TOPOLOGICAL MAPPING AND MAP MERGING WITH SENSING-LIMITED ROBOTS

By

Kristopher R. Beevers

A Thesis Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Approved:

____

Wesley H. Huang
Thesis Adviser

Rensselaer Polytechnic Institute
Troy, New York

April 23, 2004
(For Graduation May 2004)

Portions of this work have been previously submitted for publication [24, 25].

CONTENTS

APPENDICES

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENT

First and foremost, many thanks to my advisor Wes Huang, who introduced me to robotics two and a half years ago as an undergraduate. He has done an admirable job of bootstrapping my graduate student career, directing me to all the right resources at all the right times. Without his meticulous attention to detail, this thesis could not exist.

Secondly, I would like to recognize the contributions of the faculty and students in the Algorithmic Robotics Laboratory at RPI. Thanks especially for sitting through my (several) talks on this material, and for the constructive feedback thereafter. Special appreciation to Jon Fink and Alden Roberts for all their hard work on the robot hardware. Thanks also to the other Computer Science faculty and students, especially Chuck Stewart and Charlene Tsai for helpful discussions on the relationship between image registration and topological map merging.

Whether they realize it or not, Eric Griffith, Alan Kimball and Mike Venzke have been primarily responsible for the maintenance of my sanity over the past year. Thanks for reminding me that, like any other human, I need to relax every now and then — no matter how robot-like I may seem. Also, a round of huzzahs for the rest of my RPI *amigos*, who have moved on to less Troy-like pastures. There are too many to name here, but I will name them anyway: Alex, Amit, Austin, Ben, Carlos, Dave, Doug, Kentaro, Lee, Mark, Mike B., Mike N., Mike Y., Peter, and also Cecile, who is not from RPI, but I don't hold that against her. For anyone I have left out: it was an accident. Remind me before I finish my next thesis!

Finally and most importantly: my family. Your support, encouragement and guidance have been and always will be the most important factor in the equation that is me. Everything that I accomplish is a testament to the lessons you have taught and the advice you have given me over the years. Mom, Dad, Kimberlee, Gramma and Grampa, and everyone else: thank you! And Luke: someday I will invent a robot that pets dogs, but until then, you'll have to make do with me.

ABSTRACT

An important task in robotics research is to enable teams of mobile robots to cooperatively explore and map unknown environments. Robotic exploration and mapping has numerous applications in tasks that may be too dangerous for humans to perform, such as search and rescue, reconnaissance and chemical and biological agent detection. Using multiple robots for these tasks offers improvements in mapping efficiency and accuracy.

This thesis focuses on mapping with "sensing-limited" robots — robots that make use of inexpensive, short-range sensors such as infrared sensors. These robots have access only to sparse and limited-range information about the world. Sensing-limited robots are relatively inexpensive to build, so constructing teams of such robots is cost-effective. Additionally, because of their small cost, sensing-limited robots are potentially disposable — important in security applications, or in situations where the robots might be contaminated by some hazardous agent.

Two aspects of the multi-robot mapping and exploration problem are examined in this thesis. First, an algorithm is developed to create topological maps of indoor rectilinear environments with individual sensing-limited robots, a non-trivial task for robots with sparse, short-range sensing. The mapping algorithm introduces a method for extending topological mapping to open spaces, long considered a deficiency of the topological approach. Also introduced is an evidential-reasoning approach to the problem of "closing the loop" — recognizing when the robot has returned to a place it has already encountered.

Second, an algorithm is presented that merges two topological maps without a common reference frame. This algorithm uses aspects of maximal subgraph matching and image registration methods to join maps produced by individual robots into a coherent global map.

Experiments, performed both in simulation and with real-world robots, verify the effectiveness of the mapping and merging algorithms.

CHAPTER 1
INTRODUCTION

Robotic map-making is a well-studied problem in the robotics field, and recent years have seen renewed growth of interest in various aspects of mapping. This thesis examines two of these aspects — the single-robot topological mapping problem and the topological map merging problem.

The long-term goal of this work is to enable teams of small "sensing-limited" robots to autonomously explore, map and navigate in unknown indoor environments. Sensing-limited robots are robots that have access only to sparse and very limited-range information about the world. As an example, our hardware platforms (Figure 5.1) have only five infrared range sensors, each with a maximum range of approximately 80 cm. These limited sensing capabilities result in a limited view of the environment — features in the world can only be detected at close range, by observing a time series of sensor readings. In many office-like environments, these robots cannot even simultaneously detect both sides of a hallway. Unlike humans, who can do blind navigation by using spatial information previously acquired from far richer sensing modalities (e.g. vision), sensing-limited robots must use their limited sensing for exploration and mapping, as well as for navigation.

A major advantage of using sensing-limited robots is their cost; low-fidelity sensors are typically inexpensive. Because of the reduced cost, it is feasible to construct large teams of such robots. Additionally, small inexpensive robots are potentially disposable — even in large numbers. Disposability is important in security or reconnaissance applications where the the nature of the environment is such that the robots may not be able to be recovered. Alternatively, a team of disposable robots might be equipped with chemical- or biological-agent sensors for use in threat-assessment of a contaminated building; even if these robots are recoverable, exposure may require that they be destroyed.

Although the sensing capabilities of our robots are limited, these capabilities are still sufficient for the exploration, navigation and mapping of an unknown environment. However, we cannot expect a sensing-limited robot to build a map as quickly as a robot with richer sensing; nor should we expect a sensing-limited robot to be as efficient at navigation within its map. Nevertheless, our robots are well-suited to the creation of certain types of maps. In particular, this thesis focuses on the construction of topological maps.

1.1 Topological maps

Topological maps, pioneered by Kuipers [30], represent the "connectivity" of an environment, generally in a graph structure. Vertices in topological maps typically represent "distinct places" in the environment — locations with unique features or easily recognizable sensor signatures. Often, these places occur at junctions of hallways. Edges in topological maps may represent:

- Specific paths between places;
- Classes of paths between places, e.g. all different paths that travel down a corridor; or
- A behavior or sequence of behaviors that take the robot from one place to another, and the initial conditions for these behaviors.

A robot with sparse, limited-range sensing is quite capable of executing low-level behaviors such as wall-following and hall-following. This thesis focuses on the creation of topological maps defined in terms of such behaviors. These maps are represented by graphs in which edges correspond to sequences of behaviors that move the robot from one location to another, and vertices correspond to the termination points of behaviors.

The maps created in this fashion are certainly useful for robotic navigation. Often, they are also suitable for human use — particularly when some metric information about the environment is encoded. Our maps store information about the locations of map vertices and the lengths of paths between them, as determined by the robot's odometry.

This odometric information is inaccurate because of error — caused by wheel slip and sensing inaccuracies — and so the precise locations of places in the map cannot be known. However, given a reasonably accurate model of the error experienced by the robot, probabilistic estimates of the metric measurements that are taken can be maintained, and these estimates can be improved by taking new measurements as necessary. Additionally, statistical techniques can be used to compare newly-taken measurements to data in our maps — useful in localization, for example.

1.2 Thesis overview

The general multi-robot mapping problem is complex and multifaceted. This thesis examines two specific aspects:

Single-robot mapping Before the multi-robot mapping problem can be tackled, the single-robot case must be examined. Given a suitable single-robot mapping algorithm for sensing-limited robots, extending the mapping to multiple robots should be reasonably straightforward. In fact, a simple approach is to just have individual robots create their own maps of different portions of the environment, and then merge these maps into one.

This thesis presents a single-robot topological mapping algorithm for sensing-limited robots in indoor environments. The algorithm is effective in the presence of sensing and odometry error. With this algorithm, a paradigm is developed to extend topological mapping to open spaces, a well-known weakness of topological maps. Also introduced is an evidential-reasoning approach to the problem of "closing the loop" — recognizing when the robot has returned to a place it has already been.

Topological map merging For a team of mobile robots working to cooperatively explore and map an environment, a necessary capability is for the maps of one robot to be merged with those from another robot. This problem is not straightforward when the robots do not share a common frame of reference. The use of topological maps simplifies the merging problem to a certain extent; topological maps provide a concise description of the navigability of a space, and comparisons can readily be made between portions of two or more such maps.

This thesis presents an algorithm for merging two topological maps that makes use of aspects of maximal subgraph matching techniques and image registration methods. The merging algorithm uses the structure of the topological maps to find a set of hypothesized matchings between vertices, and then groups these hypotheses into consistent clusters based on geometric transformation.

Issues with the implementation of the merging algorithm are also discussed. In particular, map data structures and map updating strategies that lend themselves well to map merging are suggested.

The remainder of this thesis is organized as follows. We begin by discussing previous work on the single-robot mapping problem, the multi-robot mapping problem, map merging and other related topics in Chapter 2. In Chapter 3 the single-robot mapping algorithm for sensing-limited robots is formulated. The topological map merging algorithm is presented in Chapter 4. The results of simulated and real-world experiments for each of the algorithms are given in Chapter 5. Finally, in Chapter 6, the contributions of the thesis are summarized, and we discuss possible future work in this area.

Appendix A provides a brief introduction to the Dempster-Shafer theory of evidence, which is used extensively in the single-robot mapping algorithm. Appendix B describes the function used to convert evidence from measurements taken during mapping into "belief" for use with our Dempster-Shafer based mapping methods. Finally, Appendix C details the specifics of the behaviors that are used by our sensing-limited robots to build their maps.

CHAPTER 2
RELATED WORK

In recent years there has been significant interest in various aspects of robotic map-making, from single-robot mapping to highly distributed multi-robot mapping, and a number of algorithms and map representations have been devised for solving these problems. This thesis builds on previous mapping research, and draws from work on related problems in other areas, such as statistical inference and graph matching.

2.1 Single-robot mapping

The single-robot mapping problem has been researched for more than twenty years. In this time, two dominant mapping paradigms have developed: metric mapping and topological mapping. In metric maps, the geometry of the world is explicitly represented, either through exact or approximate representations. In topological maps, "places" in the world (typically junctions of hallways) are represented by vertices in a graph, and paths between places are represented by the edges. Metric maps provide a detailed world representation but require more storage and are sensitive to measurement errors. Topological maps offer a more concise representation but, as traditionally implemented, cannot represent open spaces. Most mapping techniques, whether metric or topological in nature, are probabilistic in some sense. Thrun's recent survey [61] offers an overview of several well-known probabilistic mapping techniques.

2.1.1 Occupancy grid maps

In 1985, Moravec and Elfes [40] introduced the idea of occupancy grid maps for representing metric information about an environment. Occupancy grids represent an environment in a grid of "cells" — much like the pixels of a digital image. For each cell, the probability that the cell is occupied by some object is maintained. More recent work by Konolige [28], Thrun, Fox, and Burgard [59] and others [35, 46] has built on and refined the occupancy grid approach, and applied it to problems such as simultaneous localization and mapping (SLAM).

Most occupancy grid maps are updated using Bayesian methods when new data is obtained. Pagac, Nebot, and Durrant-Whyte [42] have proposed the use of belief functions instead of probabilities. They use the Dempster-Shafer theory of evidence to maintain belief about the presence of obstacles in occupancy grid cells.

2.1.2 Topological maps

Topological maps, first introduced by Kuipers [30, 31, 43], take a different approach — extracting information about the connectivity of the environment, and in many cases discarding most metric information. Some work has focused on extracting topological maps from occupancy grid data [7,58]. Most topological mapping research, however, attempts to acquire topological information about the environment during the exploration process [33, 50]. Among recent work in this area, Rybski *et al.* [45] build topological maps by identifying visual features in image data acquired during the exploration process, and Tovar, LaValle, and Murrieta [64] maintain a dynamically-updated visibility tree representation that captures the topology of the environment.

Some researchers have combined the metric and topological mapping approaches by maintaining local metric information (i.e. occupancy data) at topological nodes [15, 60]. Tomatis, Nourbakhsh, and Siegwart [62, 63] use a topological map to represent a network of hallways and use metric maps to represent rooms. In contrast, the work presented in this thesis extends the topological mapping paradigm to represent open spaces. The use of a wall-following behavior to generate an initial map of an open area is related to the idea of "coastal navigation" [44] which recognizes that areas near walls and obstacles have "high information content" due to the features they produce. Once the robot has built a map of the world boundaries, it can safely explore the interior.

2.1.3 Closing loops

One of the fundamental problems in topological mapping is "closing the loop" — recognizing when the robot has returned to a place it has previously visited. (An analogous problem for metric maps is the "correspondence problem.") A simple but impractical approach is to "drop a pebble" [1] — upon returning to the pebble, the loop has been closed. Kuipers and Beeson [32] take a different approach: they use supervised learning to recognize the "distinctive places" that are nodes in their topological maps. With sufficiently rich sensing, distinctive places are easily differentiated by recording a unique "sensing signature" for each place.

Without sensing rich enough to recognize distinctive places, the robot must continue to traverse the environment to deduce whether a loop has been closed. Kuipers' "rehearsal procedure" [33] encapsulates the general idea of using the map topology to make this decision. Choset and Nagatani [8], whose topological maps are based upon the Generalized Voronoi Diagram of the environment, describe an approach where structural characteristics of the map (e.g., the degree of vertices and the order of incident edges) are the primary criteria for verification. Tomatis *et al.* [63] embed this comparison in a POMDP that should show a single peak when the loop has been closed.

2.1.4 Sensing-limited robots

Others have specifically addressed mapping and exploration with limited sensing capabilities. Butler, Rizzi, and Hollis [6] describe a method for coverage of a rectilinear environment using robots with only contact (bump) sensors but with near-perfect odometry. Coverage is largely equivalent to metric mapping.

Doty and Seed [13] have shown preliminary results in creating "landmark" maps using a robot with four short-range infrared sensors and one long-range SONAR sensor. They take an approach similar to ours in constructing basic maps: the robot circumnavigates obstacles using simple behaviors, creating "contour maps" of the obstacles. However, their approach to the loop-closing problem is somewhat simplistic: the robot deduces that an obstacle has been completely circumnavigated when the total angle through which the robot has rotated is within $\pm 45^o$ of 360^o; verification of this deduction is performed by making a "second pass" around the obstacle. Additionally, they do not address issues of odometry or sensor error beyond specifying "thresholds" for most actions.

2.1.5 Other work

Our approach to the loop-closing problem is based on accumulating evidence to verify or reject loop-closing hypotheses. The evidence is based on belief functions derived from the odometry error model of the robot and is combined using Dempster-Shafer theory [41,47–49]. This approach is related to that of Cox and Leonard [10], who maintain multiple hypotheses about the state of a dynamic world. In their work, the probability of each hypothesis is assigned and updated using a Bayesian framework. The main advantage of a Dempster-Shafer based method is the ability to represent "ignorance" — lack of belief in either the correctness or incorrectness of the loop-closing hypothesis.

In order to make hypotheses about loops in the environment, we draw on the techniques of "stochastic mapping," first introduced by Smith and Cheeseman [52] and later promulgated by Leonard and Feder [18,34]. In stochastic mapping, feature locations have associated positional uncertainty probability distributions that are based on the robot's error models. Our robots maintain such distributions and use them to generate loop-closing hypotheses and to gather evidence about the hypotheses.

2.2 Multi-robot mapping and map merging

Intuitively, a team of robots should be able to create maps more efficiently and accurately than a single robot. Recent research on cooperative mapping has shown that this is generally the case.

2.2.1 Approaches

Most multi-robot mapping work has focused on the creation of occupancy grid maps [22,27,57]. Grabowski *et al.* [22] discuss the application of a hierarchical team of robots to the exploration and mapping problem; in the lowest level of their hierarchy are “millibots,” small robots with minimal sensing capabilities.

Some work has also been done on multi-robot feature-based or topological mapping [11, 19, 55]. Jennings, Kirkwood-Watts, and Tanis [26] use individual robots to create maps from Voronoi paths, and then use a simple distance metric to merge the maps. Dudek *et al.* [16] create maps of a graph-like world under the assumption that all robots start at the same vertex in the graph. Konolige *et al.* [29] pose the map merging problem in a decision-making framework, and show the efficacy of taking a feature-based approach to merging instead of attempting to match occupancy data.

Much of the multi-robot map making and merging work assumes that all robots in the team begin the mapping process with a common reference frame — a restrictive assumption that is not made in this thesis. One notable exception is the approach taken by Ko *et al.* [27] in which robots exchange occupancy maps and attempt to localize themselves in each others' maps using a version of particle filtering.

The work most related to ours is that of Dedeoglu and Sukhatme [11], who present a method for merging landmark-based maps without a common reference frame. They attempt to estimate a transformation between two maps given a single-vertex match. The approach presented here is similar in that it estimates geometric transformations between the two maps. However, rather than use these transformations to generate vertex correspondences, our method computes the transformations *using* correspondences that are found from the structure of the maps. This use of the maps' topologies results in quick and effective discovery of potential matches.

2.2.2 Graph matching

The topological map merging problem can be viewed in a graph-matching framework. Graph matching [9] is an area that has seen an explosion of research over the past three decades, particularly in the pattern recognition literature. When no metric information is embedded in the maps, the matching problem is largely structural in nature — most useful information comes from the edge connectivity of the map — though there is often some extra information available, such as node types. In particular, the map merging problem can be phrased as an instance of the maximal common subgraph problem [4, 17, 38], since it is desirable to find the largest possible match between two maps. This problem is closely related to the problems of structural matching and error-tolerant subgraph isomorphism [5, 37, 39, 66].

2.2.3 Image registration

If the maps retain metric information — for example, the estimated locations of features, or lengths of paths between them — the problem can be viewed in the framework of image registration and mosaicking. A thorough (but aging) survey of image registration techniques was written by Brown in 1992 [3]. Most image registration algorithms, such as the widely known iterative closest point (ICP) algorithm [2], are iterative in nature [51, 53, 54]. Iterative registration algorithms attempt to find transformations between image coordinate frames by repeatedly minimizing the distances of closest-point correspondences of features.

The approach taken in this thesis draws from both the graph matching and image registration literature. Matches are "grown" based on the topology of the map graphs. These matches are primarily structural in nature, though they do take into account the similarity of relevant metric vertex and edge attributes. Using these structural matches, procedures from the image registration literature are employed to find geometric transforms that minimize the distance between matched vertices. Clustering matches by the similarity of their transforms yields consistent groups of matches. This method performs well even when there are multiple disconnected overlapping regions in the separate maps.

CHAPTER 3
MAPPING WITH SENSING-LIMITED ROBOTS

Consider the following situation: you are navigating through your house in the dark — you've woken up in the middle of the night, hungry for a snack. You get out of bed, but leave the lights off so you don't wake anyone else. You can't see anything, so you run one hand along the wall as you walk down the hallway. There are small discontinuities in the wall, like the closet door, but you keep walking straight. You feel the wall "end," so you turn the corner and keep going. Perhaps at some point, you leave the wall and walk across a (known obstacle-free) room, holding your hand out in front to detect when you reach the opposite wall. You eventually reach the kitchen without much trouble.

This "blind" navigation is an apt analogy for how a sensing-limited robot "sees" the world. By "sensing-limited," we mean that the robot has sparse and very limited-range information about the world. Because of its sensing limitations, the robot's view of its environment is severely restricted — features of the world can only be detected at close range by observing a time series of sensor readings.

In order to create maps with teams of cooperating sensing-limited robots, the problems of exploration, mapping and navigation with a single robot must first be tackled. Our problem is for a sensing-limited robot to explore an unknown environment, creating a topological map that it can later use to navigate between any two previously-visited locations. The resulting paths should be efficient to the extent that safety, limited-range sensing, and errors in sensing, robot motion and odometry allow.

This chapter demonstrates that sensing-limited robots can complete this task, though there are some limitations, particularly on map completeness, based on the size and geometric complexity of the world. We should not expect the same performance from a sensing-limited robot as we would from a robot with richer sensing modalities (e.g. a laser scanner). While we will permit lesser performance in terms of time and purview, we still expect reasonable accuracy.

Our approach constructs a basic map which can be considered a topological map. In this map, vertices are places where a robot behavior terminates, and edges represent a sequence of behaviors that move the robot from one place to another. Using this philosophy, we can transcend the typical weakness of topological maps in representing open spaces. First, we introduce the concept of *portals* which link open spaces (mapped with a wall-following behavior) to narrow corridors (mapped with a hall-following behavior). Second, we introduce the idea of *forays*, which cross open spaces to produce more direct links between vertices in the map.

Our use of sensing-limited robots has also led us to a new approach to the problem of "closing the loop" — recognizing that the robot has returned to a place in the world that it has already visited. Since sensing-limited robots cannot uniquely recognize a place based on instantaneous sensor readings, we create a hypothesis that the robot has closed the loop based on its geometric position estimate, and attempt to verify this hypothesis using an evidential-reasoning approach that compares characteristics of subsequent edge traversals.

3.1 Assumptions

We consider a sensing-limited robot that has a small number of short-range sensors; these sensors return the distance r to the nearest obstacle along a straight line within the maximum range. Note that "short-range" is relative to the dimensions in the environment. There must be enough sensors to robustly execute the behaviors described in Section 3.2.1. As an example, our robot (Figure 5.1) has five infrared range sensors: one forward sensor and two sensors on each side. One of the robot's sensors (typically a right-facing sensor) is specified as the "wall-following sensor," and is the primary sensor used during map construction.

There is known error in the robot's movement, odometry and sensors. In particular, the distance a robot travels and the angle it turns may be inaccurate. Without any exteroceptive feedback, it may not travel in a straight line. We assume these errors are random and zero-mean, but we do not assume a particular probability distribution — only that we can merge and compound measurements and compute confidence bounds for a given confidence level. We assume that the robot experiences no error in detecting the "types" of features it encounters.

We assume an enclosed, static, rectilinear environment. Though most of our ideas extend directly to general polygonal worlds, the rectilinear assumption removes uncertainty in the robot's orientation, allowing us to focus on our extensions to topological mapping. We make no assumption about the "smoothness" of the environment, e.g., there may be protrusions and recesses in a hallway due to door frames, structural columns, etc.

Our algorithm takes a parameter α, which represents a confidence limit with which all the algorithm's "decisions" must be made. In particular, this parameter is used when closing loops and when making "refinements" to the map without the availability of sensor feedback.

3.2 Basic mapping algorithm

The "basic map" created by our robots is a topological map that represents the circumnavigation of open spaces with a wall-following behavior, and the traversal of narrow corridors with a hall-following behavior. First, we describe the required robot behaviors and then detail construction of the map.

3.2.1 Behaviors

In order to create the basic map, we require that our robot be capable of executing the following behaviors:

Move-to-wall The MOVE-TO-WALL behavior causes the robot to move forward from its current configuration until one of its sensors detects a wall. The robot then turns to align itself so that its wall-following sensor (a side sensor) faces the wall.

Wall-follow The WALL-FOLLOW behavior (Figure 3.1) causes the robot to move forward, maintaining a range $r_0 \in [r_{\min}, r_{\max}]$ to the wall with its wall-

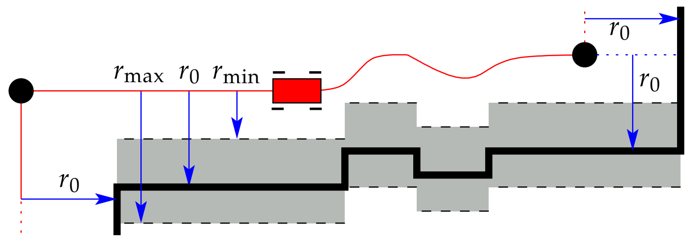


Figure 3.1: Wall-following and feature detection. The "safe" wall-following range $[r_{min}, r_{max}]$ is indicated by the shaded area. Vertices are placed at a distance of r_0 from both incident walls of a well-defined corner. The leftmost corner is exterior; the rightmost is interior. The rest of the discontinuities in the wall constitute ill-defined features.

following sensor. The wall-following behavior terminates when the robot detects a *well-defined corner*. A well-defined *exterior* corner occurs when the wall "falls away" by more than the range $r_{max} - r_0$ (i.e., a robot following at r_0 sees the wall disappear beyond r_{max}); likewise, a well-defined *interior* corner occurs when the wall juts inward more than $r_0 - r_{min}$. Any discontinuity that remains within the range $[r_0 - r_{min}, r_{max} - r_0]$ of the robot's wall-following sensor is an *ill-defined* corner (typically unused in our maps).

In the case of an exterior corner, wall-following terminates at a distance of r_0 past the discontinuity — an open-loop motion must be performed. For an interior corner, wall-following stops at a distance of r_0 *before* the discontinuity. Note that exterior corners will typically be detected by the wall-following sensor, whereas interior corners will be detected by either front or side sensors. The wall-following behavior must be capable of terminating in a reasonably repeatable location, regardless of the direction of travel.

Note that because of the termination conditions of the wall-following behavior — in particular, the r_0 offset from walls incident to well-defined corners — no two map vertices representing well-defined corners of the same type can be within a distance

$$r_* = \sqrt{r_0^2 + r_{min}^2} \tag{3.1}$$

of each other.

Hall-follow The HALL-FOLLOW behavior is similar to the wall-following behavior, except that it is only used when the robot is traversing a "corridor" of width $\leq 2r_0$. In this case, both of the robot's side sensors can "see" walls, and the robot moves through the center of the corridor. When hall-following, the robot may encounter "junctions" of hallways or corners for which it must perform open-loop motions similar to the navigation of exterior corners while wall-following.

Refer to Appendix C for complete specifications of the MOVE-TO-WALL, WALL-FOLLOW and HALL-FOLLOW behaviors.

3.2.2 Mapping process

We first consider an environment that consists entirely of "open" spaces — an environment that contains no "narrow" spaces of width less than $2r_0$. Essentially, open spaces are areas in which the robot cannot "see" obstacles with both side sensors simultaneously. (The open space assumption is removed in Section 3.4.) The basic map-making process is as follows:

- From an arbitrary starting location, the robot initiates the MOVE-TO-WALL behavior, which aligns the robot with some wall in the environment.
- The WALL-FOLLOW behavior is activated, and its termination point becomes the start vertex v_0 — the first vertex in our topological map.
- After finding v_0, the robot turns as appropriate to follow the next wall (incident to v_0). Vertices are added for each well-defined feature that terminates the WALL-FOLLOW behavior. The placement of each vertex is such that it is offset a distance of r_0 from each incident wall (this ensures unique vertex placement relative to the feature regardless of direction of travel). Each new vertex is connected, with an undirected edge, to the previously discovered feature (i.e. $v_0 \leftrightarrow v_1$, $v_1 \leftrightarrow v_2$, and so on).
- This process repeats until the robot returns to v_0. This is guaranteed to happen, since the environment is enclosed.

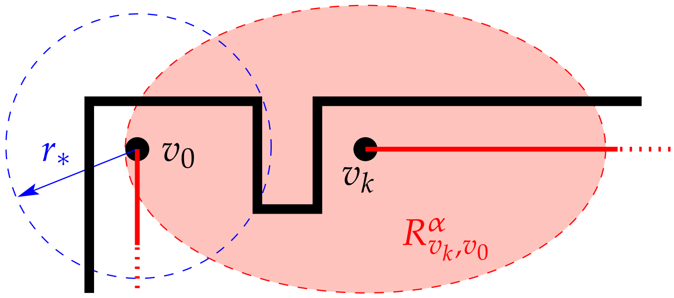


Figure 3.2: A situation in which we hypothesize that $v_k = v_0$. The region $R^{\alpha}_{v_k,v_0}$ represents the positional uncertainty in the location of v_k with respect to v_0, and is bounded according to the confidence limit α, an algorithm parameter. Since this region contains v_0, it is possible that the two vertices are the same. However, since the region is not completely bounded by the circle of radius r_* around v_0, we are not certain that the hypothesis is correct, and must obtain further evidence before making a decision.

Recognizing when the robot returns to the start vertex is the problem of "closing the loop."

3.3 Closing the loop

The loop-closing problem is a difficult one when the robot has no prior knowledge of the complexity of the environment, and a number of different methods for solving the problem have been proposed [1, 8, 32, 33, 63]. We take the following approach:

- First, we must recognize when the robot may have returned to the start vertex v_0. Recognizing this possibility causes us to "hypothesize" that we have closed the loop.

- Upon making such a hypothesis, we must verify or reject it. There are several potential methods; we propose an evidential-reasoning approach based on the Dempster-Shafer theory of evidence.

3.3.1 Making a hypothesis

For each vertex v_i in the topological map, we use the model of the robot's odometry error to maintain a probability density function U_{v_i,v_0} that describes the location of v_i with respect to v_0. The uncertainty in this distribution grows monotonically as the map expands. For a confidence limit α specified to our mapping algorithm, let $R^{\alpha}_{v_i,v_0}$ represent confidence bounds on U_{v_i,v_0}. This is a region in which we believe the robot to be located with confidence α.

If the robot reaches a vertex v_k that is the same type as v_0 (e.g. both are interior corners, or both are exterior corners) and v_0 lies within $R^{\alpha}_{v_k,v_0}$, then we create a hypothesis

$$H_0 \equiv v_k = v_0 \tag{3.2}$$

that vertex v_k closes the loop. See Figure 3.2 for an example of such a situation. Note that if the entire region $R^{\alpha}_{v_k,v_0}$ falls completely within a radius r_* of v_0 (generally not the case), we can immediately declare the hypothesis correct with at least the specified confidence since no other vertex of the same type can be within that radius of v_0.

3.3.2 Verifying a hypothesis

Once a loop-closing hypothesis is made, we attempt to verify it. One way to do this is to *backtrack* — travel back and forth between v_k and v_0, taking additional measurements of all the edges. Under the assumption of random zero-mean errors, backtracking allows us to reduce positional uncertainty by merging multiple measurements of the same edges. Backtracking will eventually reduce the size of $R^{\alpha}_{v_k,v_0}$ until it either lies entirely within r_* of v_0 or until v_0 does not lie in $R^{\alpha}_{v_k,v_0}$. This is the only certain way to verify or reject a hypothesis, but it is impractical, requiring many trips back and forth between v_k and v_0.

Instead, after making a loop-closing hypothesis, our robot continues to traverse edges in the "forward" direction, comparing the features it encounters and the measurements it takes to those it expects to see under the hypothesis. If we come upon some structural difference (e.g. vertices of different types or incident edges at different orientations), then the hypothesis can be rejected. In order to verify the hypothesis, however, we compare lengths of edges that should match under the hypothesis, and use the comparison to build "evidence" for or against the hypothesis using methods prescribed by the Dempster-Shafer theory of evidence [49].

A brief overview of the central concepts of Dempster-Shafer theory is provided in Appendix A. Dempster-Shafer theory gives us a natural way to assign "belief" to a set of possibilities — in this case, that the hypothesis is correct, which we denote as C, or incorrect, denoted I. It also provides a way to represent "ignorance" about these possibilities. Ignorance, denoted $\{C, I\}$, can be thought of as uncertainty about which possibility is supported by evidence. The representation of ignorance is particularly useful in inherently uncertain situations such as ours.

For each possibility C, I and $\{C, I\}$, we maintain a basic probability as we gather evidence: $m(C), m(I)$ and $m(\{C, I\}$, respectively. Together, these basic probabilities constitute a *belief function*. Dempster-Shafer theory provides rules for combining belief functions constructed from multiple sets of evidence.

Suppose that we have made a hypothesis H_0 that two vertices in our map are the same. Then, continuing our forward traversal of the walls of the environment, we take a series of new wall length measurements. Under H_0, each new measurement ℓ has one or more corresponding, previously-taken measurements that originate and end at the same vertices in the map as measurement ℓ.

Together, ℓ and the corresponding older measurements constitute a set of measurements

$$L = \{\ell_0, \ell_1, \ldots, \ell_n\} \tag{3.3}$$

with $n \geq 2$. Under H_0, all the measurements in L originate from common vertex v_i and end at common vertex v_j; the set L thus corresponds to a single edge in the map graph. We wish to determine the evidence provided for our hypothesis by L. We use Dempster-Shafer theory to merge evidence provided by multiple sets of such measurements (each set corresponds to a different pair of adjacent map vertices). Over time, evidence accumulates in support of (or against) H_0.

It is reasonable to use only the "side-by-side" comparison of individual measurements in determining evidence — as opposed to comparing map "geometry" (shape) determined by the measurements of several walls — because for a correct hypothesis, errors in each measurement are independent. A large discrepancy among a previous set of measurements of a single wall should certainly provide evidence against the hypothesis, but it should not be "used" more than once — as would be the case were we to use geometry to determine evidence.

Given a set of measurements L and their associated uncertainty distributions, we must determine basic probability values for $m_L(C), m_L(I)$ and $m_L(\{C, I\})$ such that

$$m_L(C) + m_L(I) + m_L(\{C, I\}) = 1 \tag{3.4}$$

Here, $m_L(C)$ represents our belief, given the measurements in L, that H_0 is correct, $m_L(I)$ represents our belief that H_0 is incorrect, and $m_L(\{C, I\})$ represents our "ignorance" — the degree to which L provides evidence neither for nor against the hypothesis. The function by which we determine $m_L(C), m_L(I)$ and $m_L(\{C, I\})$ is termed a "belief function."

Our particular belief function draws on well-known methods of statistical inference. It essentially computes probabilities that the measurements in L were taken from the same wall, given the robot's error model. The "probability information content" [56] of these probabilities is used to measure our degree of ignorance. Refer to Appendix B for the full details of the belief function.

As we take new measurements, we must combine the evidence they provide with previous evidence we've acquired, in order to update our global belief in the correctness of the hypothesis. To combine evidence, we use Dempster's rule of combination. (For discussion of the details of this rule, see Appendix A.) In our particular application, the combination rules are as follows: given global belief m and belief m_L attributed to a specific set of measurements L,

$$m \oplus m_L(C) = \frac{m(C)m_L(C) + m(C)m_L(\{C, I\}) + m(\{C, I\})m_L(C)}{1 - m(C)m_L(I) - m(I)m_L(C)} \tag{3.5}$$

$$m \oplus m_L(I) = \frac{m(I)m_L(I) + m(I)m_L(\{C, I\}) + m(\{C, I\})m_L(I)}{1 - m(C)m_L(I) - m(I)m_L(C)} \tag{3.6}$$

We continue building evidence about a loop-closing hypothesis until our overall belief in the correctness (or incorrectness) of the hypothesis surpasses some threshold β. In general, we choose $\beta = \alpha$, the confidence bound we used when generating the hypothesis. Refer to Appendix B for full details on making decisions about the correctness of loop-closing hypotheses, and also for discussion of the initial values for the global belief function.

3.3.3 Discussion

The worlds in which it is most difficult to close the loop are those with structural self-similarity. Two ways in which a world might be self-similar occur when the world:

- Consists of similar sequences of walls that "spiral" inward or outward;
- Consists entirely or partially of "repeating similar sequences" of walls.

Our evidential approach fares well in "spiral" worlds, as long as the error experienced by the robot is not large relative to the "tightness" of the spiral. By their nature, spiral worlds yield hypotheses for which edge measurements are different. Worlds of "repeating similar sequences" are generally more difficult, though they can often be properly mapped despite the presence of error, given a high enough threshold for the acceptance of a hypothesis. In Appendix B we further discuss the problem of structural self-similarity. We point out a potential solution based on measuring the "distinctiveness" of path length measurements, and discuss the tradeoffs of this approach.

Continuing the forward traversal of the environment after making a loop-closing hypothesis presents a number of issues. The key concern with this approach is that one can never be completely certain that a hypothesis is correct; this is a deficiency of nearly all hypothesis-based methods for closing loops. The backtracking approach resolves this issue, but only after an indefinite number of measurements. We believe that the idea of building "evidence" in support of and against the hypothesis is both intuitive and reasonable for use in practical situations.

We have made several further observations concerning the loop-closing problem:

- Upon truly returning to v_0, the robot will have disproved all inconsistent hypotheses by structural/topological mismatches. However, the robot cannot know when it has returned to v_0 because the complexity of the world is unknown.

- The robot's orientation must pass through a full 360^o (net) before it can return to the start vertex.

- False positive or false negative loop-closings are not "show-stoppers" (we can disprove them later), but it is preferable to find the right hypothesis during basic mapping. As such, acceptance thresholds should be reasonably high unless prior knowledge of the environment indicates that self-similarity will not be an issue — often the case.

- In general topological mapping scenarios in which vertices may have degree greater than two, we should choose the path at a junction that we expect to yield the most evidence.

- Prior knowledge of some aspects of the world (e.g. "feature density," distribution of potential wall lengths, etc.) might lead to significantly more efficient loop-closing by allowing us to design more "informed" belief functions and letting us be more aggressive in accepting or rejecting hypotheses.

3.4 Portals and narrow corridors

We have so far addressed only the case where open space lies to one side of the robot. When the area around the robot is narrow enough that the robot can see walls on both sides, the robot should switch from wall-following to the hall-following behavior. We define *portals* to make the connection between these two different mapping modes.

We can extend our basic mapping to use portals in order to skip over narrow corridors that may lead to other open areas, leaving these areas for later exploration. This typically has the advantage of making the loops in the basic map smaller and therefore easier to close. Although doorways are prime candidates for portals, there is nothing special about them except that they constitute a narrow corridor.

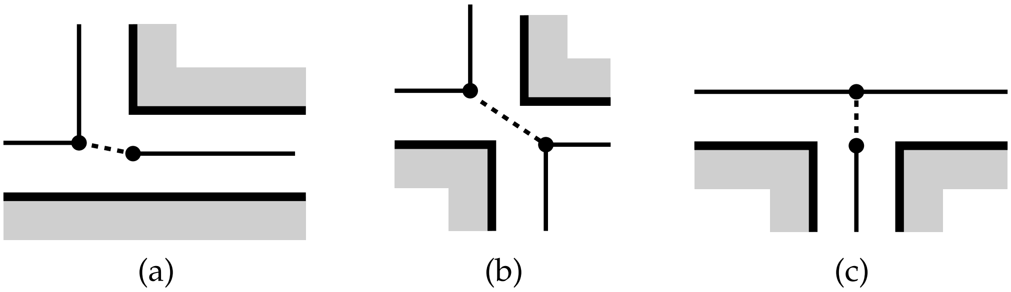


Figure 3.3: Portals occur under special wall-following or hall-following terminating conditions. In each figure, the robot moves from left to right. In Figure (a), the interior-corner portal is detected while following a wall. In Figure (b), the portal, detected while turning an interior corner, leads directly to another open space. In Figure (c), the portal is detected while turning an exterior corner.

We consider a narrow corridor to be a space $\leq 2r_0$ wide, and we assume that the robot can only hall-follow in corridors that are at least $2r_W$ wide. A portal occurs at a place where these conditions first appear while wall-following, or where they first disappear while hall-following (for examples, see Figure 3.3). In our maps, portals are generally represented by special edges to terminal hall-following vertices. In some special cases, such as that depicted in Figure 3.3(b), a portal between two wall-following vertices is possible.

We defer a complete discussion of the methods by which portals are detected while wall-following and hall-following to Appendix C, where the behaviors are specified in their entirety.

3.5 Navigation with the basic map

After we have completed the basic mapping process, navigation between any two points in the map is straightforward: we use a graph search of the map to construct a sequence of behaviors, and then execute these behaviors. However, navigation through the basic map is inefficient in many cases: the robot may need to circumnavigate half of a building to reach a point across a wide hallway. The second phase of our algorithm introduces *refinements* into the map that improve its usefulness for navigating through open spaces (and for discovering "islands" in these spaces when they exist).

3.6 Refinements

Refinements to our basic map attempt to connect disjoint places in the environment with paths that pass through open (featureless) regions. A common deficiency of many topological mapping methods is the inability to deal with "open spaces." Refinements offer a tool for extending topological maps to such spaces in a natural way. Note that refinements apply only to portions of our map generated using the wall-following behavior; refinements are unnecessary in narrow corridors explored by hall-following.

Our refinements are based on the geometry of the basic map. So, prior to generating candidate refinements, we must embed the vertices of the map in the plane. Because of measurement error during basic mapping, once we close the loop the map is not planar — our measurements are geometrically inconsistent with the loop-closing hypothesis that we've accepted.

The problem of generating a consistent geometric map from the local distance and angular measurements in a topological map has been addressed by several researchers, including Duckett *et al.* [14], Lu and Milios [36], and Golfarelli *et al.* [21]. Any of these methods suffice; the reference frame for the vertices can be placed arbitrarily.

3.6.1 Forays

Suppose that we would like to add a path connecting vertex v_i in our map with vertex v_j, where v_i and v_j are not already connected in the basic map. If our robot experienced no movement error, we could do this simply by initiating a behavior for which the robot (at v_i) turned toward v_j and drove straight until it reached its target. Unfortunately, we must allow for error, both in our basic map and in the robot's "foray" through open space.

We already have a means for estimating the relative uncertainty of vertex locations, using uncertainty distributions based on the robot's error model and centered at each vertex. Confidence limits specify a region within which we believe (with the desired confidence) the vertex lies.

We assume that the "diffusion" of the error in the robot's location estimate,

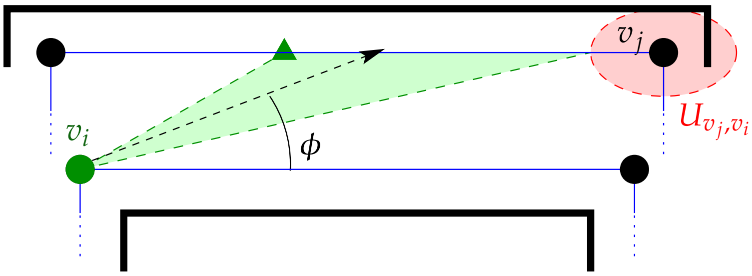


Figure 3.4: A refinement from vertex v_i to vertex v_j. The robot turns to angle ϕ and "forays" toward the opposite wall. The robot's angular uncertainty (highlighted) is small enough that it is confident it will land on the target wall.

as it forays from v_i, can be bounded by an angular range originating at v_i. The robot's positional uncertainty grows with the length of the foray.

There are two potential paths from any vertex v_i to some other vertex v_j, corresponding to the two walls incident to v_j. For each path, the robot forays in such a way that it lands on the desired wall. So, a valid refinement is possible only when the positional uncertainty accumulated by the robot during a foray is less than the total length of the target wall (between the confidence bounds in the locations of its endpoints). For an example foray, see Figure 3.4.

Refinements are represented as edges between vertices in the map. Associated with these edges is the following "behavior sequence":

1. Turn ϕ degrees toward the target wall (incident to the target vertex);
2. MOVE-TO-WALL;
3. WALL-FOLLOW in the direction of the target vertex.

We can make two types of refinements with this approach: *passive* refinements, which can be computed directly from the basic map, and *active* refinements, which require further exploration of the world.

3.6.2 Passive refinements

Passive refinements use knowledge of the area "swept out" by the robot's sensors as it performed its initial exploration. The union of all the area swept out by the robot's sensors is the "known space." A simple approximation of the known space can be obtained by storing the shortest range detected by the sensor opposite the wall-following sensor when traversing each wall.

Using the geometry of our basic map, we can immediately determine which candidate refinements pass only through the known space, and add these refinements to our map.

3.6.3 Active refinements / Exploration

Potential refinements that pass partially or entirely through unknown space constitute "exploration targets." We must actively explore the paths suggested by these refinements in order to verify that they can be traversed; there may be obstacles or "islands" in the unknown space.

Suppose that, when foraying, the robot encounters an obstacle. In this case the robot *must* be able to safely return to a known place in its basic map — otherwise, it is "lost" and will have to rebuild its basic map! (Alternatively, the robot could use pattern-matching techniques, or an evidential approach similar to the one we use in closing loops in our basic map, in an attempt to localize itself when it thinks it has returned to the basic map.) In general, prior to making forays, we discard those that are so lengthy that the robot may not be be able to return to a known place because of accumulation of uncertainty.

We compute a sequence in which to explore the remaining exploration targets, e.g. using a traveling-salesman like approach. We then visit each exploration target, and perform the associated behavior sequence.

If we encounter no obstacles until we reach our target wall, we keep the refinement; otherwise we discard it. Depending on the robot's error model, we may have leeway to explore an obstacle we encounter; if so, we create a basic map of it, and add refinements connecting it to our original map.

3.7 Limitations

Our mapping approach is not without limitations. In particular, our robots are limited in the worlds they can successfully map. These limitations are primarily dependent on the robot's sensing deficiencies; for example, we should not expect a robot with large odometry error to be capable of efficiently making complete maps of vast open spaces. While creating precise maps of large spaces with a sensing-limited robot certainly seems possible, our approach tends toward mapping and navigation efficiency rather than completeness.

Our robots do not capture every feature of the environment. In fact, the details of small features are explicitly "smoothed out" in favor of capturing the overall structure of the environment. This mitigates the risk of sensing inaccuracies, which occur more frequently with small features.

The "perceptual aliasing" problem — the inability to detect small features because of low sensor density — is another issue when creating maps with sensing-limited robots. There are situations in which the sparseness of the robot's sensors allows the robot to encounter an obstacle without first sensing it. For example, the robot may bump into a chair leg that "falls between the cracks" of its range sensors. For such situations, bump sensors may be necessary in order to create accurate maps. Bump sensors can be thought of as extremely short-range sensors, and can be incorporated into the mapping algorithm accordingly.

CHAPTER 4
MERGING TOPOLOGICAL MAPS

Suppose we release two robots without a common reference frame in some environment and allow them each to build partial maps of the world. We would like to construct a single, unified map from the two robots' maps by finding the overlapping regions between the separate maps. The ability to merge separate maps efficiently is important for multi-robot systems that must quickly and accurately build large maps of unknown environments.

Recall that topological maps use a graph to represent possibilities for navigation through an environment; vertices represent certain "places" in the environment and edges represent paths (or classes of paths) between these places. Often vertices and edges are annotated with metric information, such as path length (for edges) or relative orientations of incident paths (for vertices). Typically, vertices are junctions of two or more straight hallways, and edges represent a path down a hallway from one junction to another. Topological maps provide a concise description of the environment specifically geared towards navigation.

Robots that have explored overlapping regions of an environment should have topological maps that have common subgraphs with similar structure. Solving the map merging problem is thus analogous to identifying a matching between subgraphs in the two maps. In general, we would expect exactly known attributes of matched vertices, such as the degree of the vertices, to match perfectly. However, attributes of vertices and edges that are subject to measurement error, such as the length of an edge or the orientation of edges leaving a vertex, must only match closely.

If a robot's topological map contains edge length measurements and incident edge orientations at vertices, there is enough information to compute estimated vertex locations with respect to the robot's world frame. This suggests that the map merging problem could also be solved using image registration techniques. The most widely used algorithms for image registration are iterative closest point (ICP) algorithms. An initial matching between feature points must be provided; the algorithm first estimates a transformation between the two feature sets by minimizing the (weighted) squared error between corresponding features. Next, the feature matching is expanded by finding features that are close together under this transformation, and the transformation is re-estimated. This process continues until the change in the transformation estimate between iterations is small.

In this chapter, we describe an algorithm that uses the structural aspect of subgraph matching and the geometric aspect of image registration. We first create hypothesized matches by pairing compatible vertices in the two maps, and then locally grow each match using only the graph structure and vertex and edge annotations. Many hypotheses can be eliminated in this phase because of incompatibilities in the structure of the maps, or because of incompatible annotations (such as edge lengths that are too different). After growing the consistent matches, we estimate geometric transformations for each hypothesis and perform clustering in the transformation space. The result of our algorithm is an ordered list of clusters, where each cluster is a set of hypotheses for which we can estimate a single transform that has acceptable squared error between all the matched vertices in the cluster.

4.1 Assumptions

The maps to be merged must be consistent: no two vertices may represent the same "place." This means that the robots can either recognize or infer when they have revisited a place and thus are able to "close the loop." (For detailed discussion of map consistency and the loop-closing problem, see the previous chapter. Note though that our map merging algorithm is for use with general topological maps, not just the maps created by the algorithm described earlier.)

The robots must record enough information that a vertex embedding in the plane can be calculated from the map. Knowledge of path shapes and lengths and the orientations of paths leaving each vertex is sufficient to meet this requirement. We assume there is a known error model for measurements and that, errors notwithstanding, only a translation and rotation (but no scaling) is necessary to merge the maps.

Although most of our examples are from rectilinear worlds, the algorithm we develop here will work with any two-dimensional topological maps. Generally speaking, additional computation is required for non-rectilinear worlds because of unrestricted rotation.

4.2 Hypothesis building

The first phase of our algorithm creates hypotheses by locally "growing" single-vertex matches. A hypothesis is a list of vertex and edge correspondences between two maps; finding them is, in essence, the problem of finding all maximal common connected subgraph matchings between the two maps.

4.2.1 Vertex matching

We start with a pair of compatible vertices, one from each map. Two vertices are evaluated for compatibility by examining their attributes: exactly known attributes (such as vertex type) must match perfectly, but inexactly known attributes (such as edge length or edge orientations at a vertex) must be compared using a similarity function.

It often makes sense to assume that the robots will know the degree of vertices exactly; in general, robots should easily be able to determine the number of paths leading from a vertex. In dynamic worlds, where the degree of vertices may change — due to opened or closed doors, for example — vertex degree cannot be treated as an exact attribute.

The relative edge orientations at a vertex are typically known with some uncertainty, so they must be compared using a similarity function. One method is to find the rotation between the two vertices that minimizes squared error between corresponding edge orientations. The amount of error, relative to the vertex degree, could be used as the similarity measure. We set some threshold on the amount of similarity required in order to decide whether to pair two vertices.

At the conclusion of this step, we have many matches, each consisting of a single pair of vertices and the correspondence between edges. Note that a single pair of vertices may be consistent under several different edge orientation correspondences; each is present as a separate match.

4.2.2 Growing matches

We now grow each match by testing corresponding pairs of edges leaving the match. If the edges are compatible and the vertices at the ends are also compatible, then they are added to the match. If the edges or vertices are incompatible, the entire match is rejected.

The vertices can be tested using the same criteria and similarity functions used to form the initial pairs. Edges may also have both exactly and inexactly known attributes. Typically, however, they will simply have a path length that must be compared using a similarity function. We set some threshold on how similar two edges must be in order to include them in the match.

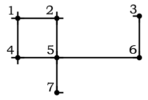
Our initial hypotheses are all the unique matches that survive the growing process. Though the approach described above creates duplicate matches, it is not hard to generate and grow only the unique matches; see the example below. Note that a subgraph in one map can be matched to multiple subgraphs in the other (under separate hypotheses), but a pair of matched vertices (with a given edge correspondence) can appear in only one hypothesis.

4.2.3 An example

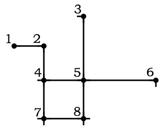
For an example of hypothesis generation, we use two maps from a rectilinear world, shown in Figure 4.1. The rectilinear world assumption implies that we know the exact orientation of paths leaving each vertex (relative to the robot's coordinate frame). For simplicity, we also assume that the world is static, so the degree of vertices must match exactly. In these maps, degree two and three vertices can match only with a single edge pairing, while degree four vertices can match with four different possible edge pairings. Edges are compared using the Φ-function discussed in Appendix B (which does double duty as part of the belief function from Chapter 3); a threshold value differentiates between acceptable

	b2	b3	b6
a3	×	×	H_1
a6	×	×	×
a7	H_2	×	H_3

Degree-2 Vertices



Map $\mathcal{A}$



Map $\mathcal{B}$

Map $\mathcal{A}$ rotation 90° 180° + 0° 270°	b4	b5	b7	b8
a1	× × + H_4 H_{11}	× × + × H_{12}	× H_9 + H_5 H_{13}	× H_2 + H_6 H_{14}
a2	× × + × ×	× × + H_4 ×	× H_2 + H_7 H_{11}	H_8 × + H_5 H_{12}
a4	× H_9 + × H_{12}	× H_2 + × ×	× H_{10} + H_4 H_{14}	× × + × H_{15}
a5	× H_2 + × ×	× × + × ×	× × + × H_{12}	× × + H_4 ×

Degree-4 Vertices

Figure 4.1: Hypothesis generation example for two maps from a rectilinear world. Degree 2 vertices have only one edge matching, whereas degree 4 vertices have four possible edge matchings, listed in terms of the relative orientation of 0, 90, 180, and 270 degrees. An "×" indicates that there is no valid hypothesis with that vertex match. There are 15 hypotheses that result from these two maps.

and unacceptable edge matches.

To construct all unique hypotheses, we create tables representing all possible combinations of single-vertex pairings and incident edge correspondences. We then grow each of these matches by adding compatible incident edges and vertices. We will sometimes find that a match is inconsistent, most often due to vertex degree mismatch. Although this match will be discarded, we first grow it as much as possible so that the associated vertex pairs can all be marked as incompatible. This avoids re-growing the same match with a different initial single-vertex match.

For example, in Figure 4.1, when attempting to match a6–b3, we expand the match to include a5–b5. We can then see that the vertices a7–b6 do not match in relative edge orientations (and also that the edges a5–a7 and b5–b6 do not match because of significantly different lengths). However, we expand the match to include a2–b4, a1–b7, and a4–b8, and then mark all these pairings as invalid.

Note though that the vertices a7–b6 can still be paired (and in fact, lead to a valid hypothesis).

4.3 Transform estimation and clustering

Our hypotheses now consist of maximal matched connected subgraphs. These matchings represent a single overlapping area, but the two maps may have several separate overlapping areas. In this phase of the algorithm, we consider the geometric relationship between hypotheses.

4.3.1 Transform estimation

In order to estimate a geometric transform, we must first embed the vertices of the maps in the plane. The situation is the same as that discussed in Section 3.6 — though the maps may be structurally consistent, they might contain measurements that are not geometrically consistent because of error, making the maps unembeddable. As mentioned, this problem has been addressed by several researchers [15, 21, 36]. Any of the methods they present suffices; the reference frame for the vertices can be placed arbitrarily.

We can now estimate a transformation (translation and rotation) for each hypothesis using the vertex correspondences of the hypothesis. In image registration, this would typically be done using iteratively re-weighted least-squares, where the weights help make the method robust to outliers. It is appropriate for us to use an unweighted least squares estimation because corresponding vertices should be close together; large error indicates that a hypothesis is geometrically bad (despite being structurally good), and such a hypothesis should be discarded.

4.3.2 Clustering

We group hypotheses into clusters to determine which hypotheses are consistent with each other. The clustering is done on the basis of the hypothesis transform estimations. The rotation angle of each hypothesis should be scaled to allow meaningful comparison of distance between transformations.

Clustering requires some threshold distance to determine when two transforms are close enough to be compatible. This distance could be defined in terms of the map, e.g., a fraction of the minimum edge length in the hypothesis. There are a variety of techniques for clustering; we used a simple agglomerative clustering method in our implementation, as follows:

1. Start with each hypothesis in its own cluster.
2. Compare all pairs of clusters and mark the pair that is closest.
3. Compare the distance between this pair to the threshold distance:
   - If the distance is less than the threshold, group the pair into a single cluster and return to Step 2.
   - If the distance is greater than the threshold, clustering is complete.

Once we have a set of hypothesis clusters, we order these clusters in terms of their "quality." This is not straightforward without information about the size, complexity, and self-similarity of the environment. We suggest the following heuristics and methods:

- Matches with only one or two vertex pairs are generally not significant.
- Total number of vertices is a good primary indicator of cluster quality.
- The amount of error (i.e., total squared error under the cluster transform) is a good secondary indicator of quality.
- The number and size of hypotheses in a cluster can be used as a secondary quality indicator; for clusters of the same size, those consisting of fewer hypotheses are more desirable.
- There is always some tradeoff between size and quality of a cluster.

In our example, we sorted the clusters first by size and then by total squared error.

4.3.3 Example continued

For rectilinear worlds, the errors in orthogonal directions are decoupled. Embedding a map in the plane is thus reduced to two one-dimensional problems. We can find the maximum likelihood embedding under a Gaussian error model by treating each edge as a spring whose nominal length is the measured distance and whose spring constant is the reciprocal of the measurement variance.

For embedding vertices on the x-axis, we treat each horizontal edge as a spring. For edge i, the spring force is

$$F_i = -\frac{1}{\sigma_i^2}(x_j - x_k - l_k) \tag{4.1}$$

Here, x_j and x_k are the x-coordinates of the edge endpoints ($x_j > x_k$), and l_k is the measured length of the edge with σ_i^2 its variance. A force F_i acts at x_j, and a force $-F_i$ acts at x_k. We then write an equation for force equilibrium at each x-coordinate; one x-coordinate can arbitrarily be chosen to be $x = 0$, and no equilibrium equation is necessary there. If there are n x-coordinates, this results in a system of $n-1$ linear equations in $n-1$ unknowns which is easily solved using standard methods. The process is repeated with the y-coordinates of edge endpoints.

We estimate a transform for each hypothesis using a two-dimensional version of the point-based rigid registration algorithm described by Fitzpatrick *et al.* [20]. This involves computing the singular value decomposition of a two-by-two covariance matrix.

Figure 4.2 shows the resulting clusters from our example. Because of the rectilinear assumption, the transform space consists of a two-dimensional translation space for each of the four possible rotations.

4.4 Implementation issues

If a hypothesis cluster is deemed correct, it is fairly straightforward to merge the two maps. The estimates of path lengths can be updated by combining the measurements from the two maps for corresponding edges. The edge orienta-

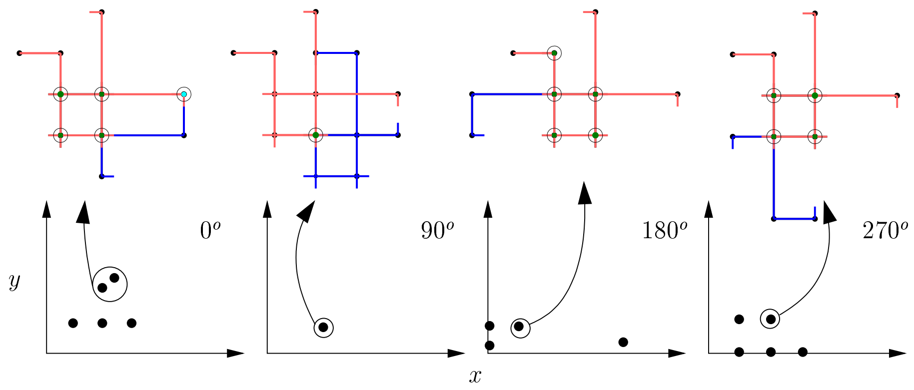


Figure 4.2: Transformation spaces and example clusters for our example maps. The 180^o, 5-vertex cluster (from a single hypothesis) is the best match; the 0^o, 5-vertex cluster consisting of two hypotheses is also a good match. Map $\mathcal{A}$ is shown in blue (dark), map $\mathcal{B}$ is shown in red (light), and matched vertices are circled.

tions at the corresponding vertices can be similarly merged. Portions of one map not present in the other should be added. However, we may not always *want* to merge the two maps according to our current "best guess."

4.4.1 Map storage

Even the best cluster choice may later turn out to be incorrect. For example, early in the process of exploring a self-similar environment, the robots might seem to be exploring the same area when in fact they are exploring similar but distinct areas. We must therefore consider how to merge and store maps so that incorrect hypotheses can be removed without discarding the entire map. In addition, we would like to be able to merge the maps of several robots, not just two.

We can think of a robot's map as being represented in several layers. Layer 0 should be a robot's own map, recording only the measurements that robot has taken. Layer 1 is used to store hypothesized matches (potentially several different ones) to other robots' Layer 0 maps. Layer 2 contains hypothesized matches to Layer 1 maps, and so on.

This approach yields a "dependency" structure that makes it straightforward to discard hypotheses that are later determined to be incorrect, along with all other hypotheses that depend on them. This layer structure does not require much additional storage, and if necessary the upper layers can be compressed into a single layer.

4.4.2 Computational issues

When the merging algorithm is used online, robots may have additional information that can be used to merge maps more quickly. For example, if two robots exchange maps only when they are within communication range, we can bias the search for hypotheses toward matches between vertices near both robots.

After merging maps once, two robots may merge their maps again at a later point in time. During subsequent mergings, an "incremental" map update can save a substantial amount of computation; this is possible with minimal book-keeping effort. The other robot's map, stored separately, can be updated with new or changed vertices and edges. We can use these updates to verify and expand hypotheses from a previous merging, and to form new hypotheses starting from the new vertices. Changed vertices or edges may give rise to new hypotheses or may eliminate hypotheses from the previous matching. All surviving hypotheses undergo the remainder of the merging process.

Although a map merging can be computed relatively quickly — in our rectilinear implementation, 200-vertex maps are merged in less than one second — it may be desirable to distribute the computation between the robots. The hypothesis building and transform estimation steps can easily be split between the robots by simply assigning sets of initial hypotheses to each robot. The transformation clustering, however, is done most straightforwardly on a single robot.

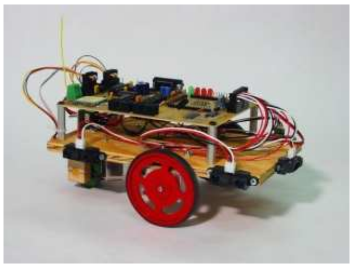
CHAPTER 5
EXPERIMENTAL RESULTS

We have implemented and experimented with both the single-robot topological mapping algorithm of Chapter 3 and the topological map merging algorithm of Chapter 4. In this chapter, the experiments that we performed and the results that we obtained are detailed and discussed.

5.1 Single-robot topological mapping

The topological mapping algorithm has been implemented both in simulation and on the mobile robot depicted in Figure 5.1. Both the simulated and real-world implementations satisfy the assumptions required by the core mapping algorithm (see Section 3.1). Among these are that the robot has left, right and front range sensors, and that an error model for the robot's odometry is (approximately) known. The experiments that were performed satisfied the assumption of an enclosed, static, rectilinear environment.

The majority of the experiments that were performed were to test the basic mapping algorithm, and in particular, the evidential-reasoning loop-closing method introduced in Section 3.3. The map refinements discussed in Section 3.6 were tested in simulation. Special handling of "portals" and narrow corridors



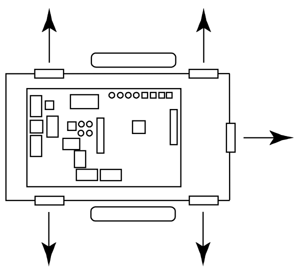


Figure 5.1: Our prototype robot is a differential drive mobile robot approximately 20 cm long. It has five Sharp GP2D12 infrared range sensors, 256 CPR encoders on each wheel, and an Atmel ATMEGA64 microcontroller as the main processor.

(see Section 3.4) was not implemented experimentally.

The error model that was used, both in simulation and with the hardware, assumes that the robot experiences odometry error that follows a Gaussian distribution. The standard deviation of this distribution, referred to as σ throughout this chapter, is assumed to be proportional to the length of measurements. Typically, σ is computed as some percentage of the measurement length. For example, under an error model assuming σ equal to 7% of the measurement length, the error distribution for a wall measured to be 10 m long has standard deviation $\sigma = 0.7$ m.

In all tests, the confidence limit specified to the algorithm (the α parameter) was 95%. The threshold belief value required to accept or reject a loop-closing hypothesis, β, was 99%.

5.1.1 Simulated results

The robot simulator is a simple kinematic simulator specifically designed to test the mapping algorithm with rectilinear worlds. The simulated robot is a point robot that implements move-to-wall and wall-following behaviors meeting the requirements discussed in Section 3.2. The wall-following behavior assumes that the robot experiences no range sensor error and is able to travel in a straight line while following walls (the real robot is able to use sensor feedback to maintain alignment). Under the same assumption, the robot experiences no orientation error.

The wall-following behavior returns the length and terminating conditions of each wall segment. Odometry error was simulated by perturbing the calculated distances with Gaussian error with standard deviation σ. In simulated experiments, σ was varied between 6% and 14% of measurement length for small experiments, and between 4% and 20% for more complicated experiments. The mapping algorithm used the same value of σ as was used by the simulator to generate noisy data. Though the belief function used for loop-closing is independent of σ, the generation of hypotheses is not, because of the fixed values of r_0, $r_{\min}$, $r_{\max}$ and r_*. Varying σ thus provides some insight into the susceptibility of the mapping algorithm to incorrect hypotheses, particularly in complicated environments.

5.1.1.1 Experiments

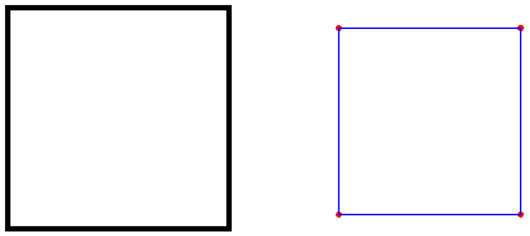
The basic mapping algorithm has been tested in the simulator with a number of simple worlds. Figure 5.2 shows four such worlds in order of increasing complexity. For these environments, the robot made a number of maps under increasingly difficult error conditions. The results were generally as expected: 97% of the maps were structurally correct after the robot accepted a loop-closing hypothesis. If the robot rejected the correct hypothesis, the map was immediately marked incorrect. Of the incorrect maps, all but two were due to rejecting the correct hypothesis after the robot observed a sequence of outlying (improbable) measurements. The remaining incorrect maps were generated when the robot failed to discover the correct hypothesis — the start vertex fell outside the robot's 95% confidence bounds on its location, when the robot had in fact closed the loop.

Several experiments have been performed to test the loop-closing method's susceptibility to incorrect hypotheses in self-similar worlds. As discussed in Section 3.3, these worlds are the most difficult in which to close loops. The basic mapping algorithm fares well in most cases, though we have been able to construct some pathologically self-similar environments for which the algorithm sometimes make incorrect decisions about a loop-closing hypothesis under certain error conditions.

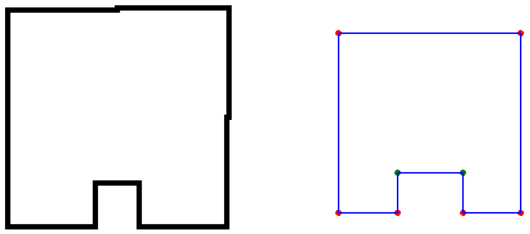
Figure 5.3 shows results for the mapping of a "spiral"-like self-similar world. A number of trial maps were generated of this world under varying degrees of induced error. Most of the time, the evidential-reasoning approach was successful in choosing the correct loop-closing hypothesis. Under extremely harsh error conditions, with error standard deviation greater than 10% of measurement length, the correct hypothesis was chosen about 87% of the time in our trials. In general, the algorithm fares well with these "spiral"-like worlds, as long as the spirals are not extremely "tight."

Figure 5.4 shows the results of mapping a world with "repeating similar

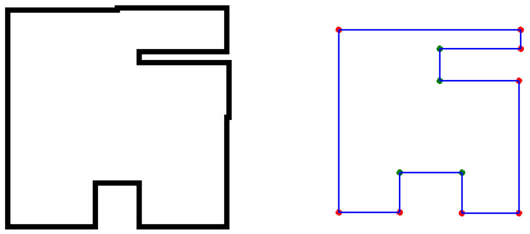
Environment | **Example map** | **Mapping statistics**



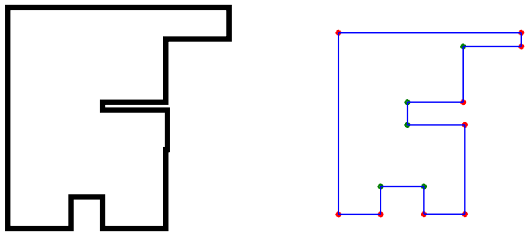
σ	Correct	Incorrect
6%	10	0
8%	10	0
10%	10	0
12%	10	0
14%	10	0



σ	Correct	Incorrect
6%	10	0
8%	10	0
10%	10	0
12%	10	0
14%	9	1

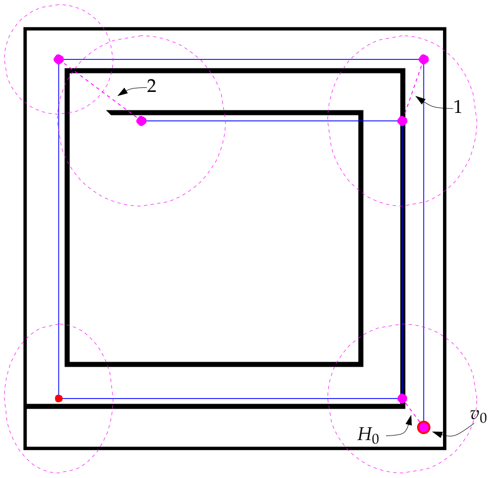


σ	Correct	Incorrect
6%	10	0
8%	10	0
10%	10	0
12%	10	0
14%	8	2



σ	Correct	Incorrect
6%	10	0
8%	10	0
10%	9	1
12%	9	1
14%	9	1

Figure 5.2: Mapping simple simulated worlds. Each environment was mapped over a number of trials under various error conditions. The tables show the frequency of correct maps (for which the right loop-closing hypothesis was accepted) versus the frequency of incorrect maps (for which either the wrong hypothesis was accepted or the correct hypothesis was rejected). Maps were made for several values of σ, the standard deviation of the measurement error; σ is defined in terms of a percentage of the true length of the measurement.



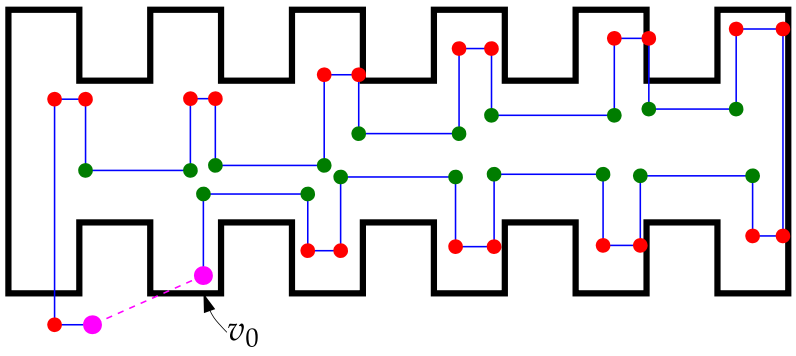
Step	$m(C)$	$m(I)$	$m(\{C,I\})$
0	0.202	0.000	0.798
1	0.007	0.975	0.019
2	0.000	0.999	0.000

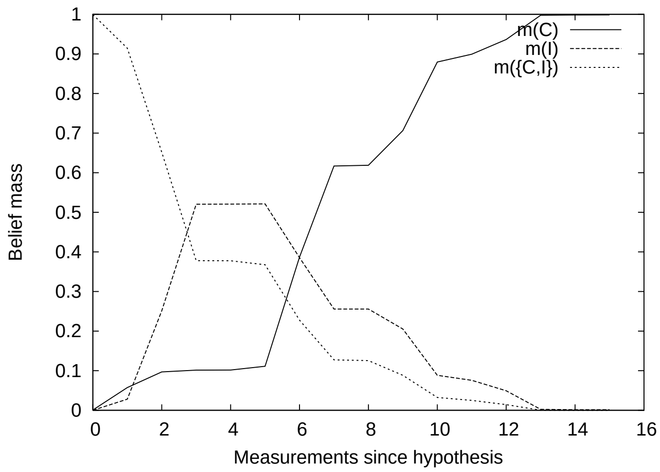
σ	Correct	Incorrect
4%	10	0
6%	10	0
8%	10	0
10%	9	1
12%	9	1
14%	9	1
16%	8	2
18%	9	1
20%	8	2

Figure 5.3: Mapping a "spiral"-like self-similar world. Ten trials were performed for each of several different amounts of error (right table). In all trials, the robot began at the same starting configuration, so that the start vertex was always the same. A representative case is pictured. Here, vertex matchings according to an incorrect hypothesis are shown as dashed lines between the vertices. 95% confidence bounds on the vertex locations are displayed as ellipses. The true start vertex is at the extreme lower right (large dot); an incorrect loop-closing hypothesis was generated after making one "lap" around the spiral. After taking two more wall length measurements, the algorithm held 0.99 belief that the hypothesis was incorrect. The table below the map shows the global belief function values after each new measurement since the hypothesis H_0 was made. Step 0 indicates the initial belief function values (see Appendix B for discussion).

sequences" of walls. These worlds are more difficult for the mapping algorithm, and this particular environment defeats the algorithm more often than most. The difficulties arise when the robot mistakes one "well" for another when attempting to close the loop. The pictured case is the most difficult, because the start vertex is such that the robot must continue its traversal for a long time before encountering a structural mismatch after making an incorrect hypothesis.

Under relatively benign error conditions, the robot is generally able to close the loop properly, because the distance between "wells" is larger than its error





σ	Correct	Incorrect
4%	10	0
6%	10	0
8%	9	1
10%	7	3
12%	4	6
14%	5	5
16%	1	9
18%	1	9
20%	3	7

Figure 5.4: An environment that defeats the evidential-reasoning loop-closing method under certain error conditions. Ten trials were performed for each of several different error levels (table). In all trials, the robot began began at the same starting configuration, so that the start vertex was always as pictured (the worst-case scenario). A representative case is pictured. Here, the start vertex is in the "well" that is second from the left on the bottom. By the time the robot circumnavigates the world to reach the leftmost well on the bottom, it mistakes this for the start and generates an incorrect hypothesis (shown as a dashed line between the matched vertices). Because of the extreme self-similarity of the portion of the environment containing the start vertex, the belief $m(C)$ in the correctness of the hypothesis exceeds the required threshold $\beta = 0.99$ before the robot encounters a structural mismatch. The plot shows the evolution of the global belief function as evidence is gathered after the hypothesis is made.

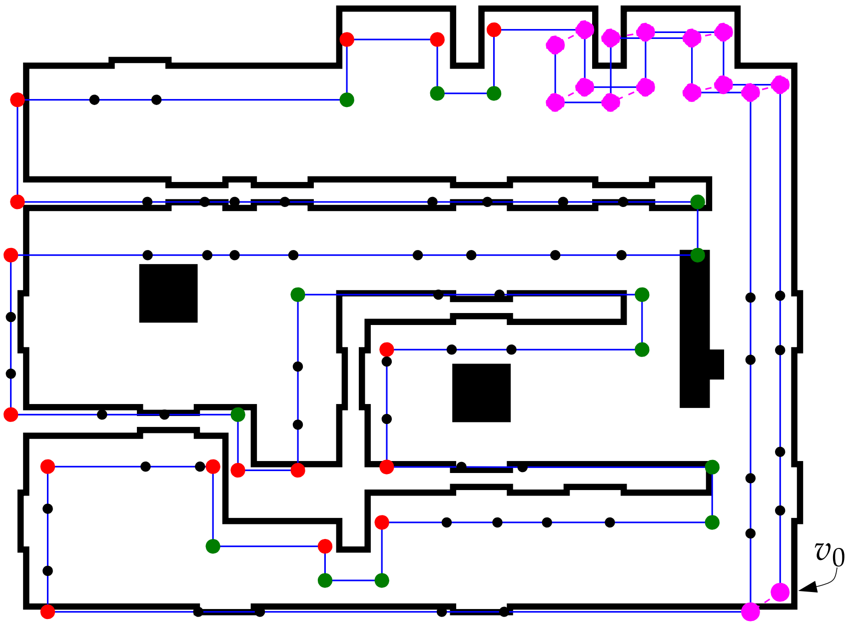
accumulation when it returns to the start vertex. Once the robot experiences error such that this is no longer the case, the loop-closing decisions it makes deteriorate rapidly.

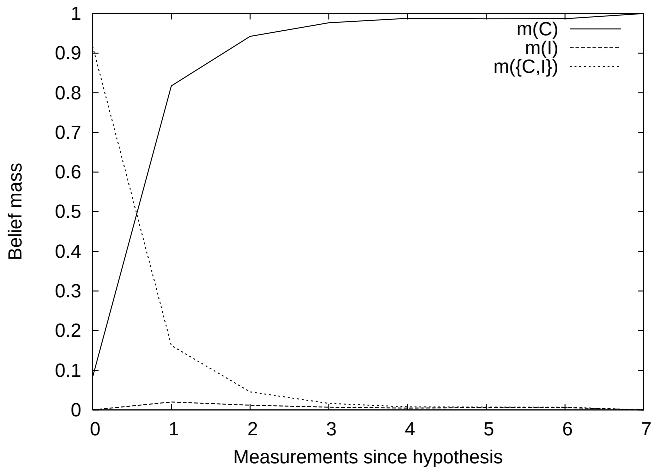
When simulating more realistic worlds, such as that shown in Figure 5.5, the evidential-reasoning approach tends to perform extremely well. Over a large number of trials in this environment, only once was an incorrect loop-closing decision made. This occurred in a situation where the robot experienced measurement error with standard deviation equal to 20% of the true wall length, and in which a series of outlier measurements occurred in sequence. In other testing, sequences of improbable outlying measurements have occurred occasionally; under such circumstances, correct hypotheses are sometimes rejected, or incorrect hypotheses are accepted.

The "refinements" to the basic map discussed in Section 3.6 add paths between map vertices by "foraying" through empty space. This approach has been tested in simulation, though not extensively. Figure 5.6 depicts a map of the world from Figure 5.5, with refinements to the basic map included. The figure was generated with a simulated robot experiencing no measurement error (refinements generated under significant error conditions tend to be visually abstruse because they cannot go directly between vertices). Refinements have been tested under various error conditions by varying σ. As expected, for significant angular or odometric error, only refinements between vertices adjacent to lengthy walls can be made. Typically however, just a few refinements are enough to greatly improve the navigability of the map.

5.1.2 Real-world results

The basic mapping algorithm has been implemented on robot hardware and tested in various indoor environments. The robot is a differential drive robot with 6.7 cm diameter wheels driven by inexpensive gearhead motors, with a passive roller-wheel (instead of a caster wheel) in the back for lateral stability. Each wheel has a 256 CPR quadrature encoder. A 20 MHz PIC 16F873 microcontroller is responsible for feedback control of the motors. The main processor on the robot is





σ	Correct	Incorrect
4%	10	0
6%	10	0
8%	10	0
10%	10	0
12%	10	0
14%	10	0
16%	10	0
18%	10	0
20%	9	1

Figure 5.5: Topological map of a large simulated environment. The pictured map was generated with measurement error standard deviation equal to 7% of the true wall length. The correct loop-closing hypothesis is shown with dashed lines between matched vertices. The convergence of the global belief function is shown in the plot; the robot needed seven measurements to reach $m(C) = 0.99$. Ten trials were performed with this world for each of several different amounts of error. For each trial, the robot began at a random starting configuration. Only once, under extremely harsh error conditions, did the robot make an incorrect decision about a loop-closing hypothesis (it rejected the correct hypothesis).

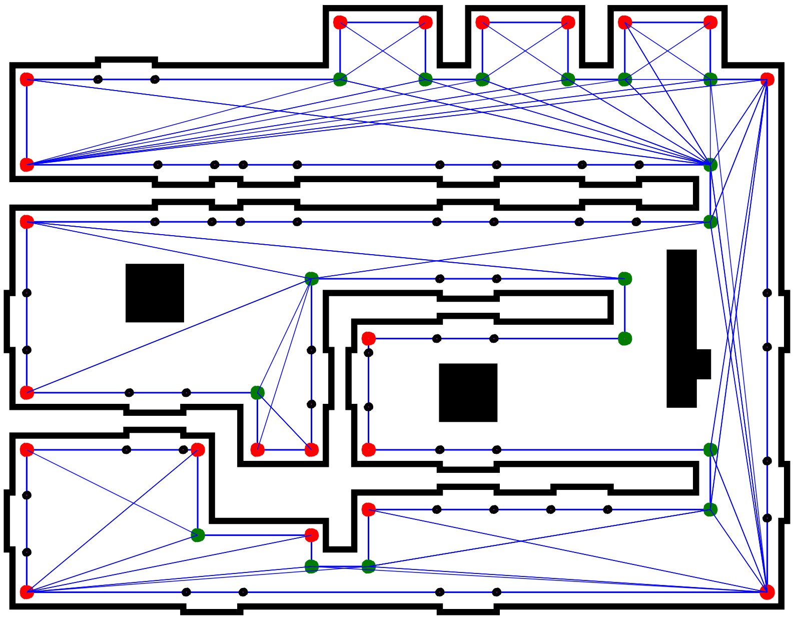


Figure 5.6: Map of the world from Figure 5.5, with refinements to the basic map included. Maps of "islands" were not generated. This map was generated with no error in order to illustrate an "ideal" map.

a 16 MHz ATMEGA64 microcontroller with 64 KB of program memory and 64 KB of data memory. The robot is powered by two 9V NiMH rechargable batteries that last approximately 1 hour on a single charge. The robot communicates with a base station using a Honeywell HRF-ROC09325XM radio module, which transmits data at rates up to 19.2 Kbps on a 900 MHz frequency. The five range sensors on the robot are Sharp GP2D12 infrared range sensors with a range of 10–80 cm.

In our real-world experiments, it was assumed that the robot experienced measurement error following a Gaussian distribution with σ equal to 7% of the measurement length. This value was determined based on early experimental data that indicated the true standard deviation of the error experienced by the robot was between 5% and 10% of the distance traveled.

The robot's computing was used only to execute a wall-following behavior repeatedly. Measured wall lengths (computed as the averaged number of encoder ticks for the two wheels) and terminating conditions were communicated to the base station using the radio. The actual generation, rejection and acceptance of loop-closing hypotheses was computed offline from data collected by the robot. In collecting its data, the robot paused after discovering each new vertex so that data transmission could be verified; after a button (on the robot) was pressed, it resumed its exploration. The robot was otherwise unaided in its data collection.

All of the environments explored by the robot were enclosed and static. For the most part, they were rectilinear in nature; non-rectilinear features were all small enough to be ignored as "ill-defined" features. There were also a number of rectilinear ill-defined discontinuities, such as recessed doorways, that were ignored (correctly) by the wall-following behavior.

As with the simulations, the hardware implementation of the mapping algorithm was used primarily for testing the evidential loop-closing method. Despite the sparse sensing of the robot and the often large errors it experienced, the algorithm was able to close loops correctly in most circumstances.

5.1.2.1 Experiments

The robot was first used to make maps of several small-scale environments, shown in Figure 5.7. The robot mapped each of these environments several times, though only individual results are presented here. There were no cases in which the evidential-reasoning loop-closing method failed to close the loop properly.

In part, the success of the loop-closing method here was due to the small size of the environments. Since the robot experiences odometry error with standard deviation approximately equal to 7% of the distance traveled, it can be reasonably certain in small worlds that any hypothesized loop-closing is correct (because its confidence bounds never grow much beyond r_*). Because of this, the robot's initial belief in the correct hypothesis for the map in Figure 5.7(d) is quite large, as shown in Figure 5.7(e). A complete discussion of how the initial belief in hypotheses is determined is available in Appendix B.

The robot was next used to map a more extensive area. Figure 5.8 shows a large map, created by the robot, of the corridors (and one room) on the first floor of an academic building, Amos Eaton at RPI. Note that a number of small features visible in the floor plan — mostly recessed doorways with closed doors — are

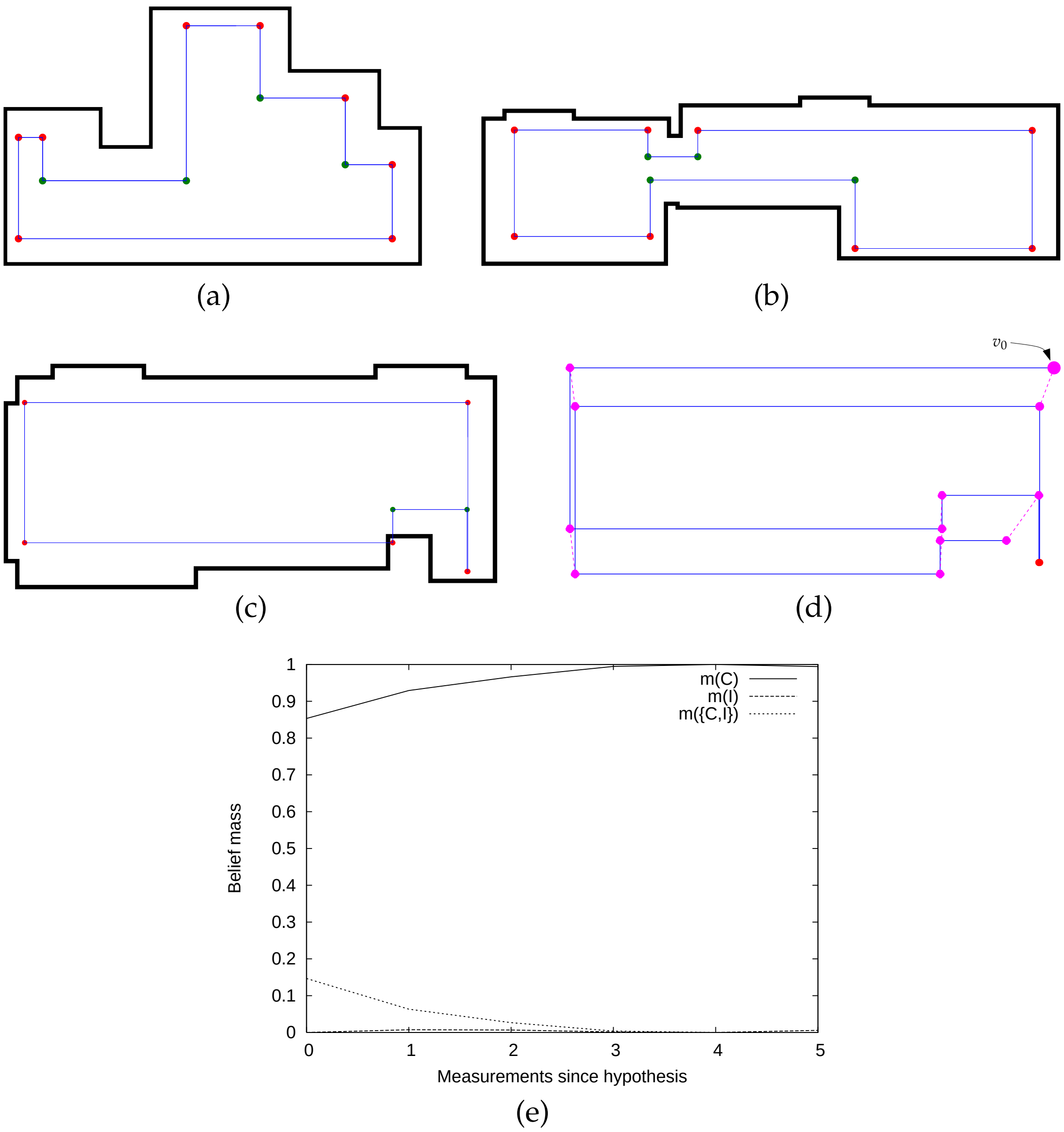


Figure 5.7: Basic maps of small-scale environments created with the robot from Figure 5.1. In Figures (a), (b) and (c), the final maps (after loop-closing and embedding) made by the robot are overlaid on the floor plans. In Figure (a), the environment is approximately 3 m $\times$ 2 m; the robot needed three measurements to accumulate 0.99 belief in the correct loop-closing hypothesis. In Figure (b), the environment is approximately 7.5 m $\times$ 2 m; the robot needed four measurements to accumulate 0.99 belief. Figure (c) shows the map made of a 4.5 m $\times$ 2 m world; the raw data for this map is displayed in Figure (d). Dotted lines are drawn between vertices that match under the correct hypothesis; the start vertex is the enlarged dot at upper right. Figure (e) shows the convergence of the belief function for the correct hypothesis; the initial belief in the hypothesis is large — about 85% (see Appendix B for discussion).

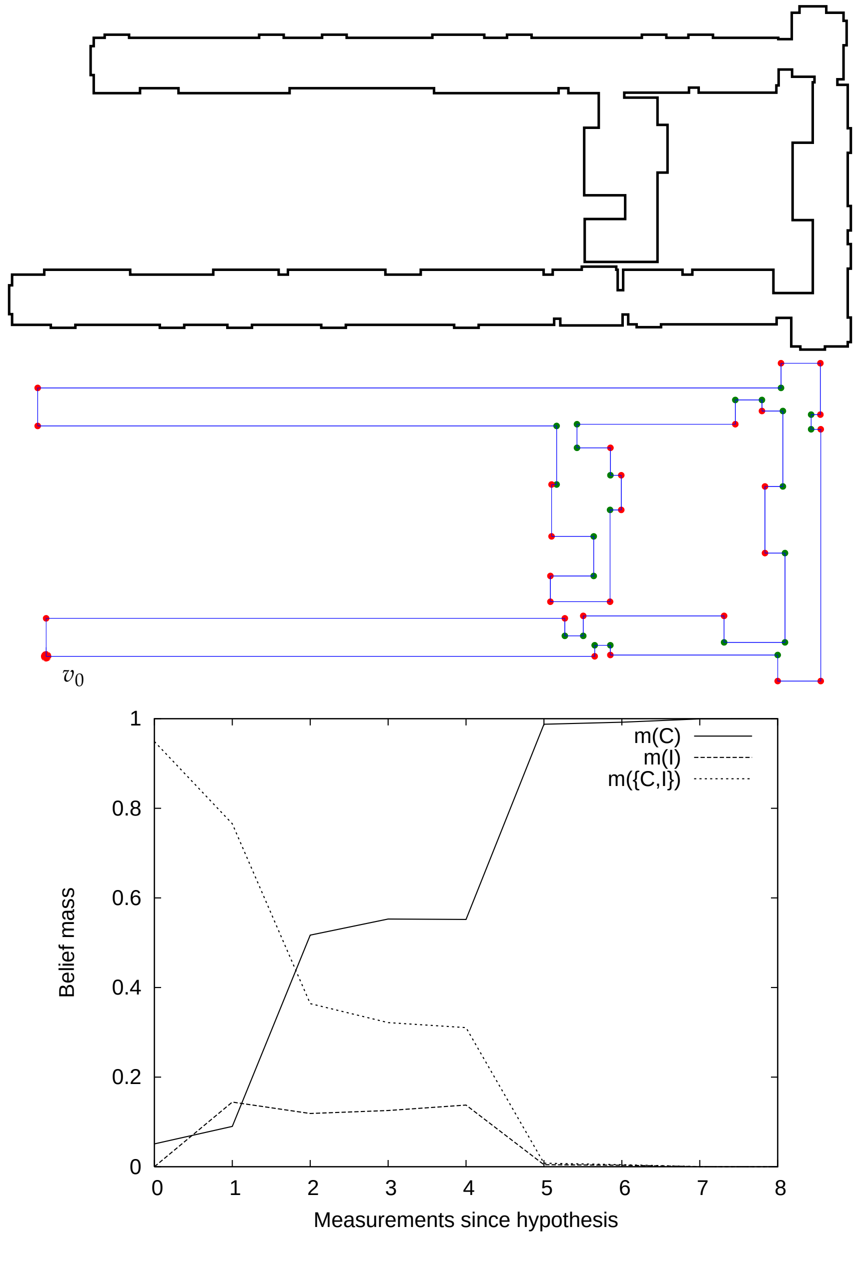


Figure 5.8: Basic map, after loop-closing and embedding, of the corridors on the first floor of an academic building (Amos Eaton) at RPI, constructed using a wall-following behavior. The precise floor plan of the 12 m × 30 m area is shown in the topmost graphic. In the map, many of the "details" of the walls, such as recessed doorways, were smoothed away as ill-defined features. Six measurements were needed to verify the correct loop-closing hypothesis (the start vertex is the large dot at bottom left). The convergence of the global belief function is shown in the plot.

"smoothed away" in the robot's map. Also observe that the robot experienced the most odometric error when measuring the longest walls. For example, the bottommost hallway is in reality longer than the topmost; due to odometry error, the robot's map does not capture this correctly.

5.1.3 Discussion

Overall, our simulated and real-world experiments indicate that the mapping algorithm works as expected in most cases — well over 95% of the time for non-self-similar worlds. Most of the experiments have focused on testing the evidential-reasoning approach to loop-closing. We have been able to defeat this approach by constructing pathological environments in simulation, but it has worked well in all of the real-world tests that have been performed. In addition, over a large number of trials in various simulated environments, the algorithm has constructed structurally correct maps in nearly every case.

In both the simulated and hardware implementations, we have made the simplistic assumption that the robot experiences odometric error with standard deviation proportional to the distance traveled. When testing in complicated environments, we have seen that the loop-closing approach is sensitive to large inaccuracies in the error model — in particular, under-estimated error variance, which can cause the robot to miss valid loop-closing hypotheses because of too-tight confidence bounds on its location. The most common problems occur, however, when a series of measurements with improbably large error are taken. Such an occurrence can cause the robot to not generate the correct loop-closing hypothesis (when the start vertex falls outside the robot's confidence bounds) or to accept an incorrect hypothesis.

The idea of "refinements" that further connect places in the basic map using forays through open space has been tested only in simulation. The idea appears sound, but simulations indicate that exploring refinements with real robots may be laborious: there are typically a large number of potential refinements, since each vertex can usually connect to several others. A possible improvement might be to try to explore multiple refinements simultaneously: often, several refinements pass through space that can be "swept out" by the robot's sensors as it explores a similar refinement.

We have not extensively tested a hall-following behavior or the idea of "portals" between tight corridors and open spaces. Though these ideas are not central to the mapping algorithm, they may improve mapping efficiency significantly.

5.2 Topological map merging

The topological map merging algorithm has been implemented for rectilinear worlds, and tested in simulation using hand-made and randomly generated environments, with maps constructed under varying degrees of induced error. It has also been tested using real-world data collected during experiments with the single-robot mapping algorithm. The results indicate that the merging algorithm is very effective, even for large maps with small overlap.

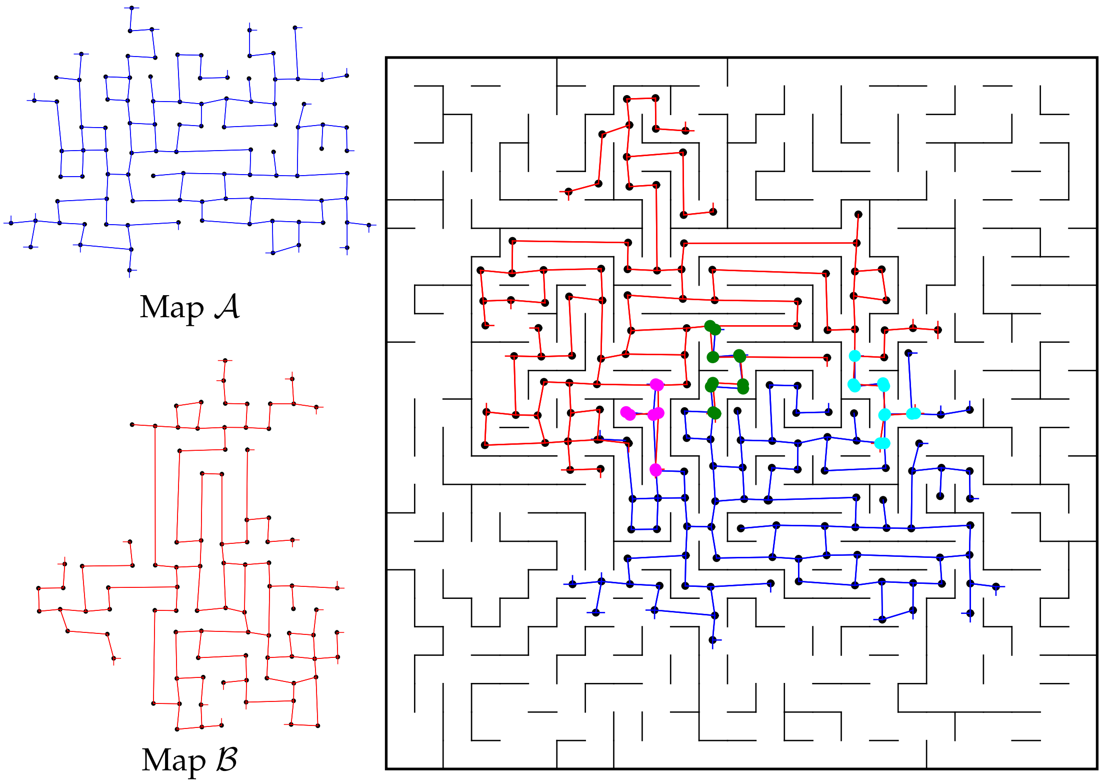
Our experiments assume the robot is able to determine the degree of map vertices upon discovering them. Thus, in all of the map merging experiments, two vertices are considered potential initial matches if they are of the same type and degree. Thus, for vertices of degree four each match yields four possible correspondences — one for each possible orientation in a rectilinear world. Vertices of degree less than four yield only one possible correspondence.

All map merging experiments with both simulated and real data were performed using a laptop with a 650 MHz Pentium 3 processor and 192 MB RAM, running Linux 2.4.19.

5.2.1 Simulated results

The simulator, implemented separately from the single-robot mapping simulator, generates large random maze-like worlds with loops. It then generates partial maps of the worlds by picking random starting locations and performing a breadth-first search to a specified depth, using a hall-following like "behavior."

Error is introduced into the maps by randomly perturbing vertex locations according to a Gaussian distribution with configurable standard deviation. Optionally, error introduced by perturbing a vertex is "propagated" through the



Single-vertex correspondences:	3918
Hypotheses after growth:	72
Clusters:	69
Best cluster:	3 hypotheses, 16 vertices
Running time:	0.04 seconds

Figure 5.9: Results of merging randomly generated simulated maps. Matched vertices are shown with large dots, and different matches have different colors/shades. The best matching occurs when map $\mathcal{B}$ is rotated approximately -90^o, and consists of three separate hypotheses that are clustered together.

maps by similarly perturbing its neighboring vertices, in order to simulate somewhat more realistic error conditions. Error propagation is "outward" from the root of the breadth-first search tree from which the map was generated (an absolute ordering of vertices is specified for perturbation so that error propagation does not affect "already-visited" vertices).

After generating partial maps, the merging algorithm is run with these maps to produce correspondences.

Figure 5.9 shows the matching found for two randomly generated maps in a maze-like world. The two maps are shown after matching, overlaid on the maze from which they were generated. The maps overlap in several "structurally disconnected" places; the final result of the matching process is a cluster of three consistent hypotheses. Note that there is another small, two vertex match that appears to be consistent with this cluster, just to the left of the leftmost (magenta) highlighted hypothesis. A conservative threshold on intra-cluster distance (in transformation space) prevented it from joining the other three hypotheses in the best cluster.

A larger example, generated using more noise and with some error propagation in the individual maps, is shown in Figure 5.10. Here, the two maps, each with more than 450 vertices, are properly matched with a cluster of two hypotheses, despite their relatively small overlap. The total running time to produce the match was under half a second on a 650 MHz Pentium 3 processor. Because of measurement errors, the best-guess geometric transform between the two maps is skewed somewhat; embedding the resulting map under the rectilinear assumption improves the results slightly.

5.2.2 Real-world results

Unfortunately, it was difficult to extensively test the map merging algorithm with real-world data — most maps produced by our robots were too small to constitute meaningful trials. The data obtained for the map shown in Figure 5.8, of the Amos Eaton building at RPI, was the largest available.

Rather than create multiple partial maps of the building (a laborious process) the data from Figure 5.8 was split into overlapping partial maps. Without measurement differences, merging the maps produced this way was trivial. To produce more interesting results, artificial error was introduced into the overlapping parts of the map by randomly perturbing vertices, in similar fashion to the method used in creating simulated maps. The resulting partial maps were then merged.

Figure 5.11 shows matchings between two sets of partial maps of the Amos Eaton building. Note that with this real-world data, there are very few remaining hypotheses after the structural growth phase of the algorithm. To some extent this is due to the relatively small sizes of the maps. In part though, the low number of hypotheses occurs because most real-world environments present few regions with structural self-similarity for maps with meaningful overlap; also, There are typically many structurally distinctive features in real-world environments.

5.2.3 Discussion

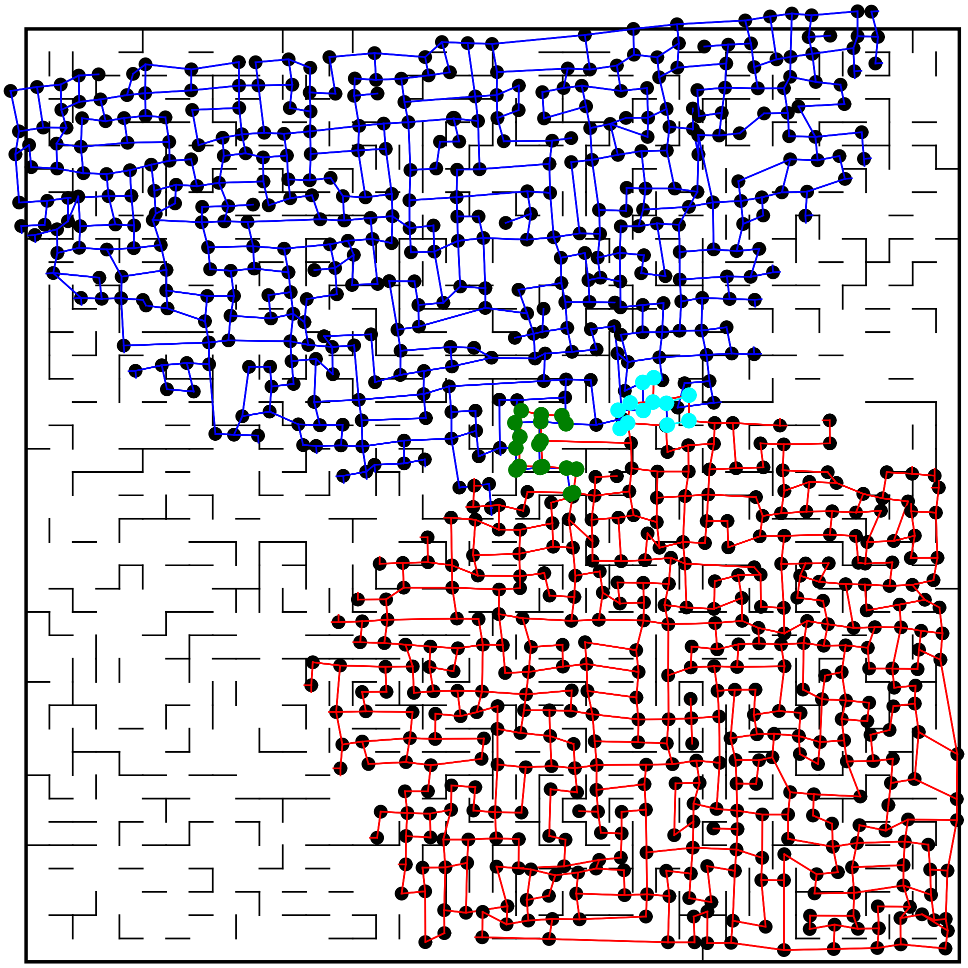
The map merging algorithm has proven quite successful for both simulated maps and the real-world maps we have been able to test. Much of this is due to the structural matching performed by the algorithm. Though structurally self-similar worlds (like those that cause difficulties for the single-robot mapping algorithm) can sometimes lead to mismatches, these mismatches are easily discarded as long as both maps being merged overlap in areas contain structurally significant features.

For large maps with greater than 100 vertices, there were generally several thousand single-vertex correspondences. After the topological growth process, the number of remaining hypotheses was less than 100; typically, there were only one or two large hypotheses containing more than three vertices.

Clustering on these matches worked well, but occasionally consistent hypotheses that were far apart in world space were also far enough apart in transformation space, because of map deformation due to error, that they were not grouped together into the same cluster. This was primarily due to the use of a conservative threshold on intra-cluster distance in transformation space.

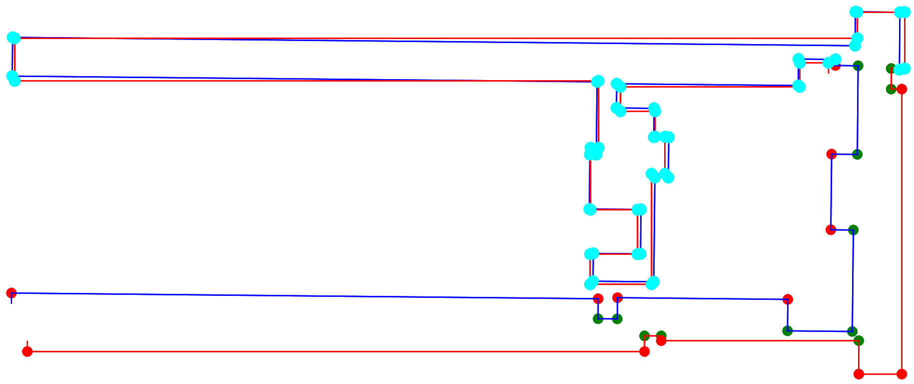
The estimated geometric transformations for correct clusters sometimes introduces significant “skew” into the merged map, as seen in Figure 5.10. The skew is largest for maps with small overlap; maps with large overlap provide more data for use in transform estimation. Note that estimated transformations can be improved as individual robots expand their own maps and compare their new data with the merged map.

Even for large maps, the matching process was very quick, usually under one second on a 650 MHz Pentium 3 processor.



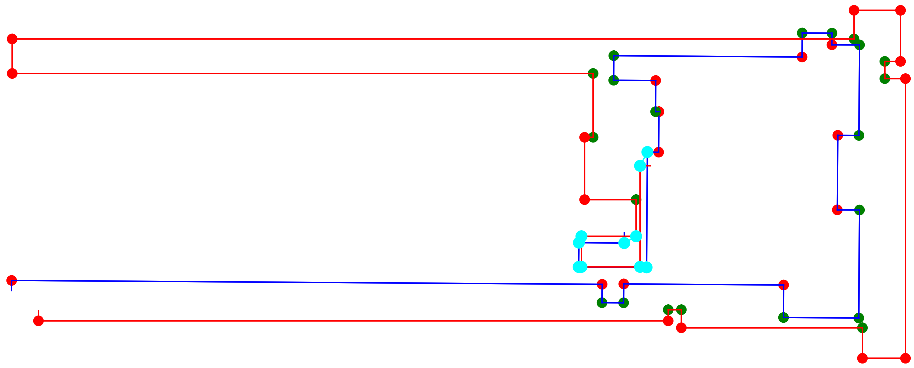
Single-vertex correspondences:	98372
Hypotheses after growth:	80
Clusters:	77
Best cluster:	2 hypotheses, 15 vertices
Running time:	0.43 seconds

Figure 5.10: A matching between large maps, each with approximately 450 vertices. The correct matching is a cluster of two hypotheses. The best geometric transform estimate for the matching leads to a slightly skewed result.



Single-vertex correspondences:	708
Hypotheses after growth:	3
Clusters:	3
Best cluster:	1 hypothesis, 25 vertices
Running time:	0.01 seconds

(a)



Single-vertex correspondences:	378
Hypotheses after growth:	4
Clusters:	4
Best cluster:	1 hypothesis, 5 vertices
Running time:	0.00 seconds

(b)

Figure 5.11: Matchings between partial maps of the Amos Eaton building at RPI. Matched vertices are the light-colored larger dots. Figure (a) shows a matching for partial maps with relatively large overlap; Figure (b) shows a matching for maps with comparatively smaller overlap. In both cases, the correct matching is found.

CHAPTER 6
CONCLUSIONS AND FUTURE WORK

This thesis has examined two aspects of the multi-robot mapping and exploration problem: single-robot topological mapping and topological map merging. It has focused primarily on the use of "sensing-limited" robots — robots capable only of sparse, short-range sensing. These robots are inexpensive and therefore potentially disposable, making them especially useful in situations that call for mapping by large teams of small mobile robots.

6.1 Single-robot topological mapping

In this thesis, a topological mapping algorithm for sensing-limited robots has been developed that allows individual robots to build maps of indoor environments. The algorithm takes an evidential-reasoning approach, based on the Dempster-Shafer theory of evidence, to the problem of "closing the loop" — recognizing when a robot has returned to a place it has already been. Additionally, the algorithm introduces ideas for incorporating open spaces into topological maps, using refinements to our map that make "forays" through such spaces, and "portals" that naturally divide the world into narrow regions and open spaces.

Experiments, both in simulation and using real-world hardware, verify the efficacy of the approach. Our robots have been successfully used to create topological maps of an academic building, and the idea of "refinements" based on forays through open space has been verified in simulation. Though in pathologically self-similar worlds it sometimes makes incorrect decisions, the evidential loop-closing method used by the mapping algorithm works very well in most practical circumstances, even in the presence of significant odometric error. In our tests, the correct loop-closing hypothesis was accepted more than 95% of the time in simulated environments; in real-world tests, the correct hypothesis was accepted in all trials.

6.2 Topological map merging

This thesis has also presented an algorithm for merging the topological maps of two robots without a common reference frame. The algorithm uses the structure of the topological maps to find a set of hypothesized matchings, and then uses the geometric transformations of these hypotheses to group them into consistent clusters. Possible heuristics for determining the quality of clusters have been described. Additionally, approaches for storing maps that are effective for the hypothesis-based merging, and that facilitate merging maps from more than two robots, have been discussed. Lastly, methods have been suggested for reducing the computational cost of the merging algorithm when robots "remerge" newly updated maps.

The map merging algorithm has been demonstrated on simulated maps. In our experiments, maps with meaningful overlap were always merged correctly. Even maps with minimal overlap are often merged correctly; for those that are not, the algorithm returns a set of consistent mergings. The algorithm's performance is quite good; for two maps each with about 450 vertices, the correct matchings between vertices were found in under half a second.

6.3 Future work

Together, the two algorithms presented here constitute a simple multi-robot mapping system. Each robot constructs a partial map of the environment, and these maps are merged into a complete map. So far, the algorithms have not been used in this manner; we intend to perform experiments to test this approach.

Both of the algorithms presented here assume that they deal only with static environments. Extending them to handle quasistatic or dynamic environments will greatly improve their applicability to real-world situations. The issue of dynamic environments is also related to the problem of structural errors and inconsistencies in the maps; detecting and resolving these errors is particularly important, since they are likely to occur when sensing-limited robots are used.

Of all the work in this thesis, the behaviors specified in Appendix C — which are integral to our mapping algorithm — require the most immediate attention. They make several unrealistic assumptions: that the environments being mapped are rectilinear, and that the robot experiences little or no odometry and sensing error during open-loop movements. Once these behaviors are extended to deal properly with general polygonal worlds, and are made robust to movement error, the single-robot mapping algorithm can be immediately applied to more complicated environments.

There are other fundamental questions brought forth by the topological mapping problem that should be explored:

- How are the quality of the map and a robot's mapping ability in general affected by sensing limitations? By the extent of sensor error? By the complexity of the world?
- What benefits are provided by specific assumptions about the world, such as rectilinearity or near-rectilinearity?
- How can geometry best be used to make inferences when attempting to "close the loop?" Should the "planarity" of the map graph be enforced, and how could this help when closing loops or embedding the map?
- What can be done to make maps in a curved world, where there are no "well-defined" features?

The topological map merging algorithm works well as-is, but it could be made even better. In particular, it would be useful to distribute the algorithm, splitting the computational load between two or more communicating robots in a team. Doing so would enable online map merging even by robots that are severely limited in their computational abilities.

A related topic is the issue of *distributed map storage*. Robots that are limited in their sensing and computational abilities are also often limited in their storage capacity. For large enough environments, even topological maps may exceed the storage capabilities of such robots. It would be beneficial to develop methods for distributing the storage of mapping information, and for the online use of this distributed information by a team of robots. Specifically, mechanisms for planning (e.g. motion planning, task-level planning) with distributed mapping information would be useful.

APPENDIX A
THE DEMPSTER-SHAFER THEORY OF EVIDENCE

The Dempster-Shafer theory of evidence, first introduced by Shafer in 1976 [49], provides a framework for the mathematical representation of uncertainty that differs from traditional probability theory in several key ways:

- Dempster-Shafer theory is capable of representing "ignorance" — a lack of belief in the correctness of any hypothesis.
- The allocation of belief mass to *sets* of mutually exclusive possibilities — not just individual possibilities — is allowed.
- Because of its ability to represent ignorance, Dempster-Shafer theory requires no *a priori* knowledge about the world; in the presence of no knowledge, all belief mass is assigned to ignorance.

In Dempster-Shafer theory, the set of mutually exclusive possibilities (or events), called the *frame of discernment*, is denoted Θ. In our loop-closing problem, the frame of discernment is

$$\Theta = \{C, I\} \tag{A.1}$$

where C represents the possibility that a loop-closing hypothesis is correct, and I represents the possibility that it is incorrect. Unlike traditional probability theory, Dempster-Shafer theory allows us to assign probability to sets of possibilities; in particular, probability can be assigned to any member of the power set of Θ, 2^Θ. In our problem, this is

$$2^\Theta = \{\emptyset, C, I, \{C, I\}\} \tag{A.2}$$

Dempster-Shafer theory defines a *basic probability assignment*. For a frame of discernment Θ, the basic probability assignment $m : 2^\Theta \to [0, 1]$ is a function meeting the following conditions:

$$m(\emptyset) = 0 \tag{A.3}$$

$$\sum_{A \subset \Theta} m(A) = 1 \tag{A.4}$$

Equation A.3 indicates that no belief should be committed to $\emptyset$. Equation A.4 specifies the convention that total belief (over all possibilities) should sum to one. Here, the quantity $m(A)$ is the *basic probability number* assigned to A; this is the measure of belief committed exactly to A — but not the total belief committed to A. The total belief in A is the sum of all belief committed exactly to A and its subsets, i.e.

$$\mathrm{Bel}\,(A) = \sum_{B \subset A} m(B) \tag{A.5}$$

The *plausibility* of A is defined as

$$\mathrm{Pl}\,(A) = 1 - \mathrm{Bel}\,\left(\overline{A}\right) \tag{A.6}$$

Plausibility is the amount of evidence that does not support A's negation. The probability of a particular possibility, in the classical sense, is bounded below by belief and above by plausibility [47], i.e.:

$$\mathrm{Bel}\,(A) \le \mathrm{P}\,(A) \le \mathrm{Pl}\,(A) \tag{A.7}$$

For our small application, with only two possibilities, we are primarily concerned with the basic probabilities $m(C)$, $m(I)$ and $m(\{C, I\})$ — evidence supporting a loop-closing hypothesis, evidence refuting it, and ignorance, respectively.

It is useful to be able to combine multiple belief functions. For example, in our application, each wall that we measure provides evidence which we use to construct a belief function. We must combine the evidence provided by multiple such measurements in order to maintain a "global" belief in the correctness of a hypothesis. To facilitate this, Dempster-Shafer theory uses Dempster's rule of combination. Given two basic probability assignments m_1 and m_2, we combine them with:

$$m_1 \oplus m_2(A) = \frac{\sum_{B,C \in 2^\Theta : B \cap C = A} m_1(B) m_2(C)}{1 - \sum_{B,C \in 2^\Theta : B \cap C = \emptyset} m_1(B) m_2(C)} \tag{A.8}$$

unless $A = \emptyset$, for which

$$m_1 \oplus m_2(\emptyset) = 0 \tag{A.9}$$

In our application, this translates to the following combination rules:

$$m_1 \oplus m_2(C) = \frac{m_1(C)m_2(C) + m_1(C)m_2(\{C, I\}) + m_1(\{C, I\})m_2(C)}{1 - m_1(C)m_2(I) - m_1(I)m_2(C)} \tag{A.10}$$

$$m_1 \oplus m_2(I) = \frac{m_1(I)m_2(I) + m_1(I)m_2(\{C, I\}) + m_1(\{C, I\})m_2(I)}{1 - m_1(C)m_2(I) - m_1(I)m_2(C)} \tag{A.11}$$

APPENDIX B
BELIEF FUNCTION FOR MEASUREMENT SETS

In Section 3.3, we introduced an evidential-reasoning approach to the problem of "closing the loop" during the basic mapping phase of our topological mapping algorithm. This approach is based on the Dempster-Shafer theory of evidence [49]; for a brief introduction to Dempster-Shafer theory, see Appendix A.

Suppose we have made a hypothesis H_0 that two vertices in our map are the same. Determining the precise evidence in support of H_0 provided by a set

$$L = \{\ell_0, \ell_1, \ldots, \ell_n\} \tag{B.1}$$

of measurements that are purportedly from the same wall is an ill-posed question without prior knowledge of the complexity of the world. A belief function for our problem must therefore be at least somewhat subjective in nature.

We denote the uncertainty distribution associated with measurement ℓ_j, taken beginning at vertex v_i and ending at vertex v_j, as U_{v_i,v_j}. This uncertainty is determined by the odometry error model of the robot. Typically, the standard deviation of the distribution increases with the length of a measurement. The accumulation of evidence based on sets of measurements does not assume any particular distribution, as long as covariance information is available.

We have assumed the availability of a reasonably accurate model of the odometry error experienced by the robot. Our belief function is designed by taking a statistical analysis approach using this error model.

B.1 Belief function properties

Any belief function we create should satisfy certain intuitive properties:

1. $m_L(C)$ (belief that H_0 is correct) should vary inversely with the variance of L — a set of widely varied measurements provides little evidence that all the measurements are of the same wall.

2. $m_L(I)$ (belief that H_0 is incorrect) should vary directly with the variance of L; if the variance of L is zero, $m_L(I)$ should be zero (a set of identical measurements should provide no evidence against H_0).

3. A single set of measurements can disprove H_0, but a single set of measurements should generally *not* be sufficient to prove H_0.

B.2 Primary belief function

We define the belief function for a set L of measurements that, according to H_0, are of the same wall:

$$\begin{aligned} m_L(C) &= \Phi(L)\eta_L && \text{(B.2)} \\ m_L(I) &= (1 - \Phi(L))\,\eta_L && \text{(B.3)} \end{aligned}$$

Here, $\Phi : L \rightarrow [0, 1]$ is a value indicating the probability of obtaining L assuming $\ell_i \in L$ are all taken from the same wall. We discuss a Φ-function meeting this criteria in Section B.3.

The value $\eta_L \in [0, 1]$ represents the "probability information content" [56] of $\Phi(L)$, computed as follows:

$$\eta_L = 1 + \frac{\Phi(L)\log(\Phi(L)) + (1 - \Phi(L))\log(1 - \Phi(L))}{\log(2)} \tag{B.4}$$

The probability information content of $\Phi(L)$ essentially measures the "distance" of $\Phi(L)$ from the uniform distribution. Thus, given a $\Phi(L)$ with high entropy (less information), we profess more "ignorance," and for a $\Phi(L)$ with low entropy (more information), we profess less ignorance.

As we traverse the environment after making a hypothesis, we gather new measurements and use them to compute beliefs according to Equations B.2 and B.3. The values we obtain are merged with the overall belief m about H_0 using Dempster's rule of combination (see Section 3.3).

B.2.1 Initial values

Upon first making a hypothesis, we must initialize the overall belief in the hypothesis. A simple method would be to assume complete ignorance (i.e. $m(\{C, I\}) = 1$). However, this is not really the case — we *do* have some belief in the correctness of the hypothesis, or we would not have made it at all.

Recall that geometrically, no two features of the same type and orientation can be within a radius r_* (see Section 3.2). This suggests the metric

$$m_L(C) = \int_{x_0-r_*}^{x_0+r_*} \int_{y_0-\sqrt{r_*-(x-x_0)^2}}^{y_0+\sqrt{r_*-(x-x_0)^2}} U_{v_0,v_h}(x,y)\, d\mathbf{A} \tag{B.5}$$

where v_0 is the start vertex and

$$H_0 \equiv v_0 = v_h \tag{B.6}$$

In other words, the initial value of $m(C)$ represents the probability, according to our error model, that $|v_0 - v_h| \leq r_*$ (in which case they must represent the same feature). We initialize $m(I)$ to zero, however; at the moment we make H_0, we have no basis for assigning belief mass to the possibility that H_0 is incorrect.

B.2.2 Stopping

We have described a belief function for determining the evidence provided for a hypothesis H_0 by a set of measurements that, according to H_0, come from the same wall. When is it safe for our robot to accept or reject H_0? A true "multiple-hypothesis" approach might maintain a set of all hypotheses and their associated belief functions indefinitely, putting off the decision as to which hypothesis is the correct one. We allow multiple hypotheses to coexist, but we also attempt to make decisions concerning their correctness.

In order to do this, we choose a threshold belief value β. If $m(C)$ exceeds β, we accept H_0; if $m(I)$ exceeds β, we reject H_0. Note that this implies that, in order for us to accept a hypothesis, $\mathrm{Pl}\,(I) < 1 - \beta$.

A reasonable choice for β is α, the confidence bound we've used throughout the rest of the mapping algorithm, e.g. for checking (geometrically) whether a vertex might be the same as the start vertex. In fact, we can think of β as a confidence bound on our belief. Though belief is not really equivalent to a classical probability that H_0 is correct (so it is not strictly correct to think of β and α in an equivalent sense), our belief in H_0 can be interpreted as a lower bound on the probability that it is correct, for the measures we have described. See Appendix B for more information.

Because $m(C)$ is a lower bound on the probability that H_0 is the correct hypothesis, setting $\beta \geq \alpha$ maintains the requirement that all mapping "decisions" be made with confidence at least α.

B.2.3 Limitations

Unfortunately, our belief function is not omniscient; it is not able to perfectly satisfy the third property stated in Section B.1 (that a single set of measurements should not be sufficient to prove H_0) — at least, not without artificial discounting of belief. In particular, suppose H_0 is incorrect. We might encounter a never-before-seen wall of precisely the same length as a wall that we had previously observed. In this case, our belief function assigns strong belief in support of H_0, despite the incorrectness of the hypothesis. Such a situation is an inherent weakness of the "forward-traversal" approach — no robot can know what it will encounter in unexplored territory. Closing the loop in pathologically self-similar worlds, while possible with our methods, is still difficult.

There are approaches that we might take to mitigate the risk of self-similar worlds, at the cost of efficiency in simpler environments. In particular, it might be useful to develop a "distinctiveness measure" for paths between vertices: unique paths should provide stronger evidence. Non-distinguishing paths are the primary difficulty with repetitive environments; discounting m_L for such paths would enable us to seek past them, until we encounter a structural mismatch or differentiating path length.

Of course, we cannot generally know whether a path is distinguishing without prior knowledge of the world. A simple approximation, which we have implemented experimentally, is to count the number of edges already in the map that are similar to a newly taken measurement. If this number is large, m_L is discounted accordingly. In many cases this method does serve to help disprove incorrect hypotheses in extremely self-similar worlds (such as that in Figure 5.4). Unfortunately, the pessimism that is designed to defeat incorrect hypotheses often makes correct ones difficult to prove in an efficient manner. Additionally, this method provides little benefit in complicated environments that are self-similar on a large scale.

Practically speaking, few real-world situations exist that are pathologically symmetric enough to warrant the use of the "distinctiveness measure" approach. For this reason, we do not present it as our primary belief function; in the large majority of cases, the function described in Section B.2 performs well.

B.3 A goodness-of-fit based Φ-function

Equations B.2 and B.3 refer to a function $\Phi : L \to [0, 1]$ that returns a value indicating the probability of obtaining L assuming $\ell_i \in L$ are all taken from the same wall. In general, Φ-functions are dependent on the underlying odometry error model of the robot. We assume that covariance information about the uncertainty of a measurement is available or can be calculated.

First we address the univariate case, in which wall-following is a "straight-line" motion with no side-to-side correction. We then extend this approach to the bivariate case in which the robot makes corrections to its offset from the wall as it traverses.

Without loss of generality, we assume $|L| = 2$ throughout this discussion; in particular, $L = \{\ell_j, \ell_k\}$. The results can easily be extended to the case for which $|L| = n \geq 2$.

B.3.1 Univariate error

Suppose our error model is such that $U_{v_i,v_j}(x)$ and $U_{v_i,v_k}(x)$ follow univariate distributions with standard deviations σ_j and σ_k, respectively. Under H_0, our "best estimate" of the true length of the wall from which we have obtained ℓ_j

and ℓ_k is $\hat{\ell}$, computed as follows:

$$\hat{\ell} = \ell_j + \frac{\sigma_j^2(\ell_k - \ell_j)}{\sigma_j^2 + \sigma_k^2} \tag{B.7}$$

For $\hat{\ell}$, we obtain a positional uncertainty distribution $\hat{U}$ according to our error model — *not* estimated from the samples in L — with standard deviation $\hat{\sigma}$. A natural metric for the "quality" of L is the "goodness-of-fit" of ℓ_j and ℓ_k with $\hat{U}$. One well-known goodness-of-fit test is based on the χ^2 distribution.

The *z-score* represents the deviation of a sample from the expected mean, expressed in units of the standard deviation of the underlying distribution. In our case, we compute the sum of the z^2-scores of our samples for the $\hat{U}$ distribution, i.e.:

$$z_{j,k}^2 = \sum_{i \in \{j,k\}} \frac{(\ell_i - \hat{\ell})^2}{\hat{\sigma}^2} \tag{B.8}$$

Note that $z_{j,k}^2$ follows a χ^2 distribution with two degrees of freedom, equivalent to a gamma distribution with $\theta \equiv 2$ and $\alpha \equiv 1$, i.e.

$$\mathrm{P}(y) = \frac{e^{-y/2}}{2} \tag{B.9}$$

In the general case for which $|L| = n$, z^2 follows a χ^2 distribution with n degrees of freedom. This is because we are only estimating the mean ($\hat{\ell}$) from the samples in L (we are not using the samples to estimate $\hat{\sigma}$). A χ^2 distribution with n degrees of freedom is equivalent to a gamma distribution with $\theta \equiv 2$ and $\alpha \equiv n/2$, i.e.

$$\mathrm{P}(y) = \frac{y^{n/2-1}e^{-y/2}}{2^{n/2}\Gamma(n/2)} \tag{B.10}$$

We use this statistic to estimate the degree of support provided by L for H_0. In other words,

$$\Phi(L) = \int_{z_{j,k}^2}^{\infty} \mathrm{P}(y)\ dy \tag{B.11}$$

B.3.2 Bivariate error

This approach is easily extended to the bivariate case (i.e., the case in which wall-following involves two-dimensional movement and error). Suppose we have a pair of (vector) measurements $L = \{\ell_j, \ell_k\}$, and that $U_{v_i,v_j}(x,y)$ and $U_{v_i,v_k}(x,y)$ follow bivariate distributions with $\mathbf{C}_j$ and $\mathbf{C}_k$ representing the respective covariance matrices according to the error model. Under H_0, our "best estimate" of the true wall length is now

$$\hat{\ell} = \ell_j + \mathbf{C}_j(\mathbf{C}_j + \mathbf{C}_k)^{-1}(\ell_k - \ell_j) \tag{B.12}$$

for which we obtain a positional uncertainty distribution $\hat{U}$ according to our error model, with associated covariance matrix $\hat{\mathbf{C}}$.

Mahalanobis distance is the multidimensional equivalent of the z-score. Using $\hat{\mathbf{C}}$, we compute the sum of the squared Mahalanobis distances between our samples and $\hat{\ell}$, i.e.:

$$M_{j,k}^2 = \sum_{i \in \{j,k\}} (\ell_i - \hat{\ell})^{\mathrm{T}} \hat{\mathbf{C}}^{-1} (\ell_i - \hat{\ell}) \tag{B.13}$$

$M_{j,k}^2$ also follows a χ^2 distribution with two degrees of freedom (or n degrees of freedom for $|L| = n$). So, the degree of support provided by L for H_0 is

$$\Phi(L) = \int_{M_{j,k}^2}^{\infty} \mathrm{P}(y)\, dy \tag{B.14}$$

APPENDIX C
MAPPING BEHAVIORS

In Chapter 3, we introduced a topological mapping algorithm for sensing-limited robots that defined its maps in terms of behaviors. The primary behaviors used in the mapping algorithm were: MOVE-TO-WALL, WALL-FOLLOW and HALL-FOLLOW. Here, we detail the specifics of these behaviors for rectilinear worlds.

We define the behaviors for a robot with only three short-range sensors: a front sensor, a left sensor and a right sensor that return ranges r_F, r_L and r_R, respectively. This is a convenient model for behavior development, though our real robots typically have extra sensors. We assume that all ranges are relative to the centroid of the robot.

Our behaviors are developed under the assumption that the robot experiences zero sensing and odometry error — a major limitation. Even with this assumption, the behaviors are quite complicated due to the large number of scenarios that can occur even in rectilinear worlds. In future work, the behaviors will be extended to deal with movement and sensing errors and with general polygonal worlds.

The behaviors take several parameters (Table C.1). Since they are used during robotic map-making, the behaviors place "vertices" in the map of a robot as appropriate; the types of vertices that can be placed are described in Table C.2[1].

The behaviors are depicted as finite state machines in which nodes represent "primitive actions" and edges represent conditions that occur during the execution of those actions. The null condition $\emptyset$ indicates that an action terminates of its own accord (no abnormal conditions are detected). The different types of primitive actions from which our behaviors are constructed are shown in Table C.3.

[1]In these behavior designs, the "SPLIT" vertices have overloaded meaning for convenience of implementation, and the terminology of "portals" (Section 3.4) is slightly modified from the text. When wall-following, a SPLIT vertex indicates a hallway entrance; when hall-following, a SPLIT vertex indicates a junction with another hallway. When wall-following, a PORTAL vertex indicates an opportunity to "leap" to another wall — a situation like that depicted in Figure 3.3(b). When hall-following, a PORTAL vertex indicates a hallway exit. The general rule is that SPLITS imply movement into a hallway; PORTALS imply movement into an open space.

Parameter	Description
r_0	"Nominal" wall-following offset
$r_{\min}$	Minimum allowable wall-following offset
$r_{\max}$	Maximum allowable wall-following offset
r_W	Minimum "half-width" of a hallway

Table C.1: Parameters to mapping behaviors. For more discussion of these parameters, see Section 3.2.

Vertex type	Description
INT	Interior corner
EXT	Exterior corner
ILL	Ill-defined corner/discontinuity (typically ignored)
PORTAL	Movement into an open space (exit hallway or switch walls)
SPLIT	Movement into a hallway (from wall or another hallway)
TERM	Dead-end in hallway

Table C.2: Vertex types in the map. In some cases, a direction is indicated when placing a vertex (e.g. "SPLIT(L)"). This direction is relative to the robot's current frame of reference, and can be converted to the global reference frame of the map for future use.

Action	Argument	Description
A{F,B}	Sensor (r_R, r_L)	Acquire wall with sensor (forward or backward)
M{F,B}	Distance	Move specified distance (forward or backward)
T{L,R}	None	Turn (left or right) 90 degrees
AL	None	Align robot at r_0 offset from wall on the right
CN	None	Center in hallway
FM	None	Move forward, maintaining r_0 offset when wall-following or centering when hall-following
GOTO	None	Move through the known map to the next unexplored place
RET	None	Return some value

Table C.3: Primitive actions that are used in constructing mapping behaviors. Except for FM, GOTO and RET, all primitive actions return the straight-line distance that the robot has moved from the time the action was initiated.

For purposes of clarity and reusability, the main behaviors have been broken down into components. Each component may take parameters as input, and may return some value or set of values. This is also true of the "primitive actions" of which the behaviors are composed; typically, these actions return the distance traveled during their execution.

In some cases it is convenient to store the values returned by some component or action. As an example, to store the distance traveled during the execution of a "move forward x meters" command in the variable δ, we write:

$$\text{MF } x \rightarrow \delta \tag{C.1}$$

Certain components of our behaviors return a list of values. For such occasions, it is convenient to define list operators. In fact, the only such operator we require is the LAST operator. Given a list R of values, $\text{LAST}(R)$ returns the final element in the list.

C.1 EXPLORE

Before we define the "real" behaviors, we define a high-level "bootstrapping" behavior, EXPLORE. This behavior simply executes the basic mapping algorithm of Section 3.2. EXPLORE is specified in Table C.4.

C.2 MOVE-TO-WALL

The MOVE-TO-WALL behavior is very simple: the robot moves forward until one of its sensors detects a wall. The robot then turns to align its wall-following sensor — typically its right-facing sensor — with this wall.

Despite this simplicity, the MOVE-TO-WALL behavior is deceptively difficult to implement robustly on a robot with sparse sensing. This is because a small obstacle or an unfortunate initial alignment can cause the robot to encounter an object without first sensing it. We do not address this issue here. In practice, it is usually unimportant — not because the robot never encounters such a situation, but because it has little effect on the mapping process (mapping does not begin until after a successful MOVE-TO-WALL has been completed). A robot that comes in contact with an obstacle before sensing it is jostled, often so that one of its sensors detects the obstacle it hit; if none does, the robot continues driving, dislodging itself eventually.

The MOVE-TO-WALL behavior is precisely specified in Table C.5.

C.3 WALL-FOLLOW

The primary wall-following behavior is specified in Table C.6. It must handle several cases; the most complicated of these occurs when a “dropoff” condition is detected. This circumstance might lead to the placing of an exterior corner, a portal or — to make things even more confusing — both! The wall-following behavior must approach closer than r_0 to walls that cause an interior corner condition, as they may open into hallways less than r_0 wide. A series of “probing” motions are used to perform this examination.

Much of the difficult work is handled by the PROBE and EXO components, which are also used by the hall-following behavior. The PROBE component (Table C.9) examines a stretch of wall of some specified length, handling any openings in the wall appropriately. The EXO component (Table C.12) is responsible for handling “dropoff” conditions, determining whether they constitute exterior corners or hallway openings.

C.4 HALL-FOLLOW

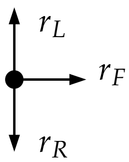
Table C.7 specifies the primary hall-following behavior. The most complicated conditions when hall-following, as when wall-following, occur when “dropoffs” are detected. When hall-following, a dropoff might indicate a T-junction, a portal (end-of-hallway and opening into a non-hallway space), a four-way junction or a corner. Under certain circumstances, a junction of *more* than four hallways can occur.

Once again, most of the work is done with a series of probing motions. Upon encountering a dropoff, the robot determines the nature of this dropoff, and then returns to the initial detection point. It then probes along the wall opposite the dropoff. Finally, it probes the wall opposite the detection point.

Dead-ends and hallway width changes are treated much like the interior corners of wall-following. The robot must approach close to corners in order to determine whether any small openings exist.

C.5 Component specifications

On the following pages are the specifications for each component of the EXPLORE, MOVE-TO-WALL, WALL-FOLLOW and HALL-FOLLOW behaviors. For each component, a description of its purpose is given, along with preconditions, postconditions, required arguments and return values. Additionally, a pointer to the figure depicting the implementation of the component is given. Finally, a series of situations in which each component is applied is depicted. In these situations, the robot is represented according to the convention shown here:



EXPLORE	
Implementation	Page 82, Figure C.1
Description	Explore an enclosed rectilinear environment, placing vertices in the map as appropriate.
Preconditions	None.
Postconditions	None (may stop at any time, on command).
Arguments	$r_0, r_{\min}, r_{\max}, r_W$ (see Table C.1)
Return value	None.

Table C.4: EXPLORE specification.

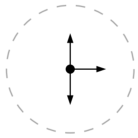
MTW(X)	
Implementation	Page 82, Figure C.2
Description	Move forward until a sensor detects a wall within range r_0. Turn so that the sensor in direction X is aligned with the wall.
Preconditions	None.
Postconditions	The robot is r_0 from a wall to its right.
Arguments	• X: the side of the robot (either R or L) that should face the wall
Return value	None.
Situations	 (No walls in sight)

Table C.5: MOVE-TO-WALL specification.

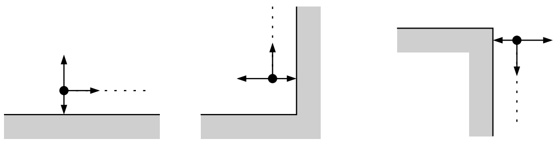
WF	
Implementation	Page 83, Figure C.3
Description	Primary wall-following behavior. The robot moves along walls in an open space, maintaining a range of r_0 from the walls. Interior corners and exterior corners are detected, and nodes are placed. Additionally, openings into hallways are detected; these are labeled as "splits." Under certain special conditions, a "portal" to another open space may be detected.
Preconditions	The robot must be able to detect a wall with its right sensor; additionally, it must be the case that $r_L + r_R > 2r_0$.
Postconditions	None (may stop at any time, on command).
Arguments	None.
Return value	None.
Situations	

Table C.6: WALL-FOLLOW specification.

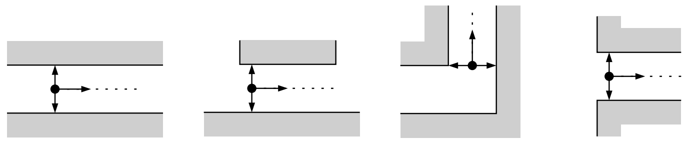
HF	
Implementation	Page 84, Figure C.4
Description	Primary hall-following behavior. The robot moves through hallways, maintaining equidistant range to the walls on each side. Dropoff conditions or constrictions may lead to "splits" in the hallway, or "portals" into open space.
Preconditions	The robot must be able to detect both sides of a hallway; in particular, $r_L + r_R \leq 2r_0$.
Postconditions	None (may stop at any time, on command).
Arguments	None.
Return value	None.
Situations	

Table C.7: HALL-FOLLOW **specification.**

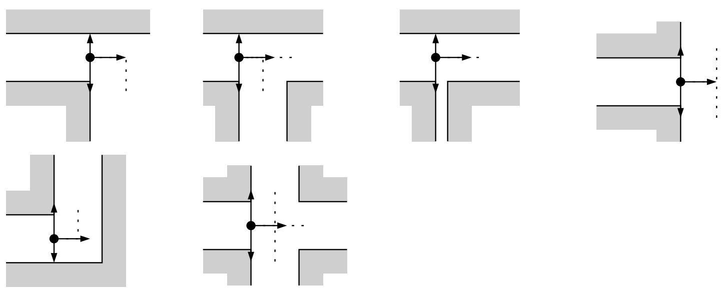
HFO(X)	
Implementation	Page 85, Figure C.5
Description	Handle a "dropoff" condition while hall-following. Explores the dropoff to side X, and then examines the wall opposite the dropoff and the wall directly in front. Nodes (e.g. SPLIT, PORTAL) are added as appropriate.
Preconditions	The robot must be positioned at a dropoff on side X ($r_X > r_{\max}$).
Postconditions	The robot has examined the opening, the wall opposite the opening and the wall directly in front of its original position, and added nodes to its map as appropriate. The robot then returns to hall-following.
Arguments	• X: direction (either R or L) in which a dropoff condition has been detected
Return value	None.
Situations	

Table C.8: HFO(X) component specification.

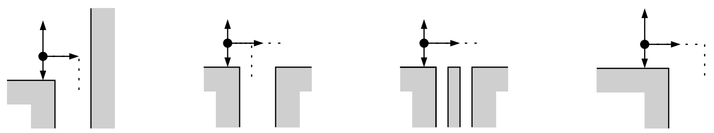
PROBE(X, D)	
Implementation	Page 86, Figure C.6
Description	Explore the wall in direction X over distance D. Multiple openings may occur; place nodes as appropriate.
Preconditions	The robot must be positioned next to a wall on side X.
Postconditions	The robot has updated its map with any appropriate nodes and has repositioned itself at its starting location.
Arguments	• X: direction facing the wall to be examined, relative to the current reference frame (either R or L) • D: length of wall that the robot should explore; if an obstacle in front of the robot is closer than this distance, stop early
Return value	Returns a list of all nodes placed during the exploration.
Situations	

Table C.9: PROBE(X, D) component specification.

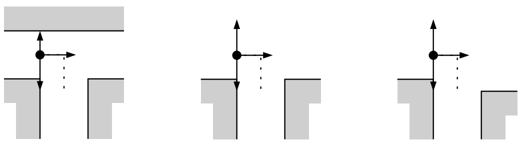
ENTER(X)	
Implementation	Page 86, Figure C.7(a)
Description	Enter a SPLIT in direction X.
Preconditions	The robot must be positioned at the end of a hallway (hall-following) or at a dropoff (wall-following); i.e. $r_X > r_{\max}$. The current hallway or wall must be connected, via a SPLIT node, to a hallway in direction X.
Postconditions	The robot is positioned r_W past the "outermost" corner of the hallway to direction X of the original robot location, and is centered in the hallway.
Arguments	• X: direction (either R or L) in which the hallway to be entered lies, relative to the robot's current frame of reference
Return value	Returns r_W.
Situations	

Table C.10: ENTER(X) component specification.

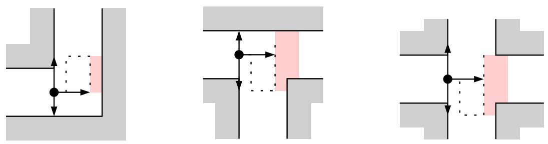
OPP(X, D)	
Implementation	Page 86, Figure C.7(b)
Description	Examine the wall opposite (i.e. directly forward from) the location at which an opening was detected to side X. (The area to be examined is highlighted in the situations depicted below.)
Preconditions	The robot must be positioned at the end of a hallway ($r_X > r_{\max}$). The current hallway must be connected, via a SPLIT node, to a hallway in direction X.
Postconditions	The robot is positioned at its original starting location. Any openings along the wall in the forward direction have been detected and connected (via SPLIT or PORTAL nodes, as appropriate) to the hallway in direction X.
Arguments	• X: direction (either R or L) in which a hallway opening exists; this hallway must be connected via a split node to the current location • D: distance to probe along the wall being explored (typically $2r_0$)
Return value	None.
Situations	

Table C.11: OPP(X, D) component specification.

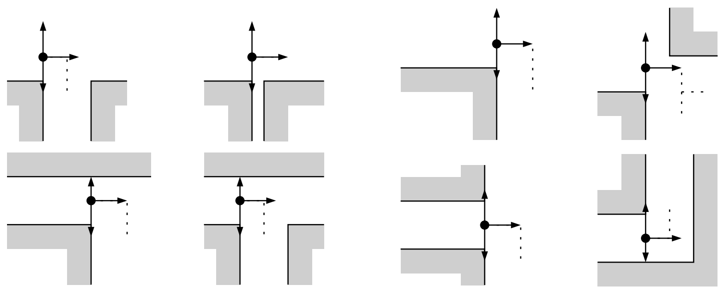
EXO(X, D, N)	
Implementation	Page 87, Figure C.8
Description	Explore a single opening and report back its width (if possible). Place a SPLIT node, a PORTAL node (when hall-following) or an EXT node (when wall-following) as appropriate.
Preconditions	The robot must be positioned at the beginning of a dropoff on side X.
Postconditions	The robot has updated its map with any appropriate nodes and is positioned “past” the opening at the original offset, as if it has moved forward a specific distance; this distance is returned.
Arguments	• X: direction of the dropoff, relative to the current reference frame (either R or L) • D: maximum distance forward that the robot is allowed to travel while exploring the opening; if an obstacle in front of the robot is closer than this distance, stop early • N: type of node to place when an “open area” is detected; PORTAL when hall-following, EXT when wall-following
Return value	Returns a pair: the type of node placed (if any) and the distance moved forward from the starting location.
Situations	

Table C.12: EXO(X, D, N) component specification.

C.6 Component implementations

On the following pages are implementations of each component of the mapping behaviors. These implementations are depicted as finite state machines, constructed using the primitive actions described in Table C.3. Each node represents the execution of a primitive action. Most actions might terminate under several conditions; edges between nodes indicate the terminating condition of the action in the node from which they originate. If none of the explicitly specified conditions is met, the null condition $\emptyset$ specifies the next action to take.

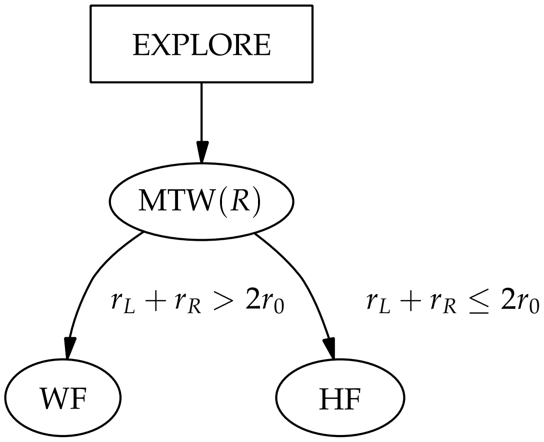


Figure C.1: EXPLORE implementation. "Bootstraps" the exploration process.

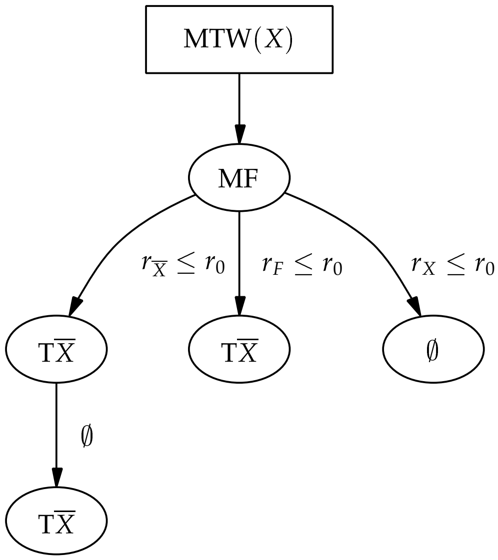


Figure C.2: MOVE-TO-WALL implementation. Moves forward until a wall is encountered, and then aligns the wall-following sensor with the wall.

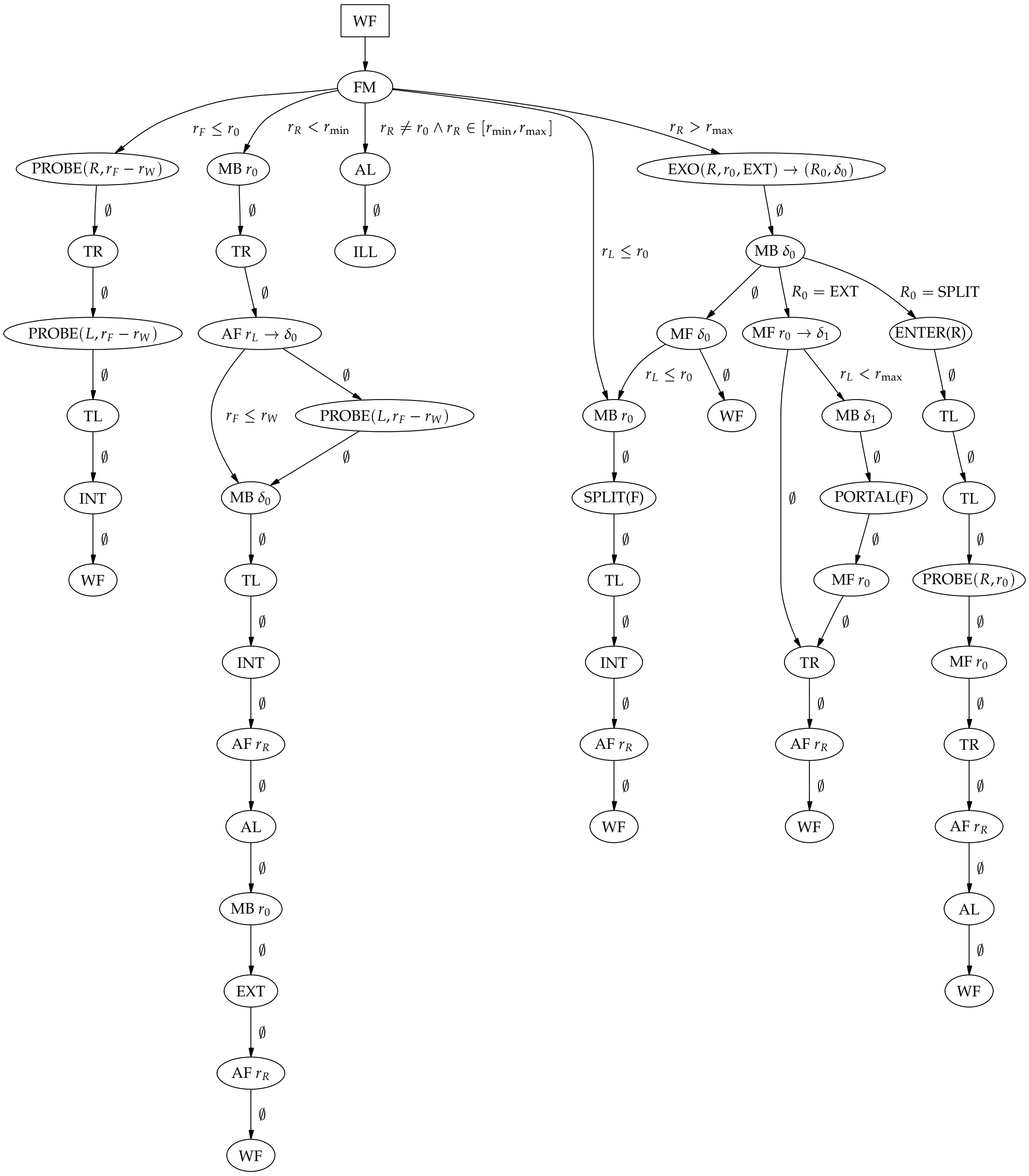


Figure C.3: WALL-FOLLOW implementation. The primary wall-following behavior, which makes use of several separately specified components.

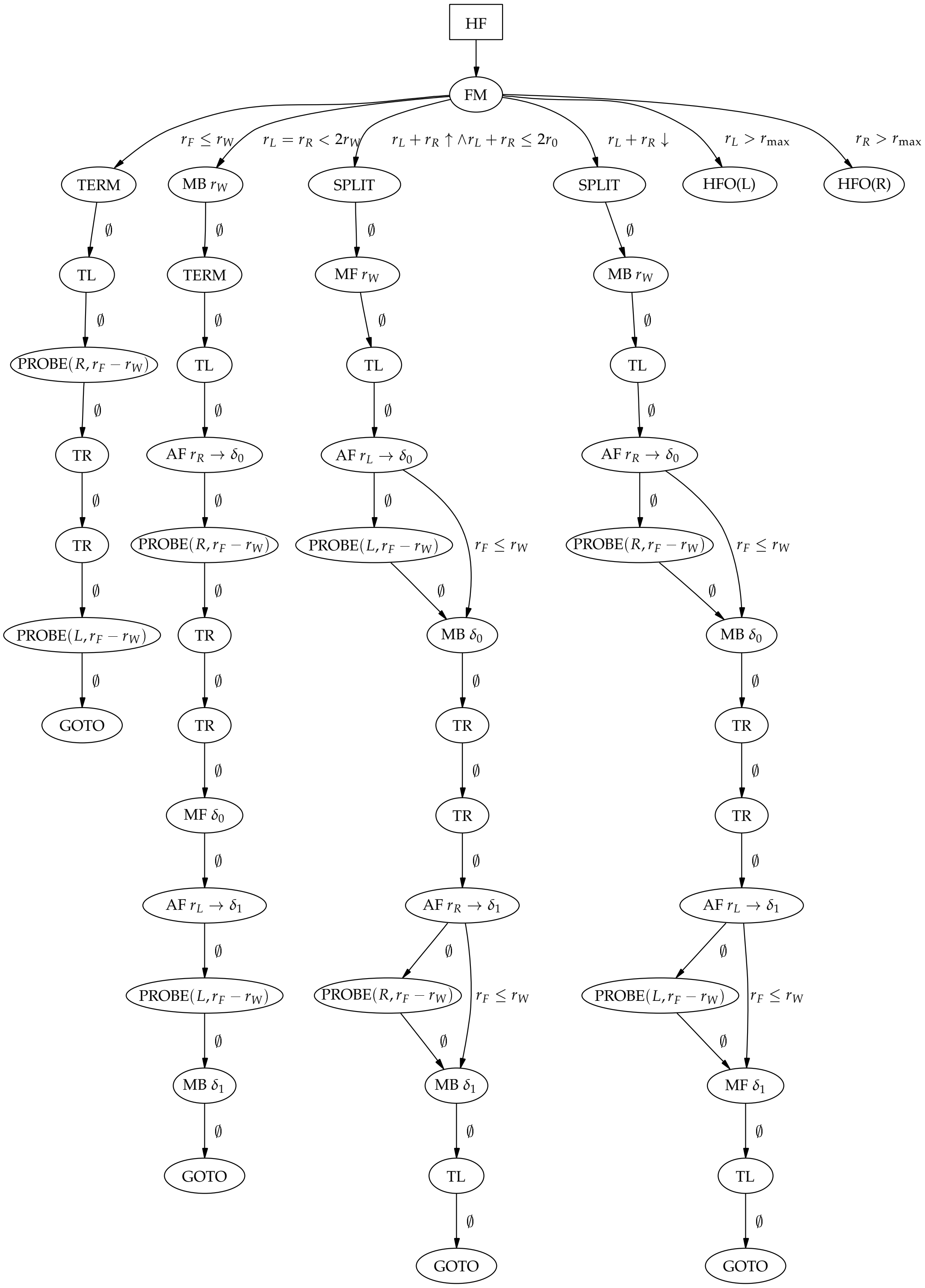


Figure C.4: HALL-FOLLOW implementation. The primary hall-following behavior, which makes use of several separately specified components.

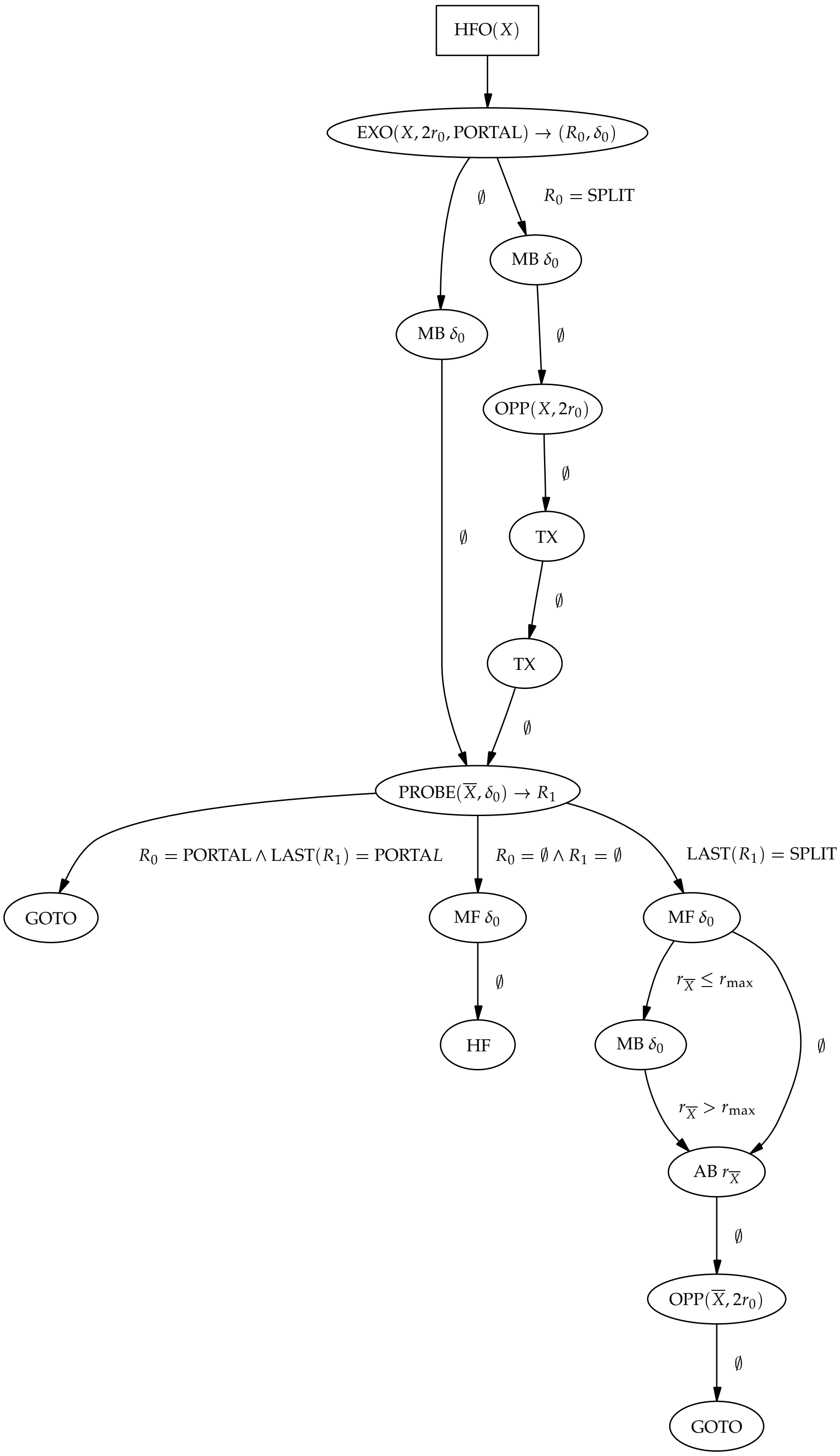


Figure C.5: HFO(X) component implementation. Handles "dropoff" conditions while hall-following.

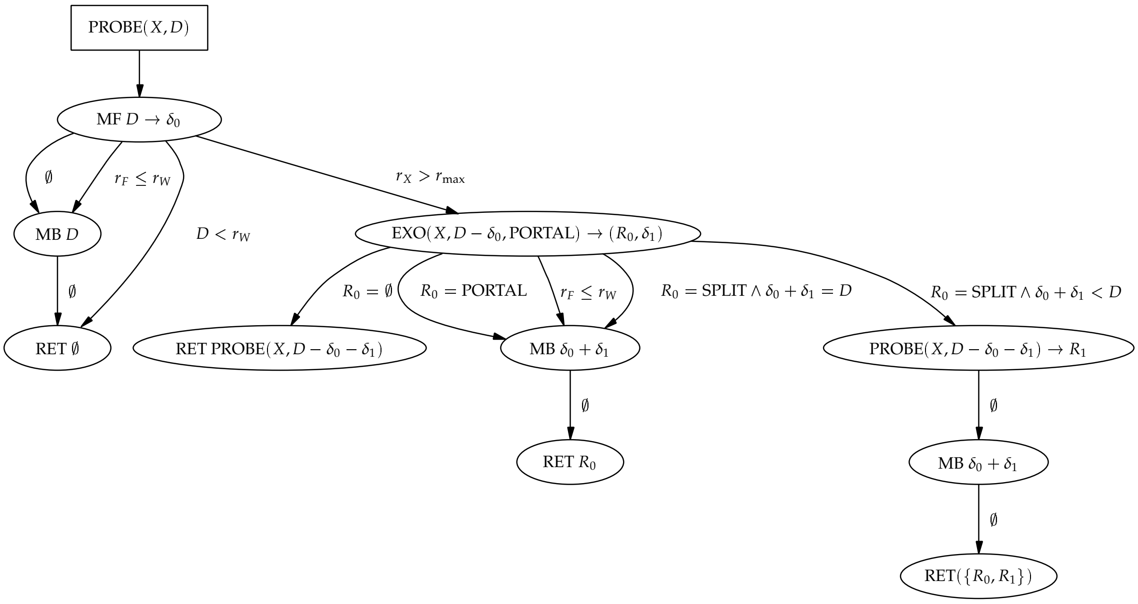


Figure C.6: PROBE(X, D) component implementation. Performs a "probing" motion to detect openings along a length of wall.

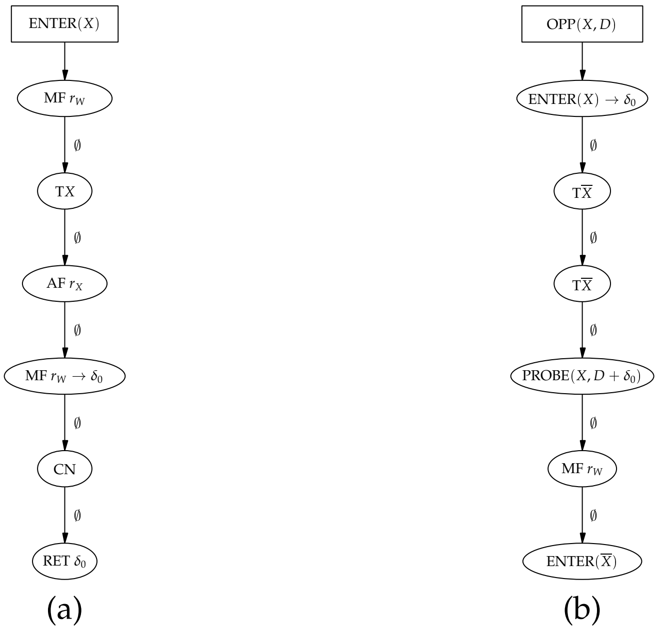


Figure C.7: (a) ENTER(X) component implementation. Enter a SPLIT in specified direction; must be positioned at the end of a hallway adjacent to the SPLIT. (b) OPP(X, D) component implementation. Examine the wall *opposite* an opening into a SPLIT.

1 1

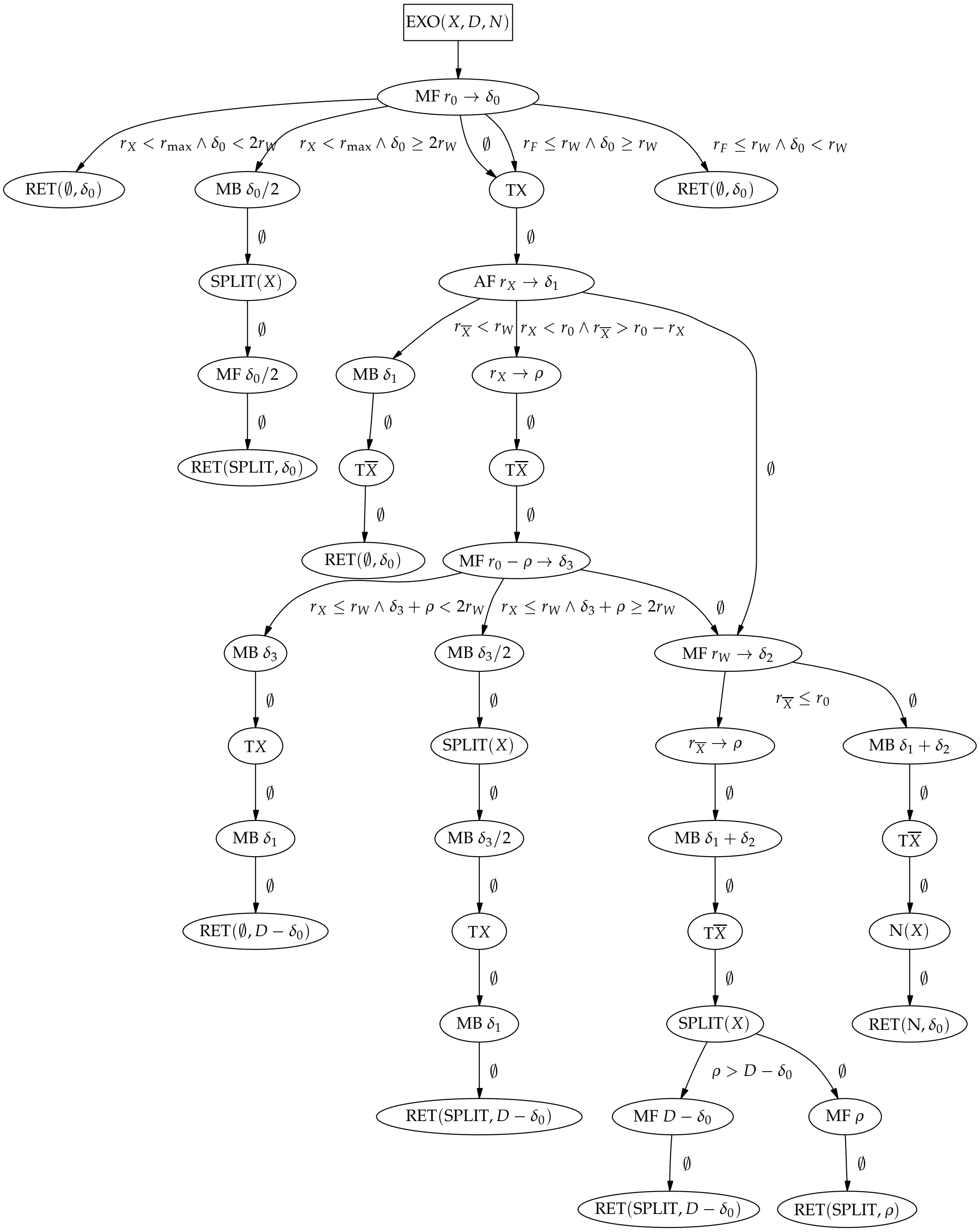


Figure C.8: EXO(X, D, N) component implementation. Examine a single opening and report back its width.

BIBLIOGRAPHY

[1] M.A. Bender, A. Fernandez, D. Ron, A. Sahai, and S. Vadhan. The power of a pebble: Exploring and mapping directed graphs. In *Proceedings of the Annual Symposium on Foundations of Computer Science*, pages 269–278, 1998.

[2] P.J. Besl and N.D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, February 1992.

[3] L.G. Brown. A survey of image registration techniques. *ACM Computing Surveys*, 24(4):325–376, 1992.

[4] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8):689–694, 1997.

[5] H. Bunke and G. Allermann. Inexact graph matching for structural pattern recognition. *Pattern Recognition Letters*, 1(4):245–253, 1983.

[6] Z.J. Butler, A.A. Rizzi, and R.L. Hollis. Distributed coverage of rectilinear environments. In *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, pages 51–62. A.K. Peters, January 2001.

[7] R. Chatila and J.-P. Laumond. Position referencing and consistent world modeling for mobile robots. In *Proceedings of the 1985 IEEE International Conference on Robotics & Automation*, pages 138–145, St. Louis, March 1985.

[8] H. Choset and K. Nagatani. Topological simultaneous localization and mapping (SLAM): Toward exact localization without explicit localization. *IEEE Transactions on Robotics & Automation*, 17(2), April 2001.

[9] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 2004. To appear.

[10] I. Cox and J. Leonard. Modeling a dynamic environment using a Bayesian multiple hypothesis approach. *Artificial Intelligence*, 66:311–344, 1994.

[11] G. Dedeoglu and G.S. Sukhatme. Landmark-based matching algorithm for cooperative mapping by autonomous robots. In *Proceedings of the 2000*, pages 251–260, Knoxville, Tennessee, October 2000.

[12] M. DiMarco, A. Garulli, A. Giannitrapani, and A. Vicino. A set theoretic approach to dynamic robot localization and mapping. *Autonomous Robots*, 16(1):23–47, January 2004.

[13] K.L. Doty and S.L. Seed. Autonomous agent map construction in unknown enclosed environments. In *Proceedings of MLC-COLT Workshop on Robot Learning*, pages 47–55, New Brunswick, N.J., July 1994.

[14] T. Duckett, S. Marsland, and J. Shapiro. Learning globally consistent maps by relaxation. In *Proceedings of the 2000 IEEE International Conference on Robotics & Automation*, volume 4, pages 3841–3846, 2000.

[15] T. Duckett and A. Saffiotti. Building globally consistent gridmaps from topologies. In *Proceedings of the 6th International IFAC Symposium on Robot Control (SYROCO)*, pages 357–361, Wien, Austria, 2000.

[16] G. Dudek, M. Jenkin, E. Milos, and D. Wilkes. Topological exploration with multiple robots. In *Proceedings of the Seventh International Symposium on Robotics with Applications (ISORA '98)*, Alaska, May 1998.

[17] P.J. Durand, R. Pasari, J.W. Baker, and C. Tsai. An efficient algorithm for similarity analysis of molecules. *Internet Journal of Chemistry*, 2, 1999.

[18] H. Feder, J. Leonard, and C. Smith. Adaptive mobile robot navigation and mapping. *International Journal of Robotics Research*, 18(7):650–668, July 1999.

[19] J.W. Fenwick, P.M. Newman, and J.J. Leonard. Cooperative concurrent mapping and localization. In *Proceedings of the 2002 IEEE International Conference on Robotics & Automation*, May 2002.

[20] J.M. Fitzpatrick, D.L.G. Hill, and C.R. Maurer, Jr. Image registration. In Milan Sonka and J.M. Fitzpatrick, editors, *Handbook of Medical Imaging, volume 2: Medical Image Processing and Analysis*, chapter 8. SPIE, 2000.

[21] M. Golfarelli, D. Maio, and S. Rizzi. Elastic correction of dead-reckoning errors in map building. In *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots & Systems*, pages 905–911, B.C., Canada, 1998.

[22] R. Grabowski, L. Navarro-Serment, C. Paredis, and P. Khosla. Heterogeneous teams of modular robots for mapping and exploration. *Autonomous Robots*, 8(3):293–308, 1999.

[23] A. Howard. Multi-robot mapping using mainifold representations. In *Proceedings of the 2004 IEEE International Conference on Robotics & Automation*, April 2004.

[24] W.H. Huang and K.R. Beevers. Topological map merging. Submitted to the *7th International Symposium on Distributed Autonomous Robotic Systems (DARS 2004)*, 2004.

[25] W.H. Huang and K.R. Beevers. Topological mapping with sensing-limited robots. Submitted to the *6th International Workshop on the Algorithmic Foundations of Robotics (WAFR 2004)*, 2004.

[26] J. Jennings, C. Kirkwood-Watts, and C. Tanis. Distributed map-making and navigation in dynamic environments. In *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots & Systems*, volume 3, pages 1695–1701, Victoria, B.C., Canada, October 1998.

[27] J. Ko, B. Stewart, D. Fox, K. Konolige, and B. Limketkai. A practical, decision-theoretic approach to multi-robot mapping and exploration. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots & Systems*, 2003.

[28] K. Konolige. Improved occupancy grids for map building. *Autonomous Robots*, 4(4), December 1997.

[29] K. Konolige, D. Fox, B. Limketkai, J. Ko, and B. Stewart. Map merging for distributed robot navigation. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots & Systems*, volume 1, pages 212–217, October 2003.

[30] B. Kuipers. Modeling spatial knowledge. *Cognitive Science*, 2:129–153, 1978.

[31] B. Kuipers. The spatial semantic hierarchy. *Artificial Intelligence*, 119(1-2):191–233, 2000.

[32] B. Kuipers and P. Beeson. Bootstrap learning for place recognition. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-02)*, Edmonton, Canada, 2002.

[33] 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.

[34] J.J. Leonard and H.J.S. Feder. A computationally efficient method for large-scale concurrent mapping and localization. In J. Hollerbach and D. Koditschek, editors, *International Symposium on Robotics Research*, pages 169–176, 1999.

[35] B. Limketkai, R. Biswas, and S. Thrun. Learning occupancy grid maps of non-stationary objects with mobile robots in indoor environments. In *Proceedings of the 2002 International Symposium on Experimental Robotics (ISER 2002)*, Italy, July 2002.

[36] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4(4):333–349, October 1997.

[37] B. Luo and E.R. Hancock. Structural graph matching using the EM algorithm and singular value decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(10), October 2001.

[38] J.J. McGregor. Backtrack search algorithms and the maximal common subgraph problem. *Software Practice and Experience*, 12(1):23–34, 1982.

[39] B.T. Messmer and H. Bunke. A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5), May 1998.

[40] H.P. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Proceedings of the 1985 IEEE International Conference on Robotics & Automation*, pages 116–121, 1985.

[41] R.R. Murphy. Dempster-Shafer theory for sensor fusion in autonomous mobile robots. *IEEE Transactions on Robotics & Automation*, 14(2):197–206, April 1998.

[42] D. Pagac, E.M. Nebot, and H. Durrant-Whyte. An evidential approach to map-building for autonomous vehicles. *IEEE Transactions on Robotics & Automation*, 14(4):623–629, August 1998.

[43] E. Remolina and B. Kuipers. Towards a general theory of topological maps. *Submitted to Artificial Intelligence Journal*, 2002.

[44] N. Roy and S. Thrun. Coastal navigation with mobile robots. *Advances in Neural Processing Systems*, 12:1043–1049, 1999.

[45] P.E. Rybski, F. Zacharias, J.-F. Lett, O. Masoud, M. Gini, and N. Papanikolopoulos. Using visual features to build topological maps of indoor environments. In *Proceedings of the 2003 IEEE International Conference on Robotics & Automation*, Taipei, Taiwan, September 2003.

[46] B. Schiele and J.L. Crowley. A comparison of position estimation techniques using occupancy grids. In *Proceedings of the 1994 IEEE International Conference on Robotics & Automation*, pages 1628–1634, San Diego, CA, May 1994.

[47] K. Sentz and S. Ferson. Combination of evidence in Dempster-Shafer theory. Technical Report SAND2002-0835, Sandia National Laboratories & Binghamton University, April 2002.

[48] G. Shafer. Perspectives on the theory and practice of belief functions. *International Journal of Approximate Reasoning*, 4:323–362, 1990.

[49] Glenn Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[50] H. Shatkay and L.P. Kaelbling. Learning topological maps with weak local odometric information. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, 1997.

[51] H. Shen, C.V. Stewart, B. Roysam, G. Lin, and H.L. Tanenbaum. Frame-rate spatial referencing based on invariant indexing and alignment with application to on-line retinal image registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(3), March 2003.

[52] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In I. Cox and G. Wilfong, editors, *Autonomous Robot Vehicles*. Springer-Verlag, 1990.

[53] C.V. Stewart, C.-L. Tsai, and A. Perera. A view-based approach to registration: Theory and application to vascular image registration. In *Proceedings of the International Conference on Information Processing in Medical Imaging (IPMI)*, May 2003.

[54] C.V. Stewart, C.-L. Tsai, and B. Roysam. The dual-bootstrap iterative closest point algorithm with application to retinal image registration. Technical Report RPI-CS-TR 02-9, RPI Department of Computer Science, Troy, NY, June 2002.

[55] A. Stroupe, M. Martin, and T. Balch. Distributed sensor fusion for object position estimation by multi-robot systems. In *Proceedings of the 2001 IEEE International Conference on Robotics & Automation*, 2001.

[56] J. Sudano. Pignistic probability transforms for mixes of low- and high-probability events. In *Proceedings of the 4th International Conference on Information Fusion*, pages 23–27, Montreal, August 2001.

[57] S. Thrun. A probabilistic online mapping algorithm for teams of mobile robots. *International Journal of Robotics Research*, 20(5):335–363, 2001.

[58] S. Thrun and A. Bücken. Integrating grid-based and topological maps for mobile robot navigation. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 944–950, 1996.

[59] S. Thrun, D. Fox, and W. Burgard. Probabilistic mapping of an environment by a mobile robot. In *Proceedings of the 1998 IEEE International Conference on Robotics & Automation*, pages 1546–1551, 1998.

[60] S. Thrun, J. Gutmann, D. Fox, W. Burgard, and B. Kuipers. Integrating topological and metric maps for mobile robot navigation: A statistical approach. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 1998.

[61] Sebastian Thrun. Robotic mapping: A survey. Technical Report CMU-CS-02-111, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, February 2002.

[62] N. Tomatis, I. Nourbakhsh, and R. Siegwart. Combining topological and metric: a natural integration for simultaneous localization and map building. In *Proceedings of the Fourth European Workshop on Advanced Mobile Robots (Eurobot 2001)*, pages 19–21, Lund, Sweden, September 2001.

[63] N. Tomatis, I. Nourbakhsh, and R. Siegwart. Hybrid simultaneous localization and map building: closing the loop with multi-hypothesis tracking. In *Proceedings of the 2002 IEEE International Conference on Robotics & Automation*, May 2002.

[64] B. Tovar, S. LaValle, and R. Murrieta. Optimal navigation and object finding without geometric maps or localization. In *Proceedings of the 2003 IEEE International Conference on Robotics & Automation*, Taipei, Taiwan, September 2003.

[65] Ray A. Waller. *Statistics: An introduction to numerical reasoning*. Holden-Day, San Fransisco, 1979.

[66] R.C. Wilson and E.R. Hancock. Structural matching by discrete relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6), June 1997.