Copyright

by

Patrick Foil Beeson

2008

The Dissertation Committee for Patrick Foil Beeson
certifies that this is the approved version of the following dissertation:

Creating and Utilizing Symbolic Representations of Spatial Knowledge using Mobile Robots

Committee:

Benjamin J. Kuipers, Supervisor

Dana Ballard

Gregory Dudek

Raymond Mooney

Brian Stankiewicz

Peter Stone

Creating and Utilizing Symbolic Representations of Spatial Knowledge using Mobile Robots

by

Patrick Foil Beeson, B.S.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August 2008

To all UTCS graduates that have held the Golden Turtle.

"The early bird may get the worm, but it's the second mouse that gets the cheese." –Steven Wright

Acknowledgments

I wish to thank the multitudes of people who helped me navigate this portion of my life.

Having Ben Kuipers as an advisor could not have worked out better for me. His boundless faith in the scientific process and constant optimism in the face of adversity was exactly what a pessimistic, sarcastic student needed to persevere. Rather than forcing me into a research area that is fashionable, he allowed me the time and space to explore the topics I felt were important. As a result, I believe that this thesis is an important contribution towards finding a robust solution to the robot mapping problem.

My thesis committee is comprised of some of the smartest and nicest people I have ever met. Peter Stone, Brian Stankiewicz, and Ray Mooney have been a great source of intellectual stimulation, and have each helped to shape my education. In my few discussions with Greg Dudek, he has certainly lived up to his reputation as one of the most forward-thinking robotics researchers in the world. Dana Ballard came to the rescue by agreeing to attend my dissertation defense in order to make a quorum. I am truly lucky to have such a great group of people on my committee.

My colleagues in the AI lab deserve loads of praise, as they were the ones that had to put up with me on a daily basis. Joseph Modayil, Jeff Provost, Matt MacMahon, and Aniket Murarka have logged countless hours discussing robot and human navigation, reading paper drafts, and humoring me by laughing at my constant jokes. I cannot forget to mention Francesco Savelli, (Ram) Subramanian Ramamoorthy, Harold Chaput, Misha Bilenko, Tal Tversky, Nick Jong, and Michael Quinlan, all of whom have been cohorts and comrades at UT-Austin.

Additional thanks go to my group of non-CS friends here in Austin. Without their alcohol-fueled taunts and insults about still being a student in my thirties, I may have stayed in school forever. I especially cannot forget to mention my beautiful and inspiring love, Denise. Her support keeps me going when things seem at their worst. She is my best friend, and I cannot imagine life without her.

Finally, I would like to thank my family, who all wonder what exactly I have been doing for the last 9 years, for their patience. My mother did not live to see me finish my undergraduate studies, but she was aware that I had decided to pursue a doctorate degree. I know she would be so proud to see this goal finally accomplished.

PATRICK BEESON

The University of Texas at Austin
August 2008

Creating and Utilizing Symbolic Representations of Spatial Knowledge using Mobile Robots

Publication No. ______________________

Patrick Foil Beeson, Ph.D.
The University of Texas at Austin, 2008

Supervisor: Benjamin J. Kuipers

A map is a description of an environment allowing an agent—a human, or in our case a mobile robot—to plan and perform effective actions. From a single location, an agent's sensors can not observe the whole structure of a complex, large environment. For this reason, the agent must build a map from observations gathered over time and space. We distinguish between large-scale space, with spatial structure larger than the agent's sensory horizon, and small-scale space, with structure within the sensory horizon.

We propose a factored approach to mobile robot map-building that handles qualitatively different types of uncertainty by combining the strengths of topological and metrical approaches. Our framework is based on a computational model of the human cognitive map; thus it allows robust navigation and communication within several different spatial ontologies. Our approach factors the mapping problem into natural sub-goals: building a metrical representation for local small-scale spaces; finding a topological map that represents the qualitative structure of large-scale space; and (when necessary) constructing a metrical representation for large-scale space using the skeleton provided by the topological map.

The core contributions of this thesis are a formal description of the Hybrid Spatial Semantic Hierarchy (HSSH), a framework for both small-scale and large-scale representations of space, and an implementation of the HSSH that allows a robot to ground the large-scale concepts of place and path in a metrical model of the local surround. Given metrical models of the robot's local surround, we argue that places at decision points in the world can be grounded by the use of a primitive called a gateway. Gateways separate different regions in space and have a natural description at intersections and in doorways. We provide an algorithmic definition of gateways, a theory of how they contribute to the description of paths and places, and practical uses of gateways in spatial mapping and learning.

Contents

Acknowledgments v

Abstract vii

Contents ix

List of Algorithms xiv

List of Tables xv

List of Figures xvi

Chapter 1 Introduction 1

- 1.1 Statement of the Problem 3
- 1.2 Thesis Overview 4
- 1.3 Major Contributions 9

Chapter 2 Background and Related Work 11

- 2.1 Cognitive Maps 11
  - 2.1.1 Human Spatial Representations 12
  - 2.1.2 Structural versus Object Landmarks 12
  - 2.1.3 Relevance of Cognitive Maps to this Thesis 13
- 2.2 Robot Maps 14

2.2.1 Metrical Maps 15

2.2.2 Relevance of Metrical SLAM to this Thesis 20

2.2.3 Topological Maps 21

2.2.4 Spatial Semantic Hierarchy Theses 28

2.2.5 Relevance of Topological Maps to this Thesis 30

2.2.6 Hybrid Maps 31

2.2.7 Relevance of Hybrid Mapping to this Thesis 32

2.3 Gateways 34

2.3.1 The Foundations of Gateways 34

2.3.2 Gateway implementations 36

Chapter 3 The Spatial Semantic Hierarchy **44**

3.1 Hybrid SSH Overview 44

3.2 The Basic SSH 47

3.2.1 The SSH Control Level 49

3.2.2 The SSH Causal Level 50

3.2.3 The SSH Topological Level 54

3.2.4 The SSH Metrical Level 57

3.3 Extending the SSH 59

Chapter 4 Modeling Obstacles and Trajectory: The HSSH Local Metrical Level **62**

4.1 Local Perceptual Map (LPM) 63

4.2 LPM Benefits 65

4.3 Building an Accurate Localization Algorithm 66

4.3.1 Action Models 67

4.3.2 Adaptive Particle Filter Localization 72

Chapter 5 Symbol Grounding:
The HSSH Local Topology Level 91
5.1 Formal Description of Local Topology 91
5.2 Grounding Local Topology in the Local Perceptual Map 94
  5.2.1 Gateways 94
  5.2.2 Local Topology Creation 95
5.3 Detecting Places 96
  5.3.1 Place Criteria 98
  5.3.2 Places from Local Topology 100
  5.3.3 Gateways versus Voronoi Junctions 102
5.4 Selecting Local Motion Targets 104

Chapter 6 Implementing Gateways 107
6.1 Algorithm Overview 108
  6.1.1 The Voronoi Skeleton of an LPM 110
  6.1.2 Defining the LPM Core 116
6.2 Constriction-based Gateways 116
  6.2.1 Discussion 123
  6.2.2 Problems with the Constriction Algorithm 125
6.3 Current Gateway Algorithm 128
  6.3.1 The Extended Voronoi Graph (EVG) 130
  6.3.2 Anchor-based Gateways 133
  6.3.3 Evaluating Anchor-based Gateways 145

Chapter 7 Human-Robot Interaction:
Evaluating the HSSH Local Levels 168
7.1 HSSH Interface for a Mobility Assistant 169
  7.1.1 Local Metrical Control 169

7.1.2 Local Topological Control 172
7.1.3 Global Topological and Metrical Control 173
7.2 Empirical Evaluation 174
7.2.1 Experimental Setup 175
7.2.2 Results 177

Chapter 8 Closing Loops:
The HSSH Global Topology Level **182**
8.1 From Small-Scale to Large-Scale Star 183
8.2 The Tree of Possible Topological Maps 185
8.3 Topological Mapping Example 188
8.4 Levels of Spatial and Temporal Granularity 191

Chapter 9 Drawing Maps:
The HSSH Global Metrical Level **193**
9.1 Terminology 194
9.2 The Theory of the Global Metrical Map 195
9.3 Global Mapping Example 197
9.3.1 Estimating $F(\lambda)$ 197
9.3.2 Estimating $G(\chi)$ 198
9.3.3 Estimating $H(x)$ 201
9.3.4 Creating a map m^* 203

Chapter 10 Conclusion **204**
10.1 Future Work 205
10.1.1 The Local Metrical and Topology Levels 205
10.1.2 The Global Topology and Metrical Levels 208
10.2 Summary 211

Bibliography 213

Vita 230

List of Algorithms

3.1 Building the tree of topological maps in the SSH 58

6.1 Finding the core of an LPM from a pruned free space skeleton 118
6.2 Finding constrictions of an LPM. 120
6.3 Finding constriction-based gateways 121
6.4 Finding anchors of an LPM 136
6.5 Finding anchor-based gateways 139

8.1 Building the tree of topological maps in the HSSH 187

List of Tables

2.1 Comparison of gateway implementations 43

4.1 Action model parameters 86
4.2 Localization accuracy of 5 different algorithms 89

7.1 User interface to the HSSH 171

List of Figures

2.1 Closing large loops 17

2.2 Incorrectly closing loops using odometry and scan matching 18

2.3 Voronoi graph of a global metrical map 26

2.4 Detecting places with an offset path 29

3.1 The Spatial Semantic Hierarchy 48

3.2 SSH Control/Causal Level abstraction 51

3.3 Topological abduction example 56

3.4 HSSH description 61

4.1 Markov localization 65

4.2 Generalized action model 71

4.3 Proposal distribution example: uncertainty in motion 73

4.4 Likelihood weighting example: uncertainty in observations 74

4.5 Localization failure example: wheel slippage 75

4.6 Poor localization due to an overestimating proposal 78

4.7 Improved localization using the `shrink` algorithm 79

4.8 Poor localization due to an underestimating proposal 81

4.9 Improved localization using the `grow` algorithm 82

4.10 Poor localization due to an offset proposal 83

4.11 Improved localization using the shrink-and-grow algorithm 84

5.1 Identifying gateways and local topology in the LPM 97
5.2 Gateway examples 101
5.3 The reduced generalized Voronoi graph (RGVG). 103
5.4 Grounding control using gateways 105

6.1 Overview of gateway detection process 109
6.2 Thinned skeleton of a global metrical map 111
6.3 Finding the skeleton of LPM free space 113
6.4 Pruning a Voronoi graph using the LPM boundary 117
6.5 Examples of the *base points* and the *core* of different local environments . . 119
6.6 Defining the search space for constriction-based gateways 122
6.7 Defining gateways by connecting nearby obstacles 124
6.8 Examples of constriction-based gateways in a variety of environments . . . 126
6.9 Unstable gateways near L intersections 127
6.10 Unstable gateways affect control 128
6.11 Unstable gateways in coastal navigation 129
6.12 Entering a room that is larger than the LPM 132
6.13 The extended Voronoi graph (EVG) of a global metrical map 133
6.14 The thinning approximation of the EVG 134
6.15 Intuition behind skeletal anchors 135
6.16 Searching for anchors 137
6.17 Anchors and gateways in coastal navigation scenarios 137
6.18 Finding gateway location hypotheses from anchors 140
6.19 Examples of anchor-based gateways in a variety of environments 141
6.20 Stable anchor-based gateways near L intersections 142
6.21 Stable anchor-based gateways at coastal L intersections 142

6.22 Handling environments without good anchors 144
6.23 Gateway (and place) detection in a university hallway 146
6.24 Traversing paths in a university hallway 147
6.25 Detecting doorways in a university hallway 148
6.26 Dealing with furniture 149
6.27 Gateway (and place) detection using a simulated robot 151
6.28 Gateway (and place) detection using a simulated robot 152
6.29 Gateway evaluation conditions 154
6.30 Impact of rotations on gateways and local topology 157
6.31 Variance of local topology classification across rotations 158
6.32 Time performance of local topology abstraction 159
6.33 Stability of gateway orientations 160
6.34 Impact of LPM resolution on gateways and local topology 163
6.35 Impact of noise on gateways and local topology 165
6.36 Determining the bounds of successful local topology abstraction 167

7.1 HSSH interface 170
7.2 Stimuli, map, and interface for the HRI experiment 176
7.3 Example experimental runs 178
7.4 HRI experiment results 179

8.1 Binding local-paths to distinctive states 184
8.2 Finding the topological map an environment with multiple nested loops . . 189

9.1 Defining local frames of reference 195
9.2 Finding the global place layout 199
9.3 Creating a global metrical map 200

Chapter 1

Introduction

> *Congratulations!*
> *Today is your day.*
> *You're off to Great Places!*
> *You're off and away!*
> Dr. Seuss, *Oh, the Places You'll Go!* (1990)

Intelligent machines have already begun to change our society. They have replaced human workers in dangerous and/or repetitive tasks such as manufacturing or mining operations. They have allowed exploration and scientific discovery of previously unseen worlds like Mars and the ocean floor. They have even started to become consumer products, such as toys, vacuums, and lawn mowers. In these domains, however, either the environmental complexity or the autonomous control of the machines is still quite limited.

As the trend of automation continues, many scientists and engineers envision a future where intelligent robots take on more autonomy, interacting with humans in our everyday lives by taking on duties like health care for the elderly or chauffeuring people across town. Similarly, there are initiatives for human-robot teams to continue exploration of outer space and maintain long-term off-world settlements. There is still a large gap between the kind of intelligence needed on a factory floor and the intelligence we expect to have when deploying autonomous robots in homes, hospitals, or space stations. Specifically, autonomous robots will need to handle communication with humans and to interact with environments that may change over time.

One important substrate of human intelligence is spatial reasoning. We rely on our understanding of space to move in the world, to make plans, to communicate with others, and to bootstrap higher-level concepts that seem to make us intelligent beings [Lakoff and Johnson, 1980]. Although psychologists do not agree exactly how human *cognitive maps* are built, maintained, and used, we know that cognitive maps are robust to uncertainty, often qualitative in nature, and inherently hierarchical [Kitchin, 1994]. Human spatial reasoning is also fundamentally related to memorization, which is necessary for building intellect from personal experiences.[1]

Mobile robots also need to understand space, but most robotics research has not been motivated by robust, qualitative human spatial reasoning. Instead most research on mobile robot navigation has been motivated by the characteristics of modern-day sensors. These solutions focus on building and using accurate metrical maps, similar to blueprints, which are monolithic and non-symbolic in representation. Though these maps may look nice, and may be useful for real-time planning over short distances, they do not lend themselves to symbolic inference, such as verbal communication or well-studied AI knowledge representation paradigms.

Can we overcome the large differences between robot and human perceptual capabilities in order to bind symbolic spatial concepts, similar to the ones humans seem to have, to the "pixel-level" data that robots experience? That is, can we produce robots that have cognitive maps?

---

[1] Expositions on rhetoric dating back to 85 BCE discuss mnemonic devices where large speeches are remembered via association of talking points to physical locations in the world. To deliver a speech, a person simply has to take a tour in their mind through a well known place, like a market or town hall. Each memory of a place helps recall a stored portion of the speech. Cicero credits Simonides of Ceos (c. 556 BCE-468 BCE) with discovering this artificial memory technique [Yates, 1966]. This "method of loci" technique is still used today by world champion memorizers (often termed the "journey method"). These individuals associate long strings of random numbers or shuffled cards with detailed tours through their houses or neighborhoods.

1.1 Statement of the Problem

The animal kingdom has conquered the problem of reasoning about space by employing a variety of techniques. Many animals utilize very precise *path integration* (or *dead reckoning*) techniques, where they know the distance and direction they have traveled from home. This is often improved by using the sun as a compass, calibrating flight speed from known static landmarks, detecting the rotation direction of the stars, or using the earth's magnetic field. These navigation methods are remarkably accurate, but such animals can be easily fooled by mischievous scientists [Gallistel, 1990].

On the other hand, we know that humans (without the use of tools) tend to have fairly poor path integration skills over large-scale environments [Golledge, 1999]. Instead we tend to impose a symbolic structure on the world, utilizing unique landmarks or imposing an artificial structure on the environment, to facilitate graph-like planning between *places* in the world connected by *paths*. Thus in normal, everyday navigation, we rely much more heavily on internal models of space, rather than external oracles, like the sun or stars. Remarkably even animals thought to posses unparalleled dead reckoning abilities sometimes prefer following longer, human-like routes, either along man-made highway networks [Guilford et al., 2004] or along routes they define themselves with artificial landmarks [Stopka and Macdonald, 2003], though scientists do not yet fully understand the reasons behind these behaviors.

The human ability to abstract the 3D world into a symbolic representation has several benefits. It allows humans to remember, plan in, and move across relatively large regions with respect to our size. It also facilitates hierarchical representations that allow us to think about space at a variety of levels, each with different types and amounts of information. From an AI perspective, symbolic abstraction facilitates learning, inference, and knowledge acquisition. Maybe most important, a symbolic representation of space allows communication of directions and description of objects in the world, either through verbal means or via drawing [MacMahon, 2007].

A key challenge from a robotics point of view is how to encode human-style spatial reasoning into a machine. This is especially difficult given that current robotic sensors are so dissimilar to human perception. On the other hand, implementing robust path integration on mobile robots has become quite trivial—especially given modern hardware that tracks global position changes, e.g., inertial measurements units and differential GPS. This precise localization, along with devices that provide precise distance measurements to nearby obstacles, like laser radar (lidar), allows robots to generate metrical models of the immediate surround using a variety of well-understood techniques [Thrun et al., 2005].

The problem we have set out to investigate in this dissertation is how to abstract symbolic spatial concepts from these metrical models in a way that facilitates robust navigational behavior while supporting symbolic topological reasoning (e.g., abductive and non-monotonic reasoning) and effective communication with human users (e.g., following symbolic route instructions). Can we bridge the gap between modern-day metrical map-building and human-inspired, symbolic concepts of space? We argue that this is not only possible, but that we can ground our symbolic spatial abstraction via a geometrically derived primitive, called a *gateway*.

1.2 Thesis Overview

For decades, AI research has moved forward on the assumption that symbolic primitives can be used for learning, inference, and planning. These symbols are often assumed to relate to some real-world knowledge, but the hard problem of *symbol grounding* [Harnad, 1990], is often ignored or left as future work. Robotics research in particular cannot overlook symbol grounding, as the act of deploying a physical agent in the real world requires some form of grounding discrete symbols to low-level perceptual inputs. There are two predominant paradigms in robot navigation research that attack the symbol grounding problem from different perspectives.

For many years, robotics researchers took a classic GOFAI approach.[2] These scientists believed that graph-like representations of space, such as *topological* networks of places and paths, could provide a low-level symbolic abstraction that would facilitate planning and may one day be useful for learning, communication, and hierarchical knowledge acquisition. This early work suffered from a lack of reliable sensors; thus, the abstraction of symbols from low-level inputs was often done using ad hoc techniques that were robot and environment dependent. As a result, much of this work concentrated on theoretical issues dealing with graph exploration, rather than on how the graph is actually abstracted from the continuous environment.

Over the years, more precise measuring devices became affordable and portable enough to integrate onto research robots. The improvement of global odometry tracking, along with precise distance measurements to nearby obstacles, yielded research that focused on building very precise *metrical* models of the location of obstacles in the world. These methods were so successful in building impressive looking maps, that much of the current-day robotics research is focused around extending the same techniques to larger and larger environments; this is despite the fact that these methods do not scale as well and have little useful semantic information. Another problem with these methods is that the robot's knowledge of the world is quite separated from the human user. Although humans can examine these maps for correctness, the robot itself has only a very primitive spatial understanding of the world: what locations in the world contain obstacles.

This thesis examines the use of human-like symbolic representations in mobile robotics. We use psychological evidence to *motivate* a theory of hybrid map-building that uses the sensor-inspired advances along with symbolic spatial reasoning. This thesis shows the first attempt to fully close the loop between local metrical sensations, symbolic topological inference, and global metrical map production. In particular, we use well-known metrical mapping techniques to ground symbolic, topological representations of space. In

---

[2] Haugeland [1985] is credited with coining the term "Good Old-Fashioned Artificial Intelligence".

doing so we model the 3D world with a 2D metrical representation. Next, we abstract the local metrical model into a symbolic model of local places and paths. Local symbolic models are combined into a graph-like network of connected decision points. A global metrical map can be built efficiently on top of the symbolic topological model if desired, but global maps are unnecessary for robot navigation once a correct topological model has been determined.

The core contribution of this thesis is to ground the concepts of place and path in a straightforward metrical model of the local surround. This metrical model is currently obtained by range-sensing, but we believe (1) that such models can soon be obtained by other techniques, such as using machine vision [Sim and Little, 2006; Murarka et al., 2006] and (2) the hybrid, hierarchical framework is independent of the sensors used to create the local metrical models. Given metrical models of the robot's local surround, we argue that places at decision points in the world can be grounded by the use of a primitive called a gateway. Gateways separate qualitatively different regions of space and have a natural description at intersections and in doorways. We provide analysis of multiple gateway algorithms, a theory of how they contribute to the description of paths and places, and examples of the benefits of gateways in spatial mapping and learning.

Additionally, we introduce a formal description of our hybrid approach to spatial reasoning, which grounds gateways (and consequently places and paths) from models of *small-scale space*. This formal description is crafted in such a way as to allows us to continue using the logical topological map-building axioms of Remolina and Kuipers [2004], while improving the computational complexity of map-building and the aesthetics of small-scale and large-scale motion through the environment. We illustrate our initial implementation of hybrid map-building and show successful human-robot interaction in a pilot experiment. We then discuss benefits of this hybrid approach to the global metrical mapping community, and propose how global metrical maps should in turn benefit the topological mapping community in the near future. Finally we discuss some of the new scientific problems that arise from our research.

Chapter 2 overviews background and related work. We discuss work on the human cognitive map that has motivated the kinds of knowledge we want our robots to obtain. We then give a summary of the recent history of mobile robotics research, as a contrast to how humans represent space. We then discuss related work on gateways, explaining their theoretical origins and detailing previous implementations. The various drawbacks of previously published gateway implementations motivates our current gateway implementation.

In Chapter 3, we describe the Spatial Semantic Hierarchy (SSH), a framework for hierarchical abstraction of spatial knowledge from experience. The SSH is motivated by the human cognitive map, but remains fairly agnostic about the type of sensations the agent may posses. This motivates the need for the *Hybrid* Spatial Semantic Hierarchy (HSSH), which directly addresses modern robotic sensors, by explicitly addressing the interactions between small-scale space and large-scale space. Here, we introduce the HSSH framework, including the concepts that are detailed in Chapters 4 through 9.

Chapter 4 discusses local mapping and localization for a particular class of robots that can use vision or range-sensing devices to create metrical descriptions of the local surrounding region. Local perceptual maps (LPMs), which model local, small-scale space are useful both for robot control and for grounding symbols in the recent experience of the robot. We discuss improvements to the traditional incremental localization schemes in order to ensure high-quality local metrical models [Beeson et al., 2006]. Reducing localization failures helps to ensure high-quality local metrical models, which in turn ensure correct place/path detection, efficient place recognition, and accurate global layouts in the higher levels of the HSSH.

Chapter 5 delves into the symbol grounding problem. We begin a formal description of the HSSH, from local metrical models through local topological hypotheses, that extends previous SSH formal descriptions [Remolina, 2001; Remolina and Kuipers, 2004; Savelli, 2005]. We focus on gateways and local topology representations of space, which are used as primitives for the detection, description, and navigation of topological places and paths. We also discuss using gateways as local motion targets to ground large-scale travel actions in the LPM.

Chapter 6 discusses particular implementation details of gateways. We detail an initial method for defining robust, stable gateways from a Voronoi skeleton that works well in corridor-type environments [Beeson et al., 2005]. We illustrate scenarios where this approach becomes unreliable and introduce a new algorithm that overcomes these issues while extending gateways to non-corridor environments. Gateways need to be fairly stable in order to provide a robust local topology of places in the world, so we evaluate the reliability of the current gateway algorithm across different types of noise.

In Chapter 7, we validate the HSSH approach with a human-robot interaction experiment. We introduce an interface for the HSSH in a mobility-assistance domain, and we test this interface at the HSSH Local Metrical and Local Topology Levels [Beeson et al., 2007]. We validate the HSSH implementation and the initial human-robot interface with an experiment that simulates visually impaired drivers of an intelligent personal transport. We compare navigation performance with and without the use of local metrical and symbol models of small-scale space.

Chapter 8 returns to the HSSH formal description; this time detailing global, symbolic models of space. We discuss closing loops in topological map-building, and detail the computational improvement of the HSSH approach over the basic SSH. Specifically, we show that gateways and the resulting local topology provides a *deterministic* description of decision points in the world, which can eliminate a large number of invalid topological hypotheses [Beeson et al., 2003; Kuipers et al., 2004].

In Chapter 9, we discuss how to utilize the local metrical data experienced during travel between places to synthesize a metrically accurate global map. This process uses the topological map as a skeleton. The topological places can be relaxed in a global metrical frame of reference *prior* to the full metrical map being hypothesized [Modayil, Beeson, and

Kuipers, 2004]. This approach scales efficiently with environment size and complexity.

Chapter 10 summarizes the contributions of this dissertation and proposes future avenues of work. Particularly, we discuss how the global metrical layout, which is computationally efficient when using a topological skeleton, can feed back into the search for the correct topological map. Issues of active exploration, visual LPMs, and learning place recognition are also discussed.

1.3 Major Contributions

There are several key contributions of this dissertation that span multiple disciplines. In the area of cognitive science, we present a formal description of a spatial reasoning architecture inspired by psychological evidence. This framework uses both small-scale and large-scale frames of reference and both metrical and symbolic representations of space.

We improve localization techniques from the robotic metrical mapping (SLAM) domain to ensure high-quality local metrical models of the nearby surround. We contribute an efficient, online method for global metrical map-building, useful in surveying, mining and other applications, by utilizing the learned symbolic topological map as a skeleton for the global layout.

We introduce a novel algorithm for determining gateways, which separate qualitatively different portions of environment based on local structure. We tackle a specific AI symbol grounding problem in the robotics domain by implementing place detection and place description that improves on the state-of-the-art. Experimental validation of the hybrid metrical/symbolic navigation in a wheelchair domain is relevant to the human-robot interaction (HRI) community.

This dissertation provides useful innovation in all the areas above. We attempt to innovate across the breadth of the hierarchical framework of spatial knowledge, rather than focusing one specific aspect; however, we believe the HSSH formal description, which is consistent with (yet improves upon) the basic SSH logical theory, along with the current implementation for determining gateways, which provides a variety of benefits at all levels of the SSH framework, are the core contributions of this thesis.

Chapter 2

Background and Related Work

> *Knowledge is of two kinds. We know a subject ourselves, or we know where we can find information on it.*
>
> Samuel Johnson, James Boswell's *Life of Johnson* (1791)

This chapter introduces the reader to the background information needed to understand both the motivations and the techniques used in our research. We begin with an overview of human cognitive map research. We then discuss recent history in the field of mobile robotics, detailing research concentrating on building extremely accurate and precise metrical representations of environments along with ideas about topological robot navigation as graph exploration. We finish with a discussion of gateways, which have been examined by several robotics researchers, and which we use as the anchoring primitive for grounding topological concepts into well-understood metrical models of the world.

2.1 Cognitive Maps

Tolman [1948] coined the term "cognitive map" to represent the mental spatial representation of an animal. Since this time, there has been much study on the way animals, especially humans, solve navigation problems in the world. Most important to this thesis is the research on human representations of *large-scale space*. Researchers from developmental psychology all the way to urban designers were all pioneers in the early work of human navigation [Piaget, 1954; Lynch, 1960; Appleyard, 1970].

2.1.1 Human Spatial Representations

Probably the most influential work on human acquisition and use of the cognitive map is by Siegel and White [1975]. This research notes that adult humans go through three stages when forming a cognitive map. First, *landmarks* define important locations in the world. Next, *routes* between landmarks are remembered. Finally, *configurations* unify route knowledge together into a more globally coherent representation. The authors remain fairly agnostic as to what exactly constitutes a landmark, other than a discussion of distinctiveness or novelty as a characteristic of landmarks. More recent work supports the notion that stability can also be an important characteristic of landmarks [Biegler and Morris, 1996].

This idea of a few distinctive, stable places, connected together by remembered routes, eventually leading to a cohesive connected network is the basis for several modern theories that support topological maps as reasonable models of the human cognitive map, including the framework we utilize in this dissertation. On the other hand, global, metrically precise maps have long been recognized as limited in their cognitive plausibility [Lynch, 1960; Downs and Stea, 1973; Kuipers, 1982]. In fact, Lynch [1960] found that errors in cognitive maps are rarely topological and frequently metrical. This is not to imply that topological maps cannot use metrical information, but that symbolic abstraction is a vital part of human spatial reasoning.

2.1.2 Structural versus Object Landmarks

Many researchers take Siegel and White's concept of "landmarks" to refer to object landmarks that facilitate homing-style navigation, which leads the human between beacons in an otherwise perceptually uniform environment. While this may be true in certain environments, most man-made environments, such as office complexes, campus walkways, street networks, and even hiking trails have a graph-like structure imposed on them. In these environments, there exist object landmarks, though there will be structural landmarks in addition to object beacons.

Specifically, *decision points* in environments are important, landmark locations when building a cognitive map, as they are both very stable and quite distinctive from a control point of view (new control opportunities arise at decision points).[1] Lynch [1960] was perhaps the first to present evidence that city dwellers abstract the world, at the finest granularity, into streets and their intersections. Urbanites use these graph-like networks to perform reliable navigation through the city. The idea of decision points (intersections) in man-made environments as “first-level” spatial concepts has been reiterated by other researchers [Golledge et al., 1985].

2.1.3 Relevance of Cognitive Maps to this Thesis

In our work, we are really concerned with detecting and describing places at decision points in the world. That is, we constrain our domains to structured, man-made environments that have natural graph-like connectivity. Environments like deserts, dense jungles, or oceans can be learned and navigated successfully by humans, but normally only after acquiring specific knowledge from experts, using external navigation tools, or utilizing detailed maps. We are much more interested in having robots build up the types of knowledge the majority of humans experience on a daily basis. Accordingly, we are currently not concerned with places that do not occur at decision points.

We do not deny important places exist along paths where no decision really exists. Places such as towers along roadways, water fountains in buildings, etc. are useful meeting points or checkpoints along routes. However we argue that this object landmark knowledge is actually secondary to knowledge of the structural connectivity of large-scale environments. Allen et al. [1979] performed an experiment where subjects were asked to pick the

---

[1] Later we will show that decision points are actually distinctive from a structural point of view as well, as *gateway* configurations change at decision points.

9 snapshots that would be best to help them remember the route captured by an ordered collection of 52 snapshots. Adults chose snapshots taken at path intersections 94% of the time. Appleyard [1969] found that mere visibility of a building is not sufficient for its recall, but proximity to an intersection virtually guarantees it. Gale et al. [1990] and later Aginsky et al. [1997] found that more visual information was remembered around intersections than between intersections along paths. Heft [1979] followed by Stankiewicz and Kalia [2007] showed that humans remember the structure of decision points better than object landmarks when exploring unfamiliar environments.

This thesis discusses integrating local metrical models built from robot experience, with large-scale symbolic models of space. We use these local models to detect and classify decision points in the world, but we also utilize the small-scale representations to improve control, exploration, and map-building efficiency.

> If the organism carries a "small-scale model" of external reality and of its own possible actions within its head, it is able to try out various alternatives, conclude which is the best of them, react to future situations before they arise, utilize the knowledge of past events in dealing with the present and future, and in every way to react in a much fuller, safer, and more competent manner to the emergencies which face it. [Craik, 1943, p. 61]

As a result, we are combining useful dead-reckoning and metrical environment sensing in local regions with symbolic reasoning about large-scale spatial knowledge.

2.2 Robot Maps

In modern day robotics, there are two main paradigms for modeling the spatial structure of the environment. One, topological mapping, attempts to discretize the continuous world as a graph-like representation of places connected by paths. The other, metrical mapping, tries to create a CAD-like layout of the location of all obstacles in the world. We first discuss metrical mapping, as it allows us to introduce common terminology and issues that arise in mobile robot navigation and map-building. We then discuss topological mapping, followed by a discussion of hybrid implementations that combine elements of both topological and metrical maps.

2.2.1 Metrical Maps

Topological maps (discussed below) were studied early on in robotics. They were seen as a symbolic way of describing of the environment that could provide a basis for higher-level reasoning; however, due to the impoverished sensors of the time, robust topological implementations were not really obtainable. As more accurate, precise, and informative sensors became available, researchers quickly realized that they could build much more detailed maps than ever before. This was a real breakthrough for modeling indoor environments. As a result, the idea of creating models of the world similar to human models of space was largely abandoned by the robotics community in favor of creating more precise and visually impressive CAD-like models of environments.[2]

Metrical Simultaneous Localization and Mapping (SLAM)

Powerful probabilistic methods have been developed for range-sensing mobile robots to perform simultaneous localization and mapping (SLAM) within a single frame of reference [Thrun et al., 2005]. These methods are accurate and reliable for online incremental localization within local neighborhoods. Sensing with sufficiently high frequency relative to local motion guarantees large overlap between successive sensory images. Current sensory information can be compared to the current map in order to improve localization. By analogy with radar signal interpretation, finding the correct match between observations and model is called the *data association* problem. After improved localization occurs, the sen-

---

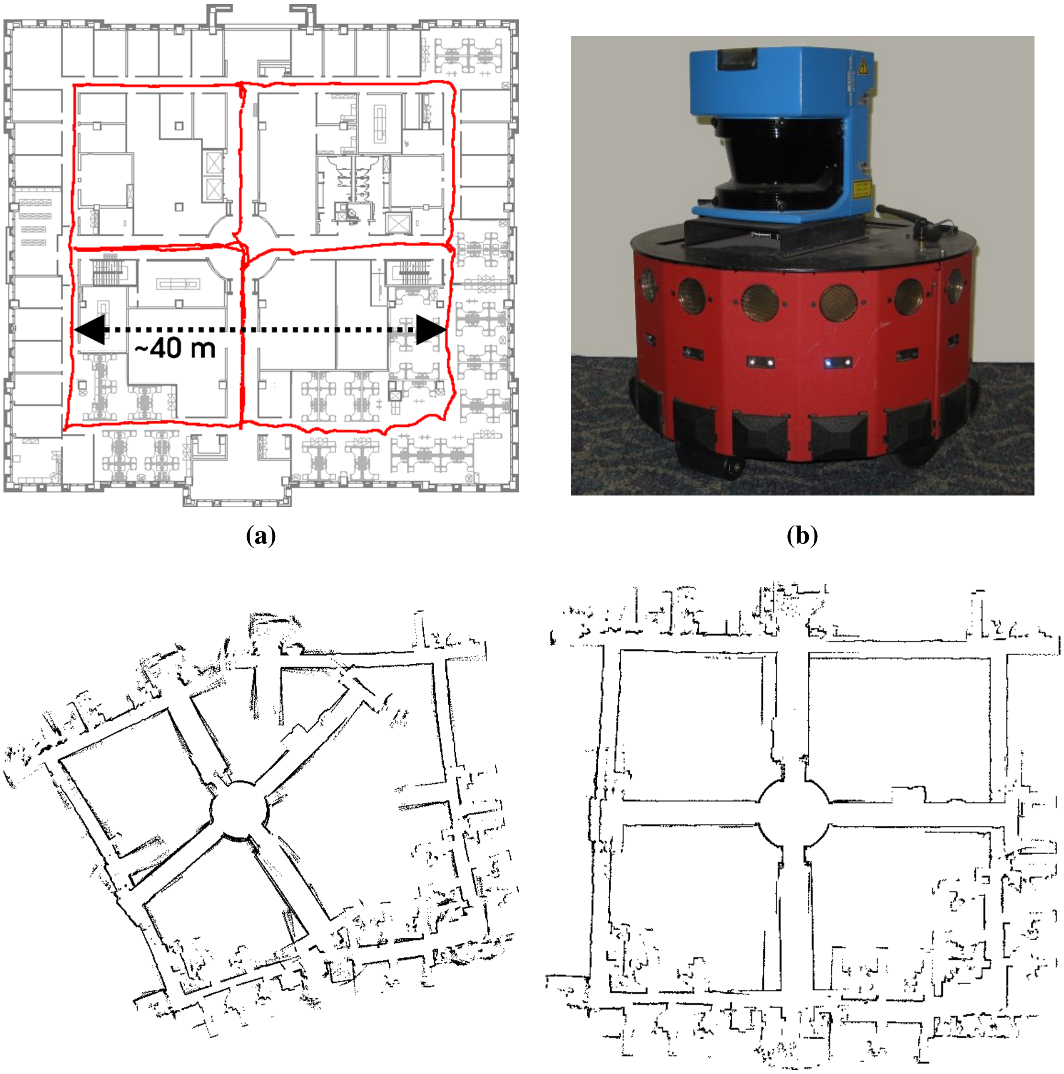
[2] In our opinion some metrical mapping research focuses too much on creating precise maps—focusing their robotics research on improving the visual quality of a map produced by offline computing, while forgetting that the main purpose for a map is to facilitate robust, efficient navigation by a mobile agent.

sory information is used to update the map for the next SLAM iteration. In local regions, many data association problems, such as the closing of large loops, can be excluded. The absence of large loops means that the problem of large-scale *structural ambiguity* does not arise in the local metrical map.

Scaling Metrical SLAM Techniques

While metrical SLAM methods work in small spaces, they do not extend well to larger environments. Global metrical maps become more expensive to update and access without clever storage schemes. More important, is the difficulty that arises when closing large loops (Figure 2.1). Even with local SLAM methods that use perception to improve the accuracy of localization, odometry error accumulates in the relation between the map's global frame of reference and the ground-truth reference frame of the real-world environment. This global error becomes even more pronounced in environments with long paths that have few distinguishing features. Without exact data association along paths, localization often drifts from the ground-truth, both in the robot's distance along the path and in the robot's heading, causing straight paths to compress, stretch, or curve in the map.

There are ad hoc methods for hypothesizing loop closures when the global odometry error is small. When a large loop is closed, accumulated error will often result in the robot's current observations clashing with older portions of the map. Methods exist that search for a nearby pose in the older portions of the map where perceptions match the prediction (propagating detected global error backwards through the exploration trace) [Lu and Milios, 1997; Hähnel et al., 2003a]; however, these solutions can fail in sufficiently large or complex environments. For example, Cummins and Newman [2008] discuss closing loops over kilometers of travel, where small rotational errors lead to large positional errors, and the correct loop closure may never be considered by odometry-based solutions. Additionally, if the environment is subject to *perceptual aliasing* (different locations look the same), then the matching process may close the loop incorrectly, distorting the map as a whole



(a) **(b)**

(c) **(d)**

Figure 2.1: *Closing large-scale loops.* Robots often need to explore, map, and navigate in large environments with multiple, nested loops. **(a)** This environment and the robot's trajectory through it are used as an example throughout. **(b)** The data comes from a Magellan Pro research robot with differential-drive odometry and a SICK-brand lidar device for precise, planar range-sensing. **(c)** This robot-made map of the environment in image (a) shows the effect of accumulated raw odometry error. **(d)** This map shows the improvement in pose estimation over image (c) by using metrical SLAM methods, but it also shows that significant errors still accumulate with respect to the real environment.

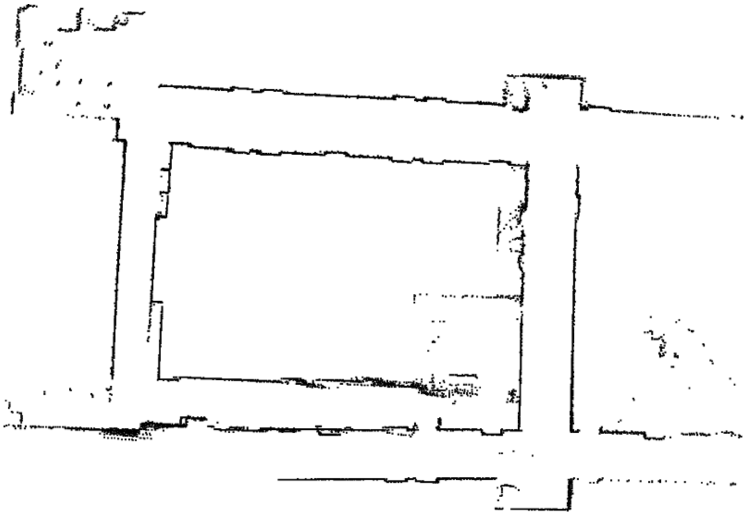


Figure 2.2: *Incorrectly closing loops using odometry and scan matching.* Metrical methods that attempt to match current observations to older portions of the map have been proposed. Sometimes these work well. Sometimes they do not work at all. As illustrated here, sometimes they seem to succeed as observations match older portions of the map, but the loop is incorrectly closed. (Adapted with permission from Eliazar and Parr [2003].)

(Figure 2.2). Depending on the amount of symmetry in the environment, a single incorrect match can lead the learner down an arbitrarily long "garden path" before the error is discovered. It is still unclear how probabilistic methods can properly discover an incorrect map and how they might efficiently backtrack to hypothesize a different loop closure [Hähnel et al., 2003b].

Some research on map-building avoids loop-closing issues by explicitly assuming that the correct data association is known [Leonard and Newman, 2003; Paskin, 2003]. In some cases, even without an explicit assumption about data association, impressive feats of large-scale map-making depend on locations in the environment being sufficiently distinguishable based on local cues [Montemerlo et al., 2002; Konolige, 2004]. Others accept false negative matches in order to avoid false positives, sometimes improperly hypothesizing that a previously seen location is new [Bosse et al., 2003]. This can eliminate the possibility of closing a loop correctly and finding the correct map, which leads to poor planning and navigation performance. In a rich environment with noise due to dynamic changes, it could be that every location is in principle distinguishable, but it is difficult or impossible to know which features identify the place, and which are noise. Methods created to distinguish between perceptually aliased states can get confused under scenarios of *perceptual variability* (the same place looks different on separate occasions) [Kuipers and Beeson, 2002] causing a single physical location to be represented multiple times in the same map.

Early approaches to probabilistic localization and mapping used particles to represent a distribution over robot poses for localization, but there was a single shared map that was updated from the maximum-likelihood pose hypothesis [Thrun et al., 2000b]. This could produce an incoherent map due to an incorrect and premature commitment to a maximum-likelihood pose hypothesis that turned out to be incorrect. A more principled approach uses Rao-Blackwellized particle filters to explicitly represent the distribution of trajectories and maps by maintaining multiple metrical map hypotheses [Montemerlo et al., 2003; Eliazar and Parr, 2003; Hähnel et al., 2003a]. These methods are run offline (due to computational demands) after exploration is completed, forgoing useful *active exploration* techniques capable of eliminating some loop-closing hypotheses. Additionally, in large, symmetric environments, intractably large numbers of particles may be required to avoid particle depletion when closing large loops. Particle depletion is a failure to have a particle in the distribution that adequately models the correct map.

In addition to the problems that arise in *building* large metrical maps, there is the associated problem of *navigating* in large metrical models. While search is relatively straightforward in small, local metrical models or in compact topological representations, planning over large distances in metrical models (represented as point clouds or large 2D or 3D grids) is quite computationally intensive.[3] Despite this fact, many robotic researchers continue to focus on improving the accuracy and preciseness of large metrical models, many times

---

[3] Probabilistic roadmaps [Kavraki et al., 1996], rapidly-exploring random trees [Kuffner and LaValle, 2000], or other modern planning techniques may improve the speed of planning tasks; however, having hierarchical representations facilitates efficient local replanning without having to replan the entire global route.

even driving the robot themselves to gather observations. This defeats the original purpose of building a model, which is to facilitate robust exploration and navigation.

2.2.2 Relevance of Metrical SLAM to this Thesis

The work in this thesis depends on a *small-scale* model that we call a *local perceptual map (LPM)*. This local map is a highly accurate metrical map of the immediate local surround. It has no long-term persistent state as it scrolls with the robot. It is useful for abstracting the local, continuous environment into a stable model of small-scale space. It supports local planning and motion control. Symbolic abstraction can then be performed using the LPM representation.

The current implementation of this LPM is an occupancy grid map [Moravec, 1988; Elfes, 1989] that uses planar lidar (laser range-sensing) devices to detect locations in the world occupied by obstacles. The occupancy grid algorithm discretizes the world into cells, each of which holds a probability of a cell being occupied by an obstacle—cells are initialized to 0.5 probability for unmapped regions. Based on occupancy, cells in the grid can be classified as occupied, free, or unknown. Our LPM is built using the SLAM particle filtering techniques described above to create a single maximum-likelihood map hypothesis. As a result of using range sensing, we are currently focusing our efforts on environments with vertical walls.

We fully expect current progress in Visual-SLAM [Sim and Little, 2006], vision-based road/sidewalk detection, and vision-based obstacle detection (along with 3D lidar sensing as it becomes more available) to eventually give us LPMs of outdoor environments with dynamic obstacles. Other related research is currently looking at ways of producing semantically labeled LPMs to facilitate campus-wide navigation [Murarka et al., 2006]. Given LPMs as a base description of the local surround, our theory (and to a large extent our implementation) of hybrid map-building and navigation can remain unchanged.

2.2.3 Topological Maps

Cognitive map research supports the creation of topological maps of large, complex environments [Lynch, 1960; Siegel and White, 1975; Kuipers, 1978; Chown et al., 1995; Kuipers, 2000]. A topological map, in its most basic form, represents an environment as a graph where nodes represent places and edges represent connections between places. Several groups of robotics researchers have presented distinct topological implementations that differ in their semantics for the graphs—how they define/describe places and actions between them [Kuipers and Byun, 1991; Mataric, 1992; Shatkay and Kaelbling, 1997; Duckett and Nehmzow, 1999; Choset and Nagatani, 2001; Morris et al., 2005]. Some implementations build topological maps autonomously, some are given topological maps *a priori* [Koenig and Simmons, 1996], and some let the robot explore autonomously while the researcher provides place names to overcome perceptual aliasing issues [Kortenkamp and Weymouth, 1994].

Topological navigation is a behavior that is used by a variety of different animal species, including humans [Lynch, 1960; Gould, 1990; Stopka and Macdonald, 2003]. This helps facilitate large-scale spatial reasoning, mainly due to the compactness of the representation. The symbolic nature of topological representations allows higher-level reasoning (such as containment, order, connectivity, regions) which can enhance large-scale spatial knowledge. Because the environment is discretized into a graph, movement errors do not accumulate globally. Additionally, the abstraction of continuous space into symbols facilitates communication about space between agents. Possibly the most important difference for future robotics research is that topological maps allow hierarchical models that support efficient navigation, planning, and communication about very large regions of space: e.g., a building may consist of many topological places, but may be a place in a higher-level topological network of a campus.

The major hurdle for topological map-building has been the reliable abstraction of useful symbols from continuous, noisy perceptions of the environment: i.e., how to reliably detect and recognize places and paths. This is an instance of the more general *symbol grounding* problem [Harnad, 1990] that has troubled the AI community for many years. Probabilistic approaches are good at overcoming the kinds of local uncertainty and systematic noise that can hinder reliable symbol extraction. Incorporating probabilistic data association techniques into the topological map-building paradigm has sparked interest in hybrid map-building, including the HSSH approach presented in this dissertation. We discuss hybrid mapping approaches below; however first we detail research on exploring and navigating graph-like models on environments.

Graph Exploration

Early work on topological mapping ignored the embodied agent altogether, bypassing difficult symbol grounding issues, and focused on the theoretical results of exploring graphs.

Deterministic Finite Automatons. Rivest and Schapire [1989] produced one of the initial pieces of work on formalizing topological-map building. They describe topological map-building as the problem of determining the most simple deterministic finite automaton (DFA) that explains symbolic input/output experience. A topological map can be thought of as a six tuple, $M = (S, A, V, \zeta, s_0, \gamma)$, where:

- S is a finite nonempty set of states
- A is a finite nonempty set of inputs or basic actions
- V is a finite set of outputs or percepts
- ζ is the transition function, $\zeta : S \times A \rightarrow S$
- s_0 is the initial state
- γ is the output function, $\gamma : S \rightarrow V$.

No perceptual aliasing exists if $\forall s_1, s_2 \in S\ [\gamma(s_1) = \gamma(s_2) \rightarrow s_1 = s_2]$. In this case determining the correct DFA is trivial. When $\exists s_1, s_2 \in S\ [\gamma(s_1) = \gamma(s_2) \wedge s_1 \neq s_2]$, then perceptual aliasing exists, making the general problem of inferring the smallest DFA NP-complete [Gold, 1978; Angluin, 1978].

Rivest and Schapire [1989] show that there is a polynomial-time algorithm for inferring the smallest DFA given that a *distinguishing sequence* can be learned in polynomial time. A distinguishing sequence, $d \in A^*$, results when $\forall s_1, s_2 \in S, s_1\langle d \rangle = s_2\langle d \rangle \rightarrow s_1 = s_2$, where $s\langle d \rangle$ denotes the sequence of outputs resulting from executing the sequence of actions d starting in s. Unfortunately, later it was shown that determining whether a DFA has a distinguishing sequence is PSPACE-complete; however, an *adaptive distinguishing sequence* can be determined in polynomial time [Yannakakis and Lee, 1991]. The output symbols of the DFA determine the branches in a decision tree, which defines an adaptive distinguishing sequence.

Dudek et al. [1991] investigate using multiple unique markers that can be dropped and picked up by the robot. In a sense, this lets the robot create distinguishing sequences on the fly—dropping markers at proposed loop closures before retracing its exploration to look for the marker at some known state from the past. Although using markers still results in polynomial time exploration, experimental evidence shows a reduction in the number of actions necessary to properly model an arbitrary environment. This reduction levels off as the number of unique markers approaches the number of aliased states in the environment.

DFAs and Uncertainty. Thinking of topological maps as deterministic finite automata allows us to keep the map-building problem tractable—if not theoretically, at least in practice. However, in many mobile robot implementations, actions or observations may have uncertainty, which is difficult to model using DFAs. Some researchers have attempted to learn stochastic finite automata under various conditions of uncertainty [Dean et al., 1995; Basye et al., 1997; Shatkay and Kaelbling, 1997].

In Rivest and Schapire [1989], the observation of a state was deterministic $\varphi(s) = \gamma$, though perceptual aliasing could still exist. Dean et al. [1995] consider the case where perceptual variability also exists (the observation function is non-deterministic), yet actions are still deterministic. Here there exists some observation function, such that it is possible that upon visiting state s on two occasions i and j, $\varphi(s_i) = \gamma$ and $\varphi(s_j) \neq \gamma$. They conclude that it is possible to find a probably approximately correct (PAC) deterministic finite automaton [Valiant, 2000]. This can be done in polynomial time given that the environment has a distinguishing sequence and such a distinguishing sequence is known prior to map-building (or can itself be learned in polynomial time).

Shatkay and Kaelbling [1997] attempt to build a model of an environment in the face of perceptual aliasing, perceptual variability, and uncertain actions. Given probabilistic actions, an environment can be modeled as a Hidden Markov Model (HMM). They utilize the Baum-Welch algorithm to converge to a model that maximizes the expectation of the observed data. Specifically, they utilize odometry information between states in order to find a model, M, that explains the experience, $E = \{v_0, a_0, ..., a_{n-1}, v_n\}$. This finds a likely local maximum, which may be far from optimal: arg max $p(E|M)$. In experiments, the models generated were consistent with all available metrical information, but far from topologically correct.

One major problem with these PAC learning approaches is that a large amount of experience is needed in order to come up with a useful model. The amount of experience needed is often far more than what is reasonably acceptable for a usable robotic implementation. Additionally, though models may be probabilistically optimal, they may not accurately reflect the ground-truth of the environment. As a result, we prefer to utilize the high-precision observations from modern sensors in order to assert that both observations and actions are deterministic. Future work should look into the impact where rare non-deterministic events occur: detecting and understanding these unusual events should allow us to still assume deterministic actions $(100 - \varepsilon)\%$ of the time.

Voronoi Graph Exploration

Building DFA or HMM models of environments depends on some way of detecting and categorizing places in the environment. Early work on place detection was often quite ad hoc [Mataric, 1992; Kortenkamp and Weymouth, 1994], as most research robots used noisy sensors. Using Voronoi graphs [Voronoi, 1907; Fortune, 1992] to model paths and places became a well-known approach to detect and classify places in a more principled way.

Voronoi Junctions as Places. For some time, researchers have proposed utilizing variants of the Voronoi tessellation as tools for detecting places and proposing paths through enclosed environments [Miller, 1985; Rao, 1995]; however, in recent years Choset and his colleagues have been the main proponents for the use of Voronoi graphs in topological map-building and navigation. Choset and Burdick [2000] define a *generalized Voronoi graph (GVG)* as a one-dimensional set of points equidistant to the n closest obstacles in n dimensions, where a preset threshold determines whether observations belong to the same "obstacle". When using planar range sensors (like common laser and sonar devices) in walled environments, this results in a set of points equidistant from two or more obstacles (Figure 2.3).

Researchers utilize the GVG both as a roadmap for robot control and as a way to discretize the continuous environment into a finite set of places. Whenever the robot arrives at a junction point in the Voronoi graph (a point equidistant from $n+1$ or more closest obstacles), it decides that it is at a topological place. In corridor environments, Voronoi junctions occur at corners and intersections.

In order to describe a place for future recognition, they count the number of edges emanating from a junction point—the degree of the graph node. In doing so, they eliminate perceptual variability completely (assuming the environment, thus the GVG, is static). This gives them an observation of a place equivalent to the one used by Dudek et al. [1993] in work on graph exploration. However, simply using the degree of a graph vertex causes a

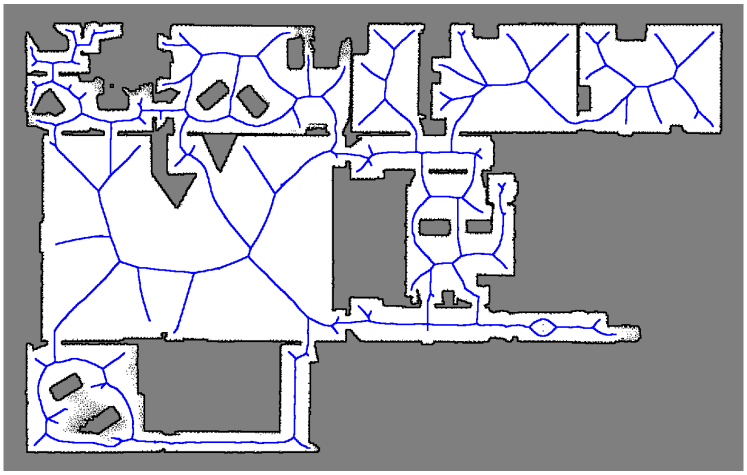


Figure 2.3: *The generalized Voronoi graph (GVG) of a global metrical map.* Using an occupancy grid, the Voronoi graph of free space can be drawn by using any occupied cells as obstacles. Junction points include intersections in corridor environments. For visual clarity, portions of the graph that are close to obstacles are removed.

large amount of perceptual aliasing.

To reduce aliasing, Choset and Nagatani [2001] also use the Voronoi radius (distance to the nearest obstacle) of the junction node to help classify places. This still does not help to eliminate aliasing between qualitatively different intersections, such as a + intersection which has two intersecting paths and a K intersection which has three intersecting paths. Both of these intersections have four Voronoi edges and could have similar Voronoi radii given the architecture of the building.

Choset and Nagatani [2001] also admit that the Voronoi graph can find many spurious topological places due to range-sensing noise or concave corners. They propose using a *reduced generalized Voronoi graph (RGVG)* that removes many “spurs” in the graph. The RGVG removes branches that terminate with only two nearest obstacles. This improves on prior Voronoi navigation approaches, but the RGVG can still leave some spurs that create false positive/unstable places. Additionally, some places may have no Voronoi junctions in the RGVG, while other place neighborhoods may encapsulate multiple nearby junctions. This is discussed in more detail in Section 5.3.3.

Voronoi-style Exploration. There are several implementations for place detection that are similar to Voronoi junction detection. Kuipers and Byun [1991] detect "distinctive places" (a precursor of the distinctive states discussed in Chapters 3 and 8) via wall-following controls that end in hill-climbing to maximize a "distinctiveness measure". The specific distinctiveness measure they use attempts to maximize the distance from the nearest distinct obstacles, which essentially moves the robot to the Voronoi junction. This requires the robot to physically move to a point place, rather than modeling the place as a region encapsulated by the radius of the Voronoi junction or by gateways. A similar control-based approach is used by Lee [1996], Menegatti [2002], and Victorino et al. [2003].

Silver et al. [2004] use local models of nearby obstacles to detect Delaunay triangles. The Delaunay triangulation [Delaunay, 1934; Fortune, 1992] of a set of planar points is a triangulation such that no point is inside the circumscribing circle of any triangle, with the constraint that the triangulation maximizes the total edge length of all the triangles. The Delaunay triangulation of a set of points is the *dual graph* of the Voronoi tessellation of the points.

Delaunay triangles exist throughout the environment, so the algorithm of Silver et al. [2004] ignores triangles that do not have all three sides longer than a predetermined threshold. This eliminates many false positive place detections that junction-based methods may find. At intersections, large Delaunay triangles exist; thus intersections are detected well.

Intersections in this implementation are characterized by the size of the associated triangles, but this is not a very distinctive measure in symmetric environments. Additionally, at Voronoi junctions, Delaunay triangles connect the $n+1$ nearby obstacles in the n-dimensional model. This means that certain simple intersections, e.g., a noise-free $+$ intersection, *must* be divided into multiple places when using Delaunay triangles. Finally, this method is only valid in corridor-style environments, where paths have two sides to anchor the triangle vertices.

Choi et al. [2002] also use skeletal graphs to detect places in local metrical models. Instead of the GVG, they utilize a thinning algorithm [Zhang and Suen, 1984] on the discrete, "pixelized" occupancy grid. A thinned skeleton approximates the Voronoi skeleton, except that many spurs that appear in the GVG do not appear in the thinned skeleton, due to the specifics of the thinning algorithm. This pre-pruned skeleton reduces false positive place detections, but like the reduced GVG (RGVG), it fails to detect important places where junctions may not occur.

Thinned skeletons, along with discretized, pixel-based Voronoi approximations, can be computed in linear time; thus they are useful, pruned approximations to the continuous Voronoi skeleton. Thinned skeletons actually "look" nicer to the eye than Voronoi graphs, as they tend to yield skeletons with straighter lines. However, these skeletons do not always end up being equidistant from multiple nearby obstacles, which can be detrimental to any algorithms that depend heavily on this Voronoi property. Section 6.1 discusses how this approximation technique is used to help find gateways.

2.2.4 Spatial Semantic Hierarchy Theses

This dissertation extends the Spatial Semantic Hierarchy [Kuipers, 2000], a framework for modeling large-scale space at a variety of related abstractions. There have been several dissertations regarding the Spatial Semantic Hierarchy throughout the years [Kuipers, 2008].

Byun [1990] introduced the Spatial Semantic Hierarchy, focusing on the low-level trajectory and hill-climbing controls needed to perform reliable exploration with a simulated robot (see Chapter 3). Lee [1996] improved upon this work by creating an SSH implementation that worked for a real robot using sonar sensors. He specifically decided against using local models as they were difficult to generate with noisy sonar devices, instead relying on hand-crafted, behavior-based hill-climbing controls to detect places and paths. Recently, Menegatti [2002] proposed using his omnidirectional vision hardware to detect vertical edges in the surround in order to hill-climb to distinctive states in indoor

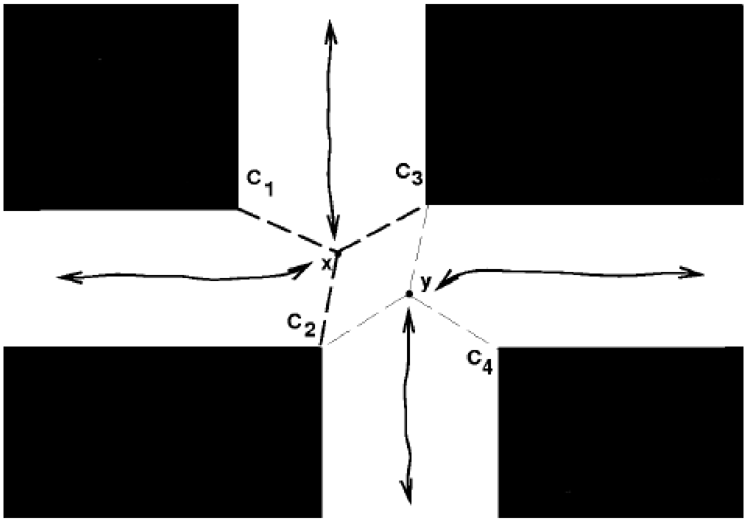


Figure 2.4: *Detecting places with an offset path.* Many place detection implementations, like those that use Voronoi junctions or hill-climbing controls, cannot detect single places with paths that "jog" slightly at the intersection. Instead they model these intersections as multiple nearby places. The *gateway* approach we discuss in Section 6.2 detects this intersection as a single place with two paths. (Image adapted from Lee [1996].)

environments.

All of these works provide distinctive state detection and recognition directly from sensor data, which made the implementations highly dependent on rectilinear walls. Certain scenarios, like paths that become slightly offset at intersections, can cause single places to be modeled as multiple places. This is because hill-climbing controls find multiple distinctive peaks when paths "jog" slightly at intersections (Figure 2.4). This is similar to the problems faced by Voronoi junction approaches.

Remolina [2001] focused on a logical formalism that allowed a robot to build cohesive topological networks from the causal experience between distinctive states. He focused on generating and comparing map-hypotheses using nested abnormality theory [Lifschitz, 1995]. This required using a circumscription policy to compare different logical models. Savelli [2005] investigated adding quantitative data to the circumscription policy in order to improve the comparison between logical models. This work also investigated improvements to the topological map-building theory via planarity constraints.

2.2.5 Relevance of Topological Maps to this Thesis

Due to the growing number of potential maps and the large amount of exploration needed to handle uncertainty in place detection (non-deterministic actions) and place recognition (uncertainty in observations), we prefer to assume a deterministic model of the world; however, this can be difficult without reliable place detection and place descriptions. As a result of the place detection algorithm discussed in Section 5.3, we demonstrate that deterministic actions and place descriptions are achievable in certain environments, and we are confident that our methods will scale nicely to near-deterministic models in more complex domains that may require visual sensing in highly dynamic environments. As a result, we model environments as deterministic finite automata (DFAs), and should be able to utilize adapting distinguishing features or other theoretically sound policies for improved exploration strategies in the future.

As we have seen, Voronoi graphs have been used to produce certain place detection/classification implementations, but these methods suffer from several shortcomings due to relying on branch points in the Voronoi graph. It is our belief that with descriptive metrical models of the local surround, we can improve upon this work, and detect/describe places more robustly. As detailed in Section 5.3.3, our gateway approach (which also relies heavily on Voronoi approximations), overcomes many of the problems of the junction-based approach to place detection and classification. Additionally, we expect our gateway method to work on true Voronoi skeletons or on the thinning-based approximations, which allows us to utilize any state-of-the-art, efficient implementation of a Voronoi-style skeleton of free space.

This dissertation improves on previous SSH work. Our implementation handles non-rectilinear intersections, eliminates the need for awkward hill-climbing motions, and improves the efficiency of generating and comparing logical topological models. Also, the

LPM representation is currently built by lidar sensors, but could also project 3D vertical edges from omnidirectional vision into the 2D local metrical representation.

2.2.6 Hybrid Maps

One useful aspect of topological maps is that they discretize large, continuous environments into a manageable set of connected places. Metrical maps are useful as they are a natural model to produce with a robot using precise lidar sensors and high-frequency odometry measurements. Additionally, they give a visually compelling picture about the model the robot has built of an environment (detecting a loop-closing error in a symbolic representation is more difficult that simply seeing a misaligned hallway in a metrical map). Recently, robotics researchers have begun to look at hybrid topological/metrical representations in order to try to leverage the benefits of both approaches. There are too many hybrid approaches to mention here, many with only very subtle differences. We refer the reader to the survey of such approaches by Buschka [2005], as we refer to well-known or prototypical examples in this discussion.

Basically, publications about hybrid metric/topological representations fall into two categories, both of which are addressed by the work in this thesis. Many robot implementations use local metrical models as local *observers* that help filter out sensor noise, aggregate observations over time, and create plans that avoid nearby obstacles. There are researchers specifically interested in using metrical models to try and determine qualitatively distinct or interesting places as the robot explores a new environment [Yeap and Jefferies, 1999; Lankenau et al., 2002; Tomatis et al., 2002; Ko et al., 2004]. This is related to our work on grounding places and paths in local metrical models, though our assumption of deterministic places (detection and categorization) is more rigid than most approaches.

The second hybrid approach focuses on using "places" in order to reduce the number of locations in the world that must be considered when hypothesizing metrical loop closures. That is, the goal of most hybrid mapping techniques is still to achieve a global metrical map; however, they use some "topological" (i.e., graph) constraints to make the closing of loops more efficient. Many of these implementations record places arbitrarily [Duckett and Saffiotti, 2000; Zimmer, 2000; Blanco et al., 2008], e.g., every 5 meters traveled, in order to reduce the number of locations in the world where loop closures can occur. Others use a feature buffer, so the robot creates a new place at every *n* corners or wall segments [Bosse et al., 2003]. Some approaches simply have the researchers press a button to define places in the world [Thrun et al., 1998]. Our research is related to these approaches as well, in that given autonomous place detection at qualitatively distinct and metrically distant places, we can provide a compact graph to make these approaches extremely efficient.

Finally, a third category of hybrid mapping has only recently been investigated. There has been recent research looking at modeling the full Bayesian distribution over topological hypotheses [Ranganathan et al., 2006; Blanco et al., 2008]. These methods are still strongly grounded in using odometry knowledge (which can be unreliable over large distances [Cummins and Newman, 2007]) and/or aligning raw lidar measurements. As mentioned above, the "places" they utilize are determined by ad hoc means: by the researcher via button presses or by using distance thresholds or finite-sized feature buffers. We believe our compact topological representation is useful here as well. Section 10.1.2 addresses how this new area of hybrid research meshes with the HSSH.

2.2.7 Relevance of Hybrid Mapping to this Thesis

Labeling a region as a place using an arbitrary rule can improve the efficiency of global relaxation by having fewer constraints. Likewise, it can improve Bayesian filtering of possible metrical loop closures as there are fewer low-level matches to consider. However, these approaches to creating "places" do little to improve the spatial knowledge of the robot itself.

In our experience, metrical and topological representations for space are very different in character, or more precisely, in ontology. The topological map describes the structure of large-scale space. It abstracts away the specific nature of sensory input and the specific methods used for matching sensory images when the topological map is created. Metrical mapping techniques that rely on local overlap of successive sensations, on the other hand, precisely capture the structure within the local sensory horizon: small-scale space.

The *Hybrid Spatial Semantic Hierarchy (HSSH)* we present in this thesis is the first framework and implementation to close the loop on the process of going from metrical sensations to both metrical and symbolic models of both small-scale and large-scale space—moving from metrical models of small-scale space to symbolic representations of small-scale space, inferring large-scale structure via symbolic inference, before producing a consistent global metrical model from the symbolic structure. Our approach autonomously detects and describes qualitatively distinct places, creating far fewer places than other "hybrid" approaches. Additionally, these places have meaning as they are inspired by the human cognitive map.

It is important to note that our model of *large-scale space* is topological; thus, the problem of closing loops is one of topological place detection and recognition. As a result, we must bridge the gap between local metrical models of small-scale space and global symbolic models of large-scale space. In Chapter 5, we show that the concepts of places and paths can be grounded in local maps via the *gateway* construct, which is discussed below.

Although the rest of this dissertation focuses heavily on an initial HSSH implementation, our contributions of autonomous place detection and place description should improve most other hybrid map-building implementations. As an example, Thrun et al. [1998] illustrate a way of integrating topological maps and metrical maps to create large, global metrical maps. The processes includes several passes of *Expectation-Maximization (EM)* using both an alpha (Markov localization) step and a beta (future observations and actions) step to converge to a consistent metrical map. Unique topological "places" are defined by the researcher and the robot uses its odometry to figure out the best possible map (large grained grid) of the world. Computation is done offline and assumes several things that our work directly addresses: (1) places can be reliably detected, (2) places can be recognized on revisits, (3) exploration of the environment is complete. Our topological map-building addresses these issues to various degrees, as we will see in Chapters 5 and 8.

2.3 Gateways

In looking for a way to parse local environmental structure in order to abstract places and paths, we came up with the idea for *gateways*. Gateways are intended to delineate the area belonging to a place from the paths or doorways that lead away from the place. Upon looking at relevant published work, we realized that the idea of a gateway has been "discovered" (in some form or another) multiple times [Yeap, 1988; Chown et al., 1995; Thrun and Bücken, 1996]. Below we discuss the history of gateways, detail previous attempts at gateway implementations, and discuss their pros and cons.

As part of this dissertation, we provide our own implementation for grounding gateways in the kinds of metrical models easily obtained by modern robotic sensors. Our approach improves on the previously published gateway approaches in that our algorithm is meant to be environment independent, resistant to noise, resilient to scaling parameters, and effective in both corridor-following and wall-following situations. This is detailed in Chapter 6.

2.3.1 The Foundations of Gateways

Chown et al. [1995] propose a cognitive model of large-scale space that tried to unify the network of object landmarks proposed by Siegel and White [1975] with visual scenes. Additionally, they argue for using a neurally plausible associative network platform to associate visual scenes with locations in the world. In this work, they recognize that the object-landmark-based cognitive models that assume open spaces, dotted with beacon landmarks, cannot handle certain common situations: doorways, entrances to caves, exiting forests, and other scenarios where large visual changes occur. They propose that *gateways* marked these types of places, and they note that gateways provide a structural-based definition of a landmark.

> In buildings, these [gateways] are typically doorways.... Therefore, a gateway occurs where there is at least a partial visual separation between two neighboring areas and the gateway itself is a visual opening to a previously obscured area. At such a [location], one has the option of entering the new area or staying in the previous area. [Chown et al., 1995, p. 32]

> In outdoor environments gateways might include mountain passes, bridges, or openings in a forest. Gateways are characteristically [locations] where people stop to pause and look around. This may be because they occur at choice points, such as intersections.... In their own way gateways are landmarks defined by structural properties. [Chown, 2000, p. 2]

Chown [1999] further compares the traditional object-landmark-based models to ones that utilize local maps (e.g., the local perceptual map of the HSSH).

> Because the local map is invested in stronger assumptions about environmental structure than the landmark structure, it is less likely to strongly apply to [an arbitrary] environment, but more likely to yield significant payoff when it does. The local map structure relies on some type of natural gateways.... Gateways are useful because they mark transitions from one [subset of an environment] to another.... Cognitively this allows the world to be broken up into smaller pieces.... Another important attribute of an effective gateway [is] that it is both an exit and an entrance. [Chown, 1999, p. 28]

2.3.2 Gateway implementations

Despite several publications that discuss the concept of gateways, Chown and colleagues never explicitly discuss an algorithm or give a precise formal description of gateways. This has led to several implementations over the years, including ours, which we discuss in Chapter 6. Figure 2.1 at the end of this chapter, compares the abilities of each of these approaches.

Sensor-based Gateways

> A simple way to detect a gateway is to look for [locations] where the perceived background distance suddenly changes. For example, when a vista that had been occluded suddenly opens up. [Chown, 2000, p. 2]

The above quote is the closest description of a gateway algorithm given by Chown and colleagues. This approach, of looking for disparities in the range estimates to nearby obstacles is utilized by several researchers investigating indoor hallway navigation [Kortenkamp and Weymouth, 1994; Schröter et al., 2004].

A disparity approach relies on the measured distances from the robot's current location. Consequently, the approach can yield different results depending on the robot's location with respect to occlusions in the environment. This is an undesirable property, as we would like stable place descriptions, independent of the robot's location in the place. Additionally, hallways of different sizes, or maps with different parameters (e.g., cell size), demand different distance metrics, which affects the robustness of such implementations. The approach we take in Section 6.3.2 is an attempt at a scale and location invariant algorithm that overcomes these issues.

A disparity-based implementation for a mobile robot was proposed by Kortenkamp and Weymouth [1994]. Their robot used sonar sensing to perform wall-following in a rectilinear environment. The gateway implementation is quite simple in that the robot simply looks for walls on the left or right to "disappear" or a wall to "appear" in front of it. Left, right, and front are defined with respect to the orientation of the wall the robot is following. Once a wall disappears or appears, the robot determines it is *at* a gateway. In this particular implementation, a gateway is basically equivalent to a *distinctive state* of the Spatial Semantic Hierarchy, as it corresponds to being at a particular place and facing along a particular path. This implementation has the drawbacks mentioned above, along with the additional limitation of only handling rectangular environments.

The most recent work on gateways, outside of our own, has been by Schröter et al. [2004]. The authors discuss a lidar-based gateway implementation that uses what they call *virtual line models (VLMs)*. Like the approach above, VLMs are "based on the assumptions that environments are rectangular and [all] hallways have the same width" [Schröter et al., 2004]. The idea is quite straightforward, in that only L, T, and + intersections, open doorways, and dead-ends need to be modeled. The algorithm has a model for the left wall, right wall, forward wall (orthogonal to the left and right walls), and a hidden wall (parallel to the forward wall, but often unseen due to forward-facing lidar sensors). It fits laser data to this model using Expectation-Maximization (EM) to hypothesize lines and preserve right angles. It then looks for hallway-sized gaps in the line models where free space exists. These gaps are the gateways. Given the size and configuration of gateways, the robot determines the type of place.

For example, when traveling down a long corridor, the laser data fits the left wall and right wall data well, but no data fits the forward wall or hidden wall data. Thus the robot can hypothesize it is not near a place. As it approaches a + intersection, it begins to see the far wall of the orthogonal hallway. This data fits the forward wall line model, and the range disparity between the end of the left and right walls and the forward wall explains the hidden wall. Thus the robot hypothesizes 4 gateways that form the square where the two hallways meet. Smaller gaps are used for doorways. This method falls short of the generalized local topology abstraction presented in Chapter 5, as it assumes specific sizes and angles of hallways and can degrade in performance when obstacles clutter the range measurements.

In his thesis, Schröter [2006] extends this gateway implementation by using a collection of recent lidar returns. This is an attempt to overcome inaccurate gateway detections due to clutter and other occlusions that affect fitting data from a single robot pose to the virtual line models. This method still does not always detect gateways correctly, even with the assumption of right angle intersections and uniformly sized hallways. More importantly, his thesis examines visual detection of closed doors to hypothesize gateways unseen by lidar sensing. He also investigates visual furniture detection to ignore non-architecture in spatial reasoning. Adding in such visual object detection can only improve the gateway methods discussed here, as well as the lidar generated LPMs that facilitate our gateway implementation.

Model-based Gateways

In an attempt to overcome the problems of robot-centered, sensor-based approaches. There has been some investigation into finding gateways in metrical models built from range sensors. Youngblood et al. [2000] outline a gateway implementation that looks for openings of a certain size in an occupancy grid. They define gateways as line segments connecting occupied cells that are separated by free cells by more than a minimum distance but less than a predetermined maximum distance. This finds gateways at doorways in their domain. Gateways at intersections cannot be found using this method as hallways generally have a reasonably uniform width throughout, including when they begin (or end) at an intersection. Like the implementations discussed above, this approach also suffers from needing environment dependent thresholds.

Thrun and Bücken [1996] give an implementation that also looks at distances between occupied cells in a grid; however, their approach is a bit more general in that they look for local minima in distances instead of looking for distances of a predetermined size. This implementation takes a global metrical map of an environment, generates a Voronoi skeleton, and traverses the skeleton looking for “critical points”, where the distance from a point on the skeleton to the nearest obstacle is smaller than the corresponding distances of the neighbors on the Voronoi skeleton.

At critical points, the algorithm connects the skeletal point to the nearest obstacle on each side of the skeletal branch. The authors call these “critical lines”, though in general a Voronoi point and the two closest obstacles that define it are not necessarily collinear—making a line segment between the two obstacles seems sufficient. This approach works fairly well in indoor, rectangular environments where local minima occur in doorways and at the edges of intersections. Complex intersections often contain multiple Voronoi junctions; thus, there can be many local minima along the Voronoi skeletons of these regions. In these scenarios this method can create strange critical points and critical lines in complex intersections.

This approach motivates our *constriction*-based implementation discussed in Section 6.2, and works well in indoor, straight-walled environments, especially after noise is smoothed out of either the grid or the Voronoi graph distances to the closest obstacles. Our approach handles complex intersections by only looking for critical points (or constrictions) near the edges of place neighborhoods. The extent of place neighborhoods are approximated by our concept of a *core* of a local region (Section 6.1.2).

Occlusion-based Gateways

Chown [1999] cites the earlier work of Yeap [1988] as partial inspiration for their concept of gateways. Yeap [1988] proposes a cognitive mapping framework that relies on local maps connected together in a topological network. The local maps, are defined as groups of line segments that depict walls (they explicitly state that they ignore the issue of obstacles such as furniture), with openings between walls marked as *exits*. These exits are very similar to gateways. The approach here is similar to the sensor-based approaches of Kortenkamp and Weymouth [1994] and Schröter et al. [2004], but no explicit assumptions about rectangular intersections or fixed-width hallways are needed.

Yeap [1988] calls the local model of the robot's surround an *absolute space representation* (ASR). ASRs are generated by first extracting line segments, representing vertical surfaces in the environment, from collections of experience in a local region. Given line segments of a local region, the algorithm then hypothesizes the true boundary of enclosed environments, like rooms, by attempting to connect radially neighboring endpoints of the same type: "occluding" versus "occluded" endpoints. By connecting "neighboring" endpoints (preferably of the same type), ASRs determine the boundary of a room by the collection of walls and connections between endpoints. Connections that extend across free space (or across disparities caused by occlusions) are related to our concept of gateways: e.g., connected "occluding" endpoints almost always correspond to doorways in rooms.

There are several problems with this method, some discussed explicitly by Yeap [1988]. Among these are possible non-symmetric ASRs in symmetric environments due to the radial clustering technique utilized in looking for neighbors. Second no notion of what determines a "neighbor" is given, but presumably this is some angular threshold. Yeap and Jefferies [1999] overcome these problems by introducing a new set of rules for connecting endpoints.

Instead of performing a radial sweep to look for neighboring endpoints of the same type, Yeap and Jefferies [1999] explicitly look to connect each "occluding" endpoint to another appropriate endpoint to create an exit. Most often, the closest other endpoint (in Euclidean distance) is connected to an occluding endpoint. The new rules they use for connecting endpoints fix the non-symmetry problems and eliminate "neighbor" thresholds. This implementation defines exits at doorways and at intersections quite nicely, as doorways create occlusions, and usually, the two door jambs are the closest obstacles to each other (though counterexamples could be created that break this algorithm). As ASRs are meant to represent local, non-overlapping space, only the minimal set of exits and walls are kept, throwing out exits and walls that can only be reached by traveling through another exit.

This new ASR approach by Yeap and Jefferies [1999] still has some disadvantages that are inherited from the ASR approach of Yeap [1988]. First, there must be a metric for deciding what constitutes an "occlusion". This is the same problem mentioned above, where disparity metrics may be environment dependent. Again we see that the ASR shape is highly dependent on the robot's location in the local region, as occlusions are defined by a "virtual" 360° range sensor from the robot's position.

More importantly, because a new ASR is only computed when the robot moves through an exit (gateway) of a previous ASR, different starting points in a large-scale environment, and different exploration paths through the environment, can lead to drastically different collections of ASRs that describe the same environment. Having non-deterministic place abstractions is not a very good starting point for a topological framework—this presents a problem for recognizing places, labeling places, etc.

Finally, there are specific, important places that may not be modeled at all. Suppose the robot starts building an ASR in the middle of a $+$, L, or T intersection, where there are no occlusions. In this situation, no endpoint connections (thus no exits/gateways) exist near the intersection. This would mean two long, orthogonal paths can be grouped into one large, complex place neighborhood. Such a place is contrary to the human defined places used in English directions for hallway environments [MacMahon, 2007]. If the robot is just outside the actual intersection when it begins creating the ASR, there would be an occlusion (due to the "hidden" wall described by Schröter et al. [2004]). Here a gateway would be defined across the path segment at the intersection (i.e., connecting the left wall and right wall). Again, this is an example of having two drastically different place abstractions, due to slightly different starting locations when exploring the same environment. This non-deterministic discretization of an environment is not ideal.

Another slight problem discussed by Yeap [1988] is that two occluded endpoints can form a connection (gateway) behind an occluding obstacle. This "gateway" may not actually exist in the real world, as a wall could be hidden behind the occlusion. This problem with occlusions is one encountered in any gateway algorithm, including ours. The simplest solution we have found is to have the robot inspect the regions near *proposed* gateways before finally accepting their existence. This is discussed in Section 5.3.2.

Summary

Table 2.1 summarizes the abilities of previous gateway implementations, along with our implementations detailed in Chapter 6. The table compares various gateway implementations by their properties: gateways are independent of robot pose; gateways are independent of the size of paths; gateways have stable locations at places; gateways can be defined for arbitrarily shaped intersections; gateways are defined at intersections *and* doorways; gateways apply in wall-following, coastal navigation, scenarios.

As the table indicates, the methods of Yeap [1988] and Yeap and Jefferies [1999] work poorly if the robot starts looking for gateways when it is in the middle of intersection. The algorithm of Thrun and Bücken [1996] does not handle complex places where multiple Voronoi junctions (thus many “critical points”) exist; thus, multiple gateways can be placed inside complex shaped intersections or rooms. The method of Schröter et al. [2004] can potentially handle coastal navigation in rectangular environments, though this is never discussed by the authors. Section 6.2.2 shows cases where our initial implementation for detecting gateways can produce somewhat unstable gateways. Our final *anchor*-based solution to detecting gateways fulfills all the desired properties mentioned here.

	Pose independent	Scale independent	Stable locations	Non-rectangular intersections	Doorways and intersections	Coastal navigation
Yeap [1988]	no	no	no	**yes**	sometimes	no
Kortenkamp and Weymouth [1994]	no	no	**yes**	no	**yes**	**yes**
Thrun and Bücken [1996]	**yes**	**yes**	**yes**	no	**yes**	no
Yeap and Jefferies [1999]	no	no	no	**yes**	sometimes	no
Youngblood et al. [2000]	**yes**	no	**yes**	no	no	no
Schröter et al. [2004]	no	no	**yes**	no	**yes**	**possibly**
Beeson: Section 6.2	**mostly**	**yes**	**mostly**	**yes**	**yes**	no
Beeson: Section 6.3	**yes**	**yes**	**yes**	**yes**	**yes**	**yes**

Table 2.1: *Comparison of gateway implementations.* **Pose independent**: gateways at places are not determined by robot's location (uses a local model of the surround); **Scale independent**: method attempts to overcome high numbers of user-defined thresholds and metrics about door width, hallway widths, etc.; **Stable locations**: theoretically (ignoring noise), gateways are located in the same locations upon different explorations of the same environment; **Non-rectangular environments**: reliably handles both non-rectangular (e.g., 5-way or K) intersections and non-rectangular rooms; **Doorways and intersections**: gateways are (theoretically) found near both hallway intersections and doorways; **Coastal navigation**: gateways can be determined along non-corridor paths and places, e.g., following walls, the outside of buildings, or the edge of a forest.

Chapter 3

The Spatial Semantic Hierarchy

> *Maps encourage boldness. They're like cryptic love letters. They make anything seem possible.*
>
> Mark Jenkins, *To Timbuktu: A Journey Down the Niger* (1997)

This chapter describes the Spatial Semantic Hierarchy (SSH), a framework for representing large-scale space at multiple levels of abstraction, and introduces its extension, the Hybrid Spatial Semantic Hierarchy (HSSH) [Beeson et al., 2003; Kuipers et al., 2004]. We give a formal description of the multiple levels of knowledge of the SSH as they pertain to map-building (the most difficult navigation task). This formal description is an abridged version of the logical formalism given by Remolina [2001], with a few slight changes that allow us to make logical connections to the spatial concepts that are introduced via the HSSH. We then discuss some of the limitations of the basic SSH approach, which motivates the need to integrate small-scale space models that will benefit all levels of the SSH framework (Chapters 4 through 9).

3.1 Hybrid SSH Overview

Most metrical approaches to mobile robot map-building define a single, global frame of reference in which to create the map. Range measurements are used to perform probabilistic inference about the location of features or about the occupancy of discretized cells in the map. Existing SLAM (simultaneous localization and mapping) methods are highly effective for building local metrical models of small-scale space and for providing reliable localization in the frame of reference of the local map; however, maintaining global consistency over large-scale environments is difficult, particularly when closing large loops in the environment. A popular approach is to use particle filters, where each particle represents a hypothesized exploration trajectory. The researcher must hope that with enough particles the distribution will include one that closes the loop correctly [Thrun et al., 2005]. Since the space of trajectories can be enormous, this hope is often optimistic.

The fundamental problem is representational: loop-closing hypotheses are alternative topological structures for the map, not alternative metrical structures. To be able to solve complex, multi-hypothesis loop-closing problems in a tractable manner, the robot must reason with symbolic topological maps. The space of metrical maps in a single frame of reference does not appropriately represent the states of incomplete knowledge that arise during exploration and map-building in complex, large-scale environments.

Our factored mapping framework is based on the Spatial Semantic Hierarchy (SSH) [Kuipers, 2000, 2008], which uses multiple coordinated representations for knowledge of large-scale space. The *Hybrid SSH* (HSSH) extends the basic SSH by including representations for small-scale space and defining the relationship between large-scale and small-scale spatial representations. Symbolic topological mapping methods such as the SSH provide a concise representation for the structural alternatives that arise in investigating loop closures. Topological maps provide the ability to store and access multiple local maps with separate frames of reference and topological connections annotated with weak metrical constraints. By separating small-scale from large-scale space, we postpone the problem of coordinating the local frames of reference until the global structure of the topological map has been identified. At that point, the global metrical map can be constructed, efficiently and accurately.

Therefore, our approach factors the mapping problem into four natural sub-goals:

(1) building a metrical representation for local small-scale spaces; (2) detecting places and determining their symbolic descriptions; (3) finding a topological map representing the qualitative structure of large-scale space; and (4) constructing a metrical representation for large-scale space in a single global frame of reference, building on the skeleton provided by the topological map. While the global metrical map is useful for some purposes, it is worth noting that many autonomous planning and navigation goals can be achieved effectively using only the global topological map and/or the local metrical maps. Therefore, this approach to hybrid mapping is more robust than one that extracts topological relations from a global metrical map that must be built first [Thrun and Bücken, 1996].

The multiple representations of the HSSH are described independently, while their semantic dependencies imply that they build on each other. However, this does *not* imply a simple serial processing pipeline. In fact, processing of sensory input to build representations of the different kinds is interleaved, providing various sorts of synergies. Two are particularly important. First, the local metrical map of small-scale space is a useful "observer" both for detecting and describing places and for low-level control with obstacle avoidance. And second, metrical knowledge of relative displacement and then layout of places may be useful for ordering candidate topological maps. Nonetheless, to clarify the distinct representational ontologies, we often describe them in this dissertation as though they operate independently.

The Hybrid SSH improves mobile robot capabilities in a variety of ways—efficient and robust map-building and navigation, "natural" human-robot interaction due to the multiple representations of space (Chapter 7), and hierarchical control. This thesis describes the HSSH theory and demonstrates key points of HSSH map-building using a particular implementation that focuses on perception using range sensors, though other sensory modalities can also be utilized in the HSSH framework [Murarka et al., 2006]. The hybrid, hierarchical framework is largely independent of the sensors used to create the local metrical model of small-scale space.

3.2 The Basic SSH

The Spatial Semantic Hierarchy (SSH) [Kuipers, 2000, 2008] represents knowledge of large-scale space with four distinct representations. Figure 3.1 illustrates the framework. At the SSH Control Level, control laws provide reliable motion among *distinctive states* (dstates) q_i. At the SSH Causal Level, state-action-state *schemas* $\langle q, a, q' \rangle$ explain how the distinctive states are linked by turn and travel *actions*, and relations $o(q) = v$ between a state and its observable *view* describe the potential experiences of the robot. Thus the Causal Level abstracts the continuous world to a deterministic finite automaton [Rivest and Schapire, 1989; Dean et al., 1995], related to the way humans utilize route instructions in navigation. At the SSH Topological Level, a map consisting of places, paths, and regions, describes the connectivity, order, containment, and boundary relations of large-scale environments. At the SSH Metrical Level, local metrical information about the location of obstacles, the magnitudes of actions, the lengths of path segments, and the directions of paths at place neighborhoods are incorporated into local and global metrical maps. One contribution of the Hybrid SSH is to clarify the relation between the metrical information and the symbolic abstractions of the basic SSH levels.

The Spatial Semantic Hierarchy factors spatial uncertainty into distinct components, controlled in distinct ways. *Movement uncertainty* is controlled by the behavior of feedback-driven motion control laws. *Pose uncertainty* is controlled in the basic SSH by hill-climbing to dstates (and in the Hybrid SSH by incremental localization within a local metrical map). *Structural ambiguity* about the large-scale topology of the environment is controlled by search in a space of alternative topological maps. *Global metrical uncertainty* is controlled by relaxing metrical information from separate local frames of reference into a single global frame of reference, guided by the topological map.

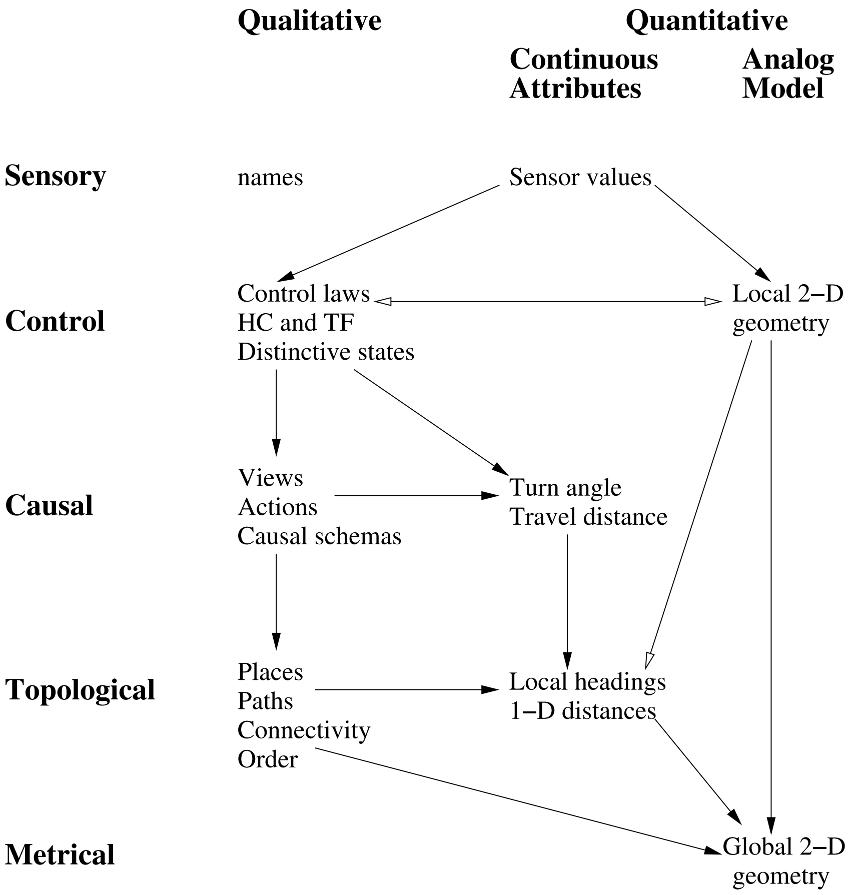


Figure 3.1: *The Spatial Semantic Hierarchy.* Closed-headed arrows represent dependencies; open-headed arrows represent potential information flow without dependency. (From Kuipers [2000].)

3.2.1 The SSH Control Level

The SSH Control Level describes the system consisting of the agent and its environment as a piecewise continuous dynamical system. The agent's experience is represented at a fine-grained sequence of time-steps $0 \leq t \leq N$. At any time t, the agent-environment system is described by the state vector x_t (the agent's *pose* in a static world), the agent's sense vector z_t, and its motor vector u_t. We assume that both the environment and the agent's sensory system are very rich, so the sense vector z_t is very high-dimensional.

The dynamical system is described by the following equations, in which the functions F and G represent the physics of the agent's body in the environment and its sensorimotor system respectively. These two functions are not explicitly known or available to the agent. The control law H_i, on the other hand, can be selected by the agent.

$$\begin{aligned} x_{t+1} &= F(x_t, u_t) \\ z_t &= G(x_t) \\ u_t &= H_i(z_t) \end{aligned}$$

The agent acts by selecting a control law H_i to determine its motor output signals as a function of its sensor input. In the basic SSH [Kuipers, 2000], motion is controlled by alternating between two types of controllers. *Trajectory-following* control laws take the robot from one *distinctive state* (dstate) to the neighborhood of another. A *hill-climbing* control law guides the robot to the destination dstate $\bar{x}$ from anywhere in its surrounding neighborhood.

Hill-climbing localizes the agent by moving it reliably to a distinctive state within the local neighborhood, preventing the accumulation of position error, and paving the way for a discrete abstraction of the continuous space. Furthermore, hill-climbing control makes very weak assumptions about the properties of the sensors and the agent's knowledge of those properties; however, hill-climbing control laws can be difficult to define and may vary across domains. An agent often *does* have stronger knowledge of the properties of its sensorimotor system, and physical motion to distinctive states seems awkward and unnecessary in light of that knowledge. A key insight behind the Hybrid SSH is that accurate localization in the small-scale space model of a place neighborhood can substitute for the physical motion of hill-climbing to a particular distinctive state in that neighborhood. In Chapter 4, we discuss how the Hybrid SSH exploits metrical knowledge of small-scale space to build *local perceptual maps* of place neighborhoods, within which localization is reliable and effective.

3.2.2 The SSH Causal Level

Given pairs of trajectory-following (TF) and hill-climbing (HC) controls that represent motion between neighboring dstates at the Control Level, we begin to represent the robot's experiences as a set of symbolic abstractions (Figure 3.2). First, we define an *action* $a \in A$, to represent a pair of trajectory-following and hill-climbing controls that connect dstates. Since the sensory image at a dstate $\bar{z} = G(\bar{x})$ is a point in a very high-dimensional space, it will, in general, never be experienced twice. We will therefore assume that each distinctive state $\bar{x}$ has an associated *view*, $o(\bar{x}) = v \in V$, which is an abstracted description of the sensory image $\bar{z}$. Kuipers and Beeson [2002] describe a bootstrap-learning method for obtaining a view representation suitable for high-performance place recognition. For this thesis, we will not require that the observation function o be learned autonomously, but only that it exists.

The SSH Causal Level describes the agent's experience as a deterministic finite automaton (DFA) [Rivest and Schapire, 1989; Dean et al., 1995]. The Causal Level deterministic finite automaton

$$M^C = \langle Q, A, V, R, o \rangle$$

consists of sets of states Q, actions A, observable views V, a transition function $R : Q \times A \to Q$, and an observation function $o : Q \to V$. As the robot travels from one distinctive state $\bar{x}$

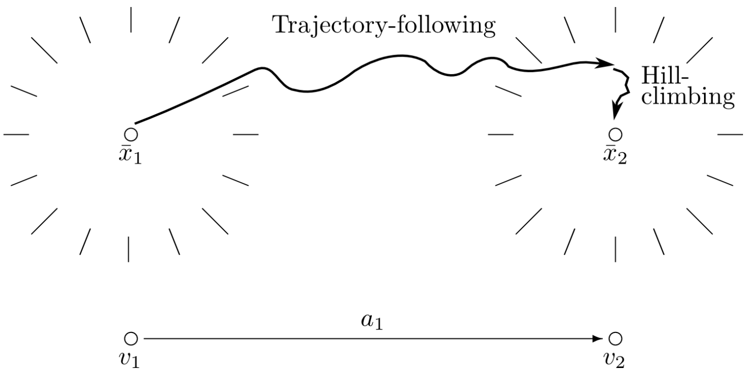


Figure 3.2: *SSH Control/Causal Level abstraction.* In the SSH, dstates are defined by pairs of trajectory-following and hill-climbing control laws. These sequences are abstracted into actions, and the observations at dstates are abstracted into views. (Adapted from Kuipers [2000].)

to the next, its experience is an alternating sequence of views and actions. Some actions are turns, while others are travels.

$$v_0 \quad a_1 \quad v_1 \quad a_2 \quad v_2 \quad \cdots \quad v_{n-1} \quad a_n \quad v_n$$

At the SSH Control Level, a view v_i is experienced only when the agent is at a distinctive state $\bar{x}_i$, so the view v_i is an observable manifestation of the distinctive state: $v_i = o(\bar{x}_i)$.

$$\begin{array}{ccccccccc}
\bar{x}_0 & a_1 & \bar{x}_1 & a_2 & \bar{x}_2 & \cdots & \bar{x}_{n-1} & a_n & \bar{x}_n \\
| & & | & & | & & | & & | \\
v_0 & & v_1 & & v_2 & \cdots & v_{n-1} & & v_n
\end{array}$$

At the Causal Level, each state $q \in Q$ represents an equivalence class of distinctive states $\bar{x}$ in the physical world.[1] Two distinctive states $\bar{x}_i$ and $\bar{x}_j$ are equivalent if they represent different experiences of the same distinctive state $q \in Q$. (We use the notation $[\bar{x}_i] = [\bar{x}_j] = q$ for this.) The set Q of distinctive states thus represents a specific hypothesis about which

---

[1] The term *distinctive state*, abbreviated *dstate*, is thus overloaded. It refers both to the state $\bar{x}$ resulting from a hill-climbing control law at the SSH Control Level, and to the state $q = [\bar{x}]$ at the SSH Causal Level which is part of the discrete abstraction of the continuous environment. It is this abstraction from continuous to symbol that facilitates causal/topological mapping in the basic SSH.

experiences $\bar{x}_i$ represent repeated encounters with the same state q in the environment; that is, Q specifies *data association* for loop closures.

All distinctive states in the same equivalent class q must have the same view.[2]

$$[\bar{x}] = [\bar{x}'] \rightarrow o(\bar{x}) = o(\bar{x}')$$

Thus, $o(q)$ is well-defined, and we can write

$$q = q' \rightarrow o(q) = o(q')$$

The full sensory input from high bandwidth sensors in a realistically complex environment is so rich that sensory images will never match exactly. Views must be defined in terms of some observation function that allows the same dstate to be reliably detected on separate occasions. Thus, an experience with repeated states such as

$$\begin{array}{ccccccccc} q_0 & a_1 & q_1 & a_2 & q_2 & \cdots & q_0 & a_n & q_1 \\ | & & | & & | & & | & & | \\ v_0 & & v_1 & & v_2 & \cdots & v_{n-1} & & v_n \end{array}$$

can only be consistent if $v_{n-1} = v_0$ and $v_n = v_1$. However, abstracted views are subject to perceptual aliasing (different places look the same), leading to ambiguities about the topological structure of the map: $o(q) = o(q') \not\rightarrow q = q'$.

The *transition function* $R : Q \times A \rightarrow Q$ is represented as a set of *schemas* $r = \langle q, a, q' \rangle$, where $\mathit{context}(r) = q$, $\mathit{action}(r) = a$, and $\mathit{result}(r) = q'$. As new observations are added to the robot's experience, new schemas $\langle [\bar{x}_n], a_{n+1}, [\bar{x}_{n+1}] \rangle$ are learned by the transition function R. The causal map is constructed by searching for an appropriate set Q of

---

[2] The axioms provided here describe the nature of the spatial knowledge represented at each SSH level, but we omit auxiliary axioms required for logical completeness (e.g., unique names axioms, etc). A complete set of axioms is provided by Remolina and Kuipers [2004]. For clarity and conciseness, we use a typed logic in which variable names encode their types, and we assume that all free variables in axioms are universally quantified.

states (i.e., equivalence classes of distinctive state observations), such that M^C has a deterministic transition function R, predicted and observed views are consistent, and M^C is consistent with the axioms for topological maps [Remolina and Kuipers, 2004].

For the purpose of building the SSH Causal Level from exploration experience, building and using a DFA is far more tractable than building and using a probabilistic state model, such as a hidden Markov model (HMM) (as discussed in Section 2.2.3). The key benefit of a DFA over HMMs (or stochastic finite automata in general) are that both the transition function and the observation function are deterministic. The deterministic transition function follows from the nature of the abstraction that results from moving reliably between dstates via TF and HC control laws. The deterministic observation function follows from the abstraction that defines the observation function o. One improvement of the HSSH over the SSH is that local small-scale space models make place detection and observational classification of states deterministic without the need for hill-climbing (Section 5.1).

The effect of the SSH hill-climbing (and HSSH place detection and localization) is that the Causal Level representation can assume that actions are deterministic. The determinism of the observation function rests on the abstraction from sensory images to views being sufficiently aggressive to eliminate perceptual *variability*. Although observations are deterministic, they are not necessarily unique since there may still be perceptual aliasing. This ambiguity is handled by creating multiple hypotheses of topological (thus causal) models, as explained in Section 3.2.3. In general, it is not possible for a robot to recover the complete spatial structure of any arbitrary environment [Dudek et al., 1991]; therefore, keeping around the tree of possible maps allows the robot to continue navigation when the best hypothesis is refuted by an experienced counter example.[3]

---

[3] Long-term experience with the HSSH has yielded deterministic actions and views in research settings, but we can envision rare scenarios, e.g., an intersection crowded with people, that could lead to an undetected or misclassified place. Detecting and understanding these unusual events should allow us to still assume deterministic actions $(100-\varepsilon)\%$ of the time.

3.2.3 The SSH Topological Level

In the SSH, a topological map is an instantiated model for two sets of axioms: one that describes topological maps in general and another that describes the exploration experience of the agent in a particular environment. We identify the global topological map by generating potential models of these axioms, discarding those that violate the axioms, and applying an ordering on the remaining ones so that a single best model can be selected. If there is no single best model, then a few closely competing models can be identified and can be used to make an exploration plan to help discriminate between models.

The SSH Topological map M^T describes the environment in terms of dstates, places, paths, regions, and the qualitative relations among them such as connectivity, order, and containment. A dstate $q \in Q$ represents a distinctive state or pose of the agent in the environment, a place $p \in P$ represents a zero-dimensional location, a path $\pi \in \Pi$ represents a one-dimensional structure, and a region $r \in R$ represents a two-dimensional subset of the environment. In this dissertation, we will not discuss regions or their relations, which are described by Remolina [2001].

We formalize a topological map as

$$M^T = M^C \cup Objects \cup Relations.$$

Here $Objects = \langle P, \Pi, R \rangle$, where P is a set of places, Π is a set of paths, and R is a set of regions. M^T thus includes (via M^C) the sets of states, actions, and views. *Relations* encodes the relations over this set of objects. These relations, including *at*, *along*, *on*, *order* (and the “star” relations that describe the local topology of places), allow for a richer description of the connectivity of places and paths, and are introduced below as needed.

At the SSH Causal Level, the experience is represented as an alternating sequence of states ($q_i \in Q$) and actions ($a_j \in A$).

$$q_0 \quad a_1 \quad q_1 \quad a_2 \quad q_2 \quad \cdots \quad q_{n-1} \quad a_n \quad q_n$$

At the Topological Level, each distinctive state $q \in Q$ corresponds to being *at* a place, and facing *along* a path in some direction. Since a path is one-dimensional, it has two directions $d \in \{+,-\}$, for which $opp(+) = -$ and $opp(-) = +$. We define a directed path, π^d, to represent facing along a path in a particular direction.

$$\forall q \in Q \; \exists p \in P,\; \pi \in \Pi,\; d \in \{+,-\} \; [at(q,p) \;\wedge\; along(q,\pi,d)] \tag{3.1}$$

$$along(q,\pi^d) \equiv along(q,\pi,d)$$

Additionally, there are two kinds of basic actions, turns and travels, and there is a TurnAround action.

$$A = Turns \cup Travels \qquad TurnAround \in Turns$$

A place $p \in P$ corresponds to a set of states linked by turn actions.

$$\langle q,a,q' \rangle \in S \wedge a \in Turns \wedge at(q,p) \rightarrow at(q',p) \tag{3.2}$$

Similarly, a path $\pi \in \Pi$ corresponds to a set of states linked by travel actions, or by a TurnAround.

$$\langle q,a,q' \rangle \in S \wedge a \in Travels \wedge along(q,\pi,d) \rightarrow along(q',\pi,d) \tag{3.3}$$

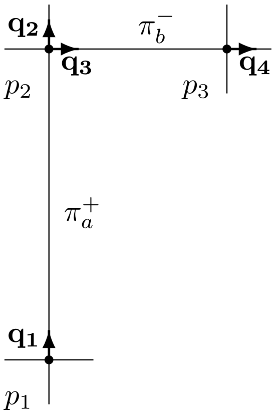
$$\langle q,a,q' \rangle \in S \wedge a = TurnAround \wedge along(q,\pi,d) \rightarrow along(q',\pi,opp(d)) \tag{3.4}$$

The relation $on(\pi,p)$ means that the place $p \in P$ is on the path $\pi \in \Pi$.

$$at(q,p) \wedge along(q,\pi,d) \rightarrow on(\pi,p) \tag{3.5}$$

A path defines an order relation over the places on it:

$$\langle q,a,q' \rangle \in S \wedge a \in Travels \wedge at(q,p) \wedge at(q',p') \wedge along(q,\pi,d) \rightarrow order(\pi,d,p,p') \tag{3.6}$$



Causal schema	**Equation #**	**Topological relation**
$\langle q_1, travel, q_2\rangle \in S$	(3.1) →	$at(q_1, p_1)$, $along(q_1, \pi_a^+)$
	(3.5) →	$on(\pi_a, p_1)$
	(3.3) →	$along(q_1, \pi_a^+)$, $along(q_2, \pi_a^+)$
	(3.1) →	$at(q_2, p_2)$
	(3.5) →	$on(\pi_a, p_2)$
	(3.6) →	$order(\pi_a^+, p_1, p_2)$
	(3.8) →	$p_1 \neq p_2$
$\langle q_2, turn, q_3\rangle \in S$	(3.2) →	$at(q_3, p_2)$
	(3.1) →	$along(q_3, \pi_b^-)$
	(3.5) →	$on(\pi_b, p_2)$
$\langle q_3, travel, q_4\rangle \in S$	(3.3) →	$along(q_4, \pi_b^-)$
	(3.1) →	$at(q_4, p_3)$
	(3.5) →	$on(\pi_b, p_3)$
	(3.6) →	$order(\pi_b^-, p_2, p_3)$
	(3.8) →	$p_2 \neq p_3$

Figure 3.3: *Topological abduction example.* Here we illustrate the abduction process, using the topological axioms to model exploration. Starting at dstate q_1, the agent reaches dstate q_2 at a place p_2 having traveled along directed path π_a^+. It then turns to dstate q_3, still at place p_2, and is ready to travel along another path, say π_b^-, from q_3 to dstate q_4 at some other place.

$$order(\pi, d, a, b) \rightarrow on(\pi, a) \wedge on(\pi, b) \tag{3.7}$$

$$\neg order(\pi, d, p, p) \tag{3.8}$$

$$order(\pi, d, a, b) \iff order(\pi, opp(d), b, a) \tag{3.9}$$

$$order(\pi, d, a, b) \wedge order(\pi, d, b, c) \rightarrow order(\pi, d, a, c) \tag{3.10}$$

In order to create a Topological Level map from a Causal Level experience, such as $\langle q_1, travel, q_2\rangle$, $\langle q_2, turn, q_3\rangle$, $\langle q_3, travel, q_4\rangle$, the agent uses abduction to hypothesize the existence of several places and paths at which these distinctive states occur. Figure 3.3 shows an example of the abduction process.

Remolina and Kuipers [2004] provide a non-monotonic axiomatization of the SSH topological map, including additional elements of the theory (regions, boundary relations, and metrical relations), along with more details and motivating examples. This theory provides a precise specification of the possible logical models (topological maps) that are consistent with the axioms and the sequence of actions and views observed while exploring. A prioritized circumscription policy (expressed as a nested abnormality theory [Lifschitz, 1995]) specifies how distinct consistent logical models are ordered by simplicity. Furthermore, Savelli and Kuipers [2004] have developed the non-local *planarity constraint*, which enforces the requirement that a topological map is a graph embedded in the plane. Algorithm 3.1 presents the process for constructing all possible topological maps by generating all possible sets Q.

3.2.4 The SSH Metrical Level

The SSH, often thought of as a framework for creating purely topological maps, has always allowed for local metrical knowledge to be utilized at the Control Level (Figure 3.1, right column). Additionally, the SSH Metrical Level has always supported a global metrical map to be created *after* the topological map—it is our belief that such a global metrical map is often unnecessary for navigation in and communication about large-scale environments. However, the SSH theory has lacked a formal description of exactly how metrical information influences the hierarchical abstractions of space. One contribution of this thesis is to clarify the relationships between metrical and symbolic knowledge in a navigational agent.

In work leading to the development of the SSH, Kuipers and Byun [1991] created a “patchwork metrical map”. Their mapping implementation annotated topological places and paths with metrical data gathered during exploration. Given a topological map hypothesis, the global place layout was relaxed to minimize errors with respect to the annotated metrical data before adding stored range information to create the obstacle map. This approach is similar to the techniques we define mathematically in Chapter 9.

0. Perform initial action a_0 that brings the robot to a place along a directed path. Initialize the tree of maps with the map hypothesis $\langle M_0, q_0 \rangle$, where M_0^C contains the single dstate q_0 with its observed view v_0, and M_0^T contains the single place p_0 and path π_0.

After performing a new action a and observing the resulting view v, for each consistent map $\langle M, q \rangle$ on the fringe of the tree:

1. If M^C includes $\langle q, a, q' \rangle$ in R and $v' = o(q')$,
   - if $match(v, v')$, then $\langle M, q' \rangle$ is the successor to $\langle M, q \rangle$, extending the tree;
   - if not, then mark $\langle M, q \rangle$ as inconsistent.
2. Otherwise, M^C does not include $\langle q, a, q' \rangle$ in R. Let M' be M extended with a new distinctive state symbol q' and the assertions $v = o(q')$ and $\langle q, a, q' \rangle$. Consider the $k \geq 0$ dstates q_j in M with $v_j = o(q_j)$, such that $match(v_j, v)$. Then $\langle M, q \rangle$ has $k+1$ successors:
   - $\langle M'_j, q' \rangle$ for $1 \leq j \leq k$, where M'_j is M' extended with the assertion $q' = q_j$.
   - $\langle M'_{k+1}, q' \rangle$, where M'_{k+1} is M' extended with the k assertions that $q' \neq q_j$, for $1 \leq j \leq k$.
3. Mark a new successor map inconsistent if it violates the axioms of topological maps.
4. Define a preference order on the consistent maps at the leaves of the tree.

In the Basic SSH:

$M = M^T$.
A view is a simple symbol.
$match(v, v')$ iff $v = v'$.
Both $a \in Turns$ and $a \in Travels$ can reach step 2 and cause a branch.
Preference order from prioritized circumscription policy [Remolina and Kuipers, 2004].

Algorithm 3.1: *Building the tree of topological maps in the SSH.* This describes the algorithm for building a tree of all possible topological consistent with a sequence of actions and observations at discrete places.

3.3 Extending the SSH

The Spatial Semantic Hierarchy depends on the assumption that the environment naturally decomposes into place neighborhoods, connected by path segments, which can then be abstracted to a topological map. That is, it uses the sparse structure of man-made environments (or man-made paths in natural environments) to define a small number of discrete places and connecting paths. Obviously, topological structure may be imposed even in unstructured environments. Defining places at visually distinctive locations along a single path (e.g., a water tower on the side of a highway) or even based on metrical path-integration in wide-open spaces (as the Puluwat navigators do when piloting dugout canoes between distant islands [Gladwin, 1970]) are currently not handled by our SSH hill-climbing controllers or the HSSH place detection methods. We believe these type of places can be represented within the SSH framework, but we leave this problem for future work.

The basic SSH makes weak (i.e., very general) assumptions about the sensory capabilities of the navigational agent; thus, abstraction from continuous sensations to discrete models of the environment depends on well-crafted control laws that move the robot reliably between distinctive states. The Hybrid SSH makes stronger (i.e., more specific) assumptions about the types of sensors available to the agent, for example, range sensors. This allows the HSSH to extend the basic SSH by using existing metrical mapping techniques to create precise observational models of the local surround.

The HSSH has four major levels of representation that correspond to the four SSH levels (Figure 3.4). At the *Local Metrical Level*, the agent builds and localizes itself in the Local Perceptual Map (LPM), a metrically accurate map of the local space within its sensory horizon. The LPM is used for local motion planning and hazard avoidance. At the *Local Topological Level*, the agent identifies discrete places (e.g., corridor intersections, rooms, etc.) in the large-scale continuous environment, and qualitatively describes the configuration of the paths through the place—its local decision structure. At the *Global Topological Level*, the agent resolves structural ambiguities and determines how the environment is best described as a graph of places, paths, and regions. The *Global Metrical Level* specifies the layout of places, paths, and obstacles within a single global frame of reference. It can be built on the skeleton provided by the topological map. Figure 3.4 diagrams the basic flow of data in the HSSH, from sensors, through the local metrical model of small-scale space and the global symbolic model of the large-scale environment, finally creating the global metrical model if desired.

Having small-scale space models of the local surround creates several advantages when implementing the HSSH versus the basic SSH. First, the robot represents the local environment using a *local perceptual map (LPM)* (Chapter 4). The robot can therefore use algorithms for local metrical motion planning and obstacle avoidance instead of relying on behavior-based controllers. Second, metrical localization can be done quickly after entering a place neighborhood, rather than requiring physical hill-climbing to a distinctive pose.

A symbolic description of the *local topology* is extracted from this precise small-scale-space model of the local surround via *gateways* (Chapter 5). Thus, the view of a distinctive state no longer need be some user-defined function of the perceptual inputs. Instead, the method relies on the local topology abstracted from the LPM to describe places, thus describing all distinctive states at each place. Using local topology to *detect and describe* places allows the robot to model more complicated intersections of paths than with hill-climbing. Additionally, using local topology constrains the global topological model search (Chapter 8), as branching in the tree of possible maps occurs only when arriving at a place, not when visiting the various dstates of a place.

Stored metrical information along topological connections between places can be used to efficiently obtain a global metrical layout of places (Chapter 9), which provides the "backbone" for a global map if desired. The HSSH also improves navigational behaviors and facilitates multi-modal human-robot interaction (Chapter 7).

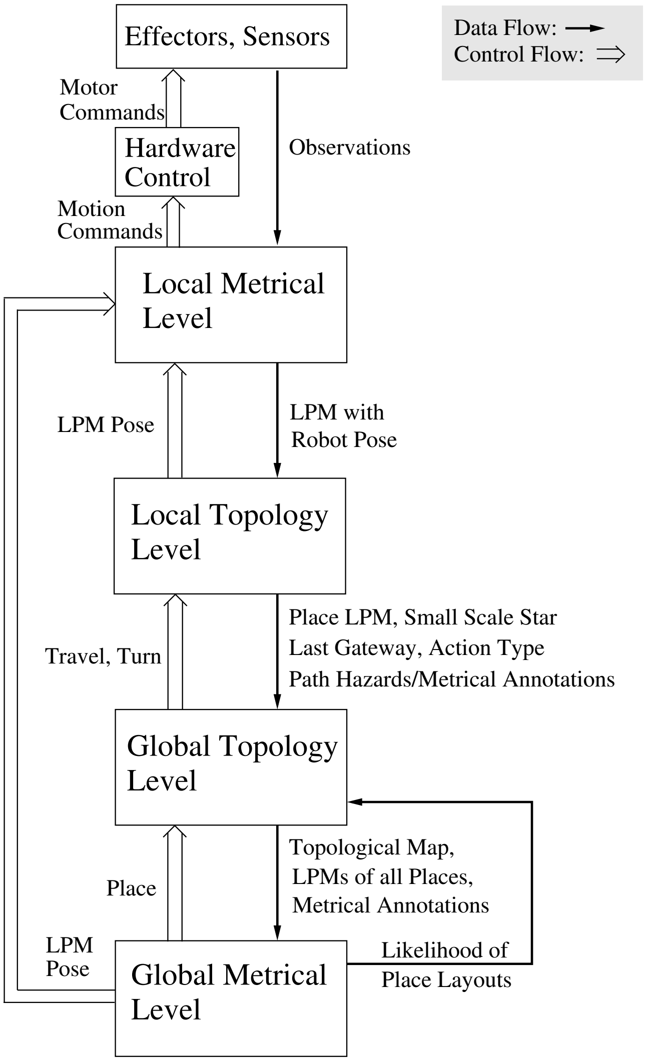


Figure 3.4: *HSSH description.* The HSSH is an integrated framework of multiple, disparate representations of spatial knowledge. Each level of abstraction uses its own ontology with concepts motivated by human cognitive abilities and grounded to the environment via local metrical observations. The four major components here correspond to the four levels of the basic SSH shown in Figure 3.1.

Chapter 4

Modeling Obstacles and Trajectory: The HSSH Local Metrical Level

> *The knowledge of the world is only to be acquired in the world, and not in a closet.*
>
> Philip Dormer Stanhope, 4th Earl of Chesterfield, *Letters to His Son* (1774)

The critical difference between the basic SSH and the Hybrid SSH is the use of a local metrical model to represent small-scale space surrounding the robot. We call this model a local perceptual map (LPM). The LPM is currently built using sensor input from laser range sensors that see walls, but the LPM could be built from visual sidewalk (or road) detection or other sensor modalities. Similarly, the current LPM representation models occupied, free, and unknown regions of space. Work by Murarka et al. [2006] investigates incorporating semantic labels into the LPM to denote drop offs, pedestrians, and other types of hazards.

This chapter discusses the characteristics and benefits of LPMs in the HSSH. One important implementation detail is to ensure a quality local map even in the face of slipping wheels, inaccurate odometry, or other factors that can affect localization performance. Improper localization leads to an incorrect update of the local map, which can possibly degrade the benefits of the LPM structure throughout the HSSH. We evaluate an *adaptive* *particle filter* algorithm for improved localization in the face of odometry errors [Beeson et al., 2006].

4.1 Local Perceptual Map (LPM)

The *local perceptual map* (LPM) is a bounded-size metrical description of the small-scale space surrounding the agent. It functions as an *observer*, integrating sensor values over time to determine the locations of obstacles and other hazards, for localization, motion planning, and the derivation of local features for larger-scale mapping. The LPM represents the small-scale space within the robot's sensory horizon, not just what is currently in view. It is small enough to avoid the problem of closing large loops. The frame of reference of the LPM is local. Its relation with the world frame may be unknown, and will drift over time due to accumulating errors.

When the agent travels from one place to another, the LPM acts as a *scrolling map*, $\tilde{m}$, that describes the robot's immediate surround. Information that scrolls off the LPM is discarded, and new cells that scroll onto the map are initialized as $\mathsf{unknown}$.[1] Because the LPM has a fixed, bounded size, the cost of updating it is constant in both time and space.

The full task of building metrical maps from exploration data can be described as finding the joint posterior over maps m and trajectories $x = (\mathsf{x}, \mathsf{y}, \theta)$ in $P(x, m | z, u)$ with the following symbol definitions.

---

[1] Our rectangular LPM scrolls, horizontally or vertically, as needed to keep to keep the robot's pose in a central cell. Information in the occupancy grid is only shifted by integral numbers of cells to avoid blurring the model by rotations or partial-cell translations.

t	The time-steps $0 \leq t \leq N$ of the agent's experience.
$x = x_{0:N}$	The sequence of agent poses x_t at each time-step t.
$z = z_{0:N}$	The sequence of observations z_t.
$u = u_{1:N}$	The sequence of actions u_t between time-steps.
m	The set of map elements, which may be landmarks or occupancy grid cells. $\tilde{m}$ (the scrolling local perceptual map) is a particular example of a metrical map m.

The joint probability of the pose history x and the metrical map m can be decomposed as

$$P(x,m|z,u) = P(m|x,z,u) \cdot P(x|z,u)$$

by the chain rule for probabilities.

For simple, local regions, the maximum-likelihood map can be estimated incrementally given knowledge of x and z, so we really just need to solve for $P(x|z,u)$. Additionally, we are not concerned with the full distribution over pose trajectories, as we are updating the map from the maximum-likelihood pose at each time step. Thus, for our online metrical mapping we only need to determine the distribution over the current pose.

$$\begin{aligned} Bel(x_t) &= P(x_t|z_{0:t},u_{1:t}) \\ &= \eta \; P(z_t|x_t,m) \int P(x_t|x_{t-1},u_t) Bel(x_{t-1}) \, dx_{t-1}, \end{aligned}$$

where η is a normalization constant. Figure 4.1 illustrates the basic structure of Markov localization [Fox et al., 1999], which allows us to determine in an efficient and incremental algorithm, the distribution of poses that best fit the current map.

Though multiple metrical mapping methods might be used for the LPM, we utilize the well-known occupancy grid representation [Moravec, 1988; Elfes, 1989], along with particle filter Markov localization [Fox et al., 1999] to overcome noisy odometry information. Stated more plainly, we model the world as a discretized grid, where each cell contains

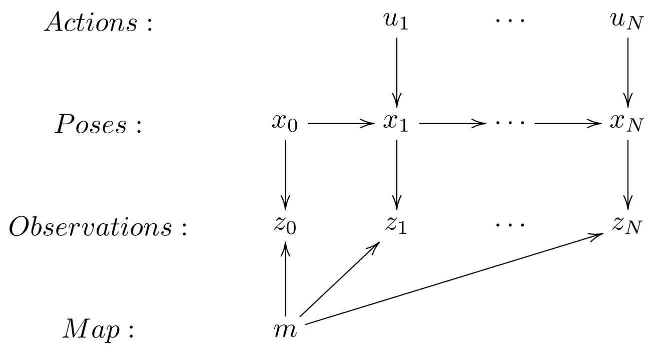


Figure 4.1: *Markov localization.* The standard graphical dynamic Bayesian network (DBN) for Markov localization within a single frame of reference: combines belief about actions $P(x_t|u_t, x_{t-1})$ and observation $P(z_t|x_t, m)$. Simultaneous localization and mapping (SLAM) algorithms combine localization, $P(x|z, u)$, with one of a number of mapping methods to estimate $P(x, m|z, u)$.

a probability of being occupied by an obstacle, as measured by a lidar sensor. Localization is performed by comparing hypothesis poses to the current map, and the map is updated accordingly. This is a well-known version of simultaneous localization and mapping (SLAM) [Thrun et al., 2005]. Discussions in this dissertation that refer to this implementation generalize to many SLAM implementations.

4.2 LPM Benefits

LPMs provide the HSSH with various information that allows both local and global abstractions of space. In Chapter 5, we discuss how the LPM supports the abstraction of a symbolic small-scale space description of the local-paths in the surround. Chapter 9 discusses how the local metrical information is used, along with the topological map, to find the global metrical place layout of an exploration trace and, if desired, the entire global metrical map on an explored environment. We forgo these details here; however, the LPM is also useful for local control at the SSH Control Level.

Given a target pose in the LPM, the robot can compute a trajectory to reach the target without colliding into obstacles. This can be done using the Vector Field Histogram [Borenstein and Koren, 1991], the Dynamic Window approach [Fox et al., 1997], gradient methods [Konolige, 2000], or even a simple search (using A* or RRTs [Kuffner and LaValle, 2000]) over the cells of the occupancy grid. The potential function (for gradient methods) or the cost function (for A*) reflects the distance of the agent from an obstacle or other hazard represented in the LPM. Object tracking may be implemented at this level, but our current robot implementation simply avoids obstacles by taking the first few steps along the planned trajectory before replanning. We discuss the selection of target poses as they apply to Causal Level *Travels* and *Turns* in Section 5.4.

In the basic SSH [Kuipers, 2000], an agent localizes itself in a place neighborhood by hill-climbing to a distinctive state. Localization by physically moving to maximize a "distinctiveness measure" requires very little knowledge about the nature of the environment or the sensors. In the Hybrid SSH, on the other hand, the agent uses an online SLAM method to localize itself unambiguously within the local perceptual map. SLAM methods depend on stronger knowledge about the relation between sensor input and the agent's location in the local frame of reference—$P(z_t|x_t,\tilde{m})$. In return for these stronger assumptions, the agent does not need to move to a particular location to be adequately localized.

Finally, when the agent is in the neighborhood of a particular topological place p, a snapshot of the LPM $\tilde{m}$ serves as a small-scale space description of the place neighborhood that is stored as a place annotation m_p in the topological map. When a place p is first encountered, the local map m_p for its neighborhood is initialized with the information from the scrolling map $\tilde{m}$. The frame of reference defined for m_p may be different from that of $\tilde{m}$, appropriate to the characteristics of the place neighborhood. When the neighborhood of p is encountered on subsequent occasions, the agent may localize itself with respect to the stored map m_p or update m_p with the more recent information in $\tilde{m}$.

4.3 Building an Accurate Localization Algorithm

In order to ensure a good quality LPM, we investigate extensions to the traditional particle filter algorithm for localization during mapping. Accurate localization is key to obtaining a good maximum-likelihood pose estimate before incorporating range measurements into the map. It is also important to keep a distribution over poses at each time step in order to ensure proper localization on future SLAM iterations.

The most commonly used paradigm for incremental localization is probabilistic localization [Thrun et al., 2005]. Here, the robot has some distribution of belief about its pose $x = (\mathsf{x}, \mathsf{y}, \theta)$. After performing a low-level control u and making an observation z, it must then determine its new pose estimate x'. Particle filters are an efficient approach to estimating a continuous distribution with a finite set of samples.

$$\begin{array}{rcl} p(x'|z,x,u,m) & = & \eta \; P(z|x',m) \cdot \int P(x'|x,u)p(x) \; dx \\ posterior & \propto & likelihood \cdot proposal \end{array}$$

When performing probabilistic localization using a particle filter, a robot must have a good *proposal distribution* in which to distribute its particles. Once weighted by their normalized *likelihood* scores, these particles estimate a *posterior distribution* over the possible poses of the robot. We investigate the results of different algorithms that modify the proposal distribution at each time step in order to obtain more accurate localization [Beeson et al., 2006]. Our resulting localization method gives a high quality LPM, but will also improve localization for any metrical mapping algorithm.

4.3.1 Action Models

The proposal distribution is determined by an *action model*. The action model is a system of equations that defines a probability distribution over resulting states x', given an initial state x and an incremental motion u, measured by odometry sensors. Using the observation, z, from the current location and a map of the local environment, m, the robot can create a likelihood distribution—a distribution over possible robot poses based on an observation model. Combining the likelihood and proposal distributions yields a posterior distribution that represents the actual uncertainty distribution of the robot's pose in the world.

Previous Work

An action model is important as it determines the location, shape, and span of each proposal distribution. There has been previous research into different differential drive action models and into how to calibrate them. Some research utilizes specific, contrived environments and constrained motion in order to disambiguate translational error from rotational error [Borenstein and Feng, 1996; Rekleitis, 2003]. In addition to constraining both the environment and the motion taken, these methods are run offline, using the final localization error to find the parameters of a static action model. If the underlying source of error changes (e.g., motors wear out, new equipment changes the load, or the travel surface changes), the researcher must run these experiments again to determine new model parameters.

Below, we discuss an approach that eliminates the need to re-tune the action model for each new environment type. The idea is to adapt the proposal distribution at each step based on observations, overcoming an incorrect action model. Before we discuss the adaptive particle filter, we define the action model we use.

Eliazar and Parr [2004] present a fairly complete sub-Cartesian action model, along with a nice offline tuning method given ground-truth. A sub-Cartesian action model uses the change in wheel shaft encoder counts between time $i-1$ and time i to calculate the incremental translational motion s_i and rotational motion ϕ_i.[2] These values are used to estimate a change in Cartesian space $(\mathsf{x}, \mathsf{y}, \theta)$.

Every pair of s_i, ϕ_i values corresponds to a change in the estimated pose $(\mathsf{x}, \mathsf{y}, \theta)$.[3] The most common way to calculate the pose is to assume that both the translation and rotation maintained constant velocity across the measured interval. For small time steps, this arc is closely approximated by assuming the robot sequentially performs half of its

---

[2] We assume that there is no opportunity for confusion between the values s, ϕ, and α used to describe the action model and their use in the SSH and HSSH formal descriptions. Terminology defined in the action model discussion has local scope.

[3] We use x to refer to the robot's pose in a planar model. We can decompose x into the $(\mathsf{x}, \mathsf{y}, \theta)$ components of the Euclidean space; thus, x is a dimension in the vector x and is distinguished in this discussion by using a different font.

rotation, all of its translation, then the other half of its rotation [Wang, 1988].

$$
\begin{aligned}
\mathsf{x}_i &= \mathsf{x}_{i-1} + s_i \cdot \cos(\theta_{i-1} + \frac{\phi_i}{2}) \\
\mathsf{y}_i &= \mathsf{y}_{i-1} + s_i \cdot \sin(\theta_{i-1} + \frac{\phi_i}{2}) \\
\theta_i &= \theta_{i-1} + \phi_i
\end{aligned}
$$

Eliazar and Parr [2004] go one step further than this traditional model and add another dimension of error that is normal to the direction of travel. This *drift* is represented by δ_i. Additionally, instead of using scalar values for s_i and ϕ_i (estimating only the mode of the uncertainty), this model tries to estimate the full distribution of error. They do this by assuming that $\tilde{s}_i$, $\tilde{\delta}_i$, and $\tilde{\phi}_i$ are Gaussians. This makes the change in Cartesian pose a distribution as well.

$$
\begin{aligned}
\tilde{\mathsf{x}}_i &= \tilde{\mathsf{x}}_{i-1} + \tilde{s}_i \cdot \cos(\tilde{\theta}_{i-1} + \frac{\tilde{\phi}_i}{2}) + \tilde{\delta}_i \cdot \cos(\tilde{\theta}_{i-1} + \frac{\tilde{\phi}_i + \pi}{2}) \\
\tilde{\mathsf{y}}_i &= \tilde{\mathsf{y}}_{i-1} + \tilde{s}_i \cdot \sin(\tilde{\theta}_{i-1} + \frac{\tilde{\phi}_i}{2}) + \tilde{\delta}_i \cdot \sin(\tilde{\theta}_{i-1} + \frac{\tilde{\phi}_i + \pi}{2}) \\
\tilde{\theta}_i &= \tilde{\theta}_{i-1} + \tilde{\phi}_i
\end{aligned}
$$

The Gaussians $\tilde{s}_i$, $\tilde{\delta}_i$, and $\tilde{\phi}_i$ are dependent on both s_i and ϕ_i. The problem now becomes how to determine the correct parameters, $c_0, \ldots, c_{11}$, that define these Gaussians.

$$
\begin{aligned}
\tilde{s}_i &= \mathcal{N}(s_i \cdot c_0 + \phi_i \cdot c_1,\ {s_i}^2 \cdot c_2 + {\phi_i}^2 \cdot c_3) \\
\tilde{\delta}_i &= \mathcal{N}(s_i \cdot c_4 + \phi_i \cdot c_5,\ {s_i}^2 \cdot c_6 + {\phi_i}^2 \cdot c_7) \\
\tilde{\phi}_i &= \mathcal{N}(s_i \cdot c_8 + \phi_i \cdot c_9,\ {s_i}^2 \cdot c_{10} + {\phi_i}^2 \cdot c_{11})
\end{aligned}
$$

Due to the large number of parameters, the authors tune the model by gathering a data set and calculating the *ground-truth* trajectory taken by the robot then solving a set of linear least squared systems to determine the parameters. Ground-truth is computed by overestimating the action model (to ensure good localization at each step) and running SLAM using a Rao-Blackwellized particle filter. Here, they can use many particles to reduce the chances of particle depletion, as this is done offline, where efficiency is not important.

Once they have ground truth, they can solve the six independent linear least squared systems. One system solves for the mean parameters of $\tilde{s}$ (c_0 and c_1) given the N localization steps in the data trace. **W** is an *NxN* diagonal matrix where each diagonal element equals the square-root of the likelihood of the winning particle at that time step. **O** is an *Nx2* matrix, where each row represents the measured s_i and ϕ_i for each time step. **C** is the column vector $[c_o\ c_1]^T$. **B** is the *Nx1* column vector representing the maximum-likelihood s value for each time step (the s value that lead to the winning particle). They then solve for **C** where $\mathbf{WOC} = \mathbf{WB}$. Using c_0 and c_1, another system solves for the variance parameters of $\tilde{s}$ (c_2 and c_3). Similarly, the other four linear systems solve for the parameters of $\tilde{\delta}$ and $\tilde{\phi}$ [Eliazar and Parr, 2004].

A New Action Model

The action model detailed above relies on having sub-Cartesian odometry information (s, ϕ). However, many localization implementations may not have access to this information. Pre-compiled odometry modules, robot simulators, and shared log files usually only provide Cartesian poses—$(\mathsf{x}, \mathsf{y}, \theta)$. This can be a problem for some researchers wanting to utilize the above models since there are many values of $\Delta(\mathsf{x}, \mathsf{y}, \theta)_i$ that yield no solution for s_i, ϕ_i (e.g., odometry "jumps" sideways for some reason).

To overcome this problem, we use the approach proposed by Rekleitis [2003]. Similar to the way the above action model approximates the constant velocity arc taken by the robot with a "turn-travel-turn" approach [Wang, 1988], we can break any change in pose into "turn-travel-turn" components—where the first turn α rotates to face the new location, the robot travels straight for distance ρ, and the robot makes a final turn β to its final

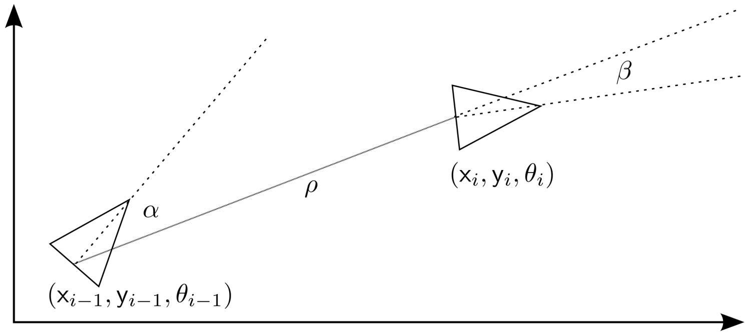

β
$(\mathsf{x}_i, \mathsf{y}_i, \theta_i)$
ρ
α
$(\mathsf{x}_{i-1}, \mathsf{y}_{i-1}, \theta_{i-1})$

Figure 4.2: *Generalized action model.* Given two successive odometry readings that give $\Delta(\mathsf{x}, \mathsf{y}, \theta)$, we can model the motion by a turn α, a translation ρ, and a second turn β.

orientation (Figure 4.2).

$$\begin{aligned}
\alpha_i &= \arctan(\frac{\Delta \mathsf{y}_i}{\Delta \mathsf{x}_i}) - \theta_{i-1}; \;\; \alpha_i \in [-\pi, \pi] \\
\rho_i &= \sqrt{\Delta \mathsf{x}_i{}^2 + \Delta \mathsf{y}_i{}^2} \\
\beta_i &= \Delta \theta_i - \alpha_i; \;\; \beta_i \in [-\pi, \pi]
\end{aligned}$$

When sitting still ($\rho_i = 0$), then $\alpha_i = 0$ and $\beta_i = \Delta\theta_i$. Because robots do not always translate forward, but can move backwards, we also consider the case where $\alpha_i' = \alpha_i + \pi$, $\alpha_i' \in [-\pi, \pi]$, $\rho_i' = -\rho_i$, $\beta_i' = \beta_i + \pi$, $\beta_i' \in [-\pi, \pi]$. We only use these values when $|\alpha_i'| + |\beta_i'| < |\alpha_i| + |\beta_i|$ (i.e., when less turning is needed when moving in reverse than when moving forward).

Using α, ρ, and β in stead of s and ϕ, we can extend the action model of Eliazar and Parr [2004]. This is the action model we use below to compare changes to the straightforward particle filter localization algorithm.

$$\begin{aligned}
\tilde{s}_i &= \mathcal{N}(\alpha_i \cdot c_0 + \rho_i \cdot c_1 + \beta_i \cdot c_2,\ c_3 + \alpha_i{}^2 \cdot c_4 + \rho_i{}^2 \cdot c_5 + \beta_i{}^2 \cdot c_6) \\
\tilde{\delta}_i &= \mathcal{N}(\alpha_i \cdot c_7 + \rho_i \cdot c_8 + \beta_i \cdot c_9,\ c_{10} + \alpha_i{}^2 \cdot c_{11} + \rho_i{}^2 \cdot c_{12} + \beta_i{}^2 \cdot c_{13}) \\
\tilde{\phi}_i &= \mathcal{N}(\alpha_i \cdot c_{14} + \rho_i \cdot c_{15} + \beta_i \cdot c_{16},\ c_{17} + \alpha_i{}^2 \cdot c_{18} + \rho_i{}^2 \cdot c_{19} + \beta_i{}^2 \cdot c_{20})
\end{aligned}$$

$$\begin{aligned}
\tilde{\mathsf{x}}_i &= \tilde{\mathsf{x}}_{i-1} + \tilde{s}_i \cdot \cos(\tilde{\theta}_{i-1} + \alpha_i) + \tilde{\delta}_i \cdot \cos(\tilde{\theta}_{i-1} + \alpha_i + \frac{\pi}{2}) \\
\mathsf{y}_i &= \mathsf{y}_{i-1} + \tilde{s}_i \cdot \sin(\tilde{\theta}_{i-1} + \alpha_i) + \tilde{\delta}_i \cdot \sin(\tilde{\theta}_{i-1} + \alpha_i + \frac{\pi}{2}) \\
\tilde{\theta}_i &= \tilde{\theta}_{i-1} + \tilde{\phi}_i
\end{aligned}$$

Notice, that we added constant parameters c_3, c_{10}, c_{17} into the variance formulas. This ensures any "constant" localization error due to the discretization of the LPM can be handled. Additionally, for testing purposes, it allows us a convenient way to create large, overestimating models or small, underestimating models by simply setting all uncertainty to be constant. Though this action model has 21 parameters, it can also be tuned by solving six linear least squares systems.

4.3.2 Adaptive Particle Filter Localization

The *particle filter* is a well-known method that efficiently approximates these distributions so that pose uncertainty can be quickly calculated by a robot moving through an environment. Particles are selected from the posterior distribution from the previous time step. The action model, then "moves" each particle to create the proposal distribution (Figure 4.3). Proposal particles are weighted according to their likelihood scores and normalized to sum to one (Figure 4.4). These weighted particles represent the new posterior distribution over poses—higher weighted particles are more likely to be considered on the next localization cycle.

Many action models are hand tuned to generate proposal distributions that overesti-

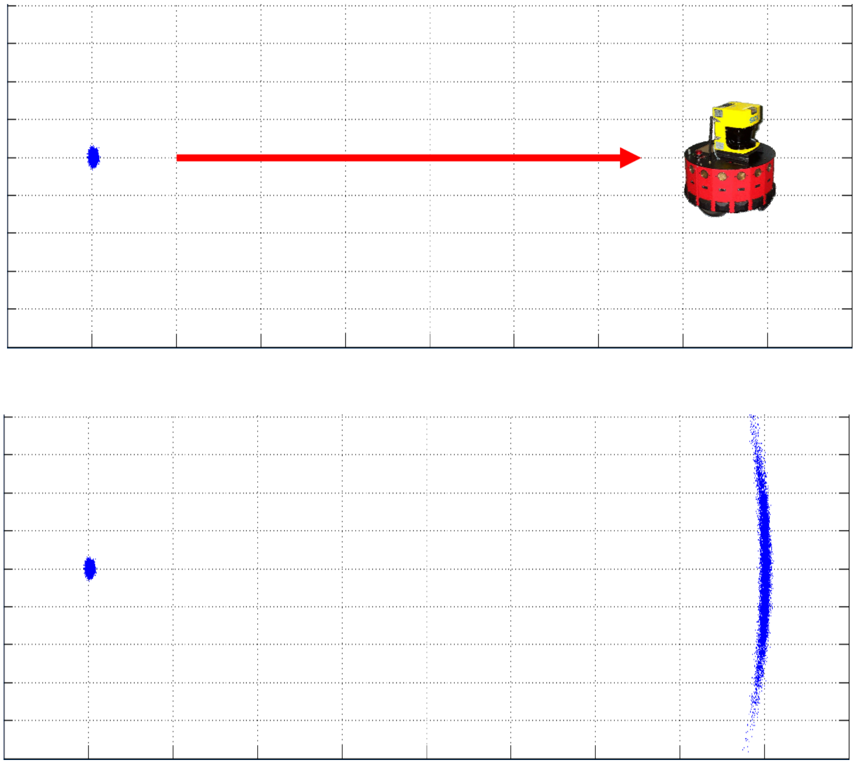


Figure 4.3: *Proposal distribution example: uncertainty in motion.* The robot's pose is estimated by a set of particle (here we show x,y locations without illustrating orientation). After an action, the robot's new pose is estimated by a proposal distribution, which generally is quite spread out.

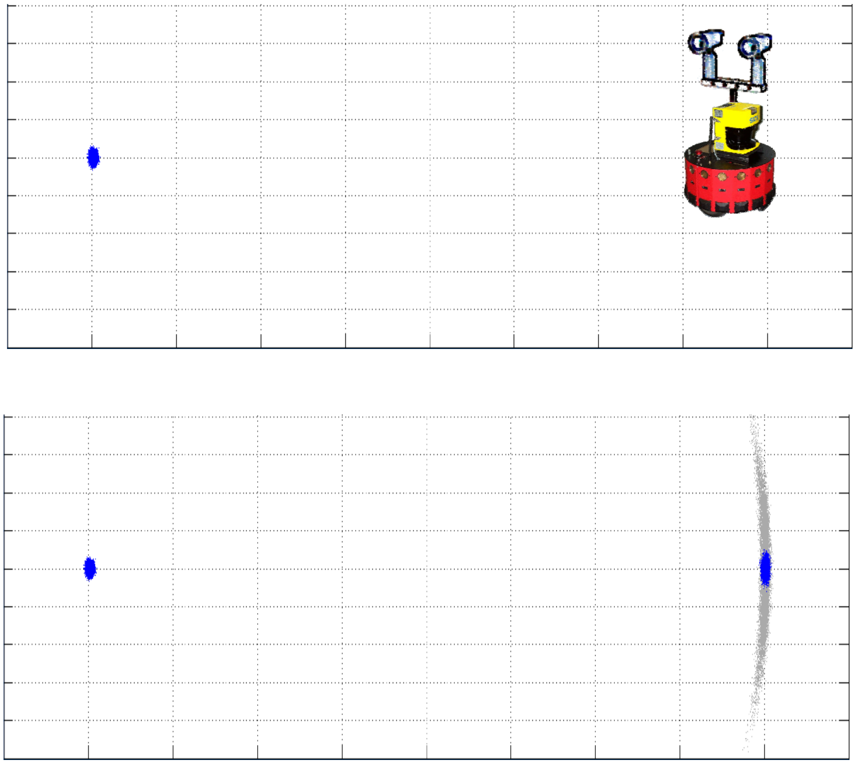


Figure 4.4: *Likelihood weighting example: uncertainty in observations.* The robot obtains an observation at the new pose, and scores each particle in the proposal with a likelihood based on matching its map of the world. The weighted particles represent the posterior distribution, modeling the robot's localization uncertainty.

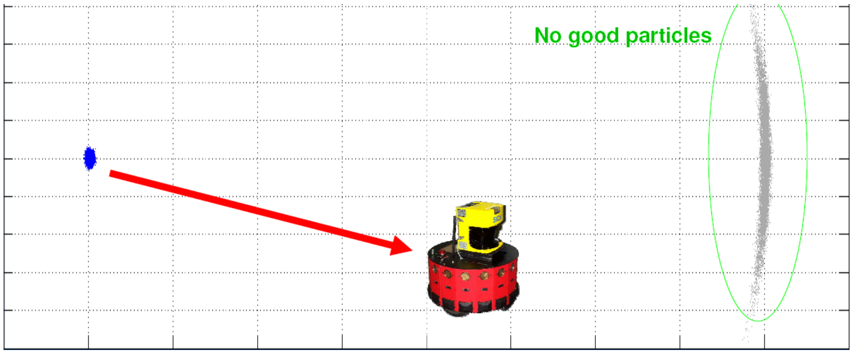


Figure 4.5: *Localization failure example: wheel slippage.* In cases where the robot's wheel slip, either on slippery floors, or when rotating in place, the odometry estimate of the incremental motion can be very far off from the actual trajectory the robot took. In this case, all particles may have low likelihood scores, as the current observation projected from the proposed particles does not match the world map.

mate the posterior. While this generally ensures high accuracy localization (the likelihood distribution definitely intersects the proposal), many particles end up with negligible likelihood scores. By tuning an action model, one can reduce these *wasted particles*. However, if improperly tuned, proposals can become too small. Underestimating proposals can cause inaccurate localization, as the proposal and likelihood distribution barely overlap. This is a far more serious problem than having wasted particles. Similarly problems like wheel slippage can often lead to proposal distributions being far from the ground-truth location of the robot. Again particles here have extremely low likelihoods, leading to incorrect posterior distributions (Figure 4.5).

We investigate a method for adapting the proposal in order to achieve better localization. The method adapts the proposal at each time step. The idea is to change each proposal distribution based on the likelihood scores generated by an initial set of particles. This allows a proposal to grow or shrink in order to place its remaining particles in a region that will lead to higher accuracy localization.

KLD-Sampling

Fox [2003] provides a method by which a robot that uses a particle filter to localize can be confident it has sufficiently sampled the proposal distribution. We utilize this method in all of the tested algorithms below for two reasons. First, it provides an upper bound on the number of samples needed for a given proposal distribution. This eliminates many redundant particles when the proposal distribution is small. Second, it also provides a lower bound on the number of particles. This ensures that large proposals are adequately sampled, which permits a fair comparison of localization accuracy between various localization algorithms.

Shrinking the Proposal

Suppose a proposal distribution overestimates the variance of the actual posterior distribution. Is there an intelligent way to shrink the proposal in order to use a small number of additional particles to obtain better localization? We adapt a method proposed by Grisetti et al. [2005] that shrinks the proposal distribution in such cases, therefore allowing more accurate localization.

Grisetti et al. [2005] observe that localization with laser range sensors (lidar) usually produces likelihood distributions that are much more highly peaked (smaller variance) than most proposal distributions—most modern-day planar lidar devices have ± 1 cm of error, which is much smaller than previous range-sensing devices. Given this assumption, it is possible to treat the proposal distribution as being constant under the likelihood distribution. Given a constant proposal, the posterior distribution is equal to the likelihood distribution. Thus, a new, better proposal would just be the estimated likelihood distribution. To obtain a closed form for the new proposal, the distribution around the likelihood peak is approximated with a Gaussian.

We borrow this idea, treating the high-precision likelihood given by the lidar measurements as a better approximation of the true posterior, than the combination of the likelihood and the (noisy) proposal. In our implementation the mean μ and covariance Λ of the Gaussian of the new proposal are estimated by the weighted mean and covariance of the particles $\{x'_j\}$ sampled from the original proposal and weighted by their normalized likelihoods. The equations are as follows:

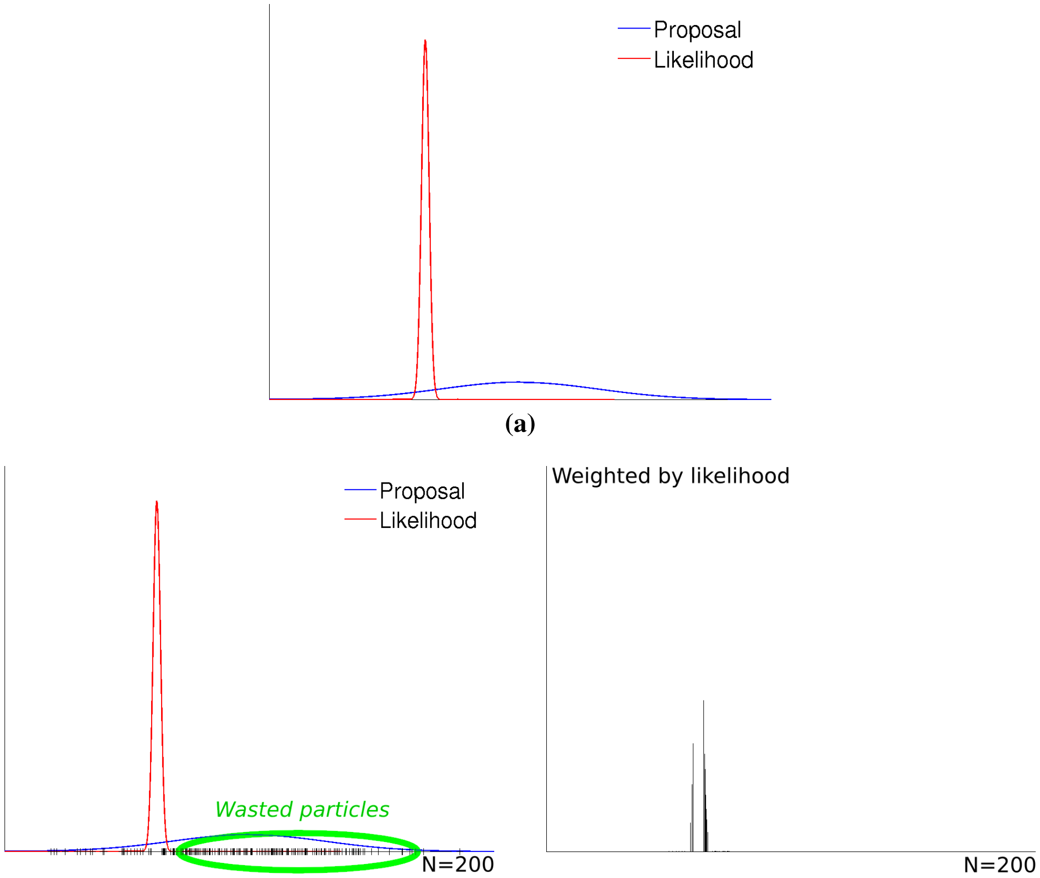
$$\begin{aligned}\mu &= \frac{1}{\eta} \sum_j x'_j \, p(z|x'_j, m) \\ \Lambda &= \frac{1}{\eta} \sum_j p(z|x'_j, m) \, (x'_j - \mu) \, (x'_j - \mu)^T,\end{aligned}$$

where $\eta = \sum_j p(z|x'_j, m)$. For this new proposal, KLD-sampling should require far fewer particles than in the original proposal due to the smaller variance. In the experiments below, we refer to this algorithm as `shrink`. Figures 4.6 and 4.7 illustrate the potential improvements of the `shrink` algorithm using a 1D example.

Growing the Proposal

Suppose that a proposal underestimates the actual posterior distribution. If the maximum likelihood particle in the proposal has a high score, then localization will be largely unaffected. The more serious case occurs if all sampled particles have low likelihood scores, i.e. the proposal is not large enough to intersect the peak of the likelihood distribution. A similar scenario occurs when the proposal's mode is displaced from the actual posterior (e.g., wheel slippage causes the action model to create a proposal far from the robot's actual pose). Is there an intelligent way to grow the proposal in order to "locate" an area of high likelihood?

If all particles in a proposal distribution have low likelihood scores, there is no signal that tells us exactly how to grow the proposal to search for better likelihood scores. By *likelihood scores*, we mean the unnormalized values of $p(z|x'_j, m)$. To handle such cases, we set a lower threshold on the maximum likelihood score. In the event that all likeli-



(a)

(b) **(c)**

Figure 4.6: *Poor localization due to an overestimating proposal.* Here we see a 1D example of a single particle filter iteration. This distribution represents the kind of problem a robot encounters quite frequently. **(a)** The proposal distribution is quite spread out due to large uncertainty (e.g., from odometry sensors), while the likelihood is quite highly peaked (e.g., from precise lidar measurements). **(b)** The algorithm approximates the proposal distribution by a set number of particles. **(c)** The particles are weighted by their likelihood scores, and this weighted distribution serves as the posterior distribution.

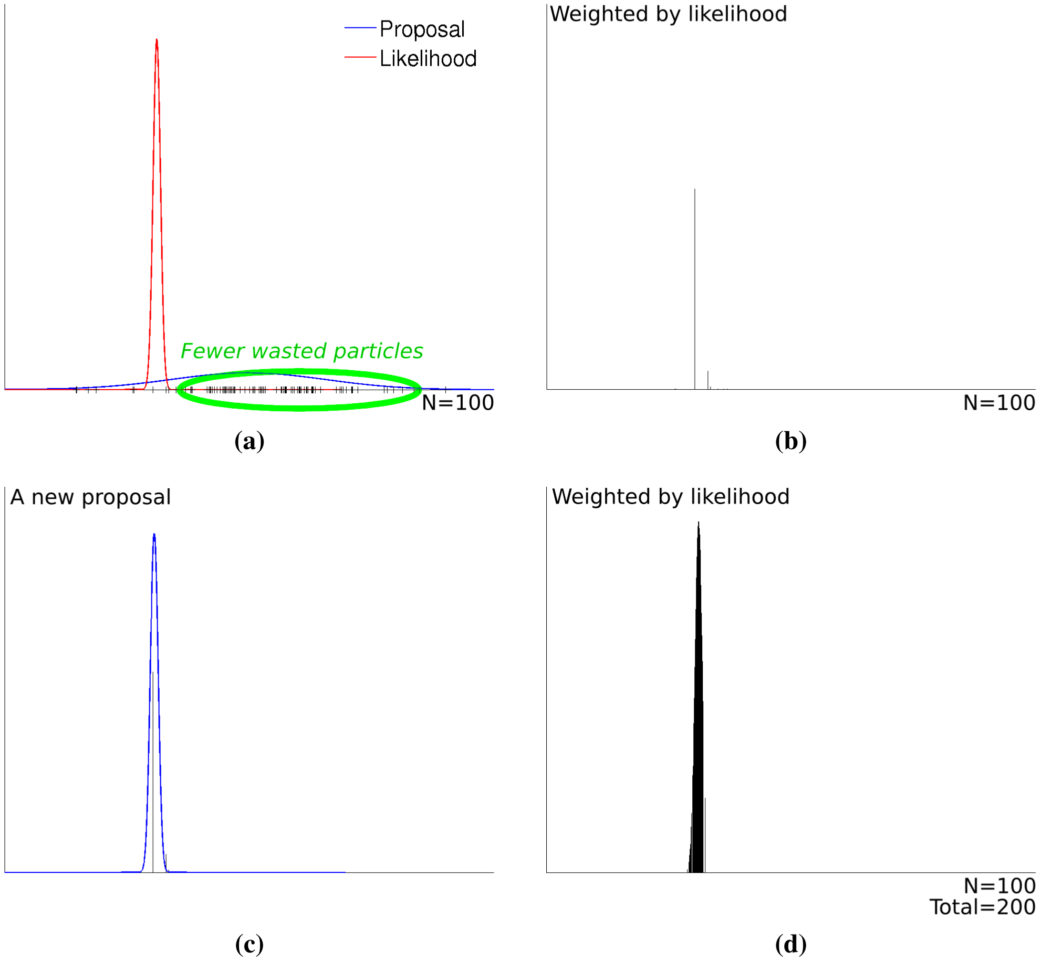


Figure 4.7: *Improved localization using the* shrink *algorithm.* Here we perform one particle filter iteration on the same example from Figure 4.6; however we use the shrink algorithm in our particle filter. **(a)** We only use 100 of the alloted 200 particles to sample the proposal. **(b)** We weight these particles based on the likelihood distribution. **(c)** We then approximate the weighted particles with a new Gaussian. **(d)** The new Gaussian acts as another proposal for the final 100 particles, which once weighted, approximate the posterior quite well. Compare this to Figure 4.6(c).

hood scores fall below this threshold, a new proposal is created that simply quadruples the covariance matrix of the current proposal.

In our current implementation, we perform this quadrupling up to five times if necessary, and the threshold is annealed to become more generous each time the proposal grows. In the experiments below, we refer to this algorithm as grow. The proposal does not always grow, only when required, due to a lack of accurate localization using the current proposal. Figures 4.8 and 4.9 illustrate the potential improvements of the grow algorithm using a 1D example.

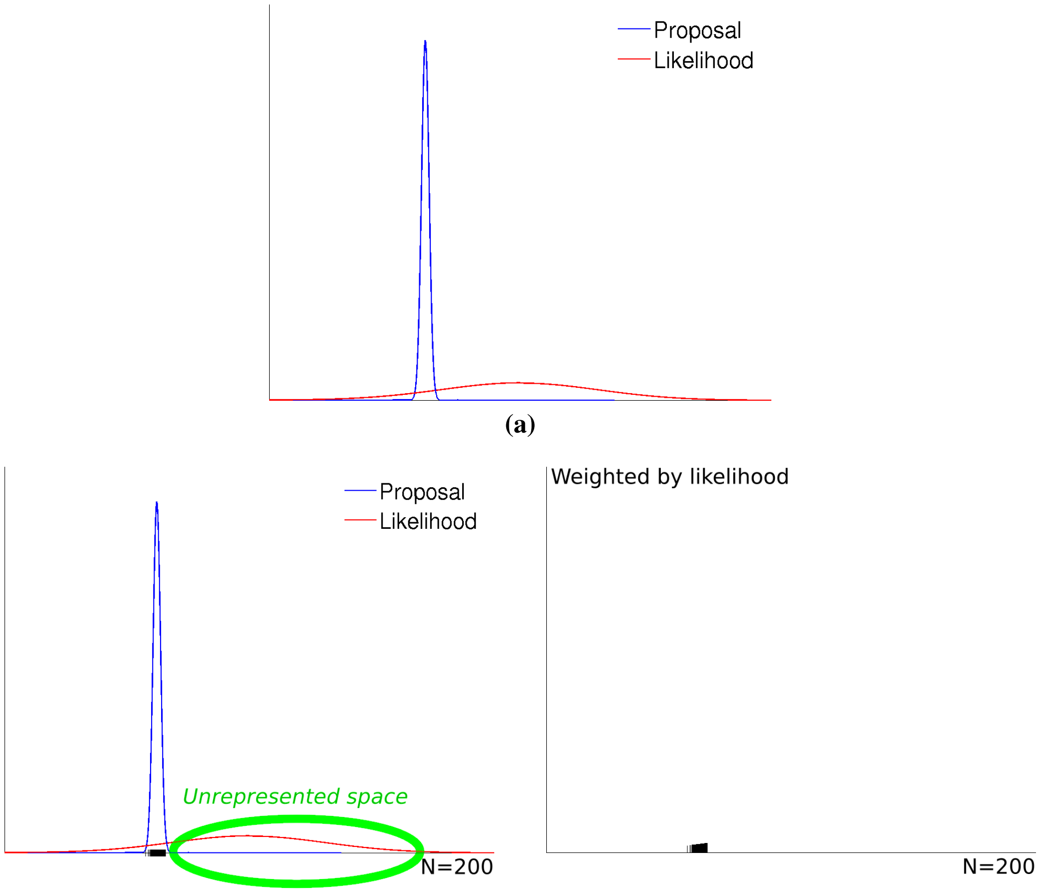
Shrinking and Growing

The shrink and grow algorithms above should be considered mutually exclusive improvements to the standard particle filter algorithm, which uses a single, static proposal distribution. In the experiments below, we also consider the shrink-and-grow algorithm. We expect the shrink-and-grow algorithm to cover all cases covered by shrink or grow.

This combined algorithm performs shrink on the original proposal, using particles wisely to get more localization precision. If no good particle is found, it then runs grow (growing the original proposal, not the "smaller" proposal created by shrink). The shrink algorithm is also performed on any new proposal generated by the grow algorithm, hoping the extra precision will find a good particle. This combination of grow then shrink is repeated until a good particle is found or a maximum number of iterations is achieved. This allows a proposal to grow until it intersects the likelihood distribution, then shrink down to fit the likelihood with a highly peaked Gaussian. Figures 4.10 and 4.11 illustrate the potential improvements of the shrink-and-grow algorithm using a 1D example.

Experimental Setup

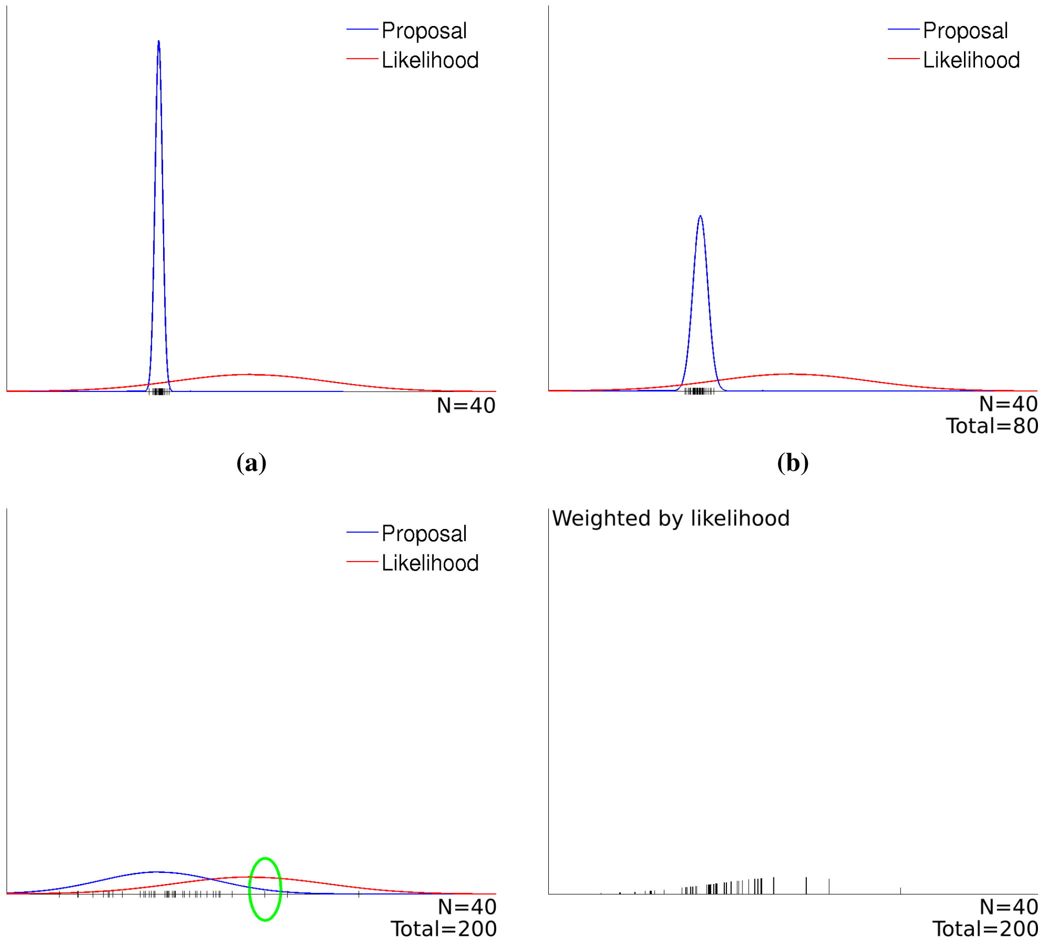
We evaluate the three algorithms above in terms of their influence on the accuracy of a particle filter in an actual robot localization setting. Here we describe our experimental



(a)

(b) **(c)**

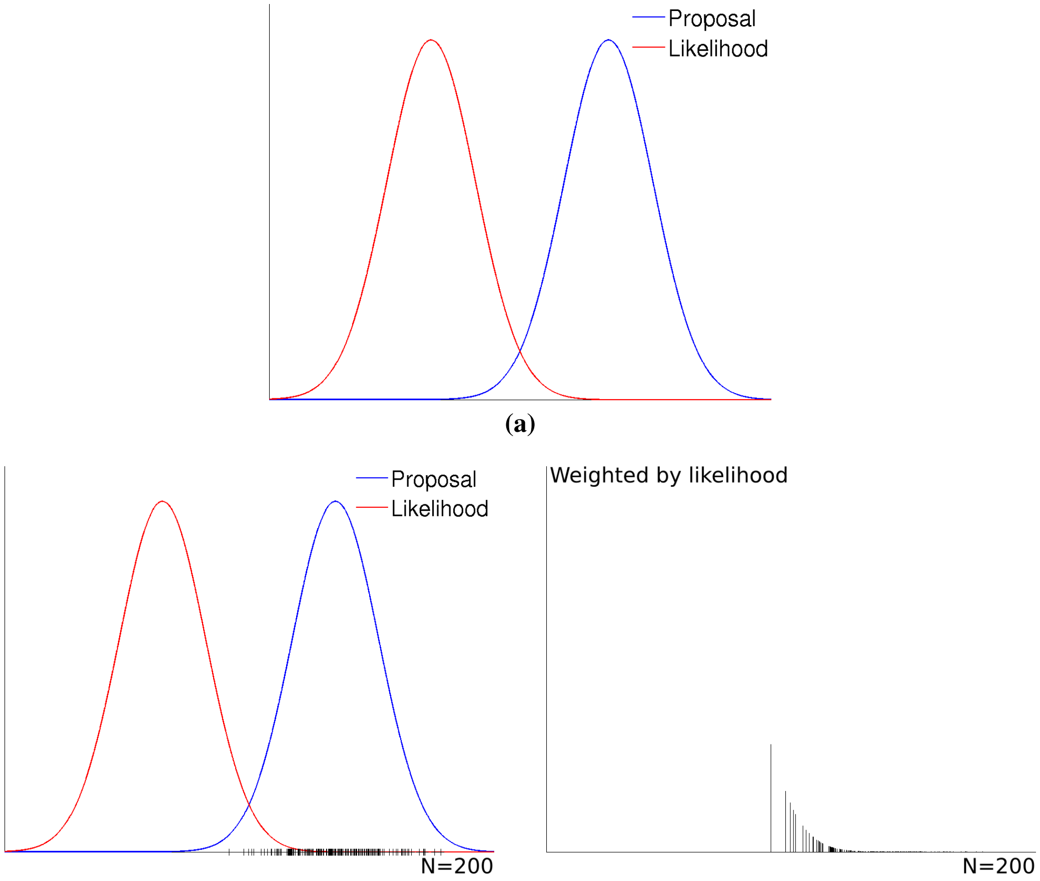
Figure 4.8: *Poor localization due to an underestimating proposal.* Here we see a 1D example of a single particle filter iteration. This distribution represents the kind of problem a robot encounters when it is in environments that are perceptually ambiguous in one or more dimensions (e.g., traveling down a long, featureless hallway, the posterior should spread out along the length of the hallway). **(a)** The proposal distribution is quite highly peaked here. (Note: The proposal and likelihood have swapped from Figure 4.6). **(b)** The algorithm approximates the proposal distribution by a set number of particles. **(c)** The particles are weighted by their likelihood scores, and this weighted distribution serves as the posterior distribution.



(a) (b)

(c) (d)

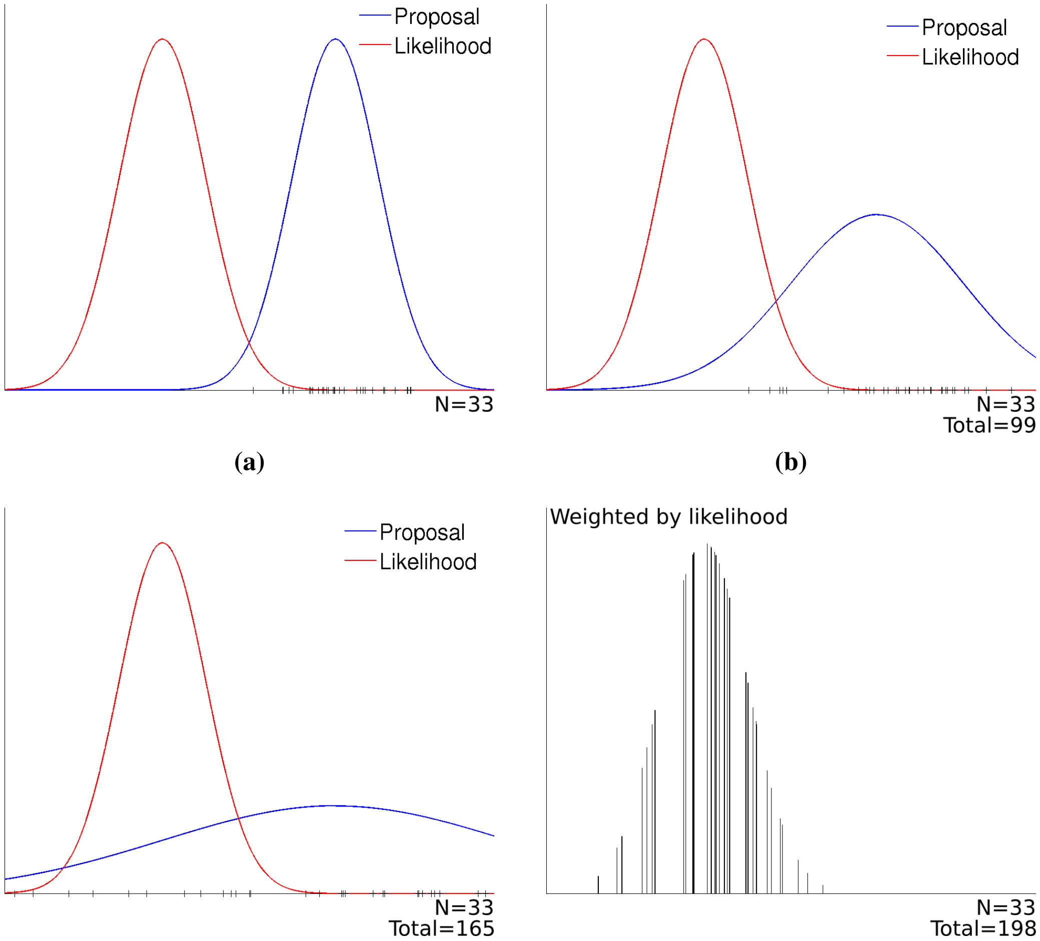
Figure 4.9: *Improved localization using the* grow *algorithm.* Here we perform one particle filter iteration on the same example from Figure 4.8; however we use the grow algorithm in our particle filter. **(a)** We only use 40 of the alloted 200 particles to sample the proposal, as the proposal has such small variance. **(b)** Because no particle had a likelihood weight over the desired threshold, the standard deviation is doubled, and another 40 particles are sampled. This continues until a good particle is found or until a maximum number of iterations is reached. **(c)** After growing the proposal three more times, a particle exists that has a high likelihood score, so the growing process terminates. **(d)** The final 40 weighted particles are a much better approximation of the posterior than the 200 particles in Figure 4.8(c).



(a)

(b) **(c)**

Figure 4.10: *Poor localization due to offset proposal.* Here we see a 1D example of a single particle filter iteration. This distribution represents a wheel slippage or very bad odometry measurement, where the action model proposes locations far from the ground-truth location of the robot. Normally we see these errors most in orientation. **(a)** The mean of the proposal distribution is far from the likelihood mean, which is a better approximation of the ground-truth location. **(b)** The algorithm approximates the proposal distribution by a set number of particles. **(c)** The particles are weighted by their likelihood scores, and this weighted distribution serves as the posterior distribution.



(a) **(b)** **(c)** **(d)**

Figure 4.11: *Improved localization using the* shrink-and-grow *algorithm.* Here we perform one particle filter iteration on the same example from Figure 4.10; however we use the shrink-and-grow algorithm in our particle filter. **(a)** We only use 33 of the alloted 200 particles to sample the original proposal, as we expect to need the rest to overcome the offset. **(b)** The algorithm created a new proposal using the shrink method, and sampled that with 33 more particles. Then because no particle had a likelihood weight over the desired threshold, it doubled the standard deviation, and sampled another 33 particles. **(c)** After shrinking and growing the proposal once more, a particle exists that has a high likelihood score, so the growing process terminates. **(d)** shrink is run one last time to get high likelihood values for the final 33 particles. The final 33 weighted particles approximate the likelihood better than the 200 particles in Figure 4.10(c). For a robot with noisy odometry but a high-precision lidar sensor, the likelihood is a better approximation of the posterior than the combination of likelihood and (noisy) proposal.

setup.

First, the robot explores an environment to create a sensor log. The robot records an alternating sequence of time labeled odometry and range observations. A SICK LMS-200 lidar sensor is used as the range-sensing device. The log file used here has 3961 odometry readings x_j (and 3961 laser readings z_j) from a nine minute run. 418 time steps involve no motion by the robot. For each pose x_j in the robot's trajectory, we calculate the relative motion of the robot $u_j = x_{j+1} - x_j$.

The log file is then used to create a global metrical map via offline methods (Chapter 9). The global metrical map provides *ground-truth* of the actual location of the robot at every laser reading, $\hat{x}_j$. The global map is represented as an occupancy grid that has 5 cm cells.

We ran localization experiments using three different action models. The parameter values of these three models are shown in Table 4.1. The large action model has a constant sized uncertainty that is centered on zero odometry error. The parameters, chosen beforehand, were selected in an attempt to overestimate the uncertainty at almost every time step. The small action model also has constant uncertainty centered on the odometry estimate. These parameters were chosen to underestimate the posterior for a majority of the sensor trace—when the robot is making very small movements, this model is sufficient for good localization. The final action model is a least squares fitted model. These parameters were calculated by finding the error between odometry and ground-truth (from the global map) and solving a series of least squares systems to find the best parameters for the action model as described by Eliazar and Parr [2004]. Thus, it is an action model created specifically for this data set.

We tested five localization algorithms on these three action models: raw odometry, standard particle filter, shrink, grow, and shrink-and-grow. For each localization algorithm (excluding raw odometry, which only uses 1 particle), KLD-sampling is used to determine the number of particles each proposal needs. Our KLD-sampling parameters [Fox, 2003]

$\tilde{\mathbf{s}}$	c_0	c_1	c_2	c_3	c_4	c_5	c_6
large	0	1	0	1e-2	0	0	0
small	0	1	0	1e-6	0	0	0
least-squares fitted	-0.012	0.99	-0.012	2.6e-05	0	0.0052	0
$\tilde{\delta}$	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}
large	0	1	0	1e-2	0	0	0
small 0	0	0	1e-6	0	0	0	0
least-squares fitted	0.0014	-0.0016	0.0019	1.4e-05	0	0.0012	0
$\tilde{\phi}$	c_{14}	c_{15}	c_{16}	c_{17}	c_{18}	c_{19}	c_{20}
large	1	0	1	3e-2	0	0	0
small	1	0	1	3e-6	0	0	0
least-squares fitted	0.98	-0.0048	0.98	4.1e-05	0	0.0093	0

Table 4.1: *Action model parameters.* large is an overestimating model. small is an underestimating model. least squares fitted is a model fit using the ground-truth posterior distributions over the entire data trace.

are set as follows: confidence=0.95, KLD error=0.5, bin size=(0.05, 0.05, 0.006). These were tuned so that our robot does not use more than 500 particles, except in very rare situations.

Comparing localization algorithms by performing Markov localization on the entire trajectory is tricky, as one bad localization due to unlucky sampling can affect all future localization. By restarting localization at ground-truth at each time step, we can compare algorithms directly, eliminating the problem of cumulative error—for each laser reading, z_j, we start the robot at $\hat{x}_{j-1}$ and create a proposal distribution according to the action model and the action u_{j-1} given by the raw odometry.

After the localization step terminates, the best (unnormalized) likelihood score of each algorithm is recorded and used as our accuracy metric. A likelihood score is measured by comparing how well the laser scan at a hypothesized odometry matches the laser scan at the ground-truth odometry, given that lasers are discretized into a 15x15 meter occupancy grid with 5 cm cells [Konolige, 1999]. Each score is a real value from 0 to 1.

The 5 localization algorithms, each run on the three action models, gives us 15 datasets (each with 3960 localization steps) to analyze (L_1-L_5, S_1-S_5, F_1-F_5 in Table 4.2). If the 5 localization algorithms were run separately, these datasets would be independent, but to save time, we ran all 5 localization algorithms at the same time: we ran shrink-and-grow and recorded the data for the other 4 algorithms. This means some of the algorithms had proposals that were supersets of others (e.g., L_5 had proposals that were supersets of the proposals in L_1-L_4). To be rigorous in our comparison, we created another 15 datasets from a different run (i.e., different sampled particles due to different random number generation). We compare these 15 datasets (L_6-L_{10}, S_6-S_{10}, F_6-F_{10}) with their counterparts.

Results

The complete results of our analysis are shown in Table 4.2. The table shows the average difference (over 3960 differences) between the best (unnormalized) likelihood scores of the algorithms. The table also shows whether the set of 3960 maximum likelihood scores from one algorithm is significantly different than the corresponding set of another algorithm: here significance means $p < 0.01$ using the standard paired t-test. We expected no significant difference between results collected from running the same algorithm on the same model (the diagonals).

Note that the shrink algorithm outperforms the standard particle filter in all cases: L_3 over L_7, L_8 over L_2, S_3 over S_7, S_8 over S_2, F_3 over F_7, F_8 over F_2. shrink outperforms grow on the overestimating action model (L_3 over L_9, L_8 over L_4). This result is expected due to a more precise search space of the adapted proposal.

The grow algorithm also improves localization accuracy over the standard particle filter for every action model. As expected, grow outperforms shrink on the underestimating model (S_4 over S_8, S_9 over S_3). This is due to the ability of the grow algorithm to search a larger area in cases where the odometry was vastly incorrect.

In the case of the least squares action model, shrink and grow both outperform the standard particle filter, with grow slightly outperforming shrink in localization accuracy. This implies that even the least squares model incorrectly hypothesizes the peak of the likelihood distribution.

Finally, as expected shrink-and-grow improves accuracy over all other algorithms on all possible action models. In addition to these quantitative results, we have observed a qualitative difference in the quality of the LPMs created when using shrink-and-grow over a standard particle filter. Because we are using KLD-sampling to adequately sample each proposal distribution, we often see little increase in the total number of particles needed—the adapted proposals created by the shrink portion of the algorithm have small variance and require few particles to represent. In situations where more particles are needed as the adaptive proposal grows, we are willing to spend the extra cycles to ensure quality localization; however, we use a maximum of 500 particles (100 particles for up to 5 possible shrink-and-grow iterations) in order to ensure localization at 3 Hz.

algorithm \ *model*	large run1	large run2	small run1	small run2	least squares run1	least squares run2
raw odometry	L_1	L_6	S_1	S_6	F_1	F_6
standard PF	L_2	L_7	S_2	S_7	F_2	F_7
shrink	L_3	L_8	S_3	S_8	F_3	F_8
grow	L_4	L_9	S_4	S_9	F_4	F_9
shrink-and-grow	L_5	L_{10}	S_5	S_{10}	F_5	F_{10}

	L_6	L_7	L_8	L_9	L_{10}
L_1	0.000	$-0.098,*$	$-0.176,*$	$-0.100,*$	$-0.183,*$
L_2	$0.098,*$	0.000	$-0.078,*$	-0.002	$-0.084,*$
L_3	$0.176,*$	$0.078,*$	0.000	$0.076,*$	$-0.007,*$
L_4	$0.100,*$	$0.003,*$	$-0.076,*$	0.001	$-0.082,*$
L_5	$0.182,*$	$0.085,*$	$0.006,*$	$0.083,*$	0.000

	S_6	S_7	S_8	S_9	S_{10}
S_1	0.000	$-0.060,*$	$-0.067,*$	$-0.090,*$	$-0.099,*$
S_2	$0.060,*$	0.000	$-0.007,*$	$-0.030,*$	$-0.039,*$
S_3	$0.066,*$	$0.006,*$	0.000	$-0.023,*$	$-0.033,*$
S_4	$0.089,*$	$0.029,*$	$0.022,*$	-0.001	$-0.010,*$
S_5	$0.099,*$	$0.039,*$	$0.032,*$	$0.009,*$	0.000

	F_6	F_7	F_8	F_9	F_{10}
F_1	0.000	$-0.117,*$	$-0.134,*$	$-0.141,*$	$-0.158,*$
F_2	$0.117,*$	0.000	$-0.017,*$	$-0.024,*$	$-0.041,*$
F_3	$0.133,*$	$0.016,*$	-0.001	$-0.008,*$	$-0.025,*$
F_4	$0.141,*$	$0.023,*$	$0.007,*$	-0.001	$-0.017,*$
F_5	$0.158,*$	$0.040,*$	$0.024,*$	$0.016,*$	0.000

Table 4.2: *Localization accuracy of 5 different algorithms.* Comparison of maximum likelihood scores (which range from 0 to 1) between different localization algorithms. The scalar values show the average difference in the maximum likelihood score of the row data minus the column data. When the maximum likelihood scores of the two sets of scores are significantly different ($p < 0.01$), there is an asterisk.

As a result of utilizing this adaptive particle filter, we improve the quality of our LPMs even in indoor environments where mobile robot wheels slip quite often. It is due to having high quality LPMs that we are able to go from model of obstacles and free space to symbolic structures of paths and places. This is discussed in Chapter 5.

Chapter 5

Symbol Grounding:
The HSSH Local Topology Level

> *It takes a wonderful brain and exquisite senses to produce a few stupid ideas.*
>
> George Santayana, *The Life of Reason, Volume 5: Reason in Science* (1905)

This chapter discusses how we abstract and utilize local topology models from the LPMs. We begin by giving a formal description of local topology at places, which will be useful for determining loop-closing hypotheses in the Global Topology Level (Chapter 8). Next we discuss how local topology is abstracted from LPMs using gateways [Beeson et al., 2003; Kuipers et al., 2004]. We illustrate how a robot with an LPM and a gateway detection algorithm can robustly detect when it is at a place versus moving along a qualitatively uninteresting path [Beeson et al., 2005]. We give an implementation for gateways, and show the robustness of the algorithm under noisy conditions.

5.1 Formal Description of Local Topology

As the robot and its scrolling LPM move continuously through the environment, the robot identifies a discrete set of isolated *places* and the *path segments* that connect them. In the small-scale space of the LPM, a place neighborhood is an extended region. In the large-scale space representation, a place is a node in the topological graph, and is connected by paths to other places. These are the local elements from which a global topological map is constructed. We abstract the structure of a place neighborhood to the *local topology* description of the place. Just as a path describes the linear order of places on it, a place describes the circular order of directed paths radiating from it. We call this the local topology S_p of a place p, and describe the circular order with a structure called a *star* [Kuipers et al., 2004]. This section discusses how this symbolic representation of a place (in large-scale space) is grounded in the metrical description m_p of the place neighborhood (in small-scale space).

A *local-path* $\tilde{\pi}$ at a place p is the fragment of a topological path that is visible within the stored local perceptual map, m_p, of the neighborhood of p. A *directed local-path* is of the form $\tilde{\pi}^d$, where $d \in \{+,-\}$ represents the direction along $\tilde{\pi}$ moving away from p. Upon arriving at a new place, a local-path and its directions may not yet have been matched with a global topological path and its directions.

A *star* S is a set of directed local-paths such that $\tilde{\pi}^+ \in S \iff \tilde{\pi}^- \in S$. There are two functions that describe stars.

$next : S \rightarrow S$ induces a clockwise circular order over $\tilde{\pi}^d \in S$. $next(\tilde{\pi}^d)$ is the next element from $\tilde{\pi}^d$ in the clockwise order.

$\alpha : S \rightarrow \{0,1\}$ associates an attribute value $\alpha(\tilde{\pi}^d)$ to $\tilde{\pi}^d \in S$, where $\alpha(\tilde{\pi}^d) = 1$ means that travel is possible along $\tilde{\pi}^d$ away from p, and $\alpha(\tilde{\pi}^d) = 0$ means that travel away from p along $\tilde{\pi}^d$ is not possible.

The star is naturally encoded as a sequence of pairs, where the sequence encodes the *next* relation (*next* of the last element being the first element), and the second element of each pair is the value of α applied to the first element. For example, consider the following local topology (star) descriptions of familiar intersection types including local-paths $\tilde{\pi}_a$, $\tilde{\pi}_b$, and sometimes $\tilde{\pi}_c$.[1] (For ease of visualization, the first directed local-path in the circular order is the one directed upward.)

$$\begin{array}{ll}
+ & [\langle\tilde{\pi}_a^+,1\rangle,\langle\tilde{\pi}_b^+,1\rangle,\langle\tilde{\pi}_a^-,1\rangle,\langle\tilde{\pi}_b^-,1\rangle] \\
\mathsf{T} & [\langle\tilde{\pi}_a^-,0\rangle,\langle\tilde{\pi}_b^+,1\rangle,\langle\tilde{\pi}_a^+,1\rangle,\langle\tilde{\pi}_b^-,1\rangle] \\
\mathsf{L} & [\langle\tilde{\pi}_a^+,1\rangle,\langle\tilde{\pi}_b^+,1\rangle,\langle\tilde{\pi}_a^-,0\rangle,\langle\tilde{\pi}_b^-,0\rangle] \\
\mathsf{Y} & [\langle\tilde{\pi}_a^-,0\rangle,\langle\tilde{\pi}_b^+,1\rangle,\langle\tilde{\pi}_c^-,0\rangle,\langle\tilde{\pi}_a^+,1\rangle,\langle\tilde{\pi}_b^-,0\rangle,\langle\tilde{\pi}_c^+,1\rangle] \\
\mathsf{K} & [\langle\tilde{\pi}_a^+,1\rangle,\langle\tilde{\pi}_b^+,1\rangle,\langle\tilde{\pi}_c^+,1\rangle,\langle\tilde{\pi}_a^-,1\rangle,\langle\tilde{\pi}_b^-,0\rangle,\langle\tilde{\pi}_c^-,0\rangle] \\
\psi & [\langle\tilde{\pi}_a^+,1\rangle,\langle\tilde{\pi}_b^+,1\rangle,\langle\tilde{\pi}_c^-,0\rangle,\langle\tilde{\pi}_a^-,1\rangle,\langle\tilde{\pi}_b^-,0\rangle,\langle\tilde{\pi}_c^+,1\rangle]
\end{array}$$

An *isomorphism* $\phi : S \to S'$ between two stars S and S' is a bijective function such that

$$\begin{aligned}
next(\phi(\tilde{\pi}^d)) &= \phi(next(\tilde{\pi}^d)) \\
\alpha(\phi(\tilde{\pi}^d)) &= \alpha(\tilde{\pi}^d) \\
path(\phi(\tilde{\pi}^d)) &= path(\phi(\tilde{\pi}^{opp(d)})),
\end{aligned}$$

where $path(\tilde{\pi}^d) = \tilde{\pi}$. An isomorphism means that the two stars have the same local topology under a suitable rotation of the circular order. Note that two stars may have multiple distinct isomorphisms. For example, there are four distinct isomorphisms between two $+$ intersections.

The local topology description provides a purely qualitative account of "left" and "right", avoiding the need to define them in terms of thresholds on some angular variable. A particular directed local-path at a place p, $\tilde{\pi}^+$, and its opposite, $\tilde{\pi}^-$, partition the other directed local-paths in the star into two groups. Those that are between $\tilde{\pi}^+$ and $\tilde{\pi}^-$ in the clockwise direction can be described as being "to the right" of $\tilde{\pi}^+$. Those between $\tilde{\pi}^+$ and $\tilde{\pi}^-$ in the counter-clockwise direction can be described as "to the left" of $\tilde{\pi}^+$. This

---

[1] Note that we do not have a fixed set of equivalence classes for local topology abstraction. Though there is an upper bound on the number of paths that can fit into an LPM, this is determined by the path width and the LPM size. Thus, many types of intersections can exist that cannot be "named" using a letter.

also defines the appropriate destination for a route instruction such as “turn right” when the agent is at a place p, facing along a directed path $\tilde{\pi}_a^+$. The pragmatics of natural language requires that “turn right” must uniquely specify a directed local-path $\tilde{\pi}_b^d$ that is “to the right” of $\tilde{\pi}_a^+$, such that $\alpha(\tilde{\pi}_b^d) = 1$ (i.e., $\tilde{\pi}_b^d$ is navigable from p).

5.2 Grounding Local Topology in the Local Perceptual Map

We have illustrated how to describe a place symbolically as a circular order of directed local-paths. Here we discuss how to use *gateways* to ground local-paths in the LPM. Gateways allow the robot to ground large-scale actions in the small-scale metrical models, abstract a symbolic local topology description from the small-scale model, and detect and compare places in the environment. The term “gateway” is adapted from Chown et al. [1995], who define gateways as the locations of major changes in visibility (Section 2.3.1). Section 5.3 discusses detecting places via gateways and local-paths, Section 5.4 discusses how motion between gateways abstracts to high-level motion. Chapter 6 addresses the issues involved in implementing algorithms for reliably detecting gateways.

5.2.1 Gateways

We define a gateway as a boundary in the local perceptual map that separates the local place neighborhood from the larger environment. That is, a gateway is the boundary where control shifts between localization within the local place neighborhood and travel from one place neighborhood to another. A gateway has two directions, *inward* (looking into the place) and *outward* (looking away from the place), according to the direction of that shift. The location, extent, and orientation of gateways at a place are saved as annotations of the local place neighborhood map m_p.

In much of human experience with large-scale environments (both natural and man-made) local place neighborhoods are separated from each other (either by boundaries or by distance), and they are connected by travel actions along paths. Navigation in large-scale space is thus typically an alternation between motion along travel paths and motion within place neighborhoods. The existence of gateways, as interfaces between the two types of travel, is therefore a requirement for the abstraction from small-scale to large-scale space. Certainly extreme situations occur, such as place neighborhoods that overlap or are immediately adjacent, or environments with (apparently) no distinctive states at all. These will be mentioned in appropriate sections below.

5.2.2 Local Topology Creation

Given an implementation for detecting gateways in a stored map of a place, m_p, we can ground the local topology concepts of local-paths in our small-scale model of the surrounding environment (Figure 5.1).

- For each outward-facing oriented gateway $\langle g, out \rangle$, define a *directed local-path* $\tilde{\pi}_g^+$ that leads away from the current place.
- Initialize a circularly ordered *star* S_p with a list (clockwise from an arbitrary starting point) of associations between directed local-paths and oriented gateways, $(\langle \tilde{\pi}_g^+, 1 \rangle \leftrightarrow \langle g, out \rangle)$. Since these are traversable paths, each $\alpha(\tilde{\pi}_g^+) = 1$. [2]
- Test each pair of gateways, g and g', via a *path continuity* test, to determine whether their directed local-paths belong to a single continuous path. If so, give both directed local-paths the same path name (e.g., $\tilde{\pi}_a$ below and in Figure 5.1), and include the inward oriented gateways in the association. For example, change

$$\begin{array}{rcl} (\langle \tilde{\pi}_g^+, 1 \rangle & \leftrightarrow & \langle g, out \rangle) \\ (\langle \tilde{\pi}_{g'}^+, 1 \rangle & \leftrightarrow & \langle g', out \rangle) \end{array} \quad \text{to} \quad \begin{array}{rcl} (\langle \tilde{\pi}_a^+, 1 \rangle & \leftrightarrow & \langle g', in \rangle, \langle g, out \rangle) \\ (\langle \tilde{\pi}_a^-, 1 \rangle & \leftrightarrow & \langle g, in \rangle, \langle g', out \rangle) \end{array}$$

---

[2] In future implementations, $\alpha : S \to \{$MIDLINE, LEFTWALL, RIGHTWALL, DEADEND, NONE$\}$ should associate directed local-paths with attributes representing the control laws for traversing the path in that direction. (LEFTWALL and RIGHTWALL imply coastal navigation scenarios. For terminating local-paths, DEADEND means that further travel is blocked, while NONE means that no control law is applicable.)

- For each $\tilde{\pi}_g^+ \in S_p$ such that $\tilde{\pi}_g^- \notin S_p$, insert the association $(\langle \tilde{\pi}_g^-, 0\rangle \leftrightarrow \langle g, in\rangle)$ into the circular order of S_p, in a position determined by its failure of the path continuity test.

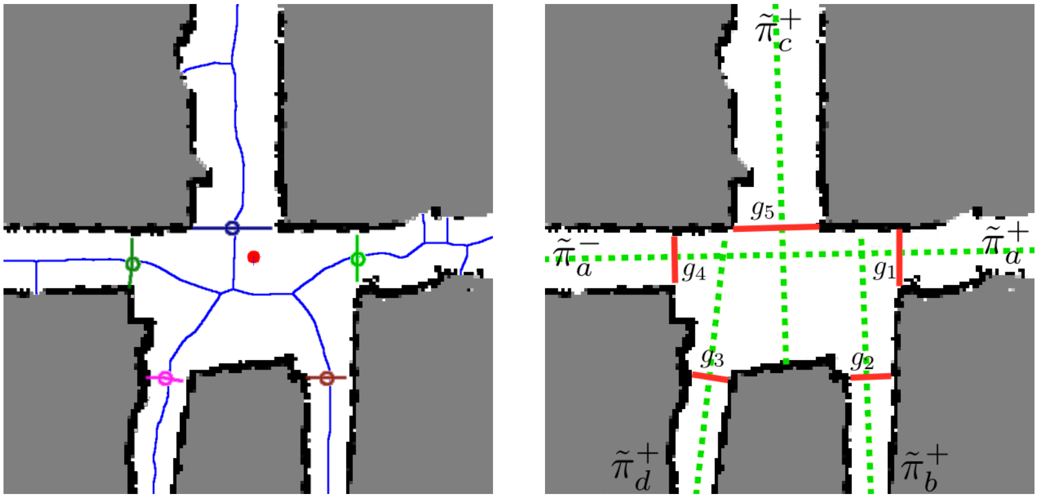
In our current implementation, the gateways g and g' belong to a single continuous path if: (1) a ray normal to the orientation of gateway g and centered at the midpoint of gateway g intersects the line segment that defines gateway g'; and (2) vice versa for a ray from g' towards gateway g. (Note that the failure of this test should determine a pair of gateways that the non-traversable path continuation falls between.)

At this point, the star S_p is a complete representation of the local topology of the neighborhood described by the LPM. Since this representation is expressed completely in terms of small-scale space (the gateways g and the directed local-paths $\tilde{\pi}_g^d$), we refer to this as the *small-scale star*. Binding the directed local-paths to directed paths in the topological map of large-scale space implies the appropriate *large-scale star*. This binding is part of the HSSH topological mapping process, and is discussed in Section 8.1.

5.3 Detecting Places

To explain local topology abstraction, we provided examples where the robot was already at a place. Perhaps surprisingly, the method for constructing the local topology of a place neighborhood does not actually depend on being at a place neighborhood. Gateways can also be defined along paths (Chapter 6), as they separate the space immediately surrounding the robot from the "frontier" of the LPM. Therefore, if we recalculate gateways and local topology at each time step, we can very easily *detect* places.

Places are important in many different ways. For spatial reasoning, people use places to define both bounded and unbounded regions of space: "in Texas" or "near the statue". Places are further used for metaphorical high-level reasoning such as mathematical ranking and social ranking: "first place" or "overstepping one's place". Places are even



(a) **(b)**

Small-scale star description

$$\begin{array}{rcl}
(((\langle\tilde{\pi}_a^+,1\rangle & \leftrightarrow & \langle g_4,in\rangle,\langle g_1,out\rangle) \\
(\langle\tilde{\pi}_b^+,1\rangle & \leftrightarrow & \langle g_2,out\rangle) \\
(\langle\tilde{\pi}_c^-,0\rangle & \leftrightarrow & \langle g_5,in\rangle) \\
(\langle\tilde{\pi}_d^+,1\rangle & \leftrightarrow & \langle g_3,out\rangle) \\
(\langle\tilde{\pi}_a^-,1\rangle & \leftrightarrow & \langle g_1,in\rangle,\langle g_4,out\rangle) \\
(\langle\tilde{\pi}_d^-,0\rangle & \leftrightarrow & \langle g_3,in\rangle) \\
(\langle\tilde{\pi}_c^+,1\rangle & \leftrightarrow & \langle g_5,out\rangle) \\
(\langle\tilde{\pi}_b^-,0\rangle & \leftrightarrow & \langle g_2,in\rangle))
\end{array}$$

(c)

Figure 5.1: *Identifying gateways and local topology in the LPM.* The local perceptual map (LPM) is implemented as a bounded occupancy grid. The robot is shown as a circle in the center of the LPM. **(a)** The gateways separating the place neighborhood from the paths are defined. In our implementation this is done using a pruned Voronoi skeleton. **(b)** Gateway locations and directions are used to identify the directed local-paths and to determine which pairs satisfy the path continuity requirements. **(c)** The small-scale star enumerates directed local-paths in clockwise order, describing their traversability and association with gateways. Note: the robot entered the place via g_5; thus, it arrived on directed local-path $\tilde{\pi}_c^-$. This environment has five gateways and four paths.

used to refer to normal roles or functions: “the place of the media” or “something is out of place”.

The broad use of places is a specific example of a crucial point in the study of intelligent agents—spatial knowledge is a foundation for high-level common-sense knowledge [Lakoff and Johnson, 1980]. We believe that by studying the problem of grounding continuous sensory experience to low-level, symbolic spatial constructs, we are developing a foundation for the high-level common-sense knowledge necessary for an intelligent agent to act in environments with people and other animals.

Here we discuss the power of using gateways and local topology to detect and describe places over other popular approaches. We then detail and evaluate our current implementation for finding gateways in the LPM.

5.3.1 Place Criteria

A topological navigation system depends on the initial low-level abstraction of places from continuous sensory experience. Here we attempt to define these places using criteria necessary for topological map-building behaviors. We define places at locations in the environment that satisfy the following two criteria.

1. *Places occur at qualitative changes along paths.*

The majority of the time, this criterion defines places at intersections, or *decision points*. The importance of path intersections in topological representations cannot be overstated. The recent work of Guilford et al. [2004], demonstrates that pigeons, long considered to use Earth’s magnetic field to fly long distances, follow man-made highway networks in well-known environments, using familiar highway intersections as markers for specific turn actions. It has long been known that humans use intersections as a basis for building spatial representations in unknown environments [Siegel and White, 1975]—preferring first to build structural models of novel environments prior to detailed visual models [Stankiewicz and Kalia, 2007].

Occasionally people may define places that are not at intersections. Dead-ends are one example of important places that are not necessarily at an intersection of two or more paths. Less useful places (from a topological mapping perspective) can be defined by using salient landmarks along paths. Such landmarks are useful for tracking progress along long paths and thus define places based on regions of visibility. Common examples are unusual buildings along highways such as water towers. From the perspective of topological map-building, places that are not at intersections or dead-ends do not yield any additional structural information. In fact, as people do start to remember landmarks, they are biased to first learn landmarks at critical path changes along routes [Heft, 1979; Aginsky et al., 1997].

2. *Places must be reliably detectable.*

Although this criterion seems simple, it is extremely important. Qualitatively interesting locations should be considered as possible places when they exist, but their persistence needs to be taken into account.

Any place detection algorithm needs to provide a stable set of places. Intersections are not only common but extremely stable (they rarely disappear). Thus, this criterion is mostly needed for places that are not at intersections. For example, a bright pink car parked on the side of a highway, while salient, does not define a reliable place. The chances that the car will remain at the location for an extended period of time are very low.

Places that are not at intersections depend on higher-level knowledge about stability, saliency, and informativeness that may not be available for most robotic implementations. For this reason, the algorithms and implementation for mobile robots detailed in the following sections focus exclusively on decision points (places at intersections, doorways, and dead-ends), although future robotic implementations with knowledge of objects [Modayil, 2007], semantically labeled LPMs [Murarka et al., 2006], or human-defined places may be able to bootstrap place detection at other interesting locations.

5.3.2 Places from Local Topology

We define the robot to be *on-path* when the local topology of the LPM contains exactly two gateways and exactly one path (e.g., Figure 5.2(a)). When the agent is on-path, it is selecting and executing control laws (and hence primitive motions) to perform a travel action (Section 5.4). The LPM scrolls as the agent moves, keeping the agent near its center cell, and serving as an observer for the local small-scale space.

When the agent is not on-path, it is in a place neighborhood. In this situation, the agent establishes a fixed correspondence between the LPM and the structure of the place neighborhood. Here, the LPM serves as a local metrical map m_p of the place neighborhood (and does not scroll with the agent's motion within the place neighborhood). Thus, the number, location, and structure of places in an environment depends in part on the predetermined size of the LPM[3], though places are not sensitive to small changes in LPM size.

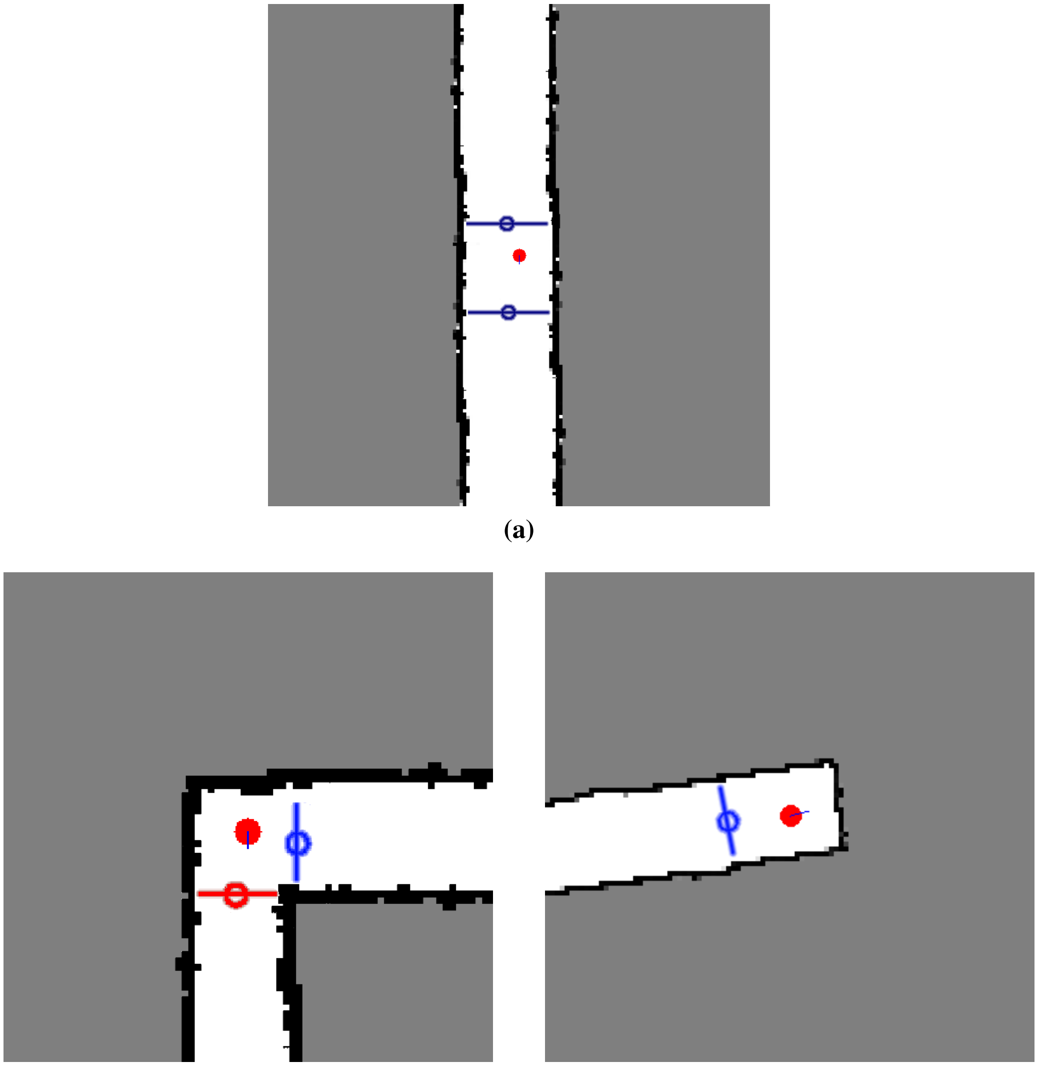
When the robot is not on-path, it either has more than one local-path (Figure 5.2(b)), which occurs at intersections or open doorways, or one local-path with only one gateway, which occurs at dead-ends (Figure 5.2(c)). These are all places. There is a degenerate case where no gateways exist. Due to our implementation of gateways using an LPM, this situation means there is no way out of the current location (the robot is completely surrounded by obstacles), so the robot's entire world is simply modeled by a single place and LPM.[4]

When traveling along a path, the robot may see multiple unaligned gateways and suspect it is at a place. Sometimes, false gateways appear in the LPM due to the boundary between observed `free` space (i.e., white cells in the occupancy grid) and `unknown` space (i.e., gray cells in the grid). This is often the case when the robot's sensors do not provide a 360° field of view, as with SICK-brand lidars. Before the robot commits itself to being at a place, it must perform some local exploration in the fixed map of the potential place

---

[3] One interesting avenue of future research is to try adapting the LPM size by environment characteristics.

[4] There is another degenerate case when the robot is in the middle of a featureless environment. As mentioned in Section 3.3, the HSSH currently does not handle these types of environments.



(a)

(b) **(c)**

Figure 5.2: *Gateway examples.* **(a)** Here we see that gateways should be defined along paths, where the robot is not at a place. Any gateway implementation needs to ensure that two aligned gateways exist along paths. **(b)** At intersections, gateways defined multiple, unaligned local-paths. Thus the robot knows it is at a place. **(c)** At dead-ends, there is a single path; however, there should only be a single gateway, so that dead-ends are places.

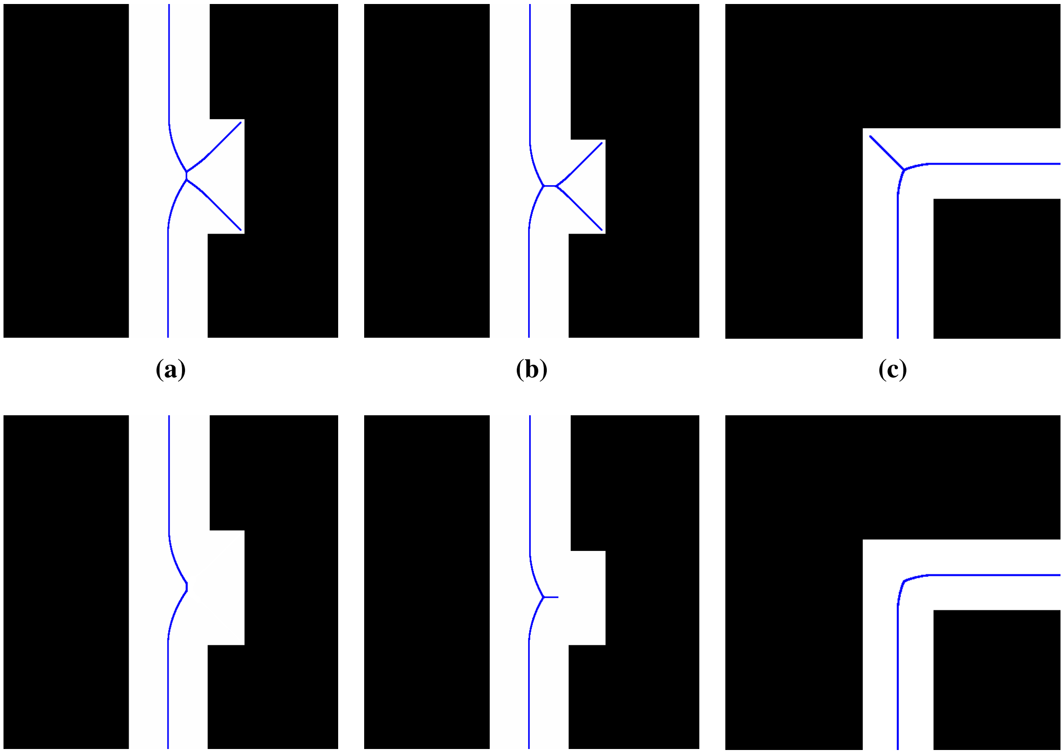
to eliminate any false gateways. We have found that for a robot with a 180° field of view, simply rotating in place eliminates most false gateways. This concept is illustrated as part of a real-world navigation example in Figure 6.24(g,h).

5.3.3 Gateways versus Voronoi Junctions

As discussed in Chapter 2, researchers use the generalized Voronoi graph (GVG) both as a way to discretize the continuous environment into a finite set of places and as a way to define paths [Choset and Burdick, 2000; Choset et al., 2000; Choset and Nagatani, 2001]. Whenever the robot is near a *junction* in the Voronoi graph, it is at a topological place. Junctions are points on the GVG equidistant from three or more closest obstacles in the 2D metrical model of local environment: where the graph forks into multiple branches. The branches that emanate from a junction define the paths that the robot can travel along to leave the place.

Using the local topology defined by gateways allows the robot to detect places more reliably than when using methods that simply look for Voronoi graph junctions. First, non-pruned Voronoi graphs can have many junctions. This is especially true given noisy occupancy grids, but even occurs in the face of no noise at small alcoves and other common architectural features (Figure 5.3(a)). Similarly, complex intersections can have multiple junctions. The gateways and local topology can see one place, whereas a junction-based approach (including Delaunay triangle approaches [Silver et al., 2004]) must define multiple strangely connected places. Figure 5.1(a) shows three junctions in one place.

Pruning is often performed on the Voronoi skeleton, either by fixed-depth pruning [Choset and Nagatani, 2001], distance-based methods of eliminating short branches [Wallgrün, 2005], or by using pixel-based approximations of the Voronoi skeleton, like a thinning-based skeleton [Choi et al., 2002]. These remove many spurious junctions, but in general not all unnecessary branches are removed (e.g., spurs still exist in the pruned skeleton in Figure 5.3(e)). In Section 6.1.1, we discuss a way to prune the Voronoi graph



(a) (b) (c)

(d) (e) (f)

Figure 5.3: *The reduced generalized Voronoi graph (RGVG).* **(a,b,c)** Many places in the GVG are created by “spurs” that are defined by noise or concave corners. **(d)** The RGVG removes branches that terminate with only two nearest obstacles, eliminating many false positive places. **(e)** Unfortunately, this fixed-depth pruning does not remove all undesired branches. **(f)** The RGVG also removes branches that create important junctions, which leaves important places undefined when using algorithms that equate Voronoi junctions with places.

in LPMs that ensures the elimination all spurious junctions. Given pruned skeletons, there exist important places detected via the gateway approach, like L intersections, that may contain no junctions at all after pruning the Voronoi graph (Figure 5.3(f)).

5.4 Selecting Local Motion Targets

Instead of relying on the dynamical system approach to motion used in the basic SSH, we introduce gateways as an alternative approach. Gateways provide a geometric method for controlling motion—where midline or coastal navigation along paths is applicable. The motion of the robot in large-scale space can be adequately captured by noting which oriented gateways the robot passes through. Figure 5.4 illustrates the Hybrid SSH approach to large-scale motion where turns and travels correspond to moving towards gateways.

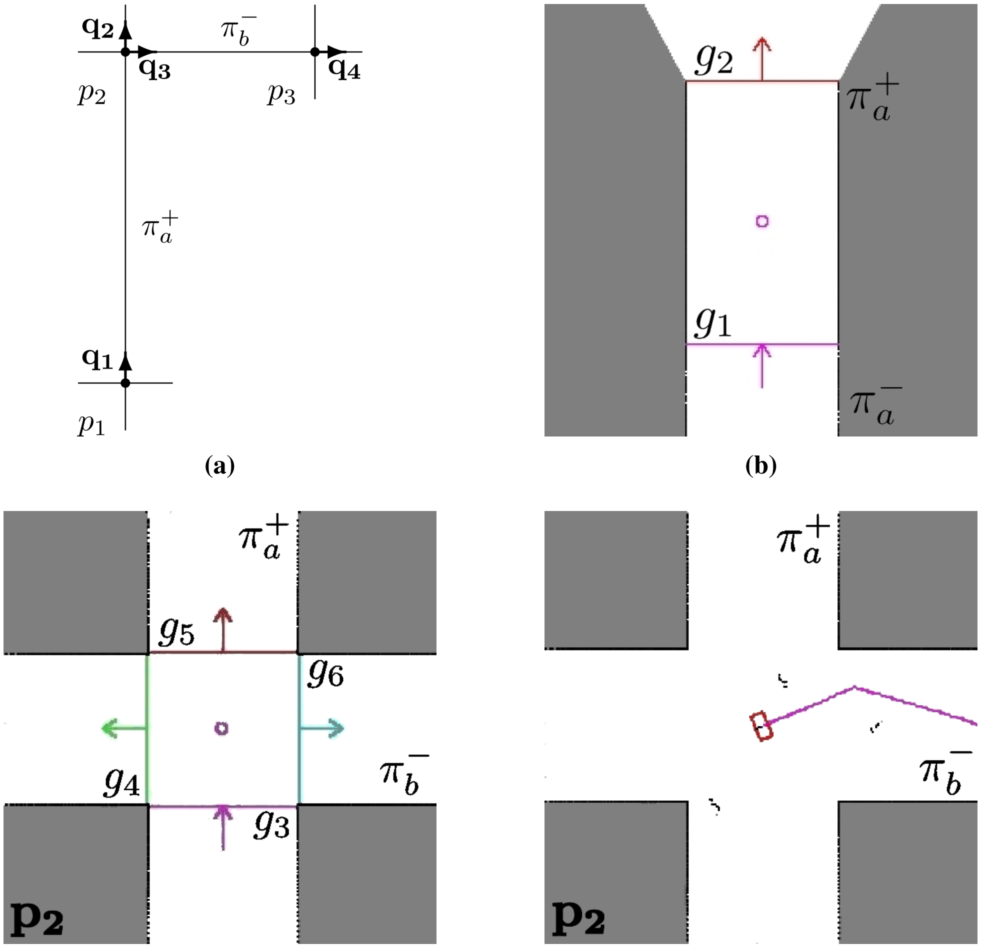
As discussed in Section 4.2, local motion planning consists of selecting a target pose in the LPM, computing a safe trajectory to it, executing the first step of that trajectory, sensing the environment, updating the LPM, and repeating the cycle. The selection of target poses for local motion control corresponds to the action or goal being pursued. There are three distinct cases.

- If the agent is not in a place neighborhood, it is *on-path*, in which case it is moving along the local-path in the LPM toward one of the two gateways. Just beyond the forward gateway, in the outward orientation, is an appropriate target for local motion planning; however, a more robust approach with respect to obstacle avoidance is to aim at a point well beyond the gateway, like the edge of the LPM. As the LPM scrolls, the gateway location is constantly refreshed. The robot never reaches the gateway until its location becomes stable (which only happens when the agent arrives at a place).[5]

- If the agent is in a place neighborhood, the LPM is fixed to the local environment, so

---

[5] Lee [1996] calls control algorithms that continuously re-plan for a moving point ahead as "red wagon" controllers. One could also think of them as "carrot-on-a-stick" controllers.



(a) **(b)**

(c) **(d)**

Figure 5.4: *Grounding control using gateways.* **(a)** The example from Figure 3.3 is further examined in a simulated 3D office environment with obstacles. The gateways are found in real-time (using the anchor-based algorithm discussed in Section 6.3.2), with arrows representing the outward orientations that leave the current area. The gateway associated with the robot's past motion is depicted using an arrow pointing in the inward orientation. **(b)** Traveling along directed path π_a^+ corresponds to aiming beyond an oriented gateway, e.g., $\langle g_2, out\rangle$, in the appropriate direction. The gateway is continually recomputed, which keeps moving the local motion target along the path, until it becomes stable at the entrance to a place. **(c)** Arriving at dstate q_2 at place p_2 corresponds to arriving at a gateway $\langle g_3, in\rangle$ associated with a directed local-path $\tilde{\pi}_a^+$ in the LPM for place p_2. The turn action from dstate q_2 to q_3 corresponds to local motion within the LPM through outward-facing oriented gateway $\langle g_6, out\rangle$ on directed local-path $\tilde{\pi}_b^-$. **(d)** In calculating local topology, "island" obstacles that are surrounded by free space are removed to ensure reliable gateway detection. Planning to move through a gateway requires consideration of these obstacles. Once the robot moves past gateway $\langle g_6, out\rangle$, two new pairs of aligned gateways appear that will flank the robot throughout the next travel action, as in image (b).

motion planning is confined to the small-scale space of the place neighborhood. The agent may have a pragmatic destination within the place neighborhood, for example an intelligent wheelchair may have the goal of bringing its driver to her desk after entering her office, in which case the local motion target is a pose associated with that destination. Such motion targets can also be generated when exploring the fixed LPM of a potential place.

- The agent may be executing a turn action as part of a route through large-scale space. In this situation, the LPM is fixed in the local frame of reference, and a large-scale turn action corresponds to moving from an inward-facing oriented gateway to a location just beyond an outward-facing oriented gateway. After passing through the outbound gateway, the robot is in position to begin following another path. Note, that the TurnAround action simply corresponds to traveling past the same gateway the robot entered the place through, facing the outward instead of the inward orientation. Continuing along a path that passes through a place (no turn) also falls into this case.

In certain scenarios, such as two large rooms connected by a doorway, it may be possible for an agent to move directly from one place neighborhood to another, moving between two distinct local topologies, without ever being significantly *on-path*. The SSH can accommodate this transition with a dummy travel action whose effect is simply to transition between the reference frames of two adjacent, or even slightly overlapping, places. Taking this idea to an extreme, the Atlas system [Bosse et al., 2003] creates new frames of reference based on feature counts, building a “patchwork” map of overlapping frames of references. However, if the entire environment is described in terms of overlapping place neighborhoods, the benefit of the topological map as a concise description of large-scale space is decreased. Likewise, the local and global distinctiveness of places is sacrificed.

Chapter 6

Implementing Gateways

The longest part of the journey is said to be the passing of the gate.

Marcus Terentius Varro, *On Agriculture, Book 1* (1st Century BCE)

We have detailed how gateways can be used to determine the local topology in the nearby surround, facilitating place detection and description. We have also discussed how gateways are used at place neighborhoods to facilitate local motion that corresponds to turns in large-scale space, and how large-scale travel actions correspond to repeatedly moving towards a gateway along a local-path. Some of the figures used to illustrate these ideas (Figures 5.1 and 5.2) have gateways that were generated using various methods we have explored over the years, including some of the techniques discussed in Section 2.3 such as looking for hallways of a certain size, looking for large disparities using radial ray-casting form the robot's position, and looking for "critical points".

Here we discuss two attempts at a more robust gateway algorithm, independent of metrical thresholds. Our gateway algorithms rely on a Voronoi skeleton computed from the free space in the LPM (previously illustrated in Figure 5.1(a)). Table 2.1 compares our two gateway algorithms with others found in the literature. Our initial gateway implementation is a *constriction*-based method that works reasonably well in corridor environments [Beeson et al., 2005]. We discuss this algorithm in detail, and we illustrate a specific problem we have encountered when using constrictions. The second implementation is an *anchor*-based method that generates similar gateways to the constriction-based method in most scenarios, while overcoming the problems of the constriction-based method. Specifically, we demonstrate that this algorithm creates more stable gateways at certain places. This alternative implementation is also necessary to adequately handle coastal navigation (wall-following) scenarios.

6.1 Algorithm Overview

Both of our gateway algorithms are composed of four main stages. This process is illustrated in Figure 6.1. First, a Voronoi-like skeleton must be computed that encodes the free space of the LPM as a 1D graph. This includes any pruning necessary to remove branches caused by noise or branches that represent notches or alcoves but do not connect free space to a new *frontier*.[1] Second, the algorithm must determine the *core* of the LPM, using the skeleton to decide the rough extent of the local place neighborhood. (Note that the same algorithm defines the core of an LPM along a single local-path; thus, a core is not restricted to places.) Third, using the core, the algorithm establishes which portions of the skeleton it will utilize in the search for gateways. Finally, using specific rules, the algorithm places line segments at locations along the graph to represent gateways.

Both of the gateway implementations discussed below use the same steps to generate the skeleton and determine the core of the LPM (though we will discuss an extension to the skeleton in order to handle coastal navigation scenarios with the second implementation). The difference between the two implementations lies in the rules used to detect gateway locations. Below, we discuss the first two stages of finding the minimal skeleton and core of an LPM. Section 6.2 discusses specifics of *constriction*-based gateways, and Section 6.3 discusses *anchor*-based gateways.

---

[1] We use the term frontier to coincide with the ideas of Yamauchi [1997] about exploring occupancy grids by eliminating unknown regions (gray cells) that border free regions (white cells).

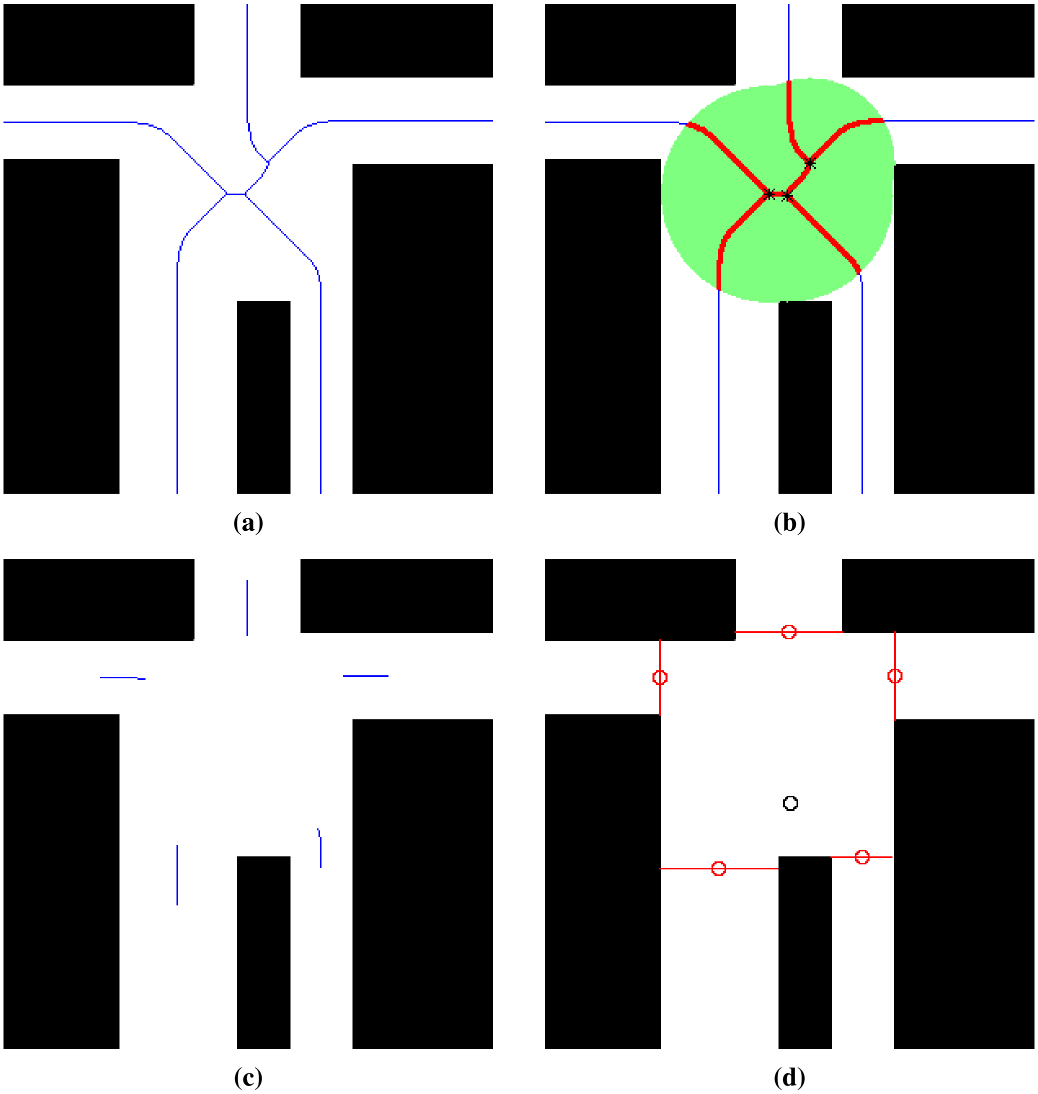


Figure 6.1: *Overview of gateway detection process.* Here we show the four major stages of finding gateways from an LPM. **(a)** The algorithm calculates a pruned Voronoi skeleton. Here we use a thinning-based approximation of the Voronoi skeleton [Zhang and Suen, 1984], which we have found to be much faster to calculate on slower processors. **(b)** The algorithm then determines the "core" of the local region. **(c)** It then uses the core to determine where on the skeleton to search for gateways. In this case, the branches outside the core are used. **(d)** The algorithm looks for reasonable gateway locations. The robot's position is denoted by a dark circle.

6.1.1 The Voronoi Skeleton of an LPM

The first step in finding gateways is to create a high-quality skeleton of free space from the current snapshot of the LPM. All gateway processing (object removal, skeletonization, pruning, etc.) on the occupancy grid representation of the local surround is done in a copy of the LPM. For example, if we state that free cells are "flipped" to occupied cells, we are referring to a copy of a recent LPM snapshot, which can be modified without affecting the actual LPM or the underlying SLAM algorithm.

Additionally, this work uses a discrete approximation of the Voronoi graph. In discussions on pruning graphs, finding the core of LPMs, and creating gateways, if we refer to Voronoi graphs, we mean any Voronoi-like skeleton. Similarly, when we discuss Voronoi junctions or Voronoi branches, we are referring to equivalent junctions and branches of the Voronoi-like skeleton.

Various Skeleton Algorithms

Multiple skeleton algorithms exist that give some approximation of the continuous Voronoi graph. Computing the precise, continuous Voronoi graph assuming point obstacles can be computed in $O(n \log n)$ time, where n is the number of obstacles in the world [Fortune, 1992]. For discretized models, Voronoi approximations can be computed by so-called "brush-fire" algorithms: imagine fires at all occupied cells that burn inward through free (fuel) cells at a constant speed such that a skeleton is created at all points where two or more fires meet. Such algorithms can be computed in $O(n)$ time, where n is the number of cells.

We prefer to utilize these discretized algorithms for several reasons. First, they are easy to implement. They also allow us to gather related metrical information during the skeletal search (e.g., which obstacles are closest to the currently considered pixel) with little overhead. Finally, they are easy to extend to modified Voronoi graph definitions (Section 6.3.1).

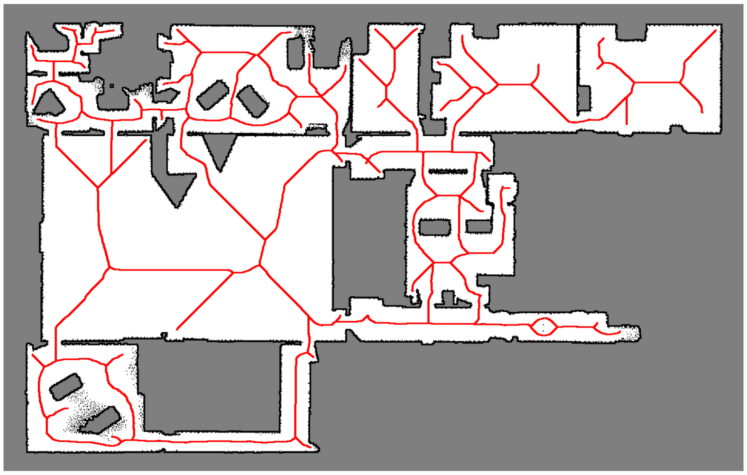


Figure 6.2: *The thinned skeleton of a global metrical map.* Using an occupancy grid, the skeleton of free space can be computed by using any occupied cells as obstacles. Junction points include intersections in corridor environments. For visual clarity, portions of the graph that are close to obstacles are removed. Compare this to the GVG in Figure 2.3.

Specifically we use a skeleton created by thinning[2] [Zhang and Suen, 1984]. Thinning is similar to the “brush-fire” algorithm, though it has very specific rules about whether a cell is flipped from “fuel” to “fire” based on its neighboring pixels. Thinning roughly approximates a subset of the Voronoi graph (Figure 6.2). A thinned skeleton has all the important skeletal branches, but allows efficient post-processing (e.g., the pruning discussed below), as many spurious terminal branches are never generated. However, thinned skeletons tend to contain long straight linear segments, thus they are quite dependent on the orientation of paths. Since our methods can properly detect gateways on these thinned skeletons, regardless of orientation (Section 6.3.3), we are confident that they can handle skeletons that better approximate the true Voronoi graph, which is independent of the orientation of point obstacles.

Discrete skeleton implementations like thinning or brush-fire algorithms flip free

---

[2] Our thinning implementation is publicly available on the web [Beeson, 2006]. It provides pre/post-processing options for removing obstacles, frontier-based pruning, and setting minimum and maximum distances from obstacles. In addition to being used in our work for determining gateway locations, this particular thinning implementation is also used for converting graphical representations of chemical structures in electronic documents to text-based chemical names [Filippov, 2007].

cells to occupied cells until a thin set of connected free cells remain; however, in the LPM, cells can be in one of three states: occupied, free, or unknown. It is important that we consider unknown cells in the LPM (gray cells in the images) to be free cells when computing the skeleton (Figure 6.3(b)). If unknown cells are used as obstacles, small patches of unknown cells (unseen by the rays of the lidar) can greatly affect the algorithm, creating skeletons that stay away from frontiers. Such skeletons are not useful for an exploring robot that needs to hypothesize unknown space as potentially free. Additionally, as we show below, efficient pruning makes use of *exits* where the skeleton touches the edge of the LPM or where free cells border unknown cells.

Pre-processing Obstacles

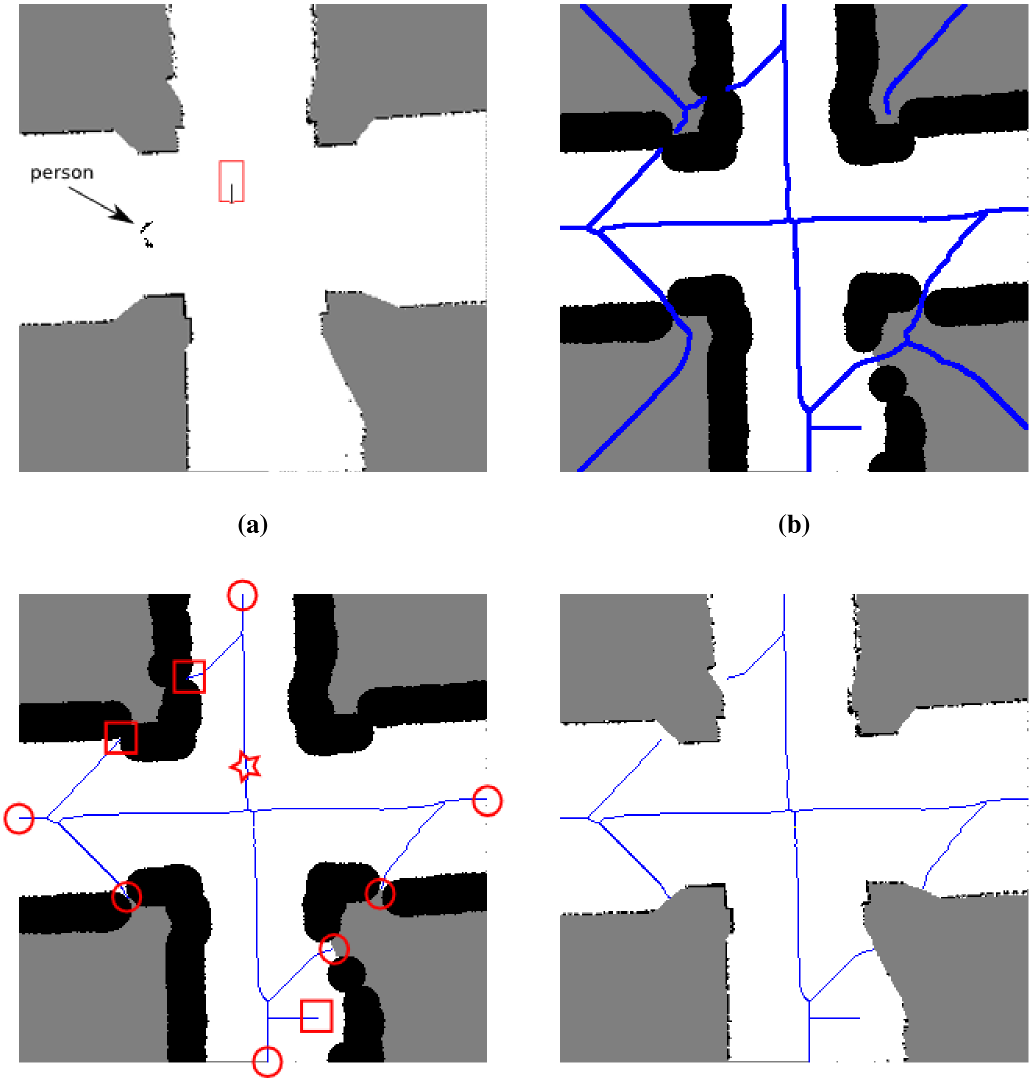
Any skeleton algorithm is going to be affected by noise and other non-architecture obstacles. We perform pre-processing on detected obstacles prior to generating the skeleton. This improves the quality of the generated free space skeletons.

First, we remove "island" obstacles from the LPM (Figure 6.3(a,b)). Groups of occupied and unknown cells that are completely surrounded by free cells are assumed not to affect the structure of the paths and places in the local region; thus, they are considered free cells in computing, pruning, and examining the local skeleton. Pedestrians, boxes in hallways, and thin posts are common examples of obstacles (often with neighboring unmapped regions due to the occluding obstacle) that can be easily removed from the LPM before computing a skeleton of the surrounding free space.[3]

We are able to utilize an obstacle removal algorithm that is linear (instead of the straightforward $O(n^2)$ algorithm) due to the fact that we are using an LPM with bounded size. We find all non-free cells on the border of the LPM. We then recursively add neigh-

---

[3] Pathological cases exist where wall segments between nearby open doors (or large, load bearing columns) can be seen as "island" obstacles. Similarly, many kitchens contain "island" counters that may be appropriate for a service robot to consider architecture. We assume such problems will be overcome as robotic perception improves, as these are obviously permanent architectural obstacles. In the meantime, we could simply keep obstacles that are larger than a predetermined size in order to avoid these problems; however, such a metric assumes outside knowledge about the environment, which we are trying to avoid in this work.



(a)

(b)

(c)

(d)

Figure 6.3: *Finding the skeleton of LPM free space.* **(a)** The algorithm starts with an LPM the robot has built as it moves through the environment. **(b)** “Island” obstacles are removed, and the remaining obstacles in the LPM are increased by a predetermined size to fill in gaps too small for the robot to pass through. The skeleton of non-occupied space is then found, treating unknown cells as free cells. This ensures the skeleton does not have artifacts due to treating unknown regions as obstacles. **(c)** The algorithm then determines the closest point on the local, free space skeleton to the robot (the star in the image). It crawls the skeleton, following branches only until the skeleton finds *exits* (circles in the image) or until branches terminate (squares in the image). **(d)** The skeleton (prior to final pruning) is simply a subset of the original skeleton examined by this skeletal crawl.

boring non-free cells, essentially finding the collection of all such cells that have a path to the edge of the LPM via non-free cells. All cells not in this set are set to free. This is an effective and efficient "island" removal method for LPMs.

Obstacles that move can be detected by simple extensions to the normal occupancy grid [Modayil and Kuipers, 2004]. If previously free cells become occupied, then obstacles that reside in those cells can be classified as "moveable", and the occupied cells can be changed to free before creating the skeleton. For example, if the robot notices a door closing, it will classify the cells occupied by the closed door as free for purposes of creating the skeleton. Future work on visual obstacle detection should allow us to remove stationary doors, furniture, and pedestrians from the LPM, while keeping walls and other architecture.

Finally, we "bleed" obstacles by a safety-radius—usually half the width of the robot plus some small safety margin (Figure 6.3(b)). This closes small, irrelevant gaps between obstacles, which reduces spurious skeletal branches and helps speed up post-processing steps like pruning. This step might seem unnecessary at first glance since branches near to obstacles can be pruned after the skeleton is fully computed. However, we have found this step convenient for two reasons: (1) it can be added as a side effect in the initialization of a discrete skeletonization algorithm, so it does not greatly increase the overall runtime of the algorithm; (2) it serves to smooth the discretized skeletons, as the "bled" obstacles have softer, less noisy corners.

Frontier-based Skeleton Pruning

The RGVG [Choset and Nagatani, 2001] was introduced as a way to eliminate some of the "weak" junction points caused by spurious branches; however, the algorithm cannot ensure all spurious branches are removed. The problem with pruning the Voronoi graph is that it is often difficult to determine what branches are part of the necessary, "base" skeleton and which branches are non-essential. In order to determine the necessary portions of a Voronoi skeleton, we again leverage the fact that the robot is computing the skeleton of a *bounded* local perceptual map (LPM). We use the frontiers in the LPM to determine which branches to keep, ensuring that all spurious branches are removed.

Given the small, manageable size of the LPM, the robot should never find itself completely enclosed by occupied cells in large-scale environments. Exceptions could be elevators or small rooms where doors get closed. In these instances, our LPM should detect the dynamic doors [Modayil and Kuipers, 2004], allowing the algorithm to ignore these as obstacles for the purposes of building the skeleton. The frontiers at the edge of the LPM or at the boundary between free and unknown regions of the LPM allow us to easily prune the graph, keeping only useful branches in the LPM.

The algorithm for pruning the skeleton in the LPM is a straightforward three step process. First, the algorithm determines the closest skeletal point to the robot that has a path to the robot through free space—here, it does not consider portions of the skeleton that overlay onto unknown regions (gray cells). Determining this closest point is necessary as occasionally multiple non-connected branches may exist after skeletonization: e.g., cases where older portions of the map are fully separated from the locally reachable free space by obstacles. We use this closest point to provide information on which of the disjoint, non-connected branches must be kept during pruning.

The algorithm then "crawls" the graph from the closest skeletal point, looking for *exits* (circles in Figure 6.3(c)) and for locations where the skeleton branches terminate (squares in Figure 6.3(c)). Exits are points on the skeleton that neighbor the edge of the LPM or neighbor unknown space (essentially skeletal points that lie on a frontier). The algorithm stops crawling a branch when it detects exits, as graph points in unknown regions will be pruned away. If we consider only the points considered by this crawling algorithm, the pruned skeleton would be the graph of connected points that overlay free cells in the local map (and have a path through free space to the robot's location) (Figure 6.3(d)).

There may be some spurious branches that do not terminate at exits. Instead of actively pruning away these branches (which is difficult, as it is hard to determine how far to prune branches), our approach is to compute the minimal spanning tree (MST) within the skeleton between all exits in the LPM (Figure 6.4). We *keep* the MST, removing any other graph points from the skeleton. This ensures that no spurious junctions or branches exist in the LPM. In the case of dead-ends, only a single exit exists. In this special case scenario, the MST that connects the exit with the closest skeleton point to the robot is used (Figure 6.4(f)).

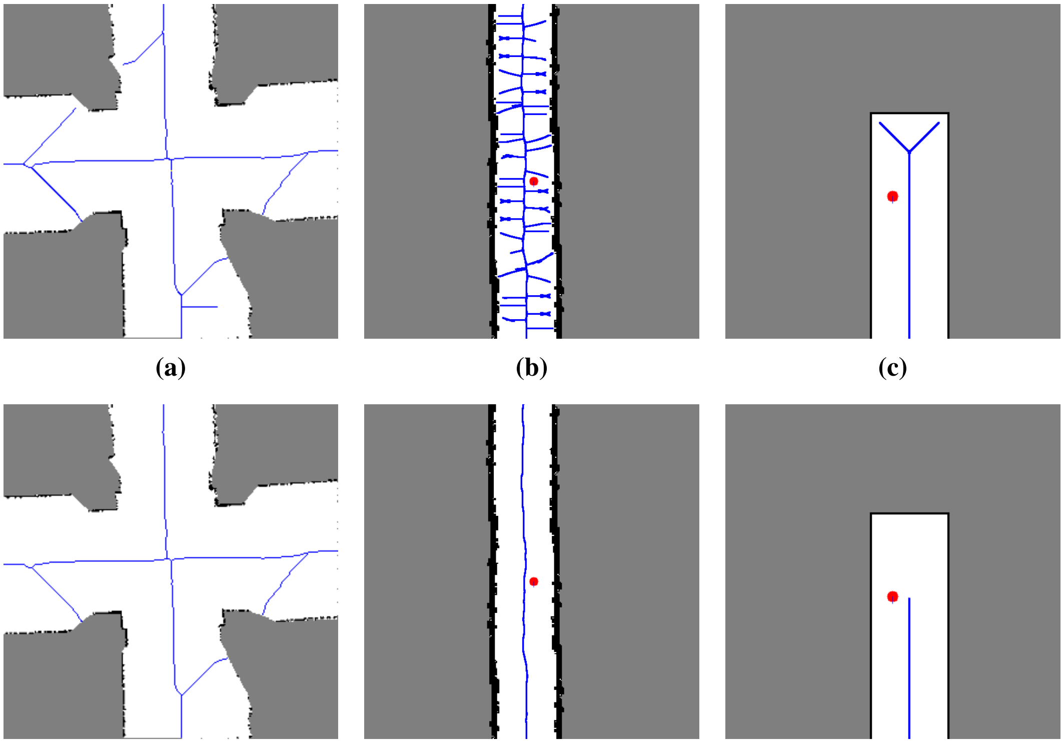
6.1.2 Defining the LPM Core

Once a fully pruned skeleton of free space is found, the gateway algorithm must look for locations to put gateways. As part of this process, we must first define the *base points* and the *core* of the local area, using the pruned Voronoi skeleton. Simply stated, we define a set of base points nearby the robot (usually neighboring Voronoi junctions), and the core is the set of skeletal points within the radius of the base points. In Algorithm 6.1, we formally describe the method by which we determine the core of the LPM.

The core serves several purposes. It gives a coarse approximation of the area the place encompasses. Similarly it gives an environment-based method for defining when to aggregate together multiple areas into a single, complex place and when nearby areas should belong to different places. Finally, it gives rough locations (near the edge of the core) to start searching for gateways. Figure 6.5 illustrates the concept of the core of an LPM with some examples.

6.2 Constriction-based Gateways

Once the robot has generated a minimal skeleton of the LPM and has computed the core of the local region, it can begin to look for gateways along the skeleton. Our first attempt at determining gateways examines the skeleton near the edges of the core, looking for *constrictions*. Algorithms 6.2 and 6.3 detail the process that finds gateways by looking for constrictions in specific regions of the skeleton. In this work, we define a constriction on a



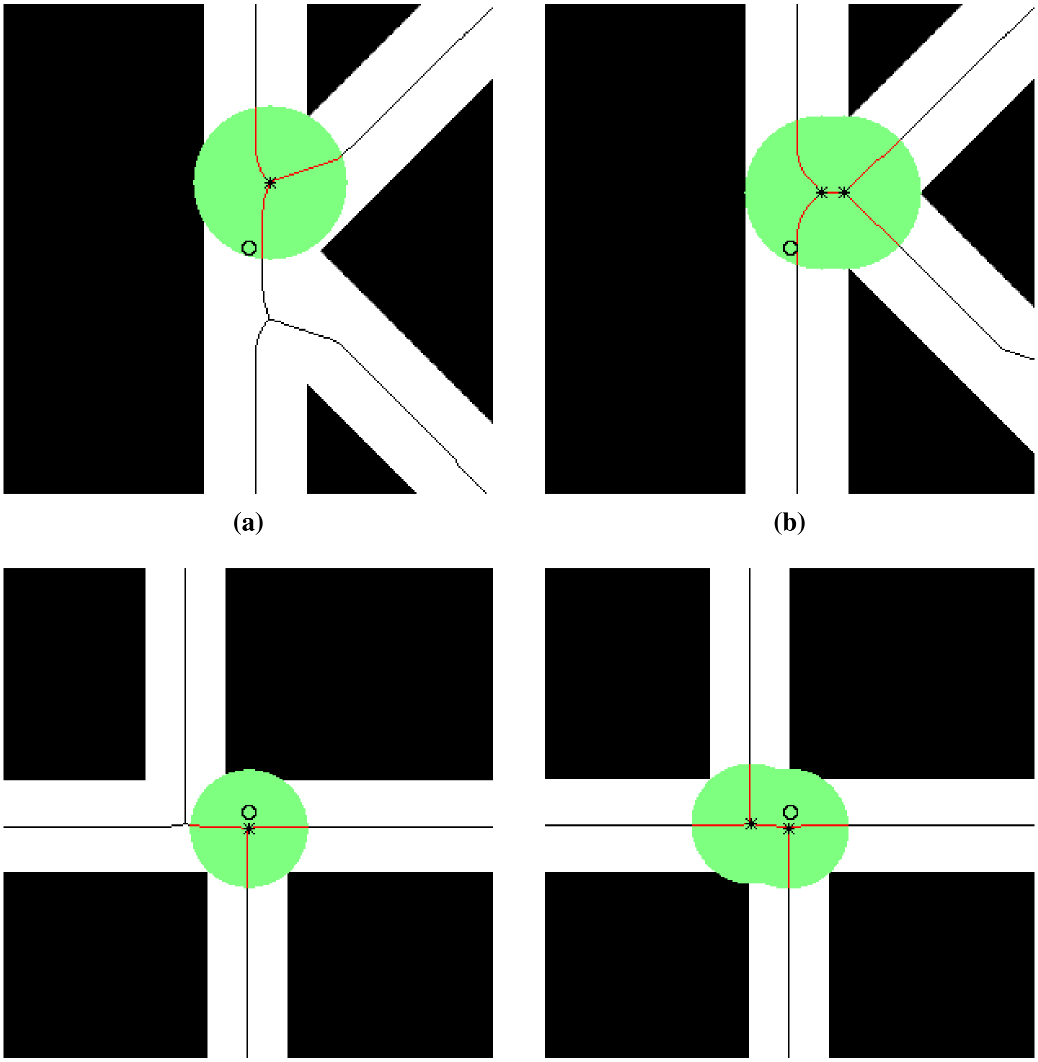
(a) **(b)** **(c)**

(d) **(e)** **(f)**

Figure 6.4: *Pruning a Voronoi graph using the LPM frontiers.* Pruning is performed by keeping the minimal spanning tree that connects all *exits* of the skeleton. **(a,b,c)** Examples of Voronoi graphs at common location. **(d)** The final pruning step for the skeleton from Figure 6.3. This removes portions of the skeleton that do not connect exits. **(e)** Skeleton algorithms that are sensitive to noise may create many branches, even in qualitatively simple LPMs. These spurious branches are properly pruned using the LPM exit method. **(f)** Dead-ends are a special case where we keep the minimal branch that connects the single exit to the skeletal point closest to the robot.

```
$l$ is the location of the physical robot w.r.t. the LPM.
$P$ is the set of points that constitute the pruned (MST) skeleton.
$dist(x,y)$ is the Euclidean distance between two points.
$c \in P$ is the skeletal point closest to $l$ that also has a path through free space to $l$.
    (Re-computed after finding the MST that connects the skeletal exits).
$r_p$ is the "Voronoi radius".
    (The distance from Voronoi point $p \in P$ to the closest obstacle).
$J$ is the set of Voronoi junctions $j$.

Define the base points:
    Two junctions $j$ and $j'$ are neighbors if $dist(j,j') \leq max(r_j, r_{j'})$.
    Determine the nearby junctions $K = \{j \in J : dist(j,c) \leq r_j\}$.
        ($c$ is within the "radius" of the junction point).
    Define the base points $F$ of the place neighborhood as the equivalence class of
        neighboring junctions that uses $K$ as the starting set of junctions.
        (asterisks in Figure 6.5)
    If $F \equiv \emptyset$, then $F = \{c\}$.
        (If no nearby junctions exist, use $c$ as the base point.)

Define the core:
    Define the core as the set of skeletal points $Q = \{q \in P : \exists f \in F\ [dist(f,q) \leq r_f]\}$.
```

Algorithm 6.1: *Finding the core of an LPM from a pruned free space skeleton.*



(a) **(b)**

(c) **(d)**

Figure 6.5: *Examples of the base points and the core of different local environments.* The robot is always assumed to be exactly in the center of the images and is shown as a circle. Skeletal points that define the base points (Voronoi junctions at places and the closest point to the robot along paths) are shown as asterisks. Skeletal points that lie inside the core are shaded lighter than skeletal points outside the core. For visual clarity, the figures also shade non-skeletal points within the radius of the base points.

```
  $skel\_dist(x,y)$ is the graph distance between two skeleton points.
  $on\_path(w,x,y)$ determines whether point $w$ lies on or between points $x$ and $y$ on the
    skeleton.
    (Because the pruned skeleton is an MST, cycles cannot exist, so this is a
    deterministic function.)

Define the gateway search spaces:
(This assumes functions and sets defined in Algorithm 6.1.)
  Define the set of skeletal points on the edge of the core
    $E = \{e \in Q : \exists f \in F\ [dist(f,e) = r_f \wedge \forall f' \in F\ [dist(f',e) \geq r_{f'}]]\}$.
  For each $e \in E$, define a search space on the skeleton branch
    $S_e = \{s \in P : dist(e,s) \leq r_e \wedge \neg on\_path(c,s,e) \wedge \forall j \in J\ [\neg on\_path(j,s,e)]\}$.
    (Collect points close to the core edge. Unlike with constrictions, these can be inside
    the core. Junctions, and points separated from the core edge by junctions, are
    excluded. Likewise, the closest skeletal point to the robot $c$ and points separated
    from the core edge by $c$ are excluded.)

Find a gateway location hypothesis for each search space $S_e$:
  Define the set of minima $M_e = \{m \in S_e : \forall s \in S_e\ [r_m \leq r_s]\}$.
  $h_e \in M_e$ is a constriction, where $\forall m \in M_e\ [skel\_dist(e,h_e) \leq skel\_dist(e,m)]$.
  The set $H = \{h_e\}$ hypothesizes the locations of the gateways.
```

Algorithm 6.2: *Finding constrictions of an LPM.* We assume the robot already has computed base points and a core of the LPM. See Algorithm 6.1.

set of skeleton points P as a minimum in the Voronoi radii $R = \{r_p : p \in P\}$ (i.e., the same rule that defines the "critical points" of Thrun and Bücken [1996]). At constrictions we define gateways as line segments that connect nearby obstacles.

Thrun and Bücken [1996] specify the size of regions in which to look for constrictions. In order to create a scale/environment independent method, our algorithm's search areas are a function of the skeleton of free space. We define the portions of the skeleton that lie near the core edge, within the Voronoi radii of the skeleton points on the edges of the core, as the constriction search areas. Figure 6.6 illustrates the need to limit the search areas for constrictions, as detailed in Algorithm 6.2. Looking too far away from the core can find local minima that are well behind the place neighborhood boundaries that we would like to define, reducing there usefulness in defining the boundaries and qualitative structure of places.

```
Calculate the gateway line segment for each gateway location $h_e$:
(This assumes functions and sets defined in Algorithms 6.1 and 6.2.)

  $n_e = h_e$.
  REPEAT
    (In general, the two closest obstacles to a skeletal point are not necessarily collinear
    with the point. This both forces collinearity and finds the exact Euclidean
    constriction.)
      Find the line segment $z_e$ that connects the closest two obstacles on each side of
        the skeleton branch to $h_e$.
      Determine the skeletal point $\hat{n}_e \in P$ where $z_e$ intersects the skeleton.
      IF $\neg on\_path(c, \hat{n}_e, e)$
        (Make sure the robot is not between the proposed gateway location and the
        edge.)
      THEN $n_e = \hat{n}_e$.
  UNTIL $n_e$ becomes stationary.
  Define the "inner-neighbors" $T_e = \{T \in P : dist(n_e, t) \leq r_{n_e} \wedge \forall f \in$
    $F\ [skel\_dist(f, t) < skel\_dist(f, n_e)] \wedge \forall j \in J\ [\neg on\_path(j, t, n_e)]\}$.
    (Collect points close to the gateway location, but closer to the core than the gateway
    location. Junctions, and points separated from the gateway location by junctions,
    are excluded.)
  Compute the headings $B_e$ from $n_e$ to each point $t \in T_e$, and the distances
    $D_e = \{dist(n_e, t) : t \in T_e\}$.
  Define the "outer-neighbors" $T'_e = \{t' \in P : dist(n_e, t') \leq r_{n_e} \wedge \forall f \in$
    $F\ [skel\_dist(f, t') > skel\_dist(f, n_e)] \wedge \forall j \in J\ [\neg on\_path(j, t', n_e)]\}$.
    (Collect points close to the gateway location, but further from the core than the
    gateway location. Junctions, and points separated from the gateway location by
    junctions, are excluded.)
  Compute the headings $B'_e$ to $n_e$ from each point $t \in T'_e$, and the distances
    $D'_e = \{dist(n_e, t) : t \in T'_e\}$.
  Calculate the weighted average $v_e$ across $\{B_e \cup B'_e\}$, where $\{D_e \cup D'_e\}$ provides the
    weights.
  Generate a line segment $g_e$, centered at $n_e$, of size $2 \cdot r_{n_e}$ (twice the Voronoi radius).
    The orientation of the segment is normal to $v_e$.
  The set $G = \{g_e\}$ represents the gateways of the LPM.
```

Algorithm 6.3: *Finding constriction-based gateways.* We assume the robot already has computed search areas and constrictions near the edges of the LPM core—see Algorithm 6.2.

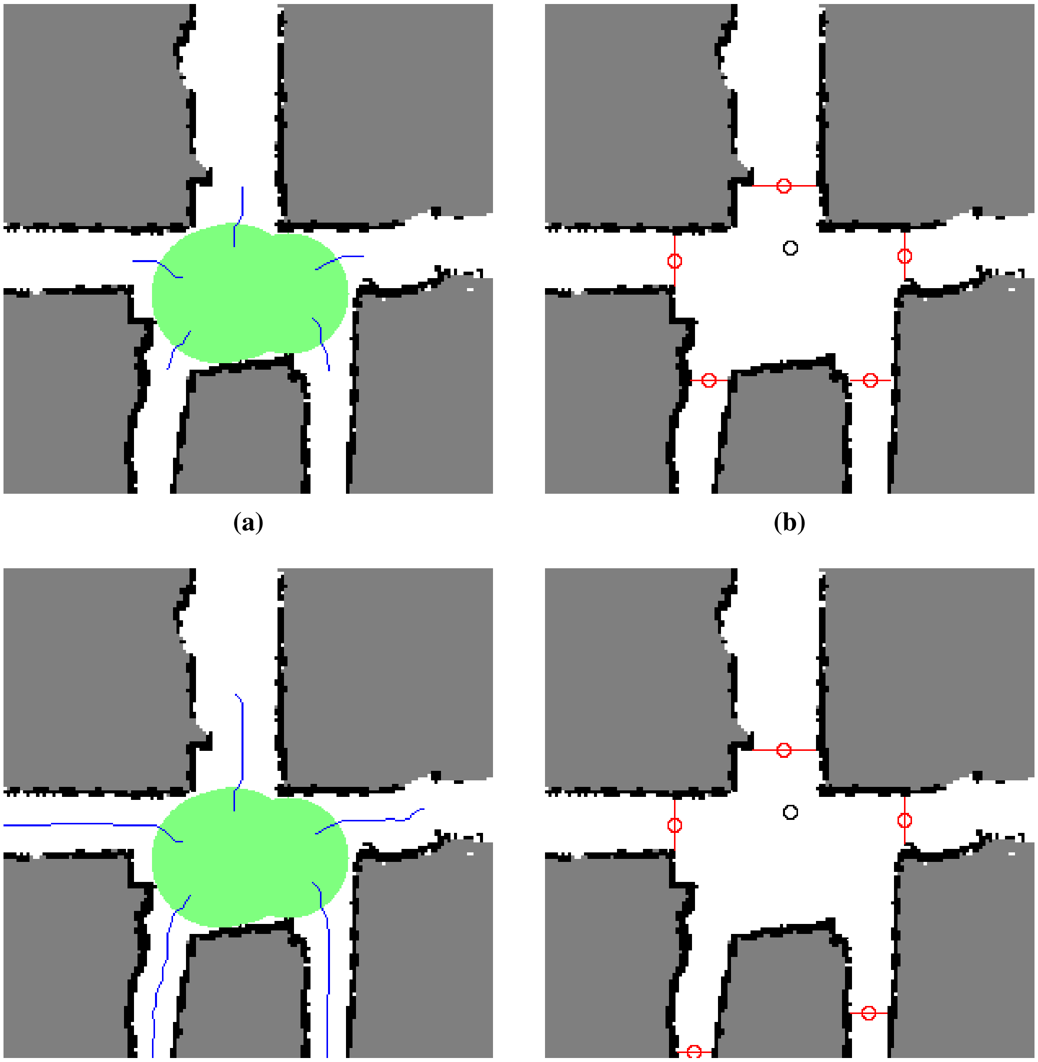


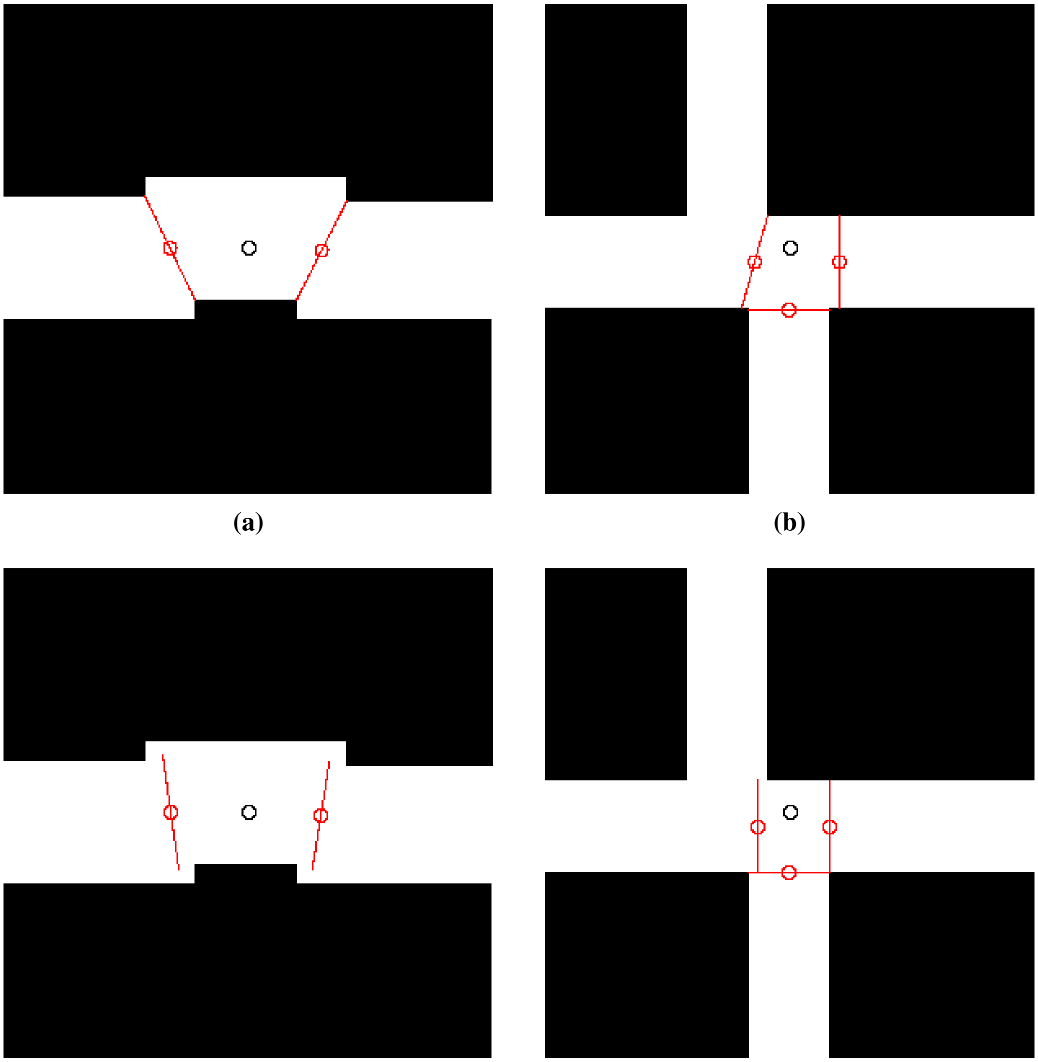
Figure 6.6: *Defining the search space for constriction-based gateways.* Here we show constriction-based gateways on the environment used in Figure 5.1(a). **(a,b)** We find reasonable constriction-based gateways when using the search areas we define in Algorithm 6.2. (The search areas are within the Voronoi radii of points at the edge of the core, which is lightly shaded). **(c,d)** Suppose we do not limit our search areas to be near the edge of the core (but we still limit the search area to not include junctions). The danger here is that we can find minima far away from the core that do not adequately represent the place boundaries.

Once “critical points” are found, Thrun and Bücken [1996] simply connect the two closest obstacles that define the Voronoi skeleton at that point. In our algorithm this would be akin to defining gateways as the set $\{z_e\}$ from Algorithm 6.3. However, because noise and obstacles can create constrictions, it is possible that the line segments z have strange orientations with respect to the actual path. Figure 6.7(a,b) illustrates how this simple rule can fail, creating gateways that could potentially be unaligned along a path due to small obstacles near the path boundaries. This motivates the need for a more complex method for determining line segments G from $\{z_e\}$ that is described in Algorithm 6.3—essentially, neighboring skeletal points are used to determine a reasonable orientation of the path at the gateway location (Figure 6.7(c,d)).

6.2.1 Discussion

This constriction-based gateway algorithm works reasonably well in indoor office environments. A recursive version of this algorithm was implemented that runs quickly enough to recompute gateways in real-time—averaging ∼2 Hz for a 300x300 cell occupancy grid (usually we use 15 meter grids with 5 cm cells) on a Pentium III 450 MHz research robot computer that is also running the computationally intensive SLAM algorithm to maintain the underlying LPM. Examples of gateways using this implementation are shown in Figure 6.8.

This implementation was also used to generate the gateways used for autonomous place detection and description in our large-scale map-building environment from Figure 2.1; however, we used an artificially large robot radius in computing the skeleton to prevent skeletal branches between office doors and cubicle opening. This allowed the robot to detect gateways in hallways and intersections, but prevented our robot from exploring people’s offices or desk areas. Chapters 8 and 9 illustrate global topological and metrical map-building using places and local topologies grounded using this gateway implementation; Figure 8.2 shows the places detected by using this constriction-based method.



(a) **(b)**

(c) **(d)**

Figure 6.7: *Defining gateways by connecting nearby obstacles.* **(a,b)** A simple rule once a constriction is found is to simply connect the closest obstacles on both sides of the skeleton. This is easy to implement but often leads to gateways that become oriented with respect to noisy walls or obstacles against walls. In image (a), the simple path continuity criteria (Section 5.2.2) fails because of misaligned gateways. **(c,d)** Algorithm 6.3 gives a more complex way to determine the line segment once a constriction is found. This method yields very good results in practice, resulting in high-quality local topology abstraction. Figure 6.8 provides more examples of constriction-based gateways as defined by Algorithms 6.2 and 6.3.

In Table 2.1, we compared a variety of gateway approaches, and stated that our constriction-based approach performs well when considering a variety of desired properties. First off, this approach is relatively scale independent. Suppose we have a 15x15 meter LPM with 5 cm cells that models a T intersection of 3 meter wide hallways. Suppose we also have an equivalent grid that is actually a 150x150 meter LPM with 50 cm cells that models a T intersection of 30 meter hallways. We expect to see very similar Voronoi skeletons, cores, constrictions, and gateways. That is, our approach does not have built-in thresholds about hallway size. Additionally, constrictions are well defined at doorways and at the beginnings of hallways, so this approach finds the kinds of gateways we want in the locations we expect. Because this approach uses a Voronoi skeleton, we also do not assume rectangular intersections, and can find gateways in a variety of intersections. This algorithm also works well in defining gateways along paths.

6.2.2 Problems with the Constriction Algorithm

The constriction approach works well in most indoor environments, which makes it equal to or better than previously published gateway approaches discussed in Section 2.3; however, as implied by Table 2.1 in certain situations, we found the constriction-based locations of gateways to be unreliable. Below we illustrate these problems with specific examples.

Figure 6.9 illustrates issues that occur at places where no junctions exist (i.e., L intersections). In these situations, constrictions can exist on portions of the skeleton where the branch is about to curve sharply. As a result, our method for determining the orientation of gateways can create strangely oriented gateways.

The strange orientation issue is mostly aesthetic, as the local topology derived from the gateways in Figure 6.9(b) does in fact correspond to an L intersection; however, we still consider this an issue to be fixed, as it may impact the comparison between two places (Section 9.3.1 discusses how we can use gateways to quickly align two LPMs in creating a globally consistent metrical map). Strangely oriented gateways may also affect human-

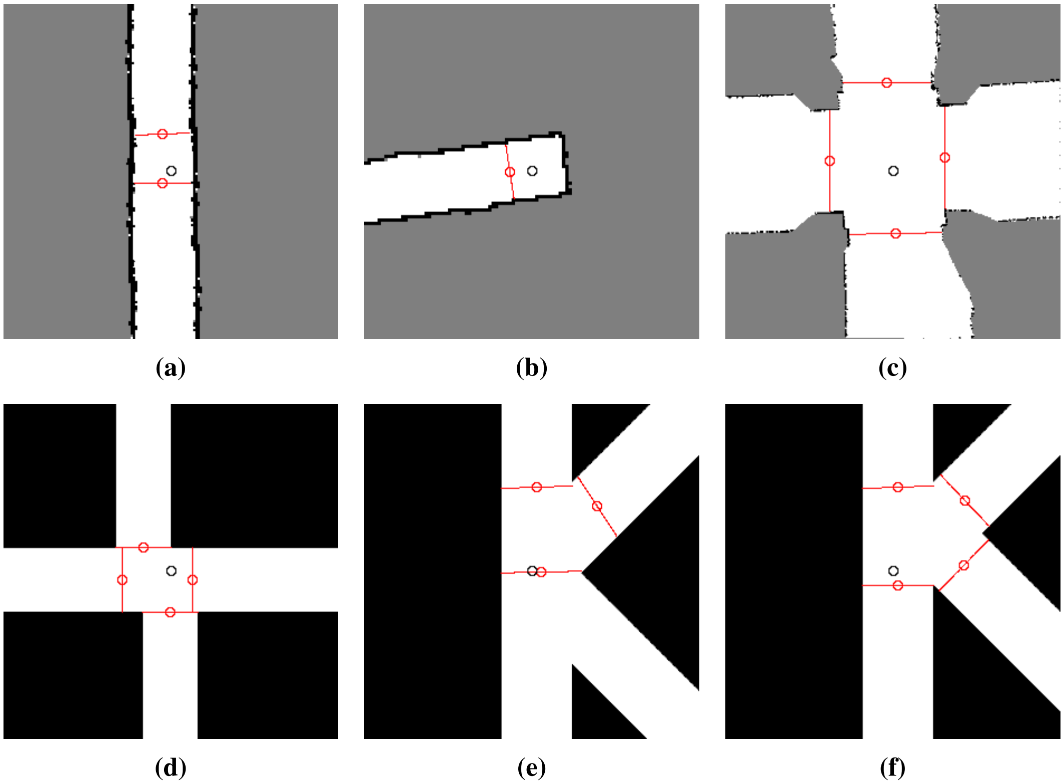
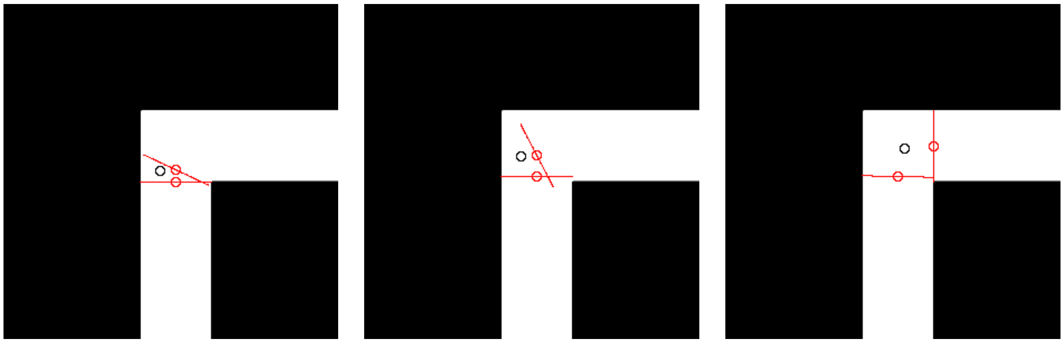


Figure 6.8: *Examples of constriction-based gateways in a variety of (real-world and pathological) environments.*



(a) (b) (c)

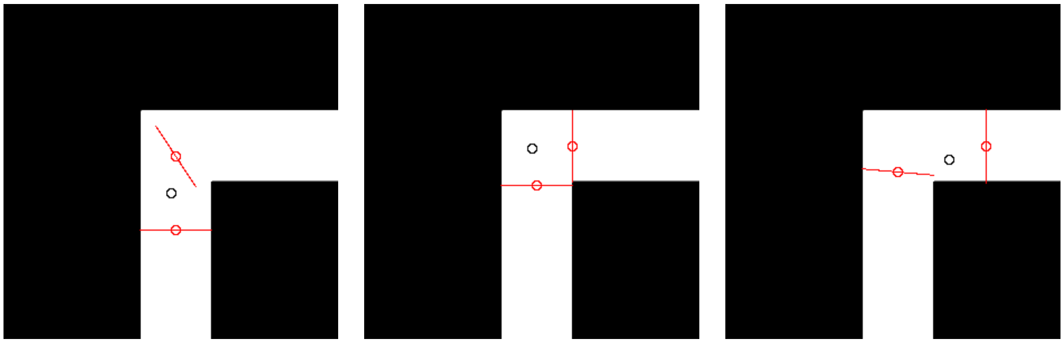
Figure 6.9: *Unstable gateways near* L *intersections.* **(a)** Here we see a strange gateway orientation because the constriction lies close to the sharp bend in the skeletal branch. **(b)** Here we see another strange gateway after to robot has moved further into the intersection. Here the local topology corresponds to an L intersection. **(c)** Once the robot moves further into the interior of the place, gateways look more appropriate, showing that gateways at L intersections change with respect to the robot's location in the place.

robot interactions in future work.

More important than gateway orientation is the issue that the place gateways change as the robot moves through the place. This has the potential to create mapping problems as the boundaries of places change with respect to the robot's location in the place. In Figure 6.9(b), the robot is at a place, and chooses a gateway to pass through to begin navigation along a path. Once it passes through the gateway, the robot detects that it is still at a place (Figure 6.9(c)). At this point it will consider this another place in the HSSH Global Topology Level. In order to overcome this problem, we must add special knowledge to handle such a scenario for *L* intersections, which is undesirable.

One might think that changing the search area to only look for gateways *outside* the core might fix these problems. Figure 6.10 demonstrates this change to the algorithm. We see that even here, we can get strangely oriented gateways, and that the boundaries of the place can be unstable. Even worse, with this change to the search area, the robot will detect places, even when it is well into a hallway (Figure 6.10(a,c)).

Additionally, once we extend our skeleton to handle *coastal navigation* (wall-following)



(a) **(b)** **(c)**

Figure 6.10: *Unstable place boundaries with constriction-based gateways.* Here we show the results of looking for constrictions only *outside* of the core. **(a)** We still see strangely oriented gateways as in Figure 6.9(a,b). **(b)** Here the robot is at a place with reasonable gateways. At this point, if the robot decides to leave the place via the gateway on the right of the image, it uses the gateway as a local motion target (Section 5.4). **(c)** After the robot completes the local motion to pass through the rightmost gateway from image (b), the robot finds itself still at a place. At the HSSH Global Topology Level, the robot must try to determine whether this is a new place or the same place. A new problem arises when looking outside the core, as the robot considers this a place even though it is well into the hallway.

scenarios (Section 6.3.1), constrictions will not exist (in theory)—points along the skeleton will be equidistant from the nearest obstacles. However, for discretized skeletons, constrictions can exist, though they are not visible because the change in radius is extremely small. These constrictions are artifacts and may be at locations unsuitable for gateways (Figure 6.11(a)). When no constriction is found in a search area along a branch, a gateway is located at the edge of the core by default. Here, the gateways become dependent on the robot's location, and we observe problems of badly oriented gateways and unstable gateways at places throughout coastal navigation (Figure 6.11(b)).

6.3 Current Gateway Algorithm

The intuition behind the constriction-based method is that the architecture in the environment creates "cinch points" or constrictions in the Voronoi radii near doorways and at the intersection of hallways. As we demonstrated above, this works well in the majority of situ-

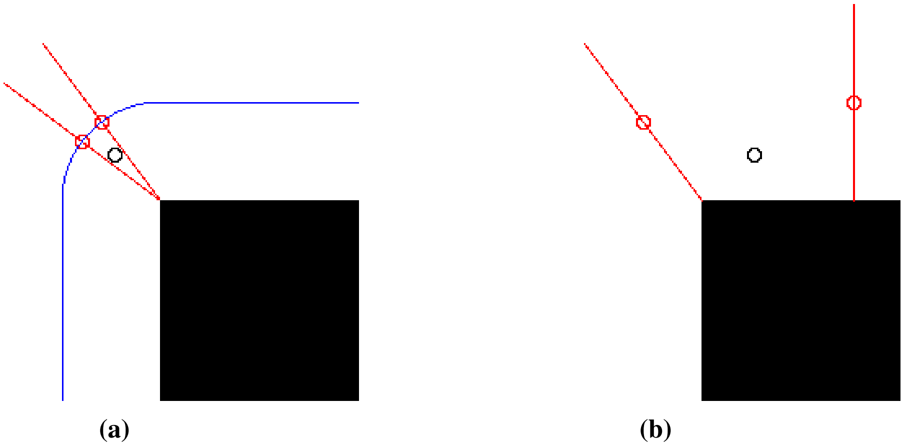


Figure 6.11: *Unstable gateways in coastal navigation.* In coastal navigation scenarios, using the extended Voronoi graph (Section 6.3.1), there are theoretically no constrictions. **(a)** However in discretized skeletons, small local minima do arise, which can create poor gateway locations. **(b)** In cases where no gateways exist, the gateways are places at the core edges by default. Thus, in coastal navigation of convex corners, the problems shown in Figures 6.9 and 6.10 occur often. We introduce anchor-based gateways as a solution to these problems.

ations, but constrictions do not always lead to nicely oriented gateways or stable gateways. This fact becomes magnified in coastal navigation scenarios, where no constrictions exist and gateways are simply placed at the edge of a core that scrolls with the robot (or artifact constrictions from the discretized skeleton create incorrect gateways).

The fundamental problem with using constrictions is that we are searching a 1D encoding of free space, with Voronoi radius annotations, for signals about the architecture instead of looking at the architecture itself. Below we present a method that looks at the relation between the skeleton and the local architecture. Our approach maintains the positive aspects of our constriction-based method (non-rectangular intersections are handled and no environment dependent metrics are needed), while also handling the problematic cases of constriction-based gateways.

Specifically, we motivate this new approach to handle coastal navigation scenarios, where walls exist on only one side of the path. So, we begin the discussion of the architecture-influenced, *anchor*-based gateways by defining a new skeleton that facilitates wall-following. We then detail a new gateway algorithm that allows us to define places at corners in coastal navigation scenarios, where no Voronoi junction exists, and the skeleton is a constant distance from nearby obstacles.

6.3.1 The Extended Voronoi Graph (EVG)

Although the Voronoi graph is well defined in corridor environments, the Voronoi graph no longer becomes a useful skeleton when the robot transitions into non-corridor environments. This is illustrated in Figure 6.12(a,b). Here, the robot moves into a room that is larger than the LPM size. At this location, there exists a large qualitative change in the environment, but the Voronoi graph simply ignores this fact.

Two related problems exist in this example. First, no place is defined. This is true of any skeleton-based approach to place detection. This place is ignored by junction-based approaches to place detection, and by our place detection scheme. The related problem is that of navigation control, which depends either on the actual skeleton for Voronoi-based navigation approaches or on target gateways in our control schemes (Section 5.4). The appropriate control in this scenario is coastal navigation, keeping at least one obstacle in sight at all times [Kuipers and Byun, 1991; Roy et al., 1999]. Instead of defining paths parallel to the walls, the Voronoi graph defines a path that will quickly bring the robot to a location where no visible obstacles exist. Once this occurs, the Voronoi graph can no longer be computed from local sensor data, and the robot is simply lost.

To handle coastal navigation scenarios, the robot needs a different skeleton than the traditional Voronoi graph—one that seamlessly transitions from a "midline" skeleton to a "wall-following" skeleton when necessary (Figure 6.12(c)). In the example, the natural and meaningful spatial configuration includes a place at the opening and gateways that inform the robot that it can perform travel actions by leaving the place through the bottom gateway or by leaving through the gateways to the left or right (Figure 6.12(d)). To obtain a graph that exhibits the appropriate behavior, we introduce an extension to the 2D version of the generalized Voronoi graph.

We define the *extended Voronoi graph (EVG)* as the subset of all points in the 2D GVG closer than a threshold M units from any obstacle, *combined with* the set of all points exactly M units away from the closest obstacle. More formally, given that the GVG in a planar model is the set of all points p equidistant from two or more closest obstacles O, we define the set of points P in the EVG as follows:

$$\begin{aligned}
P &= \{p : \exists o, o' \in O\,[o \neq o' \wedge dist(p,o) = dist(p,o') \wedge \\
&\qquad dist(p,o) \leq M \wedge \forall o'' \in O\,[dist(p,o'') \geq dist(p,o)]]\} \\
&\cup\ \{q : \exists o \in O\,[dist(q,o) = M \wedge \forall o' \in O\,[dist(q,o') \geq M]]\},
\end{aligned}$$

where M is a preset maximum threshold. Because LPMs limit the sensory horizon of the robot, and we want deterministic place detection regardless of how the robot enters a place, it is practical to make $M = side\_size(LPM)/c$, where $c \geq 4$: e.g., for the 300x300 cell LPMs

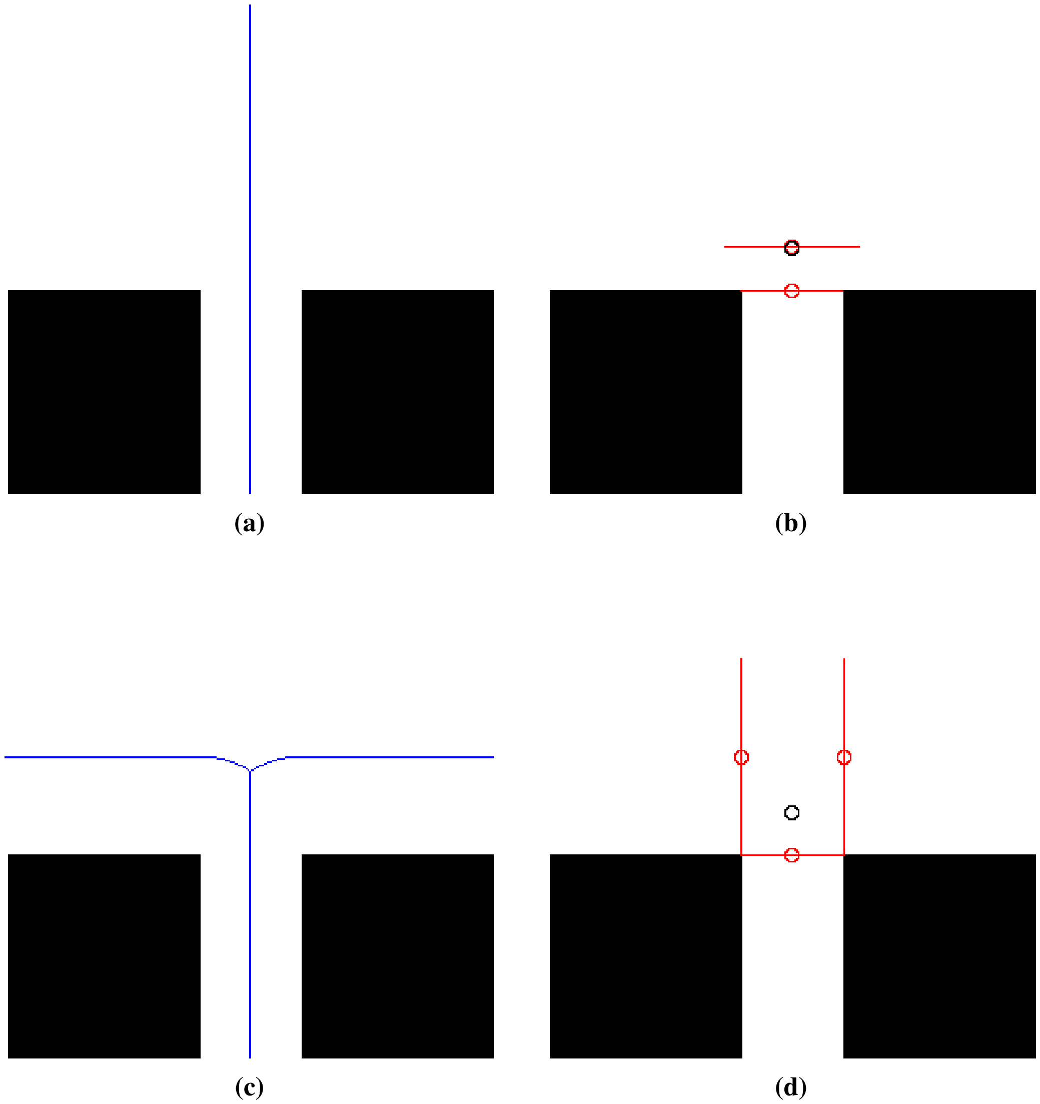


Figure 6.12: *Entering a room that is larger than the LPM.* Voronoi graphs are not useful in non-corridor environments due to the robot's limited sensory horizon in the LPM. **(a)** Here, the robot should begin to follow a wall, but the graph continues into unknown territory. **(b)** Gateways, which are defined using the skeleton, simply define a single, local-path, forcing the robot to continue traveling forward until all obstacles disappear from the LPM. At this point, the robot will be lost. **(c)** Here the extended Voronoi graph (EVG) defines a transition from corridor-following to wall-following, which is what we want. **(d)** Even constriction-based gateways work here, properly defining this as a T intersection.

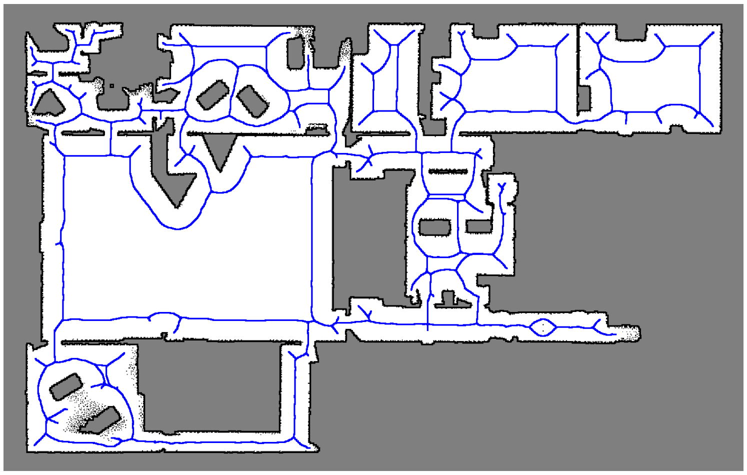


Figure 6.13: *The extended Voronoi graph (EVG) of a global metrical map.* Using an occupancy grid, the extended Voronoi graph can be drawn by using any occupied cells as obstacles. Given a reasonable maximum distance from wall, the graph now approximates "coastal navigation" routes in large rooms. Compare to Figure 2.3. Note: This image illustrates the idea of a coastal navigation graph. This is not an LPM, thus no pre-processing of the grid and no pruning of the skeleton are performed.

we use in this work, $c = 5$ and $M = 60$ cells.

Although this extension to the traditional Voronoi graph is simple, we were unable to determine a sub-quadratic algorithm for computing the continuous EVG. The continuous EVG of the same pre-computed global metrical map from Figure 2.3 is shown in Figure 6.13. For discretized brush-fire or thinning approximations, it is straightforward to keep track of the distance to the nearest obstacle when deciding whether to flip a cell and simply stop the "fire" or "wavefront" at that point. The linear time thinning approximation of the EVG, which we utilize in this work, is illustrated in Figure 6.14. Given this skeletal approximation of the EVG, the same LPM pruning discussed in Section 6.1.1 can be used to approximate the *reduced extended Voronoi graph (REVG)*.

6.3.2 Anchor-based Gateways

Given a minimal skeletal of free space that adequately handles coastal navigation scenarios, we need a gateway implementation that can overcome the problems of the constriction-

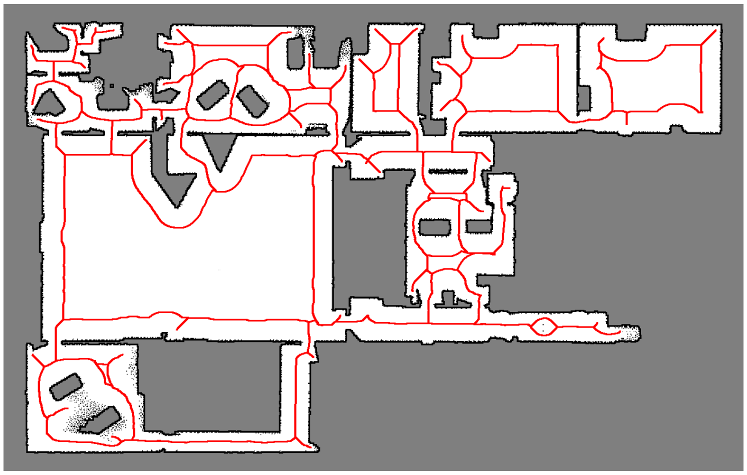
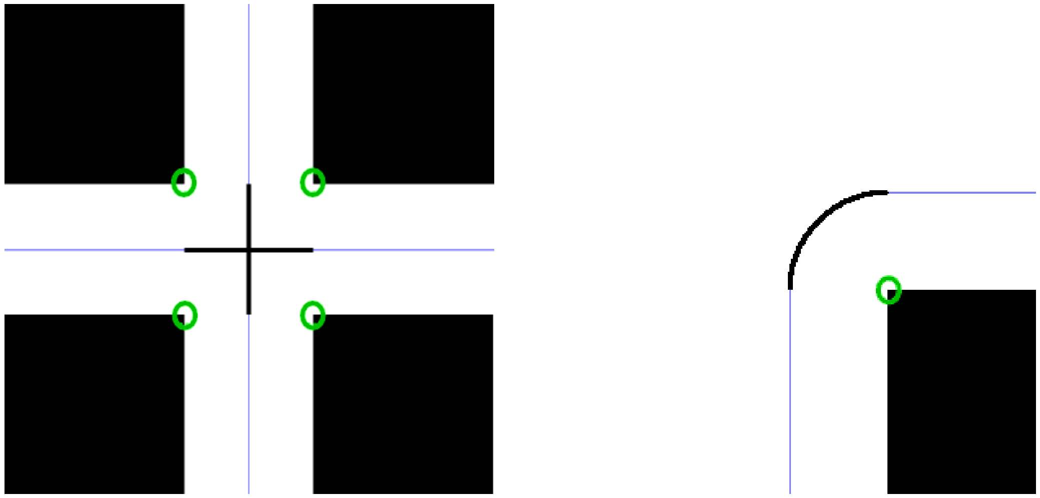


Figure 6.14: *The thinning approximation of the EVG.* The thinning algorithm can be extended to approximate the EVG skeleton in an efficient fashion. Compare to Figure 6.13.

based method to give stable, reliable gateways in all situations. Here we detail our anchor-based method which improves upon the state-of-the-art.

The intuition behind our anchor-based gateway approach is straightforward. In corridor environments, each skeletal point has at least two nearest obstacles (one on the left of the branch and one on the right) that define that point as belonging to the skeleton. At constrictions like doorways, these nearest obstacles lie at the gateway line-segment endpoints. Examining these constriction-defining obstacles, we notice that they not only determine the skeletal point at the constriction, but they also help define many other nearby skeletal points. This is illustrated in Figure 6.15(a). However, we see in Figure 6.15(b) that in coastal navigation scenarios there also exist points that contribute heavily to defining the skeleton. We term these points *anchor points*.

The idea of our anchor-based gateway method is to find these *anchor points* and use them to help define reasonable gateway locations. Algorithm 6.4 details the process for finding anchors from a pruned skeleton of free space and nearby obstacles. It defines search areas near the edge of the core in the same fashion as the constriction method. The obstacle that contributes to defining the most skeletal points in this search area is chosen at



(a) **(b)**

Figure 6.15: *Intuition behind skeletal anchors.* **(a)** Constrictions occur where the skeleton gets close to obstacles (Algorithm 6.2). The nearest obstacles to skeletal points at constrictions (circles in the image) are also the closest obstacles to many other skeletal points (the emphasized portions of the skeleton). These important obstacles will be used as *anchor points* to help determine gateway locations. **(b)** In coastal navigation scenarios, we also notice that there exist obstacles that define many skeletal points. These anchor points lie at locations where we expect gateway line segments to end.

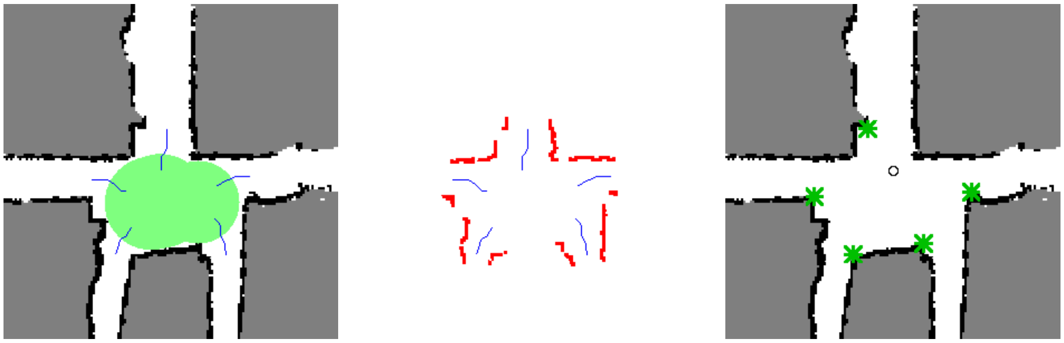
```
    $O$ is the set of obstacles in the LPM.
    $left\_of(s,o)$ determines whether an obstacle $o \in O$ is on the left side of the branch
        containing $s \in S_e$ at point $s$.
        (“Left” is determined calculating the orientation at $s$ at its neighboring point that is
        further from the edge of the core $e$.)
    $closest\_obs(s,o,O)$ determines whether $o$ is the closest obstacle to $s$ in $O$.
        $(o \in O \wedge \forall o' \in O\ [dist(s,o) \leq dist(s,o')])$

  Define the gateway search spaces $S_e$:
    (Use the same process from Algorithm 6.2.)

  Find the anchor for each search space $S_e$:
    Define the obstacles that are to the left of the points in the search area
        $O'_e = \{o' \in O : \exists s \in S_e\ [left\_of(s,o')]\}$.
    Define the obstacles that are to the right of the points in the search area
        $O''_e = \{o'' \in O : \exists s \in S_e\ [\neg left\_of(s,o'')]\}$.
    Count the number of times each obstacle $o' \in O'_e$ is the closest left obstacle to the
        points $s \in S_e$, and create a similar count for right obstacles $O''_e$.
    Determine the anchor $a_e \in \{O'_e \cup O''_e\}$ that is the obstacle with the highest count.
        (This obstacle is the closest left or right obstacle to the most skeletal points in the
        search area. If multiple obstacles are valid, use the one closest to the core edge $e$.)
    The set $A = \{a_e\}$ defines the anchors of the LPM.
```

Algorithm 6.4: *Finding anchors of an LPM.* We assume the robot already has computed base points and a core of the LPM—see Algorithm 6.1, which applies even to the extended Voronoi graph.

the anchor for that branch. An illustration of this search process is given in Figure 6.16.

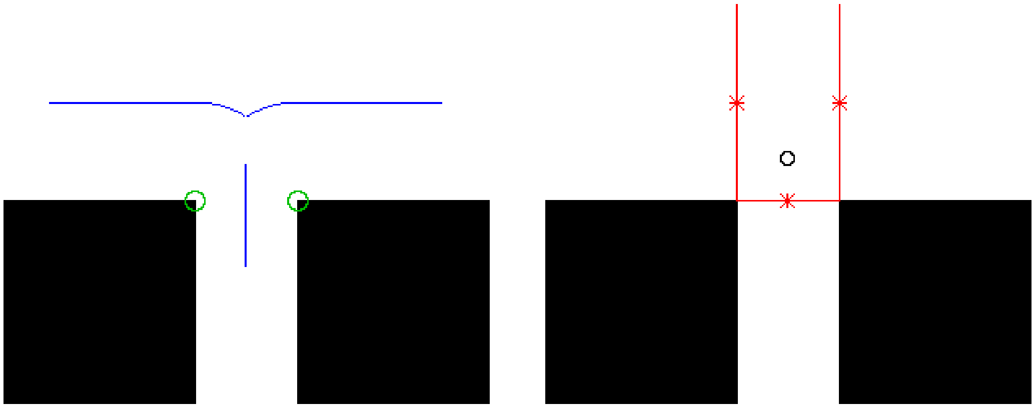
Once anchors are found, the next step is to determine reasonable gateway location hypotheses. In the constriction-based algorithm, the closest obstacles to constrictions were used to create a line segment representation of a gateway. A naïve approach might be to try something similar with anchors, by looking for the closest skeletal point. However, in cases where convex corners exist, multiple skeletal points are equidistant from an anchor. Similarly, multiple branches may share an anchor point, so one anchor point might determine multiple gateways (Figure 6.17).

Algorithm 6.5 details how we propose gateway location hypotheses on the skeleton for four cases. These cases are illustrated by Figure 6.18. Basically, if the anchor point is the closest “left” or closest “right” obstacle to the point at the edge of the core along the



(a) **(b)** **(c)**

Figure 6.16: *Searching for anchors.* **(a)** The first step in finding anchors is to determine the core of the LPM from the pruned skeleton. From the core, search areas are created that consist of points that lie within the Voronoi radii of the points at the edge of the core (points separated from the core edge by junctions or the closest point to the robot are excluded). **(b)** Next, the algorithm determines the closest obstacles to the skeletal points in the search areas. **(c)** For each search area, the obstacle involved in defining the largest number of points is found. That is, the anchor.



(a) **(b)**

Figure 6.17: *Anchors and gateways in coastal navigation scenarios.* **(a)** Anchors where a wall-following path intersects a corridor-following path. Each of the three search areas have an anchor. Sometimes the same anchor point is found for multiple branches. **(b)** From two anchor points, we get three distinct gateways. This illustrates that a simple rule cannot be used to find the location of the gateways with respect to the anchors.

branch, the algorithm looks for skeletal points in the search area where the anchor is *not* the closest left or right obstacle. Skeletal points outside the core are examined only if none exist inside the core that meet the above criterion. Of these points the closest to the edge is used as the gateway hypothesis. However, if the anchor point is *not* the closest left or right obstacle to the edge point, then the algorithm looks for a skeletal point in the search area where this *is* the case. Again points in the interior of the core are preferred, and of these, the one closest to the edge is chosen.

Once gateway location hypotheses are determined, the same process from Algorithm 6.3 may be used to find the line segment that represents the gateway. This is only true for cases where two obstacles define the skeleton at the gateway hypotheses. In coastal navigation scenarios (where the EVG creates skeleton points based on only one nearby obstacle), the gateway line segment is simply defined at the hypothesized location, with an orientation determined by the line that connects the gateway location point with the anchor.

Discussion

This anchor-based gateway algorithm works as intended. The algorithm is simple to implement, and does not add any noticeable compute time to the real-time robot navigation: local topology can still be updated at approximately 2 Hz for a 300x300 cell occupancy grid. In fact, finding the skeleton (including pre-/post-processing) is the main bottleneck in the local topology computation.

Examples of gateways using our implementation are shown in Figure 6.19 and can be compared to the constriction-based gateways of Figure 6.8. Comparing examples of constriction-based gateway with anchor-based gateways, we notice that anchor-based gateways of corridor intersection are virtually identical to the gateways we obtain with constriction-based gateways. The cases we used to show the constriction problems in Figures 6.9, 6.10, and 6.11 are shown to be solved by the anchoring solution in Figures 6.20 and 6.21.

```
Find a gateway location hypothesis for each search space $S_e$ and anchor $a_e$:
(This assumes functions and sets defined in Algorithms 6.1 and 6.4.)
  IF $closest\_obs(e, a_e, O'_e) \vee closest\_obs(e, a_e, O''_e)$
    ($a_e$ is either the closest left obstacle or the closest right obstacle to edge point $e$.)
  THEN
      Find the set of search points inside the core whose closest left obstacle and
        closest right obstacle is not the anchor
        $I_e = \{i \in S_e : [\neg closest\_obs(i, a_e, O'_e) \wedge \neg closest\_obs(i, a_e, O''_e)] \wedge \forall f \in F\ [skel\_dist(i, f) < skel\_dist(e, f)]$.
      IF $I_e \equiv \emptyset$ THEN
          Find the set of search points outside the core whose closest left obstacle and
            closest right obstacle is not the anchor
            $I_e = \{i \in S_e : [\neg closest\_obs(i, a_e, O'_e) \wedge \neg closest\_obs(i, a_e, O''_e)] \wedge \forall f \in F\ [skel\_dist(i, f) > skel\_dist(e, f)]$.
      IF $I_e \equiv \emptyset$ THEN
          $I_e = \{i \in S_e : \forall s \in S_e\ [skel\_dist(i, e) \leq skel\_dist(s, e)]\}$.
            (Choose the point in $S_e$ closest to the core edge.)
  ELSE
  ($a_e$ is not the closest left obstacle and not the closest right obstacle to $e$.)
      Find the set of search points inside the core whose closest left obstacle or closest
        right obstacle is the anchor.
      IF $I_e \equiv \emptyset$ THEN
          Find the set of search points outside the core whose closest left obstacle or
            closest right obstacle is the anchor.
  $h_e \in I_e$ is the gateway location hypothesis, where
    $\forall i \in I_e\ [skel\_dist(e, h_e) \leq skel\_dist(e, i)]$.
    (Use closest point in $I_e$ to edge $e$.)
  The set $H = \{h_e\}$ hypothesizes the locations of the gateways.

Calculate the gateway line segment for each gateway location $h_e$:
  IF $\forall o' \in O'\ [dist(h_e, o') > r_{h_e}] \vee \forall o'' \in O''\ [dist(h_e, o'') > r_{h_e}]$
    (coastal navigation scenario)
  THEN
      Generate a line segment $g_e$, centered at $h_e$, of size $2 \cdot r_{h_e}$ (twice the Voronoi
        radius). The orientation of the segment is the same as the orientation between
        $a_e$ and $h_e$.
      The set $G = \{g_e\}$ represents the gateways of the LPM.
  ELSE
      Calculate gateway line segment $g_e$ from $h_e$ using the same process as the
        constriction-based algorithm (Algorithm 6.3).
```

Algorithm 6.5: *Finding anchor-based gateways.* We assume the robot has already determined the search areas and anchors near the edges of the LPM core—see Algorithm 6.4. Once hypothesis gateway locations are determined, gateway line segments are computed in the same fashion as in Algorithm 6.3 for non-coastal-navigation situations.

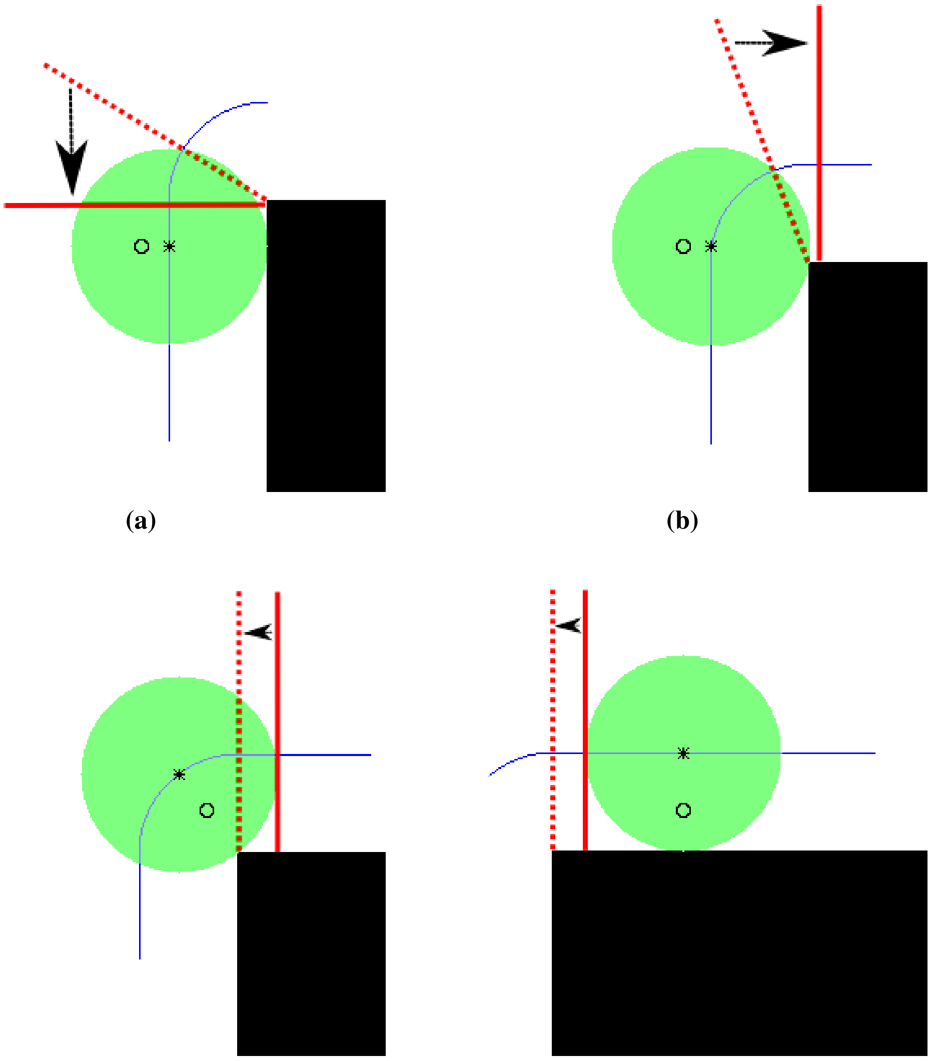
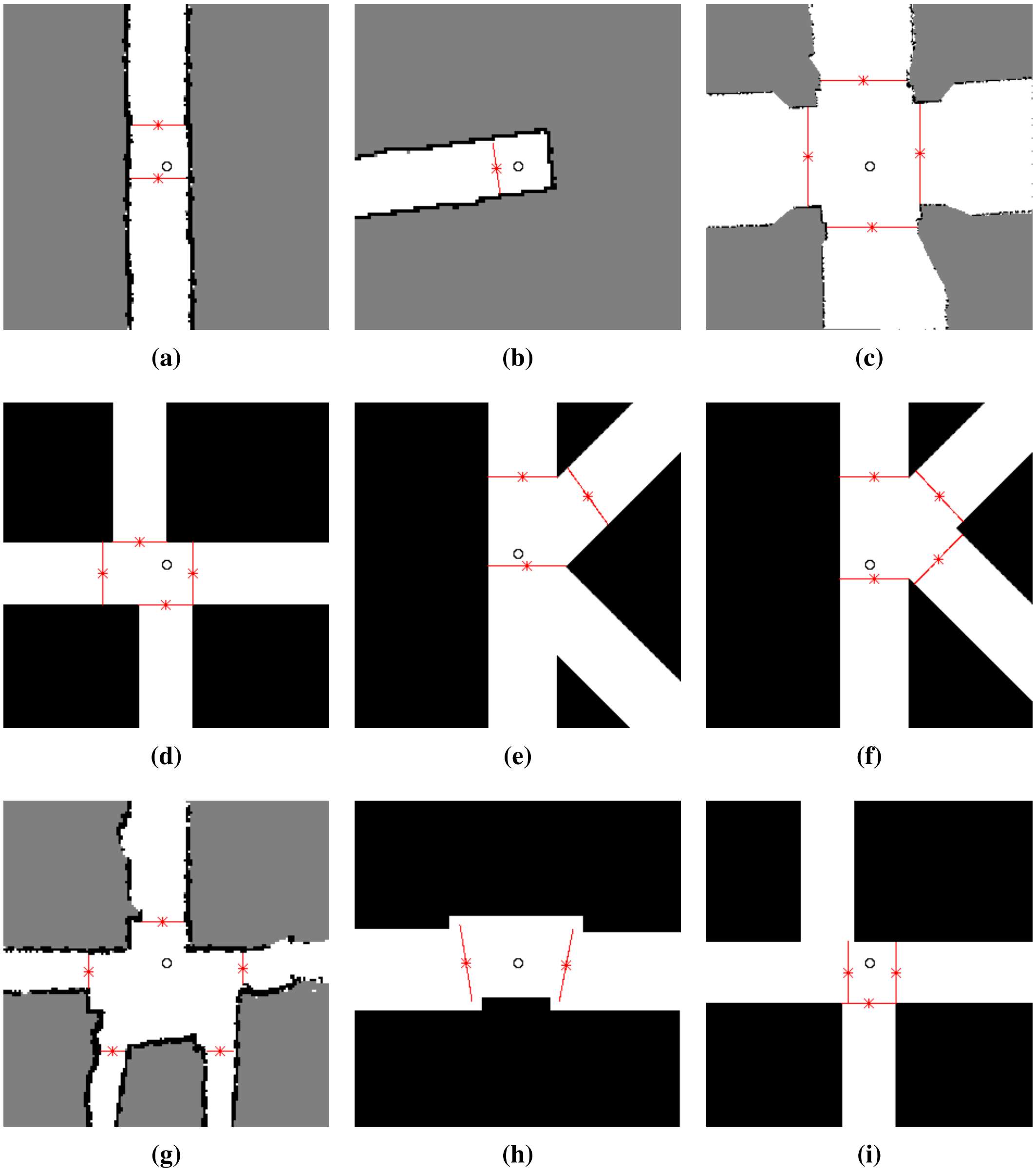


Figure 6.18: *Finding gateway location hypotheses from anchors.* **(a)** If the anchor *is* the closest obstacle to the core edge point, crawl the graph from the edge *into* the core looking for a skeletal point where this is *not* true. **(b)** If no points are found in (a), crawl the graph from edge *away from* the core looking for points where this is not true. **(c)** Sometimes, the anchor is not the closest obstacle to the edge of the core. In this case, crawl the graph into the core looking for a point that *does* have the anchor as its closest point. **(d)** Otherwise, crawl the graph away from the core, looking for a point where the anchor is becomes the closest obstacle.

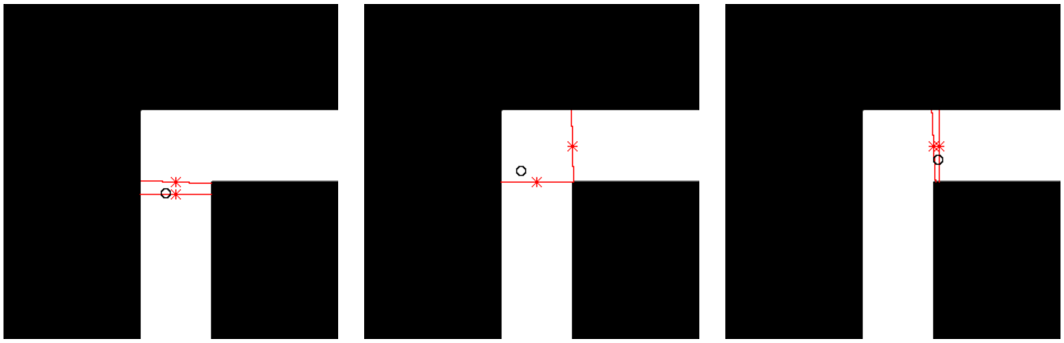


(a) **(b)** **(c)**

(d) **(e)** **(f)**

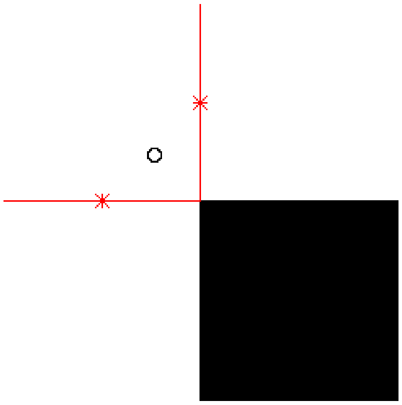
(g) **(h)** **(i)**

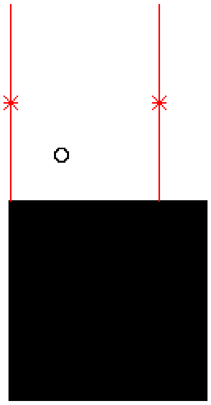
Figure 6.19: *Examples of anchor-based gateways in a variety of (real-world and pathological) environments.* Compare (a-f) to Figure 6.8. Compare (g-i) to Figures 6.6(b), 6.7(c), and 6.7(d) respectively.



(a) **(b)** **(c)**

Figure 6.20: *Stable anchor-based gateways near* L *intersections.* Compare with Figures 6.9 and 6.10.



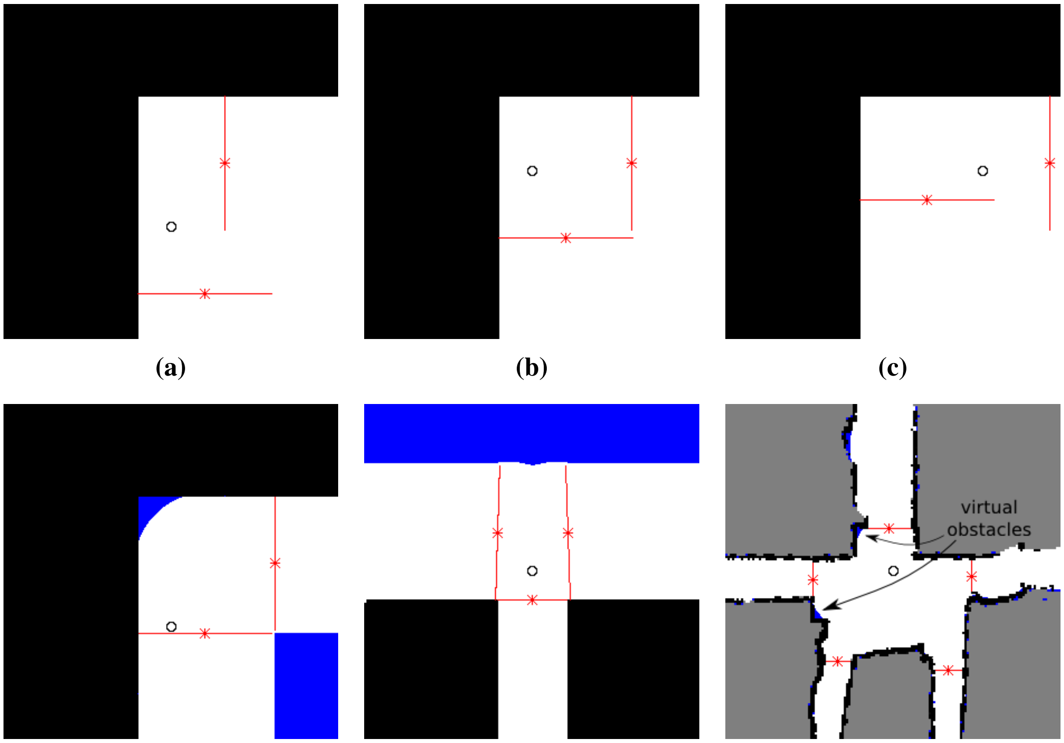


(a) **(b)**

Figure 6.21: *Stable anchor-based gateways at coastal* L *intersections.* Compare to Figure 6.11.

Though the anchor-based solution solves the gateway stability problem illustrated in Figure 6.10 for L intersections where a convex corner exists, we have observed that this stability problem is still an issue for coastal navigation at *concave* corners (Figure 6.22(a,b,c)). At noise-free concave corners, there is no obstacle to provide an anchor, so the gateways are placed at the edge of the core by default, again causing the gateways of a place to become dependent on the robot's location in the place. A simple solution is to change any free cells that are beyond the Voronoi radius of all pruned EVG skeletal points into occupied cells. This is done after creating the pruned skeleton, but before looking for anchors. This creates virtual obstacles on all side of the skeletal branches, and provides anchors in situations where they are needed (Figure 6.22(d)). Performing this operation on every LPM should work fine in theory (Figure 6.22(e,f)); however this operation is worst-case $O(n^2)$ where n is the number of cells in the LPM (slower than the skeleton and the gateway algorithm combined). Currently the addition of this small amount of code swells local topology compute time in the full HSSH mapping suite from $\sim$2 Hz to $\sim$0.5 Hz in coastal navigation scenarios using a 300x300 cell LPM. Finding an efficient solution to this problem should be a reasonable short-term goal for future work.

Overall, the anchor-based gateway implementation works as intended. It generates gateways similar to the constriction-based method, while overcoming the unstable issues discussed in Section 6.2.2. This algorithm is reasonably independent of pose—at intersections with skeletal junctions, gateways do not change while the robot is within the Voronoi radii of the base points, and at intersections without junctions, gateways are stable due to using anchors. This algorithm is also independent of scale—though the robot's radius and the LPM size will determine the location of gateways, thus the location and description of places, the algorithm does not expect any particular sized paths or doorways. Additionally, we have shown examples where this algorithm produces stable gateways in non-rectangular intersections and coastal navigation situations.



(a) **(b)** **(c)**

(d) **(e)** **(f)**

Figure 6.22: *Handling environments without good anchors.* Concave corners are examples of places where good anchors or constrictions may not exist. **(a,b,c)** Here, the robot's gateways are unstable because they are placed at the core edges, due to a lack of useful information. **(d)** By changing all free cells that are outside the Voronoi radii of *all* skeletal points in the pruned EVG into occupied cells, we can create virtual obstacles that provide anchors. Compare to image (a). **(e,f)** This does not unduly affect gateways in other coastal navigation situations or non-coastal, corridor environments.

6.3.3 Evaluating Anchor-based Gateways

The anchor-based gateway detection and subsequent local topology abstraction have proven to be quite reliable in our experience. Chapter 7 details a human-robot interaction experiment that uses this anchor-based gateway implementation to detect and describe places with a simulated robot. The algorithm's long-term reliability can be inferred from error-free performance in this experiment. In the more than 360 places encountered and the hundreds of thousands of LPMs examined during this experiment, there were no places missed or incorrectly described in terms of local topology. Additionally, we had no false positive place detections despite randomly placed stationary obstacles in the environment.

We have also experienced high-quality place detection and local topology abstraction on real-world robots that explore dynamic indoor environments using noisy sensors; however no long-term quantitative data has been gathered. Samples of LPMs and local topologies generated by our Magellan Pro mobile robot (Figure 2.1(b)), equipped with a SICK LMS lidar, are provided in Figures 6.23-6.26.

Figure 6.23 illustrates the quality of the gateway algorithm at places in a university building. Gateways are reliable and stable at places despite multiple visits, entering places from different directions, and having various LPM rotations with respect to the path directions. We observe deterministic place detection at intersections in university hallways and near-deterministic local topology abstraction. Figure 6.23(h,i) illustrates an example of local topology ambiguity, due to a boundary case of the simple ray-casting gateway alignment operation (Section 5.2.2). Here a small shift in the LPM affects the skeleton enough to change the detected local topology change from that of a T intersection to that of a Y intersection.

Figure 6.24 shows the performance along paths. The algorithm handles hallways of various sizes (Figure 6.24(a,b)) and even handles paths that contain "non-island" obstacles, as seen in Figure 6.24(c-f). Figure 6.24(g,h) illustrates how the robot stops at a potential place, but, after more deliberate examination of the local region, it determines that it is

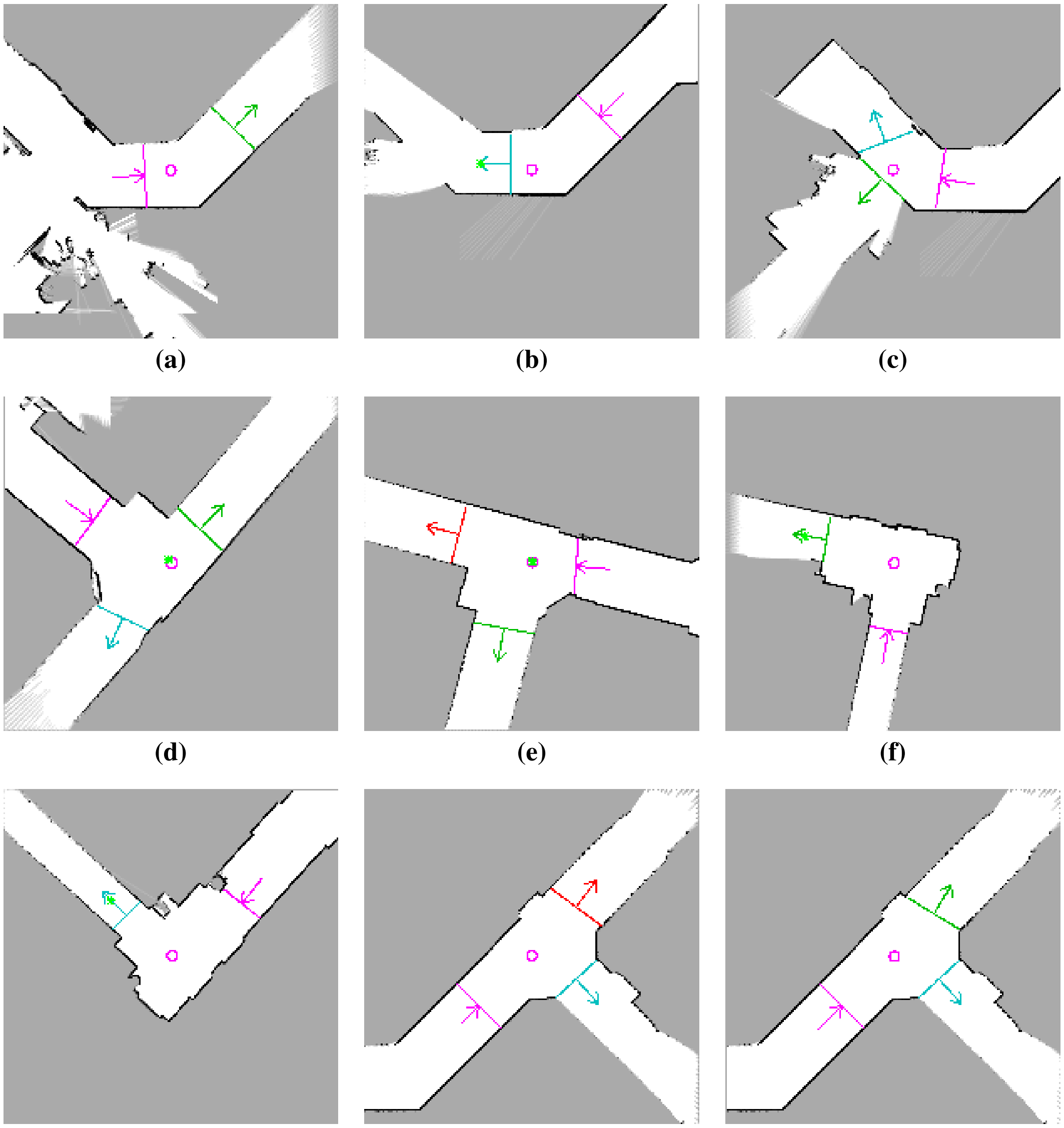
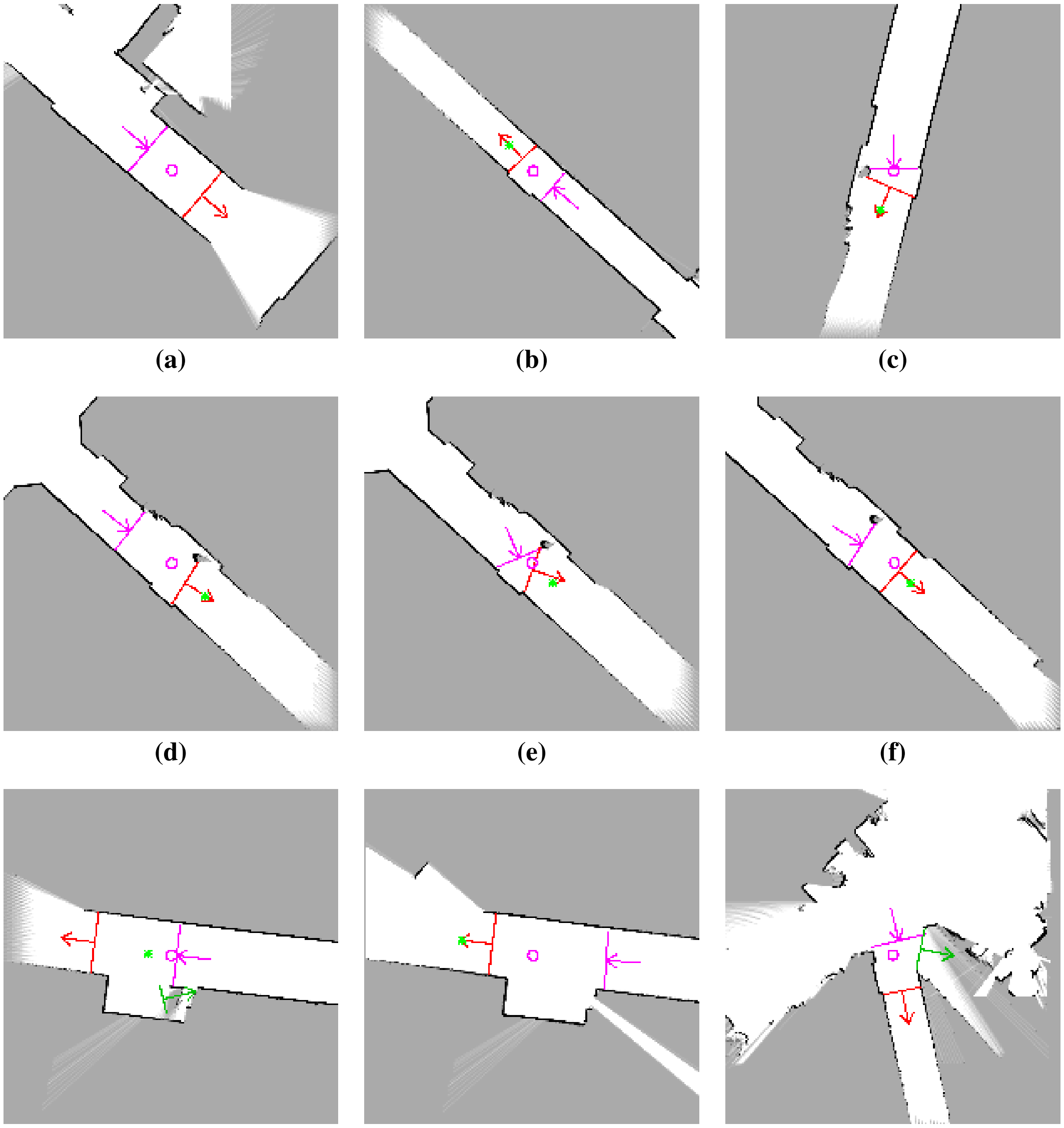
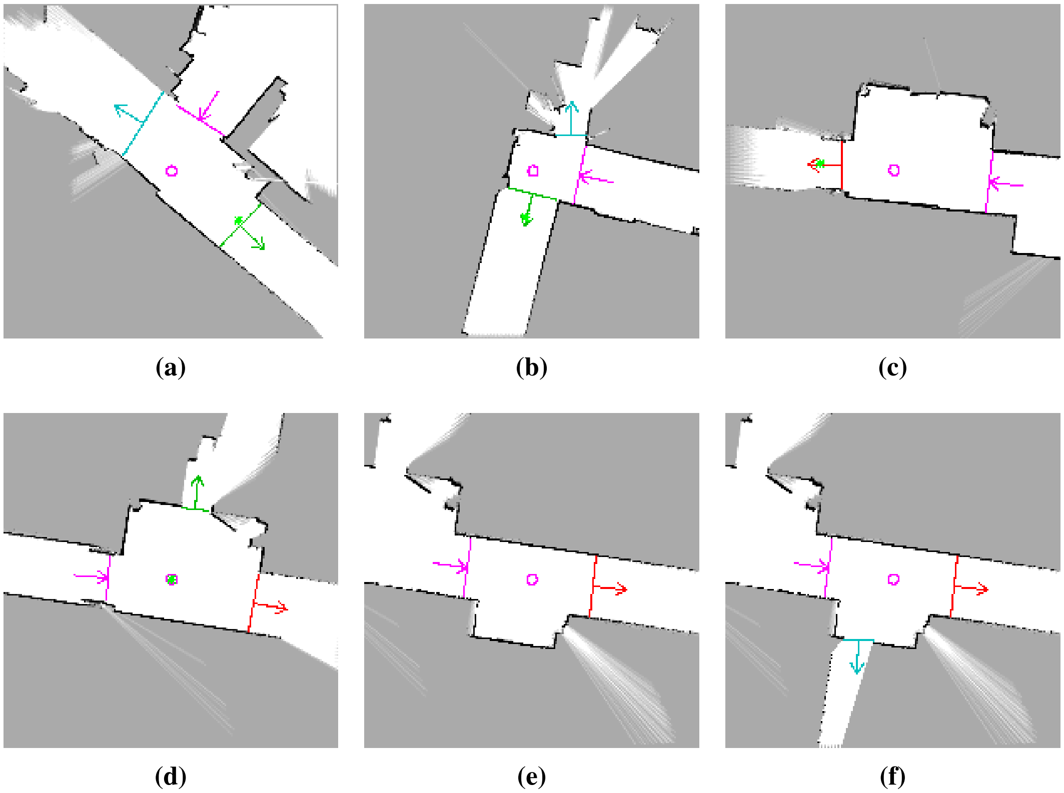

(a) (b) (c)
(d) (e) (f)
(g) (h) (i)

Figure 6.23: *Gateway (and place) detection in a university hallway.* **(a,b)** This 45° bend in the hallway is seen as an L intersection. This place is detected by the robot when traveling in either direction. **(c)** At this place, the Voronoi skeleton causes the rightmost gateway to be slightly angled, creating a Y local topology. This is actually a reasonable assumption, especially since most people navigating this environment actually turn left here. **(d,e) & (f,g)** We illustrate the stability of gateways for the same intersection visited on separate occasions (and entered from different directions). **(h,i)** Due to an almost imperceptible shift in the LPM, the topmost gateway in these two images changes from being aligned with the lower gateway (image (h)) to being "left of" the lower gateway (image (i)). This demonstrates that boundary cases in our simple gateway alignment criterion can affect local topology abstraction.



(a) (b) (c) (d) (e) (f) (g) (h) (i)

Figure 6.24: *Traversing paths in a university hallway.* **(a,b)** Our scale independent gateway algorithm allows the robot to handle various sized paths. **(c)** Even when encountering “non-island” obstacles, the anchor-based gateway algorithm produces aligned gateways that represent a single path. **(d-f)** Here we see the same location as in image (c), but with the robot traveling in a different direction. **(g,h)** Our robot can only see in front of it, so unmapped portions of the LPM may create *exits* in the skeleton, which cause gateways. By examining potential places, the robot can remove incorrect gateways to avoid false positive place detections. (Note: the lidar sees “through” the wall a bit in image (h) because the metal legs of a chair reflect the instrument’s laser beams.) **(i)** Here the rightmost gateway should not exist—smooth rubber tires slipping on the linoleum floor creates a localization failure in the LPM that moves the hallway from its initial position.



(a) **(b)** **(c)**

(d) **(e)** **(f)**

Figure 6.25: *Detecting doorways in a university hallway.* **(a)** Here the robot detects a place in front of some double doors (the doorway is the middle gateway). Actually, the doors here are closed; however, the robot saw them as open when it passed through them. Because the robot can remove obstacles that reside in cells that it has previously seen as free, the doors are ignored by the gateway detection algorithm. **(b)** A single open door to an office at an L intersection. **(c,d)** When passing this alcove the first time, the robot did not consider it a place. The next time, someone had opened the door, so the robot decided this is a place. **(e,f)** This is another scenario where the robot is traveling past a door without stopping. Here, we stop the robot, and open the door. Upon seeing this, the robot immediately decides that this is a place. (Notice the metallic chair legs cause the robot to again assume free space beyond the wall—see Figure 6.24(h). The robot still sees the wall behind the chair legs—notice the obstacles defining the wall still exist near the perceptual mistake—so no gateway is detected at this spot.)

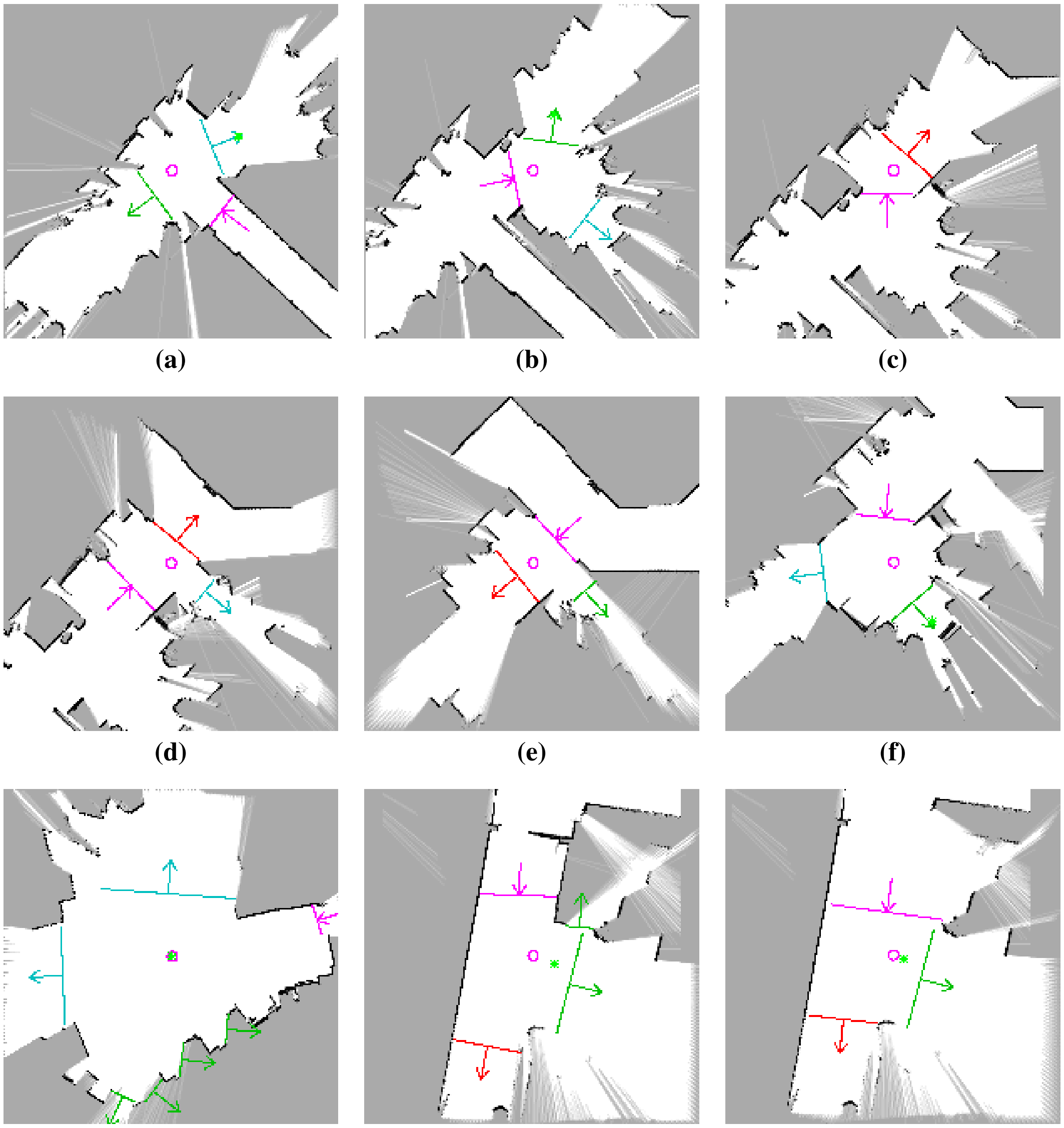

(a) (b) (c) (d) (e) (f) (g) (h) (i)

Figure 6.26: *Dealing with furniture.* **(a-d)** In this computer lab, the furniture is mostly large desks that divide the lab into distinct regions. The robot parses this space fairly well: detecting a T decision, then a Y, then a small path in image (c), followed by another T. **(e,f)** When traveling in the other direction, the robot detects the same places (compare images (d) and (e) and also images (b) and (f)). **(g)** Desks and chairs often create occlusions that produce gateways. These false gateways are often hard to eliminate because chairs prevent physical exploration under the desks. Such exploration is also time intensive, so it is not desired. We expect visual object recognition to help overcome this issue. **(h,i)** Furniture with shiny surfaces can lead to false free space in the LPM (the top of image (h) corresponds to the bottom of Figure 6.24(h)). If real free space intersects this false free space, important pieces of architecture can be seen as “islands”, which affects gateways.

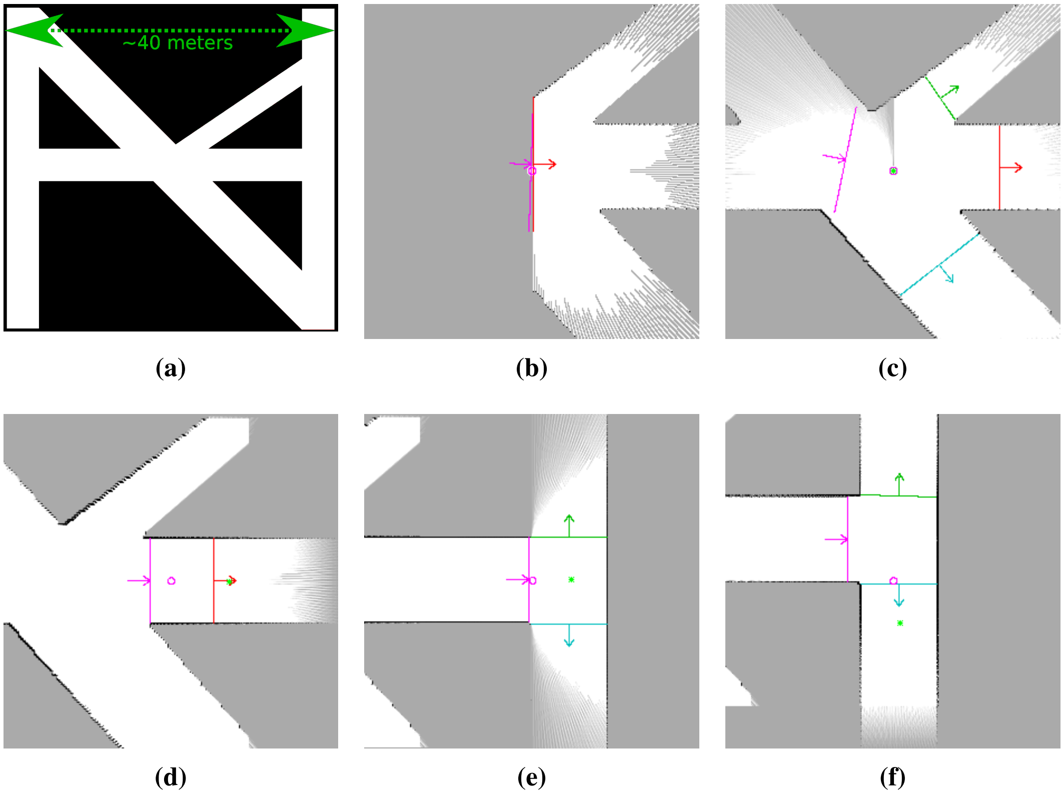
still on a single path. Figure 6.24(i) shows how a simple localization failure in the LPM at the Local Metrical Level can be problematic at the HSSH Local Symbolic Level. A reasonable piece of future work would be detecting and recovering from these localization failure scenarios.

Figure 6.25 shows how the gateway algorithm handles doors. Figure 6.25(c-f) illustrates how regions that are seen as paths when doors are closed are seen as places when doors are opened. Not being able to detect closed doors with range sensors is a problem that affects almost all modern-day indoor mobile robots.

Finally, Figure 6.26 shows how the algorithm behaves in rooms with lots of furniture. Large furniture cannot be distinguished from architecture. Sometimes this does not actually present a problem, like in Figure 6.26(a-f). However, many times furniture creates "frontiers" in the LPM, where multiple false gateways are proposed (Figure 6.26(g)). Additionally, office furniture that has metallic legs or slick, black surfaces will reflect the laser beams of the lidar device, creating false free space in the map. Once obstacles are surrounded by free space, regardless of whether the LPM is correct or not, they are removed for the purposes of computing gateways by the "island" removal pre-processing. Figure 6.26(h,i) shows a bad case where a wall is seen as an "island" obstacle because reflected lidar measurements from metallic chair legs corrupt the LPM.

Because it is difficult to find non-rectangular or complex intersections in university hallways, we also include snapshots from a simulated robot exploring an environment with more complex places in Figures 6.27 and 6.28. Here we can verify the performance of local topology abstraction in adjacent, non-rectangular places and places at sharp turns. In all cases, we obtain deterministic, correct local topology abstraction.

Though we believe the figures in this chapter are compelling examples of our gateway algorithm, we want to thoroughly evaluate the anchor-based gateway algorithm to ensure it is robust to various conditions that we suspect could be problematic. Below, we evaluate the robustness of the anchor-based gateway implementation under certain noise



(a) **(b)** **(c)**

(d) **(e)** **(f)**

Figure 6.27: *Gateway (and place) detection using a simulated robot.* **(a)** We created a simulated environment that approximates a 40 meter wide environment with complex and non-rectangular intersections. We illustrate gateway detection and local topology abstraction using simulated robot with low-noise actions and perception. **(b)** When starting, robots with only forward-facing observations will create gateways that may not be correct (here a single path). **(c)** Once the robot fully observes its local surround, it abstracts the correct local topology. **(d)** The robot chooses to go straight, and moves out of the place onto a path. **(e)** When the robot arrives at the next place, it gets the local topology correct; however, the LPM is not fully explored. (The light asterisk in some images denotes the local motion target of the robot in the LPM). **(f)** Only after filling the observations for the local region does the robot decide it is actually at a place. The robot's exploration is continues in Figure 6.28.

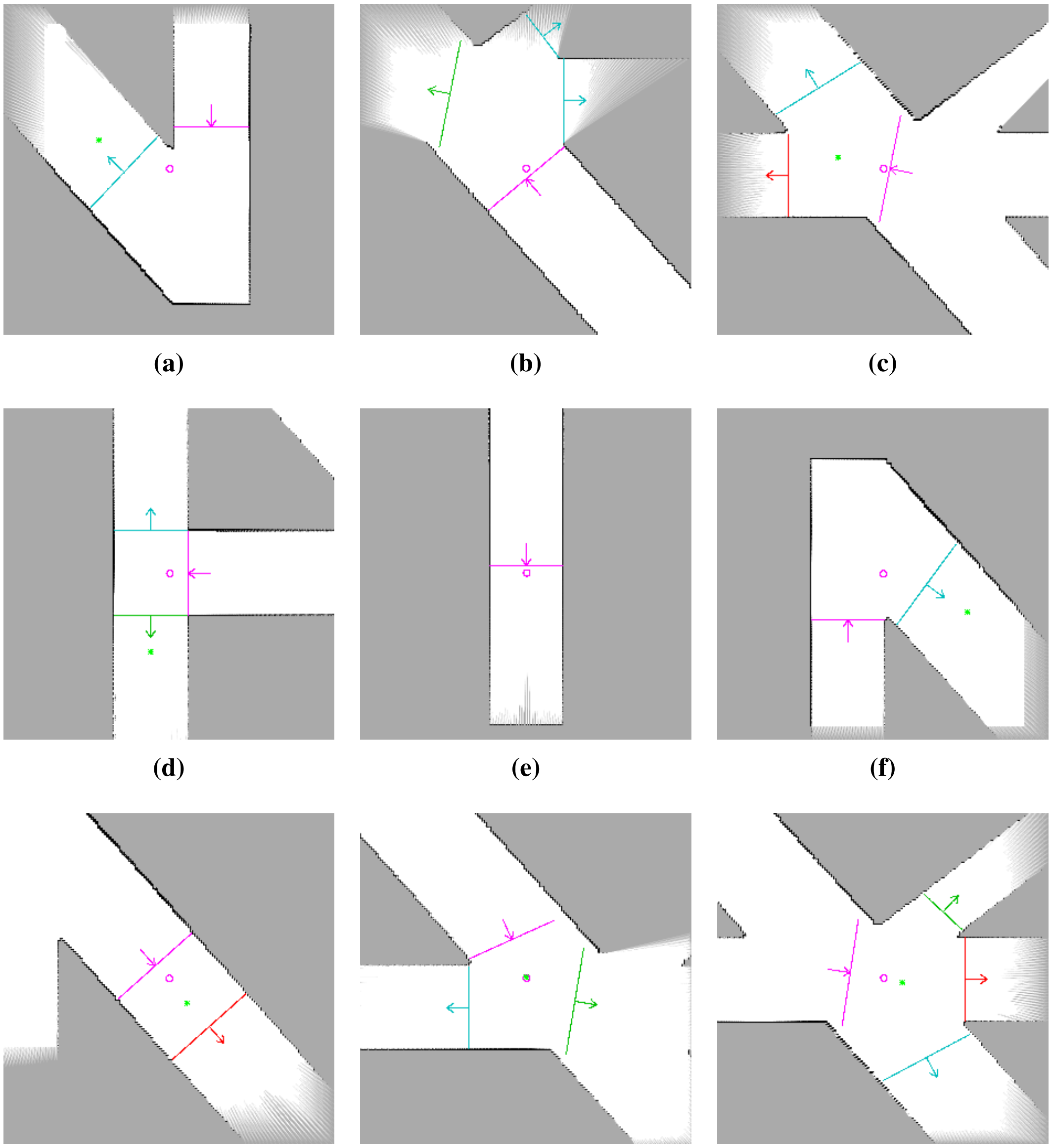

(a) **(b)** **(c)** **(d)** **(e)** **(f)** **(g)** **(h)** **(i)**

Figure 6.28: *Gateway (and place) detection using a simulated robot.* **(a)** The robot turns right in the previous place and arrives at the non-rectangular intersection in the bottom right of Figure 6.27(a). **(b)** Arriving at the first place again, the same local topology is found. **(c)** The complex middle portion of image (a) is considered two places by our algorithm. **(d)** The robot travels left, to the other T intersection. **(e)** It then travels down to the dead-end. **(f,g,h,i)** The robot makes it way up to the top left of the environment before returning to the middle. It detects all non-rectangular places deterministically and correctly, despite how it enters each place.

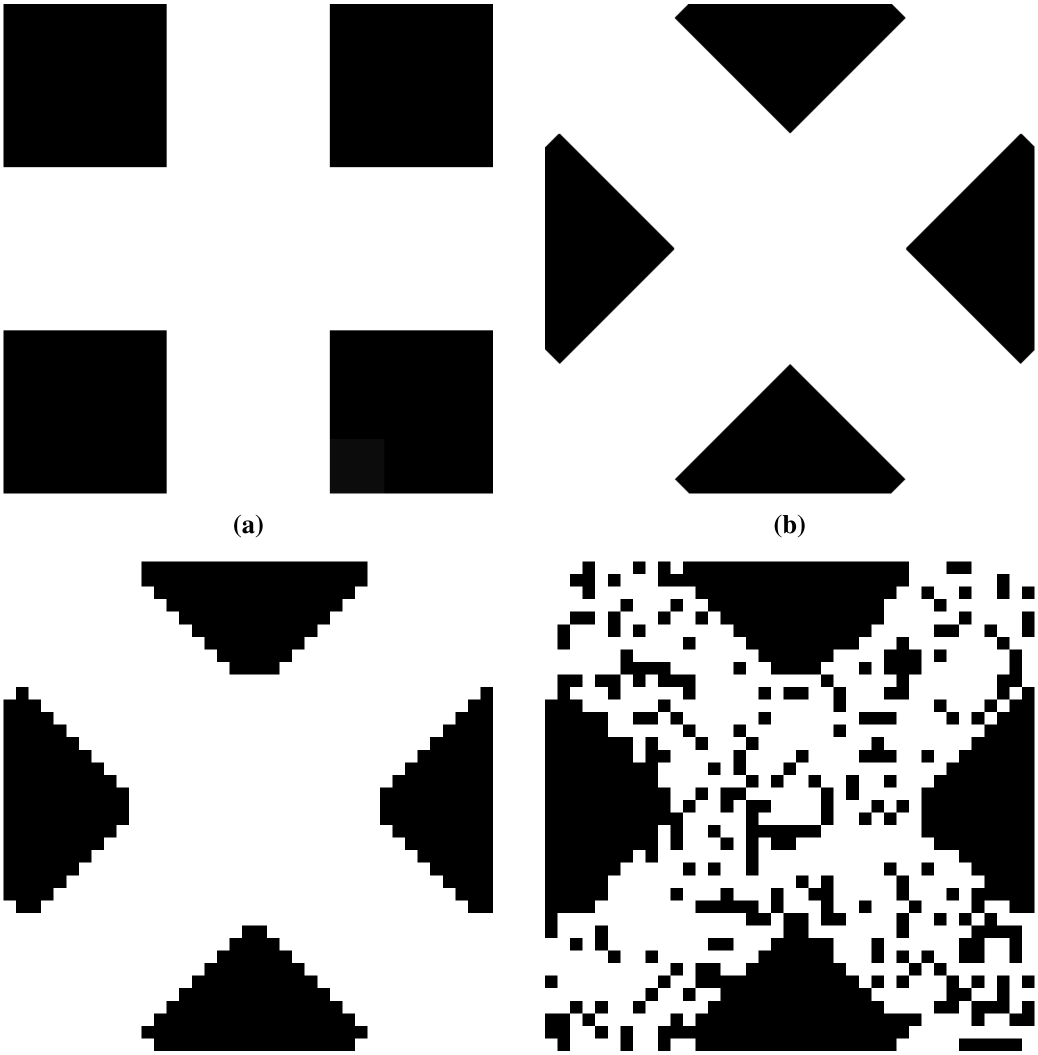
conditions by observing the performance of the local topology abstraction (which uses the anchor-based gateway implementation and the simple gateway alignment criterion from Section 5.2.2) as we alter a simple LPM in various ways. Specifically, we investigate the issues of LPM rotations, LPM resolution/size, and uniformly distributed noise in order to evaluate the sensitivity of the gateway algorithm to conditions known to affect the shape of free space skeletons.

Evaluation Setup

To evaluate various properties of our anchor-based gateway algorithm, we created a corpus of images that provide reasonable approximations of common LPMs. We used a high-resolution image of a $+$ intersection (Figure 6.29(a)) and created a corpus of 12000 images from transformations of this initial image. We chose to use a $+$ intersection for several reasons: (1) these intersections are extremely common (consider road networks); (2) a $+$ is a sort of superset of both L and T intersections; (3) due to the symmetry, we only need to rotate the image up to 45° to exhaust the rotational search space.

To generate test images, we first rotated the image from 0° to 45° in 5° increments, creating 10 possible rotations (Figure 6.29(b)). As noted several times in this dissertation, thinning skeletons are rotation dependent. By examining how rotation affects the accuracy of local topology abstraction, we can determine whether using efficient, thinned skeletons is reasonable or whether we should be utilizing continuous Voronoi diagrams that are rotation independent.

Next, we scaled the rotated images from high-resolution images to 60 different resolutions from 9x9 cells to 599x599 cells in 10 cell (per side) increments (Figure 6.29(c)). Because the gateway detection process is linear in the number of cells, doubling LPM sizes (in the number of cells per side) increases the time of the local topology abstraction quadratically. For many robotic platforms, a minimum-sized area needs to be modeled in order to facilitate good localization and control; however, often small mobile robot computers do



(a) (b) (c) (d)

Figure 6.29: *Gateway evaluation conditions.* **(a)** We start with a simple, high-resolution + intersection. **(b)** We rotate the image from 0 to 45° in 5° increments. This image is rotated by 45°. **(c)** We scale the image to be between 9x9 and 599x599 cells in 10 cell (per side) increments. This image is 39x39 cells. **(d)** We add random noise by probabilistically flipping a free cell to occupied. This image shows a 25% noise probability.

not posses the speed required to process large grids in real-time. By halving the resolution of cells (each cell represents a larger area of the local space) the robot can model the same sized region using quadratically fewer cells, decreasing the run time of LPM-based algorithms. However, decreasing resolution can overestimate the size of small objects and create noisy obstacle boundaries (e.g., see the 0.1 meter cells in Figure 6.8(a) as opposed to the 0.05 meter cells in Figure 6.8(c)). Because the region modeled by the images is a constant, reducing the image size is akin to decreasing the resolution of LPM cells; therefore we can examine how our local topology abstraction works in LPMs with a small number of cells and low resolution.

Rotations can create gray cells by averaging over white and black cells. Likewise, scaling can create gray cells from black and white cells due to anti-aliasing. If we classify gray cells as unknown, we can potentially get false *exits*, which will cause unwanted false gate gateways. To avoid this problem, we classify all cells as either free or occupied, using a gray-scale value of 127 as the separating threshold. Additionally, as we scale the image, the robot's size can affect the size and location of places it will consider. To avoid this problem, we set the robot's radius to zero.

Finally, we add noise to the images by flipping each free cell to an occupied cell with some probability (Figure 6.29(d)). This probability ranges from 0 to 95 in 5% increments. Several factors exist that can create noise in a real-world LPM. First is the accuracy and precision of localization. We discussed how to handle this using our shrink-and-grow localization scheme in Chapter 4. Next is the noise from the sensors. Modern-day planar lidar devices have only ± 1 cm of error with an 80 meter range; thus many SLAM implementations simply do not consider sensor noise when building a metrical map with cells >2 cm. Finally, the environment itself can have posts, garbage cans, table legs, or pedestrians that are essentially noise when trying to extract the structure of the nearby paths and places.

Though adding uniformly distributed noise over free space is not the most accurate model of typical lidar-based LPM noise, we feel it is a reasonable starting point for this evaluation. Additionally, we will show that most of this noise is actually filtered out by the "island" obstacle removal process prior to generating the skeleton. As a result, increasing the noise essentially increases the noise along the path boundaries. This may be a reasonable model for vision-based LPMs, as outdoor path extraction (e.g., sidewalk or streets) may have shadows, mud, tall grass, etc. that create noisy path boundaries. It is certainly a reasonable model of the hallways near our robot lab, which often have physics lab equipment along the walls.

Evaluation Results

Transforming the original image over rotation, resolution, and noise dimensions we obtain $(10*60*20)$ 12000 test images that we use to evaluate the performance of the anchor-based gateway extraction. Rather than considering 4 detected gateways as the only criteria for success, we must also consider if the 4 gateways lead to the correct local topology for the $+$ intersection. We use this more strict criteria to determine whether a particular run of our algorithm on a test image was successful or not.

Rotations. Because thinned skeletons are rotation dependent, we expected to see a significant effect of rotation of local topology abstraction. Figures 6.30-6.33 show the (lack of) impact of rotation on our algorithm.

Figure 6.30 illustrates that successful gateway detection and subsequent local topology abstraction are largely independent of rotation. The horizontal nature of the accuracy graphs show that we get the same accuracy over each 1200 image subset of the data that varies across image resolution and noise. Additionally, if we take each of these 1200 image subsets and graph them with respect to image resolution and noise (calculating the mean and standard deviation over the 10 rotation cases), we see that the standard deviation error bars are reasonably small (Figure 6.31(a,b)). These error bars seem even smaller when we compare these variances to the variances experienced when examining rotation accuracy over various noise and image resolutions (Figure 6.31(c,d)).

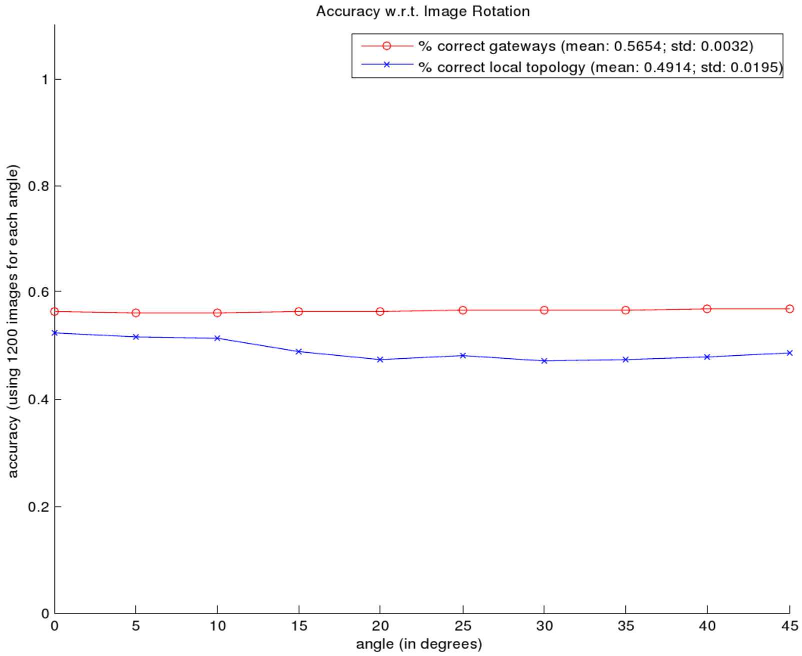
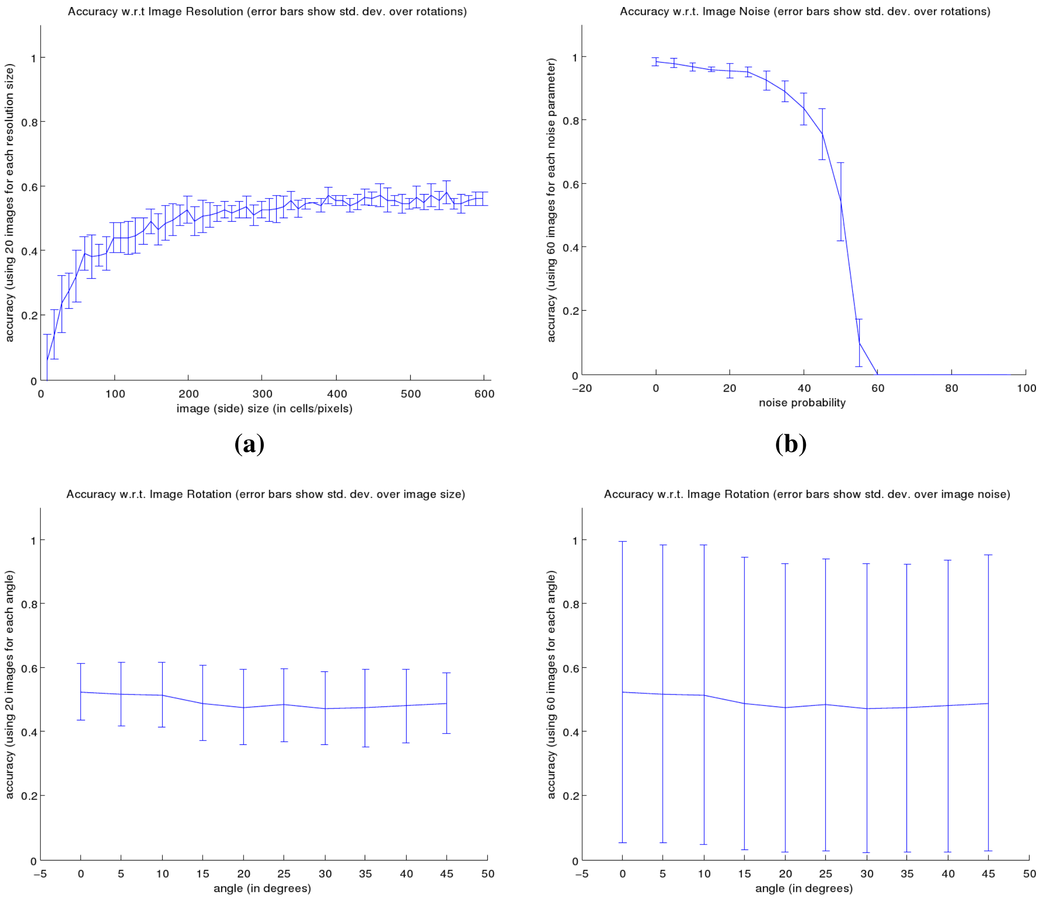


Figure 6.30: *Impact of rotations on gateways and local topology.* The classification accuracy of local topology abstraction is plotted over the different angles used in the test data. The accuracy is expected to be rather low, as each angle uses 1200 images from all size and noise values. The fact that the accuracy does not change with rotation demonstrates that rotation is largely independent of classification accuracy.


Accuracy w.r.t Image Resolution (error bars show std. dev. over rotations)

Accuracy w.r.t. Image Noise (error bars show std. dev. over rotations)

Accuracy w.r.t. Image Rotation (error bars show std. dev. over image size)

Accuracy w.r.t. Image Rotation (error bars show std. dev. over image noise)

(a) **(b)**

(c) **(d)**

Figure 6.31: *Variance of local topology classification across rotations.* If classification accuracy is largely independent of rotation, then we expect to see small standard deviation error bars when we graph the accuracy over LPM size or LPM noise while using each rotational value as a separate set of test images. Notice that in images (c,d) we have much larger variance with respect to the LPM size and LPM noise than in images (a,b) where the error bars show the variance over rotations. **(a)** We plot the classification accuracy of local topology abstraction over the LPM size. Classification accuracy is the success rate for 20 different test images—one for each noise level. The error bars show the standard deviation of classification accuracy across 10 different runs—one for each rotation value. Notice how small these variances are when compared to image (c). **(b)** Similarly, we plot the classification accuracy for 60 different test images (corresponding to all LPM sizes) over the 20 noise levels. The error bars are across the 10 different rotations. **(c)** Here, classification accuracy is the success rate for 20 test images (different noise parameters) for each of the 10 rotations. The error bars show the standard deviation of classification accuracy across 60 different runs—one for each LPM size. **(d)** Accuracy for the 60 test images that correspond to different LPM sizes at each of the 10 rotations. The error bars are for 20 runs across different noise levels.

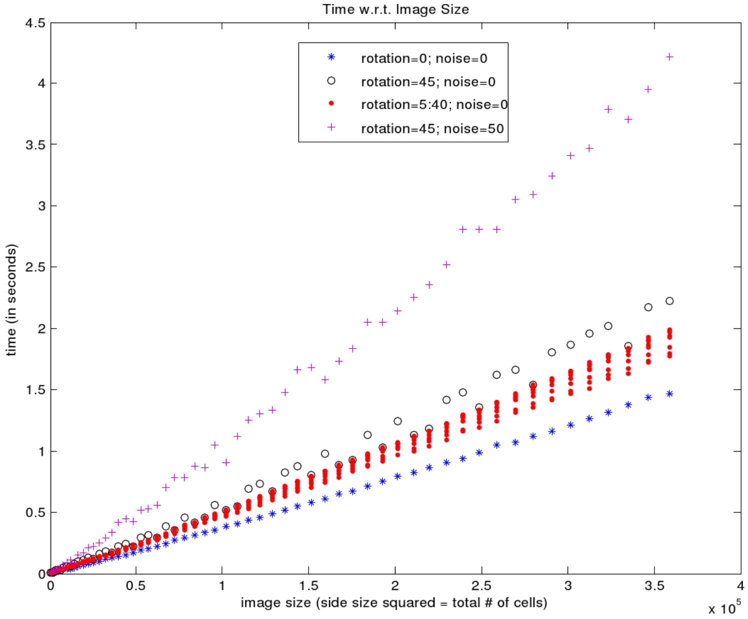
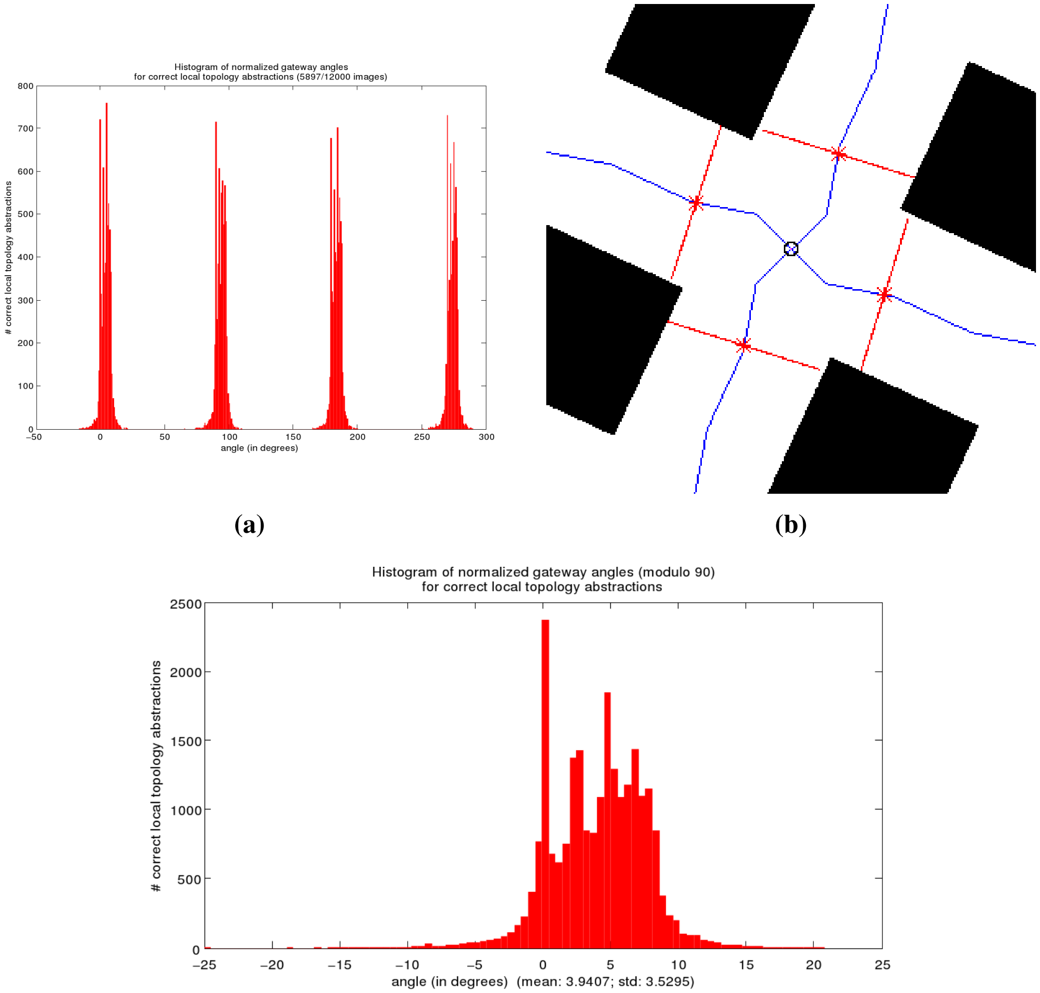


Figure 6.32: *Time performance of local topology abstraction.* Though the gateway (and the entire local topology) algorithm remains linear with respect to total image size, we observe that images rotated by 45° use 50% more compute time than images of the same size that are not rotated. This demonstrates that the thinning algorithm slows down when computing the skeleton along diagonals (w.r.t. the grid axes). (Adding noise increases run time as well. Though the "island" obstacle removal is linear, the full local topology algorithm takes almost twice as long at 50% noise as with no noise.)



(a) **(b)**

(c)

Figure 6.33: *Stability of gateway orientations.* For each image where the correct local topology is found, we rotate the image back to the original orientation and record the orientation of the gateways with respect to the image axes. **(a)** The histogram of the gateway orientations shows that we have 4 well separated clusters. **(b)** Here is a sample (299x299 cells, 25° rotation, 0 noise) where all four gateways are approximately 4 degrees off from the theoretical orientations of 0, 90, 180, and 270 (after the image is rotated back to the original orientation). This is due to the type of skeleton the thinning algorithm creates along diagonal paths. **(c)** The normalized gateway orientations can be grouped into a single cluster (modulo 90°). This illustrates that we have some images where gateways are oriented perfectly with the paths, but most are slightly offset due to the phenomenon in image (b).

We do observe that using a thinning algorithm does create a slight computational overhead when dealing with diagonals. Figure 6.32 shows that our gateway/local topology abstraction algorithm is linear in the number of LPM cells. We see a $\sim$50% increase in the time, which is largely due to examining the same cells multiple times when trying to compute a skeleton along diagonals.

We are also interested in observing how image rotations affect the gateway orientations with respect to the 4 straight path segments that emanate from the intersection. For each image where the correct local topology is found, we rotate the image back to the original orientation and record the orientation of the gateways with respect to the image axes. Figure 6.33(a) illustrates that we get 4 distinct clusters of gateway orientations. This shows that rotations are not largely affecting the skeleton, which is used to hypothesize gateway orientation.

Figure 6.33(c) shows these angles modulo 90°. We observe an *approximate* Gaussian with standard deviation 3.5295. Notice the mean and median deviate from 0 by 3.9407 and 4.3115 respectively. This is the only artifact of using the thinning algorithm on arbitrarily rotated paths that we observe in this evaluation. Figure 6.33(b) illustrates why this occurs—the thinned skeleton approximates angled paths with a collection of piecewise linear segments. While this does not affect the functional outcome of the local topology abstraction for the $+$ intersection, it could easily cause two distinct local topologies to be created for the same place at more complex intersections (e.g., where the gateways are further apart with respect to the LPM size). As such, alternative gateway orientation and/or gateway alignment methods should be investigated for future implementations.

LPM Resolution and Noise. We have determined that our algorithm is relatively robust to rotations in this $+$ intersection example. We now examine how reliable our algorithm is under different LPM resolutions and noise conditions. We use the fact that classification accuracy is largely independent of rotation in order to consider each of the 10 rotations as independent samples for each of the 1200 resolution/noise pairs.

As LPM size increases, we expect to see better classification performance until a plateau is reached—at some point adding more resolution to an LPM will not enhance the performance of local topology abstraction. This is shown by Figure 6.31(a) when all noise levels are used, while Figure 6.34(a) graphs the accuracy of local topology abstraction over LPM size in the cases where no noise exists. We observe that at a resolution of 39x39 cells, we achieve 100% classification accuracy for the 10 different rotations. Figure 6.34(b) shows the gateways found for the single (noise-free) 29x29 case that failed—the resolution is simply too low to properly align gateways to obtain the correct local topology.

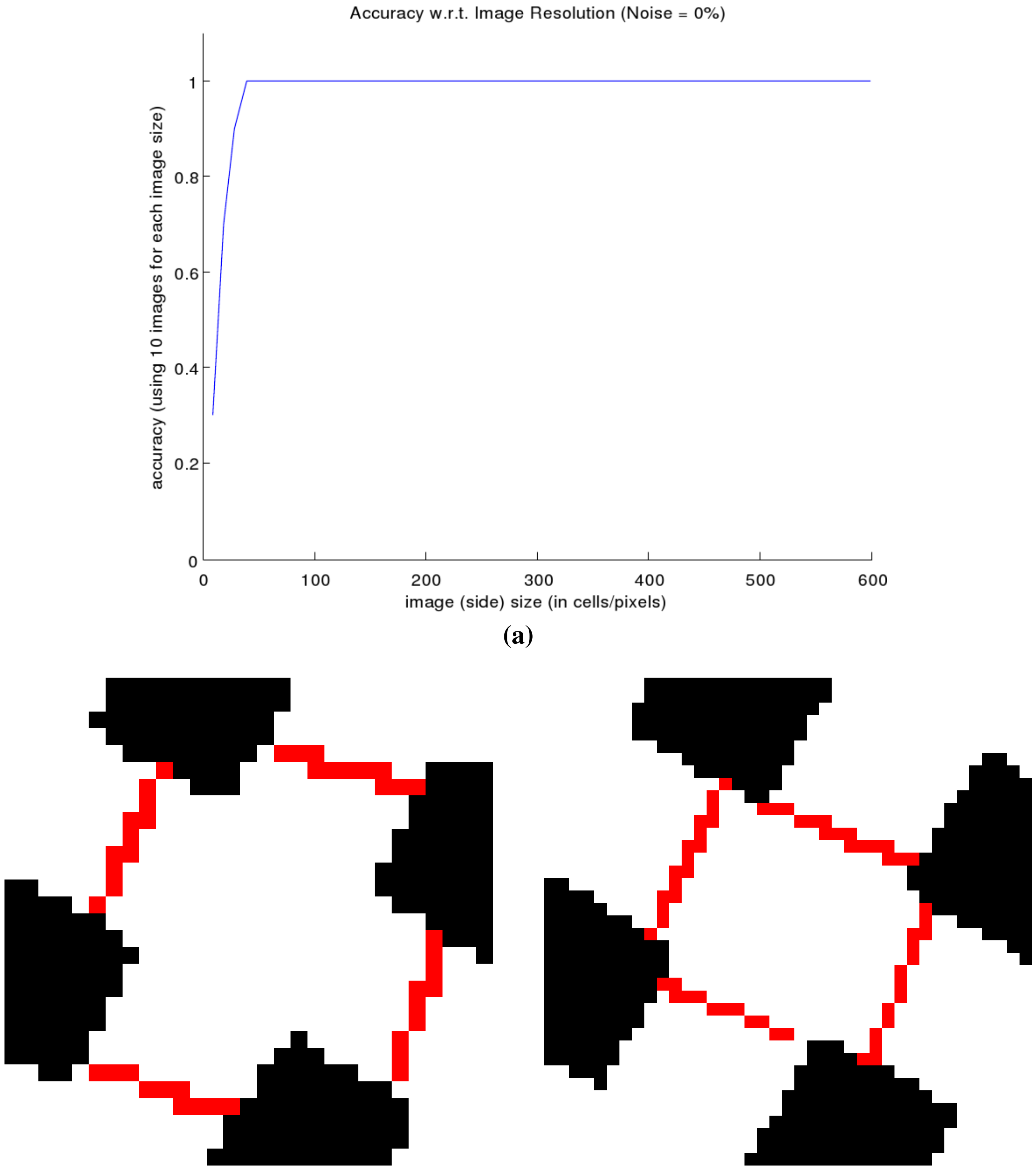
The success of the 39x39 cell case provides a lower bound on the resolution of an LPM. This shows that in order for our algorithm to work as intended, a path in the LPM should be at least 13+δ cells wide—13 is the width of the free space paths in the 39x39 cell grid, and δ is the number of cells (always rounded up) that equals the width of the robot. We can compute lower bounds on resolution and LPM size as follows:

$$\arg\max_{x} max\_path\_width/x - \lceil robot\_width/x \rceil \geq 13.$$

For example, if we have a 30 cm wide robot with very limited processing power, and we know that a path will never be wider than 3 meters, we can estimate the smallest usable LPM: $\arg\max_{x}$ 300 / x - $\lceil$30 / x$\rceil \geq$ 13; here, x = 20 cm/cell. For robustness we probably want a side size that can fit the path and the robot: at least 17 cells ($\lceil$330 / 20$\rceil$).

This lower-bound estimate assumes noise-free LPMs; however no real-world environment or robot can facilitate noise-free metrical modeling. By examining the classification accuracy of various noise levels with respect to a large LPM size, we obtain an upper bound on the amount of uniform noise that our algorithm can handle. Figure 6.35(a) shows the performance of local topology classification over different noise levels using the 10 rotated images of size 599x599.

We observe that for high resolution LPMs, our algorithm can handle complete uniform noise in free space (50% probability of flipping a free cell to occupied). On first



(a)

(b)

(c)

Figure 6.34: *Impact of LPM resolution on gateways and local topology.* **(a)** Local topology classification accuracy over all LPM sizes when no noise is present. We see that in the noise-free LPMs, all images ≥39 cells on a side abstract the correct local topology, despite rotation. **(b)** At 29x29 cells, there is one rotation (30°) where the local topology fails. Here the LPM resolution is simply too low to adequately orient the gateways; thus the algorithm finds 3 paths—the top gateway has normalized angle of 106° (instead of 90), so it does not properly align with the bottom gateway. **(c)** At 39x39 cells, the algorithm succeeds for all rotations when no noise is present.

thought, the idea that our algorithm can handle uniform noise in free space seems surprising; however, Figure 6.35(b,c) illustrates the reason why this occurs—the obstacle removal we perform as pre-processing to the skeleton removes much of this noise.

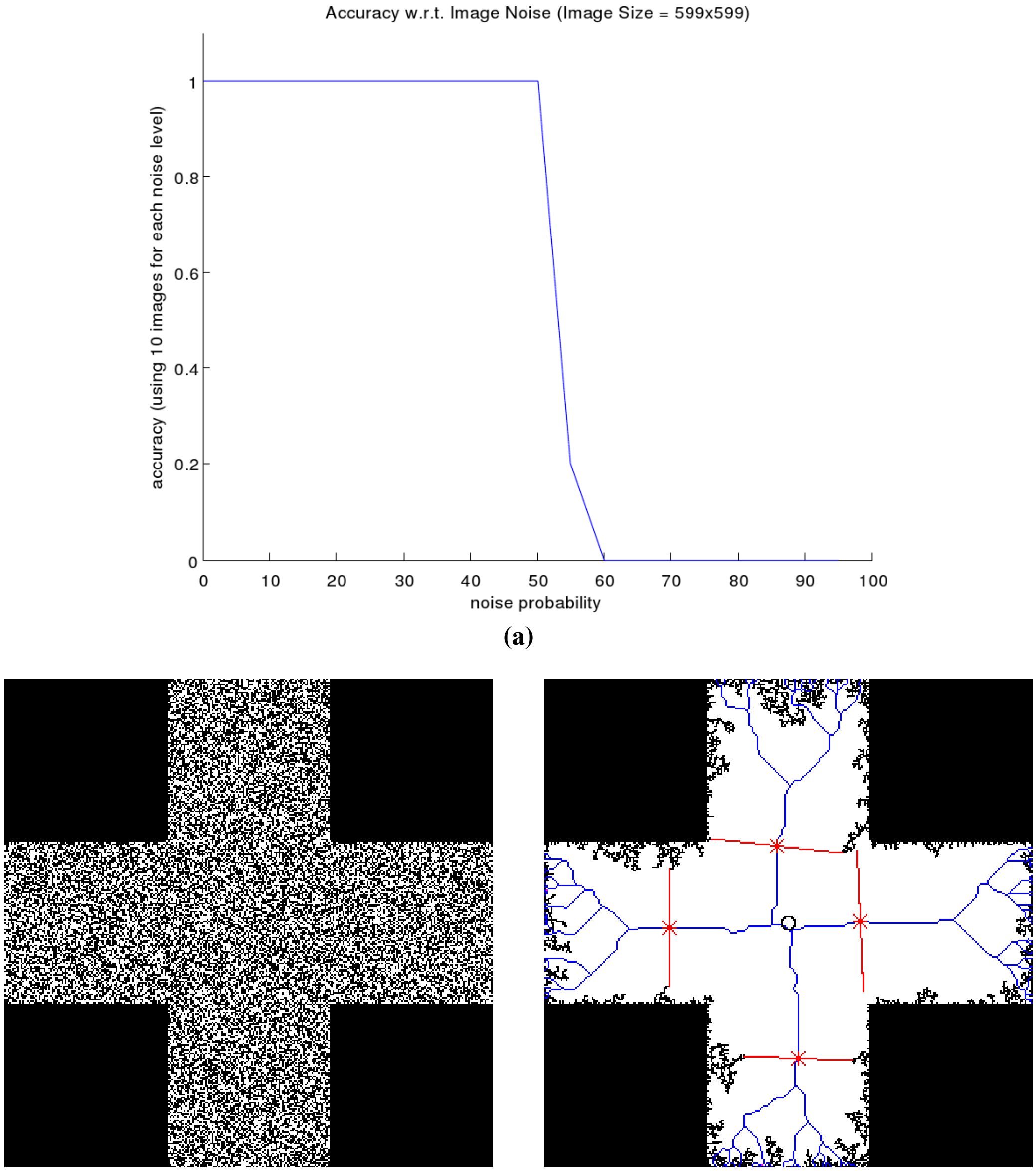
So far, we have determined the best-case upper and lower bounds where our algorithm works: 39x39 cell grids and 50% noise respectively. Of course, we do not expect our algorithm to work robustly in situations with both low resolutions and high noise. Figure 6.36 illustrates the resolutions and noise levels where we get 100% accurate local topology abstraction for all 10 rotations. We see that in general, our algorithm works at almost all noise levels $<$50% when the LPM is at least 300x300.

We do notice occasional spots where lower resolution images reach 100% accuracy for a given noise level, while higher resolution images at the same noise level do not. We have found two separate cases for this. The first is the case where the remaining noise (after removing "island" obstacles) is near the true anchors of the $+$ intersection. This causes the skeleton branches to curve, and we see a situation very similar to Figure 6.9(a), where the gateway is located at a bend in the skeleton, thus its orientation is incorrectly determined (Figure 6.36(b)). This can most likely be reproduced for lidar-based LPMs of real-world environments by simply placing an obstacle in an intersection, touching an anchor point.

The second case is where obstacles along the path boundaries iteratively pull the gateway towards the edge of the LPM. Figure 6.36(c) illustrates such a case, and highlights two issues with the actual algorithm. First is the problem where the iterative portion of Algorithm 6.3—meant to find three collinear points to define a gateway in corridor-style environments—can fail in certain scenarios due to the discretized skeleton.[4] We do not believe this is an issue if a more precise, continuous Voronoi graph is utilized. Second is the fact that our linear "island" obstacle removal does not remove obstacles that touch the edge of the LPM. Neither of these algorithmic issues has been an issue outside of this particular evaluation.

---

[4] The discretized skeleton may only have one obstacle within the Voronoi radius r_p of a skeletal point. So, we need to look for obstacles within $r_p + \sqrt{2}$. In rare scenarios (Figure 6.36(c)) this can lead to problems.

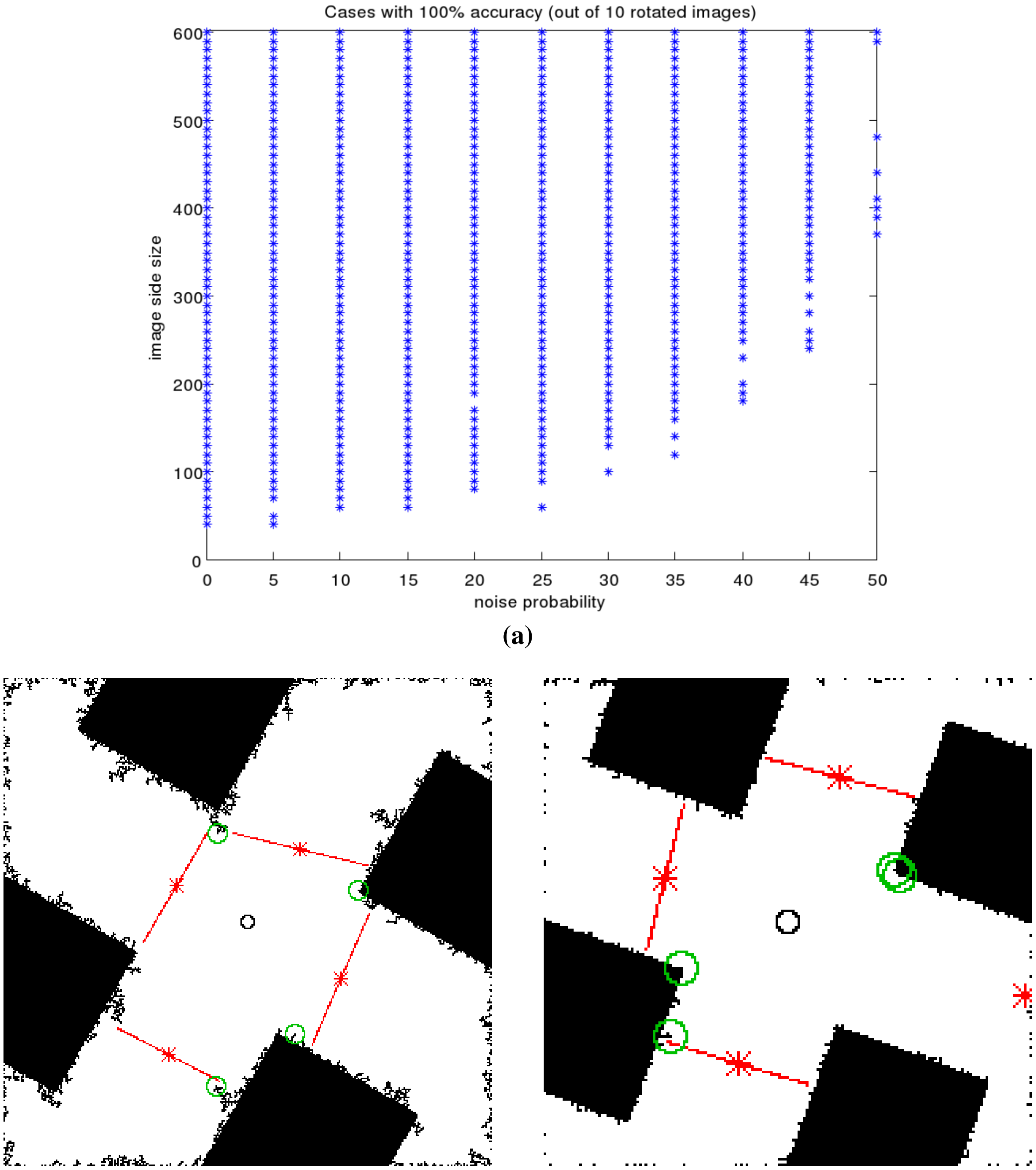


(a)

(b) **(c)**

Figure 6.35: *Impact of noise on gateways and local topology.* **(a)** Local topology classification accuracy over all noise levels for a 599x599 cell grid. We see that in high resolution LPMs, all images with $\leq$50% uniform noise abstract the correct local topology, despite rotation. **(b,c)** Here we show a test image (299x299 cells, 0° rotation) with 50% noise, and the final pruned skeleton and gateways abstracted. The local topology derived from these gateways is correct despite the noise. Note that because our "island" obstacle removal keeps obstacles that touch the edge, we get many spurs near the LPM edges.

Overall, the anchor-based gateway algorithm yielded 100% accuracy (3200/3200) for all test images $\geq$200 cells on a side and $\leq$ 35% free space noise. In cases between the lower bound of 39 cells per side and the upper bound of 50% noise, the algorithm achieved 91.72% accuracy (5751/6270). Of these 6270 images, 98.8% (6195) of them were found to have 4 gateways. Again this implies that we should consider a more robust gateway alignment technique in future work, while the anchor-based gateway detection algorithm seems quite dependable.



(a)

(b)

(c)

Figure 6.36: *Determining the bounds of successful local topology abstraction.* **(a)** Here we show all cases (of image size and noise level) where the anchor-based gateway algorithm yielded 100% successful local topology classification for the 10 possible rotations. **(b)** Notice in (a) that for 45% noise, <100% accuracy is obtained for images with side size of 309 cells. Here we show the single incorrect classification, which has 30° rotation. The noise affects the top and bottom gateways enough to make them unaligned by our simple alignment criterion. **(c)** For 20% noise, there is another misclassification (20° rotation) for LPM of side size 179 cells.

Chapter 7

Human-Robot Interaction: Evaluating the HSSH Local Levels

> *Never trust anything that can think for itself if you can't see where it keeps its brain.*
>
> J. K. Rowling, *Harry Potter and The Chamber of Secrets* (1999)

One of the unique characteristics of the Spatial Semantic Hierarchy framework with respect to other map-building approaches is that it is inspired by human spatial knowledge. Earlier, we argued that this would not only benefit robot navigation (Chapter 4), exploration (Chapter 5), and map-building (Chapters 8 and 9) but would also provide a useful framework for human-robot interaction. Here we show an initial evaluation of an interface for the local levels of the HSSH that sketches an optimistic picture of a human-robot interaction in the near future.

Chapter 4 discussed the LPMs of the Local Metrical Level, and we provided evidence that we could achieve robust localization, thus a high-quality LPM, during navigation. Chapter 5 discussed the Local Topology Level, where place detection and local topology abstraction are grounded by finding gateways in the LPM. We provided evidence of robust gateway detection in commonly seen LPMs. Here we evaluate these two levels together, expecting robust robot navigation, and observing the performance of human subjects utilizing the HSSH implementation [Beeson et al., 2007].

7.1 HSSH Interface for a Mobility Assistant

We evaluate the HSSH using a scenario where we believe the HSSH can improve on the current state-of-the-art: a robotic transport, such as an intelligent wheelchair, an autonomous vehicle, etc. Our specific domain is that of a transport that allows a person with a visual disability to achieve the same level of navigation performance as a person with full visual capabilities. Evaluation of the robotic transport is done in simulation, so that we can simulate a user with visual disabilities that also needs mobility assistance. Vizard [WorldViz LLC, 2007] provides a high-fidelity 3D virtual reality environment in which we created a simulated robot with range sensors.

The human-robot interaction levels for a robotic transport correspond nicely with the distinct representations in the Hybrid Spatial Semantic Hierarchy (Figure 7.1 and Table 7.1). The higher levels of interaction require more intelligence on the part of the robot, but they also require less effort for communication and supervision by the human driver. In order to maximize human autonomy, the driver can shift freely between the different levels at any time.

7.1.1 Local Metrical Control

Humans have reliable metrical models of their local surroundings [Golledge, 1999]. People can navigate safely through complex small-scale spaces, and they can use verbal and graphical descriptions to communicate spatial relationships [Skubic et al., 2004a,b]. The HSSH emulates this functionality using the fixed-size local perceptual map (LPM) that follows its motion and describes its small-scale surroundings. Obstacles in the LPM can be classified as static or non-static [Modayil and Kuipers, 2004], which makes it possible to identify and model dynamic hazards such as cars or pedestrians, and to identify structures such as doors that can change the apparent topology of places.

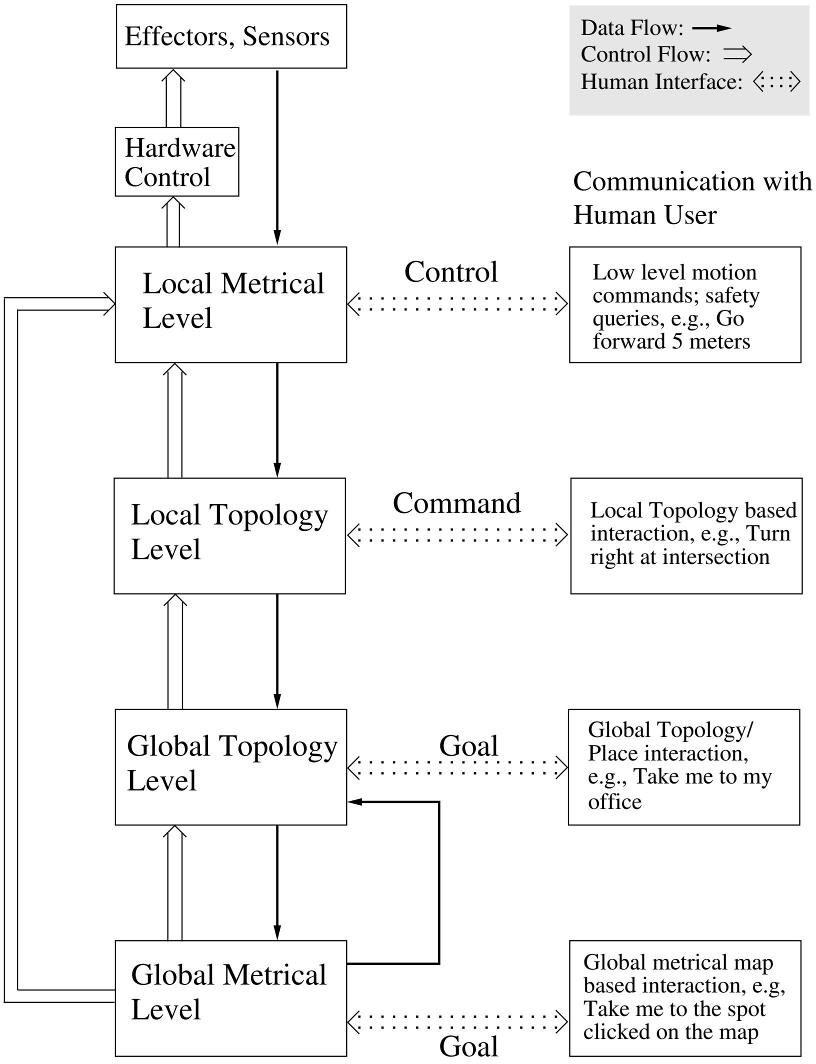

Effectors, Sensors
Data Flow:
Control Flow:
Human Interface:
Hardware Control
Communication with Human User
Local Metrical Level
Control
Low level motion commands; safety queries, e.g., Go forward 5 meters
Local Topology Level
Command
Local Topology based interaction, e.g., Turn right at intersection
Global Topology Level
Goal
Global Topology/ Place interaction, e.g., Take me to my office
Global Metrical Level
Goal
Global metrical map based interaction, e.g, Take me to the spot clicked on the map

Figure 7.1: *HSSH interface.* The HSSH is an integrated framework of multiple, disparate representations of spatial knowledge, each with its ontology motivated by human cognitive abilities. The HSSH interface uses three levels, *Control*, *Command*, and *Goal*, that allow a user to interact with a robot using the HSSH.

Hybrid Spatial Semantic Hierarchy (HSSH)	**Human-Robot Interface**
Local Metrical Level: Environment is modeled as a bounded map of small-scale space within the agent's perceptual surround that scrolls with the agent's motion, without being tied to a global frame of reference. Useful for "situational awareness" of the immediate surround.	**Control**: Driver uses the joystick to give direction and speed of motion. Robot checks commands against the local map for safety. Driver may specify a target or direction of motion within the local map. Robot plans hazard-avoiding motion toward that target.
Local Topological Level: Environment is modeled as a set of discrete decision points, linked by actions: *turn* selects among options at a decision point, and *travel* moves to the next decision point.	**Command**: Driver specifies actions to take at, and to move among, decision points. Robot identifies decision points in the environment and interprets instructions appropriately, translating actions into hazard-avoiding control laws.
Global Topological Level: Environment is modeled as a network of places, on extended paths, contained in regions, allowing efficient route planning by graph search.	**Goal**: Driver specifies a destination place in a topological map, by name or in a schematic diagram (like a subway map). Robot plans a route to that goal, which is translated into a sequence of commands, which are translated into hazard-avoiding control laws.
Global Metrical Level: Environment has a geometric model in a single global frame of reference. Useful for route optimization when available.	**Goal**: Driver specifies a destination place in the global metrical map, like a printed city map. Robot plans a route to that destination in the topological map.

Table 7.1: *Robot transport interface to the HSSH.* The levels of our human-robot interface for a robotic transport correspond to the levels of spatial knowledge as represented in the HSSH.

In our domain, the human driver specifies a desired direction of motion using the joystick. (This is the *Control* interface of Table 7.1 and the evaluation below.) The HSSH trajectory planner ensures this corresponds to a safe trajectory in the LPM. The LPM abstraction also supports a GUI that allows the driver to specify a destination point in the local region. Eventually, the Control interface should also support gestures or natural language to specify pre-labeled destination poses in the local environment, such as "*Go to the desk*".

7.1.2 Local Topological Control

Starting from the perceptual model of the local surrounding small-scale space, humans generate symbolic descriptions of the navigational affordances of the local space, and therefore its qualitative decision structure. This is reflected in the language people use to describe intersections by shape (e.g. "T" or "corner") or number of paths ("three-way intersection" or "dead end") [MacMahon et al., 2006]. A robotic transport should understand terms the human driver finds useful and comfortable, including qualitative decision commands that presuppose knowledge of the local decision structure, such as "Turn right" or "Take the second left". Fortunately, these terms correspond well with the HSSH Local Topological Level.

As the robot moves through the environment, maintaining the LPM as an accurate metrical model of local small-scale space, it describes the local topology of that space in terms of local-paths and gateways. The local place topology is described as a circular order of directed path fragments and gateways, which translates directly to the large-scale space description of a place as a node in a graph, connected to paths. In large-scale space (where most route planning takes place), a command such as "Turn left" selects an outgoing directed path, given the incoming one. In small-scale space (where motion control actually takes place), the same command translates to a desired trajectory of motion from an incoming gateway to an outgoing gateway.

Thus, at the *Command* interaction level, both the driver and the robot represent space as a graph of decision points, and a command specifies the option to select at the next decision point. In the current implementation, the available commands are Forward, Right, Left, Turn Around, and Stop. Compound commands such as "Take a sharp right at the second intersection" will be implemented in future work by integrating the use of complex verbal instruction [MacMahon et al., 2006] with the resolution of qualitative references to small-scale space [Skubic et al., 2004b].

By instructing the robot at the *Command* level, the human driver is delegating the autonomy for hazard-avoiding travel between places (the robot plans its own safe route along the path using the LPM), for recognizing the decision structures of places, and for selecting the intended option at that place. The human driver can always reassert his authority by issuing a *Control* level command using the joystick or by pressing the Stop button.

7.1.3 Global Topological and Metrical Control

Human-drawn maps based on exploration tend to be topologically correct, even though they often contain gross metrical errors [Lynch, 1960; Siegel and White, 1975; Golledge, 1999]. People tend to solve way-finding problems primarily based on their topological knowledge of the environment. On the other hand, people often find it helpful to use metrically accurate graphical maps in familiar environments.

The HSSH builds a global topological map from the agent's travel experience, expressed as a sequence of places with local topologies, linked by travel actions. The global topological map is found by creating a tree of all possible topological maps, pruning out inconsistencies, and focusing on the most likely consistent map. A global metrical map in a single frame of reference can then be built robustly on the skeleton provided by the global topological map.

At the *Goal* level of human-robot interaction, the driver instructs the robot by specifying a destination. The intended destination could be specified by name ("the main library"), by description ("the corner of Speedway and 24th Street"), by GUI selection from a topological diagram (like a subway map), or by GUI selection from a graphical metrical map of the environment.

The robot can learn the global topological map from its experience, accumulating metrical annotations, and constructing the global metrical map if resources permit. Such a map, learned from experience or extracted from a graphical map, is necessary for *Goal* level control to be possible. Assuming that the robot has already learned a sufficiently complete topological map, planning a suitable route is a straightforward graph search. Carrying out the plan consists of translating it first to a sequence of decisions to take at specific places, and then to a sequence of hazard-avoiding control laws for traveling from place to place, and from gateway to gateway in each place neighborhood.

Building a correct global map may require large amounts of exploration, maybe using specific exploration strategies; however, when the robot is serving a human driver, its opportunities for learning the (local and global) structure of the environment are constrained by the needs and desires of the human driver. This potential conflict between the driver's authority and the robot's need to explore the environment should be reduced. As such, we have postponed working on the *Goal* interface until we explore this issue of human-in-the-loop exploration versus exploitation in more detail. Future experiments that evaluate the *Goal* interface need to measure human cognitive load or other useful information that allow us to evaluate all of the advantages of a fully autonomous human transport system.

7.2 Empirical Evaluation

The bottom-line question is: *How well does the robotic transport help people with disabilities get around*? Our goal for the robotic transport is for the robot's own perception, mapping, and decision-making capabilities to augment those of the human driver, under the human driver's ultimate authority. The robot will initially be most useful to subjects who have perceptual as well as motor disabilities, so we evaluated the robot's HSSH interface by comparing subjects with normal and degraded visual stimuli.

7.2.1 Experimental Setup

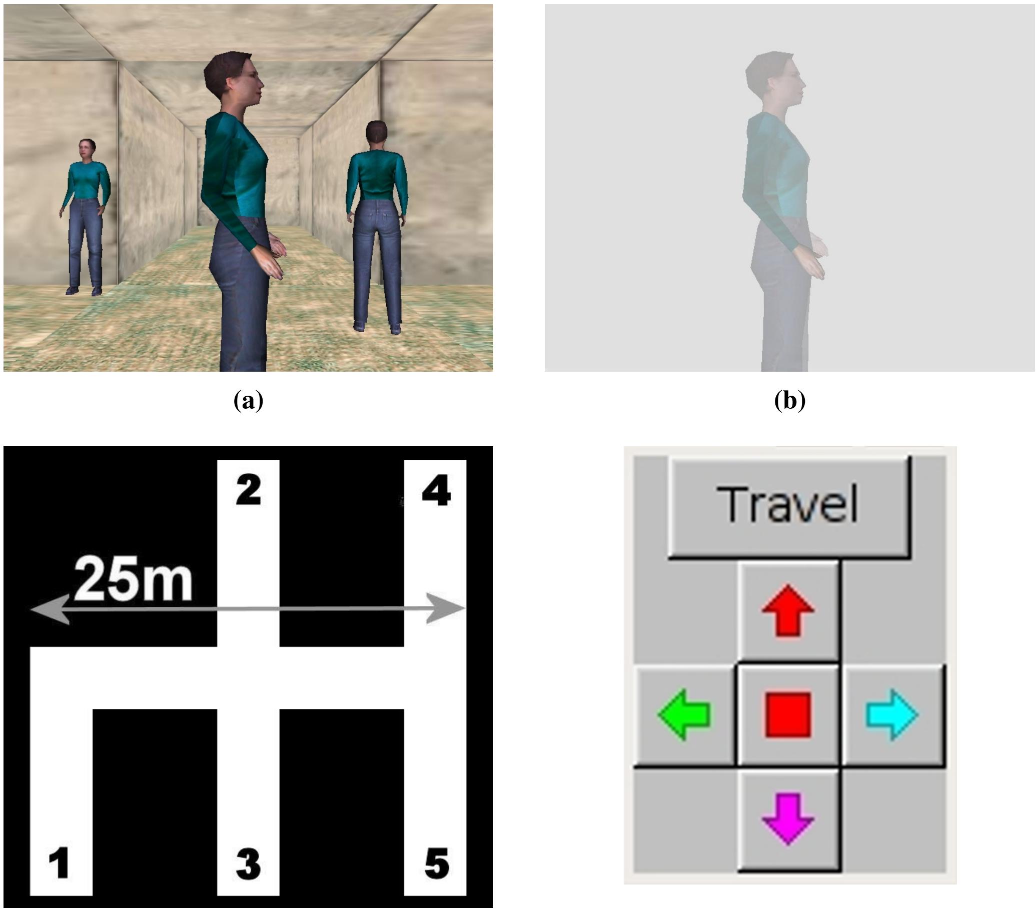
The algorithms for robot navigation were implemented and tuned using physical robot platforms; however, to simulate visual impairment in a manner safe for the human subjects, we ran our initial experiments in high-fidelity virtual reality environments using Vizard [WorldViz LLC, 2007]. These experiments were run in an indoor virtual environment for two perceptual cases: Degraded, in which the subjects' visual input was degraded by adding fog to the environment so that objects farther than 3.0 meters from the camera could not be seen, and Normal, which had no visual degradation (Figure 7.2(a,b)). The simulated robot itself perceives the virtual environment using omnidirectional range sensors, unaffected by the fog.

There were *four* experimental conditions—Normal:Manual, Degraded:Manual, Degraded:Control and Degraded:Command—each defined by the subjects' perception and the level of autonomy available. With the *Manual* interface, the joystick controls the motors directly, with no robotic intelligence. The *Control* interface requires the robot to build and maintain an LPM in order to avoid collisions while following the instructions given by the user via the joystick—speed is assumed to be a predetermined constant (1 meter/s) that is instantaneously achieved.[1] The *Command* interface requires the robot to detect places and describe their local topologies in order to execute the turn and travel decisions specified by the driver via a GUI with command buttons (Figure 7.2(d)); however the buttons on the joystick are easily mapped to these button presses, which will be needed for future experiments with actual visually impaired subjects.

The experimental task was to navigate between a sequence of known places. The large-scale virtual environment consisted of seven hallway segments (Figure 7.2(c)) with ten avatars (Figure 7.2(a)). Each subject knew the layout (and place labels) of the environment but not the locations of the avatars (stationary obstacles), which were randomly distributed for each trial.

---

[1] One advantage in simulated environments is that we do not have to model acceleration and deceleration based on robot/driver weight, surface properties, etc.



(a) **(b)**

(c) **(d)**

Figure 7.2: *Stimuli, map, and interface for the HRI experiment.* **(a)** Subject's environment view in the Normal Vision condition. **(b)** Subject's limited view in Degraded Vision (fog) condition. **(c)** Environment layout used. Numbers correspond to goal states. **(d)** The local symbolic control GUI. Individual buttons light up when applicable for the current environmental situation.

Each trial started by placing the simulated robotic transport at the same starting position (the lower end of the center corridor). At the beginning of a trial, the subject was told by the computer the current location and destination (e.g., "Position 3. Go to position 5"). Depending on the condition of the trial, the subject used the joystick (*Manual* and *Control* interfaces) or the *Command* level interface (Figure 7.2(d)) to travel to the specified goal. The computer would indicate when they had reached each goal location by announcing the goal location and would then give the subject another goal location (e.g., "Position 5. Go to Position 2").

Each trial consisted of a sequence of five goal locations. There were five predetermined sequences that were used for each of the four experimental condition; thus each subject ran a total of 20 trials. To avoid learning effects, the 20 trials were randomly ordered for each subject. For these initial experiments we ran three subjects. We logged their motion through the environment and when/where they collided with both avatars and walls.

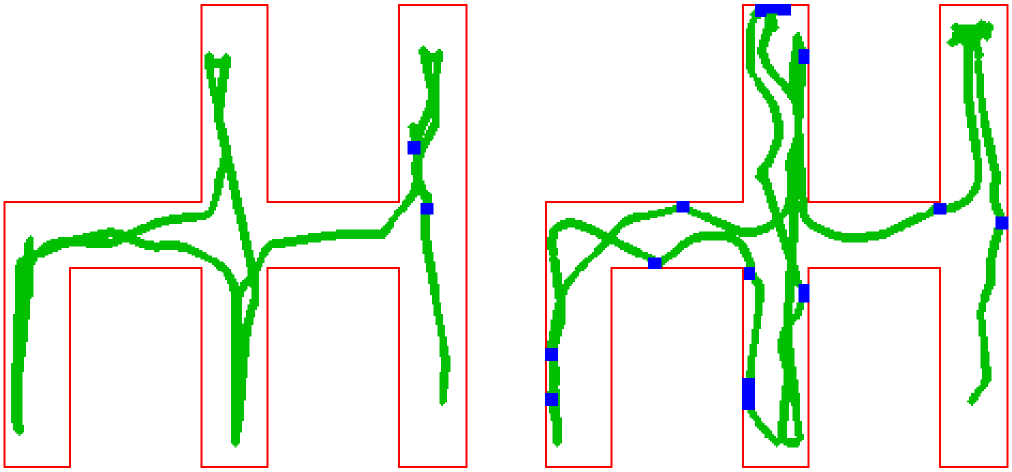
This evaluation is intended to address three questions:

1. Does reducing the visual information by adding fog make the task more difficult?
2. Is there a performance benefit from adding the *Control* level (collision avoidance) navigation aid?
3. Is there a performance benefit from adding the *Command* level aid (autonomous low-level control and place detection/local topology abstraction)?

7.2.2 Results

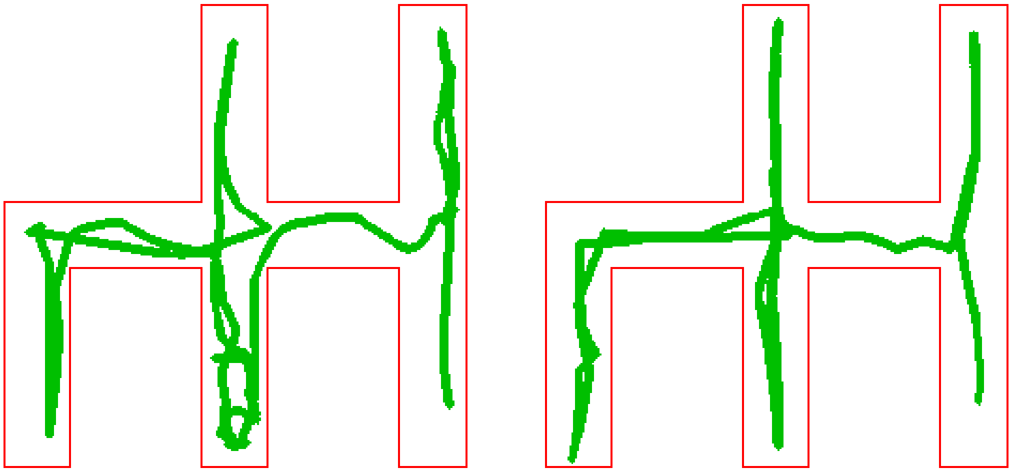
The answer to all three questions is, "Yes". Figure 7.3 shows sample recorded traces from subjects in the four conditions. Figure 7.4 shows the mean distance traveled and the mean number of collisions of a trial in each condition.

Not surprisingly, we see a strong effect from degrading the visual input by comparing the performances in the Normal:Manual versus Degraded:Manual in Figure 7.4. There



Normal:Manual

Degraded:Manual



Degraded:Control

Degraded:Command

Figure 7.3: *Example experimental runs.* Sample traces of the paths used by the human subjects in the four different conditions, with collisions indicated by small squares.

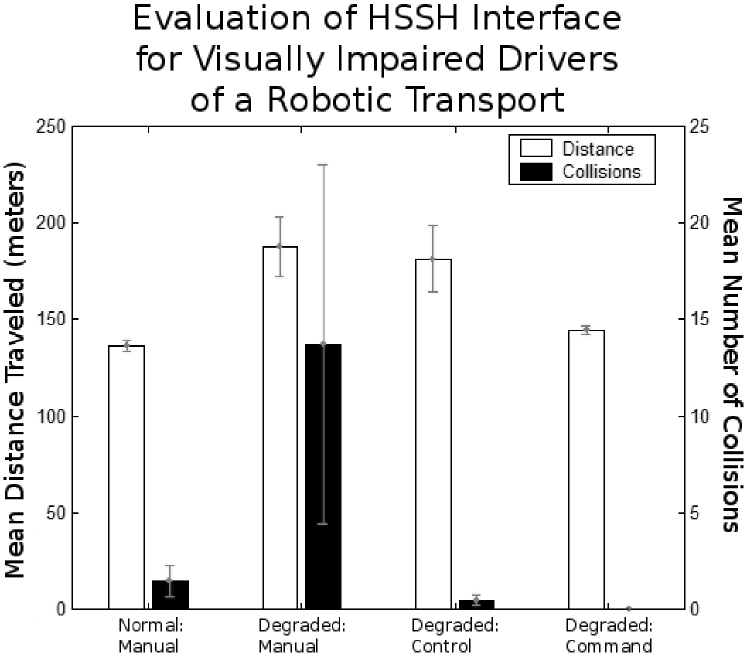


Figure 7.4: *HRI experiment results.*Summary data for the evaluation of the four conditions. The open (white) bars show the mean distance traveled to complete a sequence (left y-axis). The solid (black) bars show the mean number of collisions (right y-axis). Error bars represent standard deviations.

was a 37% increase in the distance traveled (from 136.53 to 187.62 meters) and a 936% increase (from 1.47 to 13.73) in the number of collisions.

To evaluate the benefit of adding *Control* level navigation, we compared performances for the Degraded:Manual versus the Degraded:Control conditions. Adding control level navigation showed little difference in the distance traveled (from 187.62 to 181.29 meters). However, there was a dramatic change in the number of collisions: a 97% decrease from 13.73 to 0.47.

To evaluate the benefit of adding *Command* level navigation, we compared performances for the Degraded:Manual versus the Degraded:Command conditions. In this case, adding *Command* level navigation did reduce the distance traveled: a 23% decrease from 187.62 to 144.31 meters. Furthermore, there was a 100% decrease in the number of collisions (from 13.73 to 0.0).

If we compare performance for the Degraded:Command condition versus the Normal:Manual condition, we can evaluate how well autonomous navigation does relative to a driver with full visual abilities. There was a benefit for Degraded:Command navigation in terms of the number of collisions (Normal:Manual = 1.47; Degraded:Command = 0.0). We also find that there is a small difference in the distance traveled (Normal:Manual = 136.53 meters; Degraded:Command = 144.31 meters). This difference in distance was mainly due the lack of "corner cutting" by the current autonomous control system (Figure 7.3(d)).

These results demonstrate distinct benefits from the *Control* and *Command* level navigation aids. In visually degraded conditions, *Control* level navigation improves obstacle avoidance, but does not reduce the distance traveled. By contrast, *Command* level navigation (which includes *Control*) reduces both the number of collisions *and* the distance traveled. Our preliminary data suggests that the resulting performance is similar to that of the condition with no visual degradation.

In future tests of the *Goal* interface, we expect to get similar results with respect to autonomous navigation. The environment used here was small enough that the human subjects made very few Causal Level mistakes in commanding the robot. Larger environments will undoubtedly demonstrate that the using the *Goal* interface reduces the human user's cognitive load and should yield optimal large-scale plans.

Chapter 8

Closing Loops: The HSSH Global Topology Level

I believe the future is only the past again, entered through another gate.

Arthur Wing Pinero, *The Second Mrs. Tanqueray* (1893)

The next two chapters will address the problems of building a global topological map to describe the qualitative structure of large-scale space and building a global metrical map to describe its geometric structure within a single global frame of reference. We describe these two map-building problems separately, but their solutions benefit from each other and should be interleaved in future research (Section 10.1.2).

The first problem is to identify the best global topological map consistent with exploration experience. The process of generating possible topological maps from experience and testing them for consistency can provide formal guarantees that the correct map is generated and never discarded [Dudek et al., 1993]. A logic-based theory of topological maps [Remolina and Kuipers, 2004] makes explicit the assumptions upon which those guarantees depend.

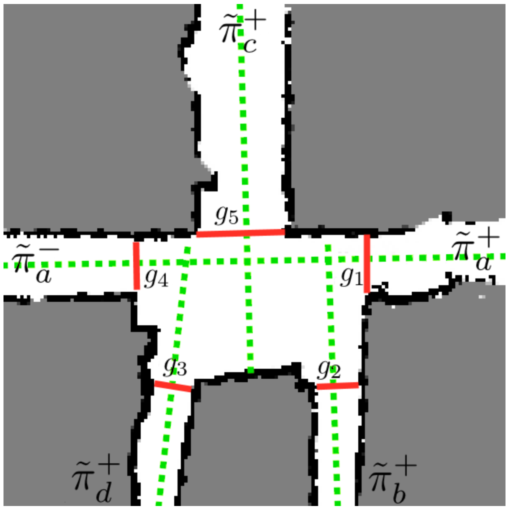
If the robot knows it is in an environment with no loops, creating a topological map is quite easy. This is especially true given deterministic actions, as the robot simply moves deterministically between known places when it revisits parts of the environment. Even with non-deterministic actions, creating the topology of such environments is still possible [Tomatis et al., 2002]. The difficulty in map-building arises from closing loops: determining when a newly-encountered place is the same as a previously-experienced place, and creating a hypothesized new loop in the topological map. When large loops in the environment result in structural ambiguity, a topological representation can concisely represent the loop-closing hypotheses by generating a single topological map for each qualitatively distinct alternative.

8.1 From Small-Scale to Large-Scale Star

In small-scale space, the LPM is used for the detection of gateways, local-paths, and places, and to create the local map m_p that is stored at places. The small-scale star describes both a circular order on the set of directed local-paths in a place neighborhood and also the correspondence between directed local-paths and oriented gateways. A place p in large-scale space is associated with the local map m_p, a model of the place neighborhood in small-scale space. At a place, each directed local-path $\tilde{\pi}^d$ in small-scale space corresponds to a directed path π^d in large-scale space. This allows us to determine the large-scale star that describes the circular ordering of topological directed paths at the place.

Assimilating the local topology of a place into the global topological map requires a 1-1 mapping between the directed local-paths in the small-scale star and a set of directed paths from the global topological map. In Figure 8.1, we illustrate such a mapping between the local-paths, $\tilde{\pi}_a$, $\tilde{\pi}_b$, $\tilde{\pi}_c$, and $\tilde{\pi}_d$, and the corresponding global topological paths π_1, π_2, π_3, and π_4, respectively. To keep this example simple, we specified $+$ and $-$ on the directed paths to correspond consistently, but of course this need not be true in general.

In large-scale space, a distinctive state q corresponds uniquely to a place, a path, and a direction on that path (Equation 3.1). Thus, the dstate q is at a particular place p, and there is a bijective association between a dstate and a directed local-path: $\psi_p(q) = \tilde{\pi}^d$ where $\tilde{\pi}^d \in$



(a)

Small-scale star description

$$
\begin{array}{rcl}
((\langle \tilde{\pi}_a^+, 1\rangle & \leftrightarrow & \langle g_4, in\rangle, \langle g_1, out\rangle) \\
(\langle \tilde{\pi}_b^+, 1\rangle & \leftrightarrow & \langle g_2, out\rangle) \\
(\langle \tilde{\pi}_c^-, 0\rangle & \leftrightarrow & \langle g_5, in\rangle) \\
(\langle \tilde{\pi}_d^+, 1\rangle & \leftrightarrow & \langle g_3, out\rangle) \\
(\langle \tilde{\pi}_a^-, 1\rangle & \leftrightarrow & \langle g_1, in\rangle, \langle g_4, out\rangle) \\
(\langle \tilde{\pi}_d^-, 0\rangle & \leftrightarrow & \langle g_3, in\rangle) \\
(\langle \tilde{\pi}_c^+, 1\rangle & \leftrightarrow & \langle g_5, out\rangle) \\
(\langle \tilde{\pi}_b^-, 0\rangle & \leftrightarrow & \langle g_2, in\rangle))
\end{array}
$$

(b)

An example large-scale star abstraction

$$
\begin{array}{rcl}
((\langle \pi_1^+, 1\rangle & \leftrightarrow & q_1) \\
(\langle \pi_2^+, 1\rangle & \leftrightarrow & q_2) \\
(\langle \pi_3^-, 0\rangle & \leftrightarrow & q_3) \\
(\langle \pi_4^+, 1\rangle & \leftrightarrow & q_4) \\
(\langle \pi_1^-, 1\rangle & \leftrightarrow & q_5) \\
(\langle \pi_4^-, 0\rangle & \leftrightarrow & q_6) \\
(\langle \pi_3^+, 1\rangle & \leftrightarrow & q_7) \\
(\langle \pi_2^-, 0\rangle & \leftrightarrow & q_8))
\end{array}
$$

(c)

Figure 8.1: *Binding local-paths to distinctive states.* **(a)** Gateway locations and directions are used to identify the directed local-paths and to determine which pairs satisfy the path continuity requirements. **(b)** The small-scale star enumerates directed local-paths in clockwise order, describing their traversability and association with gateways. Note: the robot entered the place via g_5; thus, it arrived on directed local-path $\tilde{\pi}_c^-$. **(c)** The large-scale star replaces local-paths with topological paths from the global topological map, and defines a distinctive state for each directed path at this place. This environment has five gateways, four paths, and eight distinctive states.

S_p. This implies that in the case where the directed local-path passes through the place, the distinctive state q will correspond with two different oriented gateways, one $\langle g, in \rangle$ entering the place neighborhood, and the other $\langle g', out \rangle$ departing from it.

An isomorphism $\phi : S \to S'$ between two stars implies a bijective mapping between the associated dstates as well. We will extend ϕ to write these implied mappings as $\phi(q) = q'$. For a topological map M^T, and the set P of places in M^T, we can now define the set of local place maps,

$$M^P = \{\langle p, m_p, S_p, \psi_p \rangle \,:\, p \in P\}$$

associating each place $p \in P$ with its local metrical map m_p, its local topology S_p, and ψ_p, the association between dstates and directed local-paths in the local topology.

Assuming that the LPM is sufficiently well explored, the set of directed local-paths and gateways in the small-scale star is complete, so the description of the distinctive states and directed paths in the circular order of the large-scale star is also complete. A *turn* action in large-scale space corresponds to motion in small-scale space within a place neighborhood from the inward-facing oriented gateway the robot arrived upon to an outward-facing oriented gateway (Figure 5.4(c)). Thus, for every pair of dstates q_i and q_j at the place, a causal schema for the turn action $\langle q_i, turn, q_j \rangle$ is implicitly defined. Exploration experience can now be described as an alternating sequence of travel actions and place neighborhoods, which simplifies construction of the global topological map (Algorithm 8.1).

8.2 The Tree of Possible Topological Maps

The topological map-builder maintains a tree whose nodes are pairs $\langle M, q \rangle$, where M is a topological map (augmented below for the HSSH) and q is a distinctive state within M representing the robot's current position. The leaves of the tree represent all possible topological maps consistent with current experience [Dudek et al., 1993]. Algorithm 8.1 reiterates the procedure for growing the tree of possible topological maps. It also details the differences between the basic SSH and the HSSH.

After each action a and resulting view v, we extend each map hypothesis at a leaf of the tree. If the current action moves within known territory, the map $\langle M, q\rangle$ will predict the resulting dstate q' and the view to be observed, so the hypothesis can be updated or refuted according to whether the prediction was correct or not. If the current action explores new territory, then either the resulting dstate is also new, or the action closes a loop and connects with a previously known dstate. Since there may be multiple possibilities that all match view v, the tree of topological map hypotheses will branch. For purposes of generating and testing candidate topological maps in the HSSH, we will extend the basic SSH topological map M^T with $M^P = \{(p, m_p, S_p, \psi_p) \ : \ p \in P\}$, the set of local metrical maps and local topologies of individual place neighborhoods.

$$M = \langle M^T, M^P\rangle$$

In the SSH, a view v is an abstracted description of the agent's perception of the local environment from a distinctive state q. We select the level of description to ensure that the view is a deterministic function of the dstate ($v = o(q)$), although we allow perceptual aliasing (different states with the same view) [Kuipers and Beeson, 2002]. In the basic SSH, a view is a symbol, abstracting away the nature of the perceptual system, and views are matched only for equality. In the Hybrid SSH, we define a view to be the local topology S_p of the current place p and the current directed local-path the robot is on; thus, the new view description is derived from the local topology, which is grounded in local perceptual map m_p.

$$\begin{aligned} v &= \langle S_p, \tilde{\pi}^d\rangle \text{ where } d \in \{+,-\} \\ &= \langle S_p, \psi_p(q)\rangle \end{aligned}$$

Given two views v and v', we say that $match(v, v')$ holds iff there is an isomorphism ϕ :

0. Perform initial action a_0 that brings the robot to a place along a directed path. Initialize the tree of maps with the map hypothesis $\langle M_0, q_0 \rangle$, where M_0^C contains the single dstate q_0 with its observed view v_0, and M_0^T contains the single place p_0 and path π_0.

After performing a new action a and observing the resulting view v, for each consistent map $\langle M, q \rangle$ on the fringe of the tree:

1. If M^C includes $\langle q, a, q' \rangle$ in R and $v' = o(q')$,
    - if $match(v, v')$, then $\langle M, q' \rangle$ is the successor to $\langle M, q \rangle$, extending the tree;
    - if not, then mark $\langle M, q \rangle$ as inconsistent.
2. Otherwise, M^C does not include $\langle q, a, q' \rangle$ in R. Let M' be M extended with a new distinctive state symbol q' and the assertions $v = o(q')$ and $\langle q, a, q' \rangle$. Consider the $k \geq 0$ dstates q_j in M with $v_j = o(q_j)$, such that $match(v_j, v)$. Then $\langle M, q \rangle$ has $k+1$ successors:
    - $\langle M'_j, q' \rangle$ for $1 \leq j \leq k$, where M'_j is M' extended with the assertion $q' = q_j$.
    - $\langle M'_{k+1}, q' \rangle$, where M'_{k+1} is M' extended with the k assertions that $q' \neq q_j$, for $1 \leq j \leq k$.
3. Mark a new successor map inconsistent if it violates the axioms of topological maps.
4. Define a preference order on the consistent maps at the leaves of the tree.

In the Basic SSH:

$M = M^T$.
A view is a simple symbol.
$match(v, v')$ iff $v = v'$.
Both $a \in Turns$ and $a \in Travels$ can reach step 2 and cause a branch.
Preference order from prioritized circumscription policy [Remolina and Kuipers, 2004].

In the Hybrid SSH:

$M = \langle M^T, M^P \rangle$.
A view is a structure $\langle S_p, \tilde{\pi}^d \rangle$, where $p = place(q)$, consisting of a local topology and the directed local-path the robot arrived upon.
$match(v, v')$ iff there exists an isomorphism $\phi : S_p \rightarrow S'$ where $\phi(q) = q'$.
Only $a \in Travels$ can reach step 2 and cause a branch.
Future work: Preference order from map probabilities (Section 10.1.2).

Algorithm 8.1: *Building the tree of topological maps in the HSSH.* This describes the algorithm for building a tree of all possible topological consistent with a sequence of actions and observations at discrete places. The different instantiations. Note that the HSSH algorithm is identical to the basic algorithm from Algorithm 3.1. The difference is in the well-defined views and in the branching factor, both of which depend on the local topology abstraction.

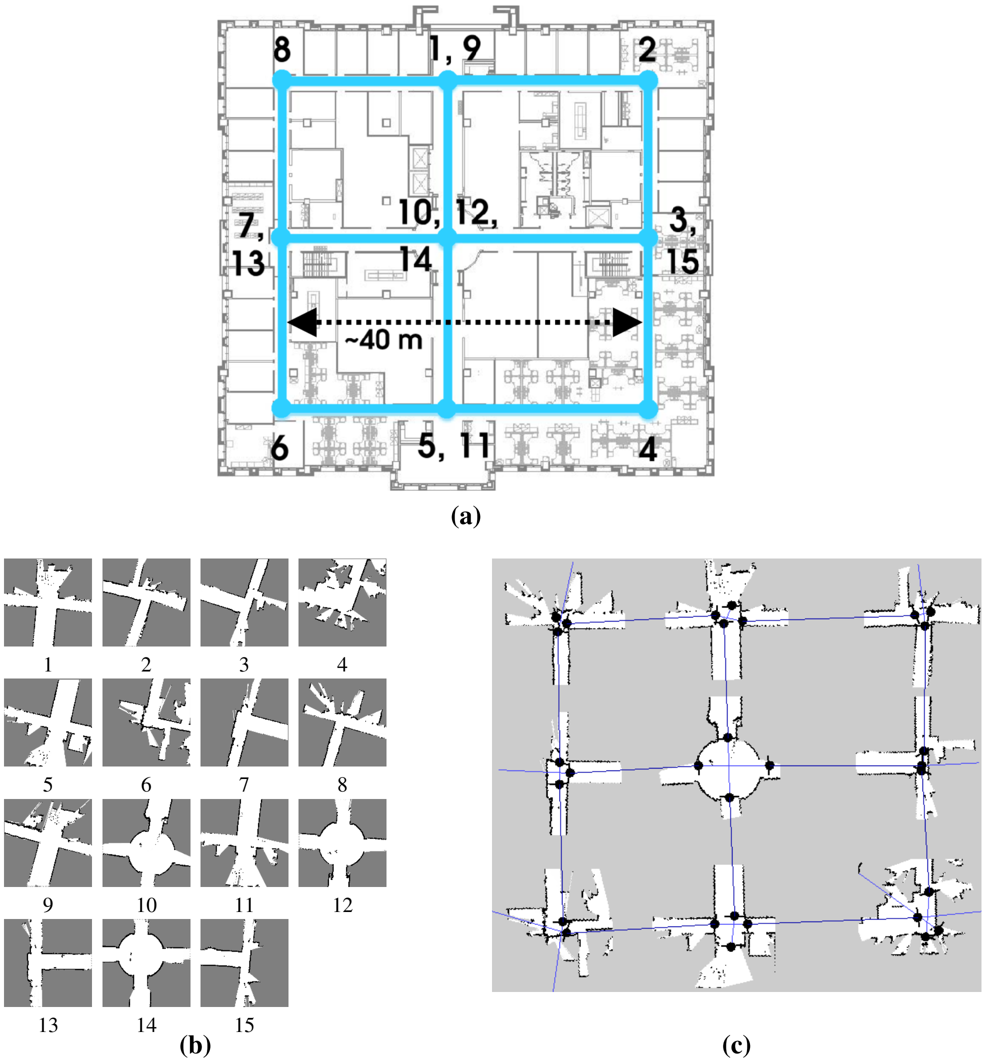
$S \rightarrow S'$ such that $\phi(q) = q'$. That is, from the perspectives of the specified dstates, the local topologies match.

As exploration progresses, the map M is extended with new information. For example, after an exploration step that closes a loop in the map, the resulting map M' is M extended with a new dstate q' and assertions $\langle q, a, q' \rangle$, $v = o(q')$, and $q' = q_j$. A new version of $M^C = \langle Q, A, V, S, o \rangle$ is created, and the implications of the loop-closing assertion $q' = q_j$ propagate through new versions of M^T and M^P to unify place and path labels as necessary. Because we are matching complete local topologies in the HSSH, the tree of maps only branches on travel actions. Turn actions are already fully described by the large-scale star.

8.3 Topological Mapping Example

We applied an implementation of the Hybrid SSH map-builder to an exploration of an office environment with multiple nested large loops. This office had a large number of cubicles and office doorways. To respect student and faculty privacy, we prune the Voronoi skeleton so that Voronoi branches, thus gateways, were defined only for large hallway intersections, not at doorways or cubicle openings. The environment, as defined by the robot, contained 6 paths and 9 places with 4 distinct local topologies. Figure 8.2 shows the exploration route as a sequence of place visits, the sequence of LPMs observed at successive place neighborhoods, and the unique simplest topological map that resulted from the mapping algorithm, with LPMs overlaid at corresponding places in the correct topological map.

After an exploration consisting of 14 travel actions, the topological mapper finds 83 possible configurations of the environment that are consistent with the observed local topologies and the topological axioms—that is there exist 83 leaves in the tree of maps. The prioritized circumscription [Remolina and Kuipers, 2004] on this set of maps produces 4 minimal models. All but one of these can be eliminated with further exploration or by simply matching LPMs using the alignments specified by the four minimal maps. This final map model is the correct topological representation of the environment.



(a)

(b)

(c)

Figure 8.2: *Finding the topological map of an environment with multiple nested loops.* In the CAD drawing **(a)**, we show the path traveled between places in the environment. We enumerate the order of places in the exploration taken by the robot. (This exploration trace was also used for Figure 2.1(d).) In **(b)**, we show the LPMs created at the places during the travel. We specifically tuned the gateway algorithm to ignore open office doors and cubicle openings to ensure places only at hallway intersections. The stars generated from these LPMs are used to search through the space of consistent topological maps. In **(c)**, we show the unique topological map generated after matching local stars and LPMs. The map is overlaid with the LPMs generated at the places, with the gateways, and with the connections between gateways which lie on the same path.

If we assume planarity of the environment, we can use a more sophisticated version of the topological map-building algorithm [Savelli and Kuipers, 2004] that rules out many more models as inconsistent. Here, there are only 46 consistent configurations of the exploration experience, and the circumscription policy produces a single minimal model, which is the correct topological map of the environment (Figure 8.2(c)). Our current implementation can build the complete tree of maps for this exploration trace and determine the unique minimal map of this environment in $\sim$200 ms on the robot's Pentium III 450 MHz processor. The results presented on this office environment would be unchanged if the path segments were longer or even very convoluted, as the number of places and paths would not change. Additionally, the tree of maps ensures the correct map is never discarded.

The complexity of the map computation is the following. Let n be the number of poses in the exploration trajectory; let m ($m \ll n$) be the number of topological places; let k ($k < m$) be the maximum number of places matching an observed view; and let l be the maximum number of directed local-paths in the local topologies (often $l \leq 4$). For example, for the environment in Figure 8.2, $n \approx 7300$, $m = 9$, $k = 4$, and $l = 4$. The maximum branching factor in the tree of maps is $k+1$. Branches only occur when the robot travels between two connected places for the first time, which can only happen at most $ml/2$ times. This means the size of the tree of maps is $O(k^m)$; thus, computing the tree of maps is exponential in m (not in n). The exponent decreases by at least a factor of 3 compared with the basic SSH version due to branching only on travels, not on turns, and matching local topologies of places.[1] Savelli and Kuipers [2004] also show that the planarity constraint gives an additional improvement in the branching factor k by rejecting many loop-closing hypotheses.[2]

---

[1] There are at most $ml/2$ unique travel actions, and there are at most l turns at each of the m places; thus, in the worst case environment, we have ml turns and $ml/2$ travels, resulting in $3ml/2$ actions in the basic SSH.

[2] Savelli and Kuipers [2004] also point out that for each map m_i in the tree to be expanded, the reduction of the branching factor k_i due to the planarity constraint is proportional to the number of closed loops already present in m_i. In other words, "the more loops [that] have been closed, the more topologically compact the map must be, and therefore the fewer ways there are to close new loops while preserving planarity," which reduces the branching factor further.

8.4 Levels of Spatial and Temporal Granularity

At this point, we summarize the three different levels of granularity, with different ontologies, that we are using to describe space and time.

The agent's experience is a trajectory through the environment. At the SSH Control Level and in the LPM, the trajectory is represented using a fine-grained representation for time t, pose x, motor signal u, and sensory image z. These are used both for control laws, and for simultaneous localization and mapping to build the LPM. Expanding Figure 4.1, the agent's exploration experience is described by

$$\begin{tikzcd}
 & \cdots & u_{t-1} \arrow[d] & u_t \arrow[d] & u_{t+1} \arrow[d] & \cdots & u_N \arrow[d] \\
x_0 \arrow[r] \arrow[d] & \cdots \arrow[r] & x_{t-1} \arrow[r] \arrow[d] & x_t \arrow[r] \arrow[d] & x_{t+1} \arrow[r] \arrow[d] & \cdots \arrow[r] & x_N \arrow[d] \\
z_0 & \cdots & z_{t-1} & z_t & z_{t+1} & \cdots & z_N
\end{tikzcd}$$

At the SSH Causal Level (which is part of the topological map), exploration experience is described by an alternating sequence of actions and distinctive states, with each distinctive state associated with a view.

$$\begin{tikzcd}
q_0 \arrow[d, dash] & a_1 & q_1 \arrow[d, dash] & a_2 & q_2 \arrow[d, dash] & \cdots & q_{n-1} \arrow[d, dash] & a_n & q_n \arrow[d, dash] \\
v_0 & & v_1 & & v_2 & \cdots & v_{n-1} & & v_n
\end{tikzcd}$$

In both the basic and hybrid versions of the SSH, distinctive states q correspond to being at a place, facing along a directed path. In the basic SSH, the distinctive states q are grounded by isolated distinctive states $\bar{x}$ where hill-climbing control laws terminate. In the Hybrid SSH, dstates are grounded by a directed local-path extracted from the LPM of a place neighborhood.

At the SSH Topological Level, a particular place p_j can correspond to several distinctive states, say q_{i-1} and q_i and the turn action a_i between them. A travel action a_{i+1} from q_i at p_j to q_{i+1} at a different place p_{j+1} can be used to infer the displacement λ_{j+1}, which is the pose of place p_{j+1} in the frame of reference of place p_j. This lets us abstract the sequence of distinctive states and actions to an alternating sequence of place p_j and displacements λ_j.

$$p_0 \quad \lambda_1 \quad p_1 \quad \lambda_2 \quad p_2 \quad \cdots \quad p_{m-1} \quad \lambda_m \quad p_m$$

As described in Chapter 9 (illustrated by Figure 9.1), in order to define the λ_i, each place neighborhood must have its own frame of reference and we must select a set of distinguished time-points $0 \leq t_0 < t_1 < \cdots \leq t_N = N$ such that adjacent time-points belong to different place neighborhoods, and the pose x_{t_i} at each time-point t_i can be unambiguously localized in its place neighborhood. To fit this into the SSH causal framework, we select distinguished time-points at the termination of each travel action: in the basic SSH, this is after hill-climbing terminates, and in the hybrid SSH, this is after a place is detected. In the Hybrid SSH, the dividing poses are near the incoming gateways in place neighborhoods. The net effect of the turn and travel actions between these dividing points are used to estimate the displacements λ_i between the frames of reference of adjacent place neighborhoods connected by path segments.

Chapter 9

Drawing Maps: The HSSH Global Metrical Level

> *A map is not the territory it represents, but if correct, it has a similar structure to the territory, which accounts for its usefulness*
>
> Alfred Korzybski, "A Non-Aristotelian System and its Necessity for Rigour in Mathematics and Physics" (1931)

This chapter details the process of building a global metrical map of an environment from a symbolic, global topological map. The topological map identifies a discrete set of places, each with its own local metrical map within its own frame of reference. The topological map also encodes decisions about how loops are closed and which aliased local neighborhoods represent the same places. The global metrical map is built on the structural skeleton provided by the topological map [Modayil, Beeson, and Kuipers, 2004]. The steps in building the global metrical map are: (1) describe the *displacements* $\lambda = \{\lambda_i\}$, each describing the change in pose from one place neighborhood to the next in the frame of reference of the first; (2) describe the *layout* $\chi = \{\chi_p\}$, specifying the poses of places in a global frame of reference; (3) describe the *trajectory* $x = \{x_t\}$ of robot poses within the global frame; and (4) create the *global map* m^* from sensor readings given the trajectory.

9.1 Terminology

The global topology τ, used below, consists of the set $M^P = \{\langle p, m_p, S_p, \psi_p \rangle : p \in P\}$ of places with their local information, the set of distinguished time-points $0 \leq t_0 < t_1 < \cdots < t_n \leq N$ that divide the fine-grained sequence of exploration experience into segments corresponding to travel between adjacent place neighborhoods, and the relation $place(t_i) = p_j$ between them. It is convenient to relabel the variables x, z, and u, defining $x_{i,j} \equiv x_{t_i+j}$. At each distinguished time-point t_i, where $place(t_i) = p_j \in P$ and $place(t_{i+1}) \neq place(t_i)$, the agent is localized in the local metrical map m_{p_j}.

Much of our metrical inference consists of defining an appropriate set of reference frames, and estimating the values of local and non-local metrical quantities. Many of these concepts can be simply understood by examining Figure 9.1.

$[x]_p$	The coordinates of the pose x in the frame of reference of place p.
O_p	The pose x such that $[x]_p = (0,0,0)$.
$L_i \equiv [x_{t_i}]_{place(t_i)}$	The coordinates of the pose x_{t_i} in the reference frame of $place(t_i)$.
$\tilde{m}_i$	The scrolling map that models the agent's surroundings between distinctive time-points t_i and t_{i+1}. The map's origin is defined as the agent's pose at time t_i. That is, $O_{\tilde{m}_i} = x_{t_i}$.
$\lambda_i \equiv [O_{place(t_i)}]_{place(t_{i-1})}$	The location of $O_{place(t_i)}$ in the reference frame of $place(t_{i-1})$, estimated using the experience from t_{i-1} to t_i.
$\chi_p \equiv [O_p]_{m^*}$	The pose of O_p in the global reference frame of m^*.
m^*	The global metrical map.

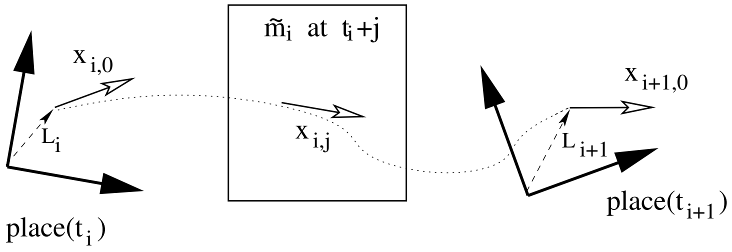


Figure 9.1: *Defining local frames of reference.* The agent creates the local scrolling map $\tilde{m}_i$ when traveling between places. The agent's poses at the distinguished time-points t_i and t_{i+1} are $L_i = [x_{i,0}]_{place(t_i)}$ and $L_{i+1} = [x_{i+1,0}]_{place(t_{i+1})}$. The displacement between the two place frames of reference is $\lambda_{i+1} = L_i \oplus [x_{i+1,0}]_{\tilde{m}_i} \oplus (\ominus L_{i+1})$.

9.2 The Theory of the Global Metrical Map

To build a global metrical map m^*, we want to find the maximum-likelihood path the robot traveled, using the topological skeleton in addition to odometry. As discussed in Section 4.1, the joint probability of the pose history x and the global map m^* can be decomposed as

$$P(x, m^*|z,u) = P(m^*|x,z,u) \cdot P(x|z,u)$$

by the chain rule for probabilities. This decomposition is valuable since $P(m^*|x,z,u)$ (map-building given accurate localization) can be computed analytically and incrementally for popular map types, so we can focus our attention on $P(x|z,u)$ (pose estimation).

To include the effect of possible global topologies τ on pose estimation, we marginalize over the space of topologies. If we assume that the correct global topology $\bar{\tau}$ has been identified, only one topological hypothesis $\tau = \bar{\tau}$ has nonzero probability.

$$\begin{aligned} P(x|z,u) &= \sum_{\tau} P(x|z,u,\tau) \cdot P(\tau|z,u) \\ &= P(x|z,u,\bar{\tau}) \end{aligned}$$

On the other hand, suppose there are multiple topologies τ with significantly non-zero values of $P(\tau|z,u)$. While the weighted sum provides a mathematically correct characterization of the probability distribution $P(x|z,u)$, it can easily lead to a nonsensical metrical map due to the dramatic qualitative impact of topological structure on the metrical map. Thus the summation should be regarded as describing a disjunction over topological maps, with $P(\tau|z,u)$ being the likelihood of each map. This is exactly the tree of possible topological maps we have already constructed. Therefore, even in the case where there are multiple plausible topological maps, we will construct global metrical maps for each one individually.

Given a particular topology $\bar{\tau}$, we can marginalize over the poses of all topological places $\chi = \chi_i$ and their estimated displacements λ.

$$P(x|z,u,\bar{\tau}) = \int\int P(x|\chi,\lambda,z,u,\bar{\tau}) \cdot P(\chi|\lambda,z,u,\bar{\tau}) \cdot P(\lambda|z,u,\bar{\tau})\, d\lambda\, d\chi$$

Because x is conditionally independent of λ given χ, and χ is conditionally independent of z,u given λ, we can simplify this equation.

$$P(x|z,u,\bar{\tau}) = \int P(x|\chi,z,u,\bar{\tau}) \int P(\chi|\lambda,\bar{\tau})\, P(\lambda|z,u,\bar{\tau})\, d\lambda d\chi$$

We divide this equation into simpler components, defining the following functions representing probability distributions over their arguments.[1]

$$\begin{aligned}
F(\lambda) &= P(\lambda|z,u,\bar{\tau}) \\
G(\chi) &= \int P(\chi|\lambda,\bar{\tau})\, F(\lambda)\, d\lambda \\
H(x) &= \int P(x|\chi,z,u,\bar{\tau})\, G(\chi)\, d\chi
\end{aligned}$$

Thus, we use the topological map $\bar{\tau}$ to factor the localization term $P(x|z,u) = H(x)$ into three separate probability distributions: place-to-place displacements $F(\lambda)$ derived from local metrical maps; the metrical layout $G(\chi)$ of places in the global topological map;

---

[1] We assume that there is no opportunity for confusion between these probability functions F, G, and H, and the dynamical system functions F, G, and H_i used in Section 3.2.1.

and the global metrical layout $H(x)$ of the robot's pose trajectory. Finally, we can combine the pose trajectory with $P(m^*|x,z,u)$ to define the joint distribution $P(x,m^*|z,u)$.

9.3 Global Mapping Example

Here we detail each step of creating the global map and discuss our current implementation, which runs offline. Figures 9.2 and 9.3 demonstrate the stages of creating an accurate global metrical map of a large, complex environment using these methods.

9.3.1 Estimating $F(\lambda)$

Given the topology τ, we can compute $F(\lambda)$. Each λ_i corresponds to a single experience of a path segment. Since closing large loops is not a problem when considering a single path segment, traditional SLAM methods may be employed to estimate $F(\lambda)$ by decoupling it into a set of independent probabilities.

$$\begin{aligned} D_i &= z_{i,0},\ldots,z_{i,n_i},u_{i,1},\ldots,u_{i+1,0} \\ F_i(\lambda_i) &= P(\lambda_i|D_{i-1},L_{i-1},L_i) \\ F(\lambda) &= \prod_{i=1}^{n} F_i(\lambda_i) \end{aligned}$$

See Figure 9.1 to understand L. Our current implementation is an incremental maximum-likelihood method [Fox et al., 1999], modeling each $F_i(\lambda_i)$ as a Gaussian.

Using the notation of the compounding operator [Smith et al., 1990], we compute the distribution of λ_i by composing three uncertain vectors: the vector L_{i-1} from $O_{place(t_{i-1})}$ to $x_{i-1,0}$; the vector $[x_{i,0}]_{\tilde{m}_{i-1}}$ from $x_{i-1,0}$ to $x_{i,0}$; and finally the vector $-L_i$ from $x_{i,0}$ to $O_{place(t_i)}$.[2]

$$F_i(\lambda_i) = P(\lambda_i = (L_{i-1} \oplus [x_{i,0}]_{\tilde{m}_{i-1}} \oplus (\ominus L_i)))$$

---

[2] Given two poses a and b, we write $[b]_a$ for the coordinates of b in the frame where a defines the origin [Smith et al., 1990]. Then, $[c]_a = [b]_a \oplus [c]_b$. The inverse operator is $[b]_a = \ominus[a]_b$.

The essential connection is that the pose $x_{i,0}$ at the end of a path-segment is described in the frame of reference of place p_{i-1} by the expression $L_{i-1} \oplus [x_{i,0}]_{\tilde{m}_{i-1}}$, and simultaneously in the frame of reference of place p_i by L_i.

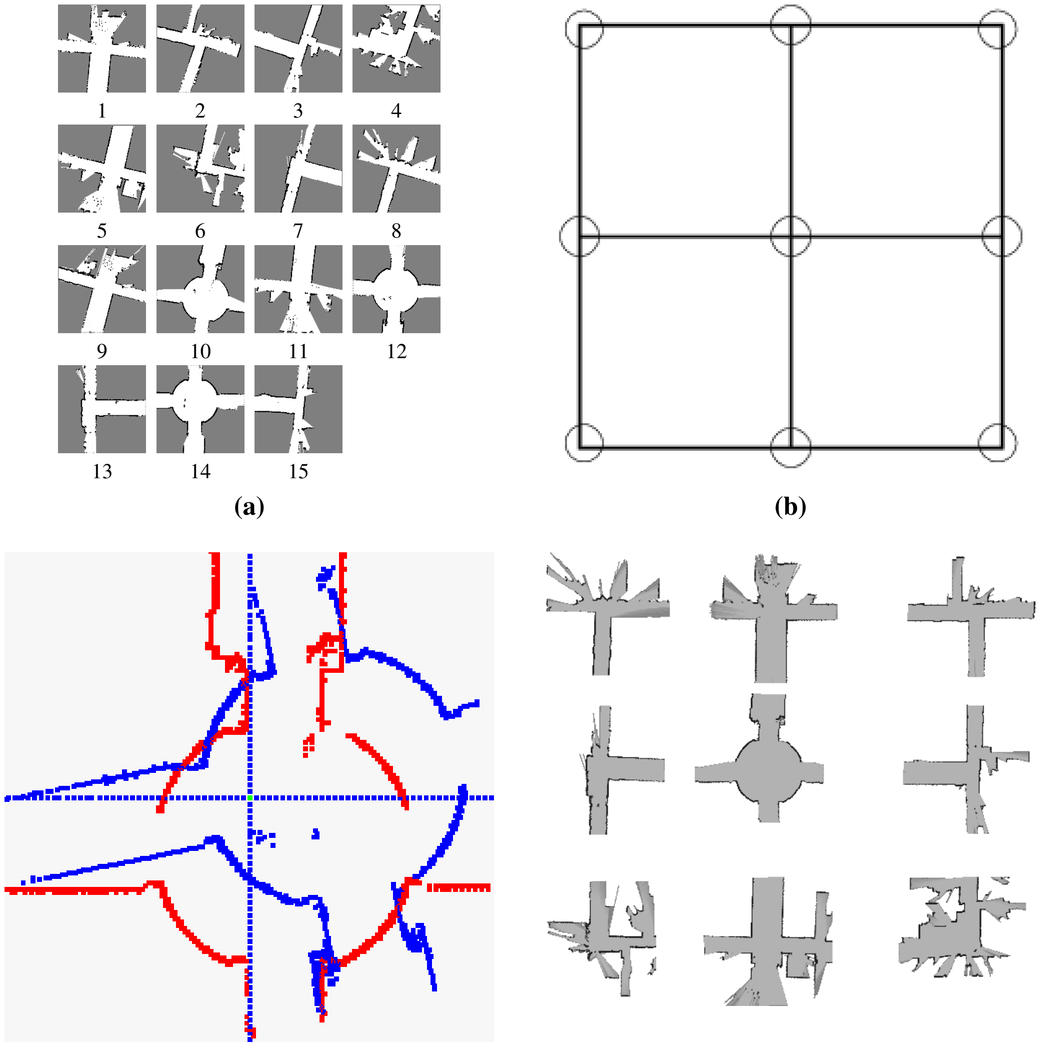
The problem of estimating $[x_{i,0}]_{\tilde{m}_{i-1}}$ is relatively simple along individual path segments, since loops cannot be involved. The more difficult problem arises from determining $\ominus L_i$ after a loop closure. Here we need to align the map m_{p_i} with a previously stored map m_{p_h} in order to determine $[O_{p_h}]_{p_i}$, which allows us to solve $\ominus L_i$. Matching maps can be expensive and can lead to false positives due to local minima (e.g., two LPMs of a $+$ intersection can be matched four ways). To eliminate this problem, we first align the LPMs based on the locations of corresponding gateways, consistent with m_h, S_h, ψ_h and m_i, S_i, ψ_i, before refining the alignment using the obstacles and free space of the LPMs. (Figure 9.2(c) omits this gateway alignment step in order to better illustrate the process of LPM alignment.)

9.3.2 Estimating $G(\chi)$

The layout $\chi = \{\chi_p\}$ represents the poses of the places in the topological map, with respect to the frame of reference of the global metrical map m^*. $G(\chi)$ is a probability density function over possible layouts χ. Among other things, it reflects the distortion in the place layout due to a loop-closing hypothesis, compared with the observed displacements λ.

Given the topological map, which specifies the data association between observations and places, we can evaluate $G(\chi)$ for an arbitrary distribution of $F(\lambda)$. For a particular value of χ, $P(\chi|\lambda,\bar{\tau})$ will only be non-zero for a single value of λ, namely when each $\lambda_i = (\ominus\chi_{place(t_{i-1})}) \oplus \chi_{place(t_i)}$. Hence, $P(\chi|\lambda,\bar{\tau})$ is a Dirac delta function, which gives us a simple expression for $G(\chi)$.

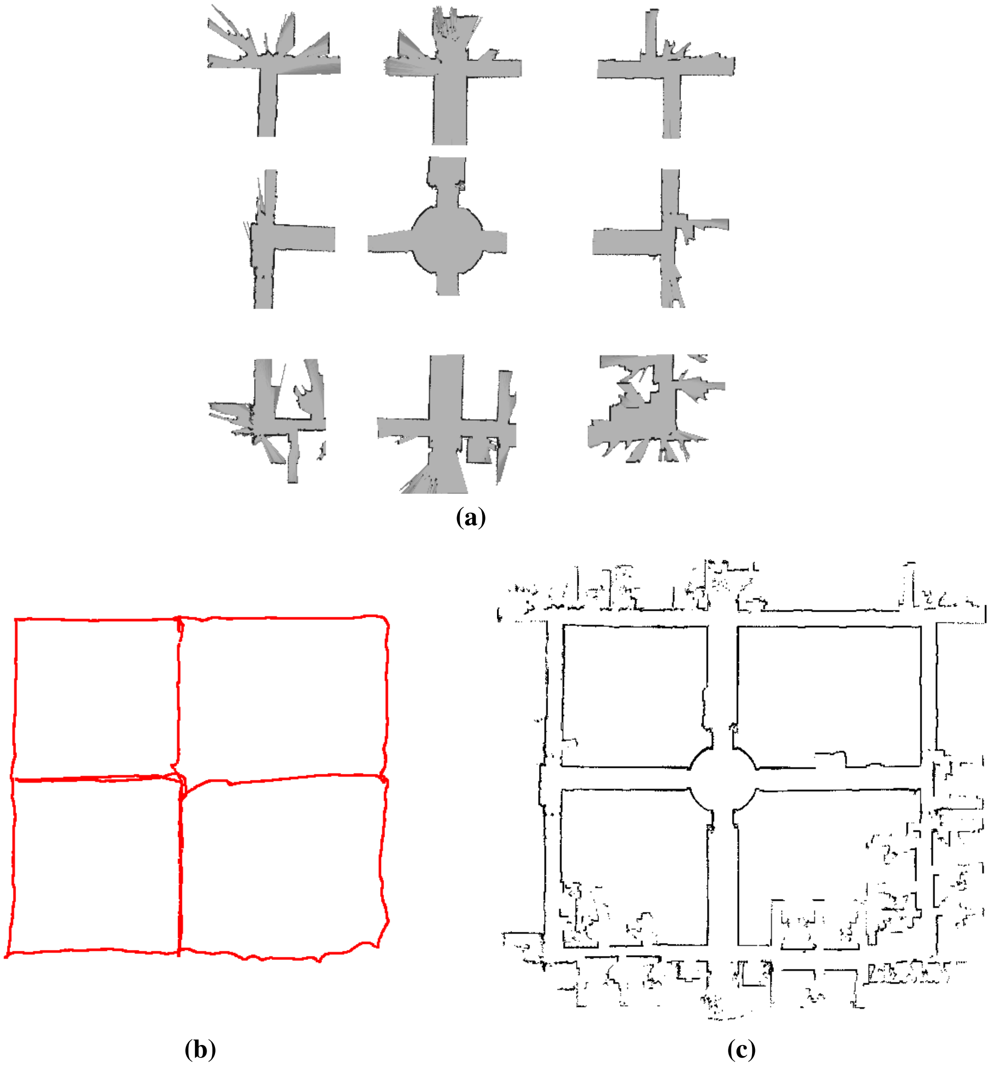
$$\begin{aligned} G(\chi) &= \int P(\chi|\lambda,\bar{\tau})\, F(\lambda)\, d\lambda \\ &= \prod_{i=1}^{n} F_i((\ominus\chi_{place(t_{i-1})}) \oplus \chi_{place(t_i)}) \end{aligned}$$



(a) **(b)**

(c) **(d)**

Figure 9.2: *Finding the global place layout.* **(a)** The sequence of local place maps m_p experienced. **(b)** The unique topological map consistent with topological and planarity constraints. **(c)** We determine λ_i for loop closures by finding the offset between the current pose and the place origin (defined on the initial place visit). **(d)** The layout χ derived from the topological map and the place-to-place displacements λ.



(a)

(b)

(c)

Figure 9.3: *Creating a global metrical map.* Figure 9.2 described the process of obtaining the global place layout χ, shown in **(a)**. Once this is known, we can quickly compute an estimate of the global metrical map. **(b)** The pose trajectory $x(t)$ anchored at points where the robot is localized in place neighborhoods in the layout χ. **(c)** Given the localized pose trajectory $x(t)$ in the global frame of reference, the global metrical map m^* is created accurately and efficiently. Compare with Figure 2.1(d).

When $F(\lambda)$ is represented as a Gaussian, an Extended Kalman Filter (EKF) is a simple way to approximate $G(\chi)$. The idea is to consider place p to be a landmark with pose χ_p. These landmarks are observed one at a time, linked by actions λ_i. This is essentially the classic approach of Smith et al. [1990]. Given Gaussian uncertainty along each action u_i connecting the n robot poses, along with constraints that give Gaussian uncertainty between poses taken from multiple visits to the same place (to associate poses after loop closures), we can solve for $H(x)$ in time $O(n \log n)$ using the sparse matrix methods of Konolige [2004]. However, often we may only want $G(\chi)$, which can be computed in $O(m \log m)$ time for m places, where $m \ll n$.

In our current implementation, we utilize a hill-climbing search to quickly converge to a local maximum of $G(\chi)$ (Figure 9.2(d)). The Levenberg-Marquardt algorithm for non-linear optimization [Press et al., 1992] treats the λ_i as "springs" between the poses of the places p_k in χ, and relaxes their configuration to reach a local minimum-energy configuration. Efficient estimations of this non-linear optimization also exist [Olson et al., 2006]. A good initial layout χ for this hill-climbing search can be derived from the displacements λ_i, which represent SLAM-corrected odometry from the scrolling map. We use the term $\hat{\chi}$ to denote the maximum-likelihood estimate of $G(\chi)$.

9.3.3 Estimating $H(x)$

An extended Kalman filter can be used to estimate $H(x)$ using $G(\chi)$ and individual pose covariances from the experienced trajectory. Alternatively, if accurate pose covariances are not available, a simple method can estimate the maximum-likelihood trajectory through the environment. We calculate the independent trajectory $H_i(x)$ along each path segment, as each place location is fully determined by a global layout χ. In most cases, there will be some discrepancy between the measured distance λ_i along the path segment and the fixed distance between the places in χ. We scale the experienced motion along the path segment to fit the global path segment distance. This process is similar to methods of distributing odometry error after closing a loop in a global metrical map [Thrun et al., 2000a].

For each trajectory between adjacent places p_i and p_{i+1}, we transform the relative, incremental displacements $\Delta(\mathsf{x}, \mathsf{y}, \theta)$ from the pose estimates in the scrolling LPM $\tilde{m}_i$ into relative displacements $\xi_{i,j}$ in the global frame of reference. This uses a simple affine transformation T.

$$\begin{aligned} [x_{i,0}]_{m^*} &\equiv T([x_{i,0}]_{\tilde{m}_i}) \\ &= \chi_{place_{t_i}} \oplus L_i \\ \xi_{i,j} &\equiv [x_{i,j}]_{m^*} - [x_{i,j-1}]_{m^*} \end{aligned}$$

We compute the maximum-likelihood trajectory by satisfying the constraint that the travel experience between the places must fit the globally defined distance between the places. First, we find the discrepancy δ between the measured trajectory and the distance given by the global place layout.

$$\begin{aligned} [x_{i+1,0}]_{m^*} &= \chi_{place_{t_{i+1}}} \oplus L_{i+1} = [x_{i,n_i}]_{m^*} \\ \delta &= [x_{i+1,0}]_{m^*} - [x_{i,0}]_{m^*} - \sum_{j=1}^{n_i} \xi_{i,j} \end{aligned}$$

We assume that the global displacements are distributed with Gaussian uncertainties. The means, μ, are simply the calculated displacements above, and we assume each Λ_j is a diagonal covariance matrix that is proportional to the absolute value of the mean (for each component in the pose vector $\xi_{i,j}$).

$$\xi_{i,j} \sim \mathcal{N}(\mu_{i,j}, \Lambda_{i,j})$$

We then calculate the maximum-likelihood displacements in the global frame of reference for each component c in x, y, θ (the pose dimensions).

$$\hat{\xi}^{c}_{i,j} = \mu^{c}_{i,j} + \frac{|\mu^{c}_{i,j}|}{\sum_{h=1}^{n_i} |\mu^{c}_{i,h}|} \cdot \delta^{c}$$

From our maximum-likelihood displacements ($\hat{\xi}$), we can estimate the trajectory x in the global frame of reference.

$$[x_{i,j}]_{m^*} = [x_{i,0}]_{m^*} + \sum_{h=1}^{j} \hat{\xi}_{i,h}.$$

9.3.4 Creating a map m^*

The maximum-likelihood trajectory above can be used as a starting trajectory for gradient descent methods to align the pose positions with map estimates to converge upon a locally optimal map [Lu and Milios, 1997; Thrun et al., 2000a]. A more principled approach is to run a Rao-Blackwellized particle-filtering algorithm, using the maximum-likelihood trajectory as the mean of a proposal distribution: $P(x,m|z,u) = P(m|x,z,u) \cdot H(x)$. However, we have found that in practice the x values defined by the above scaling method adequately approximate the mode of the posterior [Modayil, Beeson, and Kuipers, 2004]; thus the global map can be built by projecting the recorded range measurements from poses in the new global coordinates. The final map produced from the topological skeleton is shown in Figure 9.3(c). Compare this to Figure 2.1(d) to see the improved map.

Chapter 10

Conclusion

> *Many places you would like to see are just off the map and many things you want to know are just out of sight or a little beyond your reach. But someday you'll reach them all, for what you learn today, for no reason at all, will help you discover all the wonderful secrets of tomorrow.*
>
> Norton Juster, *The Phantom Tollbooth* (1961)

This chapter concludes the dissertation with various discussions about related future work and a summary of the thesis. This dissertation describes the Hybrid Spatial Semantic Hierarchy (HSSH) in detail, formally describing the interaction between small-scale and large-scale representations of space, and gives preliminary results at all four levels of the hybrid extension to the SSH: improved metrical localization and mapping in local regions, robust place detection and classification, more efficient topological map-building, and high-precision global metrical mapping from topological skeletons.

As noted throughout this dissertation, the current implementation can be improved upon at all four of the abstraction levels. In addition, we would like extend the human-robot interaction studies to include experiments that test the entire HSSH interface on a large population of subjects across various navigation domains. Finally, we plan to relax the assumptions of deterministic actions, allowing rare undetected places or misclassified places to be handled gracefully by our logical topological map-building paradigm.

10.1 Future Work

10.1.1 The Local Metrical and Topology Levels

The current implementation of the local perceptual map (LPM) uses planar lidar sensors to build a metrically accurate model of the local surround. This is a typical SLAM formulation. There has been recent work in vision-based SLAM [Sim and Little, 2006], detecting paths from monocular vision [Posner et al., 2007], and fusing laser data with stereo-vision data in occupancy grids [Murarka et al., 2006]. Advancements in these fields should make it possible to create LPMs for sidewalk networks, for streets, and for non-planar environments, increasing the navigational abilities of a robot using the HSSH.

LPMs that can classify obstacle properties (architecture, doorway, furniture, pedestrians, pavement, gravel) as well as annotating regions based on their safety properties (overhang, drop-off, ramp) should significantly improve control. Annotated LPMs should also prove useful at the Local Topology Level, for reasoning about non-structural paths (e.g., painted crosswalks) and discriminating between places with aliased local topologies. As pointed out by Adams et al. [2000], furniture and other obstacles that humans abstract away may cause the robot to mark gateways at locations where humans do not.

Similarly, one drawback of the current LPM implementation is that we cannot detect closed doors with range sensors. Currently, if the robot observes a door being opened or closed, the doorway is seen as free space in the copy of the LPM used for finding gateways [Modayil and Kuipers, 2004]. This a gateway to be generated for the doorway. However, if the door remains closed as the robot moves past it, there is no way for a robot utilizing only range-sensing devices to distinguish the door from the wall. Schröter [2006] investigated a visual door detection algorithm that he used to define gateways. Incorporating a robust visual door detection algorithm into the either the LPM or directly into the local topology abstraction would allow our robot to detect places in front of each doorway and generate the proper local topology.

Achieving this level of robustness is one of our goals; however, it raises questions about whether doorways represent a different type of gateway than the types of gateways at hallway or sidewalk intersections. Doorways seem to be “conditional gateways” while hallway intersections seem to be “unconditional gateways”. This may give rise to conditional versus unconditional places: e.g., when following directions through a building, places in front of doors are usually not considered; however when looking for a specific office, an agent may begin to look for doorways until the correct one is found. Similarly, there may be paths that a robot wheelchair might travel on (e.g., sidewalks), but there may exist other paths that it models but should not travel on (e.g., streets). (A robot car would have the reverse situation.) These are examples of the kinds of interesting work in robot navigation and cognitive map research that remain to be explored.

Another important improvement in classifying places should be relatively straightforward to implement in the near future. We should extend the current *star* representation of a local topology to better discriminative qualitatively different places that have similar topology. We have already discussed how using local topology is more descriptive than simply counting the number of paths leaving a place, or storing the area of a place neighborhood. The current local topology representation cannot distinguish between a T intersection at intersecting corridors and one created by a corridor leading into a room larger than the LPM (i.e., a coastal navigation or wall-following scenario). These two places share the same topology but have drastically different LPMs. Annotating the local topology with knowledge about the type of path (i.e., $\alpha : S \rightarrow \{\text{MIDLINE}, \text{LEFTWALL}, \text{RIGHTWALL}, \text{DEADEND}, \text{NONE}\}$), will allow us to keep deterministic place recognition, while reducing perceptual aliasing.

Other useful improvements we hope to add to the LPM implementation are: efficient planning algorithms (e.g., adaptive A* [Koenig and Likhachev, 2006]), high-fidelity control in tight spaces, dynamic obstacle tracking, action models that adapt to surface properties, and multi-core or GPU parallelism. Other useful improvements we hope to add to the local topology implementation are: efficient extended Voronoi graph implementations (i.e., a modified version of the dynamic brushfire algorithm [Kalra et al., 2006]), efficient generation of "virtual" obstacles in (concave) coastal navigation scenarios, robust gateway alignment algorithms, intelligent exploration of potential places, and semantic labeling of places. Additionally, improvements to the gateway algorithm (and to gateway alignment) should be verified using more sophisticated models of local environments. This includes better models of the types of LPM noise a real robot might experience. We also plan to stress-test gateways on more places, simple and complex, during evaluation. Perhaps this requires a large virtual reality world that contains pathological examples that are rare in university buildings.

Extending Places

The HSSH is a first attempt at integrating small-scale space into a framework for large-scale knowledge about the environment. By looking at decision points, we can begin to bootstrap a useful symbolic abstraction from local metrical models built by robots. We believe there is sometimes a need for larger, more complex places, defined by function or some other criteria than just gateways.

We plan to begin investigation on how to group metrically separated places together to form places at a higher-level topology. One good starting point is to group places in buildings by floor (adding in elevator control as non-deterministic actions along paths). Another level would be to group all places in a building (or network of buildings), using sensors to detect when the robot leaves a particular building, or moves outdoors.

Another related area is that of medium-size spaces that have a bounded structure but are larger than an LPM. A good example is a large research laboratory. Large rooms are often complex in shape, larger than our typical LPM size, yet still enclosed by walls, with doors (conditional gateways) defining the exits of the place. Defining where these scenarios exist and how to handle these scenarios (e.g., adapting the LPM size, clustering places defined by unconditional gateways, etc.) are all important tasks that remain to be examined.

10.1.2 The Global Topology and Metrical Levels

Currently, the tree of maps contains every topological map consistent with exploration experience and the topological axioms. This guarantees soundness, which is useful in the case where observations refute the current best map and the next best map must be identified. However, there remain two related problems that need to be addressed in future work. First is the need for a reliable method to identify the best candidate among a set of possible topological maps, given odometry and perceptual information [Ranganathan et al., 2006]. Second, is the need to reduce the tree of maps from a "breadth-first" search to a more focused search that tracks a small number of maps at a time.

In Section 8.3, we identified the "best" map as the simplest one based on a prioritized circumscription policy [Lifschitz, 1995] over the models generated by the non-monotonic theory of topological maps [Remolina and Kuipers, 2004]. This is sufficient for the example environment and exploration we have used throughout this dissertation. Although metrically large, this environment appears to have a simple topology; however, the simplicity of the topology is deceptive—there is quite a bit of non-local, topological (graph) symmetry as well as local perceptual aliasing of the places, both of which make map-building *much* more difficult. Nonetheless, with the addition of a planarity constraint [Savelli and Kuipers, 2004], the correct topology is identified as the unique simplest consistent topological map.

In Section 8.3, we also discussed the size of the tree in our example: 46 final maps consistent with the topological axioms. Savelli and Kuipers [2004] describe larger environments where extreme symmetry and aliasing cannot so easily be resolved by purely qualitative methods because the tree of maps grows too large to maintain in real-time. These are not entirely unrealistic examples, since large grid-structured neighborhoods in real cities provide opportunities for vast topological ambiguity [Lynch, 1960].[1] We would like to investigate methods for expanding a small set of maps at a time, comparing leaves at different levels in the tree to identify the best model that explains the robot's current experience: i.e., a "best-first" search approach.

One obvious improvement that will limit the number of map hypotheses is to perform active exploration, that occasionally exploits knowledge of the environment to eliminate entire branches from the tree of maps. Such strategies are similar to the localization procedures advocated by proponents of DFA-style maps [Kuipers and Byun, 1991; Dean et al., 1995; Rekleitis et al., 1999]. Dudek et al. [1991] propose an exploration algorithm that finds the correct topological structure in only $O(n^3)$ travel actions (n is the number of places), but this requires that the robot drop markers and backtrack to determine which loop-closing hypothesis was correct.

Outside of exploration strategies, we would like reduce the tree of maps by drawing on perceptual information currently unused in the topological map-building process. We should be able to use observational data to define weights on the tree of maps. These weights should allow us to have a quantitative ordering on the map hypotheses, and should allow best-first style expansion of the tree that focuses on a limited number of highly ranked candidates at a time, allowing the robot to map larger environments including those with large amounts of symmetry and perceptual aliasing.

Incorporating the Global Metrical Layout

The tree of maps is an exhaustively enumerated set of topological map hypotheses consistent with the axioms for topological maps and with the agent's exploration experience. The maps along a path from the root to the fringe of the tree are not mutually exclusive, but represent the growth of the topological map during exploration experience. Given the ex-

---

[1] Ranganathan and Dellaert [2005] show that because (in the worst case) the number of aliased places grows with the amount of exploration experience, the number of possible topological maps is given by Bell's number, which grows hyper-exponentially with the number of perceptually aliased places.

ploration experience from which the tree of maps was constructed, the maps $\langle M, q \rangle$ on the fringe of the tree are an exhaustive set of mutually exclusive topological map hypotheses.

A global topological map provides a structural skeleton, on which a global metrical map is built. The use of a topological skeleton helps prevent multi-modal distributions in the metrical uncertainty. At the same time, metrical inference about the layout of places in the global frame of reference should help discriminate among topological map hypotheses. Similar to ideas published by Hähnel et al. [2003b] and Ranganathan and Dellaert [2005], we should be able to compute beliefs for topological map hypotheses by interleaving the computation of the global place layout with the search for the global topological map.

We believe that we should be able to weight the leaves on the tree of maps by the likelihood of the global place layout for each map hypothesis: $P(\hat{\chi}|M,Y)$, where $Y = [q_0, \ldots, q_i]$. Obviously, there needs to be some discount for maps hypotheses that simply add a new place after each travel. Ranganathan and Dellaert [2005] utilize an odometry based penalty for places too close together in their Bayesian topological framework. Savelli [2005] discusses how, and how well, Bayesian probabilistic reasoning can express qualitative preferences of topological maps.

Improved Place Recognition

Another compelling idea is to weight the leaves on the tree of maps by the latest observation. One idea is to replace our current local topology matching with probabilistic matching, e.g., where a Y intersection might be perceived as a T intersection due to noise. Allowing non-deterministic place descriptions would make finding the correct global map even more challenging [Basye et al., 1995]; however, by first comparing the symbolic local topology, we forgo having to match high-resolution images or LPMs of the current place with *all* previously visited places.

If local topology matching and/or the global place layout has a high probability, map hypotheses could be scored by matching the stored small-scale place maps $m_p : p \in P$, or even by using a robust visual place recognition system. In previous work [Kuipers and Beeson, 2002], we examined learning discriminating features in lidar measurements from known places. Recently, similar work has been examined by others, investigating visual place recognition and even semantic labeling of places [Mozos et al., 2005; Tapus and Siegwart, 2005; Newman et al., 2006; Friedman et al., 2007].

Just as when considering the global layout, one important characteristic is that the algorithm must have a prior probability for the case where the robot is at a new place. For example, Cummins and Newman [2008] present a robust visual recognition system that gives a reasonable prior to visual observations based on the database of images used for training. Incorporating this visual place detection system into the map-building process is a likely candidate for extending the current Hybrid SSH implementation, and we expect by using the HSSH place detection we can show large improvements in properly detecting loop closures from vision.

10.2 Summary

We have presented a hybrid metrical/topological framework that processes information at both small-scale and large-scale abstractions. Our *Hybrid Spatial Semantic Hierarchy* is inspired by human cognitive maps; thus, it represents the environment using human-like concepts, such as places and paths, which support hierarchical navigation, human-robot interaction, and logical reasoning. Specifically, we focused on the problem of map-building—discussing how the HSSH builds metrical representations for local small-scale spaces, finds a topological map representing the qualitative structure of large-scale space, and constructs a metrical representation for large-scale space in a single global frame of reference by building on the skeleton provided by the topological map.

Unlike many robotic implementations that attempt to build a monolithic, Cartesian global metrical map, we propose an alternative approach that handles closing large loops by hypothesizing symbolic place matches. This ensures all possible loop closures are considered, not just ones where the robot, with accumulated odometry error, happens to be near some older portion of the map. The minimal topological map that results from large-scale exploration is sufficient for navigation and necessary for efficient planning, especially to rule out alternative topological structures during exploration.

The thrust of this dissertation has been to formally describe how concepts of large-scale space can be grounded in the robot's low-level observations. This problem has hindered topological map-building research, as it is an example of the hard AI problem of symbol grounding [Harnad, 1990]. Our innovation has been to utilize metrical approaches to model the immediate, local surround of the robot in order to ground gateways in small-scale space. Gateways provide the robot with local motion targets that facilitate control along paths. They also provide a local topology description of the local surround, useful for detecting and describing places and the paths that emanate from places.

Additionally, we detailed improvements to the traditional incremental localization schemes in order to ensure high-quality local metrical models. We demonstrated the robustness of local topology abstraction from gateways over various transformations and noise characteristics of the local metrical models. We validated the Local Topology and Local Metrical Levels of the HSSH with a human-robot interaction experiment for visually impaired drivers of a robot.

At the Global Topology and Global Metrical Levels of the HSSH, we provided insight into the computational benefits of using local topology when building the tree of possible topological maps, and we demonstrated that a global layout of places is easily achieved given a topological map hypothesis. We presented an implementation of HSSH global topological map-building within an environment with fairly large, nested loop closures. The results support our claims of efficient, online map-building in the presence of multiple loop closures. We also detailed an efficient algorithm for achieving a full global metrical map by filling in exploration experience along the path segments that connect places in the environment.

Bibliography

W. Adams, D. Perzanowski, and A. Schultz. Learning, storage, and use of spatial information in a robotics domain. In *Proceedings of the International Conference on Machine Learning Workshop on Machine Learning of Spatial Knowledge*, pages 23–27, Palo Alto, California, 2000.

V. Aginsky, C. Harris, R. Rensink, and J. Beusmans. Two strategies for learning a route in a driving simulator. *Journal of Environmental Psychology*, 17:317–331, 1997.

G. L. Allen, K. C. Kirasic, A. W. Siegel, and J. F. Herman. Developmental issues in cognitive mapping: The selection and utilization of environmental landmarks. *Child Development*, 50:1062–1070, 1979.

D. Angluin. On the complexity of minimum inference of regular sets. *Information and Control*, 39:337–350, 1978.

D. Appleyard. Why buildings are known: a predictive tool for architects and planners. *Environment and Behavior*, 1(2), December 1969.

D. Appleyard. Styles and methods of structuring a city. *Environment and Behavior*, 2(1), June 1970.

K. Basye, T. Dean, and L. P. Kaelbling. Learning dynamics: system identification for perceptually challenged agents. *Artificial Intelligence*, 72:139–171, 1995.

K. Basye, T. Dean, and J. S. Vitter. Coping with uncertainty in map learning. *Machine Learning*, 29(1):65–88, 1997.

P. Beeson. EVG-Thin software, 2006. URL `http://www.cs.utexas.edu/users/qr/software/evg-thin.html`.

P. Beeson, M. MacMahon, J. Modayil, J. Provost, F. Savelli, and B. Kuipers. Exploiting local perceptual models for topological map-building. In *Proceedings of the International Workshop on Reasoning with Uncertainty in Robotics (RUR)*, pages 15–22, Acapulco, Mexico, 2003.

P. Beeson, N. K. Jong, and B. Kuipers. Towards autonomous topological place detection using the extended Voronoi graph. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4373–4379, Barcelona, Spain, April 2005.

P. Beeson, A. Murarka, and B. Kuipers. Adapting proposal distributions for accurate, efficient mobile robot localization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 49–55, Orlando, Florida, 2006.

P. Beeson, M. MacMahon, J. Modayil, A. Murarka, B. Kuipers, and B. Stankiewicz. Integrating multiple representations of spatial knowledge for mapping, navigation, and communication. In *Proceedings of the Symposium on Interaction Challenges for Intelligent Assistants*, AAAI Spring Symposium Series, pages 1–9, Stanford, CA, 2007. AAAI Technical Report SS-07-04.

R. Biegler and R. G. M. Morris. Landmark stability: Studies exploring whether the perceived stability of the environment influences spatial representation. *Journal of Experimental Biology*, 199:187–193, 1996.

J.-L. Blanco, J.-A. Fernández-Madrigal, and J. González. Toward a unified bayesian approach to hybrid metric-topological SLAM. *IEEE Transactions on Robotics*, 24(2):259–270, April 2008.

J. Borenstein and L. Feng. Measurement and correction of systematic odometry errors in mobile robots. *IEEE Transactions on Robotics and Automation*, 12(6):869–880, December 1996.

J. Borenstein and Y. Koren. The Vector Field Histogram—fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, 1991.

M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller. An Atlas framework for scalable mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1899–1906, Taipei, Taiwan, 2003.

P. Buschka. *An Investigation of Hybrid Maps for Mobile Robots*. PhD thesis, Örebro University, 2005.

Y.-T. Byun. *Spatial Learning Mobile Robots with a Spatial Semantic Hierarchical Model*. PhD thesis, The University of Texas at Austin, 1990.

C.-H. Choi, J.-B. Song, W. Chung, and M. Kim. Topological map building based on thinning and its application to localization. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 552–557, Lausanne, Switzerland, 2002.

H. Choset and J. Burdick. Sensor-based exploration: the hierarchical generalized Voronoi graph. *International Journal of Robotics Research*, 19(2):96–125, 2000.

H. Choset and K. Nagatani. Topological simultaneous localization and mapping (SLAM): toward exact localization without explicit localization. *IEEE Transactions on Robotics and Automation*, 17(2):125–137, April 2001.

H. Choset, S. Walker, K. Eiamsa-Ard, and J. Burdick. Sensor-based exploration: incremental construction of the hierarchical generalized Voronoi graph. *International Journal of Robotics Research*, 19(2):125–148, 2000.

E. Chown. Making predictions in an uncertain world: Environmental structure and cognitive maps. *Adaptive Behavior*, 8(1):17–33, 1999.

E. Chown. Gateways: An approach to parsing spatial domains. In *Proceedings of the International Conference on Machine Learning Workshop on Machine Learning of Spatial Knowledge*, pages 1–6, Palo Alto, California, 2000.

E. Chown, S. Kaplan, and D. Kortenkamp. Prototypes, location, and associative networks (PLAN): Towards a unified theory of cognitive mapping. *Cognitive Science*, 19(1):1–51, 1995.

K. J. W. Craik. *The nature of exploration*. Cambridge University Press, London, 1943.

M. Cummins and P. Newman. FAB-MAP: Probabilistic localization and mapping in the space of appearance. *International Journal of Robotics Research*, 27(6):647–665, 2008.

M. Cummins and P. Newman. Probabilistic appearance based navigation and loop closing. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2042–2048, Rome, Italy, 2007.

T. Dean, D. Angluin, K. Basye, S. Engelson, L. Kaelbling, E. Kokkevis, and O. Maron. Inferring finite automata with stochastic output functions and an application to map learning. *Machine Learning*, 18(1):81–108, 1995.

B. N. Delaunay. Sur la sphère vide. *Bulletin of Academy of Sciences of the USSR*, 7: 793–800, 1934.

R. M. Downs and D. Stea. *Image and Environment*. Aldine Publishing Company, Chicago, 1973.

T. Duckett and U. Nehmzow. Exploration of unknown environments using a compass, topological map and neural network. In *Proceedings of the IEEE International Sympo-*

sium on Computational Intelligence in Robotics and Automation (CIRA), pages 312–317, Monterey, California, 1999.

T. Duckett and A. Saffiotti. Building globally consistent gridmaps from topologies. In *Proceedings of the International IFAC Symposium on Robot Control (SYROCO)*, pages 357–361, Vienna, Austria, 2000.

G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. Robotic exploration as graph construction. *IEEE Transactions on Robotics and Automation*, 7(6):859–865, 1991.

G. Dudek, P. Freedman, and S. Hadjres. Using local information in a non-local way for mapping graph-like worlds. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1639–1647, Chambéry, France, 1993.

A. Elfes. *Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation*. PhD thesis, Carnegie Mellon University, 1989.

A. Eliazar and R. Parr. DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1135–1142, Acapulco, Mexico, 2003.

A. I. Eliazar and R. Parr. Learning probabilistic motion models for mobile robots. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 69, page 32, Banff, Canada, 2004.

I. Filippov. OSRA: Optical structure recognition, v.0.9.9, 2007. URL `http://osra.sourceforge.net/`.

S. Fortune. Voronoi diagrams and Delaunay triangulations. In D.-Z. Du and F. Hwang, editors, *Computing in Euclidean Geometry*, volume 1 of *Lecture Notes Series on Computing*, pages 193–234. World Scientific, 1992.

D. Fox. Adapting the sample size in particle filters through KLD-sampling. *International Journal of Robotics Research*, 22(12):985–1003, 2003.

D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1), March 1997.

D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391–427, 1999.

S. Friedman, H. Pasula, and D. Fox. Voronoi random fields: Extracting the topological structure of indoor environments via place labeling. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2109–2114, Hyderabad, India, 2007.

N. Gale, R. G. Golledge, J. W. Pellegrino, and S. Doherty. The acquisition and integration of route knowledge in an unfamiliar neighborhood. *Journal of Environmental Psychology*, 10(1):3–25, 1990.

C. R. Gallistel. *The Organization of Learning*. MIT Press, Cambridge, Massachusetts, 1990.

T. Gladwin. *East is a Big Bird: Navigation and Logic on Puluwat Atoll*. Harvard University Press, Cambridge, Massachusetts, 1970.

E. M. Gold. Complexity of automaton identification from given sets. *Information and Control*, 37:302–320, 1978.

R. G. Golledge, editor. *Wayfinding Behavior: Cognitive Mapping and Other Spatial Processes*. The Johns Hopkins University Press, Baltimore, Maryland, 1999.

R. G. Golledge, T. R. Smith, J. W. Pellegrino, S. Doherty, and S. P. Marshall. A conceptual model and empirical analysis of children's acquisition of spatial knowledge. *Journal of Environmental Psychology*, 5(2):125–152, 1985.

J. L. Gould. Honey bee cognition. *Cognition*, 37:83–103, 1990.

G. Grisetti, C. Stachniss, and W. Burgard. Improving grid-based SLAM with Rao-Blackwellized particle filters by adaptive proposals and selective resampling. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2443–2448, Barcelona, Spain, 2005.

T. Guilford, S. Roberts, D. Biro, and I. Rezek. Positional entropy during pigeon homing II: navigational interpretation of Bayesian latent state models. *Theoretical Biology*, 227(1): 25–38, March 2004.

D. Hähnel, W. Burgard, D. Fox, and S. Thrun. An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 206–211, Las Vegas, Nevada, 2003a.

D. Hähnel, S. Thrun, B. Wegbreit, and W. Burgard. Towards lazy data association in SLAM. In *Proceedings of the International Symposium on Robotics Research (ISRR)*, pages 83–105, Sienna, Italy, 2003b.

S. Harnad. The symbol grounding problem. *Physica D*, 42:335–346, 1990.

J. Haugeland. *Artificial Intelligence: The Very Idea*. MIT PRess, Cambridge, Massachusetts, 1985.

H. Heft. The role of environmental features in route learning: Two exploratory studies of way finding. *Environmental Psychology and Nonverbal Behavior*, 3:172–185, 1979.

N. Kalra, D. Ferguson, and A. Stentz. Incremental reconstruction of generalized Voronoi diagrams on grids. In *Proceedings of the International Conference on Intelligent Autonomous Systems (IAS)*, pages 114–123, Tokyo, Japan, 2006.

L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, August 1996.

R. M. Kitchin. Cognitive maps: What are they and why study them? *Journal of Environmental Psychology*, 14(1):1–19, March 1994.

B.-Y. Ko, J.-B. Song, and S. Lee. Real-time building of a thinning-based topological map with metric features. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 797–802, Sendai, Japan, 2004.

S. Koenig and M. Likhachev. Real-time adaptive A^*. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 281–288, Hakodate, Japan, 2006.

S. Koenig and R. G. Simmons. Unsupervised learning of probabilistic models for robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2301–2308, Minneapolis, Minnesota, 1996.

K. Konolige. Large-scale map-making. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 457–463, San Jose, California, 2004.

K. Konolige. Markov localization using correlation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1154–1159, Stockholm, Sweden, 1999.

K. Konolige. A gradient method for realtime robot control. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 639–646, Takamatsu, Japan, 2000.

D. Kortenkamp and T. Weymouth. Topological mapping for mobile robots using a combination of sonar and vision sensing. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 979–984, Seattle, Washington, 1994.

J. J. Kuffner and S. M. LaValle. RRT-Connect: An efficient approach to single-query path planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 995–1001, San Francisco, California, 2000.

B. Kuipers. The Spatial Semantic Hierarchy. *Artificial Intelligence*, 119:191–233, 2000.

B. Kuipers. An intellectual history of the Spatial Semantic Hierarchy. In M. E. Jefferies and W.-K. Yeap, editors, *Robotics and Cognitive Approaches to Spatial Mapping*, volume 38 of *Springer Tracts in Advanced Robotics*, pages 243–264. Springer, Berlin, Germany, 2008.

B. Kuipers and P. Beeson. Bootstrap learning for place recognition. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 174–180, Edmonton, Canada, 2002.

B. Kuipers, J. Modayil, P. Beeson, M. MacMahon, and F. Savelli. Local metrical and global topological maps in the Hybrid Spatial Semantic Hierarchy. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4845–4851, New Orleans, Louisiana, 2004.

B. J. Kuipers. Modeling spatial knowledge. *Cognitive Science*, 2:129–153, 1978.

B. J. Kuipers. The ‘Map in the Head’ metaphor. *Environment and Behavior*, 14(2):202–220, 1982.

B. J. Kuipers and Y.-T. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics and Autonomous Systems*, 8:47–63, 1991.

G. Lakoff and M. Johnson. *Metaphors We Live By*. The University of Chicago Press, Chicago, 1980.

A. Lankenau, T. Röfer, and B. Krieg-Brückner. Self-localization in large-scale environments for the bremen autonomous wheelchair. In *Spatial Cognition III*, volume 2685 of *Lecture Notes in Artificial Intelligence*, pages 34–61. Springer-Verlag, Berlin, Germany, 2002.

W.-Y. Lee. *Spatial Semantic Hierarchy for a Physical Mobile Robot*. PhD thesis, The University of Texas at Austin, 1996.

J. Leonard and P. Newman. Consistent, convergent, and constant-time SLAM. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1143–1150, Acapulco, Mexico, 2003.

V. Lifschitz. Nested abnormality theories. *Artificial Intelligence*, 74:351–365, 1995.

F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997.

K. Lynch. *The Image of the City*. MIT Press, Cambridge, Massachusetts, 1960.

M. MacMahon. *Understanding and Executing Natural Language Route Instructions*. PhD thesis, The University of Texas at Austin, 2007.

M. MacMahon, B. Stankiewicz, and B. J. Kuipers. Walk the talk: Connecting language, knowledge, action in route instructions. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 1475–1482, Boston, Massachusetts, 2006.

M. J. Mataric. Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on Robotics and Automation*, 8(3):304–312, 1992.

E. Menegatti. *Omnidirectional Vision for Mobile Robotics*. PhD thesis, The University of Padua, 2002.

D. Miller. A spatial representation system for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 122–127, St. Louis, Missouri, 1985.

J. Modayil. *Robot Developmental Learning of an Object Ontology Grounded in Sensorimotor Experience*. PhD thesis, The University of Texas at Austin, 2007.

J. Modayil and B. Kuipers. Bootstrap learning for object discovery. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 742–747, Sendai, Japan, 2004.

J. Modayil, P. Beeson, and B. Kuipers. Using the topological skeleton for scalable, global, metrical map-building. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 1530–1536, Sendai, Japan, 2004.

M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: a factored solution to the simultaneous localization and mapping problem. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 593–598, Edmonton, Canada, 2002.

M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM 2.0: an improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1151–1156, Acapulco, Mexico, 2003.

H. P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, pages 61–74, Summer 1988.

A. C. Morris, D. Silver, D. Ferguson, and S. Thayer. Towards topological exploration of abandoned mines. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2117–2123, Barcelona, Spain, 2005.

O. M. Mozos, C. Stachniss, and W. Burgard. Supervised learning of places from range data using Adaboost. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1742–1747, Barcelona, Spain, 2005.

A. Murarka, J. Modayil, and B. Kuipers. Building local safety maps for a wheelchair robot using vision and lasers. In *Proceedings of the Canadian Conference on Computer and Robot Vision (CRV)*, page 25, Quebec City, Canada, 2006.

P. Newman, D. Cole, and K. Ho. Outdoor SLAM using visual appearance and laser ranging. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1180–1187, Orlando, Florida, 2006.

E. Olson, J. Leonard, and S. Teller. Fast iterative optimization of pose graphs with poor initial estimates. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2262–2269, Orlando, Florida, 2006.

M. A. Paskin. Thin junction tree filters for simultaneous localization and mapping. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1157–1166, Acapulco, Mexico, 2003.

J. Piaget. *The Construction of Reality in the Child*. Ballantine, New York, 1954.

I. Posner, D. Schröter, and P. Newman. Describing composite urban workspaces. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4962–4968, Rome, Italy, 2007.

W. H. Press, S. A. Teukolsky, W. T. Vitterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, London, second edition, 1992.

A. Ranganathan and F. Dellaert. Data driven MCMC for appearance-based topological mapping. In *Proceedings of Robotics: Science and Systems (RSS)*, pages 209–216, Cambridge, Massachusetts, 2005.

A. Ranganathan, E. Menegatti, and F. Dellaert. Bayesian inference in the space of topological maps. *IEEE Transactions on Robotics*, 22(1):92–107, February 2006.

N. S.V. Rao. Robot navigation in unknown generalized polygonal terrains using vision sensors. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(66):947–962, 1995.

I. Rekleitis. *Cooperative Localization and Multi-Robot Exploration*. PhD thesis, School of Computer Science, McGill University, Montreal, Quebec, Canada, 2003.

I. M. Rekleitis, V. Dujmovic, and G. Dudek. Efficient topological exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 676–681, Detroit, Michigan, 1999.

E. Remolina. *Formalizing the Spatial Semantic Hierarchy*. PhD thesis, The University of Texas at Austin, 2001.

E. Remolina and B. Kuipers. Towards a general theory of topological maps. *Artificial Intelligence*, 152:47–104, 2004.

R. L. Rivest and R. E. Schapire. Inference of finite automata using homing sequences. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 411–420, Seattle, Washington, 1989.

N. Roy, W. Burgard, D. Fox, and S. Thrun. Coastal navigation – mobile robot navigation with uncertainty in dynamic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 35–40, 1999.

F. Savelli. *Topological Mapping of Ambiguous Space: Combining Qualitative Biases and Metrical Information*. PhD thesis, University of Rome ”La Sapienza”, 2005.

F. Savelli and B. Kuipers. Loop-closing and planarity in topological map-building. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 1511–1517, Sendai, Japan, 2004.

D. Schröter. *Region & Gateway Mapping: Acquiring Structured and Object-Oriented Representations of Indoor Environments.* PhD thesis, Technical University of Munich, 2006.

D. Schröter, T. Weber, M. Beetz, and B. Radig. Detection and classification of gateways for the acquisition of structured robot maps. In *Proceedings of the Symposium of the German Association for Pattern Recognition (DAGM)*, pages 553–561, Tübingen, Germany, 2004.

H. Shatkay and L. P. Kaelbling. Learning topological maps with weak local odometric information. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 920–929, Nagoya, Japan, 1997.

A. W. Siegel and S. H. White. The development of spatial representations of large-scale environments. In H. W. Reese, editor, *Advances in Child Development and Behavior*, volume 10, pages 9–55. Academic Press, New York, 1975.

D. Silver, D. Ferguson, A. C. Morris, and S. Thayer. Feature extraction for topological mine maps. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 773–779, Sendai, Japan, 2004.

R. Sim and J. J. Little. Autonomous vision-based exploration and mapping using hybrid maps and Rao-Blackwellised particle filters. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 2082–2089, Beijing, China, 2006.

M. Skubic, S. Blisard, C. Bailey, J. A. Adams, and P. Matsakis. Qualitative analysis of sketched route maps: translating a sketch into linguistic descriptions. *IEEE Transactions on Systems, Man and Cybernetics – Part B: Cybernetics*, 34(2):1275–1282, 2004a.

M. Skubic, D. Perzanowski, S. Blisard, A. Schultz, W. Adams, M. Bugajska, and D. Brock. Spatial language for human-robot dialogs. *IEEE Transactions on Systems, Man and Cybernetics – Part C: Applications and Reviews*, 34(2):154–167, 2004b.

R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In I. J. Cox and G. T. Wilfong, editors, *Autonomous Robot Vehicles*, pages 167–193. Springer Verlag, New York, 1990.

B. J. Stankiewicz and A. A. Kalia. Acquisition of structural versus object landmark knowledge. *Journal of Experimental Psychology: Human Perception and Performance*, 33(2): 378–390, April 2007.

P. Stopka and D. W. Macdonald. Way-marking behaviour: an aid to spatial navigation in the wood mouse (Apodemus sylvaticus). *BMC Ecology*, 3(3), 2003.

A. Tapus and R. Siegwart. Incremental robot mapping with fingerprints of places. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 2429–2434, Edmonton, Canada, 2005.

S. Thrun and A. Bücken. Integrating grid-based and topological maps for mobile robot navigation. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 944–950, Portland, Oregon, 1996.

S. Thrun, S. Gutmann, D. Fox, W. Burgard, and B. J. Kuipers. Integrating topological and metric maps for mobile robot navigation: A statistical approach. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 989–995, Madison, Wisconsin, 1998.

S. Thrun, W. Burgard, and D. Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 321–328, San Francisico, California, 2000a.

S. Thrun, D. Fox, and W. Burgard. Monte Carlo localization with mixture proposal distribution. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 859–865, Austin, Texas, 2000b.

S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, Massachusetts, 2005.

E. C. Tolman. Cognitive maps in rats and men. *The Psychological Review*, 55(4):189–208, 1948.

N. Tomatis, I. Nourbakhsh, and R. Siegwart. Hybrid simultaneous localization and map building: Closing the loop with multi-hypotheses tracking. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2749–2754, Washington, DC, 2002.

L. G. Valiant. Robust logics. *Artificial Intelligence*, 117(2):231–253, 2000.

A. C. Victorino, P. Rives, and J.-J. Borrelly. Safe navigation for indoor mobile robots. Part I: a sensor-based navigation framework. *International Journal of Robotics Research*, 22 (12):1005–1018, 2003.

G. Voronoi. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. *Journal for Pure and Applied Mathematics*, 133:97–178, 1907.

J. O. Wallgrün. Autonomous construction of hierarchical voronoi-based route graph representations. In *Spatial Cognition IV. Reasoning, Action, Interaction*, volume 3343 of *Lecture Notes in Artificial Intelligence*, pages 413–433. Springer-Verlag, Berlin, Germany, 2005.

C. M. Wang. Location estimation and uncertainty analysis for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1231–1235, Philadelphia, Pennsylvania, 1988.

WorldViz LLC. Vizard virtual reality toolkit v3.0. `http://www.worldviz.com/products/vizard/index.html`, 2007.

B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, pages 146–151, Monterey, California, 1997.

M. Yannakakis and D. Lee. Testing finite state machines. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, pages 476–485, New Orleans, Louisiana, 1991.

F. S. Yates. *The Art of Memory*. The University of Chicago Press, Chicago, 1966.

W.-K. Yeap. Towards a computational theory of cognitive maps. *Artificial Intelligence*, 34: 297–360, 1988.

W.-K. Yeap and M. E. Jefferies. Computing a representation of the local environment. *Artificial Intelligence*, 107(2):265–301, 1999.

G. M. Youngblood, L. B. Holder, and D. J. Cook. A framework for autonomous mobile robot exploration and map learning through the use of place-centric occupancy grids. In *Proceedings of the International Conference on Machine Learning Workshop on Machine Learning of Spatial Knowledge*, pages 13–20, Palo Alto, California, 2000.

T. Y. Zhang and C. Y. Suen. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3):236–239, March 1984.

U. R. Zimmer. Embedding local metrical map patches in a globally consistent topological map. In *Proceedings of the International Symposium on Underwater Technology (UT)*, pages 301–305, Tokyo, Japan, 2000.

Vita

Patrick Beeson was born in Baton Rouge, Louisiana in 1977. He received a B.S. in Computer Science from Tulane University in 1999. Since that time, he has been working on his doctorate at the University of Texas at Austin. In addition to the hybrid spatial knowledge framework described in his thesis, Patrick is interested in robot learning, creating efficient implementations of common robotic algorithms, and human-robot interaction. In 2007, he was a lead programmer on the Austin Robot Technology team, supervising undergraduate programmers to produce an autonomous vehicle. This vehicle placed among the top 21 teams in the semi-finals of the DARPA Urban Challenge. After graduation, Patrick plans to hone his teaching skills by continuing to supervise undergraduate research on the autonomous vehicle platform.

Permanent Address: 3840 Far West Blvd
#206
Austin, TX 78731

This dissertation was typeset by the author.