNN($D = \{d_1, d_2, ..., d_m\}, M = \{m_1, m_2, ..., m_n\}$)
1: for all $i = 1$ to $m$ do
2:     $k \leftarrow \underset{1 \leq j \leq n}{\operatorname{argmin}} M^2(d_i, m_j)$
3:     if $M^2(d_i, m_k) < \chi^2$ then
4:         $\mathrm{da}(d_i) \leftarrow k$
5:     else
6:         $\mathrm{da}(d_i) \leftarrow 0$
7:     end if
8: end for
```

still ambiguous: An observation may be compatible with multiple map features or multiple observations may be compatible with the same map feature. One simple approach to resolve such ambiguities often used in combination with validation gating is nearest neighbor assignment. This means each observed feature d_i is associated with the closest map feature m_j in terms of $M^2(d_i, m_j)$ (or Euclidean distance in a non-stochastic setting) that is compatible. Hence, the time complexity of the approach is $O(mn)$ where m is again the number of observed features and n is the number of features in the map. Neira & Tardós (2001) call the combination of validation gating and nearest neighbor assignment the *individual compatibility nearest neighbor* (ICNN) algorithm. As we will use a variant of this approach for comparison later in this work, we provide a pseudocode version in Algorithm 5.

A simple approach sometimes used to enforce the all-different constraint in combination with the ICNN algorithm is to only consider map features which have not been paired previously. However, this approach often yields suboptimal solutions with regard to the overall distance between matched features. Finding the optimal data association complying to the all-different constraint can be formulated as a maximum weight matching problem in a bipartite graph and be solved by the Hungarian method in $O(mn^2)$ time where n is the number of nodes and m is the number of edges in the bipartite graph (Kuhn, 1955; Munkres, 1957).

As pointed out by Neira & Tardós (2001), testing individual compatibility of pairings neglects the fact that observations obtained from the same position are correlated and, hence, accepts hypotheses that contain mutually exclusive pairings. Avoiding this problem requires us to consider the complete set of simultaneously observed features, a procedure often called *batch association*. Neira and Tardós propose a batch association approach called *joint compatibility branch and bound* (JCBB), which is a branch and bound search for the hypothesis with the largest number of jointly compatible pairings. The downside of this approach is the increased computational costs caused by considering an exponentially sized space of hypotheses. The authors give an empirically determined estimate of $O(1.53^n)$ for the complexity of their algorithm, where n is the number of features in the map.

Another approach to batch association is to employ hard constraints based on the configuration of features to filter out data associations which are not jointly compatible. For instance, Wolter (2008) shows that optimal matching with general m-to-n correspondences can be computed in $O(m^2n^2)$ time when the matching is subject to cyclic order constraints. Relative geometric constraints, which play an important role in our approach, have been employed by multiple authors (e.g., Arras et al., 2003; Bailey, 2001; Lim & Leonard, 2000). For instance, Bailey (2001) employs binary distance constraints between pairs of features that have to be preserved by the data association. The solution is based on a maximum clique search in the correspondence graph formed by compatible pairings of features.

As we see, many data association approaches are based on graph-theoretical methods. Graph matching approaches in particular are very relevant for our case, as the AGVGs we are concerned with are annotated graph structures (see Bunke, 2000; Bunke & Messmer, 1997, for a general overview on graph matching). Many important graph matching problems have been shown to be NP-complete, such as exact subgraph isomorphism (Garey & Johnson, 1979) or inexact graph matching and inexact subgraph matching (Abdulkader, 1998). However, for some restricted types of graphs polynomial solutions do exist. For instance, the graph isomorphism problem has been shown to be solvable in polynomial time for planar graphs (Hopcroft & Wong, 1974).

In the remainder of this chapter, we develop an AGVG matching approach for data association in which we exploit the fact that graph topology and combinatorial embedding restrict the set of valid data associations. Our approach is based on inexact graph matching techniques because we need to take into account variations resulting from the instability of the underlying GVD. We employ the notion of edit distance (Eshera & Fu, 1986; Sanfeliu & Fu, 1983) to describe the similarity of AGVGs. Matching based on edit distance has been applied to structural shape descriptions similar to Voronoi graphs in the area of computer vision (e.g., Sebastian et al., 2004). Our approach can be seen as an adaptation of this work to AGVG representations and to the data association problem in which local graph and model graph may only partially overlap. In addition to the edit distance-based matching, absolute position estimates and relative geometric constraints are applied to further reduce the set of considered matchings.

5.2 AGVG Matching Based on Ordered Tree Edit Distance

In the context of mapping with Voronoi graph representations, data association involves the identification of node and edge features computed from local data with node and edge features from either the map or the previously computed local Voronoi graph. A typical situation showing two consecutively perceived AGVGs overlaid based on an estimate of the robot pose is shown in Fig. 5.3. The circles depict the maximal inscribed circles for inner Voronoi nodes, while the crosses mark the generating points. The thinly drawn edges are unexplored ones. Each AGVG contains branches that are not contained in the other AGVG.

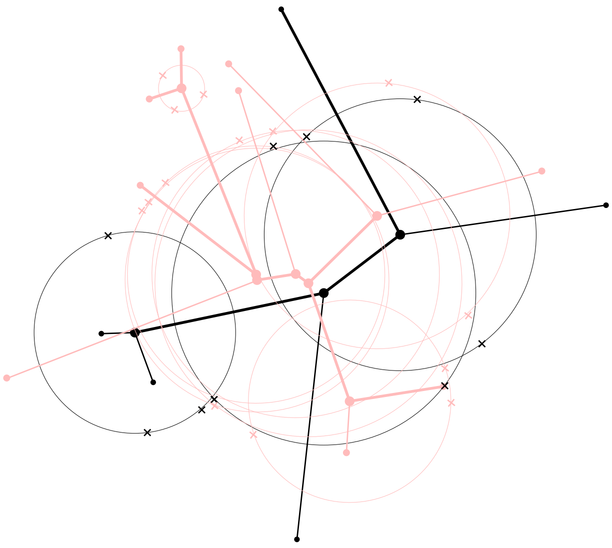


Figure 5.3: A typical AGVG matching problem: The local and the model AGVG each contain branches not contained in the other. In addition, node positions, positions of generating points (crosses), and radii of the inscribed circles vary

Besides providing position estimates for observed features and map features, the AGVG representation offers additional information that can be exploited in order to improve the data association algorithms. This includes

- node signature information,
- information about the graph topology and embedding in the plane,
- stability information, and
- geometric information about the local configurations of nodes.

The uncertainty of global position estimates can grow without bounds (e.g., while moving along a large loop in the environment) and become less useful for identifying the correct data association. Configurational information based on relative geometric relations holding between features perceived simultaneously or in short order, on the other hand, is generally much better suited to identify the right data association. As the distribution of Voronoi nodes is typically rather dense and signature information is usually insufficient to reliably identify nodes, an approach that makes heavy use of structural information (graph topology and combinatorial embedding) and geometric constraints is particularly promising.

As a result of the instability problems of Voronoi graphs, however, it cannot be expected that the overlapping parts of the two compared AGVGs are completely isomorphic with regard to the graph structure. Hence, our approach will be based on inexact graph matching techniques. It will employ the concept of edit distance, which captures the costs of transforming one AGVG into the other. Furthermore, we restrict ourselves to the matching of trees because tree matching problems can in most cases be solved more efficiently than the corresponding graph matching problems[1], and this does not significantly restrict the applicability of the approach. In practice, we enforce acyclicity in the AGVGs by splitting the most distant edges that belong to a cycle.

Resulting from the fact that our AGVGs are combinatorially embedded, our matching is one between ordered trees. This is a crucial property for our approach because computing the edit distance for unordered labeled trees has been shown to be NP-complete (Zhang et al., 1992), while it is in P for ordered trees (Zhang & Shasha, 1989). We start the presentation of our data association algorithm by first defining the underlying tree matching together with the required edit operations and their costs. In later sections, we extend this approach by incorporating additional constraints.

5.2.1 Ordered Tree Matching Based on Edit Distance

The basic idea of the edit distance approach for inexact tree (or graph) matching is to define costs for operations that modify the trees (e.g., operations for adding or deleting nodes and edges), and compute a cost-minimal sequence of operations that transforms both trees into the same tree. A sequence of transformation operations directly implies a matching between the nodes (and edges) of the two graphs. An example of this idea involving two trees A and B is given in Fig. 5.4. Assuming that the only edit operation allowed is deleting a complete branch, and that the cost for this operation is the sum of nodes in the removed subtree, cutting off branch a in A and branch b in B is the cost-optimal solution for transforming both trees into the same tree, and, hence, their edit distance is 2.

In our approach to matching two tree-formed AGVGs G and G', we start by considering all possible initial matches of a node u from G with a node u' from G'. This choice turns the AGVGs into *rooted trees* with respective root nodes u and u', as shown in Fig. 5.5. In addition to being rooted, the fact that for each node the edges are cyclically ordered as specified by the nodes' combinatorial embedding means that the resulting trees also are *ordered*. Hence, if u and u' have the same degree n, there are n ways of mapping the edges of u to those of u' while preserving the cyclic orders, a process that we will refer to as *aligning*. The different alignment variants can then be specified by providing a single edge offset parameter. For each start matching $u \rightsquigarrow u'$ and edge offset i, the edit distance can be computed from the edit distances of matching the corresponding subtrees of u and u'. Thus, the key component of our approach

[1]For instance, tree isomorphism and subtree isomorphism problems can be solved in polynomial time (Aho et al., 1974).

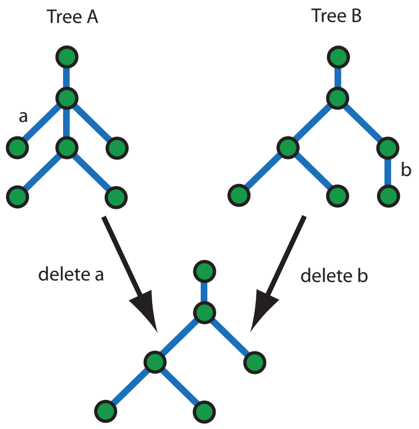


Figure 5.4: Tree matching based on edit distance: The trees A and B can be transformed into the same tree by deleting branch a in tree A and branch b in tree B

is an edit distance $\text{dist}_{\text{subtree}}$ for subtrees in an AGVG, and we start by defining the required edit operations, their costs, and how the subtree edit distance is computed.

5.2.1.1 Edit Distance for Subtrees in AGVGs

In the remainder of this chapter, we will use the following notations: For nodes of the observed data AGVG G we use u, v, w, etc., while using u', v', w', etc. for nodes from the model AGVG G'. We will denote a subtree with root v and ancestor u as $\triangle_v^{(u)}$. Providing the ancestor u here only serves the purpose of specifying which of v's edges leads upward in the tree, namely the one leading to u, and which ones lead to v's child nodes. Hence, we put the ancestor label in brackets.

Given two subtrees and assuming that their roots correspond, the correspondences between their edges are uniquely determined. We write $\text{child}_i(v)$ for the ith child node of node v in a given subtree which is the node connected to the ith successor edge of the edge connecting v to its parent in the cyclic edge order.

For the root node u of a rooted AGVG, only the order of child nodes is determined, but it is not clear which adjacent node should be considered as the first one. For such cases we introduce the function $\text{neighbor}_i(v) = \text{other}(e_i^{[v]}, v)$ to refer to the node connected to v via edge $e_i^{[v]}$. In addition, we will use the notation $i \oplus j$ for the index of the jth successor of the element with index i in a cyclically ordered set. For instance, $\text{child}_{(2\oplus 2)}(v)$ for a node with three children is equivalent to $\text{child}_1(v)$. $\text{neighbor}_{(1\oplus 2)}(v)$ with $\deg(v) = 3$ is equivalent to $\text{neighbor}_3(v)$.

Figure 5.6 shows an example situation of two subtrees $\triangle_v^{(u)}$ and $\triangle_{v'}^{(u')}$ with roots v and v' that should be compared. We consider three different matching cases which

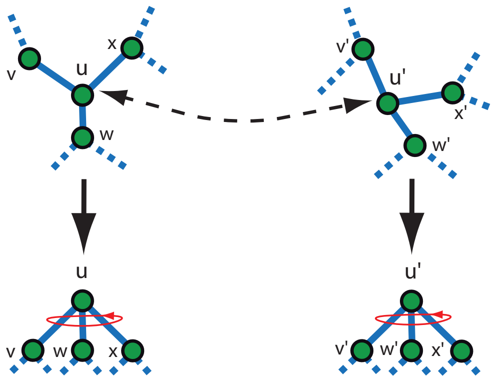


Figure 5.5: Choosing u and u' as the roots of two tree-formed AGVGs results in two rooted and ordered trees. If u and u' both have degree n, there are n possible mappings of subtrees of u to subtrees of u'

directly correspond to our edit operations. The names of the operations are chosen from the perspective of modifications made to the subtree of the local AGVG.

1. **Match operation:** The match operation implies that the two root nodes v and v' will be considered as matched. This means that no modification is required, but the costs for matching each of the subtrees formed by the child nodes of v to those of the corresponding child nodes of v' have to be considered. We assume here that v and v' have the same degree and, hence, the same number of child nodes. As we will see later, we will reject matchings for which this is not the case beforehand (cf. Sect. 5.3.1).

2. **Delete operation:** With this operation we conclude that node v is not contained in the model and, hence, it has to be deleted from subtree $\triangle_v^{(u)}$ together with all the subtrees formed by its child nodes except one. This remaining child subtree will then need to be matched to the model subtree $\triangle_{v'}^{(u')}$. Consequently, the delete operation actually subsumes $\deg(v) - 1$ different cases of modification, one for each subtree of v. To compute the minimal costs, all cases have to be considered and the one that results in the lowest costs has to be chosen.

3. **Add operation:** The add operation is symmetrical to the delete operation. We assume here that the root node of the model subtree has not been perceived, so that this node would have to be added to the local AGVG before node v. Analogously to the delete case, now one of the child subtrees of v' needs to correspond to $\triangle_v^{(u)}$, while all other subtrees would have to be added together with v'. Again, all cases have to be considered to determine the one that results in minimal costs.

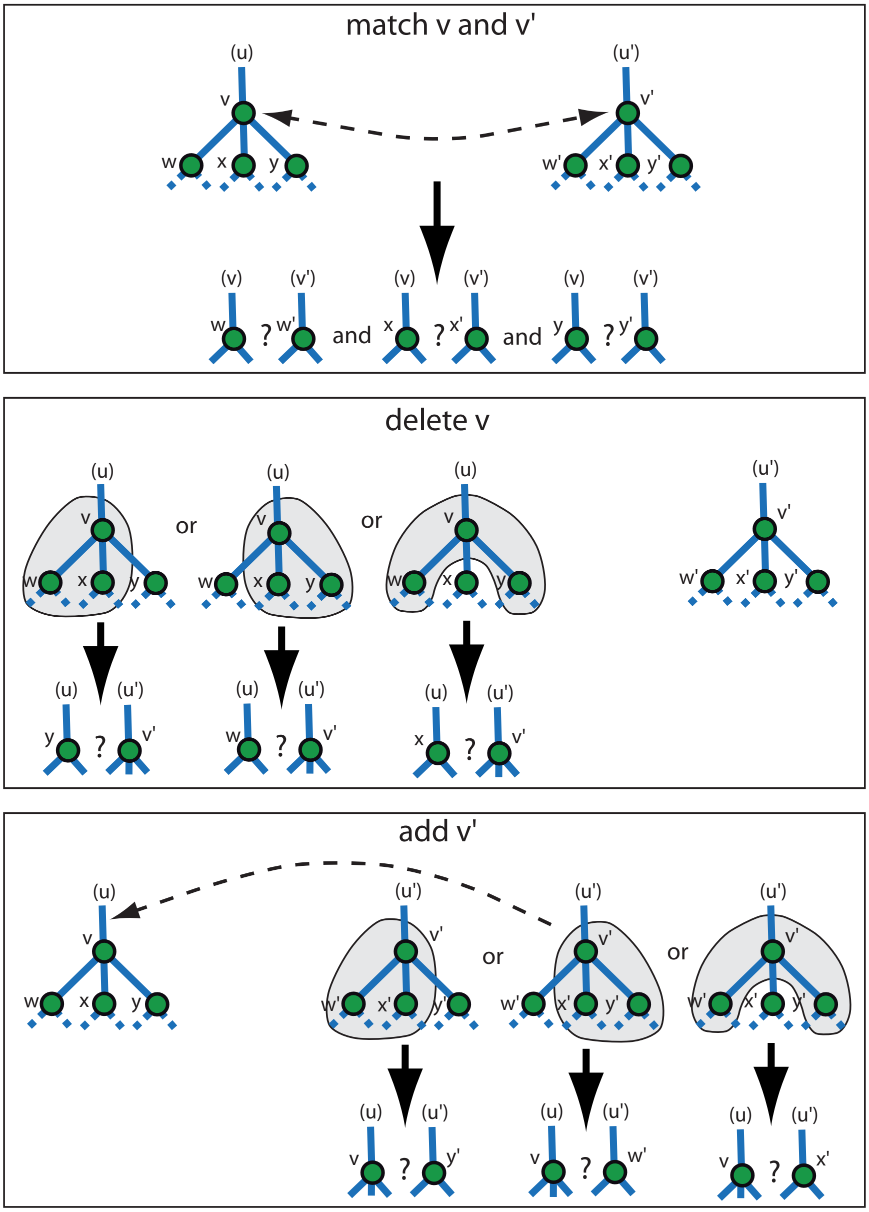


Figure 5.6: The three edit operations for matching subtrees from two AGVGs: matching the root nodes, deleting the root of the local AGVG, and adding the root of the map AGVG

Based on the three operations described above, we can now recursively define the edit distance of two rooted and ordered subtrees $\triangle_v^{(u)}$ and $\triangle_{v'}^{(u')}$. First, the edit distance $\text{dist}_{\text{subtree}}(\triangle_v^{(u)}, \triangle_{v'}^{(u')})$ is the minimum over the costs resulting from applying the individual operations: $\text{dist}_{\text{match}}(\triangle_v^{(u)}, \triangle_{v'}^{(u')})$ for the costs of the match operation, $\text{dist}_{\text{del}}(\triangle_v^{(u)}, \triangle_{v'}^{(u')})$ for the delete operation, and $\text{dist}_{\text{add}}(\triangle_v^{(u)}, \triangle_{v'}^{(u')})$ for the add operation. In addition, the recursion terminates if either v or v' marks the end of a still not completely explored edge. We introduce the node attribute $\text{unknown}(v)$ to indicate if this is the case. If $\text{unknown}(v)$ is true, the costs for matching the subtrees are simply 0. If $\text{unknown}(v')$ is true, we want the costs to be based on $\triangle_v^{(u)}$ in order to penalize for leaving parts of the local AGVG unmatched. For now, we only introduce a cost function $\text{unmatched}(\triangle_v^{(u)})$. This function will be defined later in this chapter. The overall $\text{dist}_{\text{subtree}}$ edit distance then is

$$\text{dist}_{\text{subtree}}(\triangle_v^{(u)}, \triangle_{v'}^{(u')}) =$$
$$\begin{cases} 0 & \text{, if unknown}(v) \\ \text{unmatched}(\triangle_v^{(u)}) & \text{, if unknown}(v') \\ \min\left\{\text{dist}_{\text{match}}(\triangle_v^{(u)}, \triangle_{v'}^{(u')}), \text{dist}_{\text{del}}(\triangle_v^{(u)}, \triangle_{v'}^{(u')}), \text{dist}_{\text{add}}(\triangle_v^{(u)}, \triangle_{v'}^{(u')})\right\} & \text{, otherwise} \end{cases} \tag{5.3}$$

The edit distance $\text{dist}_{\text{match}}(\triangle_v^{(u)}, \triangle_{v'}^{(u')})$ resulting from applying the match operation is the sum of the edit distances resulting from matching corresponding child subtrees:

$$\text{dist}_{\text{match}}(\triangle_v^{(u)}, \triangle_{v'}^{(u')}) = \sum_{i=1}^{\deg(v)-1} \text{dist}_{\text{subtree}}\left(\triangle_{\text{child}_i(v)}^{(v)}, \triangle_{\text{child}_i(v')}^{(v')}\right) \tag{5.4}$$

The edit distance $\text{dist}_{\text{del}}(\triangle_v^{(u)}, \triangle_{v'}^{(u')})$ for the delete operation is defined based on two components: First, $\text{removal}_j(v)$ are the costs for removing the subtree corresponding to the jth child of v. We will later define these costs based on the rsm values of the corresponding accessible region introduced in Chap. 4. Second, we have to add the costs of matching one of the child subtrees to $\triangle_{v'}^{(u')}$ as given by $\text{dist}_{\text{subtree}}$. To determine the best case, we have to consider all $\deg(v) - 1$ possibilities and for each possibility sum up the costs for removing the other child subtrees. Then we add the costs for the recursive matching to it. Deletion is not possible if $\deg(v) = 1$ because we have arrived at an end node where a Voronoi curve ends in a corner of the environment. In this case the costs for the delete operation will be ∞. $\text{dist}_{\text{del}}(\triangle_v^{(u)}, \triangle_{v'}^{(u')})$

then is defined as follows:

$$\text{dist}_{\text{del}}(\triangle_v^{(u)}, \triangle_{v'}^{(u')}) =$$
$$\begin{cases} \min\limits_{1\le i\le \deg(v)-1} \left\{ \underbrace{\left(\sum_{j=1,j\neq i}^{\deg(v)-1} \text{removal}_j(v)\right)}_{\text{costs for deleting all except } i\text{th subtree}} + \underbrace{\text{dist}_{\text{subtree}}\left(\triangle_{\text{child}_i(v)}^{(v)}, \triangle_{v'}^{(u')}\right)}_{\text{costs for matching } i\text{th subtree}} \right\} & \text{, if } \deg(v) > 1 \\ \infty & \text{, otherwise} \end{cases} \tag{5.5}$$

The costs for the add operation are computed accordingly, this time employing $\text{addition}_j(v')$ for the costs of adding the entire jth child subtree of v' to $\triangle_v^{(u)}$. Similarly to the delete case, the costs will be ∞ if $\deg(v') = 1$ as no terminal node can be inserted in the middle of an edge.

$$\text{dist}_{\text{add}}(\triangle_v^{(u)}, \triangle_{v'}^{(u')}) =$$
$$\begin{cases} \min\limits_{1\le i\le \deg(v')-1} \left\{ \underbrace{\left(\sum_{j=1,j\neq i}^{\deg(v')-1} \text{addition}_j(v')\right)}_{\text{costs for adding all except } i\text{th subtree}} + \underbrace{\text{dist}_{\text{subtree}}\left(\triangle_v^{(u)}, \triangle_{\text{child}_i(v')}^{(v')}\right)}_{\text{costs for matching } i\text{th subtree}} \right\} & \text{, if } \deg(v') > 1 \\ \infty & \text{, otherwise} \end{cases} \tag{5.6}$$

Clearly, when computing the minimal distance for different choices of start nodes, we have a problem with overlapping subproblems (the computation of $\text{dist}_{\text{subtree}}$ for a particular subtree) and optimal structure ($\text{dist}_{\text{subtree}}$ is computed recursively from the edit distances of smaller subtrees). Hence, we employ a dynamic programming approach to compute the overall best matching. The already computed $\text{dist}_{\text{subtree}}$ values for pairs of subtrees from the observed AGVG and the model AGVG are stored either in an array or in a hash table. As each edge in an AGVG defines two subtrees, one for each orientation of the edge, the maximal number of entries is $2 \times e \times 2 \times f$, where e is the number of edges in the observed AGVG and f in the model AGVG. In addition to the costs, we store the operation that resulted in the lowest costs, and in case of the delete or add operation we also store the parameter i which resulted in the minimal edit distance. This allows for reconstructing the sequence of operations that lead to the minimal edit distance.

We now proceed by explaining how the edit distance of subtrees is used to compute the best overall matching.

5.2.2 Overall Edit Distance

Based on the edit distance for subtrees, we can now define $\text{dist}_{\text{node}}(u, u')$, the edit distance for an initial matching of two nodes u and u'. The overall edit distance of two AGVGs $\text{dist}_{\text{AGVG}}(G, G')$ is then defined as the optimum over all possible initial matchings.

As we already mentioned, when considering a start matching of two compatible nodes u and u', both of degree n, there exist n possible ways of aligning the edges of u and u' in a way that preserves the cyclic order information. Assuming that the minimal cost for the variant with edge offset i is given by $\text{dist}_{\text{aligned}}(u, u', i)$, the costs for this start matching are given by the minimum over all possible i:

$$\text{dist}_{\text{node}}(u, u') = \min_{1 \leq i \leq \deg(u)} \text{dist}_{\text{aligned}}(u, u', i) \tag{5.7}$$

For an alignment given by edge offset i, the edit distance $\text{dist}_{\text{aligned}}(u, u')$ is given by the sum of the costs of transforming the subtrees of corresponding edges into each other:

$$\text{dist}_{\text{aligned}}(u, u', i) = \sum_{j=1}^{\deg(u)} \text{dist}_{\text{subtree}} \left(\triangle^{(u)}_{\text{neighbor}_j(u)}, \triangle^{(u')}_{\text{neighbor}_{(j \oplus i)}(u')} \right) \tag{5.8}$$

As mentioned, the overall edit distance is then computed as the minimal one over all possible start matchings:

$$\text{dist}_{\text{AGVG}}(G, G') = \min_{v \in V(G), v' \in V(G')} \left\{ \text{dist}_{\text{node}}(v, v') \right\} \tag{5.9}$$

The set of recursive equations Eqs. 5.3–5.9 directly describes our basic matching algorithm. Following the dynamic programming approach, every time we require a $\text{dist}_{\text{subtree}}$ value, we first check if this value has already been computed and, thus, is stored in the table. Only if this is not the case is the recursive computation triggered, and the result is then added to the table.

5.2.3 Modeling Removal and Addition Costs

As our edit distance approach is supposed to model the typical variations of AGVGs resulting from sensor limitations, the relevance and stability measures developed in Chap. 4 are well suited to describe the costs for deleting or adding subtrees. Therefore, the costs for deleting or adding subtrees of a node, $\text{addition}_j(u)$ and $\text{removal}_j(u)$, are determined by way of the rsm values of the corresponding accessible regions $R^{[u]}_j$:

$$\text{addition}_j(u) = \text{removal}_j(u) = \text{rsm}(R^{[u]}_j) \tag{5.10}$$

As we store the rsm values as attributes to the edges, we do not need to consider the other nodes contained in the subtrees that are supposed to be removed. Hence, computing the complete sums in Eqs. 5.5 and 5.6 only requires $O(k)$ time with $k = \deg(v)$.

The costs unmatched($\triangle_v^{(u)}$) for leaving a complete subtree unmatched when the other subtree consists of an unknown node are determined in the same way: We simply add up the rsm values of the accessible regions $R_i^{[v]}$ which correspond to the ith child of v:

$$\text{unmatched}(\triangle_v^{(u)}) = \sum_{i=1}^{\deg(v)-1} \text{rsm}(R_i^{[v]}) \tag{5.11}$$

By choosing the penalty costs for unmatched parts to be of the same order of magnitude as the costs for removal and addition operations, we prevent that the matching algorithm prefers solutions with only very little overlap between the two AGVGs. This approach is advantageous in our case because for applications like incremental mapping or tracking of Voronoi nodes the overlap between compared AGVGs is generally very large. For other kinds of applications, a different approach for balancing the individual costs might be better suited.

5.2.4 Optimizations

The top-down dynamic programming approach allows restricting the computation of $\text{dist}_{\text{subtree}}$ values to those that are really required for the AGVGs at hand. This is usually only a fraction of the maximal entry number. In addition, this approach facilitates the utilization of branch and bound techniques to further increase the efficiency of the edit distance computation. This is done at two different levels of the computation:

1. At the global level, we pass the edit distance of the best complete solution found so far down the recursion. As soon as the accumulated costs pass the current optimal solution, the computation backtracks to a higher level.
2. At the local level, when computing the best operation for a given subtree, we also pass the costs of the best operation tested so far on to the computation of the other operations. Again, this often allows stopping computation early. Since a successful match operation often results in lower cost solutions, we test this operation before trying the delete and add operations.

An additional heuristic we use to improve the efficiency of the branch and bound approach concerns the order in which we generate the start matchings in $\text{dist}_{\text{AGVG}}$. Since nodes with a high vnrm value will typically result in higher cost solutions when not matched, it is advantageous to attempt finding assignments for them early on. Therefore, we generate the start matchings in order of decreasing vnrm values.

5.2.5 Complexity

The exploitation of the ordered tree structure in the matching process results in a data association algorithm with a rather low time complexity considering that the general interpretation tree is exponentially sized. As mentioned above, the size of a table storing all dist$_{\text{subtree}}$ values is $4 \times e \times f$ for a local AGVG with e edges and a model AGVG with f edges. As we consider trees, this lies in $O(mn)$ where m and n are the number of nodes in the trees, respectively. Computing a table entry when the entries of the required subtrees are already known requires $O(k^2)$ time, where k is the maximal node degree occurring in both trees (typically 3). The quadratic dependency on k results from the computation of dist$_{\text{del}}$ and dist$_{\text{add}}$ (see Eqs. 5.5 and 5.6). Hence, setting up the complete table can be done in $O(mnk^2)$ time.

The computation of dist$_{\text{node}}$ from a set up table takes $O(k^2)$ time because k table entries have to be added for k ways of aligning the edges of the two nodes. Finally, the overall dist$_{\text{AGVG}}$ function for matching two trees requires $m \times n$ times the computation of dist$_{\text{node}}$. As a result, the overall time complexity of the matching remains in $O(mnk^2)$.

As k can be considered constant, the resulting $O(mn)$ time complexity of the matching algorithm is that of the same order as of the standard nearest neighbor algorithm. In addition, it only yields data associations that comply to the constraints arising from graph topology and combinatorial embedding. Moreover, because of the top-down approach and optimizations described in the previous section, it is usually sufficient to compute a small fraction of all table entries. The effects of these additional efficiency improvements are hard to grasp theoretically and, hence, are not reflected in the coarse complexity assessment given here. As we will see in the next section, a further reduction of matchings that actually need to be considered can be achieved by incorporating hard constraints based on additionally available information.

5.3 Incorporating Constraints

In the following, we extend the basic edit distance approach for matching tree-formed AGVGs by incorporating a set of hard constraints into the matching process. This exploitation of additional information further reduces the number of valid matchings, but the extensions we have to make in order to incorporate certain kinds of constraints increase the worst-case complexity of the matching approach. In exchange, we gain a significant improvement of the matching quality. The restriction to local constraints holding between neighboring entities ensures that the computational costs remain acceptable. In principle, it would have been possible to model the restrictions stemming from the additional information as soft constraints and to incorporate them as additional costs into the edit distance. However, we prefer hard constraints as they allow us to reject matchings early and no additional parameters for weighing the constraints are required.

Constraints on potential data associations can be distinguished based on their arity: *Unary constraints* apply to individual matchings of features from the data AGVG and the model AGVG based on the properties of the features. *Binary constraints*, on the other hand, concern pairs of matchings, and *ternary constraints* concern triples of matchings. The following kinds of constraints will be considered in the next sections:

1. unary distance constraints for Voronoi nodes based on estimated robot pose and unary node similarity constraints,
2. binary distance constraints holding between adjacent nodes (in terms of both, line-of-sight distance and distance based on edge length), and
3. ternary constraints about angles formed by edges.

To restrict possible matchings between observed Voronoi nodes and Voronoi nodes in the map based on estimated position, we need to introduce additional attributes to store global position estimates for the nodes in an AGVG. We assume that this is done using a probabilistic approach involving Gaussian probability distributions (cf. Appendix A.2).

5.3.1 Unary Constraints Based on Pose Estimates and Node Similarity

Unary hard constraints concern the compatibility of two features that are supposed to be matched with each other. The first condition we already mentioned previously is that the node degrees of two nodes v and v' need to agree in order to be matched:

$$\text{compatible}_{\text{deg}}(v, v') \Longleftrightarrow \deg(v) = \deg(v') \tag{5.12}$$

When absolute position estimates for the nodes are available, these can be used to restrict possible matchings in the same way as is done in the ICNN approach (cf. Sect. 5.1.2). Hence, another unary constraint we consider is that the Mahalanobis distance between two nodes v and v' is below a given threshold γ_M [2]:

$$\text{compatible}_{\text{absdist}}(v, v') \Longleftrightarrow M^2(v, v') < \gamma_M \tag{5.13}$$

As we mentioned, when comparing two nodes in a subtree the correspondences between their edges is already given. The same naturally holds for the generating points of the two nodes. As a result, whenever absolute position information is available, we demand that the Mahalanobis distance between all corresponding generating points falls below the given threshold γ_M:

$$\text{compatible}_{\text{absdist-gp}}(v, v') \Longleftrightarrow \forall k, 1 \leq k \leq \deg(v) : M^2(gp_k^{[v]}, gp_{(k \oplus i)}^{[v']}) < \gamma_M \tag{5.14}$$

$gp_k^{[v]}$ here stands for the kth generating point of v, and i is the offset between the generating points of v and v' as implied by the alignment of the subtrees.

In the next criterion, we capture the compatibility of the local geometry as described by the nodes' signature information. This information consists of the radii

[2] For simplicity, we equate nodes with their respective feature vectors here.

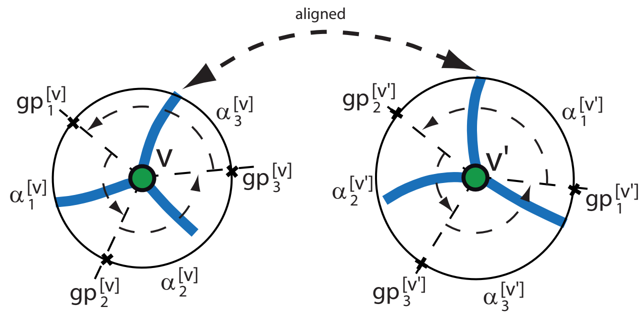


Figure 5.7: Comparison of the local geometry of Voronoi nodes: The given edge alignment allows the comparison of the angles between corresponding generating points. In addition, the difference between the radii of the maximal inscribed circles is taken into account

$r(v)$ of the nodes' respective maximal inscribed circle and angles between the connections to the generating points. Figure 5.7 illustrates how we measure the relative angles for two nodes with degree 3: As the alignment between the two nodes is given by the edge offset parameter i, generating point $gp_j^{[v]}$ of v corresponds to $gp_{(j\oplus i)}^{[v']}$ of v' ($i = 1$ in the example). Hence, we compare the angle between the generating points $gp_j^{[v]}$ and $gp_{(j\oplus 1)}^{[v]}$, $\alpha_j^{[v]}$, with the corresponding angle $\alpha_{(j\oplus i)}^{[v']}$. For the corresponding angles the difference needs to be under a given error tolerance γ_a in order to be considered compatible. The same goes for the difference in the radii of inscribed circles, resulting in the following compatibility criterion:

$$\text{compatible}_{\text{localgeom}}(v,v') \iff \begin{array}{l}\left(\forall j, 1 \le j \le \deg(v) : |\alpha_j^{[v]} - \alpha_{(j\oplus i)}^{[v']}| < \gamma_a\right) \\ \wedge\, |r(v) - r(v')| < \gamma_{\text{radius}}\end{array} \tag{5.15}$$

Finally, we consider the relevance values assigned to the nodes and their leaving edges as described in the previous chapter. The basic approach here again is to tolerate certain differences in the vnrm and rsm values. However, we have to take into account that some of the values are lower bound estimates as both AGVGs are typically incomplete. Therefore, we first define an appropriate difference measure $\text{diff}_{\text{vnrm}}$ for vnrm values that becomes 0 when the lower vnrm value is only a lower bound:

$$\text{diff}_{\text{vnrm}}(v,v') = \begin{cases} 0 & \text{, if } (\text{lb}(v) \wedge \text{vnrm}(v) \le \text{vnrm}(v')) \\ & \quad \vee (\text{lb}(v') \wedge \text{vnrm}(v') \le \text{vnrm}(v)) \\ |\text{vnrm}(v) - \text{vnrm}(v')|, \text{otherwise} & \end{cases} \tag{5.16}$$

$\text{diff}_{\text{rsm}}(R_j^{[v]}, R_j^{[v']})$ can be defined analogously. As the relevance compatibility criterion we then get

$$\text{compatible}_{\text{relevance}}(v, v') \iff \text{diff}_{\text{vnrm}}(v, v') < \gamma_{rel} \\ \wedge \left(\forall j, 1 \leq j \leq \deg(v) : \text{diff}_{\text{rsm}}(R_j^{[v]}, R_{(j\oplus i)}^{[v']})\right) \tag{5.17}$$

We subsume the unary constraints defined in Eqs. 5.12–5.17 in this section under a single compatibility criterion for nodes in a subtree called $\text{compatible}_{\text{unary}}(v, v')$. When testing compatibility of node matchings, we generally check the different criteria in order of increasing computational costs, which is reflected in the definition below.

$$\text{compatible}_{\text{unary}}(v, v') \iff \text{compatible}_{\text{deg}}(v, v') \wedge \text{compatible}_{\text{relevance}}(v, v') \\ \wedge \text{compatible}_{\text{absdist}}(v, v') \wedge \text{compatible}_{\text{absdist-gp}}(v, v') \\ \wedge \text{compatible}_{\text{localgeom}}(v, v') \tag{5.18}$$

Incorporating the unary constraint into the overall edit distance computation is straightforward. Since it concerns the matching of two (aligned) nodes, it has to be included into the definitions of $\text{dist}_{\text{aligned}}(u, u', i)$ (Eq. 5.8) and $\text{dist}_{\text{match}}(\triangle_v^{(u)}, \triangle_{v'}^{(u')})$ (Eq. 5.4).

First, modifying $\text{dist}_{\text{aligned}}(u, u', i)$ yields

$$\text{dist}'_{\text{aligned}}(u, u', i) = \\ \begin{cases} \sum_{j=1}^{\deg(u)} \text{dist}'_{\text{subtree}}\left(\triangle_{\text{neighbor}_j(u)}^{(u)}, \triangle_{\text{neighbor}_{(j\oplus i)}(u')}^{(u')}\right), & \text{if compatible}_{\text{unary}}(u, u') \\ \infty & \text{, otherwise} \end{cases} \tag{5.19}$$

The edge offset parameter used in the compatibility criteria here is directly given by i. As $\text{compatible}_{\text{deg}}$, $\text{compatible}_{\text{absdist}}$, and $\text{compatible}_{\text{abdist-gp}}$ do not depend on the edge alignment, they can be included in the definition of $\text{dist}_{\text{node}}(u, u')$ (Eq. 5.7). However, we refrain from providing a modified definition here.

For $\text{dist}_{\text{match}}(\triangle_v^{(u)}, \triangle_{v'}^{(u')})$ we obtain

$$\text{dist}'_{\text{match}}(\triangle_v^{(u)}, \triangle_{v'}^{(u')}) = \\ \begin{cases} \sum_{i=1}^{\deg(v)} \text{dist}'_{\text{subtree}}\left(\triangle_{\text{child}_i(v)}^{(v)}, \triangle_{\text{child}_i(v')}^{(v')}\right), & \text{if compatible}_{\text{unary}}(v, v') \\ \infty & \text{, otherwise} \end{cases} \tag{5.20}$$

Here the edge offset needed for the compatibility criterion is given indirectly by the alignment of the edges that connect the v and v' to their respective parents in the tree. All other equations are identical to their original definitions. Therefore, we will not define $\text{dist}'_{\text{AGVG}}$, $\text{dist}'_{\text{subtree}}$, $\text{dist}'_{\text{del}}$, and $\text{dist}'_{\text{add}}$ here. Instead, we now turn away from unary constraints towards binary compatibility criteria.

5.3.2 Binary Constraints Based on Relative Distance

Although we encountered geometric aspects in the constraints described in the previous section, this only served to assess the similarity of nodes and did not concern the geometric configuration of the nodes in an AGVG. Specifying the configuration of objects can be done in different ways. For instance, Bailey (2001) compares the relative distances for each pair of matched features. This approach is generally well suited for the task of describing a local configuration of point objects in the plane. It only fails to distinguish the two mirror images of a configuration. An alternative approach—the one we are going to adopt here—is to describe the geometric configuration of nodes in an AGVG in terms of distances between adjacent nodes and the angles formed at the nodes by the straight edges. While this approach has the downside that it is less strict because error tolerances can add up along the path connecting two nodes, it has one crucial advantage: The first approach cannot not be employed without breaking the optimal substructure condition underlying the dynamic programming approach. The best matching of a subtree here does not necessarily lead to an optimal solution to the overall problem because the best matching for the overall solution now depends on the compatibility with matches made at a higher level of the compared trees.

In the latter approach, on the other hand, the dependency is restricted to the next higher matching (in the case of binary distance constraints) or the next two higher matchings (in the case of ternary angle constraints), as we will see in the following. This allows an extension of our approach that increases the worst-case complexity to a much lesser degree. But before we look into this in more detail, we will define the binary distance criterion that we are going to use.

Our binary criterion restricting pairs of matchings $u \rightsquigarrow u'$ and $v \rightsquigarrow v'$ describes the distance between neighboring nodes. This happens in terms of both their line-of-sight distance l_{los} derived from the edge lengths (and possible intermediate angles) and the distance l_{course} along the Voronoi curves derived from the course information and stored as attributes of the edges.

The binary distance criterion, which we simply call $\text{compatible}_{\text{binary}}$, is then given by

$$\begin{aligned}\text{compatible}_{\text{binary}}((u,u'),(v,v')) \quad \Longleftrightarrow \quad & |l_{\text{los}}(u,v) - l_{\text{los}}(u',v')| < \gamma_{l_{\text{los}}} \\ & \wedge |l_{\text{course}}(u,v) - l_{\text{course}}(u',v')| < \gamma_{l_{\text{course}}}\end{aligned} \tag{5.21}$$

As we mentioned previously, even for the chosen approach of incorporating configurational information, we have to modify our matching approach because the best matching of two subtrees for the overall solution now depends on the previous node

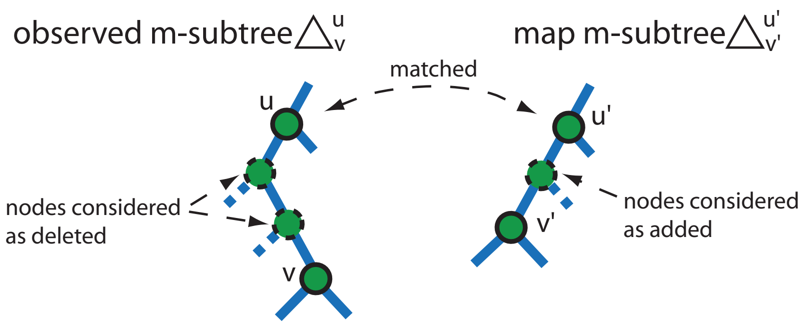

observed m-subtree $\triangle_v^u$
map m-subtree $\triangle_{v'}^{u'}$
matched
u
u'
nodes considered as deleted
nodes considered as added
v
v'

Figure 5.8: Matching two m-subtrees: The m-subtrees consist of root nodes u and u' which are considered as matched, followed by a sequence of edges with intermediate nodes considered as deleted (in the case of the local AGVG) or added (global AGVG), and subtrees with v and v' as roots for which the next edit operation needs to be determined

matching. Consequently, we turn from simple subtrees to matching subtree structures that consist of matched root nodes u and u' followed by sequences of edges whose intermediate nodes are considered as deleted (or added in the case of the model AGVG), and that lead to the currently considered nodes v and v'. We will call this kind of structure *m-subtree* (because it contains one matched node) and denote it by $\triangle_v^u$, leaving out the brackets that we used for simple subtrees. Figure 5.8 illustrates the concept of an m-subtree.

Since each m-subtree in a tree-formed AGVG can be specified by an ordered pair of nodes, the matched node u and the currently considered node v, the number of m-subtrees in an unrooted tree with n nodes is $n(n+1)$. Hence, going from simple subtrees to m-subtrees increases the number of possible matchings and, as a result, the worst-case complexity to $O(m^2n^2)$. However, the additional constraints further reduce the fraction of the subproblems that actually need to be considered. For the equations defining the edit distance, the change to m-subtrees mainly concerns $\text{dist}_{\text{match}}$, while in all other equations we only need to replace simple subtrees $\triangle_v^{(u)}$ with m-subtrees $\triangle_v^u$.

For $\text{dist}_{\text{match}}$ we obtain

$$\text{dist}''_{\text{match}}(\triangle_v^u, \triangle_{v'}^{u'}) =$$
$$\begin{cases} \sum_{i=1}^{\deg(v)-1} \text{dist}''_{\text{subtree}}\left(\triangle_{\text{child}_i(v)}^{v}, \triangle_{\text{child}_i(v')}^{v'}\right), & \text{if } \text{compatible}_{\text{unary}}(v,v') \\ & \wedge\, \text{compatible}_{\text{binary}}((u,u'),(v,v')) \\ \infty & \text{, otherwise} \end{cases} \tag{5.22}$$

We again refrain from providing the rest of the modified equations and directly come to the final version of our algorithm, which allows to express the complete positional information by also including the angles formed by the edges.

5.3.3 Ternary Angle Constraints

In the following, we want to include the angle $\beta(u, v, w)$ formed by connecting a node v with one of its ancestors u and one of its child nodes w into the matching scheme. Hence, we are now dealing with ternary constraints involving three pairs of matched nodes $u \rightsquigarrow u'$, $v \rightsquigarrow v'$, and $w \rightsquigarrow w'$.

We define the criterion for accepting such a matching in a way that it is still true if either $u = v$ or $u' = v'$, and, hence, no angle is defined for one of the AGVGs. However, we explicitly demand that $v \neq w$ and $v' \neq w'$. The reasons for this will become clear in a moment. The ternary compatibility then is

$$\begin{aligned} \text{compatible}_{\text{ternary}}((u,u'),(v,v'),(w,w')) \iff\ & u = v \vee u' = v' \\ & \vee |\beta(u,v,w) - \beta(u',v',w')| \\ & < \gamma_{angle} \end{aligned} \tag{5.23}$$

In order to make the matching of subtrees independent of previous matchings at higher levels, we need to draw both u and v into the considered structure when making a decision for w. Hence, we have to make the step from m-subtrees to *mm-subtrees*. mm-subtrees consist of

1. a node u considered as matched,
2. a first sequence of edges whose intermediate nodes are considered as deleted (or added),
3. another matched node v,
4. a second sequence of edges with intermediate deleted (or added) nodes,
5. a subtree with the currently considered node w as its root.

Admitting that the matched nodes u and v are identical and hence the first sequence of deleted or added nodes is empty allows us to treat m-subtrees as a special case of mm-subtrees. This is important because when starting the matching for the chosen start modes of the AGVGs in $\text{dist}_{\text{node}}$ and $\text{dist}_{\text{aligned}}$, we have to start with an m-subtree as no previously matched node exists. Our definition enables us to get by with a single definition of $\text{dist}_{\text{subtree}}$.

An example of an mm-subtree matching is shown in Fig. 5.9. We denote mm-subtrees as $\triangle_w^{u,v}$. Modifying the edit distance equations by replacing m-subtrees by mm-subtrees and including the $\text{compatible}_{\text{ternary}}$ criterion yields our final matching algorithm. We provide the complete set of edit distance equations below.

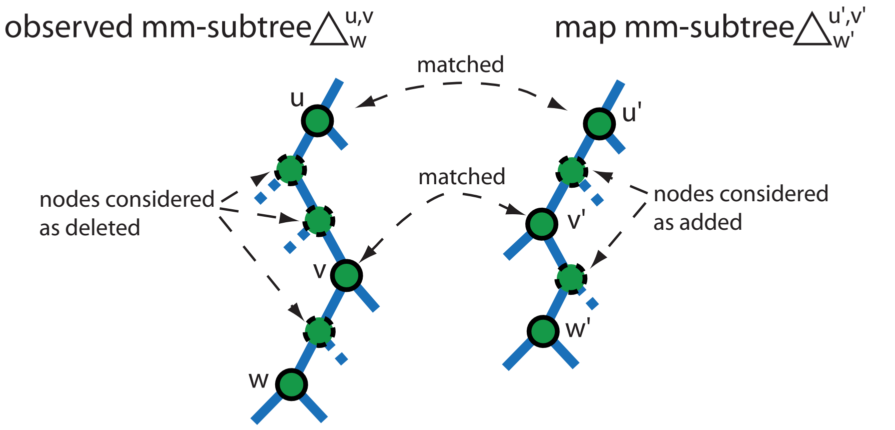

observed mm-subtree $\triangle_w^{u,v}$
map mm-subtree $\triangle_{w'}^{u',v'}$
matched
matched
u
u'
v
v'
w
w'
nodes considered as deleted
nodes considered as added

Figure 5.9: Matching of mm-subtrees: mm-subtrees result from extending m-subtrees with another matched node and a sequence of deleted (or added) nodes. w and w' are the nodes for which the next edit operation needs to be determined

From the definitions of the edit distances for the three edit operations, only $\text{dist}_{\text{match}}$ changes by including $\text{compatible}_{\text{ternary}}$, while the others simply replace m-subtrees with mm-subtrees:

$$\text{dist}'''_{\text{match}}(\triangle_w^{u,v}, \triangle_{w'}^{u',v'}) = \begin{cases} \sum\limits_{i=1}^{deg(w)-1} \text{dist}'''_{\text{subtree}}\left(\triangle_{\text{child}_i(w)}^{v,w}, \triangle_{\text{child}_i(w')}^{v,w}\right), & \text{if compatible}(u,v) \\ & \wedge\, \text{compatible}((v,v'),(w,w')) \\ & \wedge\, \text{compatible}_{\text{ternary}}((u,u'),(v,v'),(w,w')) \\ \infty & \text{, otherwise} \end{cases} \tag{5.24}$$

$$\text{dist}'''_{\text{del}}(\triangle_w^{u,v}, \triangle_{w'}^{u',v'}) = \begin{cases} \min\limits_{1\le i\le \deg(w)-1} \left\{ \left(\sum\limits_{j=1, j\neq i}^{\deg(w)-1} \text{removal}_j(w) \right) + \text{dist}'''_{\text{subtree}}\left(\triangle_{\text{child}_i(w)}^{u,v}, \triangle_{w'}^{u',v'}\right) \right\}, & \text{if } \deg(w) > 1 \\ \infty & \text{, otherwise} \end{cases} \tag{5.25}$$

$$\text{dist}'''_{\text{add}}(\triangle_w^{u,v}, \triangle_{w'}^{u',v'}) = \begin{cases} \min\limits_{i=1}^{\deg(w')-1} \left\{ \left(\sum\limits_{j=1, j\neq i}^{\deg(w')-1} \text{addition}_j(w') \right) + \text{dist}'''_{\text{subtree}}\left(\triangle_{w}^{u,v}, \triangle_{\text{child}_i(w')}^{u',v'}\right) \right\}, & \text{if } \deg(w') > 1 \\ \infty & \text{, otherwise} \end{cases} \tag{5.26}$$

Similarly, $\text{dist}'''_{\text{subtree}}$ directly corresponds to $\text{dist}_{\text{subtree}}$, and $\text{dist}'''_{\text{node}}$ to $\text{dist}_{\text{node}}$:

$$\text{dist}'''_{\text{subtree}}(\triangle_w^{u,v}, \triangle_{w'}^{u',v'}) =$$
$$\begin{cases} 0 & \text{, if unknown}(w) \\ \text{unmatched}(\triangle_w^{u,v}) & \text{, if unknown}(w') \\ \min\left\{\text{dist}'''_{\text{match}}(\triangle_w^{u,v}, \triangle_{w'}^{u',v'}), \text{dist}'''_{\text{del}}(\triangle_w^{u,v}, \triangle_{w'}^{u',v'}), \text{dist}'''_{\text{add}}(\triangle_w^{u,v}, \triangle_{w'}^{u',v'})\right\} & \text{, otherwise} \end{cases} \tag{5.27}$$

$$\text{dist}'''_{\text{node}}(u,u') = \min_{1 \le i \le \deg(u)} \text{dist}'''_{\text{aligned}}(u,u',i) \tag{5.28}$$

$\text{dist}'''_{\text{aligned}}$ is the mentioned special case where the computation starts with actual m-subtrees represented by mm-subtrees with identical u and v. No parent node exists and, hence, no angle will be checked here. Including an angle constraint for the root node would make the optimal solutions for subtrees of u dependent on the matchings in its other subtrees, breaking the optimal substructure condition.

$$\text{dist}'''_{\text{aligned}}(u,u',i) =$$
$$\begin{cases} \sum_{j=1}^{\deg(u)} \text{dist}'_{\text{subtree}}\left(\triangle_{\text{neighbor}_j(u)}^{u,u}, \triangle_{\text{neighbor}_{(j\oplus i)}(u')}^{u',u'}\right) & \text{, if compatible}_{\text{unary}}(u,u') \\ \infty & \text{, otherwise} \end{cases} \tag{5.29}$$

Finally, the overall edit distance remains unchanged:

$$\text{dist}'''_{\text{AGVG}}(G,G') = \min_{v \in V(G), v' \in V(G')} \left\{\text{dist}'''_{\text{node}}(v,v')\right\} \tag{5.30}$$

With regard to complexity, the change from m-subtrees to mm-subtrees means that the size of a table containing all $\text{dist}'''_{\text{subtree}}$ values now depends cubicly on m and n. We already know that the number of mm-subtrees $\triangle_w^{u,v}$ with $u = v$ in a graph with n nodes is $n(n+1)$ because they correspond to m-subtrees. The number of mm-subtrees with $u \neq v \neq w$ depends on the structure of the graph. In the worst case, the graph is a linear chain. In this case, we get $\frac{n(n+1)(2n+1)}{6} - \frac{n(n+1)}{2}$ additional mm-subtrees. As a result, the overall number of mm-subtrees in the worst-case is of order $O(n^3)$ and we get an upper boundary on the time complexity of $O(m^3 n^3)$. However, AGVGs typically are not linear chains but approximately 3-regular graphs. In addition, when we will evaluate the proposed AGVG matching algorithm experimentally in Chap. 7, it will turn out that the actual computational costs are much lower, while the approach achieves a very high level of robustness.

Besides being usable for tracking individual Voronoi nodes or for localization, the data association computed by our matching algorithm can be employed to update the model AGVG based on the observed information. In the last section of this chapter, we will briefly sketch how this can be done. The described procedure is analogous to the way the actual data association can be retrieved when the matching algorithm terminates.

5.4 Map Merging Based on a Computed Data Association

When the data association should be used to incorporate newly perceived subgraphs into the map, this can be done straightforwardly by following the operations that lead to the minimal edit distance. When computing the edit distance, we therefore not only store the computed $\text{dist}'''_{\text{subtree}}$ values but also the operation chosen for this particular subproblem. When this operation is a delete or add operation, we also record the i for which the minimal distance has been achieved (see Eqs. 5.25 and 5.26). In addition to this, we need to keep the pair of start nodes and the edge offset used in the best matching to recover the complete sequence of operations (see Eqs. 5.28 and 5.30).

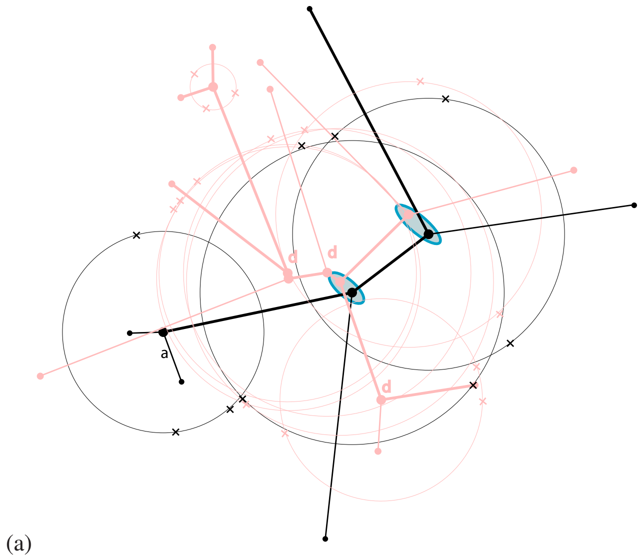
Incorporating new subgraphs into the model AGVG has to take place for every delete operation as this means that subtrees contained in the local AGVG are not contained in the map. The entire map transformation algorithm works analogously to the edit distance computation. We provide a pseudocode version in Algorithm 6. The procedure names have been chosen in accordance with the names of the edit distance functions. The main procedure transformAGVG basically combines $\text{dist}'''_{\text{AGVG}}$, $\text{dist}'''_{\text{node}}$, and $\text{dist}'''_{\text{aligned}}$. In addition to new parts being added to the model AGVG, information about already known features which have been matched can be updated (e.g., updating the position estimates of the nodes). This happens in line 3 of transformAGVG and in line 1 of the transformMatch function.

In Fig. 5.10, we again see the matching example from Fig. 5.3. The optimal matching computed by our algorithm is given by the two matched node pairs connected by the small ellipses, and the edit operations are marked by the letters ‘a’ for add and ‘d’ for delete. On the right, we see how the model AGVG has been complemented with the additional branches from the local AGVG. The basic update approach sketched here can be further extended by incorporating improved techniques for deciding which features should be kept in the map. For instance, an add operation contained in the optimal matching is an indication that a node contained in the map may have been a spurious one that had better be removed.

Possible applications of the data association algorithm for AGVGs developed in this chapter are the tracking of Voronoi nodes, localization within a given map AGVG, and the incremental adding of new information to the map. As the quantitative comparison with the basic ICNN data association algorithm in Sect. 7.2 will show, our approach achieves a much smaller error rate. The experimental analysis will also show that the matching is much more efficient in practice than could be expected from the theoretically determined worst-case time complexity of $O(m^3n^3)$. The computational costs are only slightly increased compared to the ICNN algorithm. The approach is still sufficiently fast to be applicable in multi-hypothesis approaches in which the data association has to be computed for each hypothesis. However, on its own it does not offer a sufficient solution to the problem of constructing a global route graph representation as it will not close loops in the environment. Therefore, we now turn to the question of how global mapping for AGVGs can be achieved.

Algorithm 6 Map transformation algorithm for local AGVG G and map AGVG G'

```
procedure transformAGVG(AGVG $G$, AGVG $G'$)
 1: $v, v' \leftarrow \text{argmin}_{v \in G, v' \in G'} \text{dist}'''_{\text{node}}(v, v')$
 2: $o \leftarrow$ edge offset which resulted in minimal costs
 3: update $v'$
 4: for $i = 1$ to $\deg(v)$ do
 5:     transformSubtree($\triangle^{v,v}_{\text{child}_i(v)}, \triangle^{v',v'}_{\text{child}_{i \oplus o}(v')}$)
 6: end for
procedure transformSubtree($\triangle^{u,v}_{w}, \triangle^{u',v'}_{w'}$)
 1: if operation($\triangle^{u,v}_{w}, \triangle^{u',v'}_{w'}$) $= match$ then
 2:     transformMatch($\triangle^{u,v}_{w}, \triangle^{u',v'}_{w'}$)
 3: else if operation($\triangle^{u,v}_{w}, \triangle^{u',v'}_{w'}$) $= delete$ then
 4:     transformDelete($\triangle^{u,v}_{w}, \triangle^{u',v'}_{w'}$)
 5: else if operation($\triangle^{u,v}_{w}, \triangle^{u',v'}_{w'}$) $= add$ then
 6:     transformAdd($\triangle^{u,v}_{w}, \triangle^{u',v'}_{w'}$)
 7: end if
procedure transformMatch($\triangle^{u,v}_{w}, \triangle^{u',v'}_{w'}$)
 1: update $w'$
 2: for $i = 1$ to $\deg(w)$ do
 3:     transformSubtree($\triangle^{v,w}_{\text{child}_i(w)}, \triangle^{v',w'}_{\text{child}_i(w')}$)
 4: end for
procedure transformDelete($\triangle^{u,v}_{w}, \triangle^{u',v'}_{w'}$)
 1: $p \leftarrow$ parameter($\triangle^{u,v}_{w}, \triangle^{u',v'}_{w'}$)
 2: add $w'$ with child$_i$ subtrees, $i \neq p$, between $v'$ and $w'$ into $\triangle^{u',v'}_{w'}$)
 3: transformSubtree($\triangle^{u,v}_{\text{child}_p(w)}, \triangle^{u',v'}_{w'}$)
procedure transformAdd($\triangle^{u,v}_{w}, \triangle^{u',v'}_{w'}$)
 1: $p \leftarrow$ parameter($\triangle^{u,v}_{w}, \triangle^{u',v'}_{w'}$)
 2: transformSubtree($\triangle^{u,v}_{w}, \triangle^{u',v'}_{\text{child}_p(w')}$)
```



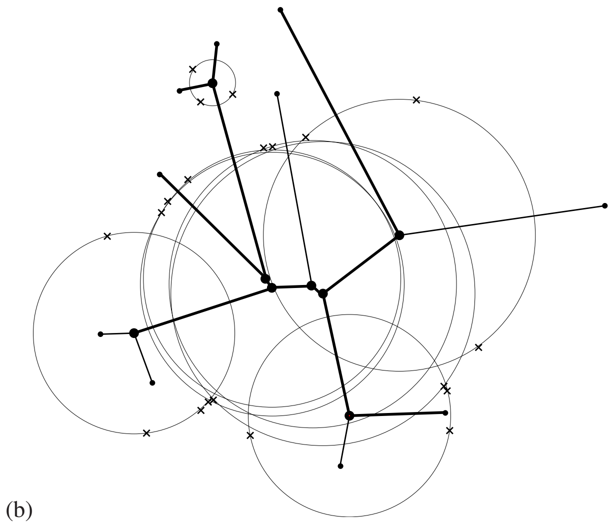


Figure 5.10: Map transformation based on the optimal matching: (a) The small ellipses show the two matched node pairs, and the letters ‘d’ and ‘a’ mark where the corresponding edit operation has been performed. (b) The complemented model AGVG

Chapter 6

Global Mapping: Minimal Route Graphs Under Spatial Constraints

The matching of AGVGs as described in the previous chapter works well as long as the annotated metric information is reliable. This makes the approach well suited for identifying correspondences in AGVGs perceived in short order. However, while the robot moves around, the uncertainty, for instance in the position estimates of the nodes, accumulates and can grow without bounds. As a result of this and due to the fact that our matching algorithm currently does not bridge between different parts of the map AGVG, correctly closing cycles in the graph which correspond to large loops in the environment becomes difficult.

Therefore, a global mapping framework has to be built on top of the AGVG matching, which is the topic of this chapter. Our approach is to deal with the global mapping and loop closing problem by focusing on determining the correct discrete graph topology, relying on coarse but dependable spatial information instead of relying on the uncertainty-afflicted concrete metric annotations. The idea is to first determine the correct high-level graph structure using a multi-hypothesis tracking approach to deal with the uncertainty at the topological level. A concrete (H)AGVG can then be derived from a specific hypothesis.

As a consequence of this idea, we here regard the global mapping problem as the problem of determining the correct topology of a graph-like environment from a sequence of observations and interpret it as the task of finding a minimal route graph model that is consistent with the observations. The minimal model finding formulation of the mapping problem directly leads to a multi-hypothesis approach in which multiple consistent route graph hypotheses are tracked simultaneously.

The problem of exploration and map learning in graph-like environments has been investigated by several authors (Bender et al., 1998; Dudek et al., 1991, 1996, 1997; Rekleitis et al., 1999). Kuipers et al. (Kuipers, 1985; Kuipers & Byun, 1991) describe a *rehearsal procedure* to verify if two observed nodes can correspond only based on node signature information. This procedure involves moving to known neighbored nodes and actively matching the neighborhoods up to a given distance. Dudek et al. (1991) point out that without further information successful map learning cannot be guaranteed. They also show that for undirected graphs a single movable marker is sufficient and that exploration requires $O(mn)$ edge traversals for a graph with m nodes and n edges. In Dudek et al. (1996), a passive map-learning approach is described in which a tree of all possible graph models is maintained.

Kuipers proposed formulating topological mapping as an abductive learning problem of finding the minimal model that explains a sequence of observations and actions (Kuipers et al., 2004; Remolina & Kuipers, 2004). In Kuipers et al. (2004), a tree of all models consistent with the axioms of the SSH is maintained in a way similar to the approach of Dudek et al. (1996). The simplest model is given by a prioritized circumscription policy. Places are derived from local metric maps and compared using descriptions of the local topology. The number of models in the tree can grow exponentially with the number of performed actions.

Planarity of the mapped environment has been exploited in the context of marker-based exploration in Rekleitis et al. (1999) and in the context of abductive map learning in Savelli & Kuipers (2004). In both cases, the result is a significant increase in the computational efficiency of the respective approaches.

In this work, we investigate the minimal model learning approach from the perspective of spatial consistency and as an application of spatial reasoning techniques developed in the area of qualitative spatial reasoning. An overview on the relevant concepts and techniques from this area of research is provided in Appendix B. Our focus is on direction information and we are especially interested in how different kinds of direction information (in the form of different qualitative spatial calculi) affects the size of the space of hypotheses which are consistent with the given information. In addition, we consider planarity as a constraint and require that the route graph models be embedded in the plane without crossing edges. We study the problem in a branch and bound search framework based on the estimated solution size, which results in a further reduction of the explored search space.

We proceed by first investigating the problem of finding the minimal route graph model in the simplified theoretical setting of a general graph world. Later, we adapt the developed approach to the Voronoi-based representations described in this book.

6.1 Theoretical Problem

Consider the following problem: A robot is roaming through a graph-like environment like the one shown in Fig. 6.1. The environment consists of junctions and straight hallways connecting the junctions. Whenever the robot arrives at a junction, it observes and memorizes a set of leaving hallways with some additional (qualitative) direction information. The direction information can either be absolute with regard to a given reference direction (e.g., “corridor x branches off to the north”) or relative (e.g., “corridors x and y meet at an obtuse angle”). We will call a description of such a perception of a junction a *junction observation*, and define it as follows.

Definition 6.1 (Junction observation)**.** *A junction observation is given by a triple* $J = (L, succ, R)$ *where*

- L *is a set* $\{l_1, l_2, ..., l_m\}$ *of pairwise different elements from a set* $\mathcal{H}$ *of hallway identifiers, one for each perceived leaving hallway,*
- $succ : L \rightarrow L$ *is a total function specifying the immediate successor in the counterclockwise cyclic order in which the leaving hallways are perceived, and*
- R *is a spatial description induced by the directions of the leaving hallways in* L *with regard to a given set of direction relations.*

Given a junction observation J, we will write $L(J)$, $s(J)$, and $R(J)$ to refer to the respective elements of the triple J. We will not further define the nature of the spatial description R here because it depends on the particular formalism used to describe the directions of the leaving hallways. An example could be a description using cardinal direction relations n, nw, w, etc. which consists of unary relation tuples, e.g., $n(l_1)$, $sw(l_2)$, and so on. The description could also involve binary relations holding between pairs of leaving hallways. Using cardinal directions, the second junction observation J_2 from the example in Fig. 6.1 could be given by

$$J_2 = (\ \{l_1, l_2, l_3, l_4\},\ \mathrm{succ}(l_i) = l_{(i \oplus 1)},\ \{n(l_1), w(l_2), s(l_3), e(l_4)\}\ )$$

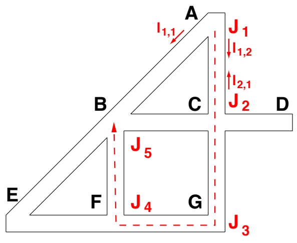
Later in this chapter, we will need to specify the distance of two objects x and y (either leaving hallways in a junction observation or edges incident to a particular node) with regard to the cyclic order defined by a successor function succ. For this purpose, we define the following distance function:

$$d_{\mathrm{succ}}(x, y) = k \Longleftrightarrow y = \mathrm{succ}^k(x) \wedge \nexists l < k : y = \mathrm{succ}^l(x) \tag{6.1}$$

$\mathrm{succ}^k(x)$ here stands for the k-times composition of successor function succ with itself and, hence, the kth successor of x in the cyclic order. For instance, for J_2 we get $d_{\mathrm{succ}}(l_2, l_4) = 2$ and $d_{\mathrm{succ}}(l_3, l_2) = 3$.

Junction observations are connected by *hallway traversal* actions consisting of leaving the current junction via one of the observed leaving hallways and arriving at the next junction via one of the leaving hallways belonging to the next junction observation.

Definition 6.2 (Hallway traversal)**.** *A hallway traversal is described by a quadruple* $T = (J_s, l_s, J_e, l_e)$ *where* J_s *and* J_e *are junction observations with* $J_e \neq J_s$ *and* l_s *and* l_e *are leaving hallway identifiers with* $l_s \in L(J_s)$ *and* $l_e \in L(J_e)$*. It describes the action of taking leaving hallway* l_s *after observing* J_s *and arriving via* l_e *of the next junction observation* J_e*.*



$$\begin{array}{rll} \langle\ J_1 &= (\{l_{1,1}, l_{1,2}\}, \mathrm{succ}_1(x), R_1), & \text{// 1st junction observation} \\ T_1 &= (J_1, l_{1,2}, J_2, l_{2,1}), & \text{// 1st hallway traversal} \\ J_2 &= (\{l_{2,1}, l_{2,2}, l_{2,3}, l_{2,4}\}, \mathrm{succ}_2(x), R_2), & \text{// 2nd junction observation} \\ T_2 &= (J_2, l_{2,3}, J_3, l_{3,1}), & \text{// 2nd hallway traversal} \\ J_3 &= (\{l_{3,1}, l_{3,2}\}, \mathrm{succ}_3(x), R_3), & \text{// ...} \\ T_3 &= (J_3, l_{3,2}, J_4, l_{4,1}), & \\ J_4 &= (\{l_{4,1}, l_{4,2}, l_{4,3}\}, \mathrm{succ}_4(x), R_4), & \\ T_5 &= (J_4, l_{4,2}, J_5, l_{5,1}), & \\ J_5 &= (\{l_{5,1}, l_{5,2}, l_{5,3}, l_{5,4}\}, \mathrm{succ}_5(x), R_5)\ \rangle & \end{array}$$

Figure 6.1: The robot walks through a simple environment passing five junctions (dashed arrow). On the right is the history of junction observations and hallway traversal actions made by the robot

We will call J_s the *start observation* and J_e the *end observation* of the hallway traversal T and use functions startj(T), starth(T), and endj(T), endh(T) for referring to the respective elements of hallway traversal T.

A list of alternating junction observations and hallway traversals corresponding to a walk of the robot through the graph will be referred to as a *history*.

Definition 6.3 (History)**.** *A history $\mathcal{H}$ is a list $\langle J_1, T_1, J_2, T_2, ..., T_{n-1}, J_n\rangle$ of pairwise different junction observations J_i and pairwise different hallway traversals T_j with*

- $\forall i, j, 1 \leq i \leq n, 1 \leq j \leq n, i \neq j : L(J_i) \cap L(J_j) = \emptyset$
- $\forall i, 1 \leq i \leq n-1 : \mathit{startj}(T_i) = J_i \wedge \mathit{endj}(T_i) = J_{i+1}$

While junction observations directly correspond to physical junctions in the environment, a leaving hallway identifier $l \in L(J)$ does not correspond to a physical hallway in our formalization but rather to a "directed hallway" leading away from the junction corresponding to the observation J. Hence, we demand that the leaving hallway identifiers used in the individual junction observations be disjoint in the definition above. Figure 6.1 shows on the right the complete history corresponding to the walk given by the dashed arrow. Each succ$_i$ function here is given by succ$_i(l_{i,j}) = l_{i,(j\oplus 1)}$. In the following, we will use the notation $\mathcal{JO}_{\mathcal{H}}$ for the set of junction observations contained in histories $\mathcal{H}$ and $\mathcal{HT}_{\mathcal{H}}$ for the contained hallway traversals. In addition, $\mathcal{LH}_{\mathcal{H}}$ stands for the union of all $L(J)$ for all $J \in \mathcal{JO}_{\mathcal{H}}$ and, thus, contains all leaving hallway identifiers used.

Histories can be displayed as acyclic graphs, which we will call *history graphs* in the following and which will serve as the starting point for our actual minimal model finding algorithm. A history graph can be seen as the *maximal* route graph model of the environment that explains the history (maximal without adding completely unrelated nodes). This means that no junction observations or leaving hallways are unified except when they definitely have to correspond to the same physical junction because the robot just backtracked along previously traversed hallways. History graphs consist of two kinds of nodes: nodes that stand for junctions which have been observed and

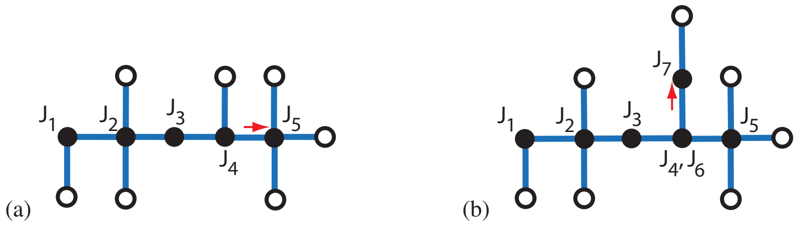


Figure 6.2: Representation of a history as an acyclic graph: Black nodes depict visited places, while white nodes stand for unvisited ones. (a) The graph corresponding to the history given in Fig. 6.1. (b) The resulting graph if the robot would continue by moving back from junction B to junction F and then turning right into the hallway leading to E

nodes that stand for the not yet observed end points of leaving hallways that have not been traversed.

Figure 6.2 shows on the left the history graph for the history from Fig. 6.1. Black nodes stand for the observed junctions and white nodes for the unobserved ones. As no backtracking took place, each black node stands for a single junction observation and each edge connecting two black nodes corresponds to a hallway traversal.

The right picture in Fig. 6.2 shows the graph that would result from continuing the walk by moving back to junction F (resulting in junction observation J_5) and then moving to E (junction observation J_7). As moving back to F means that the robot moves back along the last traversed hallway, J_4 and J_6 *have* to correspond to the same junction and, hence, have to be unified in every valid route graph model derived from the history. As a result, the history graph remains unchanged by the traversal action. In contrast, continuing the walk by moving to E results in assigning J_7 to the corresponding white node, changing it into a black node, and adding new edges and white nodes for the new leaving hallways (one in this case).

Following this construction scheme, complementing the history graph based on a hallway traversal and junction observation pair takes constant time and, hence, the complete construction takes $O(n)$ time for a history of length n (n hallway traversals and $n+1$ junction observations). The history graph representation is not strictly equivalent to the definition of a history though, as the exact number and order of traversal actions and observations cannot be retrieved without further information being stored. However, when annotated with the direction information included in the junction observations and with the final position of the robot and the last traversed edge, it contains all the information needed to transform it into smaller route graph models that are still valid explanations of the history. When depicting history graphs or smaller route graph models derived from the history graph, we will typically not provide the direction relations explicitly, but in most cases place the nodes in compliance with the direction information.

Given a particular history, we are now interested in the problem of finding the minimal route graph model that explains the sequence of observations and traversal actions. As explanations we consider route graph models together with mappings from the history to walks through the hypothetical graphs. Route graph model here refers to an undirected graph together with a combinatorial embedding similar to the EGVGs defined in Chap. 3 but without loops or parallel edges, as we restrict ourselves to environments consisting of straight hallways. In order to avoid confusion with the nodes in the search tree discussed later in this chapter, we will refer to the nodes and edges of a route graph hypothesis explicitly as *route graph nodes* (RGNs) and *route graph edges* (RGEs), respectively.

A hypothesis (route graph model and mapping) is a valid explanation of the history if the following holds: We can draw the RGNs of the combinatorial embedded route graph into the plane with all RGEs being non-crossing straight line segments so that the hypothetical walk through this route graph model would then reproduce the history. This condition can be split into three classes of constraints that have to be satisfied in order to offer a valid explanation for a given history:

- **Structural constraints:** Under the term structural constraints we subsume all constraints that are not linked to the actual assignment of coordinates to the RGNs by the drawing: (1) The hypothetical walk through the given route graph hypothesis needs to reproduce the sequence of leaving hallway numbers of the original walk. (2) The cyclic order of leaving hallways needs to match the cyclic order of leaving edges of the corresponding nodes. (3) The distance between the arriving and leaving hallway in the cyclic order of leaving hallways needs to match for each passing of a node.

- **Planarity constraint:** The combinatorial embedding given by the cyclic edge orderings needs to be planar and the drawing must be a straight-line drawing of this embedding.

- **Direction constraints:** For each passed RGN, the drawing of the graph into the plane needs to induce the same direction relations for the leaving hallways as described in the junction observations.

To provide a more formal definition of these constraints, we first introduce the concept of a *map hypothesis*, which captures the structural constraints as these can be directly applied when constructing a hypothetical route graph model from a given history. We then define a map hypothesis to be *consistent* when the planarity constraint and direction constraints are satisfied. As already indicated previously, a map hypothesis not only consists of a route graph model but also contains a description of how the perceived junction hallways of the history map to the RGNs and RGEs of the route graph model. This mapping defines the hypothetical walk through the route graph that corresponds to the history.

Definition 6.4 (Map hypothesis)**.** *Given a history $\mathcal{H} = \langle J_1, T_1, J_2, T_2, ..., T_{n-1}, J_n \rangle$, a map hypothesis $M_{\mathcal{H}}$ for a $\mathcal{H}$ is a triple (G, mj, ml), where*

- *$G = (V, E, O)$ is an undirected graph (V, E) with node set $V = \{v_1, v_2, ..., v_n\}$, edge set $E \subseteq V \times V$, and combinatorial embedding given by a set $O = \{succ_{v_1}, succ_{v_2}, ..., succ_{v_n}\}$ of successor functions of which $succ_{v_i}$ specifies the cyclic order of edges incident to v_i,*
- *$mj : \mathcal{JO}_{\mathcal{H}} \to V$ is a total function mapping the junction observations from $\mathcal{H}$ to nodes of G, and*
- *$ml : \mathcal{LH}_{\mathcal{H}} \to E$ is a total function mapping the leaving hallway identifiers from $\mathcal{H}$ to edges of G,*

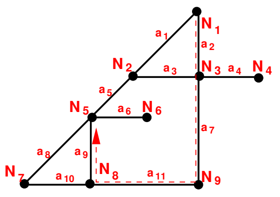
that satisfies the following conditions:

- $\forall J \in \mathcal{JO}_{\mathcal{H}} : (\forall l \in L(J) : mj(J) \in ml(l))$ *(incidence preservation)*
- $\forall T \in \mathcal{HT}_{\mathcal{H}} : ml(startl(T)) = ml(endl(T)) = \{startj(T), endj(T)\} \in E$ *(traversal-edge mapping)*
- $\forall J \in \mathcal{JO}_{\mathcal{H}} : |L(J)| = \deg(mj(J))$ *(degree match)*
- $\forall J \in \mathcal{JO}_{\mathcal{H}}, s = s(J) : \big(\forall l \in L(J) : succ_{mj(J)}(ml(l)) = ml(s(l))\big)$ *(cyclic order preservation)*
- $\forall i, 1 < i < n, s = s(J_i) :$ $d_s(endl(T_{i-1}), startl(T_i)) = d_{succ_{mj(J_i)}}(ml(endl(T_{i-1})), ml(startl(T_i)))$ *(cyclic order distance preservation)*

The first two conditions of the definition above simply ensure that junction observations and their leaving hallways are mapped to incident RGNs and RGEs and that the leaving hallways of a hallway traversal are mapped to the same RGE. The other three conditions are the structural constraints mentioned earlier: matching numbers of leaving hallways, preservation of the cyclic order information, and preserved distance between the arriving hallway and the leaving one in the cyclic order. We introduce the function $G(M)$ for accessing the route graph model G in a map hypothesis $M = (G, \text{mj}, \text{ml})$.

Figure 6.3 shows four map hypotheses for the history given in Fig. 6.1. Each hypothesis is depicted by one drawing of the route graph into the plane. All examples preserve the combinatorial embedding of the hypothesis but not all consist only of straight lines and avoid crossing edges. The walk resulting from the mapping of the history to the route graph is shown by the dashed arrows. Disregarding the direction information, the walks resulting from the mappings would in all cases correctly reproduce the history of observations and hallway traversals.

(a)



$mj(J_1) = N_1, ml(l_{1,1}) = a_1, ml(l_{1,2}) = a_2,$

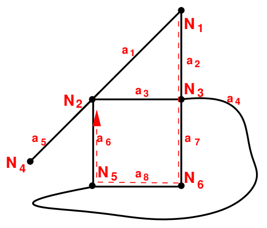
$mj(J_2) = N_3, ml(l_{2,1}) = a_2, ml(l_{2,2}) = a_3, ml(l_{2,3}) = a_7, ml(l_{2,4}) = a_4,$

$mj(J_3) = N_9, ml(l_{3,1}) = a_7, ml(l_{3,2}) = a_{11},$

$mj(J_4) = N_8, ml(l_{4,1}) = a_{11}, ml(l_{4,2}) = a_9, ml(l_{4,3}) = a_{10},$

$mj(J_5) = N_5, ml(l_{5,1}) = a_9, ml(l_{5_2}) = a_6, ml(l_{5,3}) = a_5, ml(l_{5,4}) = a_8$

(b)



$mj(J_1) = N_1, ml(l_{1,1}) = a_1, ml(l_{1,2}) = a_2,$

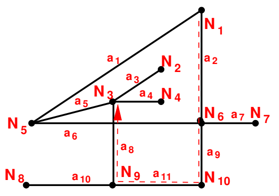
$mj(J_2) = N_3, ml(l_{2,1}) = a_2, ml(l_{2,2}) = a_3, ml(l_{2,3}) = a_7, ml(l_{2,4}) = a_4,$

$mj(J_3) = N_6, ml(l_{3,1}) = a_7, ml(l_{3,2}) = a_8,$

$mj(J_4) = N_5, ml(l_{4,1}) = a_8, ml(l_{4,2}) = a_7, ml(l_{4,3}) = a_4,$

$mj(J_5) = N_2, ml(l_{5,1}) = a_6, ml(l_{5,3}) = a_3, ml(l_{5,4}) = a_1, ml(l_{5,5}) = a_5$

(c)



$mj(J_1) = N_1, ml(l_{1,1}) = a_1, ml(l_{1,2}) = a_2,$

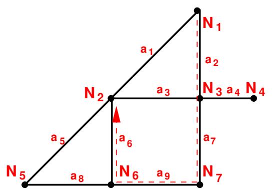
$mj(J_2) = N_6, ml(l_{2,1}) = a_2, ml(l_{2,2}) = a_6, ml(l_{2,3}) = a_9, ml(l_{2,4}) = a_7,$

$mj(J_3) = N_{10}, ml(l_{3,1}) = a_9, ml(l_{3,2}) = a_{11},$

$mj(J_4) = N_9, ml(l_{4,1}) = a_{11}, ml(l_{4,2}) = a_8, ml(l_{4,3}) = a_{10},$

$mj(J_5) = N_3, ml(l_{5,1}) = a_8, ml(l_{5,2}) = a_4, ml(l_{5,3}) = a_3, ml(l_{5,4}) = a_5$

(d)



$mj(J_1) = N_1, ml(l_{1,1}) = a_1, ml(l_{1,2}) = a_2,$

$mj(J_2) = N_3, ml(l_{2,1}) = a_2, ml(l_{2,2}) = a_3, ml(l_{2,3}) = a_7, ml(l_{2,4}) = a_4,$

$mj(J_3) = N_7, ml(l_{3,1}) = a_7, ml(l_{3,2}) = a_9,$

$mj(J_4) = N_6, ml(l_{4,1}) = a_9, ml(l_{4,2}) = a_6, ml(l_{4,3}) = a_8,$

$mj(J_5) = N_2, ml(l_{5,1}) = a_6, ml(l_{5,2}) = a_3, ml(l_{5,3}) = a_1, ml(l_{5,4}) = a_5$

Figure 6.3: Four possible map hypotheses for the history of Fig. 6.1, given by the depictions of their route graph models and the mappings mj and ml. Only the examples in (a) and (d) are consistent. The model in (d) is also a minimal route graph model; it corresponds to the original environment

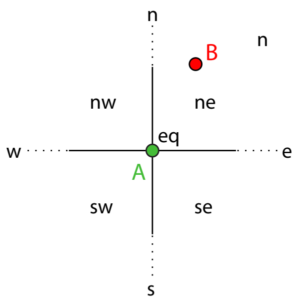


Figure 6.4: Relations from the cardinal direction calculus

Let us for now again assume that the spatial information is given in form of qualitative cardinal direction relations. The exact relations we are going to use are those of the cardinal direction calculus (Ligozat, 1998), a calculus for qualitative spatial reasoning which we are going to introduce more formally in Sect. 6.3.2.1 (see Appendix B for more details on qualitative spatial calculi). In the cardinal direction calculus, n, w, s, and e correspond to specific angles, while nw, sw, se, and ne comprise angle intervals as indicated in Fig. 6.4.

Storing the cardinal directions derived from compass readings for each leaving hallway of a junction observation results in the following spatial descriptions R_i for our exemplary history from Fig. 6.1:

$$\begin{aligned}
R_1 &= \{sw(l_{1,1}), s(l_{1,2})\}\\
R_2 &= \{n(l_{2,1}), w(l_{2,2}), s(l_{2,3}), e(l_{2,4})\}\\
R_3 &= \{n(l_{3,1}), w(l_{3,2})\}\\
R_4 &= \{e(l_{4,1}), n(l_{4,2}), w(l_{4,3})\}\\
R_5 &= \{s(l_{5,1}), e(l_{5,2}), ne(l_{5,3}), sw(l_{5,4})\}
\end{aligned}$$

The drawing of the hypothesis of Fig. 6.3(a) shows that this hypothesis is consistent with the spatial information as it would correctly reproduce the cardinal directions for each junction observation. The drawing of the second hypothesis (Fig. 6.3(b)), however, is not consistent with the spatial information contained in the history: A spatial inconsistency arises because the hallway leaving from N_5 to the west has been connected with the one leaving N_3 to the east. As we assume straight hallways, it follows that N_5 has to be to the east of N_3. Locally, this information is consistent. However, as it is also known that N_6 is south of N_3 and N_5 is west of N_6, it follows that N_5 has to be in the southwest sector of N_3. This contradicts our previous conclusion that N_5 lies to the east of N_3. This reasoning shows not only that the depicted drawing is inconsistent with respect to the direction information, but also that no such drawing can exist for the given hypothesis. Hence, the entire hypothesis is inconsistent and does not offer a valid explanation of the history.

A second cause of inconsistency within a hypothesis is shown in the third example (Fig. 6.3(c)). Here the drawing is spatially consistent with regard to the direction information and it preserves the cyclic edge orders. However, it has crossing edges (N_5-N_6 and N_3-N_9) and, more importantly, no drawing without crossing edges exists because for this graph the combinatorial embedding is not planar.

These two inconsistent examples illustrate that the structurally valid map hypotheses can still be inconsistent with the available information or the underlying assumption that the environment is planar. This leads to the following definition of a consistent map hypothesis.

Definition 6.5 (Consistent map hypothesis)**.** *A map hypothesis $M_{\mathcal{H}} = (G, mj, ml)$ is consistent if there exists a straight-line drawing $D(G)$ of G into the plane that satisfies the following conditions:*

- *for every node in G, the natural cyclic order of leaving edges in the drawing $D(G)$ corresponds to the cyclic edge order specified in the combinatorial embedding (cyclic order preservation),*
- *the line segments in $D(G)$ corresponding to the RGEs of G do not cross each other (planar drawing), and*
- *for every junction observation $J \in \mathcal{JO}_{\mathcal{H}}$, the spatial description induced by $D(G)$ for node $mj(J)$ matches the spatial description one gets by replacing all $l \in L(J)$ with $ml(l)$ in description $R(J)$ (matching direction information).*

The general idea of the minimal route graph model approach to the global mapping problem is to prefer among all consistent map hypotheses the one that is minimal in terms of the number of RGNs in the route graph. We will call such a hypothesis a minimal route graph model.

Definition 6.6 (Minimal route graph model)**.** *Given the set $\mathcal{CM}_{\mathcal{H}}$ of consistent map hypotheses for $\mathcal{H}$, $M \in \mathcal{CM}_{\mathcal{H}}$ is a minimal route graph model for $\mathcal{H}$ if and only if no $N \in \mathcal{CM}_{\mathcal{H}}$ exists with $|V(G(N))| < |V(G(M))|$.*

The last example in Fig. 6.3(d) shows a second consistent map hypothesis, the one that corresponds to the original environment. The number of RGNs is smaller than in the first example and no consistent hypothesis with even less RGNs exists for the given history. Hence, it is a minimal route graph model for this particular history.

Following the minimal route graph model approach, the problem of mapping an unknown environment becomes a combinatorial optimization problem of incrementally computing one or all minimal consistent map hypotheses from the observations gathered during exploration. One thing that makes this problem interesting and complex is the fact that it combines combinatorial aspects with questions of spatial consistency. While the space of structurally possible map hypotheses grows exponentially with the length of the history, the direction constraints and planarity restriction reduce the search space by allowing us to prune inconsistent branches.

The fact that the definition of a consistent map hypothesis is based on the existence of a drawing of the graph that satisfies certain conditions is a further indication of the complexity of the problem: Clearly, it it infeasible to consider all drawings as there are infinitely many straight-line drawings of a graph. General techniques like describing the constraints in a system of (geometric) equations are typically much too expensive computationally to be applicable.

On the positive side, there exist techniques that solve individual aspects of the problem. For instance, checking whether a graph with combinatorial embedding is planar—which means that a drawing without crossing edges exists that preserves the combinatorial embedding—can be done in $O(n)$ time. In addition, for every planar drawing there exists one with straight edges. Consequently, we can filter out many inconsistent hypotheses by discarding all hypotheses with non-planar embeddings.

With regard to additional spatial information, in our case the direction relations, techniques for checking the consistency of a set of qualitative spatial relations have been developed in the research field of qualitative spatial reasoning. These techniques allow for determining whether a given network of spatial constraints is satisfiable. Again, employing these techniques allows filtering out a substantial part of inconsistent hypotheses. The minimal route graph model problem provides an interesting testbed for these kinds of approaches as it benefits from expressive spatial formalisms for which the consistency problem can be solved efficiently. One of our goals in this work is to investigate the suitability of existing qualitative direction calculi for this kind of problem and identify potential need for further research in this area.

In the following, we describe an approach to determine a minimal route graph model based on individually enforcing the planarity constraint and the consistency of the direction information. As a result of the individual constraint checking, the approach is incomplete in the sense that it may not filter out all inconsistent map hypotheses. For instance, it could happen that there exists a drawing for a given map hypothesis that is planar and one that is compliant with the direction constraints but not one that is both. However, not finding all constraints is still preferable to not using the available information at all. We start by first looking at the combinatorial optimization problem and developing a best-first branch-and-bound-based search procedure to solve it. In the next section, we integrate planarity and spatial consistency checks into the framework. The results of the empirical evaluation of this approach for two spatial calculi are provided in Sect. 7.3.

6.2 Branch and Bound Search for Minimal Model Finding

In this section, we describe a solution to the minimal route graph model problem that consists of a branch and bound search through the search tree of possible associations of RGNs and, hence, junction observations. The search starts with the history graph and effectively folds the graph onto itself by unifying the RGNs with their corresponding junction observations. A lower bound estimate of the model size as implied by the associations already made is used to efficiently guide the search towards a minimal model in a best-first manner. As a result, the approach effectively performs an A* search through the interpretation tree.

6.2.1 Search Through the Interpretation Tree

Deriving route graph hypotheses from a history of junction observations and hallway traversals is mainly a problem of correctly identifying junction observations that correspond to the same physical junction in the environment. Hence, similarly to approaches on data association we encountered in the previous chapter, a solution to the problem can be formulated as a search through the tree of possible associations, the interpretation tree. In the case of the data association problem we had two disjoint sets of objects, the data set and the model set. Here, however, we only have one set of objects, the RGNs. In principle, each RGN can be associated with multiple other RGNs, resulting in a partition of the set of junction observations into equivalence classes. We still end up with a tree-formed search space in which each level corresponds to matching one particular RGN and each edge corresponds to one particular matching if we make matching assignments in the following way (see also Fig. 6.6):

- At each level in the interpretation tree, the corresponding RGN can only be matched to objects already matched at a higher level. Hence, the fact that junction observations mapped to RGN A and RGN B should correspond to the same physical junction could be expressed by first assigning A to 0 (new junction) and then at a lower level matching B to A.
- When more than two RGNs should be unified (more than two junction observations correspond to the same physical junction), each RGN has to be matched with the one previously associated in the tree: Assuming A, B, and C should be unified to form a single junction and are associated in this order in the tree, first A is assigned to 0, then B is assigned to A, and finally C is assigned to B.

For the following discussion we extend the graph representation for histories and derived route graph hypotheses so that it allows us to describe the partial matchings corresponding to inner nodes of the interpretation tree. This is done by distinguishing RGNs in the graph depending on whether they have already been matched or not. Already matched RGNs are depicted by circles as before, black for visited ones and white for unvisited ones. Not yet assigned RGNs are depicted by black and white crosses instead (see Fig. 6.5).

An additional deviation from the standard interpretation tree is that an assignment can lead to multiple child nodes as there may exist multiple route graph hypotheses resulting from joining two RGNs. One could argue that this would not be required if one would match hallways instead of junctions, but as our hypotheses also comprises junctions that have not been observed and for which consequently the number of leaving hallways objects is unknown, we think the chosen approach is more adequate.

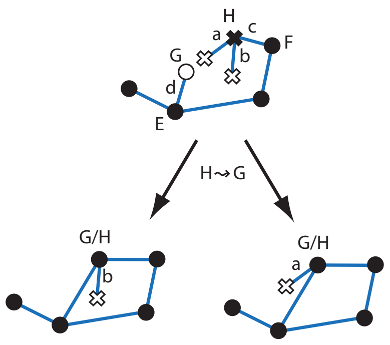


Figure 6.5: Assigning H to G results in two possibilities because the hallway d can correspond to either a or b. Therefore, this matching results in two successor nodes in the interpretation tree

Multiple successor nodes can arise when there exist multiple ways to map the leaving hallways of the matched junctions onto each other while preserving the cyclic order information. Figure 6.5 shows an example of such a situation: The picture shows at the top the partially constructed route graph hypothesis corresponding to a node in the search tree. We now assume that we want to identify RGN H with RGN G. Since G is an unvisited junction and H is a visited one, the RGE d of G must correspond to one of the leaving RGEs of H, either to a or to b. It cannot correspond to c as RGNs E and F have already been assigned and, thus, cannot be unified anymore in this hypothesis. As a result two successor hypotheses are possible, shown below, and there would be two child nodes for this particular matching in the interpretation tree.

A complete search tree for a small example walk consisting of three junction observations is shown in Fig. 6.6. While in principle each RGN could be matched with any of the RGNs associated at a higher level, two junctions can only be connected by a single hallway, and matchings can directly imply other matchings. As a result, not all matchings occur in the interpretation tree depicted in the figure. Still, the number of map hypotheses grows exponentially with the length of the history.

As also illustrated in Fig. 6.6, we store the following information for each node in the search tree: a (partial) route graph hypothesis reflecting the partial matching and and an RGN list which contains all RGNs from the original history graph. For already matched RGNs the list states the assigned other RGN. For the still unassigned RGNs, the RGN list contains a list of matching candidates called its *match list*. Additional stored information not shown in the figure includes the mapping from the history to the junctions and hallways in the route graph model, given in the form of annotations to the RGNs and RGEs and the robot's current location within the route graph.

Several examples of node hypotheses are included in Fig. 6.6. H_1 is the original history graph, with all nodes depicted by crosses because no assignments have been made yet. As RGN A is the first RGN considered, it has to be new, which results in node hypothesis H_2. B, C, and D are the end points of the leaving hallways of A and, hence, have to be new junctions as well (their match lists only contain 0), which leads to hypothesis H_5. At the next level, E can only be matched to B or C (or to 0) and not to A because a connecting RGE between A and D already exists. H_5, H_7, H_{14}, and H_{33} are intermediate node hypotheses along the marked path through the tree leading to H_{83}. As all nodes in H_{83} are assigned, it is a complete map hypothesis. It is also the hypothesis that reflects the actual environment. The remaining examples are alternative complete map hypotheses of which H_{47} assumes less junctions than H_{83}, while H_{112} requires more. In H_{83}, it has been hypothesized that C, E, and G correspond to a single physical junction and, as mentioned, this has to be realized by associating E with C before associating F with E.

Overall, expanding a node in the interpretation tree during the search for the minimal model involves the following three steps: (1) The first still unassigned RGN in the RGN list is chosen; (2) the successor nodes for every matching of this RGN with a candidate in its match list are generated; (3) the successor nodes are added to a list containing the current fringe of the search tree. More details on the generation of successor nodes will be given in Sect. 6.2.3.2.

6.2.2 Best-First Branch and Bound Search Based on Solution Size

As it is our goal to find a minimal consistent map hypothesis in terms of the number of RGNs, upper and lower bounds on the model size over all hypotheses that can be generated from a particular node in the search tree can be used to decide whether an optimal solution can be contained in this part of the search space. Nodes with a lower bound higher than the currently found minimal upper bound can be completely excluded from the search. Hence, by employing a branch and bound search approach, we achieve a reduction of the search space.

In addition, a lower bound estimate can be used to efficiently guide the search towards a minimal model in a best-first manner by always expanding the node with the currently smallest lower bound. Once the chosen node with the minimal lower bound contains a hypothesis in which all nodes have been assigned, we have found a minimal map hypothesis.

Every inner node in the search tree contains a route graph model with still unassigned RGNs. Hence, it stands for a set of map hypotheses which can be generated by performing the remaining assignments. A lower bound on the number of RGNs contained in a map hypothesis in this set can be computed efficiently in the following way based on the node's RGN list: Every RGN that is assigned to 0 counts as one because it will occur as an RGN in any of the derived map hypotheses. The same holds for every still unassigned RGN for which the match list only contains 0 since this RGN cannot be matched with an already established RGN any longer. All other RGNs from the list have either been matched with an already counted RGN or their match lists still allow for such a matching. Hence, they do not count.

The resulting estimate is only a lower bound on the minimal number of RGNs

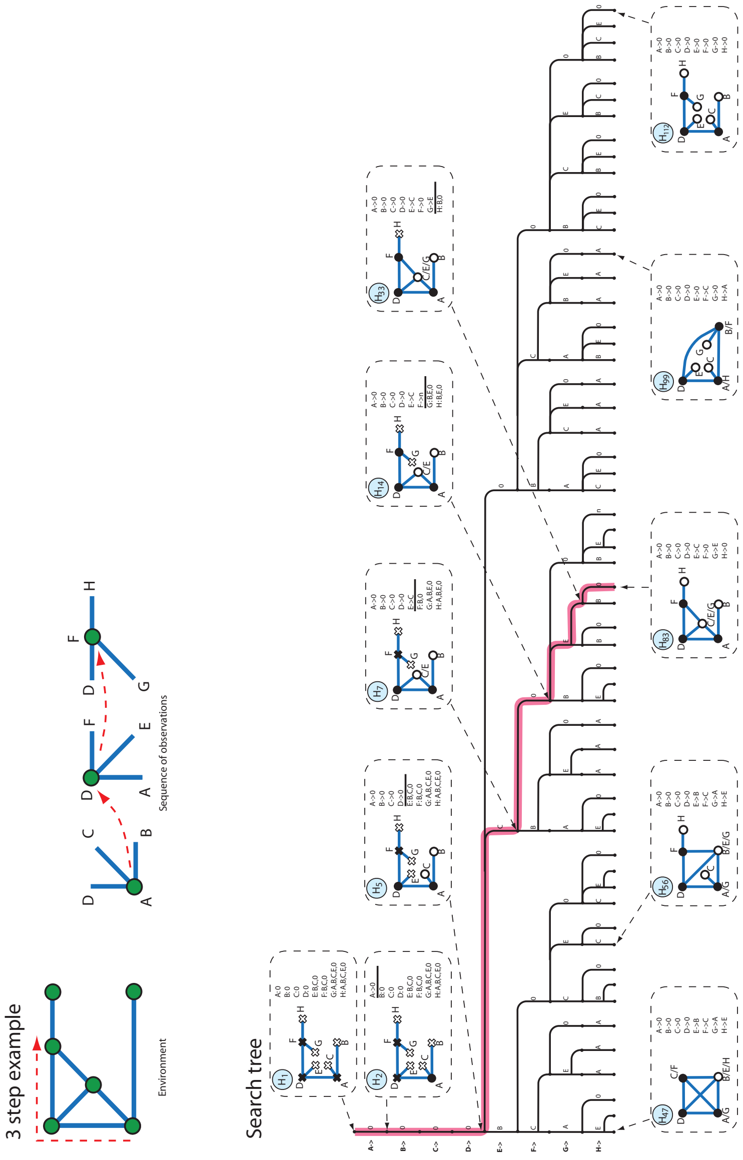


Figure 6.6: The complete interpretation tree resulting from a three-step walk through a simple environment. The environment and the sequence of observations are given at the top of the figure. The node data (graphs and RGN lists) is depicted for several nodes in the search tree

because later assignments may actually lead to cases in which the match list of an RGN is reduced to 0. This would result in an additional junction. However, to get a better lower bound would require us to perform a costly analysis of the dependencies in the match lists, while in our case we simply have to update some counters during the search.

In summary, the lower bound on the model size for a node in the search tree is defined as follows.

Definition 6.7 (Model size lower bound)**.** *Given a node n in the interpretation tree, the model size lower bound mslb(n) of n is defined as*

$$mslb(n) = N + O \tag{6.2}$$

where N is the number of RGNs assigned to 0 in the RGN list of n and O is the number of RGNs with 0 as the only element in their respective match lists.

In the implementation of the search algorithm described here, the current fringe of the search tree is stored in a priority queue sorted by the nodes' lower bounds. In case two nodes have the same lower bounds, a secondary criterion and, if needed, a tertiary one are used to sort the nodes. The secondary criterion is the number of still unassigned RGNs in the RGN lists. As a consequence, nodes deeper in the tree and, hence, closer to a complete map hypothesis will be preferred. The tertiary criterion is the upper bound msub(n) on the possible minimal model size for node n. It is computed by summing up the number of RGNs assigned to 0 and the RGNs that still need to be assigned and have 0 in their match list.

Definition 6.8 (Model size upper bound)**.** *Given a node n in the interpretation tree, the model size upper bound msub(n) of n is defined as*

$$msub(n) = N + P \tag{6.3}$$

where N is the number of RGNs assigned to 0 in the RGN list of n and P is the number of RGNs with 0 contained in their match list.

To provide an example of the results of applying the best-first branch and bound search as described in this section, Fig. 6.7 shows at the top the original interpretation tree from Fig. 6.6 and in the middle the parts searched for a minimal model using the best-first branch and bound algorithm. The order in which the nodes are expanded is given by the numbers within the circles representing the nodes. The bottom figure shows the result of applying the same approach but searching until all minimal models have been found.

6.2.3 Expand and Update Operations

The main operation of the search procedure is the expansion of a node in which the child nodes based on all possible matchings are generated. However, in order to incorporate newly available history information without performing a new search from

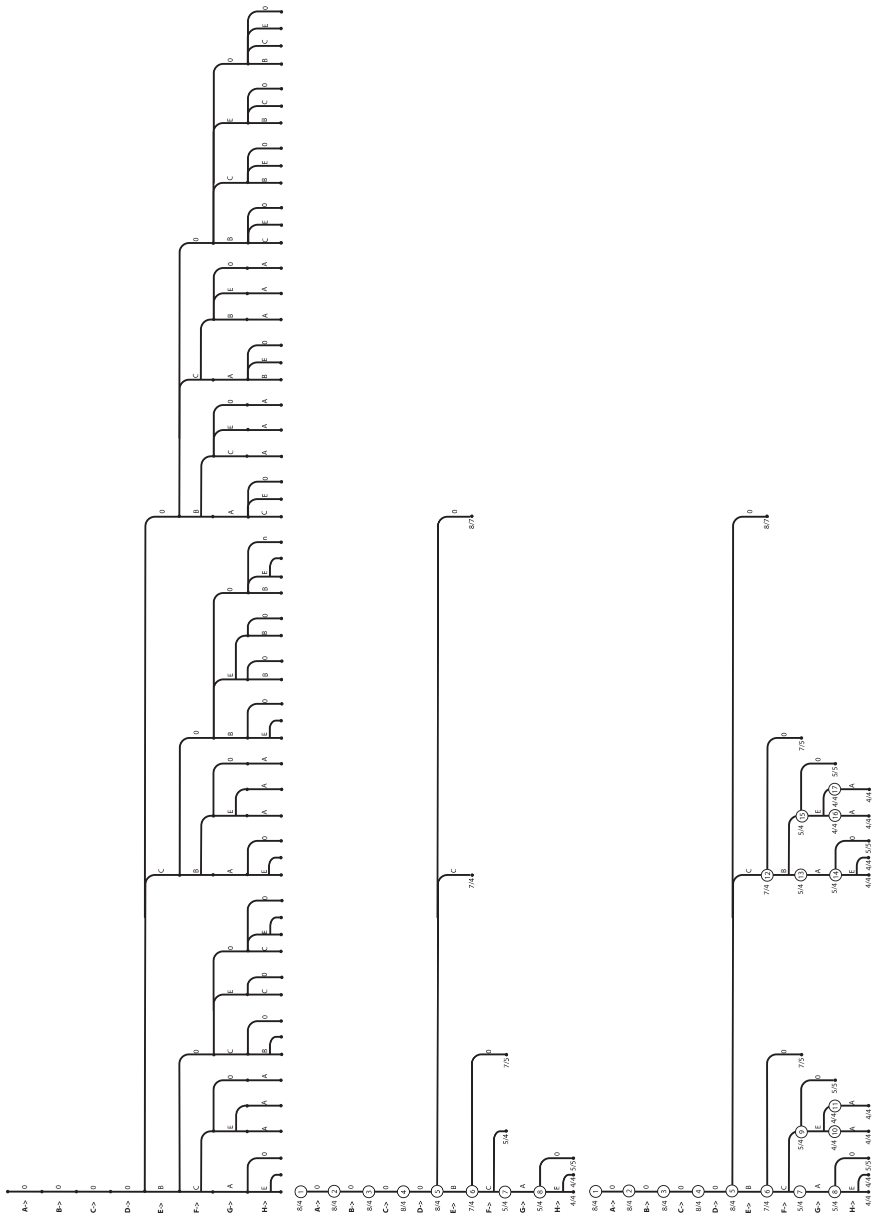


Figure 6.7: Effects of different versions of the minimal model algorithm (part 1): At the top the entire search tree, in the middle the tree searched with branch and bound for the first minimal model, and at the bottom the same searching for all minimal models

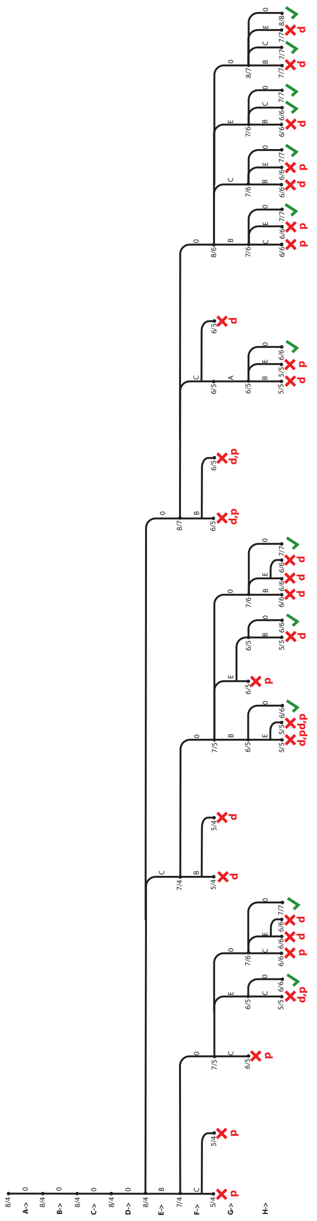
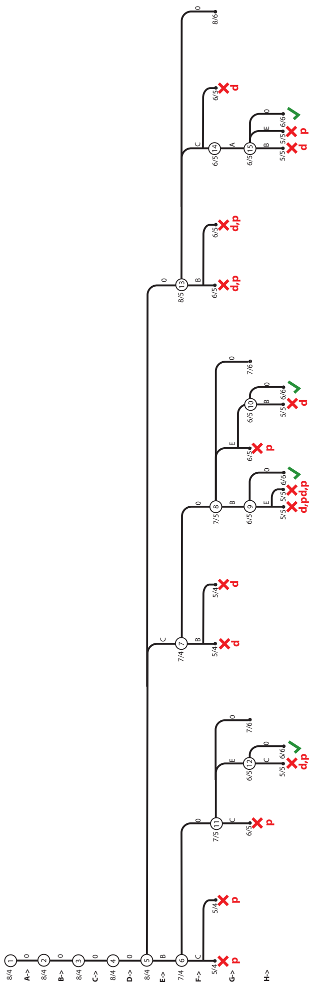



Figure 6.8: Effects of different versions of the minimal model algorithm (part 2): At the top the tree pruned by planarity and direction constraints and at the bottom the result of applying constraint-based pruning and branch and bound search

Algorithm 7 Main loop of the minimal model finding algorithm

```
procedure minimalModel(History $\mathcal{H}$, PriorityQueue Q)
1:  minimalModelFound ← $false$
2:  while $|Q| > 0$ and not minimalModelFound do
3:      $n \leftarrow$ pop$(Q)$
4:      if not uptodate$(n)$ then
5:          $L \leftarrow$ update$(n, \mathcal{H})$
6:          insert all elements from $L$ into $Q$
7:      else if node has unassigned RGNs then
8:          $L \leftarrow$ expand$(n)$
9:          insert all elements from $L$ into $Q$
10:     else
11:         minimalModelFound ← $true$
12:         insert $n$ into $Q$
13:     end if
14: end while
```

scratch, a second operation is required in which the node information is updated based on the new action and observation. We will refer to these two operations as the *expand* and *update* operations.

A pseudocode version of the main loop of the actual search procedure is shown in Algorithm 7. This procedure is called whenever new history information becomes available and terminates when a minimal model has been found. The current fringe of the search tree is provided in the form of the priority queue Q sorted by criteria described in the previous section. In the main loop, the first element is taken from Q. It is then tested whether this node n is up-to-date (meaning all history information has been incorporated) or not. If this is not the case, the update operation will be performed, which incorporates the next hallway traversal and junction observation. Like the expand operation, the update operation may result in multiple successor hypotheses. Therefore, it returns a list of new nodes which are inserted into Q.

If the node n does not need to be updated, it is checked whether it still has unassigned RGNs. If this is the case, n is expanded and successor nodes which are returned as a list are inserted into Q. Otherwise, n has to be a minimal model and the search terminates. Before that, n is put back into the queue so that we can continue with the search when new history information becomes available.

In the following, we consider the update and expand operations in more detail.

6.2.3.1 Update Operation

The update operation updates a node n based on one new hallway traversal T and a new junction observation J. The result is a set of updated successor nodes. The following steps have to be performed:

1. A list L is initialized as empty; in the end this list will contain the successor nodes.
2. The robot's location within the route graph hypothesis of n is updated based on T.
3. Case 1: If the new location loc is an already visited RGN, we check whether J fits the current location. If this is the case, a copy s of n is added to L. Then $\text{mj}(J)$ is set to loc and each $\text{ml}(l_i)$ for a leaving hallway in J is set to the corresponding RGE in s. If J and the visited RGN do not match, no successor nodes for n will be created, effectively closing this branch of the interpretation tree.
4. Case 2: If the new location is an unvisited RGN, there may be multiple ways to map the already existing RGEs to the observed leaving hallways. Every existing RGE needs to be mapped to a different leaving hallway while the cyclic order needs to be preserved. For each valid mapping the following is done: A new node s_i is constructed in which the graph has been updated accordingly. A new edge ending at a new unvisited RGN is attached at the right position in the cyclic order for each leaving hallway that does not correspond to an existing RGE. mj and ml are updated as in case 1 and the nodes s_i are added to L.
5. For each node from L, new RGNs are added to the RGN list for each end point of the leaving hallways in J (even though no actual RGN may have been added to the route graph model), and their match lists are set accordingly.
6. L is returned as the result of the update operation.

To illustrate this procedure, two examples of update operations are depicted in Fig. 6.9. On the left, we see the current node hypothesis before the first update operation. All RGNs except E are already matched. The new hallway traversal to be incorporated now leads the robot from C to B/D as indicated by the dashed arrow. A depiction of the new junction observation can be found between the arrows. It contains two more leaving hallways in addition to the arriving one. The end points of these will require the instantiation of two more RGNs, F and G. B/D is an unvisited RGN in the given hypothesis (case 2), which means that every existing RGE needs to correspond to an observed leaving hallway but not vice versa, and we already know that the RGE connecting C and B/D corresponds to the leaving hallway via which the robot arrived. Therefore, two mappings are possible, leading to the two new hypotheses at the end of the arrows. In the first one, F is identified with A, and as a result its match list is set to $\{A\}$. For G a new unvisited RGN is generated and connected to B/D with a new edge. In the second hypothesis, G and A are identified, resulting in similar changes to the graph and RGN list.

Let us now assume, another hallway traversal is performed and both hypotheses are updated. Again, the traversed hallway is marked by the dashed arrows and the

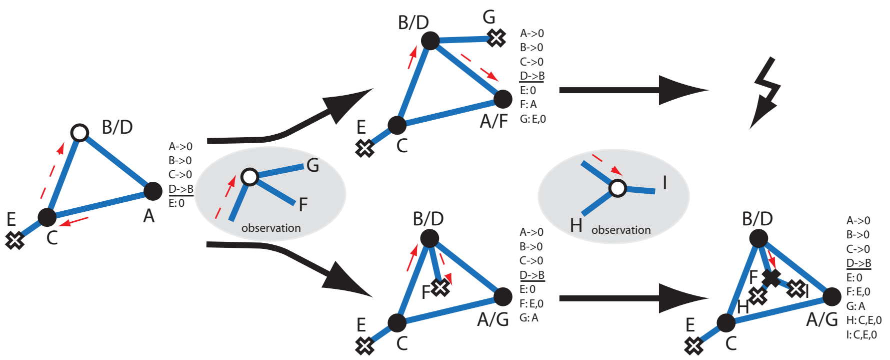


Figure 6.9: Example of two consecutive update operations resulting in one valid successor hypothesis

new observation is shown between the arrows. For the first hypothesis, the traversal would mean that the robot moves to A/F, which is a visited junction (case 1) with two RGEs overall. In this case, there must be a one-to-one mapping between existing RGEs and leaving hallways, which in this example is not possible as there are three observed leaving hallways. Therefore, the hypothesis is discarded. In contrast, updating the second hypothesis results in another instance of case 2, but here with only one possible mapping between RGEs and observed hallways. Hence, only a single updated hypothesis will be generated.

6.2.3.2 Expand Operation

Expanding a node based on matching the next unmatched variable RGN X with an RGN W higher in the RGN list involves the generation of child nodes with modified graph structures and RGN lists for every valid way of folding the graph onto itself so that X and W are merged. The list of new child nodes is returned in the same way as it is by the update operation. In more detail, the expand operation performs the following steps:

1. A list L is initialized as empty, which in the end will contain the successor nodes.
2. As X and W can both be visited or unvisited RGNs, four general cases of merging have to be distinguished. Typically, there are multiple possibilities of merging X with W; and for some cases multiple ways of mapping the RGEs of X to the RGEs of W exist that preserve the cyclic order information. For every possible way of merging and edge mapping a new node c_i is created and processed by the following steps:
   - The graph hypothesis is transformed according to the merging variant and edge mapping. For RGEs of X that correspond to existing RGEs of W

the entire subtree attached to this edge needs to be matched recursively in accordance with the match lists of the involved RGNs. This can lead to additional possibilities, in which case the node c_i is further split into multiple new ones, or to contradictions, in which case the hypothesis is discarded.

- X is marked as matched to W in the RGN list.
- For all unmatched RGNs that get merged in the recursive process, their match list is set to the RGN they are merged with because their matching is now determined as well.
- W is removed from the match lists of the remaining unmatched variables.
- c_i is added to L.

3. L is returned as the result of the expand operation.

As mentioned, there are four general cases of merging, which we will not discuss in detail here. Instead, we will restrict ourselves to providing one rather complex example of matching a visited RGN to an unvisited one. We use the notation $A \rightleftharpoons B$ to refer to the RGE connecting the RGNs A and B.

The starting hypothesis of our example can be seen at the top of Fig. 6.10. RGNs $A - F$ are already assigned; the others are unassigned. The variable to be matched in this expansion step is G and we consider the matching with RGN B. As B is unvisited, there are two possible merging variants, one in which RGE $A \rightleftharpoons B$ corresponds to RGE $G \rightleftharpoons I$ and one in which it corresponds to $G \rightleftharpoons H$.

In the first case, G is merged with B and the RGE $G \rightleftharpoons I$ is removed. The unvisited end node I is merged with A and its match list is changed accordingly. Finally, B is removed from the remaining match lists and the new node is added to the successor list L.

In the second case, H corresponds to B, but H is a visited RGN with two more RGEs. As we are comparing two visited RGNs now, there has to be a one-to-one mapping between the RGEs so that RGE $H \rightleftharpoons K$ has to correspond to $A \rightleftharpoons D$ and $H \rightleftharpoons J$ to $A \rightleftharpoons C/F$. Both RGE pairs are merged in the following. As I is only an unvisited end point, the recursion ends on this side. However, J is a visited RGN which has to correspond to unvisited RGN C/F and there are two possible edge mappings. This means that the current hypothesis has to be split into two new ones, one for each mapping. In the first one, $J \rightleftharpoons M$ is merged with $C/F \rightleftharpoons E$, in the second one, $J \rightleftharpoons L$ with $C/F \rightleftharpoons E$. In both cases, the recursion ends. Overall, we end up with three successor nodes for the original node depicted in the bottom row of the figure.

6.2.4 Two Variants of the Minimal Model Finding Problem

Up to now, we have described a version of the minimal model finding problem in which each model is a complete closed environment which might contain unvisited

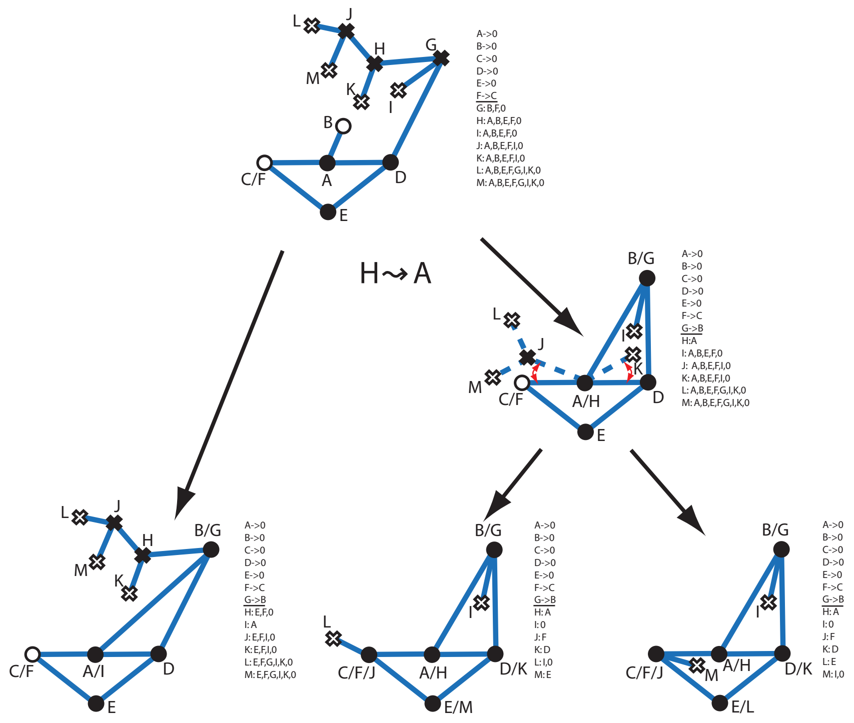


Figure 6.10: Example of matching G to B in an expand operation, resulting in three different child nodes

junctions that form the end points of perceived but never traversed hallways. A less complex version of the problem can be obtained by restricting the models to visited places and allowing hallways with open endings. This problem is less complex because the number of possible matchings is reduced significantly and because the lack of information about the unvisited junctions allows for more variations in general.

In the experimental analysis of this work, we will compare both variants of the algorithm. The version dealing with complete environments will be referred to as the *CompEnv* variant, while the version only determining the layout of the visited parts of the environment will be called the *VisOnly* variant.

The implementation of the VisOnly variant is simply a modified version of the CompEnv algorithm in which variables are only instantiated for junction observations and not for the end points of the leaving hallways. As a result, the search tree from Fig. 6.6 would be reduced to a simple linear chain of three edges as none of the visited RGNs can be joined. While this illustrates the reduced complexity of the VisOnly variant, it is an extreme case as the history only consists of three junction observations and no junction is visited twice.

Besides the complexity issues, the question of which variant is better suited in practice needs to be answered in the context of a concrete application scenario. The advantage of the CompEnv variant is that it includes a certain predictive power which can, for instance, be useful to predict shortcuts when applied to route networks like street networks or the hallway networks used as an example here. However, for more low-level graph abstractions like Voronoi graphs, CompEnv often leads to a higher number of wrong predictions until the entire environment has been explored.

After providing a solution to the purely combinatorial problem, we now turn to the question of how additional constraints, based either on planarity or on direction information, can be incorporated into the search algorithm.

6.3 Pruning Based on Spatial Constraints

As mentioned previously, our approach is to check planarity constraints and consistency of the direction information separately. We start with a discussion of the planarity constraint.

6.3.1 Checking Planarity

In this work, we are exclusively dealing with graph environments that are plane graphs. This means that they are embedded into the plane without crossing edges. This fact allows us to reduce the set of possible hypotheses. Each graph hypothesis for which the cyclic order information does not describe a planar embedding can be immediately discarded because such a graph cannot be drawn into the plane without crossing edges in a way that preserves the cyclic edge orders. The criterion for deciding whether a general graph with a combinatorial embedding describes a planar embedded graph is that its *genus* is 0. The genus of an undirected graph $G = (V, E)$ is given by Euler's formula:

$$\text{genus}(G) = (|E| + 2c - |V| - i - f)/2 \tag{6.4}$$

where c is the number of connected components in the graph, i is the number of nodes of degree 0 (isolated nodes), and f is the number of faces formed by traversing edges in accordance with the cyclic ordering information (a more precise definition will be given below). We only consider connected graphs without isolated nodes here and thus the formula becomes

$$\text{genus}(G) = (|E| - |V| - f)/2 + 1 \tag{6.5}$$

Our approach to planarity checking is similar to the one described in Savelli & Kuipers (2004). First of all, it is advantageous to internally transform the undirected route graphs into bidirected graphs in which each RGE from the original graph is represented by a pair of edges with opposite directions. The information that e and f correspond to the same RGE and thus are *reversals* of each other (rev$(e) = f$ and

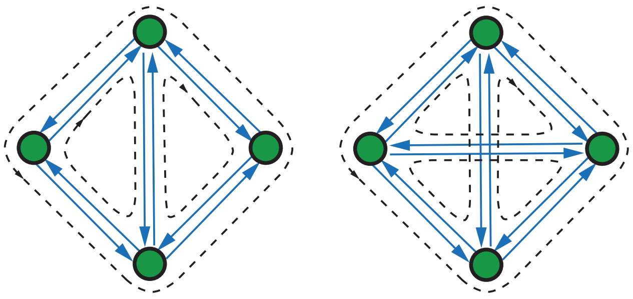


Figure 6.11: Two bidirected graphs of which the first has three faces and thus according to Eq. 6.5 depicts a planar embedding, while the second with an additional (undirected) edge has two faces, which means the embedding is not planar

rev$(f) = e)$ is stored in the form of edge attributes[1]. In addition, the cyclic order information is transformed so that now each RGN v is annotated with the cyclic order of directed edges which have v as source. In the following, we assume that pred(e) and succ(e) yield the predecessor and successor edge of e in the cyclic order of leaving directed edges at the source node of e. One can then define sequences of edges by a function next(e) as follows:

$$\text{next}(e) = \text{succ}(\text{rev}(e)) \tag{6.6}$$

Based on this function, each directed edge now is part of exactly one cycle of directed edges $e_1, e_2, ..., e_n$ with $e_1 = e_n$ and $e_{i+1} = \text{next}(e_i)$. These cycles are called the *faces* of the graph. Figure 6.11 shows two combinatorial embedded graphs. Their faces are depicted by the dashed arrows. The left graph has three faces, while the right one with an additional (undirected) edge has only two. As a consequence, Eq. 6.5 yields that genus $= 0$ for the first graph and genus $= 1$ for the second graph. Hence, only the first depicts a planar combinatorial embedding.

Planarity checking can be performed in linear time (Hopcroft & Tarjan, 1974; Lempel et al., 1967). We integrate planarity checking into our search algorithm by representing the route graph hypotheses as bidirected graphs and updating the face information and pointers for pred, succ, and next whenever we modify the graph structure. As soon as Eq. 6.5 is violated, the hypothesis at hand can be discarded as the planarity constraint is violated. In our approach, the faces are numbered and each edge of the bidirected graph stores the number of the face it belongs to (given by facenumber(e)). The relevant operation which has the potential of changing planarity in our approach is inserting a new edge into the graph. After inserting an edge which results in the two new directed edges e and rev(e) and updating the successor information, three cases have to be distinguished, illustrated in Fig. 6.12:

[1]The resulting structure is often referred to as a "map" (Mehlhorn et al., 1999) but we will not use this term here in order to avoid ambiguities.

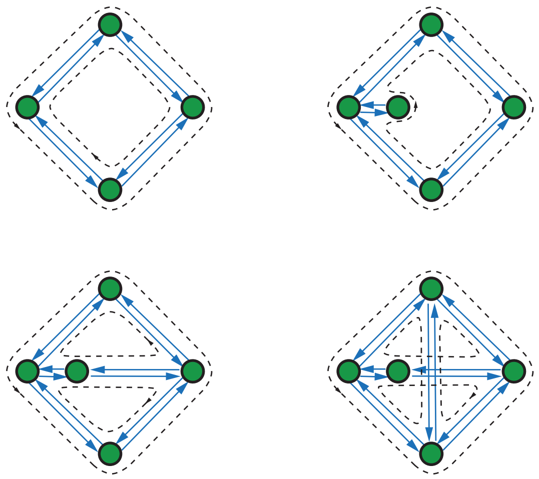


Figure 6.12: Three cases of modifying a graph, starting with the planar graph at the top left. Only in the last case (bottom right) does the genus change, and, hence, the resulting embedding is not planar

1. $\text{next}(e) = \text{rev}(e)$: This is the case when we insert a new node and connect it to an old one. The number of faces stays the same while the number of nodes and edges increase by one, leaving the genus unchanged. The face numbers of both, e and $\text{rev}(e)$ are set to the number of $\text{next}(\text{rev}(e))$.
2. $\text{facenumber}(\text{next}(e)) = \text{facenumber}(\text{next}(\text{rev}(e)))$: The new edge connects two already contained nodes and splits an existing face into two new ones. The total number of faces increases by 1. The face number of e is set to the number of the old cycle, while $\text{rev}(e)$ gets a new number, and numbers of all edges belonging to the same face as $\text{rev}(e)$ are updated accordingly. Since the number of edges also increased by one and the node remained unchanged, the genus again remains unchanged.
3. $\text{facenumber}(\text{next}(e)) \neq \text{facenumber}(\text{next}(\text{rev}(e)))$: In this case, the old face would be replaced by a new one that combines both faces, and hence the face number would decrease by 1. The edge number would increase by one, while the number of nodes remains the same. As a result, the genus would increase to 1. Since this means that the embedding is not planar anymore, the hypothesis can be immediately discarded.

While the first and third case require constant time, the second case takes linear time in the number of edges because of the need to update the face numbers for one face. We provide more details on the incorporation of the planarity check into the overall search algorithm in Sect. 6.3.3.

6.3.2 Checking Spatial Consistency

One of the main goals of the work described in this chapter is to investigate how the presence of spatial information provided in the form of qualitative direction relations that only represent coarse information but can be perceived reliably reduces the number of hypotheses that have to be considered. Checking the consistency of a route graph hypothesis with regard to spatial constraints stemming from the perceived directions of the leaving hallways requires us to determine whether an assignment of points in the plane to the RGNs of the hypothesis exists which induces the same set of relations.

Hence, we are faced with a constraint satisfaction problem in which the domain (points in $\mathbb{R}^2$) is infinite. However, research on qualitative spatial reasoning has produced constraint-based techniques to deal with this kind of problem. The solution typically consists of a qualitative constraint calculus defining a set of spatial relations and algebraic operations like converse and composition on the set of relations. Depending on the particular calculus, consistency checking can be performed by employing the so-called algebraic closure algorithm or a more involved backtracking search over the set of all possible scenarios which are then tested again by the algebraic closure algorithm. The algebraic closure algorithm requires $O(n^3)$ time, where n is the number of related objects. We provide an overview on these concepts and techniques in Appendix B.

Based on these result, our approach is to formulate the perceived direction information in the form of qualitative direction relations from particular qualitative spatial calculi, derive a network of constraints from the given route graph hypothesis, and apply the standard consistency check methods to the constraint network. For performing the consistency check we use the spatial reasoning toolbox SparQ, which provides implementations of a large set of spatial calculi and the standard reasoning techniques (Wallgrün et al., 2006, 2007).

As we are particularly interested in comparing the effects of absolute and relative direction information on the search space and on the number of solutions, we have chosen the absolute cardinal direction calculus (Ligozat, 1998) and the relative $\mathcal{OPRA}_2$ calculus (Moratz, 2006; Moratz et al., 2005) for our analysis.

Both calculi cannot be considered as ideal, but no better candidates or other reasoning formalisms exist to our knowledge. Hence, the problem investigated here can also be seen as a challenge for qualitative spatial reasoning research. As an ideal calculus we would consider one with the following properties:

1. good computational properties with regard to the consistency check,
2. expressive enough to rule out many hypotheses,
3. dealing with relations that are easily and reliably accessible,
4. able to express the cyclic edge order information.

The cardinal direction calculus, on the one hand, is rather efficient as a large tractable subset exists for which the algebraic closure algorithm decides consistency (cf. Appendix B.4). This subset contains all relations required in our context. The downside is that the calculus does not allow for expressing the cyclic ordering information about the leaving RGEs in the route graph. As a result, it can happen that a constraint network deemed consistent by the consistency check only has solutions for which the cyclic order information is not preserved.

The $\mathcal{OPRA}_2$ calculus, on the other hand, can express the cyclic ordering information. However, employing the algebraic closure algorithm to $\mathcal{OPRA}_2$ constraint networks generated from our route graph hypotheses only results in an incomplete method to rule out inconsistent cases. This is also true if the much more inefficient backtracking search would be employed because algebraic closure does not decide consistency even if the constraints are all base relations. We still have chosen $\mathcal{OPRA}_2$ as to our knowledge no relative direction calculus exists with significantly better computational properties. In addition, the calculus offers a similar level of granularity as that of the cardinal direction calculus. This is beneficial for the comparison. How problematic the application of an incomplete consistency checking method is has to be evaluated experimentally.

In the next two sections, we describe how we model the direction information for both calculi.

6.3.2.1 Modeling Spatial Configurations in the Cardinal Direction Calculus

The cardinal direction calculus is an absolute binary qualitative direction calculus describing the cardinal direction of one point object from another point object using the nine base relations we saw in Fig. 6.4. Here, we use the base relations from the calculus to describe the directions of leaving hallways as seen from the corresponding junction, but then transfer this information into a constraint over the possible positions of the connected junctions in the plane. As a result, each RGE in a route graph hypothesis yields a direction constraint between the connected RGNs.

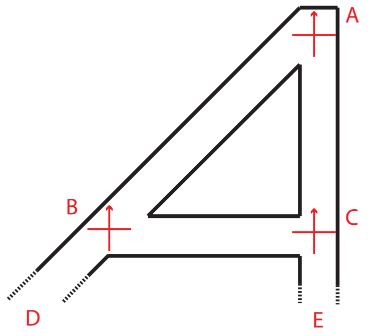
Absolute direction information like the cardinal direction information can actually be exploited in multiple ways in the minimal model finding algorithm. It can be used to enforce three different requirements:

1. **Valid direction orderings:** When adding a new RGE to an RGN, it can only be inserted into the cyclic edge order at a position where the edges also preserve the cyclic order of cardinal directions. For instance, a resulting cyclic order of edges with directions n, s, w is not valid as w would have to appear between n and s.

2. **Valid junction matching:** When matching two RGNs, mappings of RGEs are only valid if corresponding RGEs have the same directions.

3. **Global consistency:** There needs to be a way of assigning coordinates to the RGN such that the direction constraints are satisfied.

As we explained, the global consistency check requires the full constraint reasoning approach based on the algebraic closure algorithm. Therefore, we first generate a constraint network from the given route graph hypothesis. This constraint network consists of one variable for each RGN and one constraint represented by a directed edge for each RGE. In Fig. 6.13, we see part of a route graph hypothesis and the corresponding set of derived constraints that make up the constraint network. For all other pairs of junctions, the constraint holding between them is the disjunction of all base relations except eq. The constraint network is then fed into SparQ, which performs the consistency check. If the algebraic closure algorithm discovers an inconsistency, the hypothesis at hand can be discarded.



A	ne	B
A	n	C
B	w	C
B	ne	D
C	n	E

Figure 6.13: A route graph hypothesis and the cardinal direction constraints derived from it

6.3.2.2 Modeling Spatial Configurations in the $\mathcal{OPRA}_2$ Calculus

The second calculus employed and investigated in this book is the $\mathcal{OPRA}_2$ calculus. In contrast to the cardinal direction calculus, it is a relative calculus describing the relative orientation of two objects to each other. Hence, a robot would only need to be able to estimate the angles between the leaving hallways and would not require a compass to determine $\mathcal{OPRA}_2$ relations.

$\mathcal{OPRA}_2$ is actually one particular instance of a calculus from the Oriented Point Relation Algebra ($\mathcal{OPRA}_m$) family. m here is the granularity parameter used to determine the number of base relations that are distinguished (Moratz, 2006; Moratz et al., 2005). The domain of ($\mathcal{OPRA}_m$) is the set of oriented points (points in the plane with an additional direction parameter).

For a given granularity parameter $m \in \mathbb{N}$ the concrete set of $\mathcal{OPRA}_m$ relations is derived as follows: For each of the two related oriented points, m lines are used to

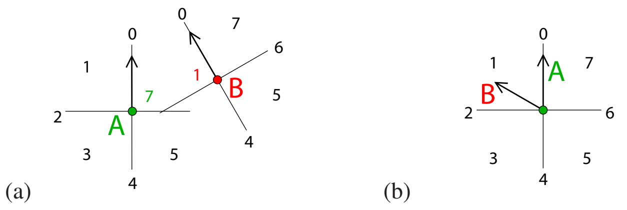


Figure 6.14: $\mathcal{OPRA}_2$ relations between two oriented points: (a) the relation $A\ {}_2\angle_7^1\ B$, (b) $A\ {}_2\angle 1\ B$

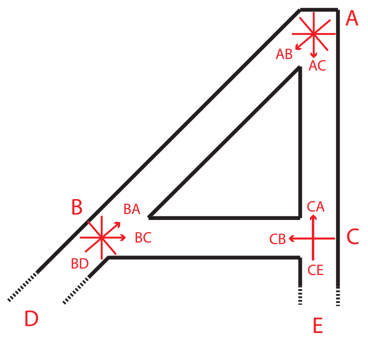
partition the plane into $2m$ planar and $2m$ linear regions. Figure 6.14(a) shows the partition for $\mathcal{OPRA}_2$. The orientation of the two points is depicted by the arrows starting at A and B, respectively. The regions are numbered from 0 to $4m-1$. Region 0 always coincides with the orientation of the point. An $\mathcal{OPRA}_m$ base relation $rel_{\mathcal{OPRA}_m}$ is then given by a pair (i, j) where i is the number of the region of A which contains B, while j is the number of the region of B which contains A. These relations are usually written as $A\ {}_m\angle_i^j\ B$, Thus, the example in Fig. 6.14(a) depicts the relation $A\ {}_2\angle_7^1\ B$. Additional base relations called *same relations* describe situations in which the positions of both oriented points coincide. In these cases, the relation is determined by the number s of the region of A which contains the orientation arrow of B (as illustrated in Fig. 6.14(b)). These relations are written as $A\ {}_2\angle s\ B$ ($A\ {}_2\angle 1\ B$ in the example). The complete set $\mathcal{R}$ of $\mathcal{OPRA}_m$ relations again is the power set of the base relations.

When we employ the $\mathcal{OPRA}_2$ calculus to describe the relative directions of leaving hallways in the junction observations, the leaving hallways are seen as oriented points positioned on the RGN and pointing in the corresponding direction. The induced spatial description $R(J)$ for a junction observation J then consists of an $\mathcal{OPRA}_2$ relation for each pair of leaving hallways from $L(J)$.

For a relative direction calculus like $\mathcal{OPRA}_2$, only two ways of exploiting the direction information exist, in contrast to the three ways we encountered in the case of absolute direction information. The reason is that enforcement of valid direction ordering is not applicable because no such order exists for relative information. Enforcing valid junction matchings, however, is still possible but now constrains the valid mappings by way of the relations holding between pairs of RGEs.

For the global consistency check, an $\mathcal{OPRA}_2$ constraint network is generated from the route graph hypothesis. Analogously to the generation of the description of a junction observation, one oriented point variable is introduced for each pair of RGN and incident RGE. Hence, we end up with $2 \times n$ variables in the constraint network, where n is the number of RGEs in the hypothesis, while we only had one per junction in the case of absolute cardinal directions. The process of generating the constraint network is illustrated in Fig. 6.15.

The names of the oriented points here are formed from the name of the corres-



AB	${}_2\angle 1$	AC
BA	${}_2\angle 7$	BC
BA	${}_2\angle 4$	BD
BC	${}_2\angle 5$	BD
CA	${}_2\angle 2$	CB
CA	${}_2\angle 4$	CE
CB	${}_2\angle 2$	CE
AB	${}_2\angle_0^0$	BA
AC	${}_2\angle_0^0$	CA
BC	${}_2\angle_0^0$	CB
BD	${}_2\angle_0^0$	DB
CE	${}_2\angle_0^0$	EC

Figure 6.15: Set of $\mathcal{OPRA}_2$ constraints describing the given route graph hypothesis. A junction of n hallways is represented by n oriented points

ponding RGN and the name of the other RGN incident to the RGE (e.g., AB for the oriented point at RGN A and the RGE leading to B). For each RGN, we generate the constraints holding between each pair of leaving RGEs which are all *same relations*. In addition, we need to state that XY and YX are facing each other (relation ${}_2\angle_0^0$), forming a single hallway. The complete set of constraints is shown on the right side of the figure.

Finally, consistency again is checked by using the algebraic closure algorithm of SparQ. However, as we already mentioned, this is only an incomplete method that may not discover all inconsistent constraint networks.

6.3.3 Incorporation into the Search Algorithm

Planarity checking and spatial direction constraints are incorporated into the search algorithm to discard inconsistent hypotheses as soon as possible and thus prune large subtrees of the search tree.

Spatial direction constraints are utilized in both the update and the expand operations. In the update operation, only valid junction matchings need to be enforced in order to verify that the hypothesis is consistent with the new information. In the expand operation, enforcement of valid direction orderings plays a role when adding a new RGE to an unvisited RGN, but only when an absolute direction calculus is used. Valid junction mappings are enforced when two RGNs are merged. A global consistency check is performed for every successor hypothesis that results from performing a matching.

Planarity checks only need to be performed when a node in the search tree is expanded. The update operation at most appends new RGEs together with a new unvisited RGN and never connects two existing RGNs. However, it still requires that the planarity-related information be updated correctly. In addition, when the match lists of the new RGNs are set, only RGNs that share a face with the new RGN are considered.

In the expand operation, planarity is checked whenever a new edge connecting two existing RGNs is inserted while transforming the graph structure.

The effects of both the planarity constraint and the direction constraints can be seen in the top picture of Fig. 6.8. A significant number of branches have been cut off because inconsistency of the hypothesis has been discovered (marked by the cross). The reasons for discarding a particular node are indicated by the letters below the cross ('p' for planarity, 'd' for direction information). For some nodes, both planarity and spatial consistency are violated. Leaf nodes containing a consistent map hypothesis are tagged by a check mark.

The bottom search tree shows the result of applying pruning based on planarity constraint and direction information together with the branch and bound search. From 112 nodes in the original search tree, only 36 are considered. Ten nodes are rejected because of planarity violation and ten because of inconsistent direction information. While generation of successor nodes and the update operation can be performed in polynomial time (with the global consistency check being the most costly operation), the size of the search tree grows exponentially with the length of history. Therefore, the important question is whether combining constraint-based pruning and best-first search can achieve a sufficient reduction of the search space to make the overall approach feasible. An experimental analysis of this issue based on randomly created graph environments and exploration runs will be conducted in Sect. 7.3.

6.4 Combining Minimal Route Graph Mapping and AGVG Representations

In the last section of this chapter, we discuss how the minimal model finding approach developed above can be applied to construct AGVG representations from a sequence of observed Voronoi nodes and traversals of Voronoi curves. Obviously, this global mapping approach should be based on the most relevant of the Voronoi nodes and only add the other nodes when the general topology of the environment has been established.

The AGVG setting deviates in several aspects from the theoretical scenario we studied in the previous sections:

- node signatures provide additional information about RGNs,
- start and end nodes of RGEs (Voronoi curves) can often be perceived together,
- multiple connections between two RGNs are possible,
- reliable perception of direction relations is not given for linear relations or near the sector boundaries,
- connections are typically not straight lines.

The information contained in the signatures of Voronoi nodes can be used as an additional criterion to decide whether two nodes are compatible as described in Chap. 5. The second point in the list refers to the fact that in a Voronoi-based mapping approach the robot typically not only perceives a single Voronoi node but a local Voronoi graph as defined in Sect. 3.5. This kind of information allows us to extend the route graph model without explicitly traversing each edge and, in addition, helps reduce the problems caused by the other deviations from the theoretical framework as we will see below. The simplest way to incorporate this additional information into the framework is by adding virtual traversal actions and junction observations to the history whenever a complete connection is perceived without actually traversing it. These virtual actions and observations simulate traversing the connecting Voronoi curve and then returning to the starting Voronoi node.

Including the possibility of multiple connections between two Voronoi nodes in the framework is straightforward. It only requires a change in the way the match lists are constructed in the update operation. This change, in principle, increases the size of the search space. However, the fact that the adjacent nodes are often part of the local observation means that the difference is negligible in practice.

More serious problems are raised by the last two points in the list. First of all, we cannot expect that the direction relations of leaving Voronoi curves can be completely reliably observed in practice. This is especially true for the linear sectors included in the two direction calculi, which is a general point of criticism with regard to typical qualitative spatial calculi. As a consequence, instead of always employing base relations from the respective calculus, we utilize disjunctions of base relations whenever the perceived direction is a linear relation or lies close to the boundary of a relation sector. For instance, the perceived cardinal direction relation n and a direction belonging to ne but very close to n would both be stored as the disjunction $\{ne, n, nw\}$. When employing disjunctions instead of only base relations, the requirement that directions of matched hallways be identical has to be replaced with the demand that the intersection of the direction relations not be empty.

A further problem for the utilization of direction information is the fact that Voronoi curves are typically not straight connections between two Voronoi nodes but, as the name suggests, curved. Therefore, we cannot expect that a connection leaving node A to the southwest arrives at node B from the northeast and that consequently B has to lie southwest of A. If the connecting Voronoi curve is completely contained in the local observation, this is not a problem as the correct cardinal direction can be read off directly. In situations in which this is not the case, we simply mark the traversal action and refrain from employing the direction information for this edge in the global consistency check. However, we still can use the local direction of the leaving Voronoi curve for matching junction observations.

The last two points and the adaptations made to deal with them mainly concern the pruning of the search space based on global consistency. The extended use of coarse information in the form of disjunctions leads to a diminished inferential power. As a result the efficiency of this kind of pruning can be significantly reduced. Enforcement of valid direction orderings and valid junction matchings are affected to a lesser degree.

Overall, while these adaptations may sound rather drastic, the effects in practice are less severe because of the already mentioned extended observation range. A quantitative analysis will be performed as part of the evaluation described in the next chapter (Sect. 7.3.5). This analysis will be based on simulated exploration runs through AGVGs of real environments. In addition, we will apply the minimal model approach within an overall Voronoi-based mapping system that combines all the techniques developed in this work in Sect. 7.4.

Chapter 7

Experimental Evaluation

In this chapter, we provide a compilation of the different experimental analyses we conducted in order to evaluate the techniques and approaches developed in the previous four chapters. The evaluation also involves the integration of our methods into different kinds of mapping systems. At the end of the chapter, we describe a mapping approach that combines all developed techniques into an overall multi-hypothesis mapping system.

7.1 Relevance Assessment and HAGVG Construction

We start this chapter by presenting the evaluation done for the Voronoi node relevance assessment, the AGVG simplification, and the HAGVG construction techniques described in Chap. 4. This evaluation consists of two parts: First, we analyze the efficiency of the relevance value computation algorithms in order to demonstrate that the relevance computation is sufficiently fast in practice. Second, we combine these techniques in an overlay representation approach with the goal of showing its ability to generate suitable hierarchical route graph representations for several data sets of real environments.

7.1.1 Efficiency of the Relevance Computation Algorithms

To empirically evaluate the efficiency of the relevance value, we implemented a random graph generator that produces pseudo-AGVGs. It first produces a 3-regular graph for a given number of edges. The nodes are placed on a hexagonal grid and each node is connected with its three neighbors in the grid. In a second step the number of nodes is varied by splitting nodes of degree 3 into three nodes of degree 1 while ensuring that the graph remains connected. This procedure allows us to generate AGVGs with a desired edge-node ratio. This ratio is directly related to the number of cycles appearing in the graph: the higher the edge-node ratio, the higher the number of cycles in the graph. In a final step, the length attributes of the edges and the radius attributes of the nodes

J.O. Wallgrün, *Hierarchical Voronoi Graphs: Spatial Representation and Reasoning for Mobile Robots*, DOI 10.1007/978-3-642-10345-2_7, © Springer-Verlag Berlin Heidelberg 2010

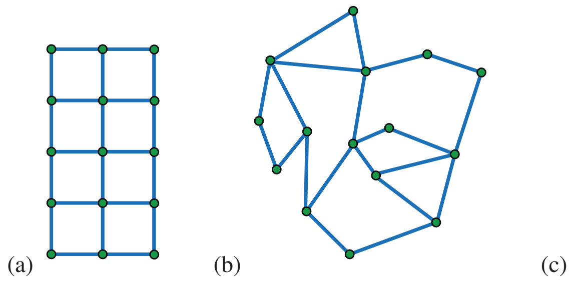


Figure 7.1: Examples of the three types of randomly generated test environments that are used in this chapter: (a) a grid environment, (b) a Delaunay triangulation-based random graph, and (c) a pseudo-AGVG

are varied randomly. An example of such a pseudo-AGVG is shown in Fig. 7.1(c). The edge-node ratio of the generated pseudo-AGVGs lies between 1 (tree-formed AGVG) and 1.5 (3-regular AGVG).

In our comparison, we systematically varied the number of edges and nodes in the randomly generated AGVGs and applied both the basic variant and the improved variant of the relevance computation algorithm to the resulting graphs. The resulting computation times on a 2 GHz Pentium M CPU are shown in Figs. 7.2(a) and 7.2(c).[1] All data points are averages taken over 30 different pseudo-AGVGs. Figures 7.2(b) and 7.2(d) each show two cuts through the data sets for fixed edge-node ratios, one for a high ratio ($\frac{|E|}{|V|} > 1.47$) and one for a ratio close to 1 ($\frac{|E|}{|V|} < 1.1$). As the data shows, the computation time of the basic version of the value computation algorithm increases significantly when the ratio of edges to nodes goes towards 1. The reason is that in this case the graphs contain almost no cycles and almost the complete graph is expanded for each node.

In contrast, the improved version only shows a slight increase towards lower edge-node ratios as typically only a small part of the graph needs to be considered even for tree-like structures. Overall, the computation times stayed below 40 ms even for very large graphs with 1,000 nodes and edges.

As a result, the improved algorithm is efficient enough for all applications considered in this work, even for utilization within a particle filter mapping approach in which an update of the relevance values has to take place for every sample in each update step. If required, a further reduction of computation time can be achieved by employing the presented incremental update method described in Sect. 4.7.

[1]Both diagrams show an increasing number of gaps for higher edge and node numbers and an edge-node ratio close to 1. These are cases in which the random graph generator failed to create suitable instances in reasonable time.

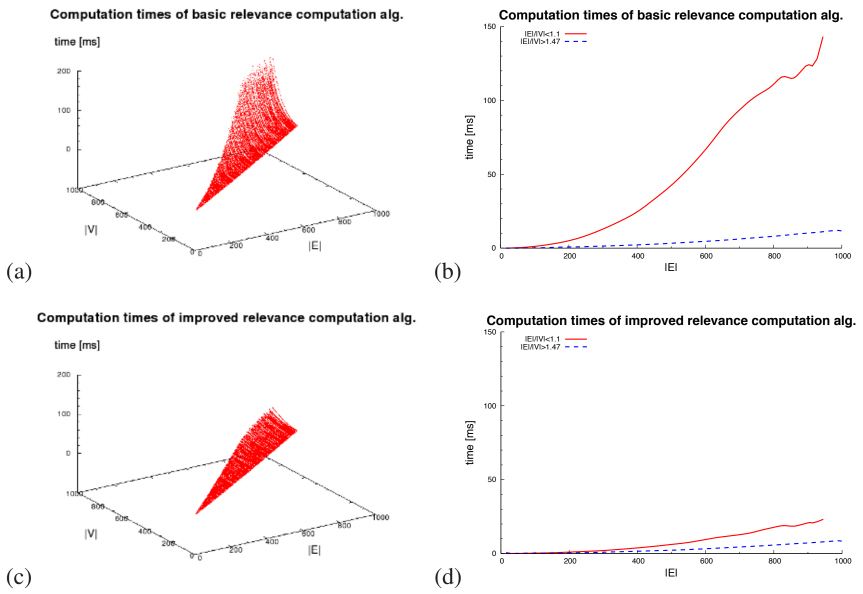


Figure 7.2: Computation times for (a) the basic relevance value computation algorithm and (c) the improved version depending on the number of nodes and edges in the randomly generated pseudo-AGVGs. (b) and (d) show the curves for AGVGs with a high edge-node ratio and graphs with an edge-node ratio close to 1

7.1.2 Combining the HAGVG Construction Methods with a Grid-Based FastSLAM Approach

In a first experiment to assess the ability of the developed HAGVG generation methods to construct adequate route graph representations for real environments perceived through 2D range sensors, we combined them with a grid-map-based FastSLAM approach (Hähnel et al., 2003a; Stachniss et al., 2005). The result is an overlay representation approach with a grid map representation at the bottom and an HAGVG on top of it. In contrast to our general approach of building the route graph representation directly by incrementally merging locally computed AGVGs, we derive here the lowest AGVG layer of the HAGVG from the global grid map representation.

The grid map is constructed using a Rao-Blackwellized particle filter as described in Appendix A.3.2.2. In contrast, to the approach described in Thrun (1998), the HAGVG is computed not only after a complete grid map has been constructed, but each time a new observation is incorporated. However, we only compute it for the most likely particle.

Before we can apply our HAGVG construction techniques, we have to compute the underlying AGVG from the occupancy grid. This is done by the following steps:

1. A threshold is applied to the smoothed occupancy grid of the most likely particle, turning it into a binary grid in which each cell is classified as either occupied or empty.
2. Another binary grid representing a discrete GVD of the environment is extracted by employing a flux-based thinning approach (see Dimitrov et al., 2000).
3. The GVG is constructed by tracing the pixels forming the discrete GVD. In addition, attributes (e.g., the node signatures) are determined and annotated to the graph structure. The result is the final AGVG.

In Figs. 3.8 and 3.9 of Chap. 3, we saw two results of this approach for two different environments. In both cases, two-level HAGVGs have been constructed. Figure 7.3 illustrates the individual steps of the combined approach for the environment from Fig. 3.8: Figure 7.3(a) shows the occupancy grid belonging to the most likely particle while the exploration is still in progress. The trajectories corresponding to all 30 particles used are shown by the red polylines. In Fig. 7.3(b), we see the smoothed grid map and the discrete GVD resulting from the thinning approach. Figures 7.3(c) and (d) show the constructed AGVG. Figure 7.3(d) also depicts the nodes' maximal inscribed circles and visualizes the vnrm values of the nodes by the node radii. In Fig. 7.3(e), we see the simplified top-level AGVG which has been extracted with a threshold value of $\theta = 2{,}500\,\mathrm{mm}$. The graph has been reduced from 168 nodes and 202 edges to 46 nodes and 61 edges by the simplification. For the second environment (Fig. 3.9) the approach lead to a reduction from 1,040 nodes and 1,189 edges to 308 nodes and 456 edges.

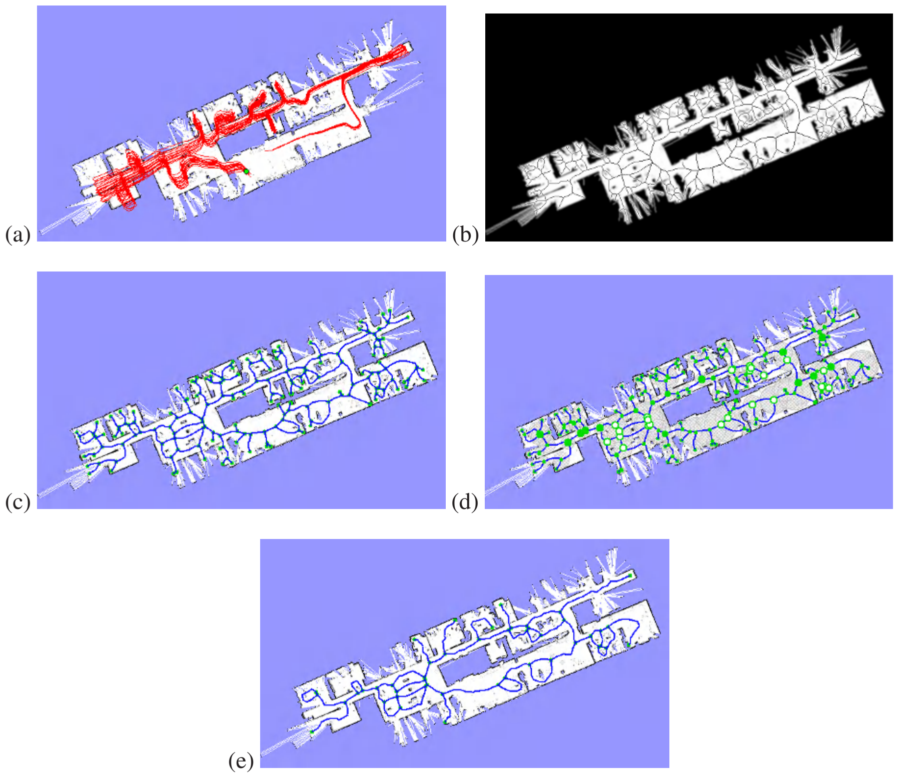


Figure 7.3: Steps of the HAGVG construction in the combined grid-based FastSLAM approach: the grid of the most likely particle together with trajectories corresponding to all particles (a), the smoothed grid with the extracted GVD (b), the constructed AGVG (c), the AGVG again together with maximal inscribed circles and vnrm values depicted by node radius (d), and the top-level AGVG derived with a threshold of 2,500 mm (e)

In Fig. 7.4, we show the results of applying different threshold levels to compute a coarser AGVG. The threshold values chosen are 500 mm, 1,000 mm, 4,500 mm, and 11,000 mm.

The experiments with the real environments reflect the results obtained for the artificially created AGVGs in the previous section. For the first environment, the basic algorithm took 19 ms and 11,968 expansion steps, while the improved version took 2 ms and 2,597 steps. For the second example, the values were 368 ms, 142,360 expansions and 24 ms, 20,650 expansions, respectively. The limiting factor in this overlay-based mapping approach turned out to be the extraction of the discrete GVD, which has a complexity of $O(n \log n)$ where n is number of cells in the grid. This started to cause delays when we applied the approach to data sets of very large environments and reveals the downsides of the overlay approach, which requires recomputation of the route graph representation for each step.

7.2 Evaluation of the Voronoi-Based Data Association

We evaluated the performance of our AGVG matching approach by comparing its performance with that of the standard ICNN data association approach. The goal of this evaluation was to determine the differences in the quality of the computed associations and the required computation times. The compared versions are the complete AGVG matching approach as described in Sect. 5.3.3 and the ICNN algorithm given in Algorithm 5 treating the Voronoi nodes as features and including two modifications: (1) We replace the validation gate in line 3 by the unary compatibility criterion from Eq. 5.18, but without the local geometry criterion compatible$_{\text{localgeometry}}$, which is not applicable without reference to the graph topology of the AGVG. (2) We enforce the all-different constraint by not considering already matched map features for further assignments. In doing so, we match the nodes from the local AGVG in order of decreasing relevance as given by their vnrm values. As one advantage of the AGVG matching approach is that, as a batch association method, it is not completely reliant on pose estimates, we also evaluate its performance when no pose information is available.

For the comparison of ICNN and AGVG matching, we incorporated both data association algorithms into a feature-based FastSLAM algorithm based on Voronoi nodes with the goal of generating different data association problems. A description of the feature-based Rao-Blackwellized particle filter approach underlying this algorithm is given in Appendix A.3.2.1. The feature-based FastSLAM algorithm maintains the global position estimates for the Voronoi nodes used for the matching.

The following modifications were made for the evaluation:

- The input of the algorithm is a sequence of local AGVGs extracted from small local metric maps describing the robot's immediate surroundings (see Sect. 7.4.1 for a more detailed description). A low threshold value of $\theta = 5\,\mathrm{mm}$ is used to filter out unstable parts based on our simplification algorithm. The local metric

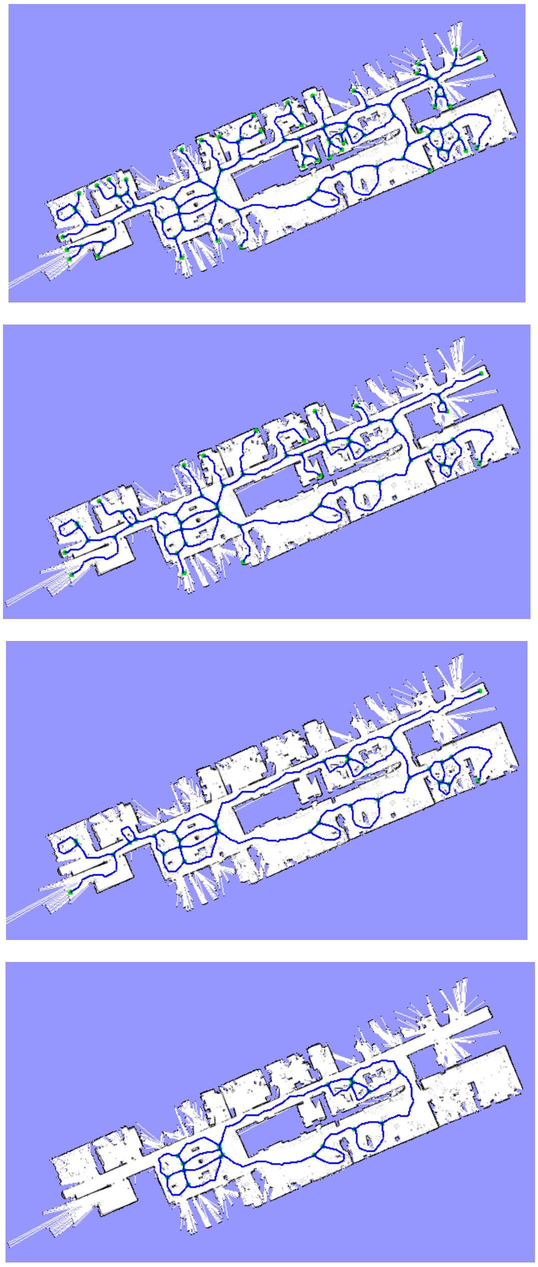


Figure 7.4: Different top-level AGVGs constructed with $\theta = 500\,\mathrm{mm}$, $\theta = 1{,}000\,\mathrm{mm}$, $\theta = 4{,}500\,\mathrm{mm}$, and $\theta = 11{,}000\,\mathrm{mm}$

maps are computed from exploration data collected by a robot equipped with a laser scanner. Ground truth information about the identity of the Voronoi nodes has been manually added to the AGVGs. In addition, the input data contains odometry information relating the local AGVGs.

- For each particle, a complete map AGVG is stored instead of only a set of Voronoi node features.
- Every time a data association needs to be computed, this is done for both ICNN and our AGVG matching, and the results are stored together with the required computation times.
- The update of the map AGVG is based on the the results of the AGVG matching.

In our experiments, we ran the algorithm for two different environments using different start positions in the data sets. The local maps contained on average 19 Voronoi nodes. The size of the map AGVG varied between 9 and 588 nodes.

To compare the data association quality of the two algorithms, we recorded the number of wrong assignments in each data association. As correct associations are more critical when they concern more relevant Voronoi nodes, we also determined the average vnrm values of the wrongly associated Voronoi nodes. In addition, we recorded the computation time for each data association. For the AGVG matching algorithm we also determined the fraction of entries in the edit distance table that actually had to be calculated. The experiment was performed on a 2 GHz Pentium M CPU.

Table 7.1 summarizes the results regarding the data association quality for the experiment using the global position estimates. AGVG matching shows a significantly increased data association quality. 94.67% of all decisions are correct in contrast to 71.50% for the modified ICNN algorithm. Additionally, the average relevance of wrongly associated nodes is much lower, which means the errors are made for more irrelevant nodes. In particular, we observed that AGVG matching never made wrong associations for the most relevant Voronoi nodes of the local map, while ICNN frequently inverts the assignments of very relevant Voronoi nodes positioned close to each other.

Method	Average number of errors per data association	Percentage of correct assignments	Average vnrm value for wrong assignments
ICNN	3.42	71.50%	0.55
AGVG matching	0.64	94.67%	0.19

Table 7.1: Comparison of data association quality of ICNN and AGVG matching

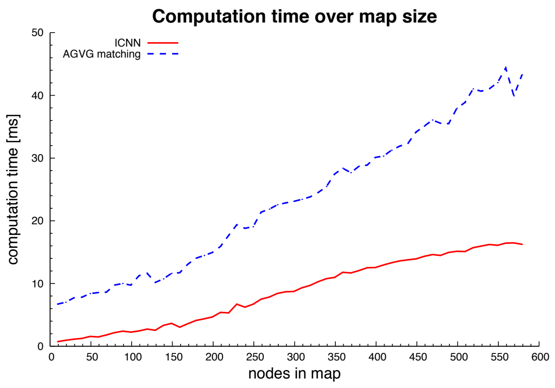


Figure 7.5: Comparison of average computation times of ICNN and AGVG matching depending on the number of nodes in the map AGVG

While an improved data association quality was to be expected because AGVG matching is a batch association technique whereas ICNN makes individual assignments, the more important question is about the computational costs at which this improvement can be achieved. Figure 7.5 depicts the average computation times over the size of the map AGVG (given by the number of Voronoi nodes). It turns out that the increase in computational costs is surprisingly low. In contrast to the theoretical upper bound of cubic dependency on the size of the map AGVG, the increase in computation time was almost linear in the investigated range of map sizes. This shows the effectiveness of the constraint-based pruning and the employed branch and bound techniques. On average, only 0.00001% of the entries in the edit distance table had to be computed.

We then repeated the previous experiment for the AGVG matching algorithm, but this time without employing the pose information in the matching algorithm. This means that the compatibility criteria $\text{compatible}_{\text{absdist}}$ and $\text{compatible}_{\text{absdist-gp}}$ were removed from $\text{compatible}_{\text{unary}}$ (Eq. 5.18). The result of this experiment was that in 87.8% of all cases the AGVG matching approach determined the same data association as with pose information and, hence, was able to localize the robot correctly. In the absence of pose information, AGVG matching still achieved 83.21% correct assignments. As a result, the approach is also well suited for global localization or map merging applications. We observed that cases in which the matching resulted in an entirely wrong localization typically involved local AGVGs with very few nodes, leading to a high level of ambiguity. The average computation times shown in Fig. 7.6 were noticeably higher than in the first experiment as more possibilities needed to be explored. However, the approach still scaled well with the size of the map AGVG. The average percentage of calculated entries in the edit distance table increased to 0.00009% but remained very low in general.

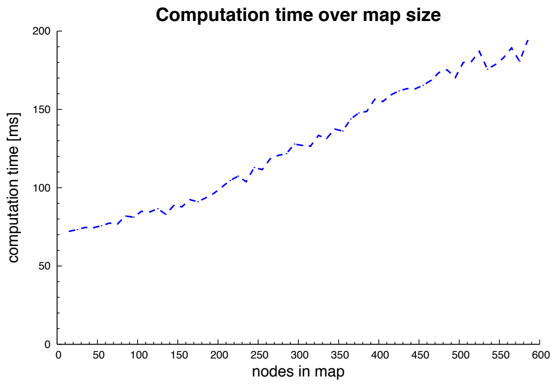


Figure 7.6: Average computation times of AGVG matching without pose information

Overall, the experiments described in this section demonstrate that the AGVG matching approach achieves an excellent ratio between quality and computational cost. As a result, the approach is not only well suited when a high level of reliability in the data association is required (e.g., in single-hypothesis Kalman-filter-based approaches) but also when computational efficiency is required, as in a particle-filter-based mapping approach, in which multiple data associations need to be determined in each update step. Furthermore, the exploitation of configurational information makes it applicable for tasks in which no pose information is available. The main shortcoming of the approach is that it does not form hypotheses which involve closing a loop in the AGVG. Hence, if it is the goal to use a mapping approach like the feature-based particle filter employed in the experiment for global mapping, an extension for loop closing would be required. For tracking and localization purposes this is not necessary.

7.3 Evaluation of the Minimal Route Graph Approach

With regard to the minimal route graph model approach to global mapping described in Chapter 6, we were especially interested in the following aspects:

- the ability of the overall approach to disambiguate different route graph hypotheses and derive correct route graph models,
- the ability of the approach to prune the search space,
- the individual contributions of planarity and direction constraints,
- the differences between absolute and relative direction constraints,
- and the applicability of the approach to Voronoi graph representations.

We evaluated these questions in several experiments, which are described in the following.

7.3.1 Solution Quality

We first investigated how much the planarity constraint and qualitative direction information helps in order to improve the solution quality by ruling out incorrect hypotheses and, as a result, increase the frequency with which the correct solution is found by the minimal model approach. Note that with 'correct' we mean here that the hypothesis reflects the actual environment and not only that it is a consistent map hypothesis as defined in Definition 6.5.

To measure the quality of a solution, we use a simple error measure: We count the number of wrong decisions in a hypothesis. A wrong decision is made when either two RGNs have been merged that correspond to different junctions in the environment or two RGNs that correspond to the same junction have not been merged. We will refer to this quality measure for map hypotheses as their *error distance*.

In the experiment, we employed randomly generated environments of the three types shown in Fig. 7.1. The aim of employing these three types is to get a good mixture of environments with different properties that are likely to affect the performance of the minimal model approach. Grid environments (Fig. 7.1(a)), for instance, show a high degree of perceptual aliasing as most junctions look the same even when absolute direction information is available. Thus, they are more dependent on pruning effects of planarity and global spatial consistency.

The second type of environment shown in Fig. 7.1(b) consists of graphs that have been constructed by randomly placing the nodes in the plane, computing the Delaunay triangulation for these points, and finally removing edges randomly. As a result, we get graphs which still contain many small cycles but the perceptual aliasing is significantly reduced. In addition, multiple leaving edges of a node may have the same qualitative direction so that their cyclic order cannot be expressed when using the cardinal direction calculus. This allows investigating whether this indeed leads to results that do not satisfy the conditions of a consistent map hypothesis. The third type of environment used consists of the pseudo-AGVGs (Fig. 7.1(c)) we encountered in Sect. 7.1.1. In contrast to the other two types, these environments are connected to a lesser degree and, hence, contain larger loops and more tree-formed substructures.

To ensure that even searching without pruning is possible in reasonable time, we used rather small problem instances. We randomly generated environments and histories so that the number of junctions contained in the correct model would vary between 4 and 16. We will from now on refer to this parameter as the *size* of the environment. The length of the involved histories varied between $1\times$ and $2\times$ the size of the environment. We then determined (1) how often the solution of the minimal model finding approach was indeed the correct solution and (2) the average error distance. This was done for both variants of the minimal model algorithm, CompEnv and VisOnly, and for the following settings:

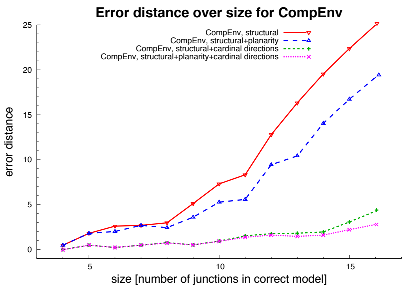
- only structural constraints,
- structural constraints and planarity constraint,
- structural constraints and cardinal direction constraints,
- structural constraints, planarity constraint, and cardinal direction constraints.

The observed frequencies in which the correct solution was found and the error distance averaged over 15,600 runs (20 histories for each of the 13 different sizes for 60 different environments, 20 of each type) are summarized in Table 7.2. For the CompEnv variant, the correct model is only found very rarely in all four settings. However, as the average error distances show, the planarity constraint and in particular the direction constraints significantly improve the solution quality. While the planarity constraint achieves a 26.27% reduction of error distance, the direction information decreases the error distance by 85.70%. Combining both planarity and direction constraints only gives slightly better results than without applying the planarity constraint. Overall, the application of the constraints is highly beneficial but in most cases not sufficient to resolve all ambiguities, typically with regard to junctions that have not been visited.

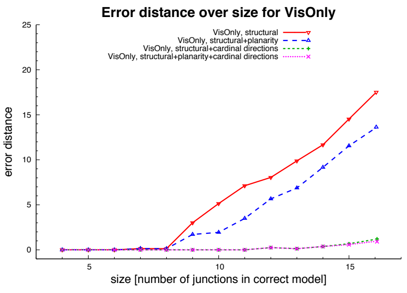
For the VisOnly case in which unvisited junctions are not included in the model, the improvements are even more drastic. While already the basic approach has a much better success rate and average error distance, the application of the cardinal direction information finds the correct model in 98.15% of all trials and has an extremely low average error distance of 0.20, or 0.17 when combined with the planarity constraint. Figure 7.7 shows how the average error distances increase with the size of the correct model throughout the experiment. The high number of correctly identified models, at least in the VisOnly case, is also an indication that the incompleteness of the approach does not often lead to wrong models in practice.

Method		Correct model found	Average error distance
CompEnv	structural only	4.77%	9.86
	structural and planarity	5.97%	7.27
	structural and cardinal dir.	50.62%	1.41
	structural, planarity, and cardinal dir.	50.92%	1.18
VisOnly	structural only	59.00%	5.63
	structural and planarity	64.77%	4.07
	structural and cardinal dir.	97.92%	0.20
	structural, planarity, and cardinal dir.	98.15%	0.17

Table 7.2: Experimental results regarding the solution quality for the different settings



(a)



(b)

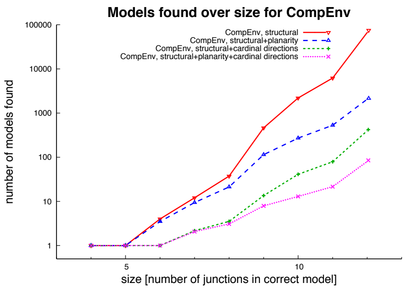
Figure 7.7: Average error distance depending on the size of the correct model for the CompEnv variant (a) and for the VisOnly variant (b)

In a second experiment, we modified the search procedure so that it runs until all hypotheses up to the same size as the correct hypothesis have been found. This allows a better assessment of the ambiguity of the available information in the different settings. We recorded the overall number of hypotheses found and the number of hypotheses which are smaller than the correct hypothesis. The results of this experiment are summarized in Table 7.3. The number of models found depending on the size of the environment is shown Fig. 7.8. Note that we use a logarithmic scale for the y-axes because of the rapid growth, especially in the setting which does not involve additional constraints.

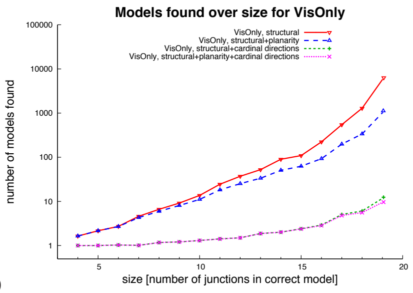
Method		Models found	Smaller models found
CompEnv	structural only	8387.85	8356.87
	structural and planarity	334.18	314.51
	structural and cardinal dir.	59.08	53.50
	structural, planarity, and cardinal dir.	14.28	10.16
VisOnly	structural only	1686.48	1450.08
	structural and planarity	237.18	191.85
	structural and cardinal dir.	3.61	1.77
	structural, planarity, and cardinal dir.	3.09	1.49

Table 7.3: Experimental results regarding ambiguity of information for the different settings

Overall, the planarity assumption reduced the number of models found by 96.02% in the case of CompEnv and by 85.94% for VisOnly. However, this still leaves a rather significant number of alternative models, most of which are actually smaller than the correct solution. The cardinal direction constraints yield a 99.30% reduction



(a)



(b)

Figure 7.8: Average number of route graph models found depending on the environment size for (a) the CompEnv variant and (b) VisOnly (y-axis logarithmically scaled)

for CompEnv and 99.79% for VisOnly. Combining both kinds of constraints results in reductions of 99.83% and 99.82%, respectively. This shows that CompEnv benefits more from also including planarity checking in addition to the cardinal direction constraints than VisOnly. An explanation for this is that merging unvisited junctions allows for many hypotheses which can only be ruled out by not being planar. In the case of VisOnly, many of these hypotheses have already been ruled out earlier by direction constraints that are not satisfied. It is also important to note that while the number of possible models in all settings grows with the size of the environment, many of these will be ruled out when the length of the exploration history is increased.

In summary, as both experiments show, the planarity constraint and in particular the cardinal direction constraints are able to resolve most of the model ambiguities remaining on the structural level, leading to a largely increased solution quality. In the following, we look more closely on how these constraints affect the search space of our minimal model finding approach.

7.3.2 Pruning Efficiency

To investigate the effects of the individual settings on the size of the hypothesis space that has to be searched, we again performed random experiments, running the search until all solutions up to the size of the correct solution had been determined. By doing so, we mask out the effects of varying solution quality which otherwise would distort the comparison. However, this also means that we do not take into account the effects of applying the branch and bound search approach.

To determine the pruning efficiency, we recorded three values for each trial: (1) the number of expanded nodes in the search tree, (2) the number of nodes that are put into the queue, and (3) the maximal queue size occurring during the search. The number of node expansions gives a good indication of the computational costs involved. The ratio of expanded nodes and hypotheses added to the queue then allows us to determine the effects of the different settings on the average branching factor in the search tree.

Finally, the maximal queue size tells us how many hypotheses were tracked simultaneously during the search and, hence, reflects the space consumption.

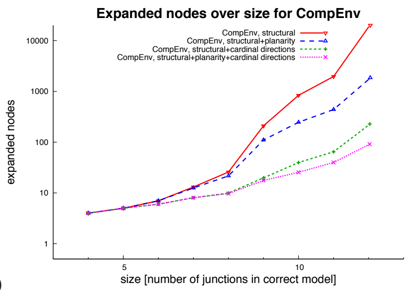
The results of this experiment are summarized in Table 7.4. Figure 7.9 shows how the respective values develop with increasing size of the environment. Logarithmic scale is used for the y-axes in the diagrams for the number of expanded nodes and maximal queue size.

Method		Exp. nodes	Queued nodes	Branch. factor	Max. queue size
CompEnv	structural only	2407.09	10795.72	4.49	833.96
	structural and planarity	284.97	619.93	2.38	86.17
	structural and cardinal dir.	39.84	98.70	2.48	13.58
	structural, planarity, card. dir.	21.85	35.90	1.64	6.10
VisOnly	structural only	790.61	2523.01	3.19	160.88
	structural and planarity	254.25	508.79	2.00	47.87
	structural and cardinal dir.	20.72	24.45	1.18	2.95
	structural, planarity, card. dir.	20.11	23.21	1.15	2.76

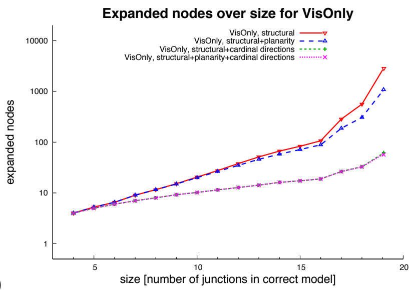
Table 7.4: Statistics concerning the pruning efficiency of the different settings

Although the values are not directly comparable as the experiment was run for different size ranges for CompEnv and VisOnly, we clearly see that the CompEnv variant of the minimal model finding problem is much more complex than the VisOnly variant. All values increase significantly faster with increasing environment size. The planarity constraint leads to an 88.16% decrease in node expansions for CompEnv and 67.84% for VisOnly. The average branching factors have been decreased by 46.99% to 2.38 (CompEnv) and by 37.30% to 2.00 (VisOnly). For the cardinal direction constraints, we see a very high reduction of node expansions of 98.35% for CompEnv and 97.38% for VisOnly. Interestingly, for CompEnv the average branching factor of 2.48 is higher than for the planarity constraint. Looking at Fig. 7.9(c), we see that for larger environment sizes the initial growth of the branching factor for the planarity constraint slows down and becomes more effective than the cardinal direction constraints. By combining both, an extreme reduction in node expansions of 99.99% was achieved for CompEnv, which corresponds to an average branching factor of 1.64. For VisOnly, we get the common picture that the cardinal direction constraints are more effective than for the planarity constraint, leading to a 99.97% reduction (branching factor 1.18), and that the combination only leads to a minor further improvement.

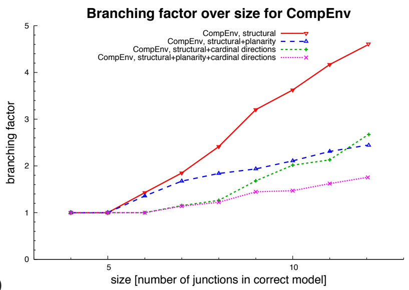
The average maximal queue sizes occurring in the experiment reflect what we already saw for the number of node expansions and the average branching factor. When using planarity and cardinal direction constraints, on average only about six hypotheses had to be tracked simultaneously for CompEnv, and less than three for VisOnly. We conclude that the planarity assumption and the coarse direction information given



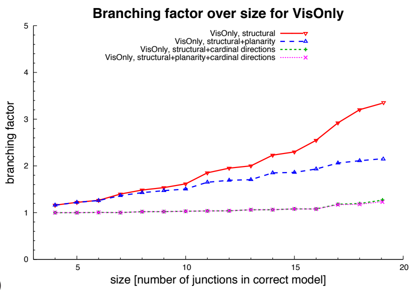
(a)



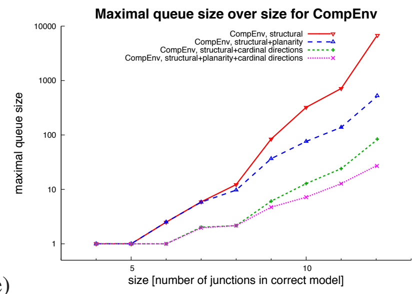
(b)



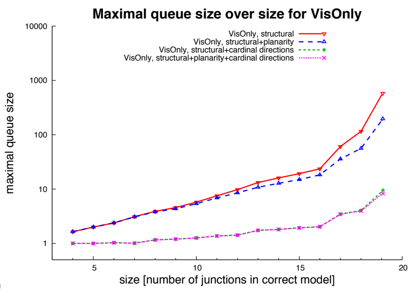
(c)



(d)



(e)



(f)

Figure 7.9: Comparison of the pruning efficiency of the different settings for CompEnv (a, c, e) and VisOnly (b, d, f)

by the qualitative cardinal relations lead to a much increased efficiency of the minimal model finding approach, which would otherwise only be feasible for very small problem instances. In the following, we investigate the performance of another form of direction information, namely relative direction information.

7.3.3 Absolute vs. Relative Direction Information

One of the goals of our analysis was to compare the effects of employing absolute direction information (e.g., relations from the cardinal direction calculus) and relative direction information (e.g., $\mathcal{OPRA}_2$ relations). For this purpose, we repeated the previous experiments for determining the solution quality and pruning efficiency, using the complete branch and bound search approach and searching until the first minimal model has been found. The two settings used (again for both CompEnv and VisOnly variants) were

- structural constraints, planarity constraint, and cardinal direction constraints, and
- structural constraints, planarity constraint, and $\mathcal{OPRA}_2$ constraints.

Besides the previously considered parameters, we distinguished the exact reasons for rejecting a hypothesis. The reasons are (1) direction ordering violation, (2) junction matching violation, and (3) global consistency violation as distinguished in Sect. 6.3.2.1. As also discussed there, direction ordering only plays a role for absolute direction information. In addition, it can only occur at unvisited junctions and, hence, when using the CompEnv variant.

Figure 7.10 shows the diagrams for the relevant assessment parameters. With regard to solution quality, the change from absolute to relative direction information increased the average error distance from 1.64 to 1.82 for CompEnv and from 0.44 to 0.68 for VisOnly. The average number of expanded nodes increased from 49.12 to 82.60 (branching factor from 1.33 to 1.42) for CompEnv and from 12.79 to 13.93 (branching factor from 1.21 to 1.24) for VisOnly. The average maximal queue size increased from 17.17 to 35.72 and from 3.74 to 4.31, respectively.

This decrease in performance is not surprising as relative direction information in general allows for more perceptual aliasing. Taking this into account, the decrease in performance seems to be rather mild and still much lower than for the less constrained settings investigated in the previous experiments. In particular for the VisOnly variant, the approach still performs very well.

When we look at the reasons for rejection that are summarized in Tables 7.5 and 7.6, we see that direction ordering violation does not play an important role at all. Junction matching violations and global consistency violations show about the same rejection ratios for absolute and relative direction information in the case of the CompEnv variant. Global consistency violation occurs much more often than rejection caused by junction matching violations.

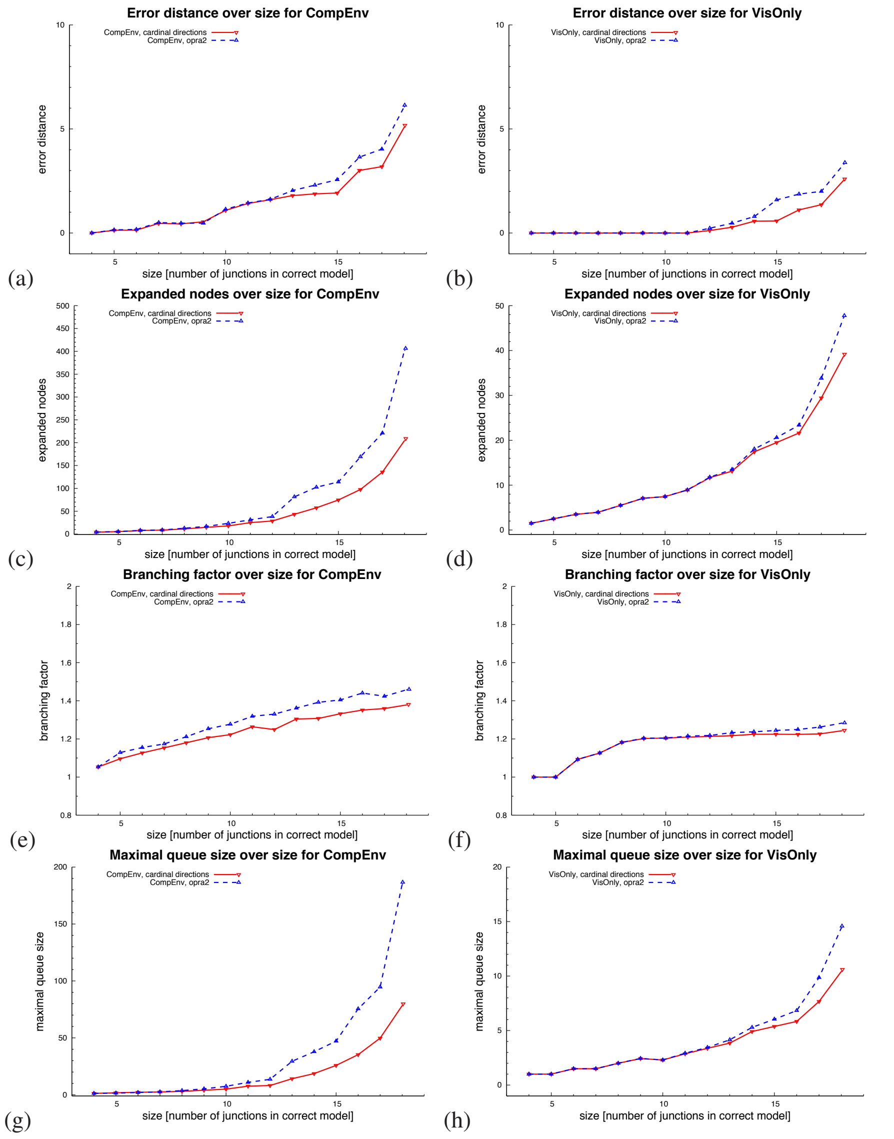


Figure 7.10: Performance comparison of absolute direction information (cardinal direction relations) and relative direction information ($\mathcal{OPRA}_2$ relations)

Method	Reason	% of rejected hypotheses
cardinal direction calculus	direction ordering violation	0.59%
	junction matching violation	23.84%
	global consistency violation	75.57%
$\mathcal{OPRA}_2$ relations	direction ordering violation	—
	junction matching violation	23.09%
	global consistency violation	76.91%

Table 7.5: Distribution of reasons for rejection involving the direction information (CompEnv variant)

Method	Reason	% of rejected hypotheses
cardinal direction calculus	direction ordering violation	—
	junction matching violation	78.73%
	global consistency violation	21.27%
$\mathcal{OPRA}_2$ relations	direction ordering violation	—
	junction matching violation	55.59%
	global consistency violation	44.41%

Table 7.6: Distribution of reasons for rejection involving the direction information (VisOnly variant)

For VisOnly, the picture changes significantly. While for relative direction information global consistency violations still are the reason for about 44.41% of all rejections, only 21.27% are caused by global inconsistencies in the case of absolute direction information. The general increase of junction matching violations clearly results from the fact that for VisOnly complete information about all junctions is available. This allows rejection of many hypotheses early, before global consistency is even tested. The difference between absolute and relative direction information in the case of VisOnly shows that absolute direction information reduces perceptual aliasing to a much higher degree and, hence, increases the predominance of rejections based on junction matching violations.

The two main conclusions from the experiment described here are that relative direction information is inferior to absolute direction information in terms of solution quality and pruning efficiency and that it depends to a larger degree on global consistency checking. Nevertheless, it is still applicable and has the advantage that it often can be obtained more easily, and sometimes also more reliably.

However, while experimenting with larger problem instances, it became clear that the application of the $\mathcal{OPRA}_2$ calculus is much more problematic when we turn from effects on the search space to computational costs in general. This aspect will be further elaborated in the next section.

7.3.4 Overall Computational Costs

When investigating the pruning efficiency in Sect. 7.3.2, we restricted ourselves to small problem instances which allowed us to apply the model finding approach even without planarity and direction constraints. In addition, we focused on the effects of the different settings on the search space. For the complete minimal model finding approaches featuring all kinds of constraints, we further investigated how the approaches perform for larger problem instances. This investigation yielded two main results: First, even when applying all constraints, the CompEnv variant seems to be applicable only to small environments. Second, the computational costs of global consistency checking when employing the relative $\mathcal{OPRA}_2$ calculus quickly becomes excessive, making this approach infeasible for large environments for both CompEnv and VisOnly.

The first point is illustrated in Figs. 7.11(a) and 7.11(b), which show the node expansions for CompEnv and VisOnly for increasing size of the environment. In comparison to the VisOnly variant, the number of node expansions grows very rapidly. Even for cardinal direction information, the computation takes several seconds for environments with only about 20 junctions. Hence, the CompEnv variant seems limited to scenarios with a small number of junctions in which the ability to predict the structure of unvisited parts is worth the increased computational costs in practice. In larger environments, employing spatial reasoning to more local prediction problems (e.g., "could this hallway be a shortcut leading to junction X?") seems to be a more promising approach.

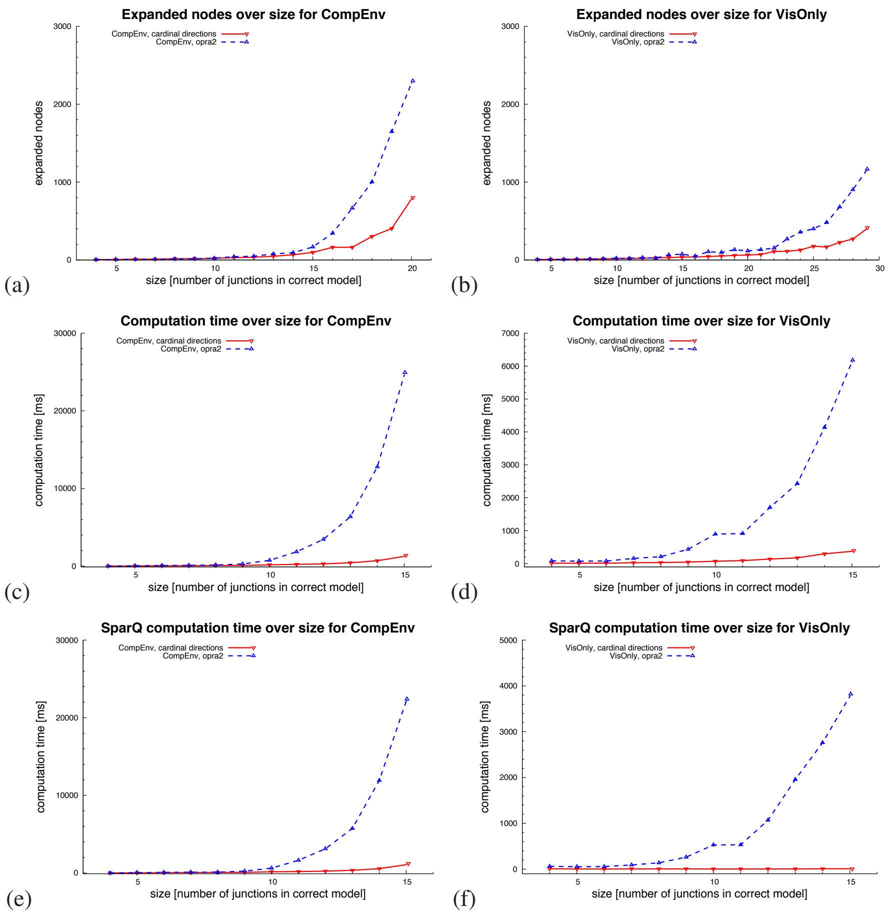


Figure 7.11: Comparison of computational costs of the overall minimal model finding approach for CompEnv and VisOnly

The second result concerning the global consistency checking for $\mathcal{OPRA}_2$ is illustrated in Figs. 7.11(c)–7.11(f). In Figs. 7.11(c) and 7.11(d) we see the overall computation time on a 2.6 GHz Opteron CPU. It becomes immediately apparent that the computation time for $\mathcal{OPRA}_2$ increases much more rapidly compared to that of the cardinal direction calculus than could be explained by the increase in node expansions. The reason for this becomes clear when we look at the time spent on the global consistency check with SparQ, which is depicted in Figs. 7.11(e) and 7.11(f). While the computation times for the cardinal direction calculus only make up a small fraction of the overall computation costs, they rise sharply for $\mathcal{OPRA}_2$. In some cases they make up 90% of the overall computation time.

There are at least two issues that contribute to this explosion in computational costs for the global consistency check. One reason is the large number of base relations in the $\mathcal{OPRA}_2$ calculus. For the 64 base relations, it is not feasible to maintain the complete composition table in memory, which would consist of $2^{64} \times 2^{64}$ entries. Hence, every composition result required by the algebraic closure algorithm is computed by SparQ from the results of composing the individual base relations making up the relations.

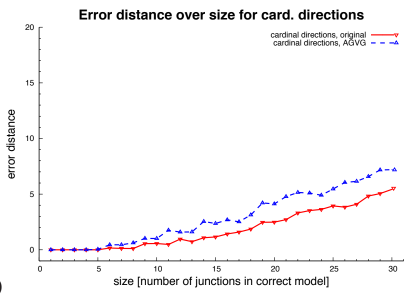
The second problem concerns the size of constraint networks. As we mentioned, the number of variables for the cardinal direction calculus corresponds to the number of RGNs in the hypothesis. For $\mathcal{OPRA}_2$, we get $2\times$ the number of edges as variables. As the time complexity of the algebraic closure algorithm grows cubically with the number of variables, the combination of these two aspects quickly leads to the quick rise in computational costs which we observed in the diagrams.

Overall, the problem of high computational costs for the global consistency check of $\mathcal{OPRA}_2$ is unfortunate because, as we saw, for VisOnly global consistency checking contributes much more to the pruning of the search space than in the case of the cardinal direction calculus. Still, leaving out the global consistency check and relying entirely on junction matching is an option for larger environment as long as no relative calculus with better properties is available. Otherwise, it seems that the VisOnly variant in combination with absolute direction information is the only one that scales sufficiently well to larger environments.

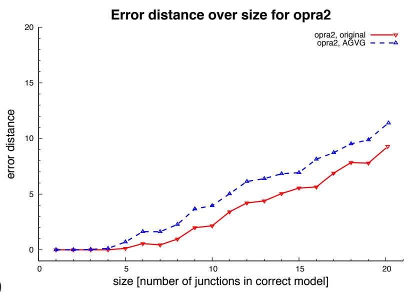
In the next section, we investigate the applicability of this approach to Voronoi graph representations.

7.3.5 Application to Real AGVG Data

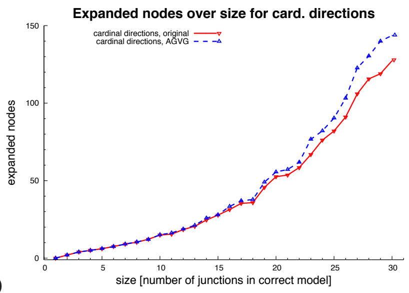
To evaluate the applicability of the minimal route graph model approach to AGVG representations, we employed AGVGs computed from real exploration data following the approach described in Sect. 7.1.2. We then generated random paths through the AGVGs and simulated the adaptations described in Sect. 6.4. We assumed that a neighbor of the current Voronoi node is simultaneously perceived if both are less than 2.5 m apart. If this is not the case, no direction relation is derived for this connection. Over the complete experiment, the neighbor was not perceived for 37% of all edges.



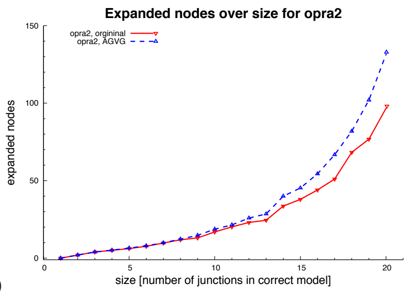
(a)



(b)



(c)



(d)

Figure 7.12: Comparison of the original minimal model approach with its adaptation for AGVGs for cardinal direction constraints and $\mathcal{OPRA}_2$ constraints

In the experiment, we restricted ourselves to the VisOnly variant and compared the version with simulated adaptations with the original version without adaptations for both cardinal direction constraints and $\mathcal{OPRA}_2$ constraints. The results of the experiment are summarized in Fig. 7.12. It shows that the required adaptations only lead to a minor degradation of performance with regard to solution quality as given by the average error distance and with regard to pruning efficiency as given by the average number of node expansions. Overall, the average error distance increased from 0.79 to 1.48 for the cardinal direction calculus and from 2.11 to 3.21 for $\mathcal{OPRA}_2$. The average number of node expansions increased from 54.87 to 59.25 and from 110.74 to 142.29, respectively.

These results allow the conclusion that the effects of perceiving entire connections and additional information about junctions (node signatures) sufficiently compensate for the overall weaker direction information. Another factor certainly is that, as we saw, the VisOnly variant relies to a lesser degree on rejections caused by global consistency violations. The results of actually applying the approach to Voronoi data from a real exploration experiment in contrast to the simulated exploration used here will be described in the next section.

7.4 A Complete Multi-hypothesis Mapping System

In our last experiment, we combine the techniques developed in Chaps. 4–6 of this book into a single multi-hypothesis mapping system. We use the extraction and data association methods to extract a history of passed relevant Voronoi nodes from local metric maps. The minimal model approach is then applied to generate a hypothesis about the topological structure of the environment. Based on such a hypothesis, a complete AGVG representation can be constructed and finally turned into an HAGVG. Figure 7.13 illustrates the interplay between the different components of the mapping system. We briefly describe the components in more detail, before we turn to the experimental evaluation.

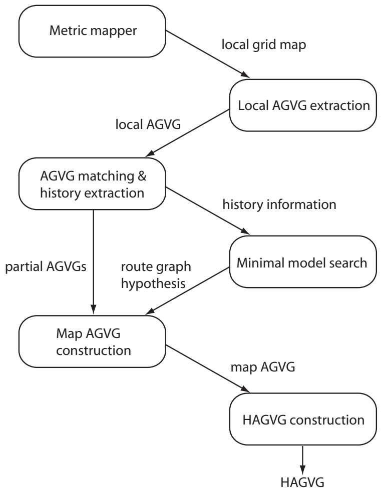


Figure 7.13: Overall multi-hypothesis mapping system integrating the techniques developed in this book

7.4.1 Local Metric Mapping and Local AGVG Computation

A single 180° scan typically does not provide sufficient information to compute a local AGVG. Therefore, we use a probabilistic grid mapping approach to compute local metric maps representing the robot's immediate surroundings. We then extract the local AGVG from these maps. The local map for a particular position of the robot is constructed from the laser scans taken along the last 5 m of travel distance before the robot reached that position and the next 5 m of travel distance. As a result, the history extraction slightly lags behind the actual motion of the robot but the quality of the extracted local AGVGs is increased.

The extraction of the local AGVGs is performed in the same way as that described in Sect. 7.1.2. It includes the removal of uncertain parts from the graph due to unknown areas in the local grid map. As a consequence, the resulting AGVG is typically not connected and we further reduce it to the component that describes the immediate neighborhood of the robot's current position. Finally, the relevance values are computed and the simplification algorithm is used with a very low threshold value to prune the most irrelevant parts of the local AGVG.

7.4.2 Data Association for Node Tracking and History Generation

The data association module uses the AGVG matching algorithm developed in Chap. 5 to identify correspondences between consecutive local AGVGs. Based on the result of the data association and the relevance values of the Voronoi nodes, new history information is generated. The result is a sequence of passed Voronoi nodes and edges. Whenever a Voronoi node with a relevance above a given threshold is passed, a new pair of traversal action and junction observation is handed over to the minimal model computation module to be added to the history.

Similarly to the previous experiment, two cases are distinguished for each direction constraint (cf. Sect. 6.4): If both relevant nodes are part of the same local map, the direction will be treated as a constraint on the positions of these two nodes, which can be employed for global consistency checking. If this is not the case, only the direction relations for the leaving edges are stored, and these will not be used for global consistency checking.

In addition, the merging algorithm of Sect. 5.4 is used to construct local route graphs for the edges traversed between two Voronoi nodes that are added to the history. These partial AGVGs can then be used to construct a complete route graph from the result of the global minimal model computation.

7.4.3 Global Mapping and Post-processing

The output of the tracking module is an extension to the exploration history, consisting of a new Voronoi curve traversal and Voronoi node observation. This new information is used by the minimal route graph finding module to update the search tree and compute a new minimal hypothesis which becomes the new current map hypothesis. For the minimal model computation, we use the VisOnly variant of the algorithm.

When a new minimal model has been computed, a complete route graph can be constructed by replacing the edges in the hypothesis with the partial AGVGs computed by merging local AGVGs. Finally, the techniques from Chap. 4 can be used to generate a complete HAGVG representation from this global AGVG.

7.4.4 Experiments

In the experiments we performed with the described mapping system, we tested its ability to create correct map hypothesis for a real exploration data set. The environment used is shown in Fig. 7.14. The figure shows a grid map of the environment together with the trajectory of the robot. During the exploration, the robot first traversed the main loop of the environment twice in clockwise direction. On the third round, it also entered all the rooms along the main loop. Then, it traversed the main loop in counterclockwise direction. Finally, it moved along the smaller loop lying inside the main loop and back again.

Placed around the central grid map, Fig. 7.14 shows a selection of local grid maps generated during the exploration together with the extracted local AGVGs. Voronoi nodes relevant enough to be included into the history are marked by the black circles around them. The local maps are mainly from the first traversals of the main loop and the inner loop.

The history generated for this exploration run consists of 150 traversal actions and 151 junction observations. Using the cardinal direction calculus, the overall minimal route graph model computed was indeed the correct model of the environment. The picture at the top of Fig. 7.15 depicts this route graph model. In the bottom figure, we see the complete AGVG constructed by replacing the edges with their respective partial AGVGs. The picture in the middle shows a coarser layer derived via simplification. Together with the AGVG in the bottom figure, it could form an HAGVG representation of the environment.

We also repeated the experiment employing the $\mathcal{OPRA}_2$ calculus instead of the cardinal direction calculus. The resulting minimal model had an error distance of 3 resulting from wrongly merging RGNs in the room at the bottom that could be entered via two different doors. However, while the computation took 16 seconds for the cardinal direction calculus, it took over ten hours for $\mathcal{OPRA}_2$.

Figure 7.16 shows how error distance of the current minimal model, number of node expansions, and queue size develop over the 150 exploration steps for both spatial calculi. The diagram for the error distance shows that the variant using the cardinal direction calculus immediately settles for the correct hypothesis when the first loop traversal is completed in step 23, while this takes almost the entire second loop for $\mathcal{OPRA}_2$. We later see another increase in error distance caused by entering the new rooms. We also see that the number of node expansions required and the number of tracked alternative hypotheses is significantly higher for $\mathcal{OPRA}_2$. However, the main reason for the hugely increased computation time is again the time required for the global consistency checking.

7.4.5 Discussion

The complete mapping system described in this section mainly serves the purpose of a demonstrator for the techniques developed in this book. The experiments have shown that at least the VisOnly variant of the minimal model finding approach combined with the cardinal direction constraints is applicable to construct AGVG representations from real range data. Relevance assessment, simplification, and AGVG matching were employed to extract the required history information, while the developed hierarchization

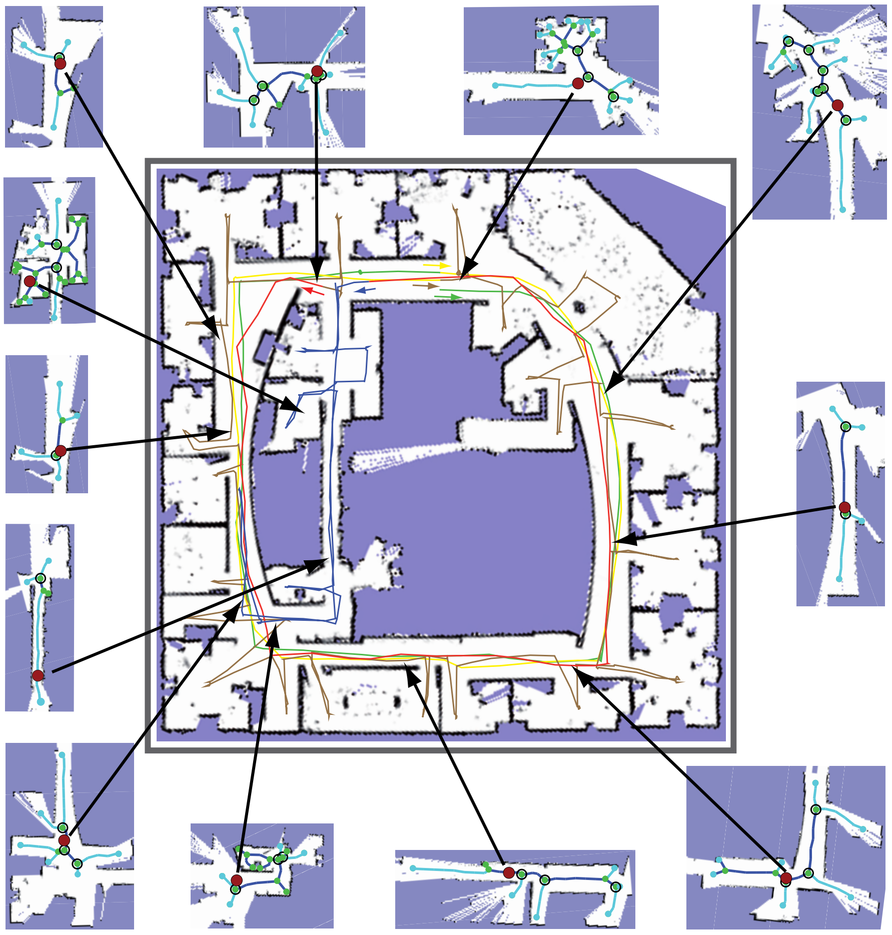


Figure 7.14: Grid map of the environment used to test the overall mapping system with selected local maps (data set has been recorded at the Intel Research Lab, Seattle, and is available at `http://radish.sourceforge.net/`, courtesy of D. Hähnel)

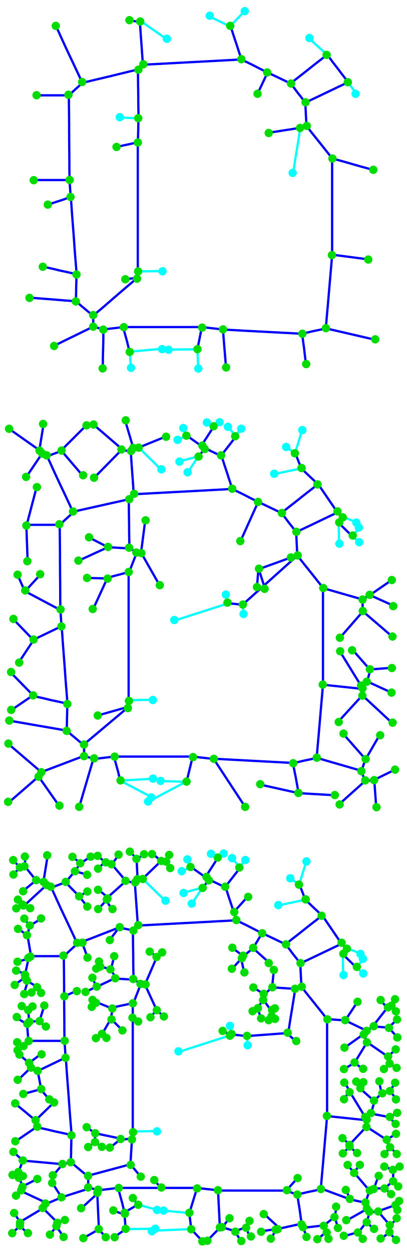
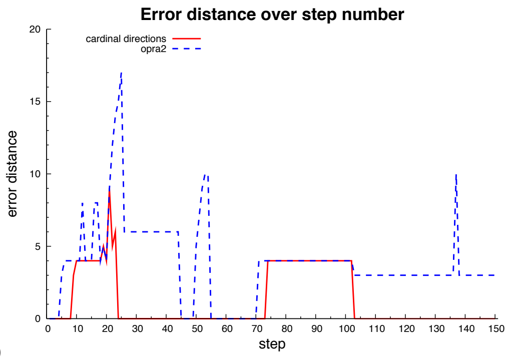
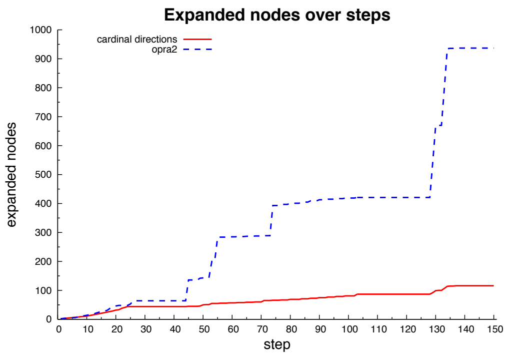


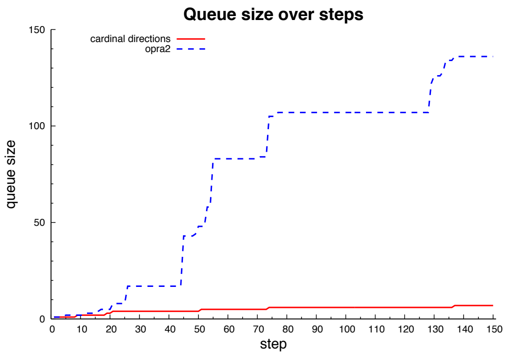
Figure 7.15: Representations computed by the overall mapping system: a depiction of the computed minimal route graph model (top), the constructed complete AGVG (bottom), and a coarser version generated by simplification (middle)



(a)



(b)



(c)

Figure 7.16: Development of error distance, node expansions, and queue size during the exploration experiment

approach allowed us to derive a complete HAGVG representation.

In contrast to the overlay approach used in Sect. 7.1.2, the AGVG is constructed directly, without the need for a global coordinate-based representation. When compared with the feature-based FastSLAM approach employed in Sect. 7.2, one advantage is that keeping track of all sound hypotheses prevents the particle depletion problem. Moreover, the contained relative metric information can be globally inconsistent without diminishing the applicability of the representation for navigation.

One disadvantage of the approach described here is that it relies on the correctness of the history information. When we experimented with different data sets that, for instance, contained dynamics caused by people moving in the vicinity of the robot, these often lead to deviations in the local AGVGs and the generated history information that would cause the minimal model computation to fail. Besides improving the methods for extracting the history information, an alternative approach would be to extend the approach so that it can cope with ambiguity in the history information.

Overall, there are many ways in which the fundamental techniques described in this book can be integrated and combined with existing mapping approaches in order to develop robust mapping systems for Voronoi-based spatial representation. We will point out some promising directions for future research in the next and final chapter of this text.

Chapter 8

Conclusions and Outlook

We conclude this book by summarizing and discussing the results achieved and by providing an outlook on promising directions of future research arising from the work described in the previous chapters.

8.1 Summary and Conclusions

We started this work with the observation that most work on robot mapping is concentrated on developing powerful techniques for uncertainty handling for a small number of elementary low-level representation approaches: grid-based representations, geometric representations, and landmark-based representations. However, as we argued, in order to realize or explain high-level spatial cognitive abilities, more emphasis needs to be put on representational matters. Robot mapping needs to be studied in a general setting involving the perspectives of both spatial representation and uncertainty handling. In addition, it needs to explicitly take into account the operations that employ the environmental model to successfully solve spatial tasks.

Although more abstract forms of representation and more complex forms of organization have been proposed, we observed that only a small number of attempts have been made to achieve a high level of robustness in the presence of uncertainty for these approaches. This led to the main thesis of this work, namely that the combination of uncertainty handling methods developed for the traditional representations with abstract high-level representations is a very promising direction of research. It has the potential to lead to mobile robots with a much higher level of spatial capabilities and competences and, hence, to open up a new range of applications.

In the following, we directed our attention at one particular class of representations, namely route graph representations, derived as abstractions of the generalized Voronoi diagram of the environment. With the annotated generalized Voronoi graph (AGVG) containing a particular set of node and edge attributes, we introduced a concrete representation approach and proposed a hierarchical organization of this representation, the HAGVG. As we discussed, the advantages of this hierarchical AGVG representation are compactness and good scalability, well-suitedness for global route planning and route-based communication, the ability to consistently represent environments without requiring metrical consistency, and the ability to serve as a core structure for anchoring any kind of additional information in the form of further annotations. Furthermore, the approach directly facilitates systematic exploration of an environment in order to construct a complete environmental model. Parts of the proposed annotations are intended to help distinguish different places and, hence, improve localization without the need for a complete geometric model.

We also identified several limitations of the approach. The area of applicability is restricted as the approach presupposes that the environment shows a clearly pronounced route structure, e.g., indoor environments or path networks. Moreover, instability of the underlying generalized Voronoi diagram when the input data is noisy poses a challenge for the development of reliable construction and maintenance methods.

The focus of this book then has been on developing fundamental techniques required for the construction and maintenance of the HAGVG representation, mainly with regard to a mobile robot equipped with a laser scanner or similar range sensors which yield a 2D geometric description of the robot's surroundings. This work has been conducted on three levels.

First, we were concerned with the reliable autonomous extraction of AGVGs and hierarchical construction from a data structure point of view. The correct identification of corresponding features was taken for granted. Second, we studied the problem of robust data association for local matching and tracking of Voronoi nodes. Third, we turned to the global mapping problem, which we formulated as the problem of finding a minimal route graph model under spatial constraints. We studied this problem with a focus on effects of reliable qualitative directional information in combination with the planarity constraint.

In the following, we briefly summarize the results achieved for each of these three areas and with regard to the overall mapping approaches discussed in the evaluation chapter.

8.1.1 Extraction and HAGVG Construction

Concerning the extraction of AGVGs and the construction of a hierarchical representation, we proposed measures for the relevance of a Voronoi node and the regions accessible via its edges. The relevance values can be computed without requiring a complete geometric description of the environment, simply from the information stored in an AGVG. We developed efficient computation and incremental update algorithms. Based on the measures, we developed a simplification algorithm for AGVGs. This algorithm forms the core of our hierarchy generation scheme used to autonomously build up HAGVGs which represent the environment at different levels of granularity.

We also demonstrated that the concept of relevance coincides well with the stability of Voronoi nodes under noisy conditions. As a result, the approach can also be employed to deal with problems of the graph construction process caused by the instability of the underlying Voronoi diagram. The described approach worked very reliably and served as a basis for the other techniques developed in this book.

8.1.2 Data Association and Matching

To achieve reliable data association, we extended constraint-based branch and bound approaches by incorporating structural constraints arising from the graph topology and the combinatorial embedding of AGVGs. We developed a dynamic programming approach based on edit distance to match AGVGs. The relevance values of nodes were used to determine the costs of the different edit operations. We incorporated several types of constraints in order to reduce the search space: (1) unary constraints derived from node similarity and, if available, position estimates, (2) binary distance constraints, and (3) ternary angle constraints. In order to achieve a high level of efficiency, we restricted the approach to tree-formed AGVGs (which we enforced if needed) and exploited the fact that the resulting trees are ordered as specified by the planar embedding of the AGVGs.

When we evaluated and compared our approach with a standard individual compatibility nearest neighbor approach, it proved to yield very low error rates with only a mild increase in computational costs. The approach worked very well to locally track Voronoi features, even when no odometry information is available. However, the approach in its current form does not support loop closing.

8.1.3 Minimal Route Graph Model Finding

Bringing together work from topological mapping and qualitative spatial reasoning, we finally formulated global mapping of a graph-like environment as the problem of finding a minimal route graph model that explains a sequence of junction observations and hallway traversal actions. A solution to the problem has been developed consisting of a best-first branch and bound search through the interpretation tree of possible associations of junctions.

We integrated planarity and spatial consistency checking into the search algorithm in order to prune the search space. For spatial consistency checking, we employed constraint-based reasoning techniques based on qualitative spatial calculi. In our evaluation we focused on direction information and compared the effects of absolute and relative direction information. We also compared two versions of the algorithm, one that computes complete environment models (CompEnv) and one that only models the junctions that have actually been observed (VisOnly).

The results of the evaluation using randomly generated graph worlds as well as data extracted from a few selected real environments showed that the combination of the planarity constraint and in particular the direction constraints significantly reduced the search space and ambiguity of the history. This resulted in a very high rate of correctly mapped environments once the exploration paths covered a large enough fraction of the environments. One general advantage of the approach is that as soon as the correct model has been determined, the time needed for incorporating new observations becomes constant.

When combining the VisOnly variant with absolute direction information, the approach was efficient enough for all the environments we used in the experiments. However, employing relative direction information in the form of relations from the $\mathcal{OPRA}_2$ calculus quickly led to runtime problems caused by the global consistency test, which does not scale sufficiently well to larger environments.

The CompEnv variant, on the other hand, allows making predictions about connections and junctions that have not been observed directly. This can, for instance, help to find shortcuts. However, the experiments showed that this approach is limited to smaller environments because of the increased complexity of the problem. Hence, it can only be employed beneficially in suitable environments (e.g., a network of hallways) and when observations are provided on a very high level of abstraction (e.g., the robot can detect junctions, rooms, etc.).

In order to apply the theoretical minimal model finding framework to our Voronoi-based representations, a set of modifications and relaxations was required. However, it turned out that these adaptations only lead to a small decrease in overall performance.

8.1.4 Complete Mapping Approaches

The evaluation of the techniques developed in this work also involved three different examples of combining the (H)AGVG representation with uncertainty handling methods which demonstrated that this approach is indeed fruitful. The overlay approach consisting of a grid map representation generated via a FastSLAM approach and an HAGVG demonstrated the robustness of the route graph extraction and abstraction methods. The approach, however, does not scale well to very large environments because of the high space consumption and the increased computation time needed to extract the underlying Voronoi graph.

By employing a feature-based FastSLAM approach that uses the developed extraction and data association techniques and builds up the route graph representation directly, we were able to avoid these problems, but our approach does not facilitate loop closing. In addition, the approach is subject to the particle depletion problem.

In the final mapping system, we used the extraction methods and the data association techniques in order to derive a history of observations and actions at a high level of abstraction. Applying the minimal route graph model finding approach then allowed us to determine the correct route graph hypothesis. This approach has the advantage that no absolute coordinates need to be assigned to the Voronoi nodes and the relative metric information contained in the route graph may be globally inconsistent without diminishing its applicability. However, like many other approaches to robot mapping, this approach still depends on the correctness of the extracted history and, hence, the extraction and data association processes. Unfortunately, errors at these levels cannot be ruled out entirely and extensions of the approach are required, for instance involving multiple alternative histories.

Nevertheless, this overall mapping system demonstrates that with the developed techniques we have achieved a significant progress towards the robust learning of Voronoi-based route graph representations for mobile robots. Part of this is due to the fact that fine metric relations are only considered at a local level, while at the top level we were more concerned with abstract relations of connectivity and coarse direction relations. This made the discrete multiple-hypothesis approach to the global mapping problem feasible. Techniques such as the data association based on ordered tree matching also could easily be adapted for other kinds of route graph representations used for robot mapping. The minimal route graph model finding framework is very general with regard to the actual representation approach and application area, and can be employed whenever local observations of decision points should be integrated into a global graph model. To give an example, an outdoor robot able to reliably detect ways and paths in a park could very well use the minimal route graph model framework for global mapping without employing a Voronoi-based approach.

8.2 Outlook

We close this text by discussing open questions and promising continuations arising from the work presented here. We start at the technical level and then gradually broaden our view coming back to the main theses of the work, resulting research directions, and general challenges for the field of mobile robot research.

8.2.1 Extensions of the Work Described in Chaps. 3–6

In Chap. 3, we argued that it is one advantage of the route graph representation that it makes it easy to anchor additional information in the form of further attributes to its nodes and edges. From a representational point of view, one such kind of information that is very relevant for navigation and route-based communication in humans is landmark information, e.g., information about salient objects encountered along a route segment or at a decision point (Richter, 2007). In addition, distinct landmarks have the potential to greatly improve localization and further reduce ambiguities with regard to different map hypotheses. Hence, it would be desirable to include and exploit landmark information in the Voronoi-based mapping approach. However, to make the step from simple geometric features to distinct landmarks that can support human-robot communication, largely increased sensor and object recognition capabilities are required on the side of the robot.

With regard to the relevance measure for Voronoi nodes and the automatic hierarchy generation approach, one thing that could be improved is the treatment of cycles. As we explained, we treat leaving edges that are part of cycles as maximally relevant, which is reasonable when we consider real loops within the environmental structure.

For small cycles caused by clutter or small obstacles, this approach can yield non-intuitive results. While it should be possible to adapt the relevance measures to distinguish these two situations, we believe that aiming at a way to compute the Voronoi graph from a local description which excludes all kinds of clutter and objects (e.g., only with respect to the walls in an indoor environment) will be more advantageous. We will come back to this point again below.

In the work on minimal route graph model finding under hard spatial constraints in Chap. 6, we focused on directional information, restricting the relative positions of the junctions in a route graph model. The reason for doing so is that directions of leaving hallways can often be perceived very reliably, which makes them a good candidate for pruning. Still, other kinds of spatial information can be available as well, and while direction information here is treated as information about junctions, information regarding the traversed hallways can also be used.

The first thing that comes to mind here is information about the length of a traversed hallway, which would restrict the distance between the connected junctions. While extending the theoretical framework for this kind of information is straightforward, the lack of qualitative distance calculi or real positional calculi comprising distance and direction information already shows that distance information is not such a good candidate for use as a hard constraint for pruning.

As we also discussed in Chap. 6, the individual checking of planarity and consistency of the direction information means that the entire procedure has to be classified as an incomplete method that may not discover and discard all inconsistent hypotheses. The facts that the cardinal direction calculus cannot express the cyclic edge orderings of the combinatorial embedded AGVGs and that the consistency check based on algebraic closure is incomplete in the case of the $\mathcal{OPRA}_2$ calculus can cause additional undiscovered inconsistencies. Although we found no indication that this is a problem in practice during the experimental evaluation, a complete approach would naturally be desirable. However, it seems doubtful that a complete approach can be achieved without increasing the computational costs in a way that makes the application within the search algorithm infeasible.

One thing we did not investigate closer in this work is the effects of applying finer-grained qualitative information. For relative direction information, this would directly be possible by switching to a higher granularity parameter within the $\mathcal{OPRA}_m$ calculi family. For absolute direction information, one option would be to employ a variant of the Star calculus (Renz & Mitra, 2004). In general, if the sensor system can reliably provide information at this finer granularity, a noticeable increase in pruning efficiency and solution quality can be expected.

8.2.2 Combining Voronoi Graphs and Uncertainty Handling

The three different ways of combining Voronoi-based route graph representations and uncertainty handling methods considered in Chap. 7 of this book have to be seen more as case studies in order to evaluate the developed techniques rather than as perfect solutions. For instance, the final overall mapping system discussed in Sect. 7.4, as mentioned, still has the shortcoming that it relies on a correct sequence of Voronoi node observations and Voronoi curve traversal actions, which in turn is reliant on correct tracking of these features while the robot moves through the environment. Hence, robustness could be further increased by considering alternative histories whenever the data association becomes ambiguous. Alternatively, uncertainty arising from extraction and data association might be expressed in the history and incorporated into the search algorithm.

Another promising approach would be to combine the constraint-based framework with probabilistic methods to estimate the likelihood of the individual hypotheses by incorporating information that is currently not used on the global level, e.g., global position estimates. This likelihood assignments could be used in multiple ways: to distinguish between multiple existing minimal hypotheses, to reject hypotheses for which the likelihood falls below a certain threshold, or to limit the number of hypotheses tracked by discarding the least likely ones when the number of hypotheses starts to exceed the limit.

In principle, it would also be possible to drop the minimal model idea and turn the algorithm into one that searches for the most likely route graph hypothesis as in the lazy data association approach described by Hähnel et al. (2003), while still enforcing the hard constraints. The crucial problem of these approaches is to formulate an adequate probabilistic model of the processes involved in the Voronoi graph computation. In how far rough approximations, e.g., based on similarity, can be employed here remains to be investigated.

8.2.3 Challenges for Voronoi-Based Representation Approaches

One downside of Voronoi-based representations we have already mentioned multiple times is their restricted applicability as the environment needs to suit a route-based approach. For robots equipped with laser range finders, problems can arise even in indoor environments, either when the environment contains very large open areas or when it is extremely cluttered.

Large open areas can cause problems because when the limited range of the laser scanner prevents the detection of even the nearest obstacle boundaries, no local GVD can be computed. With the range of today's laser range finders and the computation of the GVD from a local metric map this problem becomes less severe. Alternatively, Beeson et al. (2005) recently proposed an extension of the GVG called the extended Voronoi graph in which open areas are represented by edges enclosing them at a certain distance from the boundaries. Incorporating this approach into our system seems in principle possible, but would require us to rework some of the techniques.

Highly cluttered environments, on the other hand, can cause a lot of difficulties because they can lead to a high density of nodes and mini-cycles (e.g., when a robot only sees the legs of chairs and tables in a cafeteria). Again, we suggest that the most adequate approach is to deal with these kinds of problems on the extraction level by generating an intermediate representation only containing the most stable obstacles and computing the underlying GVD from there. However, this requires scene understanding capabilities which are beyond current technology.

In recent years, a lot of research has been undertaken to extend 2D mapping approaches to full 3D mapping. From the perspective of a route graph representation, this step only would make sense if the agent is also able to freely move in 3D space. As this is not the case for the wheeled mobile robots we have been considering here, it seems more reasonable to exploit the 3D sensor information in order to improve the computation of a normal 2D embedded route graph. For instance, for the extraction of the underlying GVD, a local metric 3D representation would be advantageous. It would for instance remove problems caused by situations in which the robot sometimes does and sometimes does not see certain obstacles because of varying height or inclination of the laser scanner. In addition, local information to improve the localization could be extracted from the 3D map as well and then annotated to the 2D route graph. The resulting system would consolidate local 3D geometric information into local 2D information and then further into local route graphs which are integrated in order to produce the global route graph representation. If indeed a 3D embedded equivalent of the (H)AGVG is needed, a good starting point would be an extension of the GVG to higher dimensions proposed by Choset which ensures that the resulting structure is still a connected network of one-dimensional curves (Choset & Burdick, 2000; Choset et al., 2000).

One major problem of the Voronoi-based approach and similar route graph approaches that we have not touched on in this work is dealing with dynamics. As the Voronoi-based approach attempts to determine the topology of free space, it is particularly sensitive to changes that affect this topology. For instance, an object temporarily placed in the middle of a hallway will induce a change from a single Voronoi curve to two new ones passing this object to the left and right. Such changes can significantly complicate the matching and may require a modification of the graph structure of already mapped areas when they are detected.

Overall, different kinds of dynamics will need to be tackled at different levels, and most of them are more adequately dealt with by working around the sensor and recognition limitations of current robots instead of within the context of mapping. Changes caused by objects moving around the robot, for instance, may be filtered from the sensor data or their effects could be completely masked out by following our long-term suggestion of basing the Voronoi computation only on stable objects. The latter approach could also deal with changes caused by the replacement of movable objects. Adequately dealing with topological changes as caused, for instance, by doors will only be possible by incorporating improved object recognition capabilities and background knowledge into the mapping process. From the mapping point of view, the hardest problem is probably caused by long-term changes to the main structure of an environment, e.g., by moving the inner walls within a configurable building. However, these kinds of changes also tend to confuse humans and may well just trigger the learning of a new model from scratch.

Finally, while in this work we have only considered mapping as a passive procedure that processes the history of observations and actions, another interesting direction of research would be to use the state of the multi-hypothesis search tree used in the minimal model approach in order to actively guide the exploration behavior of the robot. For instance, the robot could be deliberately moved to a place that allows deciding between two possible hypotheses. As a result, the number of hypotheses that need to be tracked simultaneously could be reduced significantly.

8.2.4 Challenges for Qualitative Spatial Reasoning

Aside from the robot mapping problem, this book has also resulted in some interesting observations with regard to abstract spatial reasoning. The results with regard to spatial consistency checking based on qualitative direction information can well be seen as a challenge for future research on qualitative spatial reasoning. As we have argued, none of the currently existing directional spatial calculi ideally fit the demands arising from our theoretical problem.

The cardinal direction calculus, for instance, while having good computational properties cannot express cyclic ordering information. For relative directional calculi like the $\mathcal{OPRA}_2$ calculus, the standard constraint reasoning techniques like algebraic closure are typically incomplete. In addition, even the application of the standard algebraic closure algorithm quickly became infeasible for $\mathcal{OPRA}_2$.

Therefore, either new qualitative calculi or new reasoning techniques for consistency checking are needed. Alternatively, if there is a fundamental conflict between the different demands, the involved trade-offs have to be investigated further. Finally, calculi integrating different aspects of space would ultimately be required if the theoretical problem is to be extended by other qualitative spatial information in addition to the directional information.

8.2.5 The Future: Towards Spatially Competent Mobile Robots

At the beginning of this text, we argued that in order to advance towards more capable robots able to operate in large-scale heterogeneous environments, to communicate about space with other robots or with humans, and to interpret and utilize externalized spatial information in the form of sketches or real maps, a turning away from the simple homogeneous and sensor-near representation approaches is required. The hierarchical route graph representations considered in this work naturally can only be seen as a tiny step in this direction, providing a suitable conceptualization for indoor environments, street networks, etc. And while realization of higher cognitive abilities based on this kind of representation was outside the scope of this text, work in this direction has already started, e.g., with regard to carrying out verbal route instructions (Shi et al., 2007). However, to achieve the overall goal, it seems likely that such an approach will only be one building block in the pool of representation schemes or conceptualizations, from which a robot can choose the most suitable one based on the properties of the considered part of the environment or the task at hand.

From the previous discussions concerning the general improvement of robustness and dealing with dynamics, it should have become clear that we think that progress with regard to more abstract spatial representations and complex forms of organization depends to a large degree on progress achieved in other areas of robotics and AI research. The key components in our opinion are improved object recognition and scene analysis capabilities and the inclusion of background knowledge, spatial and also non-spatial (e.g., regarding stability, temporal aspects, or typical behavior), into the mapping process. As we pointed out, the techniques developed in this work can directly benefit from advancement in these fields.

In the long term, such a development could mean that robot mapping will turn more and more into an all-embracing AI-complete problem. At any rate, there can be little doubt that robot mapping will remain a challenging and exciting field of research for years to come.

Appendix A

Mapping as Probabilistic State Estimation

In the following, we provide a brief summary of the fundamental mapping techniques developed in the area of probabilistic robotics. For more detailed overviews, we refer you to Thrun et al. (2005), Thrun (2002), and Frese (2006a).

In probabilistic robotics, robot mapping is treated as a state estimation problem: The robot's environment is a dynamic system with a state that is only partially observable by the robot through its perceptions and is affected by the indeterministic results of its actions. Thus, the robot can only estimate the true state of the environment from its history of observations and actions (provided by odometry measurements or motion controls).

The state of the environment is described as a set of random variables that are often continuous and comprise variables for all modeled properties of the external world as well as for the robot's own state. For instance, if the environment is modeled by the positions of five important point landmarks in a two-dimensional coordinate system, this would result in 5×2 continuous random variables (two variables for each landmark, representing its x and y coordinates) plus typically three continuous variables describing the robot's pose. The complete state is then referred to by a single n-dimensional random vector x with one dimension for each random variable. In a dynamic system, state variables may change over time. Typically, discrete time steps are assumed and indices are used to indicate the time step considered, e.g., x_t for the state at time step t, starting with x_0.

The state estimation problem now is the task of computing the conditional joint probability density function $p(x_t|a_{1:t}, o_{1:t})$ over all state variables from a given starting distribution $p(x_0)$ and the sets of observations $o_{1:t} = o_1, ..., o_t$ and actions $a_{1:t} = a_1, ..., a_t$. a_i here is the action that leads from x_{i-1} to x_i, o_i is the observation corresponding to x_i, and we assume that the robot starts in x_0 by performing action a_1. The conditional probability distribution represents the robot's belief about the state of the world.

Generally, it is assumed that a successor state x_t only depends on the previous state x_{t-1} and (although indeterministically) on the action a_t chosen at time point $t-1$. This means the entire history of states passed before x_{t-1} plays no role once x_{t-1} is known. This is known as the *Markov property*. The dynamic system then becomes a (partially observable) Markov chain and $p(x_t|a_{1:t}, o_{1:t}) = p(x_t|x_{t-1}, a_t)$.

However, the Markov property is only an approximation when it comes to robot applications in the real world. One reason for this is that the state model is always only an approximate model of the current state of the world, focusing on the most relevant aspects for the application at hand. Hence, some parameters affecting the result of actions are typically not modeled. For instance, if the state vector does not contain the current status of the robot's battery, the result of the robot's actions does not depend on the current state alone but also on the sequence of actions that lead to it (if the series of actions has completely exhausted the battery, the robot is unlikely to move at all).

A.1 The Recursive Bayes Filter

In probabilistic robotic approaches it is assumed that development of the state of the dynamic system is determined by the laws of probability theory. The fundamental formula underlying these approaches is the *recursive Bayes filter*, which is derived by applying Bayes law to express the *posterior probability distribution* $p(x_t|o_{1:t}, a_{1:t})$:

$$\begin{aligned} p(x_t|o_{1:t}, a_{1:t}) &= \frac{p(o_t|x_t, o_{1:t-1}, a_{1:t})\; p(x_t|o_{1:t-1}, a_{1:t})}{p(o_t|o_{1:t-1}, a_{1:t})} \text{ (Bayes law)} && \text{(A.1)} \\ &= \eta\; p(o_t|x_t, o_{1:t-1}, a_{1:t})\; p(x_t|o_{1:t-1}, a_{1:t}) && \text{(A.2)} \end{aligned}$$

where η is the *normalization factor* that ensures that the probabilities integrate to one. η is independent of x_t and can be reformulated by applying the formula of total probability:

$$\begin{aligned} \eta &= \frac{1}{p(o_t|o_{1:t-1}, a_{1:t})} \\ &= \frac{1}{\int p(o_t|x_t, o_{1:t-1}, a_{1:t})\; p(x_t|o_{1:t-1}, a_{1:t})\; dx_t} \text{ (tot. prob.)} && \text{(A.3)} \end{aligned}$$

Assuming that the Markov property holds as described above, Eq. A.2 can now be reduced by realizing that given the state x_t, the previous history of actions, observations, and states provides no further evidence for the probability distribution of receiving observation o_t:

$$\begin{aligned} p(x_t|o_{1:t}, a_{1:t}) &= \eta\; p(o_t|x_t, o_{1:t-1}, a_{1:t})\; p(x_t|o_{1:t-1}, a_{1:t}) \\ &= \eta\; p(o_t|x_t)\; p(x_t|o_{1:t-1}, a_{1:t}) \text{ (Markov prop.)} && \text{(A.4)} \end{aligned}$$

$p(o_t|x_t)$ is typically referred to as the *sensor model* as it describes the probability of obtaining a particular observation given the current state of the world and has to be determined for the sensor apparatus of the robot at hand.

In the next two steps, Eq. A.4 is turned into a recursive version: $p(x_t|o_{1:t}, a_{1:t})$ is computed from $p(x_{t-1}|o_{1:t-1}, a_{1:t-1})$. First, the law of total probability is applied to the rightmost term:

$$\begin{aligned} p(x_t|o_{1:t}, a_{1:t}) &= \eta\, p(o_t|x_t)\; p(x_t|o_{1:t-1}, a_{1:t}) \\ &= \eta\, p(o_t|x_t) \int p(x_t|x_{t-1}, o_{1:t-1}, a_{1:t})\; p(x_{t-1}|o_{1:t-1}, a_{1:t})\; dx_{t-1} \\ &\quad \text{(tot. prob.)} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(A.5)} \end{aligned}$$

Then, a_t is removed from $p(x_{t-1}|...)$ under the conditional independence assumption that controls are not selected based on current state:

$$\begin{aligned} p(x_t|o_{1:t}, a_{1:t}) &= \eta\, p(o_t|x_t) \int p(x_t|x_{t-1}, o_{1:t-1}, a_{1:t})\; p(x_{t-1}|o_{1:t-1}, a_{1:t})\; dx_{t-1} \\ &= \eta\, p(o_t|x_t) \int p(x_t|x_{t-1}, o_{1:t-1}, a_{1:t})\; p(x_{t-1}|o_{1:t-1}, a_{1:t-1})\; dx_{t-1} \\ &\quad \text{(cond. ind.)} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(A.6)} \end{aligned}$$

In a final step, the Markov property is used again to simplify the leftmost term of the integral by considering that given the current state and the action taken in this state, past actions and observations bear no further evidence on the successor state and thus can be dropped from the conditional probability:

$$\begin{aligned} p(x_t|o_{1:t}, a_{1:t}) &= \eta\, p(o_t|x_t) \int p(x_t|x_{t-1}, o_{1:t-1}, a_{1:t})\; p(x_{t-1}|o_{1:t-1}, a_{1:t-1})\; dx_{t-1} \\ &= \eta\, p(o_t|x_t) \int p(x_t|x_{t-1}, a_t)\; p(x_{t-1}|o_{1:t-1}, a_{1:t-1})\; dx_{t-1} \\ &\quad \text{(Markov prop.)} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(A.7)} \end{aligned}$$

The term $p(x_t|x_{t-1}, a_t)$ is the so-called *motion model* that similarly to the sensor model has to be determined for the robot. Given these two models, the recursive Bayes filter as provided in Eq. A.7 describes how the probability distribution for time step t can be determined from the probability distribution for time step $t-1$.

However, in most realistic cases exact computation and representation of the probability distribution is not feasible. Exceptions are discrete state spaces, where the integral is replaced by a sum, and situations in which the probabilities follow a particular parametric distribution and can be computed by a closed form version of Eq. A.7 (see Sect. A.2 about parametric filters). In all other cases, approximations have to be employed which can broadly be classified into parametric filters (Sect. A.2) and particle filters (Sect. A.3).

In general, incorporating a new action by computing the result of the integral in Eq. A.7 is referred to as the *prediction step*, while incorporating a new observation by multiplication of the sensor model with the result of the prediction step (followed by normalization) is called the *correction step*.

A.2 Parametric Filters

Parametric filters employ parametric probability density functions in order to represent $p(x_t|o_{1:t}, a_{1:t})$. As a consequence, their application is limited to scenarios in which $p(x_t|o_{1:t}, a_{1:t})$ is either guaranteed to always follow this particular parametric distribution or at least to be reasonably approximated by this kind of parametric distribution.

The main approaches in this class are the *Kalman filter* (Kalman, 1960) and the *information filter* (Mutambara, 1998), which both employ multivariate normal distributions. We will only describe the Kalman and its extension, the *extended Kalman filter*, here.

A.2.1 Kalman Filter

The Kalman filter was developed as a filtering method for systems which fulfill the following conditions:

1. the state space is continuous (no discrete state variables),
2. the Markov assumption underlying the recursive Bayes filter holds,
3. the motion model is linear Gaussian (linear in its arguments plus Gaussian noise),
4. the sensor model is linear Gaussian,
5. the initial state probability $p(x_0)$ is Gaussian.

Under these conditions the posterior of the Bayes filter always stays normally distributed and the integral and multiplications of Eq. A.7 can be computed in closed form.

In the Kalman filter, the normal probability distributions are represented by their means and covariance matrices. The linear Gaussian assumption for the motion model means that the system develops according to the linear equation:

$$x_t = A_t x_{t-1} + B_t a_t + v_t \tag{A.8}$$

where A_t and B_t are non-random matrices ensuring the linearity of the state transition and v_t is an additive random noise vector that is normally distributed with 0 mean and covariance matrix Q_t.

The linear Gaussian sensor model assumption means that observations depend on the state according to the equation:

$$o_t = C_t x_t + w_t \tag{A.9}$$

where C_t is a non-random matrix and w_t is an additive noise vector that is normally distributed with 0 mean and covariance matrix R_t.

The closed form formulas for updating the mean and covariance matrix based on the recursive Bayes filter are then given by

$$\overline{\mu_t} = A_t\mu_{t-1} + B_t a_t \tag{A.10}$$
$$\overline{\Sigma_t} = A_t\Sigma_{t-1}A_t^T + Q_t \tag{A.11}$$
$$K_t = \overline{\Sigma_t}C_t^T(C_t\overline{\Sigma_t}C_t^T + R_t)^{-1} \tag{A.12}$$
$$\mu_t = \overline{\mu_t} + K_t(o_t - C_t\overline{\mu_t}) \tag{A.13}$$
$$\Sigma_t = (I - K_tC_t)\overline{\Sigma_t} \tag{A.14}$$

I here stands for the identity matrix, and the auxiliary quantity K_t is called *Kalman gain*. The difference between observation and expected observation $(o_t - C_t\overline{\mu_t})$ in Eq. A.13 is referred to as the *innovation*.

In the context of robot localization and mapping, the most limiting factor of the Kalman filter is that it does not provide a good approximation when multiple distinct hypotheses exist (for instance, being in one particular room or another in the case of localization), meaning that the real probability distribution is multimodal. In contrast, the normal distribution employed in the Kalman filter is unimodal. When using the Kalman filter to approximate a multimodal distribution, the mean will most likely lie somewhere between the modes of the real distribution.

During mapping, the mean vector and covariance matrix grow whenever an observed feature is classified as not yet contained in the map during the data association step. The Kalman filter approach generally relies on a correct data association as otherwise the filter tends to diverge rather quickly. The size of the covariance matrix Σ_t grows quadratically with the number of features contained in the map. If the dimensionality of the observation vector is small compared to the dimensionality n of the state space, the time complexity of the Kalman filter is dominated by the $O(n^2)$ costs of the matrix multiplications.

A.2.2 Extended Kalman Filter

The requirements of linear state transition and linear sensor model are rather restrictive and rarely hold in practice. Therefore, different extensions have been conceived for the Kalman filter, in which the linearity assumption is weakened. In the *extended Kalman filter* a normally distributed approximation of the posterior is computed by using a linear approximation of the motion model and the sensor model when computing the covariance matrices in the prediction and correction steps. The linearizations used are the first-order Taylor expansions developed around μ_t and $\overline{\mu_t}$, respectively.

As we are now considering potentially nonlinear functions, the motion model and sensor model of Eqs. A.8 and A.9 are replaced with

$$x_t = f(a_t, x_{t-1}) + v_t \tag{A.15}$$
$$o_t = g(x_t) + w_t \tag{A.16}$$

This results in the following update equations for the extended Kalman filter:

$$\begin{aligned}
\overline{\mu_t} &= f(a_t, \mu_{t-1}) && \text{(A.17)}\\
\overline{\Sigma_t} &= F_t \Sigma_{t-1} F_t^T + Q_t && \text{(A.18)}\\
K_t &= \overline{\Sigma_t} G_t^T (G_t \overline{\Sigma_t} G_t^T + R_t)^{-1} && \text{(A.19)}\\
\mu_t &= \overline{\mu_t} + K_t(o_t - g(\overline{\mu_t})) && \text{(A.20)}\\
\Sigma_t &= (I - K_t G_t)\overline{\Sigma_t} && \text{(A.21)}
\end{aligned}$$

where f and g have been replaced by the Jacobians $F_t = \frac{\partial f}{\partial x}\Big|_{\mu, a_t}$ and $G_t = \frac{\partial g}{\partial x}\Big|_{\overline{\mu_t}}$ in Eqs. A.18, A.19, and A.21 because of the linearization.

The time complexity of the extended Kalman filter is the same as for the basic Kalman filter.

A.3 Nonparametric Filters

In contrast to parametric filters, nonparametric filters can approximate arbitrary probability distributions. They typically represent the probability distributions by a finite number of samples over the state space. Two main groups can be distinguished: *Histogram filters* partition the state space into regions and store a single probability value for each region. In a *particle filter* the samples are randomly drawn from the probability distribution they represent. We will focus on particle filters here.

A.3.1 Particle Filter

In the particle filter approach, a set of n samples (or *particles*) $\mathbf{x}_{k,t}, 0 \leq k < n$ is maintained at each time step. The probability is represented by the density of the particles. More precisely, the likelihood for a particular state x_t to be part of the sample set at time step t is supposed to be proportional to $p(x_t|a_{1:t}, o_{1:t})$.

The basic particle filter algorithm works as follows:

1. For every sample $\mathbf{x}_{k,t-1}$ from the previous sample set, a sample $\overline{\mathbf{x}}_{k,t}$ is randomly determined according to $p(x_t|a_t, \mathbf{x}_{k,t-1})$ (prediction step).

2. For every new sample $\overline{\mathbf{x}}_{k,t}$, an importance factor $w_{k,t}$ is computed which is the probability of observing o_t given $\overline{\mathbf{x}}_{k,t}$ (correction step):

$$w_{k,t} = p(o_t|\overline{\mathbf{x}}_{k,t}) \tag{A.22}$$

3. Particles $\mathbf{x}_{k,t}$ are randomly chosen from the set $\overline{\mathbf{x}}_{k,t}$ determined in step 1 according to the importance weights determined in step 2 (resampling step). The result is a new set of particles representing the posterior for time step t.

Due to its stochastic nature, the particle filter has a variance which depends on the number of particles used: the higher the number of particles, the lower the variance. Strictly speaking, the particle filter only converges to the correct posterior for an infinite number of particles. A related problem is the so-called *particle depletion problem*, the problem that no particle may reside near the correct state because the number of particles is too small to cover all regions with a high probability (the probability distribution has too many modes). Overall, the "right" number of particles depends on the dimensionality of the state space and the complexity of the represented probability distribution. For many applications, a rather small number of particles has been demonstrated to be sufficient.

The variance problem caused by losing diversity of particles can be alleviated by reducing the frequency of resampling. In this case, the important factors are also stored and, when no resampling is done, are updated by the following operation:

$$w_{k,t} = p(o_t|\mathbf{x}_{k,t})\; w_{k,t-1} \tag{A.23}$$

In a resampling step, the importance factors then have to be reinitialized to 1. The decision on when to perform resampling can, for instance, be based on the variance of the importance factors: If the variance is high, resampling is advisable; otherwise not.

A.3.2 Rao-Blackwellized Particle Filter and FastSLAM

One important particle filter approach is the so-called Rao-Blackwellized particle filter (Doucet et al., 2000). The general idea is to marginalize out substructures from the posterior distribution. In the context of robot mapping, the key realization is that if the state vector x_t is split into two parts, the robot's pose s_t and the rest of the state variables simply referred to as the map m, the posterior for the complete history of poses $s_{0:t}$ (the robot's *trajectory*), and m can be factorized in the following way:

$$p(s_{0:t}, m|o_{1:t}, a_{1:t}) \quad = \quad p(s_{0:t}|o_{1:t}, a_{1-t})\; p(m|s_{0:t}, o_{1:t}, a_{1:t}) \tag{A.24}$$

The result is a product of two probability distributions: The right term describes the problem of estimating the map for known poses and can typically be computed exactly. The left part is a probability distribution that estimates the trajectory of the robot from the history of observations and actions and will be computed approximately using a particle filter approach. Accordingly, Rao-Blackwellized particle filters developed for robot mapping work in the following way: The particles represent the distribution over all trajectories, which means each particle represents a particular trajectory. In addition, each particle maintains its own map posterior $p(m|s_{0:t}, o_{1:t}, a_{1:t})$ based on its trajectory.

Rao-Blackwellized particle filters for robot mapping have been first described for a simple discrete state space by Murphy (2000) and then been realized on real robots for feature-based representations by Montemerlo et al. (2002) and later for grid maps (Hähnel et al., 2003a) under the term FastSLAM.

A.3.2.1 Feature-Based FastSLAM

Montemerlo et al. applied the idea of Rao-Blackwellized particle filters to feature-based representations (Montemerlo, 2003; Montemerlo et al., 2002). In this case, the map m consists of the positions m_i of l landmarks. Given the sequence of poses $s_{0:t}$, the position of each feature is conditionally independent of the positions of the other features. Thus, each feature location can be marginalized out individually and Eq. A.24 is replaced with:

$$p(s_{0:t}, m|o_{1:t}, a_{1:t}) = p(s_{0:t}|o_{1:t}, a_{1:t}) \prod_{i=1}^{l} p(m_i|s_{0:t}, o_{1_t}, a_{1_t}) \tag{A.25}$$

Each particle of the particle filter used for $p(s_{0:t}|o_{1:t}, a_{1:t})$ then maintains a set of extended Kalman filters, one for estimating the position of each landmark. The landmark estimators are organized in a binary tree so that the resampling step can be performed in logarithmic time, resulting in an overall time complexity of $O(n \log l)$ for one update, where n is the number of particles employed. In addition, the prediction step and the correction step only require the robot's current pose and not the complete trajectory; therefore each particle only needs to maintain the current pose. Data association can be performed for each particle individually or by sampling over different data associations. An improvement that incorporates o_t in the prediction step is described in Montemerlo et al. (2003) and the resulting algorithm is called FastSLAM 2.0.

A.3.2.2 Grid-Based FastSLAM

In the approach developed in Hähnel et al. (2003a) (see also Stachniss et al. (2005)) the maps associated with each particle are grid maps which are updated using a standard occupancy grid mapping approach to compute $p(m|s_{0:t}, o_{1:t}, a_{1:t}) = p(m|s_{0:t}, o_{1:t})$. The particle filter is only updated after several steps and is based on odometry estimates derived from scan matching and a learned parametric motion model. This results in a significant performance gain, allowing us to map large environments or use fewer particles.

A similar improvement to the prediction step as developed in the FastSLAM 2.0 algorithm is described for the grid-based variant in Grisetti et al. (2007a). More improvements are discussed in Grisetti et al. (2007b).

Appendix B

Qualitative Spatial Reasoning

Research on qualitative spatial reasoning (QSR) started at the end of the 1980s inspired by earlier work on temporal reasoning (Allen, 1983; van Beek, 1992; Freksa, 1992a). One goal of this research field is the development of efficient reasoning formalisms about sets of qualitative spatial relations. The most important reasoning problem is the *satisfiability* problem of deciding whether a set of spatial statements is consistent. The results of this research typically come in the form of so-called constraint calculi for expressing and reasoning about a particular aspect of space like mereotopology (Egenhofer, 1989; Randell et al., 1992; Renz & Nebel, 1999), direction and orientation (Dylla & Moratz, 2005; Frank, 1991; Freksa, 1992b; Ligozat, 1993, 1998; Moratz et al., 2000, 2005; Renz & Mitra, 2004; Schlieder, 1995), and position (Moratz et al., 2003). The expressiveness is limited to conjunctive relational statements, but as a result a better efficiency is achieved compared to more general reasoning approaches such as full geometric reasoning. Overviews on QSR research can be found in Cohn & Hazarika (2001) and Ligozat & Renz (2004).

B.1 Qualitative Constraint Calculi

An *n-ary qualitative constraint calculus* is concerned with n-ary relations over a potentially infinite *domain* $\mathcal{D}$ of (spatial) objects. The most common forms are binary calculi ($n = 2$) and ternary constraint calculi ($n = 3$). Typically the relations are derived from a *jointly exhaustive and pairwise disjoint* (JEPD) set of so-called *base relations* $\mathcal{B}$.

Definition B.1 (JEPD set of n-ary relations)**.** *A set of n-ary relations $\mathcal{B} \subseteq 2^{\mathcal{D}^n}$ over a domain $\mathcal{D}$ is called jointly exhaustive and pairwise disjoint (JEPD) if the relations of $\mathcal{B}$ cover $\mathcal{D}^n$ ($\bigcup_{R \in \mathcal{B}} R = \mathcal{D}^n$) and no two relations overlap ($\forall R, S \in \mathcal{B} \, : \, R \cap S = \emptyset$).*

The complete set of relations considered in a calculus is the set of *general relations* $\mathcal{R}_{\mathcal{B}}$ derived from the base relations $\mathcal{B}$ as follows.

Definition B.2 (Set of general relations of a set of JEPD relations)**.** *Given a JEPD set of n-ary relations $\mathcal{B}$, the set of general relations $\mathcal{R}_\mathcal{B}$ of $\mathcal{B}$ is the set of all unions of relations from $\mathcal{B}$:*

$$\mathcal{R}_\mathcal{B} = \left\{ R \subseteq \mathcal{D}^n \mid \exists X \subseteq \mathcal{B} : R = \bigcup_{x \in X} x \right\} \tag{B.1}$$

In the following, we will simply write $\{B_1, B_2, ..., B_m\}$ for the general relation which is the union of the base relations $B_1, B_2, ..., B_m \in \mathcal{B}$. $\emptyset$ will be used for the empty relation and U for the universal relation ($U = \bigcup_{B \in \mathcal{B}} B = \mathcal{D}^n$).

A qualitative constraint calculus employs operations defined over its set of general relations for elementary reasoning. The calculus has to be closed under these operations, meaning that the result is always again a relation from $\mathcal{R}_\mathcal{B}$. The operations can be grouped into three classes:

1. set-theoretic operations,
2. unary operations that permute the order of objects in the relation tuples, and
3. composition operations that combine information from two (or more) relations.

The set-theoretic operations are the standard *complement* ($\bar{\ }$), *union* ($\cup$), and *intersection* ($\cap$) operations applied to the relations of $\mathcal{R}_\mathcal{B}$.

Definition B.3 (Complement, union, intersection)**.** *Given relations $R, S \in \mathcal{R}_\mathcal{B}$, the operations of complement, union, and intersection are defined as follows:*

$$\bar{R} = U \setminus R = \{ x \mid x \in U \wedge x \notin R \} \quad \textit{(complement)} \tag{B.2}$$
$$R \cup S = \{ x \mid x \in R \vee x \in S \} \quad \textit{(union)} \tag{B.3}$$
$$R \cap S = \{ x \mid x \in R \wedge x \in S \} \quad \textit{(intersection)} \tag{B.4}$$

A set of general relations is trivially closed under these three operations. The results of the operations can directly be computed by applying the set-theoretic operations to the set notation introduced above (e.g., $\{B_1, B_4, B_7\} \cap \{B_4, B_7, B_8\} = \{B_4, B_7\}$).

For general n-ary calculi two operations are required to construct all permutations of the ordering of objects in the relation tuples: The *converse* operation ($^\smile$) exchanges the order of the last and second-to-last objects in the tuples, while the *rotation* operation ($^\frown$) rotates the relation tuples to the right.

Definition B.4 (Converse, rotation)**.** *Given a relation $R \in \mathcal{R}_\mathcal{B}$, the operations of converse and rotation are defined as follows:*

$$R^\smile = \{ (d_1, ..., d_{n-2}, d_n, d_{n-1}) \in \mathcal{D}^n \mid (d_1, ..., d_{n-2}, d_{n-1}, d_n) \in R \} \quad \textit{(converse)}$$
$$R^\frown = \{ (d_n, d_1, ..., d_{n-1}) \in \mathcal{D}^n \mid (d_1, ..., d_{n-1}, d_n) \in R \} \quad \textit{(rotation)} \tag{B.5}$$

Converse and rotation can be used to generate all permutations of the tuples and thus allow for a change of perspective: From knowing that relation R holds between objects A, B, C ($(A, B, C) \in R$), it follows that the relation holding between B, A, C is $(R^{\smile})^{\frown}$ (the rotation of the converse of R). For binary calculi, converse and rotation do exactly the same thing, namely swap the elements of all relation pairs, and thus typically only converse is specified. Given that our set of general relations is closed under converse and rotation, these two operations can be specified by providing the results in table form. In principle, it is sufficient to provide the result of applying them to the set of base relations. The result for general relations can then be computed by taking the union of the results for the contained base relations.

Finally, the *composition* ($\circ$) operation takes two relations and combines them into a new relation. For better readability, we define the composition for binary and ternary calculi individually instead of giving a general definition.[1]

Definition B.5 (Binary composition)**.** *Given two binary relations $R, S \in \mathcal{R}_\mathcal{B}$, their composition is defined as*

$$R \circ S = \{ (A, C) \in \mathcal{D}^2 \mid \exists B \in \mathcal{D} \ : \ (A, B) \in R \wedge (B, C) \in S \} \tag{B.6}$$

Definition B.6 (Ternary composition)**.** *Given two ternary relations $R, S \in \mathcal{R}_\mathcal{B}$, their composition is defined as*

$$R \circ S = \{ (A, B, D) \in \mathcal{D}^3 \mid \exists C \in \mathcal{D} \ : \ (A, B, C) \in R \wedge (B, C, D) \in S \} \tag{B.7}$$

As for the converse and rotation operations, the composition operation of a calculus is typically specified as a table, either for only the base relations (an $m \times m$ matrix for m base relations) or for the complete set of general relations (a $2^m \times 2^m$ matrix for m base relations). In the first case, the composition of two general relations is computed by taking the union of the component-wise compositions.

We now have all the parts in place to give a definition of an n-ary qualitative constraint calculus.

Definition B.7 (n-ary qualitative constraint calculus)**.** *An n-ary qualitative constraint calculus is an 8-tuple $(\mathcal{D}, \mathcal{B}, \bar{}, \cup, \cap, ^{\smile}, ^{\frown}, \circ)$ where*

- *the domain $\mathcal{D}$ is a potentially infinite set of objects,*
- *the set of base relations $\mathcal{B} \subseteq 2^{\mathcal{D}^n}$ is a finite, nonempty JEPD set of n-ary relations over $\mathcal{D}$,*
- *$\bar{}, \cup, \cap, ^{\smile}, ^{\frown}, \circ$ are the operations over the set of general relations $\mathcal{R}_\mathcal{B}$ of $\mathcal{B}$ as defined above, and*
- *$\mathcal{R}_\mathcal{B}$ is closed under $\bar{}, \cup, \cap, ^{\smile}, ^{\frown}, \circ$.*

[1] Ternary composition has also been defined as a ternary operator in Condotta et al. (2006).

One simple example of a binary calculus used in this work is the cardinal direction calculus by Ligozat (1998) (cf. Fig. 6.4). It consists of nine base relations for relating points in the plane ($\mathcal{D} = \mathbb{R}^2$) corresponding to the eight cardinal directions $n, ne, w, sw, s, se, e, ne$ and the equal relation eq. The relations are typically written as $A\ ne\ B$ instead of $(A, B) \in ne$, and the meaning is that object A is to the northeast of B.

Tables for the converse and composition of the cardinal direction calculus state, for instance, that the converse of se is nw (if A is to the southeast of B, B is to the northwest of A) and that the composition of n and sw is $\{nw, w, sw\}$ (if A is to the north of B and B is to the southwest of C, it follows that A is either to the northwest, to the west, or to the southwest of C).

B.2 Weak vs. Strong Operations

In our definition of a qualitative constraint calculus, we demand that the set of general relations be closed under converse, rotation, and composition (as well as under the three set-theoretic operations). However, for many sets of JEPD base relations, the corresponding general relations are not closed at least under some of these operations (typically under composition). Often it can be shown that no finite set of relations exists that contains the base relations and is closed under all operations.

In this case, some kind of approximation has to be used in order for us to still remain with a finite set of relations. This is done by weakening the definition of the problematic operations by taking the smallest relation from $\mathcal{R}_\mathcal{B}$ that contains the result of the strong operation. For instance, a *weak composition* (in the binary case) would be defined as the union of base relations that have a nonempty intersection with the result of the strong composition:

$$R \circ_w S = \{\ B \in \mathcal{B} \mid B \cap (R \circ S) \neq \emptyset\ \} \tag{B.8}$$

The consequences of having to resort to weak operations for constraint-based reasoning as described in the next section are still the topic of ongoing research. A discussion of this matter can be found in Renz & Ligozat (2005).

B.3 Constraint Networks and Consistency

For the following considerations, we restrict ourselves to binary constraint calculi. All techniques can be straightforwardly transferred to calculi of a higher arity.

A qualitative constraint calculus provides a formal language restricted to stating conjunctive relational facts holding between object constants. This can for instance be used to describe an observed spatial configuration of objects. Uncertainty can be expressed by referring to general relations that are unions of multiple base relations, where U represents complete ignorance.

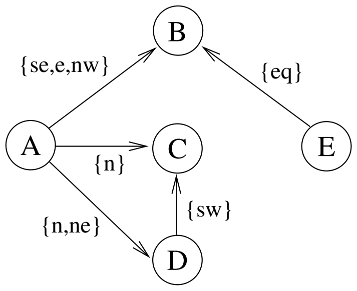


Figure B.1: A constraint network for the cardinal direction calculus

Such a set of relation statements over a finite set of objects can be depicted as a *constraint network*, as shown in Fig. B.1. The objects are denoted by the nodes and each directed edge is labeled by a constraint which is a relation from $\mathcal{R}_\mathcal{B}$ of the employed calculus. If the edge connecting object A with object B is annotated with relation R, this means that the pair of values from $\mathcal{D}$ (e.g., coordinates in the plane) that can be assigned to A and B is restricted to being one contained in R. If no edge connects two nodes, this corresponds to an edge labeled with the universal relation U, which is usually omitted.

A constraint network defines a particular kind of constraint satisfaction problem (CSP). We will call it a spatial CSP here.

Definition B.8 (Spatial constraint satisfaction problem). *A spatial constraint problem is a triple* $(\mathcal{QCC}, \mathcal{V}, \mathcal{C})$ *consisting of a finite set* $\mathcal{V}$ *of variables* $V_1, ..., V_m$ *over the domain of a given qualitative constraint calculus* $\mathcal{QCC}$ *and a set* $\mathcal{C}$ *of binary constraints* $C_{ij} \in \mathcal{R}_\mathcal{B}$ *(*$1 \leq i < j \leq m$*) holding between* V_i *and* V_j *where* $\mathcal{R}_\mathcal{B}$ *is the set of general relations of* $\mathcal{QCC}$.

A spatial CSP has a solution if one can assign values from the spatial domain to the variables so that all constraints are satisfied.

Definition B.9 (Assignment). *An assignment is a total function assign* $: \mathcal{V} \to \mathcal{D}$ *that maps each variable from a spatial CSP to a value from the domain.*

Definition B.10 (Solution). *A solution of a spatial CSP is an assignment assign which satisfies all the constraints:*

$$\forall V_i, V_j \in \mathcal{V}, 1 \leq i < j \leq m : (\mathit{assign}(V_i), \mathit{assign}(V_j)) \in C_{ij} \tag{B.9}$$

One fundamental reasoning problem of QSR is the *satisfiability* or *consistency problem*, concerned with whether a spatial CSP has a solution or not.

Definition B.11 (Satisfiable, consistent). *A spatial CSP is said to be satisfiable or consistent if it has at least one solution.*

In contrast to many other kinds of CSPs, the domain in our case is infinite. Therefore, special techniques for checking consistency had to be developed which are based on the operations of the calculus. Before we turn to these techniques, we need to define two more concepts.

Definition B.12 (Scenario, atomic)**.** *A spatial CSP (or constraint network) is called a scenario or atomic if all constraints are base relations.*

Definition B.13 (Refinement)**.** *A refinement of a spatial CSP P with variable set $\mathcal{V}$ and constraints C_{ij} is another spatial CSP P' over the same set of variables with $C'_{ij} \subseteq C_{ij}$ for all $1 \leq i < j \leq |\mathcal{V}|$.*

Thus, refinements can be constructed by removing base relations from the constraints of a CSP. If a constraint network contains constraints that are not base relations (meaning it is not a scenario), we are often interested in finding a scenario that is a refinement of this network. If we can show this scenario to be consistent, we know that the original network is consistent as well. Figure B.2 shows in (a) a non-atomic network and in (b) and (c) the two only scenarios which are refinements of this network.

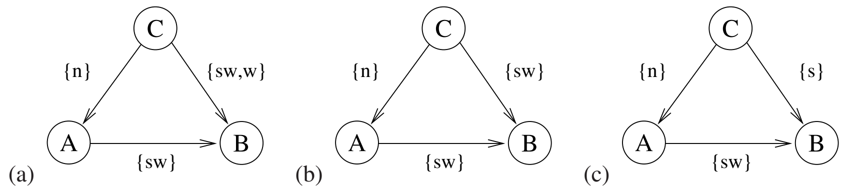


Figure B.2: A non-atomic constraint network (a) with possible scenarios (b) and (c)

B.4 Checking Consistency

The techniques developed for relational constraint problems are based on weaker forms of consistency called *local consistencies* which can be enforced based on the operations of the calculus and which are under particular conditions sufficient to decide consistency. If this is the case, enforcing this local consistency without one constraint becoming the empty relation proves consistency of the original network.

One important form of local consistency is *path consistency*, which (in binary CSPs) means that for every triple of variables each consistent evaluation of the first two variables can be extended to the third variable in such a way that all constraints are satisfied. To enforce path consistency, a syntactic procedure called the *algebraic closure algorithm* is used (Mackworth, 1977; Montanari, 1974). The algorithm runs

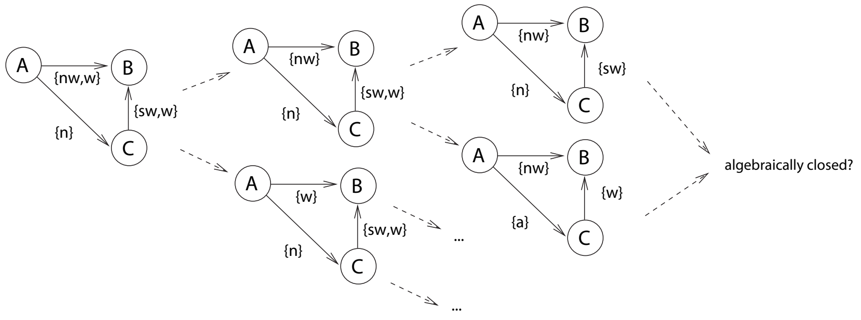


Figure B.3: Backtracking search through the refinements until an algebraically closed scenario is found

in $O(n^3)$ time and in essence performs the following operation on triples of variables V_i, V_j, V_k until a fix point has been reached:

$$C_{ik} = C_{ik} \cap (C_{ij} \circ C_{jk}) \tag{B.10}$$

If one constraint becomes the empty relation in the process, the original network was inconsistent. It has to be noted that for some algorithms this procedure does not actually enforce path consistency but only approximates it. Moreover, for some calculi, algebraic closure is not sufficient to decide overall consistency in general but it is sufficient if the network is a scenario. In this case, a backtracking search can be used in which all possible scenarios of the original network are generated by recursively splitting constraints as indicated in Fig. B.3. Each generated scenario is checked using the algebraic closure algorithm. If one is shown to be consistent, the original network is consistent as well. If no consistent scenario exists, inconsistency of the original network has been shown. Unfortunately, this approach has an exponential worst-case time complexity.

For some calculi larger so-called *maximal tractable subsets* of relations including the base relations are known for which algebraic closure decides consistency. These subsets can be used to increase performance of the backtrack algorithm by only refining until all constraints are relations from the subset (Ladkin & Reinefeld, 1992). Even when algebraic closure cannot be employed as a decision procedure for consistency, it can still be used as an incomplete method that may not discover all inconsistencies.

The standard constraint-based reasoning techniques described here have been realized for the most common spatial calculi in the SparQ toolbox (Wallgrün et al., 2006, 2007). This toolbox is used to implement the approach described in Chap. 6 of this book.

Bibliography

Abdulkader, A. M. (1998). *Parallel Algorithms for Labelled Graph Matching*. Ph.D. thesis, Colorado School of Mines.

Aguirre, E. & González, A. (2002). Integrating fuzzy topological maps and fuzzy geometric maps for behavior-based robots. *International Journal of Intelligent Systems*, 17(3):333–368.

Aho, A., Hopcroft, J., & Ullman, J. (1974). *The Design and Analysis of Computer Algorithms*. Addison-Wesley.

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.

Arras, K., Castellanos, J., Schilt, M., & Siegwart, R. (2003). Feature-based multi-hypothesis localization and tracking using geometric constraints. *Robotics and Autonomous Systems Journal*, 44(1).

Aurenhammer, F. (1991). Voronoi diagrams – A survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345–405.

Bailey, T. (2001). *Mobile Robot Localisation and Mapping in Extensive Outdoor Environments*. Ph.D. thesis, Australian Centre for Field Robotics, University of Sydney.

Bailey, T., Nieto, J., & Nebot, E. (2006). Consistency of the FastSLAM algorithm. In *IEEE International Conference on Robotics and Automation (ICRA-06)*, (pp. 424–429).

Bar-Shalom, Y. & Fortmann, T. E. (1988). *Tracking and Data Association*. Academic Press.

van Beek, P. (1992). Reasoning about qualitative temporal information. *Artificial Intelligence*, 58:297–326.

Beeson, P., Jong, N. K., & Kuipers, B. (2005). Towards autonomous topological place detection using the Extended Voronoi Graph. In *IEEE International Conference on Robotics and Automation (ICRA-05)*, (pp. 4373–4379).

Bender, M. A., Fernández, A., Ron, D., Sahai, A., & Vadhan, S. (1998). The power of a pebble: Exploring and mapping directed graphs. In *STOC '98: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, (pp. 269–278). New York, NY, USA: ACM.

Blum, H. (1967). A transformation for extracting new descriptors of shape. In W. Wathen-Dunn (ed.), *Models for the Perception of Speech and Visual Form*, (pp. 362–381). MIT Press.

Bosse, M., Newman, P., Leonard, J., Soika, M., Feiten, W., & Teller, S. (2003). An Atlas framework for scalable mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-03)*.

Bourgault, F., Makarenko, A., Williams, S., Grocholsky, B., & Durrant-Whyte, H. (2002). Information based adaptive robotic exploration. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-02)*, vol. 1, (pp. 540–545).

Braitenberg, V. (1984). *Vehicles: Experiments in Synthetic Psychology*. MIT Press.

Briggs, R. (1973). Urban cognitive distance. In R. Downs & D. Stea (eds.), *Image and Environment: Cognitive Mapping and Spatial Behaviour*, (pp. 361–388). Chicago: Aldine.

Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics And Automation*, 2:14–23.

Bunke, H. (2000). Graph matching: Theoretical foundations, algorithms, and applications. In *International Conference on Vision Interface*, (pp. 82–88).

Bunke, H. & Messmer, B. (1997). Recent advances in graph matching. *International Journal on Pattern Recognition and Artificial Intelligence*, 11(1):69–203.

Burgard, W., Fox, D., Jans, H., Matenar, C., & Thrun, S. (1999). Sonar-based mapping with mobile robots using EM. In *Proceedings 16th International Conference on Machine Learning*, (pp. 67–76).

Buschka, P. (2005). *An Investigation of Hybrid Maps for Mobile Robots*. Ph.D. thesis, Örebro University.

Buschka, P. & Saffiotti, A. (2004). Some notes on the use of hybrid maps for mobile robots. In *Proceedings of the 8th International Conference on Intelligent Autonomous Systems (IAS-04)*, (pp. 547–556).

Castellanos, J. A., Montiel, J. M. M., Neira, J., & Tardós, J. (1999). The SPmap: A probabilistic framework for simultaneous localization and map building. *IEEE Transactions on Robotics and Automation*, 15(5):948–952.

Chatila, R. & Laumond, J.-P. (1985). Position referencing and consistent world modeling for mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA-85)*, (pp. 138–145).

Choset, H. & Burdick, J. (2000). Sensor-based exploration: The Hierarchical Generalized Voronoi Graph. *The International Journal of Robotics Research*, 19(2):96–125.

Choset, H. & Nagatani, K. (2001). Topological simultaneous localization and mapping (SLAM): Toward exact localization without explicit localization. *IEEE Transactions on Robotics and Automation*, 17(2):125–137.

Choset, H., Walker, S., Eiamsa-Ard, K., & Burdick, J. (2000). Sensor-based exploration: Incremental construction of the Hierarchical Generalized Voronoi Graph. *The International Journal of Robotics Research*, 19(2):126 – 148.

Cohn, A. G. & Hazarika, S. M. (2001). Qualitative spatial representation and reasoning: An overview. *Fundamenta Informaticae*, 46(1-2):1–29.

Condotta, J.-F., Ligozat, G., & Saade, M. (2006). A generic toolkit for n-ary qualitative temporal and spatial calculi. In *Proceedings of the 13th International Symposium on Temporal Representation and Reasoning (TIME'06)*, (pp. 78–86).

Cox, I. J. & Leonard, J. J. (1994). Modeling a dynamic environment using a Bayesian multiple hypothesis approach. *Artificial Intelligence*, 66(2):311 – 344.

Crowley, J. (1989). World modeling and position estimation for a mobile robot using ultrasonic ranging. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA-89)*, (pp. 674–680).

Darken, R. P., Allard, T., & Achille, L. B. (1999). Spatial orientation and wayfinding in large-scale virtual spaces II: Guest editors' introduction. *Presence: Teleoperators & Virtual Environments*, 8(6):iii–vi.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B*, 39:1–38.

Denis, M. (1997). The description of routes: A cognitive approach to the production of spatial discourse. *Cahiers Psychologie Cognitive*, 16(4):409–458.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271.

Dimitrov, P., Phillips, C., & Siddiqi, K. (2000). Robust and efficient skeletal graphs. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 417–423).

Dissanayake, M. G., Newman, P., Clark, S., Durrant-Whyte, H., & Csorba, M. (2001). A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241.

Doucet, A., de Freitas, N., Murphy, K. P., & Russell, S. J. (2000). Rao-Blackwellised particle filtering for dynamic Bayesian networks. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI '00)*, (pp. 176–183). Morgan Kaufmann Publishers Inc.

Downs, R. M. & Stea, D. (1973). Cognitive maps and spatial behavior: Process and products. In R. M. Downs & D. Stea (eds.), *Image and Environment*. Chicago: Aldine.

Duckett, T. & Nehmzow, U. (1999a). Exploration of unknown environments using a compass, topological map and neural network. In *Proceedings 1999 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, (pp. 312–317).

Duckett, T. & Nehmzow, U. (1999b). Knowing your place in real world environments. In *1999 Third European Workshop on Advanced Mobile Robots (Eurobot'99). Proceedings*, (pp. 135–142).

Duda, R. O. & Hart, P. E. (1973). *Pattern Classification and Scene Analysis*. John Wiley & Sons.

Dudek, G., Jenkin, M., Milios, E., & Wilkes, D. (1991). Robotic exploration as graph construction. *IEEE Transactions on Robotics and Automation*, 7(6):859–865.

Dudek, G., Freedman, P., & Hadjres, S. (1996). Using multiple models for environmental mapping. *Journal of Robotic Systems*, 13(8):539–559.

Dudek, G., Jenkin, M., Milios, E., & Wilkes, D. (1997). Map validation and robot self-location in a graph-like world. *Robotics and Autonomous Systems*, 22(2):159–178.

Dylla, F. & Moratz, R. (2005). Exploiting qualitative spatial neighborhoods in the situation calculus. In C. Freksa, M. Knauff, B. Krieg-Brückner, B. Nebel, & T. Barkowsky (eds.), *Spatial Cognition IV. Reasoning, Action, Interaction: International Conference Spatial Cognition 2004*, vol. 3343 of *Lecture Notes in Artificial Intelligence*, (pp. 304–322). Berlin, Heidelberg: Springer.

Egenhofer, M. J. (1989). A formal definition of binary topological relationships. In W. Litwin & H.-J. Schek (eds.), *3rd International Conference, FODO 1989 on Foundations of Data Organization and Algorithms*, vol. 367 of *Lecture Notes in Computer Science*, (pp. 457–472). Springer.

Elfes, A. (1989). Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57.

Eliazar, A. I. & Parr, R. (2003). DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, (pp. 1135–1142).

Eliazar, A. I. & Parr, R. (2004). DP-SLAM 2.0. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation (ICRA-04)*, (pp. 1314–1320).

Engelson, S. & McDermott, D. (1992). Error correction in mobile robot map learning. In *Proceedings of the IEEE Conference on Robotics and Automation*, (pp. 2555–2560).

Eshera, M. A. & Fu, K. S. (1986). An image understanding system using attributed symbolic representation and inexact graph-matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(5):604–618.

Fernández, J. A. & González, J. (1997). A general world representation for mobile robot operations. In *Seventh Conference of the Spanish Association for the Artificial Intelligence (CAEPIA'97)*, (p. 3544).

Fernández, J. A. & González, J. (1998). Hierarchical graph search for mobile robot path planning. In *Proceedings 1998 IEEE International Conference on Robotics and Automation (ICRA-98)*, (pp. 656–661).

Fernández, J. A. & González, J. (2001). *Multi-Hierarchical Representations of Large-Scale Space – Applications to Mobile Robots*. Microprocessor-based and Intelligent Systems Engineering. Kluwer Academic Publishers.

Fernández-Madrigal, J. A. & González, J. (2002). Multihierarchical graph search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):103–113.

Frank, A. (1991). Qualitative spatial reasoning about cardinal directions. In *Proceedings of the American Congress on Surveying and Mapping (ACSM-ASPRS)*, (pp. 148–167).

Franz, M. O., Schölkopf, B., Mallot, H. A., & Bülthoff, H. H. (1998). Learning view graphs for robot navigation. *Autonomous Robots*, 5(1):111–125.

Freksa, C. (1992a). Temporal reasoning based on semi-intervals. *Artificial Intelligence*, 54(1):199–227.

Freksa, C. (1992b). Using orientation information for qualitative spatial reasoning. In A. U. Frank, I. Campari, & U. Formentini (eds.), *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, vol. 639 of *Lecture Notes in Computer Science*, (pp. 162–178). Berlin: Springer.

Frese, U. (2006a). A discussion of simultaneous localization and mapping. *Autonomous Robots*, 20(1):25–42.

Frese, U. (2006b). Treemap: An O(log n) algorithm for indoor simultaneous localization and mapping. *Autonomous Robots*, 21(2):103–122.

Frese, U. & Hirzinger, G. (2001). Simultaneous localization and mapping – A discussion. In *Proceedings of the IJCAI Workshop on Reasoning with Uncertainty in Robotics*, (pp. 17–26).

Frese, U. & Schröder, L. (2006). Closing a million-landmarks loop. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-06)*, (pp. 5032–5039).

Galindo, C., Saffiotti, A., Coradeschi, S., Buschka, P., Fernández-Madrigal, J. A., & González, J. (2005). Multi-hierarchical semantic maps for mobile robotics. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-05)*, (pp. 3492–3497).

Garey, M. R. & Johnson, D. S. (1979). *Computers and Intractability*. Freeman.

Golledge, R. G. (1999). Human wayfinding and cognitive maps. In R. G. Golledge (ed.), *Wayfinding Behavior: Cognitive Mapping and Other Spatial Processes*, (pp. 5–45). Johns Hopkins Press.

Grimson, W. E. L. (1990). *Object Recognition by Computer – The Role of Geometric Constraints*. MIT Press, Cambridge, MA.

Grisetti, G., Stachniss, C., & Burgard, W. (2007a). Improved techniques for grid mapping with Rao-Blackwellized particle filters. *IEEE Transactions on Robotics*, 23:34–46.

Grisetti, G., Tipaldi, G., Stachniss, C., Burgard, W., & Nardi, D. (2007b). Fast and accurate SLAM with Rao-Blackwellized particle filters. *Journal of Robotics & Autonomous Systems*, 55(1):30–38.

Guivant, J. & Nebot, E. (2003). Solving computational and memory requirements of feature-based simultaneous localization and mapping algorithms. *IEEE Transactions on Robotics and Automation*, 19(4):749– 755.

Guivant, J. & Nebot, E. M. (2001). Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *IEEE Transactions on Robotics and Automation*, 17(3):242–257.

Hähnel, D., Burgard, W., Fox, D., & Thrun, S. (2003a). An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-03)*, (pp. 206–211).

Hähnel, D., Burgard, W., & Thrun, S. (2003b). Learning compact 3D models of indoor and outdoor environments with a mobile robot. *Robotics and Autonomous Systems*, 44:15–27.

Hähnel, D., Burgard, W., Wegbreit, B., & Thrun, S. (2003). Towards lazy data association in SLAM. In *Proceedings of the 11th International Symposium of Robotics Research (ISRR'03)*.

Hart, P., Nilsson, N., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4:100–107.

Hirtle, S. C. & Jonides, J. (1985). Evidence of hierarchies in cognitive maps. *Memory & Cognition*, 13:208–217.

Hopcroft, J. & Tarjan, R. (1974). Efficient planarity testing. *Journal of the ACM*, 21(4):549–568.

Hopcroft, J. E. & Wong, J. K. (1974). Linear time algorithm for isomorphism of planar graphs (preliminary report). In *STOC '74: Proceedings of the Sixth Annual ACM Symposium on Theory of Computing*, (pp. 172–184). New York, NY, USA: ACM.

Howard, R. A. (1960). *Dynamic Programming and Markov Processes*. New York: Wiley.

Hübner, W. & Mallot, H. A. (2007). Metric embedding of view-graphs. *Autonomous Robots*, 23(3):183–196.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME—Journal of Basic Engineering*, (pp. 35–45).

Krieg-Brückner, B., Frese, U., Lüttich, K., Mandel, C., Mossakowski, T., & Ross, R. (2005). Specification of an Ontology for Route Graphs. In C. Freksa, M. Knauff, B. Krieg-Brückner, B. Nebel, & T. Barkowsky (eds.), *Spatial Cognition IV. Reasoning, Action, Interaction: International Conference Spatial Cognition 2004*, vol. 3343 of *Lecture Notes in Artificial Intelligence*, (pp. 390–412). Springer.

Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97.

Kuipers, B. (1985). A map-learning critter. *Tech. Rep. AITR85-17*, AI Laboratory, UT Austin.

Kuipers, B. (2000). The Spatial Semantic Hierarchy. *Artificial Intelligence*, (119):191–233.

Kuipers, B. & Byun, Y.-T. (1991). A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics and Autonomous Systems*, (8):47–63.

Kuipers, B., Modayil, J., Beeson, P., MacMahon, M., & Savelli, F. (2004). Local metrical and global topological maps in the hybrid Spatial Semantic Hierarchy. In *Proceedings IEEE International Conference on Robotics and Automation 2004 (ICRA-04)*, (pp. 4845–4851).

Kuipers, B. J. (1978). Modeling spatial knowledge. *Cognitive Science*, 2:129–153.

Kuipers, B. J. & Byun, Y.-T. (1988). A robust, qualitative method for robot spatial learning. In *AAAI 88. Seventh National Conference on Artificial Intelligence*, (pp. 774–779).

Kuipers, B. J. & Levitt, T. S. (1988). Navigation and mapping in large-scale space. *AI Magazine*, 9(2):25–43.

Ladkin, P. & Reinefeld, A. (1992). Effective solution of qualitative constraint problems. *Artificial Intelligence*, 57:105–124.

Latecki, L. J., Lakämper, R., Sun, X., & Wolter, D. (2005a). Geometric robot mapping. In *Proceedings of the 12th International Conference on Discrete Geometry for Computer Imagery (DGCI-05), Poitiers, France*, vol. 3429 of *Lecture Notes in Computer Science*, (pp. 11–22). Springer.

Latecki, L. J., Lakämper, R., & Wolter, D. (2005b). Incremental multi-robot mapping. In *Proceedings of 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-05)*, (pp. 3846–3851).

Latombe, J.-C. (1991). *Robot Motion Planning*. Kluwer Academic Publishers.

Lee, D. T. & Drysdale III, R. L. S. (1981). Generalization of Voronoi diagrams in the plane. *SIAM Journal on Computing*, 10(1):73–87.

Lempel, A., Even, S., & Cederbaum, I. (1967). An algorithm for planarity testing of graphs. In P. Rosenstiehl (ed.), *Theory of Graphs*, (pp. 215–232). Gordon and Breach.

Leonard, J. J. & Durrant-Whyte, H. F. (1991). Simultaneous map building and localization for an autonomous mobile robot. In *Proceedings of IEEE/RSJ International Workshop on Intelligent Robots and Systems*, (pp. 1442–1447).

Levitt, T. & Lawton, D. (1990). Qualitative navigation for mobile robots. *Artificial Intelligence*, 44:305–360.

Ligozat, G. (1993). Qualitative triangulation for spatial reasoning. In A. U. Frank & I. Campari (eds.), *Spatial Information Theory: A Theoretical Basis for GIS, (COSIT'93), Marciana Marina, Elba Island, Italy*, vol. 716 of *Lecture Notes in Computer Science*, (pp. 54–68). Springer.

Ligozat, G. (1998). Reasoning about cardinal directions. *Journal of Visual Languages and Computing*, 9:23–44.

Ligozat, G. & Renz, J. (2004). What is a qualitative calculus? A general framework. In C. Zhang, H. W. Guesgen, & W.-K. Yeap (eds.), *PRICAI 2004: Trends in Artificial Intelligence, 8th Pacific Rim International Conference on Artificial Intelligence, Proceedings*, vol. 3157 of *Lecture Notes in Computer Science*, (pp. 53–64). Springer.

Lim, J. H. & Leonard, J. J. (2000). Mobile robot relocation from echolocation constraints. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(9):1035–1041.

Lisien, B., Silver, D., Kantor, G., Rekleitis, I., & Choset, H. (2003). Hierarchical simultaneous localization and mapping. In *Proceedings of the 2003 IEEE/RSJ Interational Conference on Intelligent Robots and Systems (IROS-03)*, (pp. 448–453).

Liu, Y., Emery, R., Chakrabarti, D., Burgard, W., & Thrun, S. (2001). Using EM to learn 3D models with mobile robots. In *Proceedings of the International Conference on Machine Learning (ICML)*, (pp. 329–336).

Lovelace, K. L., Hegarty, M., & Montello, D. R. (1999). Elements of good route directions in familiar and unfamiliar environments. In C. Freksa & D. M. Mark (eds.), *Spatial Information Theory. Cognitive and Computational Foundations of Geopraphic Information Science (COSIT)*, vol. 1661 of *Lecture Notes on Computer Science*, (pp. 65–82). Berlin: Springer.

Lu, F. & Milios, E. (1997). Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4(4):333–349.

Mackworth, A. (1977). Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118.

Mahalanobis, P. (1936). On the generalized distance in statistics. In *Proceedings of the National Institute of Sciences of India*, vol. 12, (pp. 49–55).

Mataric, M. J. (1992). Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on Robotics and Automation*, 8(3):304–312.

Mayya, N. & Rajan, V. T. (1996). Voronoi diagrams of polygons: A framework for shape representation. *Journal of Mathematical Imaging and Vision*, 6(4):355–378.

McNamara, P. T. (1986). Mental representations of spatial relations. *Cognitive Psychology*, 18:87–121.

Mehlhorn, K., Näher, S., Seel, M., Seidel, R., Schilz, T., Schirra, S., & Uhrig, C. (1999). Checking geometric programs or verification of geometric structures. *Computational Geometry: Theory and Applications*, 12(1-2):85–103.

Modayil, J., Beeson, P., & Kuipers, B. (2004). Using the topological skeleton for scalable global metrical map-building. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-04)*, (pp. 1530–1536).

Montanari, U. (1974). Networks of constraints: Fundamental properties and applications to picture processing. *Information Science*, 7(2):95–132.

Montello, D. (2005). Navigation. In P. Shah & A. Miyake (eds.), *The Cambridge Handbook of Visuospatial Thinking*, chap. 7, (pp. 257–294).

Montemerlo, M. (2003). *FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association*. Ph.D. thesis, Robotics Institute, Carnegie Mellon University. Tech. report CMU-RI-TR-03-28.

Montemerlo, M. & Thrun, S. (2003). Simultaneous localization and mapping with unknown data association using FastSLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-03)*, (pp. 1985–1991).

Montemerlo, M., Thrun, S., Koller, D., & Wegbreit, B. (2002). FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, (pp. 593–598).

Montemerlo, M., Thrun, S., Koller, D., & Wegbreit, B. (2003). FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, (pp. 407–411).

Moratz, R. (2006). Representing relative direction as binary relation of oriented points. In *Proceedings of the 17th European Conference on Artificial Intelligence (ECAI 2006)*, (pp. 407–411).

Moratz, R., Renz, J., & Wolter, D. (2000). Qualitative spatial reasoning about line segments. In W. Horn (ed.), *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI)*, (pp. 234–238). Berlin, Germany: IOS Press.

Moratz, R., Nebel, B., & Freksa, C. (2003). Qualitative spatial reasoning about relative position: The tradeoff between strong formal properties and successful reasoning about route graphs. In C. Freksa, W. Brauer, C. Habel, & K. F. Wender (eds.), *Spatial Cognition III*, vol. 2685 of *Lecture Notes in Artificial Intelligence*, (pp. 385–400). Berlin, Heidelberg: Springer.

Moratz, R., Dylla, F., & Frommberger, L. (2005). A relative orientation algebra with adjustable granularity. In *Proceedings of the Workshop on Agents in Real-Time and Dynamic Environments (IJCAI 05)*.

Moravec, H. & Elfes, A. (1985). High resolution maps from angle sonar. In *Proceedings of the IEEE Conference on Robotics and Automation (ICRA-85)*, (pp. 116–121).

Moravec, H. P. (1996). Robot spatial perception by stereoscopic vision and 3D evidence grids. *Tech. Rep. CMU-RI-TR-96-34*.

Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the Society of Industrial and Applied Mathematics*, 5(1):32–38.

Murphy, K. (2000). Bayesian map learning in dynamic environments. In S. A. Solla, T. K. Leen, & K.-R. Müller (eds.), *Advances in Neural Information Processing Systems 12*, (pp. 1015–1021). The MIT Press.

Mutambara, A. (1998). *Decentralized Estimation and Control for Multisensor Systems*. USA: CRC Press.

Nagatani, K. & Choset, H. (1999). Toward robust sensor based exploration by constructing reduced generalized Voronoi graph. In *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-99)*, (pp. 1687–1692).

Nagatani, K., Choset, H., & Thrun, S. (1998). Towards exact localization without explicit localization with the generalized Voronoi graph. In *Proceedings 1998 IEEE International Conference on Robotics and Automation (ICRA-98)*, (pp. 342–348).

Neira, J. & Tardós, J. D. (2001). Data association in stochastic mapping using the joint compability test. *IEEE Transactions on Robotics and Automation*, 17:890–897.

Newman, P. M. & Leonard, J. J. (2003). Pure range-only subsea SLAM. In *IEEE International Conference on Robotics and Automation (ICRA-03)*, (pp. 1921–1926).

Nieto, J., Guivant, J., Nebot, E., & Thrun, S. (2003). Real time data association for FastSLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA-03)*, (pp. 412–418).

Nüchter, A., Surmann, H., Lingemann, K., Hertzberg, J., & Thrun, S. (2004). 6D SLAM with an application in autonomous mine mapping. In *Proceedings IEEE 2004 International Conference Robotics and Automation (ICRA-04)*, (pp. 1998–2003).

Nüchter, A., Lingemann, K., Hertzberg, J., & Surmann, H. (2005). 6D SLAM with approximate data association. In *Proceedings of the 12th International Conference on Advanced Robotics (ICAR-05)*, (pp. 242 – 249).

Ogniewicz, R. L. & Kübler, O. (1995). Hierarchic Voronoi Skeletons. *Pattern Recognition*, 28(3):343–359.

Okabe, A., Sugihara, K., Chiu, S. N., & Boots, B. (2000). *Spatial Tessellations – Concepts and Applications of Voronoi Diagrams*. John Wiley and Sons.

O'Keefe, J. & Nadel, L. (1978). *The Hippocampus as a Cognitive Map*. Oxford University Press.

Palmer, S. E. (1978). Fundamental aspects of cognitive representation. In E. Rosch & B. Lloyd (eds.), *Cognition and Categorization*, (pp. 259–303). Hillsdale, NJ: Erlbaum.

Paskin, M. A. (2003). Thin junction tree filters for simultaneous localization and mapping. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, (pp. 1157–1166).

Pfeifer, R. & Bongard, J. C. (2007). *How the Body Shapes the Way We Think: A New View of Intelligence*. Cambridge, MA: MIT Press.

Randell, D. A., Cui, Z., & Cohn, A. (1992). A spatial logic based on regions and connection. In B. Nebel, C. Rich, & W. Swartout (eds.), *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR'92)*, (pp. 165–176). San Mateo, CA: Morgan Kaufmann.

Reid, D. (1979). An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6):843–854.

Rekleitis, I., Dujmovic, V., & Dudek, G. (1999). Efficient topological exploration. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (ICRA-99)*, (pp. 676–681).

Remolina, E. & Kuipers, B. (2004). Towards a general theory of topological maps. *Artificial Intelligence*, 152(1):47–104.

Remolina, E., Fernández, J. A., Kuipers, B., & González, J. (1999). Formalizing regions in the spatial semantic hierarchy: An AH-graphs implementation approach. In *Proceedings of the International Conference on Spatial Information Theory: Cognitive and Computational Foundations of Geographic Information Science*, vol. 1661, (pp. 109–124).

Renz, J. & Ligozat, G. (2005). Weak composition for qualitative spatial and temporal reasoning. In *Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming (CP 2005)*, (pp. 534–548).

Renz, J. & Mitra, D. (2004). Qualitative direction calculi with arbitrary granularity. In C. Zhang, H. W. Guesgen, & W.-K. Yeap (eds.), *PRICAI 2004: Trends in Artificial Intelligence, 8th Pacific Rim International Conference on Artificial Intelligence, Auckland, New Zealand, Proceedings*, vol. 3157 of *Lecture Notes in Computer Science*, (pp. 65–74). Springer.

Renz, J. & Nebel, B. (1999). On the complexity of qualitative spatial reasoning: A maximal tractable fragment of the region connection calculus. *Artificial Intelligence*, 108(1-2):69–123.

Richter, K.-F. (2007). A uniform handling of different landmark types in route directions. In S. Winter, M. Duckham, L. Kulik, & B. Kuipers (eds.), *Spatial Information Theory (COSIT)*, vol. 4736 of *Lecture Notes in Computer Science*, (pp. 373–389). Berlin: Springer.

Samet, H. (1988). An overview of quadtrees, octrees and related hierarchical data structures. In *NATO ASI Series, Vol. F40, Theoretical Foundations of Computer Graphics*, (pp. 51–68). Springer-Verlag Berlin Heidelberg.

Sanfeliu, A. & Fu, K. (1983). A distance measure between attributed relational graph. *IEEE Transactions on Systems, Man and Cybernetics*, 13:353–362.

Savelli, F. & Kuipers, B. (2004). Loop-closing and planarity in topological map-building. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-04)*, (pp. 1511–1517).

Schlieder, C. (1993). Representing visible locations for qualitative navigation. In N. P. Carreté & M. G. Singh (eds.), *Qualitative Reasoning and Decision Technologies*, (pp. 523–532).

Schlieder, C. (1995). Reasoning about ordering. In A. U. Frank & W. Kuhn (eds.), *Spatial Information Theory – A Theoretical Basis for GIS*, (pp. 341–349).

Schölkopf, B. & Mallot, H. (1995). View-based cognitive mapping and path planning. *Adaptive Behavior*, 3(3):311–348.

Sebastian, T. B., Klein, P. N., & Kimia, B. B. (2004). Recognition of shapes by editing their shock graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence,*, 26(5):550–571.

Shi, H., Mandel, C., & Ross, R. J. (2007). Interpreting route instructions as qualitative spatial actions. In T. Barkowsky, M. Knauff, G. Ligozat, & D. Montello (eds.), *Spatial Cognition V - Reasoning, Action, Interaction*, vol. 4387 of *Lecture Notes in Artificial Intelligence*, (pp. 327–345). Springer.

Siddiqi, K. & Kimia, B. B. (1996). A shock grammar for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 507–513).

Siegel, A. & White, S. (1975). The development of spatial representations of large-scale environments. In H. Reese (ed.), *Advances in Child Development and Behavior*, vol. 10, (pp. 9–55). Academic Press.

Simhon, S. & Dudek, G. (1998). A global topological map formed by local metric maps. In *Proceedings 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-98). Innovations in Theory, Practice and Applications*, (pp. 1708–1714).

Smith, C. M. & Leonard, J. J. (1997). A multiple hypothesis approach to concurrent mapping and localization for autonomous underwater vehicles. In *Proceedings of International Conference on Field and Service Robotics*.

Smith, R. C. & Cheeseman, P. (1986). On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research*, 5(4):56–68.

Smithson, M. (1989). *Ignorance and Uncertainty: Emerging Paradigms*. New York: Springer.

Sorrows, M. E. & Hirtle, S. C. (1999). The nature of landmarks for real and electronic spaces. In C. Freksa & D. M. Mark (eds.), *Spatial Information Theory. Cognitive and Computational Foundations of Geopraphic Information Science (COSIT)*, vol. 1661 of *Lecture Notes on Computer Science*, (pp. 37–50). Berlin: Springer.

Stachniss, C. & Burgard, W. (2003a). Exploring unknown environments with mobile robots using coverage maps. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-03)*, (pp. 1127–1132).

Stachniss, C. & Burgard, W. (2003b). Mapping and exploration with mobile robots using coverage maps. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-03)*, (pp. 476–481).

Stachniss, C., Hähnel, D., Burgard, W., & Grisetti, G. (2005). On actively closing loops in grid-based FastSLAM. *Advanced Robotics – The International Journal of the Robotics Society of Japan (RSJ)*, 19(10):1059–1080.

Tardós, J., Neira, J., Newman, P., & Leonard, J. (2002). Robust mapping and localization in indoor environments using sonar data. *International Journal of Robotics Research*, 21(4):311–330.

Thrun, S. (1998). Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71.

Thrun, S. (2000). Probabilistic algorithms in robotics. *AI Magazine*, 21(4):93–109.

Thrun, S. (2002). Robotic mapping: A survey. *Tech. rep.*, Carnegie Mellon University, Computer Science Department.

Thrun, S., Bücken, A., Burgard, W., Fox, D., Fröhlinghaus, T., Hennig, D., Hofmann, T., Krell, M., & Schmidt, T. (1998a). Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R. Bonasso, & R. Murphy (eds.), *AI-based Mobile Robots: Case Studies of Successful Robot Systems*. MIT Press, Cambridge, MA.

Thrun, S., Burgard, W., & Fox, D. (1998b). A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, 31(1-3):29–53.

Thrun, S., Koller, D., Ghahramani, Z., Durrant-Whyte, H., & Ng, A. (2002). Simultaneous mapping and localization with sparse extended information filters. In J.-D. Boissonnat, J. Burdick, K. Goldberg, & S. Hutchinson (eds.), *Proceedings of the Fifth International Workshop on Algorithmic Foundations of Robotics*.

Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic Robotics*. MIT Press.

Tolman, E. C. (1948). Cognitive maps in rats and men. *The Psychological Review*, 55(4):189–208.

Trullier, O., Wiener, S. I., Berthoz, A., & Meyer, J.-A. (1997). Biologically-based artificial navigation systems: Review and prospects. *Progress in Neurobiology*, 51:483–544.

Tversky, B. (1992). Distortions in cognitive maps. *Geoforum*, 23:131–138.

Tversky, B. (1993). Cognitive maps, cognitive collages and spatial mental models. In A. Frank & I. Campari (eds.), *Spatial Information Theory: A Theoretical Basis for GIS – Proceedings of COSIT'93*, (pp. 14–24). Berlin: Springer.

Varela, F. J., Thompson, E. T., & Rosch, E. (1992). *The Embodied Mind: Cognitive Science and Human Experience*. The MIT Press.

Wallgrün, J. O. (2002). Exploration und Pfadplanung für mobile Roboter basierend auf Generalisierten Voronoi-Graphen. *Diplomarbeit*, Fachbereich Informatik, Universität Hamburg.

Wallgrün, J. O., Frommberger, L., Dylla, F., & Wolter, D. (2006). SparQ user manual v0.6. *Tech. Rep. 007-07/2006*, SFB/TR 8 Spatial Cognition, Universität Bremen.

Wallgrün, J. O., Frommberger, L., Wolter, D., Dylla, F., & Freksa, C. (2007). Qualitative spatial representation and reasoning in the SparQ-toolbox. In T. Barkowsky, M. Knauff, G. Ligozat, & D. Montello (eds.), *Spatial Cognition V: Reasoning, Action, Interaction: International Conference Spatial Cognition 2006*, vol. 4387 of *Lecture Notes in Computer Science*, (pp. 39–58). Springer Berlin Heidelberg.

Werner, S., Krieg-Brückner, B., & Herrmann, T. (2000). Modelling navigational knowledge by route graphs. In C. Freksa, C. Habel, W. Brauer, & K. F. Wender (eds.), *Spatial Cognition II – Integrating Abstract Theories, Empirical Studies, Formal Methods, and Pratical Applications*, vol. 1849 of *LNCS,LNAI*, (pp. 295–316). Springer.

Williams, S. B., Newman, P. M., Rosenblatt, J., Dissanayake, G., & Durrant-Whyte, H. F. (2001). Autonomous underwater navigation and control. *Robotica*, 19(5):481–496.

Wolter, D. (2008). *Spatial Representation and Reasoning for Robot Mapping – A Shape-Based Approach*, vol. 48 of *Springer Tracts in Advanced Robotics*. Springer Berlin/Heidelberg.

Wolter, D. & Richter, K.-F. (2004). Schematized aspect maps for robot guidance. In *Proceedings of the ECAI Workshop Cognitive Robotics (CogRob)*.

Wolter, D., Latecki, L. J., Lakämper, R., & Sun, X. (2004). Shape-based robot mapping. In *Proceedings of the 27th German Conference on Artificial Intelligence (KI-2004)*, (pp. 439–452).

Yamauchi, B. (1997). A frontier-based approach for autonomous exploration. In *Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, (p. 146).

Yamauchi, B. & Beer, R. (1996). Spatial learning for navigation in dynamic environments. *IEEE Transactions on Systems, Man and Cybernetics*, 26(3):496–505.

Yeap, W. K. & Jefferies, M. E. (1999). Computing a representation of the local environment. *Aritificial Intelligence*, 107:265–301.

Zelinsky, A. (1992). A mobile robot exploration algorithm. *IEEE Transactions on Robotics and Automation*, 8(6):707–717.

Zhang, K. & Shasha, D. (1989). Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262.

Zhang, K., Statman, R., & Shasha, D. (1992). On the editing distance between unordered labeled trees. *Information Processing Letters*, 42(3):133–139.