MAPPING WITH LIMITED SENSING

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
DOCTOR OF PHILOSOPHY
Major Subject: Computer Science

Approved by the
Examining Committee:

Dr. Wesley H. Huang, Thesis Advisor

Dr. I. Volkan Isler, Member

Dr. Malik Magdon-Ismail, Member

Dr. Arthur C. Sanderson, Member

Rensselaer Polytechnic Institute
Troy, New York

January 2007
(For Graduation May 2007)

For Grampa,
Who has inspired me to be a scientist since t_0

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

ACKNOWLEDGMENT

There are, of course, dozens of people whose contributions have led in some way to my writing this thesis. I could write *another* lengthy document on this topic alone, but for the sake of brevity I'll instead just begin with a broad "Thank you all!" So:

Thank you all!

My advisor, Wes Huang, has worked closely with me for a number of years, and his guidance has certainly been the most singularly driving force behind this thesis and indeed, all of the research I have done as a graduate student. Wes has gone out of his way to be involved in my research at every level, even after leaving RPI, and for that I cannot thank him enough.

Volkan Isler has offered his advice, encouragement, and insight to me almost since he arrived on campus. His thoughtful comments have helped shape my recent research and changed the way I think about research in general. Malik Magdon-Ismail and Art Sanderson's feedback has also led to some of my newer research. Chapter 3, in particular, stems from their comments.

I have been at RPI for quite a long time, and the faculty and staff of the Computer Science Department are largely responsible for making my stay so enjoyable. I must especially thank Shannon Carothers, Terry Hayden, Chris Coonrad, and the department's hard-working Labstaff for their frequent help, and Srinivas Akella, Barb Cutler, and Jeff Trinkle for their many insights.

The other graduate students in the Algorithmic Robotics Laboratory have listened to me talk about this research on countless occasions, have given me many suggestions and thoughtful comments, and have generally been a great bunch of labmates. My thanks especially to Arjun Arumbakkam, Kartik Babu, Steve Berard, Nilanjan Chakraborty, Eric Meisner, Binh Nguyen, and Evan Shechter for many interesting discussions, help with math and implementation issues, and general camaraderie. I must also give extra special thanks to Jon Fink for his great work on designing and building (with Wes) our fleet of Ratbots from scratch.

I am also fortunate to have a great group of friends away from the lab. This group is so large that I cannot thank everyone, though there are some names listed in my previous thesis (Beevers, 2004). Thank you all for providing an escape from the often all-consuming world of research. Special thanks to Leah, whose excitement about my finishing this thesis has been as great a motivator as any other during these last few months.

As seems to nearly always be the case with acknowledgments in Ph.D. theses, I will finish with the most important thank you: to Mom, Dad, and Kimberlee. You have patiently worried from afar, encouraged me during rough patches, cheered me on during less-rough patches, and given me a safe-haven which is the most comfortable place in the world. I love you very much!

January 2007

P.S.: Luke — I am still working on a robot that pets dogs. It turns out to be a hard problem! Hang in there!

ABSTRACT

For a mobile robot to interact with its environment in complex ways, it must possess a model of the environment, i.e., a *map*. It is generally infeasible for an accurate map to be constructed by hand and provided to the robot prior to deployment, so the robot must build a map online.

Most mapping algorithms assume the availability of frequent, high fidelity feedback from the environment acquired using sensors such as scanning laser rangefinders. Such sensors provide dense, accurate, and long-range data at great expense: they typically cost thousands of (US) dollars, consume significant power and space resources, and require the use of powerful computers for data processing. These sensors are therefore unsuitable for deployment in many applications, e.g., consumer robots, disposable robots for search and rescue or hazardous material detection, or mobile sensor networks.

This thesis instead examines the mapping problem in the context of *limited sensing*. Sensors such as infrared rangefinder arrays are inexpensive, low-power, and small, but they give only low-resolution, short-range feedback about the environment and are thus difficult to use for mapping. The main contributions of this thesis are a theoretical characterization of the relative capabilities of mapping sensors, and the development and demonstration of several practical algorithms for building maps with real, inexpensive sensors.

The thesis begins by examining a generalized model of mapping sensors. The model is applied with a simple mapping algorithm to characterize the space of sensors in terms of their suitability for mapping, and theoretical bounds on map error in terms of the capabilities of the sensor are derived. The thesis then turns to practical algorithms for simultaneous localization and mapping (SLAM) with limited sensors. In the context of particle filtering SLAM, an approach is formulated for accumulating low-resolution sensor measurements in "multiscans" that contain sufficient information for feature extraction and data association. To manage uncertainty due to limited feedback, a new technique for incorporating prior knowledge in the mapping process is developed, and new sampling techniques to combat estimation degeneracies in particle filtering SLAM are described. The algorithms presented in the thesis are validated experimentally through simulation, tests on well-known benchmark data sets, and implementation and deployment on real robots, and a working implementation of particle filtering SLAM on an 8-bit microcontroller with a small infrared rangefinder array is described in detail.

1 INTRODUCTION

The ability of a mobile robot to move efficiently and accurately throughout the world depends crucially on the robot's model of the world — i.e., its *map*. A robot's map-building capability is primarily a function of its perceptual and computational capacities. Unlike humans, whose sensing capabilities capture minute details of environment geometry, structure, texture, and color, today's robots must make do with much less information. Nevertheless, by employing accurate (and expensive) sensors, such as scanning laser rangefinders, research in robot mapping has led to algorithms that can build reasonably precise maps of real-world environments.

Most current mapping algorithms fail, however, when presented with much lower-fidelity sensors. For example, a simple array of a few infrared rangefinders provides data that is too sparse to be used to observe meaningful environment features frequently and accurately. Yet, for robot mapping to be widely employed — e.g., in consumer products — it must be possible to build maps with inexpensive sensors. Other constraints on power usage, sensor size, and computation further reinforce the need for mapping techniques that work with very limited sensing.

1.1 The impact of sensing limitations

Imagine a person, relocated to an unfamiliar building, perhaps outfitted with heavy mittens, and blindfolded (all voluntarily, of course). Our subject is given the following objective:

> Explore the building, using only outstretched arms and contact with walls. When commanded, return quickly to the starting location.

A simple enough task, but upon trying it yourself, you may find the goal remarkably difficult to achieve. Without visual and other feedback from the environment, you quickly become disoriented and the accuracy of your estimated "configuration" with respect to the world is severely reduced. In exploring a building while blindfolded, feedback is restricted to what can be "observed" by your outstretched hands, which is generally insufficient to recognize a specific location. In particular, it is extremely difficult to uniquely recognize a place — say, a specific office or desk among many — given only "touch" information from mitten-covered hands.

This example illustrates a notion of limited sensing that closely corresponds to the sensing of some of the robots studied in this thesis. The *range* at which obstacles can be detected is relatively small — an arm's length. Furthermore, the *spatial resolution* of the subject's sensing is limited: s/he has only two arms. These and other related limitations, such as sensor noise, are examined in the following chapters.

1.2 Thesis overview

This thesis proposes algorithms for mapping with limited sensing. The algorithms can be used to build accurate maps of indoor environments with low-resolution, short-range sensors such as arrays of infrared rangefinders. Toward this end, the thesis addresses both theoretical and practical aspects of the mapping problem and documents the results of simulated and real-world experiments for the presented algorithms.

The thesis starts with a theoretical examination of the relationship between a robot's sensing capabilities and the quality of the map that is produced. A broad model of mapping sensors is employed, along with several simplifying assumptions about the robot and environment, to obtain analytical bounds on map error in terms of the model parameters. These results are applied to realistic sensors, including scanning laser rangefinders, SONAR arrays, and infrared rangefinder arrays, to compare the relative mapping abilities of the sensors.

The remainder of the thesis focuses on practical algorithms for simultaneous localization and mapping (SLAM) with limited sensing. All of the proposed SLAM algorithms are based on the relatively new but widely successful particle filtering paradigm. The thesis first shows how particle filtering SLAM can be adapted to accommodate low-resolution sensors that cannot instantaneously observe meaningful features of the environment. Second, a new particle filtering algorithm is developed to enable the incorporation of prior knowledge about the environment in the filtering process, in the form of relative constraints on the map. Third, the thesis describes several new sampling techniques for particle filtering SLAM that significantly improve the consistency of the estimation process. While the last two contributions are especially relevant to the limited sensing case, they are useful as well for robots with traditional mapping sensors such as laser rangefinders.

One of the most important advantages of particle filtering SLAM algorithms is their computational efficiency and flexibility. A robot with severe sensing constraints is likely to also be limited in its computational abilities. The last part of the thesis demonstrates that particle filtering SLAM algorithms can be implemented not only for robots with limited sensing, but *on board* robots with limited computing and sensing. A particle filtering implementation for robots with an 8-bit microcontroller and a five-sensor infrared rangefinder array, using only fixed point numbers, is described and demonstrated.

1.3 Sensors for mapping

For a robot to interact in complex ways with its environment — traversing corridors, driving to the office, playing soccer, or petting a dog — it generally must possess a model, or map, of the environment. In some circumstances, like the controlled and fixed environment of an automated factory, it is reasonable to create a map before the robot is ever deployed. Most of the time however, this is not an option: CAD models cannot capture the haphazard placement of desks, boxes, couches, shelves, lamp posts, parked cars, and sleeping dogs encountered in reality. It is therefore necessary for a robot to build a map of its surroundings *online*, i.e., as it moves through those surroundings.

The ability of a robot to build a map relies most heavily on its perceptual and computational capabilities. A robot that cannot sense its surroundings cannot model them, and a robot that cannot process sensory feedback in real time is as bad off as a robot with no feedback whatsoever. Modern robots vary widely in the sensors and processors they employ. Computing on robots ranges from bare-bones 8-bit microcontrollers to onboard multiprocessor high-end PCs. Sensing is similarly diverse, extending from the simplest on-off bump sensors to GPS, scanning laser rangefinders, and pan-tilt stereo cameras. While intuitively it seems desirable to use the most powerful computing and sensing possible to build maps, in reality there are a number of tradeoffs, ranging from power consumption to sheer cost.

This section investigates some of the characteristics of sensors that can be used for mapping. We show that, in order to do mapping with low-cost, low-power, lightweight sensors, we must make do with much less data than is normally available to robot mapping algorithms, and thus, we must investigate new algorithms for building maps with limited sensing.

1.3.1 Types of sensing limitations

This thesis primarily focuses on limitations in *exteroceptive* perception — sensors that observe the world. *Proprioceptive* sensing, which observes the robot's internal state, is also important to robot mapping. The most frequently used proprioceptive information for mapping is encoder information from actuators, e.g., odometry from a wheeled robot's motors. Clearly proprioceptive information is useful in mapping, but it is not *sufficient* for mapping because it is impossible to build a model of the world without observing it.

For now, we categorize exteroceptive sensors according to three capabilities: the *range* of the sensor, its spatial *resolution*, and its *accuracy*. (Chapter 3 examines these and other sensing limitations in more detail.) To make precise our interpretation of these characteristics, we assume the sensor (or array of sensors) is mounted in a fixed location on the robot chassis, and consider the "instantaneous" returns of the sensor. This is appropriate for most formulations of mapping, for which an "observation" is the data produced by the sensor at a specific time, or over a very short time interval.

Table 1.1 on page 4 presents a comparison of several different exteroceptive sensors that can be used for mapping. The table is sorted (approximately) according to the cost of the sensor. The primary conclusion that can be drawn from the table is that that to do mapping with low-cost sensors, one must settle for sensors that are somehow limited, be it in range, resolution, or accuracy. Below, we discuss these three sensor capabilities in more detail and discuss how limitations in each affect mapping. In Section 1.3.2 we address some of the costs associated with different sensors.

1.3.1.1 Spatial resolution limitations

The spatial resolution of mapping sensors is widely varied. The resolution of arrays of simple sensors such as bump, SONAR, and infrared sensors is dependent upon the arrangement of the array. Typically, a small number of individual sensors (e.g., 5, 8, 16, or 24) are spaced evenly around the perimeter or half-perimeter of the robot. (In Chapters 4–5 of this thesis we generally assume 5 sensors placed at $-90°, -45°, 0°, 45°$, and $90°$ in the forward-facing reference frame of the robot, as shown in the diagrams in Table 1.1.) With this "sparse" sensing it is impossible to instantaneously observe significant features of the environment that could be used to build maps.

In contrast, a scanning laser rangefinder offers dense coverage of the environment, typically in one or one-half degree increments over a $180°$ or $360°$ field of view. Monocular and stereo cameras are similarly high-resolution, although they generally have smaller fields of view. An omnidirectional camera gives fairly dense coverage over the full $360°$ field of view.

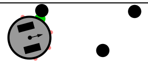
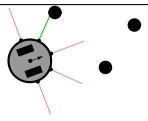
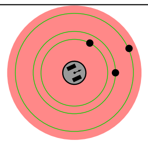
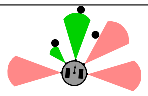
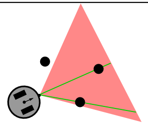
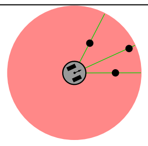
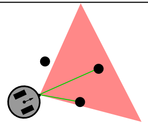
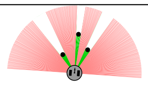
One approach to compensate for low-resolution sensing is to sweep the sensor over the environment, either by mounting it on some actuated mechanism, or by using the motion of the robot. The former approach introduces extra mechanical complexity and costs but may be feasible in some circumstances. This thesis instead examines the latter technique in Chapter 4, building maps with low-resolution sensors without introducing mechanical complications.

1.3.1.2 Range limitations

The range capabilities of robot sensors vary from zero range (e.g., bump sensors) to very long range (e.g., cameras, which are limited only by visibility conditions). The simple consequence of range-limited sensing is that a robot cannot observe, act on, or map portions of the environment that are outside the sensor's effective range.

Thus, for a mapping algorithm to build accurate maps with range-limited sensing, the algorithm must actively control the robot's motion to pass the sensor over portions of the workspace to be mapped. Most mapping algorithms are "passive" in nature and do not direct the robot's motion, but by incorporating an exploration strategy the impact of range limitations can be reduced. While this thesis does not address the exploration problem, Chapter 3 examines analytically the influence of range restrictions on mapping efficiency.

Table 1.1: Sensors: capabilities and costs. For each sensor an overhead schematic is shown, depicting the sensor's (ideal) returns for a simple situation with several obstacles.

Sensor	Diagram	Range	Resolution	Accuracy	Cost	Power	Size/weight	Computation
Bump sensor array		zero	low[a]	accurate distance inaccurate bearing[a]	low	low power	small size lightweight	little
Infrared array		short (~1 m)	low[a]	accurate distance accurate bearing	low	low power	small size lightweight	little[c]
RF signal strength[b]		medium	N/A	inaccurate distance no bearing information	low	moderate power	small size lightweight	little
SONAR array		medium (~10 m)	low[a]	inaccurate distance (reflections) inaccurate bearing (wide beamwidth)	medium	low power	medium size lightweight	little[c]
Monocular camera		long	high[d]	no distance information accurate bearing	medium	low power	small or medium size lightweight	significant[c]
Omnidirectional camera		long	high	no distance information accurate bearing	high	low power	large size moderate weight	significant
Stereo camera		long	high[d]	moderately accurate distance accurate bearing	high	moderate power	large size moderate weight	significant
Laser rangefinder		medium (~10-40 m)	high ($\frac{1}{2}^\circ$-1°)	accurate distance accurate bearing	high	moderate power	large size moderate weight	significant[c]

[a]Depending upon the configuration of the sensor array
[b]Assumes "obstacles" are RF beacons
[c]Depending on the feature extraction approach that is employed
[d]Within the field of view of the camera, which may be small

1.3.1.3 Distance accuracy limitations

In addition to range and resolution limitations, sensors are also subject to accuracy constraints due to noise or other characteristics of the sensor. Most approaches for mapping make use of range and bearing measurements to build maps, so we consider the accuracy of sensors in measuring both the distance to an obstacle and the angle to the obstacle in the robot's reference frame.

Within their maximum range, sensors are subject to inaccuracy in measuring the distance to an object. For example, a SONAR beam may reflect off a specular surface, yielding an inaccurate measurement. Infrared and laser sensors are typically fairly accurate at measuring distance, although their performance may vary with environmental conditions, e.g., temperature. Stereo cameras measure range but the accuracy of the measurements depends crucially on the calibration and resolution of the cameras. By combining the information from multiple sensor returns, inaccurate range information can be used to obtain an accurate estimate of the features being observed.

Mapping can also be done with sensors that do not measure range at all — so-called "bearing-only mapping." Monocular or omnidirectional cameras cannot instantaneously determine depth information, but by exploiting the motion of the robot and image registration techniques, range information can be estimated in similar fashion to stereo cameras.

1.3.1.4 Bearing accuracy limitations

Many sensors measure the bearing to an observed object. For sensors in an array, the bearings at which objects can be observed depend on the placement of the individual sensors. Depending on the beam width of a sensor, measured bearings may be very uncertain. For example, a SONAR sensor's beam width is typically around 30° and it is impossible to instantaneously determine the bearing within the beam of an obstacle generating a return. A bump sensor can be considered to have a "beam width" as well, i.e., the physical width of the bumper that triggers the sensor toggle. Typically infrared and laser sensors have very small beam widths, and the accuracy of bearing measurements returned by cameras is limited mainly by the camera resolution.

Analogous to bearing-only mapping is the idea of "range-only mapping." Sensors that only measure distance are uncommon in mapping, but one example is the use of RF signal strength to measure the range to beacons. To build maps encoding the locations of the beacons, one can compensate for the lack of bearing information by observing beacons from multiple vantage points. When faced with highly inaccurate bearing measurements such as those given by SONAR sensors, one may wish to treat mapping as a range-only problem, as in (Leonard et al., 2002).

1.3.2 Sensor cost considerations

Barring other considerations, we should use the longest range, highest resolution, most accurate sensors available to build maps. This has traditionally been the approach taken in robot mapping research, in which laser rangefinders are predominant. However, for mapping outside the context of research, there are several tradeoffs to consider depending on the intended purpose of the robot. These tradeoffs lead to the conclusion that in many circumstances, we must build maps using limited sensors.

1.3.2.1 Monetary cost

The most immediate consideration is the monetary cost of the sensor. Scanning laser rangefinders typically cost thousands of US dollars, far too much for use in, e.g., consumer products or "disposable" robots for search and rescue or hazardous waste cleanup. Other reasonably accurate, long-range, high-resolution sensors such as stereo cameras are somewhat less expensive but remain costly. To build mass-producible robots with tight production cost constraints, cheaper sensors such as infrared rangefinder arrays or basic monocular cameras must be used.

1.3.2.2 Power

Many real-world robotics applications are subject to power constraints. For example, mobile nodes in a sensor network may need to operate for hours or days without recharging. Furthermore, the cost and size of batteries and the power requirements of other devices (such as computers) on the robot may demand the use of sensors that need very little energy to operate. Fortunately, when compared with the power costs of actuation, computation, and communication, most sensors are reasonably efficient.

1.3.2.3 Size, weight, and convenience

Related to energy costs are issues of sensor size and weight: a robot must be large enough to support the sensor it uses, and in many cases larger robots are too cumbersome to be useful. Laser rangefinders typically consume significant space and weigh at least a kilogram, as opposed to several (collective) grams for a small array of infrared or SONAR sensors. Cameras (CCD or CMOS) are also reasonably small, in part due to miniaturization for inclusion in consumer-level digital cameras and cellular telephones.

Some sensors have special placement requirements as well. For example, an omnidirectional camera must be mounted so as not to be obstructed by any part of the robot. Such requirements may be inconvenient depending on the nature of the particular robot.

1.3.2.4 Computational costs

Finally, the computational costs associated with the data produced by a sensor must be considered. A high-resolution sensor produces many readings per scan that must be processed to build a map. Generally the amount of computation associated with a sensor for mapping is simply a function of the amount of data produced by the sensor, and typically much of the computation is expended in extracting features from the raw sensor returns. For example, image processing for edge or point extraction is usually applied to pixel data produced by monocular cameras.

Occasionally it is reasonable to use raw sensor data for mapping and no feature extraction is necessary. This is the case, for example, for laser scan-matching techniques that compare raw laser readings directly to do mapping. Nevertheless, these techniques still require computation proportional to the amount of data produced by the sensor, in order to perform the matchings.

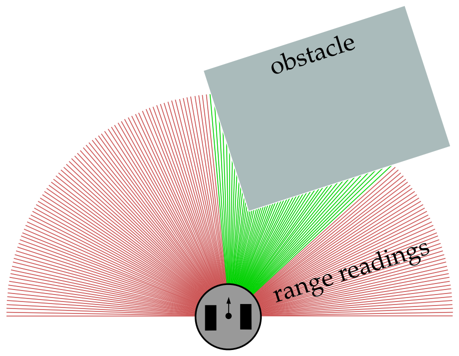
1.4 Mapping with limited sensing

The end result of the above discussion is that in many cases we must make do with a sensor that is less than ideal for mapping. However, most widely used robot mapping algorithms work poorly with sensors such as infrared or SONAR arrays that produce very little data. To see why, consider the basic high-level approach used by the majority of mapping techniques, shown in Algorithm 1.1.

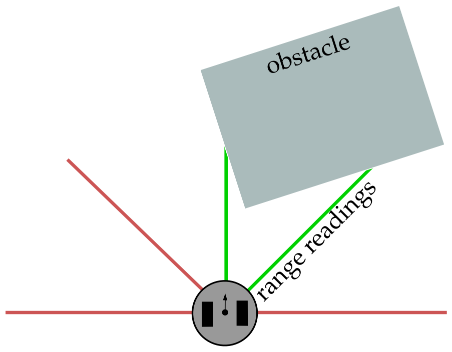
Algorithm 1.1 A basic mapping algorithm

```
1: loop
2:     Move; update the robot's pose estimate based on proprioceptive information
3:     Sense the environment
4:     Extract features from the raw sensing data
5:     Match features with the current map
6:     Based on matches, update the robot's pose estimate and the estimated map
7:     Add unmatched features to the map
8: end loop
```

This approach relies critically on the ability to extract features from raw sensor data (Step 4). Typically these features are simple geometric objects, e.g., lines or points. However, the data produced by severely limited sensors is usually insufficient for feature extraction, as depicted in



(a) A typical scanning laser rangefinder arrangement, with 180 degrees of dense coverage, can instantaneously observe the parameters of environment features



(b) An infrared rangefinder array, with only five range-bearing readings in a single scan, cannot instantaneously observe feature parameters

Figure 1.1: Comparison of high- and low-resolution sensors.

Figure 1.1.[*] In the figure, it can be seen that lines representing the sides of the obstacle observed by the laser are clearly reflected in the data. This is not the case for an infrared rangefinder array. Furthermore, range-limited sensors produce no observations whatsoever when the robot is positioned too far from an obstacle.

The underlying issue in mapping with limited sensing is that it is generally impossible to instantaneously observe the entire "state" (i.e., all the parameters) of features of the environment. For example, a line feature to represent a side of the obstacle in Figure 1.1 can be parameterized by two variables. However, the infrared rangefinder array in the figure only observes a single point on any edge of the obstacle — insufficient to determine the parameters of a line.

To collect enough data for feature initialization using a sparse array, it is necessary to acquire multiple sensor returns from different robot poses and consider them together. Note that this is similar to approaches used for bearing-only and range-only mapping described above — those cases are similarly characterized by the inability to instantaneously observe the complete state of features in the world. Chapter 4 describes a new mapping algorithm that takes into account data from multiple poses.

A side effect of extracting features using sensing returns from several poses is that noise in the robot's motion is injected into the feature extraction process, from which it affects the accuracy of the robot's map and pose estimates, sometimes dramatically. Thus, as we might expect, mapping with limited sensing leads to increased uncertainty over approaches that rely on high-fidelity sensors. We must therefore find ways to manage this uncertainty so that it is still possible to produce useful maps with limited sensors. Chapters 5 and 6 address this issue.

1.5 Related work

Robot mapping has a long history and remains an area of intensive research. While much of the recent work on mapping has exploited hardware advances in sensors such as the scanning laser rangefinder, there is an increasing interest in developing mapping algorithms for robots with less expensive sensors such as SONAR arrays and monocular cameras. Others have also begun to ask the simple question: what sensing capabilities are required to build maps? Finally, much of the recent work in mapping has focused on developing and improving filtering techniques for managing uncertainty and consistently estimating the robot's trajectory and maps of the environment.

In this section, we first provide a brief chronology of previous work on mapping with lim-

[*]As noted earlier, some SLAM algorithms avoid feature extraction altogether by matching the raw sensor data with the map. In this case, the same issue arises in the matching step.

ited sensing, and then review in more detail some of the research related to the results presented in this thesis. Chapter 2 provides a much more thorough review of the robot mapping problem, and Chapters 4-7 describe specific related work in detail.

1.5.1 Previous work on mapping with limited sensing

A theme that has spread throughout robotics research — not just in mapping — is the idea of *minimalist* robotics, which asks the question: what capabilities are needed for a robot to complete a given task? In the case of mapping, the question is an eminently practical one: most robot mapping research so far is not easily applicable to inexpensive robots with low-fidelity sensors and small microcontrollers. Researchers have examined the sensing requirements for various other problems, from manipulation (Erdmann and Mason, 1988; Goldberg, 1993; Erdmann, 1995) to localization (Roy and Thrun, 1999; O'Kane and LaValle, 2006b,a) and coverage (Butler et al., 2001). The coverage problem (which is mostly equivalent to mapping) has been examined by Acar et al. (2001) for robots with short-range omnidirectional range sensors.

In the realm of mapping, several researchers have recognized the need for mapping approaches that work with minimal sensing requirements. Tovar et al. (2003) describe a topological mapping algorithm for a robot with a hypothetical "gap" sensor, capable only of detecting the bearings to range discontinuities in the surrounding environment. Beevers (2004) and Huang and Beevers (2005b) describe a mapping algorithm based on very simple behaviors that enables a robot with low-resolution, short-range sensing to build a topological map. Huang and Beevers (2005a) also introduce an algorithm to build Voronoi-based topological maps using similar robots.

While several researchers have previously considered the geometrical mapping problem with various sensing restrictions, this thesis is the first to investigate the case of mapping with the class of sensors having both low spatial resolution and small beam width, such as with small arrays of infrared rangefinders. It is also the first to systematically examine the general sensing requirements for mapping. Doty and Seed (1994) describe a behavior-based approach for building landmark maps with a robot possessing a small infrared rangefinder array and a single SONAR sensor. Grabowski et al. (1999) use a hierarchical team of robots to do SONAR-based mapping. Wijk and Christensen (2000); Zunino and Christensen (2001); Leonard et al. (2002); and Tardós et al. (2002) all address the problem of partially observable features for robots with SONAR arrays. This problem has also been studied in the context of bearing-only mapping for robots equipped with monocular cameras (Deans and Hebert, 2000a; Bailey, 2003; Solá et al., 2005; Lemaire et al., 2005), and in the context of range-only mapping of RF beacons (Kantor and Singh, 2002; Kurth, 2004; Djugash et al., 2005). The ideas of Leonard et al. (2002) are discussed in more detail in Chapter 4, and other work on mapping with partial observability is described further in the following section, along with a discussion of the contributions of the thesis in this area. The algorithms in this thesis are mostly focused on the case of mapping with small arrays of infrared rangefinders, although many of the techniques can be applied independent of sensor modality.

1.5.2 Partial observability

A characteristic common to "limited" sensors is *partial observability*: the sensors are unable to instantaneously observe all the (geometric) parameters of features in the environment, such as walls or corners. Some types of sensing, like range-only radio signals from beacons or bearing-only observations from monocular cameras, are inherently incapable of observing sufficient range-bearing information to perform feature extraction from a single sensor scan. Others, like the returns from SONAR arrays, are limited by characteristics of the sensors such as beam width and noise. A number of researchers have recognized that, when faced with partially observable features, it is necessary to combine multiple observations of the features to recover the state of the environment (Wijk and Christensen, 2000; Zunino and Christensen, 2001; Leonard et al., 2002; Tardós et al., 2002). This thesis employs similar techniques for feature extraction and develops general SLAM algorithms for the case where data associations cannot be determined for the observations from a single scan.

Bearing-only and range-only sensors have recently garnered particular interest in the mapping community. Monocular cameras are relatively inexpensive, offer dense coverage of their field of view, and enable mapping researchers to draw on the vast computer vision literature, e.g., for feature extraction using algorithms like SIFT (Lowe, 2004). Much of the work on mapping with cameras has focused on the initialization issues that arise with partially observable features (Bailey, 2003; Solá et al., 2005; Lemaire et al., 2005).

The interest in range-only mapping stems mainly from the recent surge in sensor networks research, since by far the most common range-only sensor is radio, which relies on beacons in the environment (e.g., sensor nodes). While data association is generally not an issue with radio beacons which can transmit unique identifiers, the large uncertainty and limited information given by radio signal strength observations makes practical range-only mapping a more difficult problem than bearing-only mapping. Much of the work with range-only sensors has focused primarily on localization (Olson et al., 2004; Kurth, 2004; Djugash et al., 2005; Detweiler et al., 2006). Some recent research has examined the SLAM problem using traditional approaches with range-only sensors (Kurth, 2004; Djugash et al., 2005), while others have applied non-traditional interval-based uncertainty representations (Detweiler et al., 2006; Stump et al., 2006) to do localization and mapping with radio beacons.

Although the particle filtering algorithms presented in Chapter 4 of this thesis are applicable to range-only and bearing-only sensors, our focus is mainly on sparse arrays of one-dimensional sensors such as infrared rangefinders. These sensors give too little data for feature extraction or data association, unlike cameras or radio beacons. Furthermore, their small beam width limits the set of observable orientations from a given pose, unlike SONAR sensors, which have wide beam widths that enable an array to cover all of the in-range orientations.

1.5.3 Sensing requirements for mapping

An interesting avenue of recent research, primarily done by O'Kane and LaValle (2006b,a), has examined the sensing requirements for the localization problem. Their work has been to "probe" the boundary between robots that can localize and those that cannot, by considering various combinations of simple sensors and devising algorithms for localizing with each robot, or proving that a particular robot is unable to complete the localization task. While they have yet to apply their approach to mapping, the similarities between localization and mapping are such that doing so would likely yield interesting results.

Chapter 3 of this thesis, however, takes a different approach to uncovering the sensing requirements for mapping. Rather than "probing" the space of mapping robots by devising different algorithms for many combinations of primitive sensors, we instead apply a generalized mapping algorithm and a generic sensor model to broadly characterize the relative capabilities of many common mapping sensors.

1.5.4 Managing uncertainty and improving consistency

Sensors that give very limited information about the environment inherently lead to more uncertainty in the mapping task. A central component of any mapping approach is a filtering technique for managing uncertainty. Commonly, extended Kalman filters or particle filters have been used in mapping. Most of this thesis focuses on designing particle filtering algorithms that work with limited sensing, and that manage uncertainty appropriately using more of the available information than do traditional approaches.

The most common particle filtering SLAM algorithm, FastSLAM (Montemerlo, 2003), works reasonably well in practice but has been shown to be inconsistent in general because it underestimates its own uncertainty (Bailey et al., 2006).

One way to combat inconsistency and improve estimation accuracy is to incorporate prior knowledge about the environment in the mapping process. Most previous work in this realm has focused on enforcing *a priori* known constraints on the map using EKFs (Durrant-Whyte, 1988; Wen and Durrant-Whyte, 1992; Smith et al., 1990). In contrast, the work presented in Chapter 5 of this thesis develops a particle filtering algorithm for exploiting uncertain prior knowledge. The algorithm performs inference in the space of constraints on the map to determine when priors on the relationships between landmarks are applicable. Recent work by Rodriguez-Losada et al. (2006) has performed similar but simpler inference using a thresholding approach, although they employ an EKF for mapping.

A variety of other techniques have been developed for improving the estimation of particle filters for SLAM. (For an introduction to particle filtering SLAM, see Appendix B.) The "improved proposal distribution" (Montemerlo, 2003; Grisetti et al., 2005), also known as FastSLAM 2, uses the most recent sensor observations to draw better Monte Carlo samples for filtering, whereas the traditional particle filter only uses the most recent observations to weight samples drawn from a higher-variance distribution. Much of the inconsistency encountered in particle filtering is due to frequent "resampling" with replacement of the current set of particles. Recent techniques resample only when many of the particles no longer contribute sufficiently to the estimation of the filter (Liu and Chen, 1995; Grisetti et al., 2005). Chapter 6 of this thesis develops new sampling approaches for particle filtering SLAM that lead to more consistent and accurate filtering, using fewer samples, than other current techniques like FastSLAM 2.

1.6 Thesis contributions

The principle contributions of this thesis are as follows.

Theoretical analysis The thesis begins with an analysis of the relative mapping abilities of different sensors. Whereas most previous research has focused on determining whether (and how) specific sensors can build maps, the analysis in this thesis instead applies a single mapping algorithm with a broad class of sensors. Using this approach, and under several simplifying assumptions on the robot and the environment, we have obtained conditions on each sensor for convergence of the map, and bounds on the error in the map for sensors that converge.

Sparse sensing Most prior mapping research has relied on sensors that give data "dense" enough to extract features from a single scan. This thesis describes an algorithm for passive mapping with *sparse* (low spatial resolution) sensors, which uses sensor data from multiple consecutive poses in the robot's trajectory to do feature extraction, offsetting the sensing limitation by exploiting the robot's motion. We introduce simplifications to a system model for this approach that enable efficient implementation in a particle filtering framework, and describe enhancements to the algorithm to enable a tradeoff between filtering accuracy and computation time.

Prior knowledge The thesis develops a new algorithm for incorporating prior knowledge, in the form of relative constraints on map landmarks, into the mapping process. The use of prior knowledge about the environment is one way of reducing the large uncertainty associated with limited sensing. The algorithm uses basic prior information, such as "most walls indoors are rectilinear," to infer when landmarks should be constrained. It also employs a new technique for enforcing constraints, based on a particle filtering approach.

Sampling techniques One of the most restricting problems with current particle filtering algorithms for mapping is the "inconsistency" of the filter. (A filter is inconsistent if it significantly underestimates its own uncertainty.) Filtering inconsistencies typically result in poor data associations, incorrect loop closings, and bad trajectory estimation. Inconsistencies in a particle filter can be partially alleviated by drawing better Monte Carlo samples. This thesis develops two new sampling strategies, fixed lag roughening and the block proposal distribution, that exploit more information than the usual sampling techniques and thus give better samples.

SLAM on a microcontroller The thesis demonstrates a working implementation of a particle filtering SLAM algorithm on a minimalist robot with an 8-bit microcontroller. In contrast, most SLAM implementations make use of modern, powerful PCs. The implementation described in this thesis shows that real time mapping is possible with inexpensive sensing and computing suitable for deployment in, e.g., consumer products.

Implementation and experimentation Finally, in addition to the above implementation, all of the algorithms described in the thesis have been implemented in full. The results in the thesis are verified experimentally using a combination of simulation, a variety of well-known benchmark data sets, and new data collected with our own robots.

1.7 Thesis outline

The remainder of the thesis proceeds as follows.

Chapter 2 provides a more thorough introduction to the robot mapping problem and discusses a number of aspects of the problem in the context of limited sensing, including robot and environment models, uncertainty representations, online, offline, passive, and active mapping, feature extraction, data association, and more. Intertwined with this discussion is a detailed review of the literature on robot mapping and on other approaches for mapping with sensing limitations.

Chapter 3 develops a simple but reasonably complete model for mapping sensors. The model is employed with some simplifying assumptions and an occupancy grid mapping algorithm to obtain a bound on maximum likelihood map error in terms of the characteristics of the sensor. The analysis is applied to real sensors to compare their effectiveness for mapping.

Chapter 4 describes an algorithm for mapping with sparse sensing and presents experimental results from real-world data sets.

Chapter 5 introduces a technique for exploiting prior knowledge as constraints on the map. An algorithm is described for enforcing constraints, as is a technique for performing inference in the space of constraints. The algorithms are implemented for a rectilinearity constraint. Experimental results with both real-world and simulated data show the efficiency and consistency improvements given by the approach.

Chapter 6 describes several new sampling strategies for use in particle filtering. Fixed lag roughening and the block proposal distribution exploit "future" information to improve estimation consistency. Simulated and real-world results show the approaches can lead to dramatic improvements. The chapter also tests two alternative resampling strategies, residual and generalized resampling.

Chapter 7 demonstrates the feasibility of real time SLAM on limited hardware with an implementation of particle filtering SLAM on a robot with a five-sensor infrared rangefinder array and an 8-bit microcontroller as the main processor. The implementation shows that SLAM can be done with limited sensing, computing, and storage on inexpensive commodity hardware.

Chapter 8 summarizes the previous chapters and the contributions of the thesis.

Appendix A provides an overview of techniques for generating samples from complicated distributions. Importance sampling, particle filtering, and Markov chain Monte Carlo approaches are discussed. These techniques are heavily used in the mapping algorithms described in the preceding chapters.

Appendix B introduces the probabilistic formulation of simultaneous localization and mapping (SLAM) upon which much of the thesis is based. SLAM is first formulated in a Bayesian context. Then, an implementation using an extended Kalman filter (EKF) is described. Finally, a widely used particle filtering technique, the basis for the algorithms described in Chapters 4-6, is introduced.

Appendix C describes the software used to obtain the results in the thesis. The software is available online at `http://www.cs.rpi.edu/~beevek/thesis/`.

2 THE MAPPING PROBLEM

Simplistically, the *robot mapping problem* is for a mobile robot to autonomously build a model of its environment based on feedback from its sensors. In reality there are many considerations that affect the scope of the problem, including the models of the robot and world, the representation of uncertainty, the nature (online or offline, passive or active) of the mapping algorithm, and so on. This chapter provides a broad overview of the mapping problem, highlighting a number of mapping approaches and making special note of sensing and computational considerations.

2.1 Robot model

Before we can even begin to discuss the specifics of the mapping problem we must first understand the capabilities of the robot that will build the maps. We have already discussed various sensing capabilities in Chapter 1, so here we focus on the underlying kinematic model of the robot. In reality, robots operate in a three-dimensional ($\mathbb{R}^3$) workspace, although often a planar ($\mathbb{R}^2$) workspace is assumed instead. Most planar robots are *oriented* (i.e., they have a forward-facing direction). A common model assumes the robot's *configuration* or *state* $\mathbf{x}^r$ can be represented by a point in the *configuration space* $\mathcal{C} \subseteq \mathbb{R}^2 \times \mathbb{S}^1$ (the $SE(2)$ manifold), i.e.:

$$\mathbf{x}^r = [x^r \;\; y^r \;\; \theta^r]^T \tag{2.1}$$

where (x^r, y^r) is the robot's position and θ^r is its orientation with respect to the x-axis of the coordinate frame. This model, depicted in Figure 2.1, is adopted throughout this thesis.

The robot state evolves according to a discrete-time *process* or *motion model*, i.e.:

$$\mathbf{x}^r_t = f(\mathbf{x}^r_{t-1}, \mathbf{u}_t) + \mathbf{v}_t \tag{2.2}$$

where $\mathbf{u}_t$ indicates a control input at time t and $\mathbf{v}_t$ is a random vector representing noise in the process, which is typically assumed to be zero-mean. Commonly, control inputs specify translation and rotation distances, i.e.:

$$\mathbf{u} = \begin{bmatrix} d \\ \alpha \end{bmatrix} \tag{2.3}$$

$\mathbf{x}^r_t, \mathbf{P}^r$	pose estimate/covariance	$\mathbf{u}_t$	control input at time t	GVD	generalized Voronoi diag. (L_2)
$\mathbf{x}^m, \mathbf{P}^m$	map estimate/covariance	$\mathbf{z}_t$	sensor measurement at time t	GVD$_\infty$	generalized Voronoi diag. (L_∞)
$\mathbf{x}^m_i$	ith map landmark	$\mathbf{n}_t$	correspondences at time t	SGVD$_\infty$	saturated GVD (L_∞)

Notation for this chapter

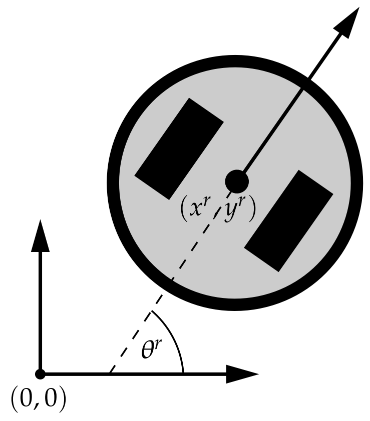


Figure 2.1: A planar robot with configuration $[x^r \;\; y^r \;\; \theta^r]^T$**.**

A simple process model for such an input is:

$$f(\mathbf{x}^r_{t-1}, \mathbf{u}_t) = \begin{bmatrix} x^r_{t-1} + d\cos\theta^r_{t-1} \\ y^r_{t-1} + d\sin\theta^r_{t-1} \\ \theta^r_{t-1} + \alpha \end{bmatrix} + \mathbf{v}_t \tag{2.4}$$

A number of alternative (and more complicated) motion models exist and may be applied, depending on the particular robot; see, e.g., Thrun et al. (2005), Choset et al. (2005), or Chapter 7 of this thesis.

Irrespective of the particular types of sensing available to a robot, the robot's *observation* or *measurement model* can be written as:

$$\mathbf{z}_t = h(\mathbf{x}^r_t, \mathbf{x}^m_t) + \mathbf{w}_t \tag{2.5}$$

where $\mathbf{x}^m_t$ is some model of the environment.[*] In other words, the measurement returned by the robot's sensors is a function of the robot's configuration and the environment, and is subject to noise, which we again generally assume is zero-mean. The nature of h depends on a number of factors, most importantly, the feature extraction approach we employ. (Section 2.7 discusses feature extraction in more detail.)

2.2 Representing uncertainty

A robot in the real world is subject to uncertainty due to sensor noise, sensing limitations, model approximations, unmodeled aspects of the environment, and so on. How this uncertainty is represented is crucial to building any kind of map.

2.2.1 Probabilistic models

By far the most widely used approach in robotics for modeling uncertainty is to represent uncertain knowledge using probability density functions (PDFs). For example, uncertainty about the configuration of the robot can be modeled as a PDF $p(\mathbf{x}^r)$ over the robot's state space. Inputs such as control commands or sensing data can be incorporated by *conditioning*. For example, the uncertainty in the robot's state $\mathbf{x}^r_t$ at time t, given a time series $\mathbf{u}_{1:t}$ of previous control inputs and a time series $\mathbf{z}_{1:t}$ of sensor measurements, can be modeled using the conditional PDF:

$$p(\mathbf{x}^r_t | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}) \tag{2.6}$$

[*]Frequently the environment is assumed static and the time subscript dropped from $\mathbf{x}^m$; see Section 2.9.4 for discussion of this assumption.

Motion and measurement models can also be represented probabilistically, i.e. by:

$$p(\mathbf{x}_t^r|\mathbf{x}_{t-1}^r, \mathbf{u}_t) \tag{2.7}$$

and:

$$p(\mathbf{z}_t|\mathbf{x}_t^r, \mathbf{x}_t^m) \tag{2.8}$$

2.2.1.1 Parametric representations

Probabilistic models of uncertainty are conceptually straightforward but in practice bring up the problem of PDF representation. Methods of representing PDFs can be classified as either *parametric* or *nonparametric*. A parametric PDF is a parameterized equation that meets the conditions of a probability density. The most common parametric PDF in robotics is the Normal (or Gaussian) density:

$$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})^T\right) \tag{2.9}$$

where $\boldsymbol{\mu}$ is the n-dimensional mean vector and $\boldsymbol{\Sigma}$ is an $n \times n$ covariance matrix. Frequently Gaussian representations are assumed for the motion and observation models, in which individual components of the control input or measurement are assumed to have independent noise. For example, the motion model is commonly assumed to be:

$$p(\mathbf{x}_t^r|\mathbf{x}_{t-1}^r, \mathbf{u}_t) = \mathcal{N}\left(f(\mathbf{x}_{t-1}^r, \mathbf{u}_t), \mathbf{H}_\mathbf{u} \begin{bmatrix} \sigma_d^2 & 0 \\ 0 & \sigma_\alpha^2 \end{bmatrix} \mathbf{H}_\mathbf{u}^T\right) \tag{2.10}$$

where the Jacobian

$$\mathbf{H}_\mathbf{u} = \frac{\partial f(\mathbf{x}^r)}{\partial \mathbf{u}} \tag{2.11}$$

is used to obtain a linear approximation of the nonlinear function f.

Figure 2.2(b) shows an example of a Gaussian representation of a 1D PDF. Many mapping approaches represent the entire state $\mathbf{x} = [\mathbf{x}^r \ \ \mathbf{x}^m]^T$ (robot pose and map) parametrically, e.g., by a single large Gaussian, an approach first applied in the seminal work of Smith et al. (1990). With a Gaussian representation of the state uncertainty, it is easy to apply an extended Kalman filter (EKF) to do geometrical mapping. See Appendix B for a discussion of the basic EKF-based mapping approach.

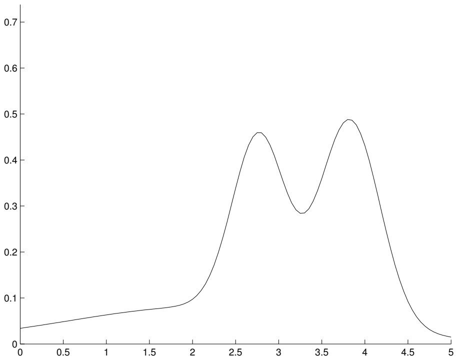
2.2.1.2 Nonparametric representations

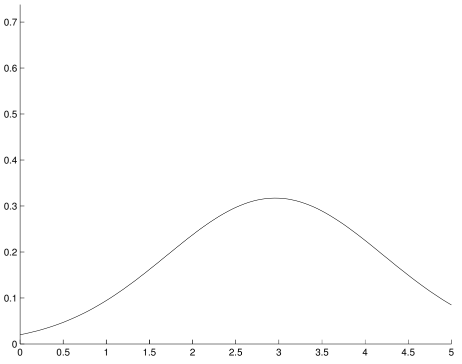
The use of a parametric representation such as the Gaussian density to model state uncertainty assumes implicitly that the true state uncertainty is approximately distributed according to the specified distribution. Most of the time in the real world, uncertainty is poorly represented by parametric distributions. For example, the sensor measurements of a robot with a known map may match well with several configurations of the robot, in which case $p(\mathbf{x}^r)$ is multimodal, a property that cannot be captured by most common parametric PDFs.

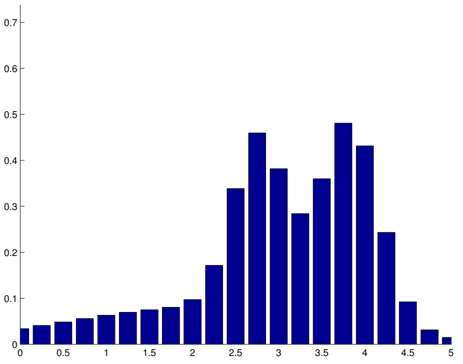
One possible remedy is to model uncertainty using *mixtures* of parametric PDFs in which a set of weighted distributions, each with different parameters, is combined (and normalized) to represent a complicated density. Another alternative that has proved particularly useful for robot mapping is to represent PDFs nonparametrically. The most common nonparametric PDF representations are grid- or sample-based.

Grids are essentially discretizations of the state space in which each grid cell is assigned a portion of the probability mass from a continuous PDF. Figure 2.2(c) shows an example of a grid representation of a 1D PDF. While grid representations have been applied to localization problems (e.g. Fox et al., 1999), they have mostly been subsumed by sample-based approaches for computational reasons.

Sampled representations approximate the PDF by a set of (possibly weighted) samples drawn from the state space according to the PDF using methods such as importance sampling.


(a) True distribution


(b) Gaussian representation


(c) Grid representation

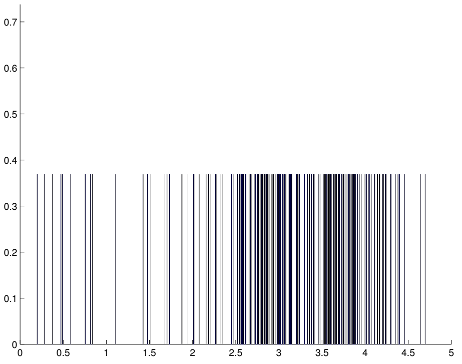

(d) Sampled representation

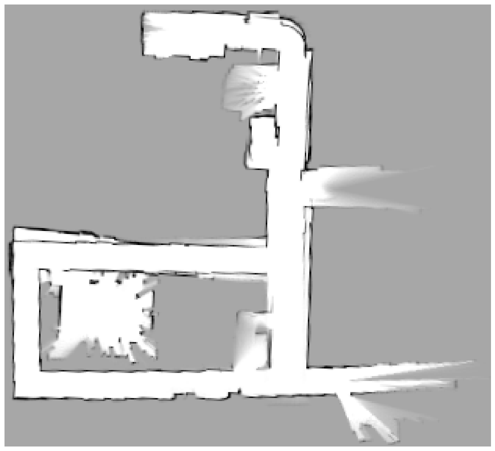
Figure 2.2: Probabilistic representations of a multimodal 1D PDF. Note how the nonparametric approximations capture the multimodal nature of the true PDF, whereas the parametric Gaussian approximation does not.

Figure 2.2(d) shows how a PDF can be represented by samples. Sampled representations have led to the application of Monte Carlo techniques such as particle filtering for robot mapping. Much of this thesis is based upon sampling approaches. Appendix B introduces the basic particle filtering formulation of the mapping problem; Appendix A offers some background on the underlying statistical techniques.

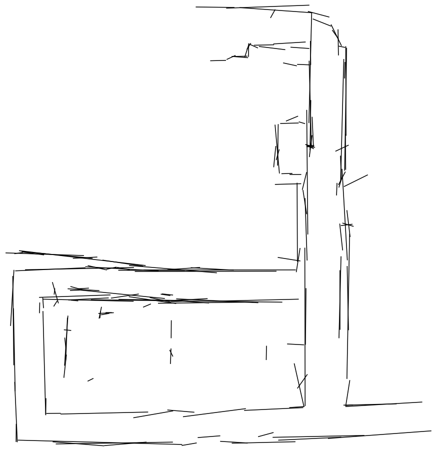
2.2.2 Interval models

A deficiency of probabilistic uncertainty representations is that they generally require distribution assumptions that are only weakly justifiable. For example, a common laser observation model assumes zero-mean Gaussian noise, whereas in reality the noise is neither zero-mean nor Gaussian. An alternative is to use a *bounded uncertainty* (or *interval*) model which represents uncertainty using a set-based approach (see Halpern, 2003). Only the maximum possible error of a measurement need be specified. Then, the portion of the state space within this bound is a set representing the uncertainty of the measurement. Interval models have been applied to the localization problem for bearing-only sensors (Briechle and Hanebeck, 2004; Isler and Bajcsy, 2005), and to localization and mapping for range-only sensors (Detweiler et al., 2006; Stump et al., 2006). This thesis employs an interval model of sensor uncertainty in Chapter 3 to obtain bounds on map error in terms of the uncertainty (and other sensor parameters).

Difficulties arise with interval representations when it is necessary to select a single, "representative" state from the interval, e.g., for use in planning. A natural solution to this problem exists for probabilistic techniques: take the most probable (maximum likelihood) state. Interval



(a) Occupancy grid map



(b) Landmark map with line segment landmarks

Figure 2.3: Occupancy grid and landmark-based geometrical maps. An occupancy grid discretizes the environment into small cells and maintains a belief that each cell is occupied or free. A landmark map represents the world as a set of parameterized geometrical objects, in this case line segments. Data set courtesy of Nicholas Roy via Radish (Howard and Roy, 2003).

representations assume no PDF over the feasible set so no feasible state is "better" than any other.

2.2.3 Other approaches

While some other representations of uncertainty exist, they have found little application in robot mapping. One exception is the *Dempster-Shafer* representation of belief (Shafer, 1976). Dempster-Shafer is similar to probabilistic techniques but explicitly represents ignorance. In a state of complete ignorance, Dempster-Shafer assigns no belief mass to any specific possibility. Probabilistic approaches instead implicitly represent ignorance by assigning equal belief to all possibilities. The Dempster-Shafer uncertainty representation has found applications in occupancy grid mapping (Pagac et al., 1998), in active topological mapping (Beevers, 2004; Beevers and Huang, 2005), and in sensor fusion (Murphy, 1998), but has computational drawbacks that limit its usefulness in most robot mapping scenarios.

The book by Halpern (2003) offers a thorough discussion of the uncertainty representations described above, along with several others.

2.3 Geometrical mapping

Paramount to the development of any mapping algorithm is the model used to represent the map. Two basic types of maps have been developed in the robotic mapping literature: *geometrical* maps, which encode metric information about the locations and relationships of objects in the environment; and *topological* maps, which encode "qualitative" structural information about the relationships of objects. Many mapping techniques combine the two approaches to build *hybrid* maps.

Geometrical maps are a natural way to model the environment because sensors measure geometrical aspects of the world, e.g., distances or bearings to obstacles. Typically, geometrical maps fall into one of two categories: *occupancy* maps and *landmark* maps.

2.3.1 Occupancy maps

An occupancy representation divides the world into grid cells (Moravec and Elfes, 1985). Associated with every cell is a belief about the occupancy of the cell (i.e., whether there is an obstacle in the cell). Typically the belief is represented as a probability and is updated using Bayes' rule whenever the cell is observed. Alternative approaches that employ Dempster-Shafer beliefs have also been developed (Pagac et al., 1998). Updates to an occupancy grid depend strongly on the probabilistic sensor model being used. Konolige (1997); Pagac et al. (1998) and Thrun (2003) have described models for SONAR sensors. Models for more accurate range-bearing sensors such as lasers or infrared sensors are generally straightforward.

Figure 2.3(a) shows an occupancy grid map of an indoor environment. Intensity of a cell (pixel) indicates its probability of being unoccupied. Dark cells correspond to walls or other obstacles, light cells to free space, and gray cells to unobserved or uncertain areas.

Implementation of occupancy grids is relatively straightforward, and they lend themselves well to range scan matching techniques for data association (discussed in Section 2.8). For these reasons they have proved fairly popular and are employed in a number of state-of-the art mapping implementations (e.g. Grisetti et al., 2005). However, grid maps impose significant memory and computational requirements that often make them unsuitable for use in inexpensive robots. Furthermore, the "convergence" of an occupancy grid map relies on obtaining many observations of each grid cell. While this is reasonable with high-resolution sensors such as laser rangefinders, it is more difficult for robots equipped with simpler sensors such as infrared rangefinder arrays. Chapter 3 examines occupancy mapping in detail in the context of limited sensing.

2.3.2 Landmark maps

An alternative approach is to represent the world as a set of objects, or *landmarks*. Typically, landmarks are modeled as simple geometric primitives such as lines or points. By representing the environment in this way, landmark parameters can be incorporated into the system state vector, i.e., $\mathbf{x} = [\mathbf{x}^r \quad \mathbf{x}^m]^T$, where $\mathbf{x}^m$ is the map. For example, suppose the map consists of line landmarks, with each line parameterized by its distance r and angle θ to the origin. (This representation is frequently used in this thesis.) Then, our goal is to estimate a map:

$$\mathbf{x}^m = [\mathbf{x}_1^m \;\; \mathbf{x}_2^m \;\; \ldots \;\; \mathbf{x}_n^m]^T = [r_1 \;\; \theta_1 \;\; r_2 \;\; \theta_2 \;\; \ldots \;\; r_n \;\; \theta_n]^T \tag{2.12}$$

Using a probabilistic representation of the state uncertainty, we wish to compute:

$$p(\mathbf{x}^m | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \tag{2.13}$$

i.e., a PDF over maps conditioned on the control inputs $\mathbf{u}_{1:t}$, sensor measurements $\mathbf{z}_{1:t}$, and *correspondences* $\mathbf{n}_{1:t}$ between measurements and landmarks in the map. (The problem of finding correspondences, known as *data association*, is discussed below.)

Many researchers have employed landmark-based mapping techniques, e.g., Smith et al. (1990); Newman (1999); Bailey (2002); Leonard et al. (2002); Montemerlo (2003). Figure 2.3(b) shows a landmark map of an indoor environment consisting of line features. Extent information, encoding which portions of the landmarks have been observed, is used for rendering purposes but is not included as part of the map state.

2.3.2.1 Initialization

A critical issue in landmark mapping is the initialization of new landmarks when they are first observed. If a sensor return provides sufficient data to observe the entire state of a new landmark (i.e., to *extract features*), landmark initialization is relatively straightforward: the map vector $\mathbf{x}^m$ is extended to include the observed new features. (Other initialization may be required, depending on the uncertainty representation used for mapping.) Section 2.7 discusses techniques for extracting features from sufficient range-bearing data.

Unfortunately, most inexpensive sensors like monocular cameras and infrared or SONAR arrays provide insufficient data to initialize a new landmark from a single return. It is therefore necessary to extract features using data from several robot poses. How this is done depends on the sensing modality. Various researchers have investigated multi-pose landmark initialization using SONAR data (Wijk and Christensen, 2000; Zunino and Christensen, 2001; Tardós et al., 2002; Leonard et al., 2002); monocular cameras (Deans and Hebert, 2000a; Bailey, 2003; Lemaire et al., 2005; Solá et al., 2005); and RF or acoustic range measurements to beacons (Kantor and Singh, 2002; Kurth, 2004; Olson et al., 2004; Djugash et al., 2005). This thesis addresses the case of infrared rangefinder arrays, which, of the above modalities, are most similar to SONAR arrays. In Chapter 4 we discuss in more detail existing approaches for multi-pose landmark initialization using SONAR, and develop a new approach which we apply to infrared arrays.

2.4 SLAM

Note that in computing the map PDF (Equation 2.13) it is convenient to also estimate the robot's pose, i.e., to jointly estimate the robot state and the map:

$$p(\mathbf{x}^r_t, \mathbf{x}^m | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \tag{2.14}$$

This is the *simultaneous localization and mapping* (SLAM) problem. The vast majority of robot mapping research falls into the context of SLAM, since a primary factor in constructing an accurate map is knowledge of the robot's location with respect to the map. The SLAM formulation is not unique to landmark-based mapping and is equally applicable in constructing any type of map, although it is most often applied in conjunction with geometrical representations.

Since SLAM techniques form the basis for much of this thesis, we introduce them in some detail here. For a more complete discussion, Appendix B fully formulates the problem and describes the two most widely used approaches for doing SLAM.

2.4.1 Bayesian formulation

One can address SLAM in a Bayesian filtering framework in which the posterior (2.14) is estimated recursively:

$$p(\mathbf{x}_t | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \quad = \quad \eta p(\mathbf{z}_t | \mathbf{x}_t, \mathbf{n}_t) \int p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{u}_t) p(\mathbf{x}_{t-1} | \mathbf{u}_{1:t-1}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t-1}) \, d\mathbf{x}_{t-1} \tag{2.15}$$

Here, η is a normalization constant. (It is generally sufficient to estimate the posterior up to a normalization factor.) Typically, the filtering is done with an EKF, which models the posterior as a multivariate Gaussian. A derivation of Equation 2.15 and an EKF-based implementation can be found in Appendix B.

2.4.2 Particle filtering SLAM

An alternative is to incorporate the entire robot *trajectory* or *pose history* into the state vector:

$$\mathbf{x} = [\mathbf{x}^r_1 \;\; \mathbf{x}^r_2 \;\; \ldots \;\; \mathbf{x}^r_t \;\; \mathbf{x}^m]^T \tag{2.16}$$

Estimating the posterior over this space is often called the "full" SLAM problem (see Thrun et al., 2005), since the goal is to recover not just the map and current robot pose, but also the path taken by the robot. It has been observed that, assuming the environment is static and the state is *complete*,* landmarks in the map are independent when conditioned on the robot's trajectory (Murphy, 2000). (See Section 2.9.4 for further discussion of this assumption, often called the *Markov assumption*; also see Chapter 5, which argues that in structured environments the Markov assumption is not met.) This independence occurs because correlation between independently distributed landmarks arises only through robot pose uncertainty. Murphy (2000) used this fact to show that the posterior over robot trajectories and maps can be factored:

$$p(\mathbf{x}^r_{1:t}, \mathbf{x}^m | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) = p(\mathbf{x}^r_{1:t} | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \prod_{i=1}^{n} p(\mathbf{x}^m_i | \mathbf{x}^r_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \tag{2.17}$$

*The state $\mathbf{x}_t = [\mathbf{x}^r_{1:t} \;\; \mathbf{x}^m]^T$ is complete if no state variables or observations prior to time t may influence the stochastic evolution of future states (Thrun et al., 2005).

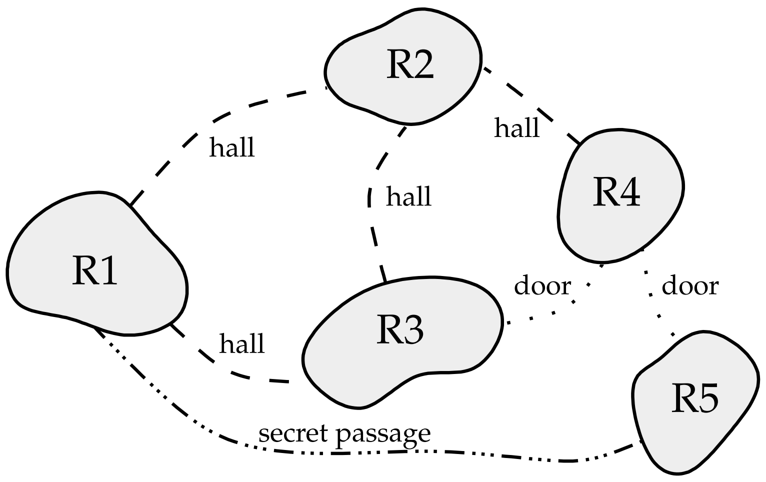


Figure 2.4: A simple topological map in which nodes represent rooms and edges represent different types of paths between rooms. A robot capable of room recognition, doorway recognition and traversal, hall following, and secret passageway finding can navigate with this map. (This map was hand-made for illustration purposes.)

This factorization is known as Rao-Blackwellization (Liu, 2001). It has led to the development of the current state-of-the-art in robot mapping: *particle filtering* techniques that represent the posterior over trajectories, $p(\mathbf{x}^r_{1:t}|\mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t})$, using samples (particles). Associated with each particle is a map consisting of a number of separate small filters (typically EKFs) for each landmark. This approach is known as Rao-Blackwellized particle filtering (RBPF) and is the basis for the well-known FastSLAM algorithm (Montemerlo, 2003) and variations, e.g., (Grisetti et al., 2005; Stachniss et al., 2005a).

While other techniques have been developed to solve the full SLAM problem, e.g., (Lu and Milios, 1997; Thrun et al., 1998a), they are mainly intended for offline mapping and are thus less interesting than the RBPF approach, which can be applied recursively and online. A detailed introduction to the RBPF algorithm for SLAM can be found in Appendix B.

2.5 Topological mapping

A markedly different approach to map representation is to maintain purely structural information about places in the environment and relationships between these places, i.e., to encode the *topology* of the environment rather than its geometry. Topological maps, first described by Kuipers (1978) and since developed into a more general theory (Kuipers, 2000; Remolina and Kuipers, 2004), are typically actualized as graphs, in which vertices correspond to *distinctive configurations* in the environment and edges represent *distinctive paths* between these configurations. An important advantage of the topological approach is the ease of planning and navigating with a topological map, since simple graph search techniques can be directly applied.

2.5.1 Distinctive configurations and paths

Distinctive configurations are robot states at which the exteroceptive sensor returns meet some particular conditions for distinctiveness. These conditions depend upon the underlying representation of the environment's topology. We discuss several topology representations below and address the issue of recognizing distinctive states.

Typically, distinctive paths in topological maps correspond to paths that can be followed by a robot executing simple behaviors, based mainly on local sensory information. For example, in previous work (Beevers, 2004; Huang and Beevers, 2005c), we have built topological maps

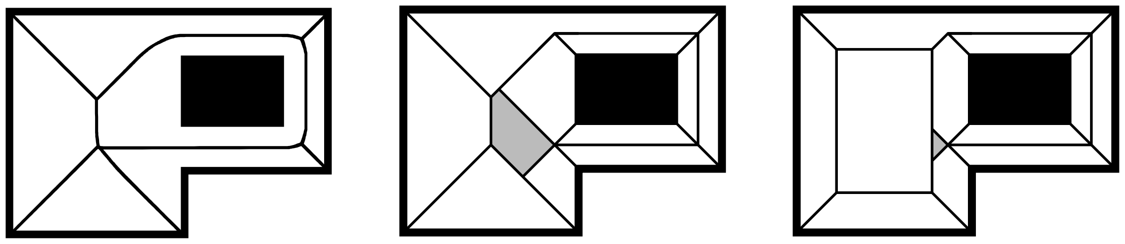

(a) GVD (b) GVD_∞ (c) SGVD_∞

Figure 2.5: Voronoi diagrams.

with sparse infrared rangefinder arrays by executing simple wall- and hall-following behaviors between corners and junctions in an indoor environment.

Often, topological maps are augmented with geometrical information such as odometry readings (e.g. Kuipers and Byun, 1991; Rybski et al., 2003; Beevers and Huang, 2005) or sensor "signatures" at distinctive configurations (e.g. Kuipers and Beeson, 2002). Occasionally, structural information can be extracted directly from an available geometrical map (as in Chatila and Laumond, 1985; Thrun and Bücken, 1996), but usually topological maps are built online by an actively exploring robot (e.g. Kuipers and Byun, 1991; Choset and Nagatani, 2001; Tovar et al., 2003).

Figure 2.4 shows an example of a simple topological map. Distinctive places correspond to rooms, and distinctive paths to various ways of moving between the rooms. A robot that is able to recognize particular rooms, find exits from the rooms, and traverse doorways and corridors can build the map by actively exploring the graph.

2.5.2 Voronoi-based approaches

A popular way to represent the structure of a workspace is by its *Voronoi diagram* (see Aurenhammer, 1991; Latombe, 1991) or *medial axis*. Given a set of points in the plane, or *sites*, the Voronoi diagram is defined as the locus of points equidistant to the two closest sites. Alternatively, the Voronoi diagram can be defined as the union of the boundaries of the *Voronoi regions* of the sites. The Voronoi region of a site is the set of points closer to that site than to any other. An advantage of Voronoi-based maps is that they offer good (in fact, the best possible) clearance properties for navigation when following paths on the Voronoi diagram.

Voronoi diagrams can also be defined in higher dimensions and with different distance metrics such as the L_∞ metric (see Lee, 1980). For mapping in realistic environments, it is useful to define Voronoi diagrams with non-point sites, termed *generalized Voronoi diagrams* (GVDs) (see Kirkpatrick, 1979; Lee and Drysdale, 1981). Figure 2.5(a) depicts an example of the GVD of a simple polygonal environment; Figure 2.5(b) shows the the GVD computed using the L_∞ distance metric, which we call the GVD_∞ (Huang and Beevers, 2005a). Under the L_2 (Euclidean) metric a GVD consists of line segments and circular arcs; under the L_∞ metric it consists of line segments and regions (as with the gray region in the figure). One can represent the GVD as a graph in which intersections of Voronoi segments (*meet points*) are nodes, and segments and region boundaries are edges.

2.5.2.1 Mapping the GVD

Choset (1997) and Choset and Nagatani (2001) have shown that the GVD of an environment can be mapped by a robot using relatively simple local behaviors. A robot can "trace" the Voronoi diagram by moving along the arc equidistant to the two nearest obstacles. Meet points are easily recognized when more than two obstacles are equidistant to the robot. Thus, by incrementally tracing Voronoi arcs a robot can actively explore the GVD to build a topological map, or use the GVD to guide exploration when building a geometrical map. However, this approach relies on the ability of the robot to detect the range and bearing to obstacles with an omnidirectional, infinite-range sensor, since the distance of GVD arcs from obstacles is unrestricted.

2.5.2.2 *The* SGVD *and* SGVD$_\infty$

In the context of coverage, Acar et al. (2001) extended the GVD approach to deal with short-range sensing by introducing the *saturated generalized Voronoi diagram* (SGVD). In regions where the distance to the two closest sites is less than the "saturation distance" (sensor range), the SGVD is identical to the GVD. At points on the GVD that are equidistant to two sites such that the distance to the sites is equal to the saturation distance, the SGVD "diverges," maintaining the saturation distance to a single site. Formally, if $\mathcal{M}$ is the Minkowski sum of the workspace obstacles with a disk of radius equal to the saturation distance, then:

$$\text{SGVD} = (\text{GVD} \cap \mathcal{M}) \cup \partial\mathcal{M} \tag{2.18}$$

where $\partial\mathcal{M}$ denotes the boundary of $\mathcal{M}$ (Huang and Beevers, 2005a). The SGVD can also be traced with relatively simple behaviors and used as a topological map. This approach is related to the idea of "coastal navigation" (Roy and Thrun, 1999) in which a robot attempts to stay within sensing range of objects in the environment to minimize uncertainty.

In (Huang and Beevers, 2005a), behaviors are developed that enable a robot to explore a version of the SGVD defined under the L_∞ distance metric (the SGVD$_\infty$), under the assumption that the environment is rectilinear. The advantage of the SGVD$_\infty$ is that its arcs are aligned along a fixed set of orientations. Thus, it is possible to develop behaviors for tracing the SGVD$_\infty$ using sparse (low-resolution) sensors. Huang and Beevers (2005a) exhibit an algorithm for exploring the SGVD$_\infty$ using an array of infrared sensors and show that it is complete. Figure 2.5(c) depicts the SGVD$_\infty$ of a simple environment.

2.5.3 Probabilistic approaches

Voronoi-based topological mapping techniques lead to map representations and exploration strategies for active mapping, but do not inherently address the issue of uncertainty in the mapping process like probabilistic geometrical mapping algorithms. Typically a robot will experience some error in place recognition and path following, particularly when "places" are not completely unique, so representing uncertainty in the robot state and the topological map is an important issue.

Several techniques have incorporated probabilistic measurement and motion models directly into the topological mapping process. One approach is to represent the topological map as a hidden Markov model (HMM) with the goal of recovering the steady-state transition probabilities between distinctive places (e.g. Koenig and Simmons, 1996; Shatkay and Kaelbling, 2002). More recently, Ranganathan et al. (2005) and Ranganathan and Dellaert (2006) have introduced techniques based on Markov chain Monte Carlo (MCMC) and RBPF sampling methods that estimate a joint posterior over the locations of distinctive places and the paths between them. They equate topologies of the environment with set partitions of distinctive place observations, and use this approach to do inference in the space of topologies. Similar approaches for inferring topology using Bayesian techniques have been described in (Modayil et al., 2004) and (Savelli, 2005).

2.6 Hybrid techniques

The strength of geometrical map representations is their ability to accurately reflect the *local* relationships between objects in the environment. On the other hand, the reduced representations used by topological techniques are conducive to recovering *global* information about the relationships between important places in the environment. Several mapping algorithms have been developed to exploit this dichotomy by building *hybrid* maps that include both topological and geometrical information.* While this thesis is mainly focused on geometrical mapping, the tech-

*Combining topological and geometrical mapping techniques is not the only notion of a "hybrid" map. For example, Guivant et al. (2004) combine landmark-based and occupancy grid mapping techniques to exploit advantages of both

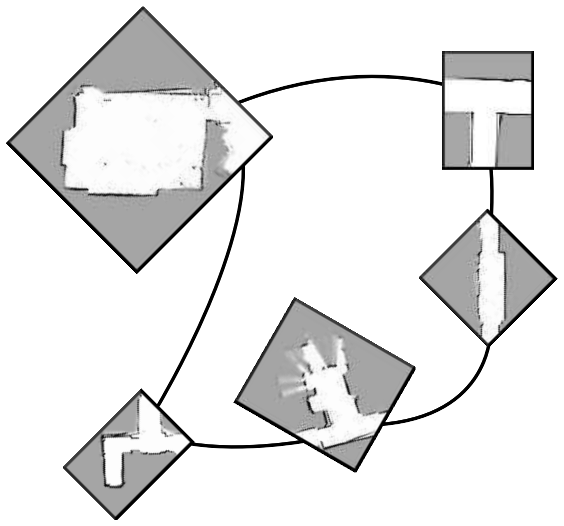


Figure 2.6: One type of hybrid topological-geometrical map. Here, local occupancy grid maps are built at each distinctive place in the environment and stored as topological nodes. Simple behaviors take the robot between distinctive places. (This map was hand-made for illustration purposes.)

niques may be useful in hybrid mapping scenarios, so we review hybrid maps briefly here.

Early approaches simply extracted topological maps from geometrical maps (e.g. Chatila and Laumond, 1985; Thrun and Bücken, 1996). This approach relies on the availability of a topologically correct geometrical map, but does not actually exploit structure when building the geometrical map. Similarly, some techniques first build topological maps and then extract geometrical maps, e.g., by taking a second pass through the raw data as in (Thrun et al., 1998c).

Other approaches incorporate basic geometrical information into topological maps as annotations of nodes and edges (e.g. Kuipers and Byun, 1991; Shatkay and Kaelbling, 2002; Huang and Beevers, 2005c; Beevers and Huang, 2005). This information is used to ensure consistent topologies and can also be used to embed topological maps in a metric space, e.g., the plane as in (Duckett and Saffiotti, 2000; Frese et al., 2005; Olson et al., 2006).

Two dominant hybrid mapping paradigms have emerged, and we discuss these in more detail.

2.6.1 Geometrical maps at topological nodes

One can exploit the strengths of each mapping technique directly by building geometrical maps in the vicinity of each distinctive state (node) in the topological map. The geometrical maps are small in scale and are typically represented as occupancy grids (e.g. Modayil et al., 2004; Kuipers et al., 2004; Savelli, 2005) or landmark-based maps (e.g. Tomatis et al., 2002a,b). Because of the limited size of the geometrical maps, they are mostly unaffected by error propagation in the robot's pose estimate. Figure 2.6 shows a simple example of this type of hybrid map.

Building accurate geometrical maps at nodes yields several important advantages in constructing a global topological map:

- Rather than requiring the robot to exactly move to the distinctive configuration associated with the node — which is normally done with a potentially time-consuming hill-climbing strategy (Kuipers and Byun, 1991) — the robot can use the local geometrical information to localize itself with respect to the distinctive configuration (Kuipers et al., 2004).

approaches.

- By constructing an accurate map of the area around the topological node, the robot can directly extract local structure from the geometrical information, e.g., the set of paths leading away from a node in the topological map. For example, Kuipers et al. (2004) construct the Voronoi diagram of a local geometrical map. Outgoing edges that overlap the edges of the local map are considered distinctive paths connected to the node in the global topological map.
- Maps at nodes can be used as "sensing signatures" for accurately recognizing the location of the robot in the topological map, as in (Martinelli et al., 2003).

Hybrid maps based on the idea of building geometrical maps at topological nodes have been applied in several circumstances. Martinelli et al. (2003) use a scanning laser rangefinder to build local maps and a fingerprinting approach based on images from an omnidirectional camera to build the global topological map. They used structural information mainly to ensure consistency in cases of large odometric error, e.g., when the robot is "kidnapped" and repositioned. Kuipers et al. (2004), Savelli (2005), and Tomatis et al. (2002a,b) use local geometrical knowledge to do structural reasoning and ensure global topological consistency. Modayil et al. (2004) extend the idea to first build a consistent topological map and then extract a global geometrical map based on the recovered topology.

2.6.2 Local submaps

An alternative but related approach is to break up a global geometrical map into a sequence of "local submaps" (e.g. Gutmann and Konolige, 1999; Leonard and Feder, 1999). The difference from the previous technique is that submaps need not be associated with any particular place or distinctive configuration. Local submaps are based on the intuition that estimates of landmarks are locally highly correlated, but less so as the distance between landmarks increases — mainly because a robot's pose uncertainty is small over short portions of its trajectory, but accumulates rapidly over longer portions. (The approach thus applies equally to occupancy grid maps as well.)

Local submaps can be combined to produce globally consistent geometrical maps. Leonard and Newman (2003) register landmarks from nearby submaps using an iterative procedure to recover transformations between local frames of reference. Gutmann and Konolige (1999) apply laser scan-matching techniques to compute the transformations. Atlas (Bosse et al., 2004), GraphSLAM (Thrun et al., 2005), and related techniques (e.g. Folkesson and Christensen, 2004) represent the sequence of local submaps as a graph. Each vertex is a local submap coordinate frame, and edges represent transformations between submaps in the sequence visited by the robot. Subgraph matching techniques and a nonlinear optimization step can be employed to obtain global consistency from the local submaps (see, e.g., Duckett and Saffiotti, 2000; Frese et al., 2005; Olson et al., 2006).

2.7 Feature extraction and place recognition

Up until now we have largely sidestepped the problem of converting raw sensor data, e.g., laser scans, SONAR returns, or infrared measurements, into forms useful for map building. Specifically, landmark mapping techniques require *feature extraction* methods that obtain parameters of geometric primitives from the raw data, and topological mapping approaches must be capable of *distinctive place recognition*. In both cases, sensor limitations such as resolution and range restrictions affect the process. We briefly discuss some common techniques for feature extraction and place recognition and make note of issues due to limited sensing.

2.7.1 Extracting features

We do not address the problem of extracting features from images produced by cameras. Instead, we focus on extracting features from range-bearing data like that produced by scanning laser rangefinders or SONAR and infrared rangefinder arrays. While many types of features can be

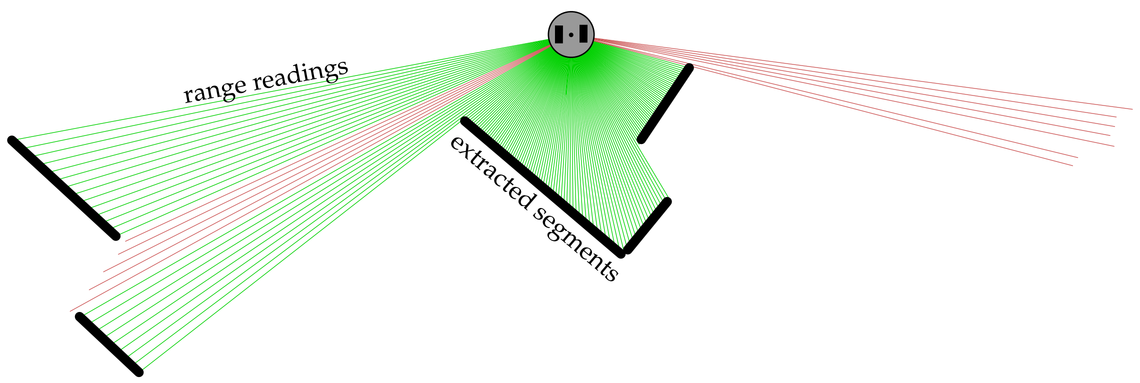


Figure 2.7: Line segments extracted from laser data using IEPF.

used, line and point features are by far the most common and there are a variety of techniques for extracting them from range-bearing data.

2.7.1.1 Extracting lines

A number of researchers have employed line or segment features for SLAM, e.g., Leonard et al. (2002); Tardós et al. (2002); Rodriguez-Losada and Matia (2003); Yuen and MacDonald (2003); Brunskill and Roy (2005). Line extraction typically proceeds in two phases: data segmentation (i.e., clustering), and line estimation. In the segmentation phase, the raw sensor returns — individual range-bearing measurements, e.g., from each sensor in an array or from each orientation of a laser scanner — are partitioned into groups corresponding to each feature. Then, in the estimation phase, lines are fit to the data from each group. Typically, the estimation phase also includes covariance estimation so that landmarks can be represented using EKFs.

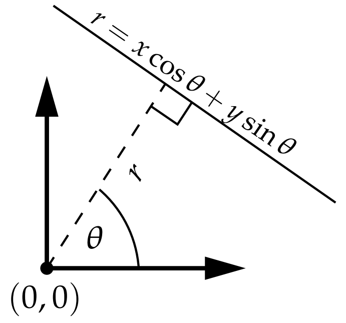
Most segmentation algorithms for range-bearing data assume radially ordered data, like that from a laser rangefinder. A common approach uses an adaptive breakpoint detector (Borges and Aldon, 2004) to form initial clusters and then applies an iterative endpoint filter (IEPF) (see Duda and Hart, 1973) to further split the clusters. This is an attractive option due to its simplicity, both for implementation and computation. Figure 2.7 shows line segments extracted from laser data using the technique. In Chapter 4 we describe a related technique that works with data from multiple robot poses.

Fitting lines to clustered data can simply be done using least squares, although this does not take into account the covariance of individual range-bearing measurements. Pfister et al. (2003) derive a full maximum likelihood line estimator for range-bearing data from a radial scan.

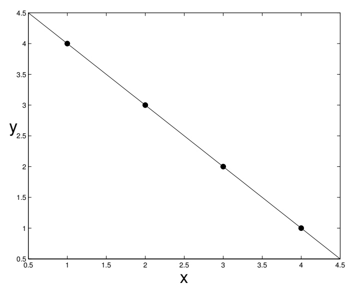
Several researchers have applied the Hough transform, a technique originally used for extracting features from image data (described by Duda and Hart, 1972), to range-bearing data with the goal of extracting line features. Note that a line can be represented by its distance r from the origin and angle θ to the x-axis, as shown in Figure 2.8(a). Furthermore, note that for a point (x, y) in Cartesian space, there are an infinite number of lines through the point. These constitute a sinusoidal curve in the (r, θ) Hough parameter space:

$$r = x \cos\theta + y \sin\theta \tag{2.19}$$

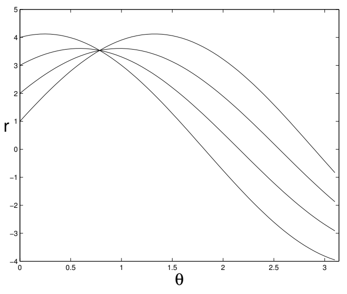
Figure 2.8(c) illustrates the transformation of points in Cartesian space (Figure 2.8(b)) into Hough space. Feature extraction using the Hough transform works by finding intersections of the transformed curves from several data points. Typically this is done by discretizing the Hough space, associating an accumulator with each discrete cell, and accumulating the cells through which a transformed sinusoidal curve passes. Peaks in the accumulation indicate line parameters for the data. The Hough transform has been applied to laser rangefinder data (e.g. Larsson et al., 1996), but has mainly found its success in extracting lines from sparse and noisy SONAR scans (e.g. Yun



(a) (r, θ) line representation



(b) Points on a line



(c) Points in the Hough space

Figure 2.8: The Hough transform. Each curve in the Hough space corresponds to a point in Cartesian space. The intersection of the curves gives the parameters of the line through the points.

et al., 1998; Großman and Poli, 1999; Tardós et al., 2002).

For purposes of data association (see below) and rendering, maps composed of infinite lines are cumbersome. One alternative is to extract line segment features instead. A typical approach is to simply project clustered data points onto the estimated line. Extremal points become segment endpoints. A problem with segment features is their higher dimensionality. Occasionally, line features are used and extent information (obtained via the same projection operation) is kept separately from the state, an approach adopted in the experiments in this thesis.

2.7.1.2 Extracting points

It is sometimes desirable to represent landmarks as points. Most often point features are extracted from camera images, but occasionally corner features are extracted from range-bearing data and represented as points. One approach is to first extract lines, e.g., using IEPF or the Hough transform, and then find intersections of the lines. An alternative is to use the Hough transform directly to extract points using a similar approach to that for extracting lines, as in (Tardós et al., 2002). Finally, in outdoor scenarios, it is common to extract point features corresponding to trees by finding local minima in a range scan (e.g. Guivant and Nebot, 2000).

2.7.2 Recognizing distinctive places

Distinctive place recognition is the topological mapping analogue of feature extraction. In the case of Voronoi-based topological maps, the problem is one of meet point detection. For a robot with an omnidirectional sensor, it is typically straightforward to detect meet points by simply examining minima in the sensor returns and recognizing when more than two minima are equidistant (Choset and Nagatani, 2001; Acar et al., 2001). In practice, it is often necessary to employ some sort of local hill climbing approach to move the robot directly to a meet point, after detecting one in the vicinity. This idea was first applied by Kuipers and Byun (1991), albeit not specifically in the context of Voronoi-based mapping.

More elaborate techniques are necessary for a robot with low-resolution sensing such as an infrared rangefinder array. This is mainly due to the fact that small objects (such as chair legs) may not be observed by the robot until they are quite close. Thus, the robot may “miss” some meet points and leave the GVD. Huang and Beevers (2005a) describe behaviors for a robot with sparse sensing to recognize meet points and return the the GVD when a meet point is missed.

A number of alternative place detection methods exist for non-Voronoi based topological mapping. Beevers (2004) and Huang and Beevers (2005c) describe a topological mapping algorithm for robots with sparse, short-range sensing in which distinctive places are corners, easily detectable as termination conditions for wall- and hall-following behaviors. Kuipers and Byun (1991) recognize junctions of corridors and hill climb to reach midpoints in the junctions. Mozos et al. (2005) apply supervised learning to classify places in the environment into semantic categories. Hybrid mapping approaches often arbitrarily divide the environment into local regions which are not necessarily "distinctive." Another common approach is to have a human operator simply specify when the robot has reached a distinctive place.

2.8 Data association and loop closure

An important component of most mapping algorithms is the ability to match current sensor returns against the map. In geometrical mapping approaches, this is usually referred to as *data association* or the *correspondence problem* — determining, e.g., which features from the current scan correspond to landmarks in the map. In environments with cycles, the additional problem of *closing loops*, or recognizing when the robot has returned to a place it has already been, must also be solved.* Typically the correspondence problem is dealt with only in the geometrical mapping case, but the loop closing problem is universal to all map representations and mapping algorithms.

2.8.1 Finding correspondences

A popular approach for robots equipped with high-resolution sensors is to directly match the raw sensor data with the map, i.e., to do "scan matching." This idea, which is highly related to image registration techniques like iterative closest point (ICP) (Besl and McKay, 1992), was first applied for offline mapping algorithms (Lu and Milios, 1997; Gutmann and Konolige, 1999), but has since found great success in online SLAM algorithms. This is particularly true for approaches that build occupancy grid maps, since laser scan data can be directly compared with individual pixels in the map, as in (Grisetti et al., 2005). In fact, given sufficient data, scan matching can be used to estimate the motion of a robot more accurately than a proprioceptive odometry sensor (see, e.g., Bailey, 2002).

Despite the success of scan matching techniques, most data association methods are based on instead matching the parameters of extracted features to those of already-mapped landmarks. Assuming probabilistic representations of feature and landmark uncertainty are available, the most popular approach is to compute the maximum likelihood correspondence of a feature, i.e., the landmark in the map that most probably matches the feature. For an EKF-based representation of the map, and for a feature $\mathbf{y}$ with covariance $\mathbf{P_y}$, this is computed as:

$$\mathbf{n}_{t,\mathbf{y}} = \underset{\mathbf{x}_i^m \in \mathbf{x}^m}{\operatorname{argmin}} \sqrt{(\mathbf{x}_i^m - \mathbf{y}) \left(\mathbf{H}_h \mathbf{P}^m \mathbf{H}_h^T + \mathbf{P_y}\right)^{-1} (\mathbf{x}_i^m - \mathbf{y})^T} \tag{2.20}$$

i.e., the landmark with the smallest Mahalanobis distance from $\mathbf{y}$ is chosen as the correspondence. Here, $\mathbf{H}_h$ linearizes the measurement function h (see Appendix B for details of the linearization). Typically, if the minimum Mahalanobis distance is above some threshold value, the feature is determined to constitute a new landmark and is added to the map.

In mapping approaches that represent the map using a single large EKF, a single correspondence (if any) must generally be chosen for each extracted feature. Cox and Leonard (1994) describe an approach for instead considering multiple data association hypotheses by decoupling correspondence uncertainty from measurement uncertainty. They maintain a tree of possible correspondences in which each path from the root to a child (i.e., each hypothesis) represents a different assignment of measurements to landmarks.

The approach of Cox and Leonard (1994) is essentially a deterministic version of the "randomized" approach taken by particle filtering SLAM algorithms, e.g., (Montemerlo, 2003), for finding correspondences. Particle filters estimate different maps for each particle, and thus do correspondence finding on a per-particle basis. As such, each particle constitutes a different data

*While loop closing is really a special case of data association, it is generally treated as a separate problem because pose uncertainty with respect to parts of the environment mapped long in the past is much higher than with respect to the robot's immediate surroundings.

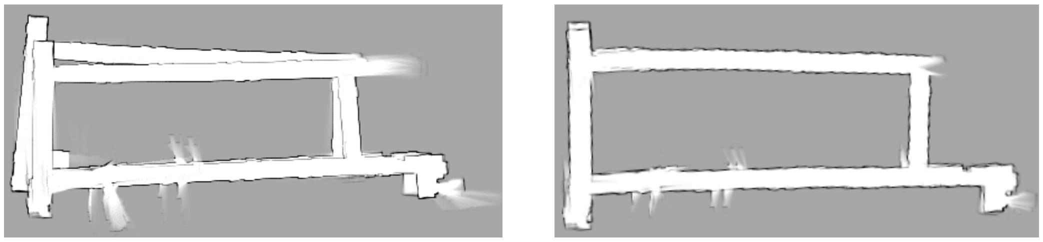

(a) Map produced from uncorrected odometry (b) Map produced with cycle detection

Figure 2.9: The loop closing problem. Without cycle detection maps are subject to significant perceptual aliasing. Data set courtesy of Andrew Howard via Radish (Howard and Roy, 2003).

association hypothesis, and resampling of particles prunes maps with incorrect data associations. Montemerlo (2003) uses maximum likelihood data association on a per-particle basis, and also describes an approach for "data association sampling," i.e., choosing a correspondence with probability proportional to its likelihood.

2.8.2 Detecting cycles

Loop closing is one of the most difficult problems in mapping. Figure 2.9 illustrates the underlying issue: mapping algorithms that rely only on a robot's uncorrected and noisy odometry are subject to significant "perceptual aliasing" (or alignment) problems because the pose error of most robots accumulates rapidly over trajectories of more than a few meters. Early mapping approaches based on EKFs ignored the cycle detection problem (e.g. Smith et al., 1990), but it has become a significant focus of recent research.

The simplest approach for closing loops is to modify the environment, e.g., by leaving markers at junctions to recognize when a loop has been closed. For example, Bender et al. (1998) show that in an environment modeled as a graph, a single "pebble" suffices for a robot to map the environment as long as an upper bound on the number of vertices in the graph is known. However, in most practical scenarios modifying the environment is not an option.

Robots with high-fidelity sensors can take snapshots or "sensing signatures" of places in the environment and use these for closing loops. Kuipers and Beeson (2002) describe a supervised learning approach for recognizing distinctive places in topological maps based on their sensing signature. While this approach is reasonable for robots with scanning laser rangefinders or cameras in environments that do not exhibit too much self-similarity, it is not applicable for robots with more limited sensing.

Other work on loop closing in topological maps has focused largely on structural matching techniques. In environments with structural asymmetries, a robot can simply traverse its map to a differentiating place in order to determine the correct loop closing hypothesis (e.g. Kuipers and Byun, 1991; Choset and Nagatani, 2001). Rather than exhaustively eliminate all hypotheses, others have taken an evidential approach in which geometrical information is employed to determine the correct hypothesis (e.g. Tomatis et al., 2002a,b; Beevers and Huang, 2005). In some cases, prior knowledge, such as planarity constraints on embeddable topological maps, can be employed to aid in loop closing, as in (Savelli and Kuipers, 2004; Savelli, 2005). Related structural matching techniques have also been successfully applied in hybrid mapping approaches (e.g. Bosse et al., 2004; Thrun et al., 2005).

Geometrical mapping algorithms frequently rely on data association to detect cycles. For particle filtering algorithms or other approaches that maintain several data association hypotheses, this method can be sufficient in reasonably complicated environments (Montemerlo, 2003). Others have examined approaches for actively closing loops. For example, Stachniss et al. (2005a) describe an exploration algorithm for particle filter mapping based on POMDP-like decision making in which the robot actively seeks to verify loop-closing hypotheses.

2.9 Other considerations

There are many other aspects of the robot mapping problem that we have not discussed above. Here, we briefly touch upon several more considerations for designing mapping algorithms.

2.9.1 Planar and 3D mapping

Maps that include geometrical information are generally embedded in $\mathbb{R}^2$ or $\mathbb{R}^3$. The vast majority of robot mapping research has focused on the 2D case since it is suitable for navigation in many interesting circumstances, e.g., indoors or in structured outdoor environments. More recently, as robots begin to venture into less structured conditions, maps that incorporate full 3D information and six degrees of (robot) freedom have been investigated.

This thesis focuses on building planar maps. We note that sensing that is "high-fidelity" in the planar case may be much less so in the 6-DOF case. For example, a scanning laser rangefinder that provides dense coverage of a 2D "slice" of the world can be used to build very accurate planar maps, but must be augmented in the 3D case, by moving or re-orienting the sensor, using several sensors, or both. Similarly, a simple 1D sensor such as an infrared rangefinder can be used to build accurate maps in a (hypothetical) 1D environment but is less sufficient in a planar environment. In this sense, by studying the problem of 2D mapping with limited sensing we can develop techniques that are useful in the 3D case as well.

2.9.2 Online and offline mapping

Robot mapping algorithms can be divided into techniques that can be applied *online* and those that are executed *offline*. Online approaches, e.g., (Smith et al., 1990; Kuipers and Byun, 1991; Choset and Nagatani, 2001; Montemerlo, 2003), construct a map sequentially as data arrives. Offline approaches, e.g., (Lu and Milios, 1997; Thrun et al., 1998b), compute maps from all the data collected by the robot over the course of its deployment and need not be sequential in nature. Offline methods are usually subject to fewer computational restrictions since online algorithms must typically meet realtime requirements.

Generally, online mapping algorithms that enable a robot to use its map for planning, navigation, and localization while the map is being built are the most practically useful techniques. This thesis is focused on developing online algorithms. However, algorithms primarily developed for offline mapping can play roles in online scenarios, e.g., as post-processing steps when extra computation time is available. Furthermore, the notion of using as much available information as possible, typical of offline mapping, can also be applied to online algorithms and is the main principle behind the new algorithms presented in Chapter 6. In these algorithms, new information, as it arrives, is used to improve past pose and map estimates.

2.9.3 Passive and active mapping

An advantage of online mapping approaches is that, when feasible, the current map can be used to direct the exploration of the robot, e.g., to reduce uncertainty in the map. Mapping algorithms that direct the robot's exploration are known as *active* techniques, as opposed to *passive* approaches that simply build the map opportunistically from sensing inputs obtained as the robot goes about some other task.

Most robot mapping algorithms are passive. Completely passive approaches can be disadvantageous if the sensing capabilities of the robot are restricted, since certain actions, such as spinning rapidly in place, lead to large uncertainty if a robot's odometry is subject to significant noise and exteroceptive sensing is limited. Commands that direct a robot to cross a large, empty room deprive the robot of sensing feedback when its sensors have limited range (Roy and Thrun, 1999). In general, there are classes of actions that are are inappropriate for creating accurate maps. Thus, approaches that automatically consider which actions will lead to accurate maps are often desirable (see, e.g., Shatkay and Kaelbling, 2002; Stachniss et al., 2005a).

However, there are certainly many situations that call for the application of passive mapping techniques. A robotic car should not switch lanes on a busy highway just to improve its knowledge of the surrounding vehicles. Nor should a robot with a specific goal (beyond creating a map) direct more effort to mapping than necessary.

The development of successful passive mapping approaches is also compelling because the same techniques remain applicable in an active mapping context. In fact, the main problem addressed by active mapping algorithms is one of exploration, i.e., how to efficiently direct the robot so a given passive algorithm generates an accurate map. This thesis focuses on passive mapping algorithms and does not address the exploration issue, although we note that some of the techniques described earlier for topological mapping, such as tracing the SGVD$_\infty$ of the environment, may be useful in this respect.

2.9.4 Dynamic environments

Robots in the real world must deal with evolving scenarios: cars drive at different speeds down a busy street, doors open and close, people crowd and jostle a robot in their curiosity. Regardless of a robot's sensing capabilities, observing dynamic components of the environment reliably can be extremely difficult. For this reason, most current robot mapping work assumes the workspace of the robot is static. Thus, the environment model $\mathbf{x}^m$ is not time-dependent so, as noted earlier, the robot's observation model (2.5) can be rewritten as:

$$\mathbf{z}_t = h(\mathbf{x}^r_t, \mathbf{x}^m) + \mathbf{w}_t \tag{2.21}$$

i.e., measurements depend only upon the current configuration of the robot with respect to the environment, *as long as the system model is otherwise complete*. This is sometimes known as the Markov (or "complete state") assumption.

In practice it is impossible to model all aspects of the environment and the robot that affect sensor measurements. Thrun et al. (2005) list several potential violations of the Markov assumption, including:

- dynamic elements in the environment, e.g., people;
- inaccuracies in the motion and measurement models (Equations 2.7 and 2.8, respectively); and
- approximation errors due to uncertainty representations such as grids or samples.

While some work has been done on recognizing dynamic components of the environment (e.g. Fox et al., 1999; Biber and Duckett, 2005), this remains a challenging problem even with high-fidelity sensors and it is not addressed in this thesis. However, as noted by Thrun et al. (2005) and others, probabilistic mapping algorithms such as particle filtering approaches have shown themselves to be reasonably robust to violations of the Markov assumption in practical scenarios.

2.9.5 Multi-robot mapping

When multiple robots exist in the same workspace and can communicate, it is sometimes reasonable for the robots to work together to build maps. The robots can share data to combine partial maps, distribute computation and storage, and even serve as "cooperative landmarks" to aid in mapping. Many researchers have studied the multi-robot mapping problem (e.g. Dudek et al., 1998; Dedeoglu and Sukhatme, 2000; Thrun, 2001; Fenwick et al., 2002; Ko et al., 2003; Konolige et al., 2003; Huang and Beevers, 2005b). While the focus of this thesis is on single-robot mapping, many of the techniques described here can be used in a multi-robot context by applying algorithms for merging the maps of individual robots.

3

THE INFLUENCE OF SENSING CAPABILITIES ON MAP QUALITY

Most research on robot mapping focuses on developing algorithms for mapping with particular sensors, such as laser rangefinders, stereo cameras, or SONAR arrays. Parts of this thesis, for example, examine the case of mapping with a small infrared rangefinder array. However, in this first technical chapter we instead take a broad approach and apply a single simple mapping algorithm to a large class of sensors, with the goal of uncovering basic relationships between different sensor capabilities and the quality of maps built by the sensors.

In order to perform such a theoretical analysis, we make some simplifying assumptions about the robot's trajectory, correlation between observations, and structure in the environment. We first describe these assumptions, and then detail a sensor model which encapsulates most of the interesting aspects of range-bearing sensors used for mapping, including sensing radius, beam width, range and bearing noise, and so on. We then apply the model to occupancy grid mapping and prove a bound on the expected error in the map, in terms of the sensor characteristics and trajectory length. In doing so, we also obtain a bound on the expected number of observations of each cell and conditions on the sensor for convergence to the correct map. The model is implemented in simulation and we show that our bounds match the empirical results. In addition, we apply the model using parameters from several real-world sensors, enabling us to quantify the relative usefulness of the sensors for mapping.

3.1 Overview

The vast majority of robot mapping research has the goal of developing algorithms to build maps with *specific* robots. For example, much of the modern research on simultaneous localization and mapping (SLAM) assumes the availability of a scanning laser rangefinder. (A decade ago, on the other hand, most SLAM research used arrays of SONAR sensors.) There are specific algorithms tailored to mapping with lasers (Grisetti et al., 2005), infrared rangefinder arrays (Beevers and Huang, 2006a; this thesis, Chapter 4), SONAR arrays (Leonard et al., 2002; Tardós et al., 2002), monocular cameras (Deans and Hebert, 2000a; Solá et al., 2005), radio beacons (Detweiler et al.,

m_{ij}	map grid cell (row i, col. j)	ρ	readings per scan	F	sensor firing frequency
M	num. map rows and columns	r^+	maximum sensing radius (m)	ε_E	false negative rate
δ	length (m) of a grid cell side	$\hat{r}$	reported range (m)	ε_F	false positive rate
d	cell occupancy probability	Δ_β	beam width (rad)	o_{ij}	num. observations of cell m_{ij}
$\mathbf{x}_t^r$	pose at time $t \in [0, T]$	σ_β	bearing uncertainty (rad)	ν	error in max. likelihood map
		σ_r	range uncertainty (m)		

Notation for this chapter

2006), and so on.

One shortcoming of the sensor-specific approach to mapping is that when a different algorithm is designed for every sensor, it is difficult to say anything general about the sensing requirements for mapping. Recent work by O'Kane and LaValle on localization with limited sensing (O'Kane and LaValle, 2006b,a) has introduced several important concepts — robot primitives, the notion of "dominance" of one robot over another with respect to a task, and equivalence classes of robots for a given task — which enable binary characterization of a robot's ability to perform a task. They have used these tools to show, for example, that a robot with a compass and a contact sensor can localize itself in a polygonal environment, but a robot with an angular odometer and a contact sensor cannot. However, their approach still requires the development of specialized algorithms for each equivalence class.

In this work we take a different approach: we employ a single occupancy mapping algorithm for a broad sensor model that encompasses most types of sensors used for mapping. This enables us to characterize which sensors are able to build a map, and *also* to provide quality bounds on the map for a particular sensor. Our bounds are in terms of the parameters of the sensor model, which enables us, for example, to verify if the map made by a given sensor converges to the correct one in the limit; if it converges, we can additionally lower-bound the rate of convergence.

We employ a simple occupancy model of the environment and build an occupancy grid map using a bounded-uncertainty model of a range-bearing sensor. Our model makes several simplifying assumptions, which are stated in Section 3.2. The model, described in detail in Section 3.3, incorporates several sensor characteristics: maximum sensing radius, number of readings per scan, beam width, firing frequency, range and bearing noise, and false positives and negatives. We first prove a lower bound in Section 3.4 on the expected number of times a cell in the map is updated. We then use this bound in Section 3.5 to give conditions on the sensor parameters for convergence to the correct map; we also derive an upper bound on the error in the map in terms of the sensor and the trajectory length.

To empirically validate our results, we implemented our sensor model and mapping algorithm in simulation. Section 3.7 examines the effects of each of the model parameters for a variety of different "base" sensors, including a SONAR-like sensor, a laser-like sensor, and bearing- and range-only sensors. We also empirically address some of the simplifications employed in deriving the analytical bounds, and point out interesting cases for future study. Finally, we describe an analytical comparison, using our bounds, of the mapping capabilities of real sensors including infrared arrays, SONAR arrays, and a scanning laser rangefinder.

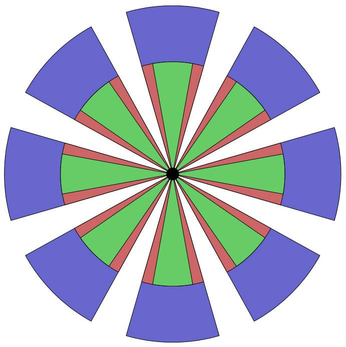
3.2 Assumptions

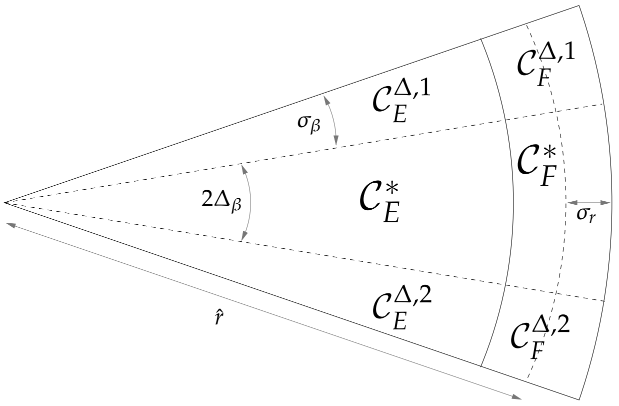
To obtain the theoretical results described in this chapter, we made several assumptions about the robot, the sensors, and the environment. This section briefly states the assumptions, which are discussed further in the following sections.

We assume the observations are taken from a sequence of poses drawn independently and uniformly at random from within the environment. This is in contrast to a more realistic continuous trajectory. In Section 3.7 we show in simulation that the analytical results obtained under this assumption still match well with the case of continuous trajectories.

We also assume that false negatives and false positives — i.e., errors in readings of a cell in the environment — occur independently for each reading. In reality, situations may instead arise where repeatedly observing the same cell or environment feature consistently yields incorrect readings. Potentially, our model could be extended to include such effects by assigning false negative and positive rates on a per-cell basis.

Finally, our analysis assumes that the environment is unstructured, i.e., that no correlation exists between the occupancies of different cells in our occupancy model. Section 3.7.1 discusses this assumption in more detail, shows the effect of structure on map quality using simulations, and describes a possible extension to our approach to incorporate structure by using a Markov


(a) Sensor arrangement.



$\mathcal{C}_E^{\Delta,1}$
$\mathcal{C}_F^{\Delta,1}$
σ_β
$2\Delta_\beta$
$\mathcal{C}_E^*$
$\mathcal{C}_F^*$
σ_r
$\hat{r}$
$\mathcal{C}_E^{\Delta,2}$
$\mathcal{C}_F^{\Delta,2}$

(b) Beam model

Figure 3.1: Illustration of the sensor model. In (a), the regions $\mathcal{C}_{\mathrm{F}}$, $\mathcal{C}^*_{\mathrm{E}}$, and $\mathcal{C}^\Delta_{\mathrm{E}}$ are depicted in different shades. Beams are arranged radially around the robot.

random field (MRF) model of the environment.

3.3 Model

We begin by introducing the environment, sensor, and map update models which we use for the analysis in Sections 3.4–3.5.

3.3.1 Environment model

We consider an environment $\mathcal{M}$ consisting of an $M \times M$ grid of cells m_{ij}, each of size $\delta \times \delta$. Cells are either full (F) with *a priori* probability d or empty (E) with probability $1 - d$. In our analysis we assume for simplicity that the environment is toroidal, i.e., a boundary is connected to its opposite side. We also assume that the occupancy of each cell is independent, although we address the case of structured environments empirically in Section 3.7.

3.3.2 Robot and sensor model

In our analysis, the robot's trajectory $\mathbf{x}_t^r = [x_t^r \; y_t^r \; \theta_t^r]^T$, $t \in [0, T]$ consists of a sequence of randomly selected poses in $\mathcal{M} \times \mathbb{S}^1$. We empirically examine the more realistic case of trajectories with velocity and turning constraints, and show that our simplifying assumption yields very similar results.

The robot has a range-bearing sensor with maximum range r^+ which fires at frequency F. The sensor takes ρ readings per scan (i.e., it fires ρ "beams"). The beams are placed at angles $\{\beta_1, \ldots, \beta_\rho\}$ where $\beta_i = i\frac{2\pi}{\rho} + U[-\sigma_\beta, \sigma_\beta]$, i.e., the beams are approximately evenly spaced around the perimeter of the robot but there is some bearing error, distributed uniformly over an interval of range $2\sigma_\beta$. Each beam has beam width $2\Delta_\beta$. The beam continues outward from the robot until it *detects* an occupied cell. For a particular empty cell observed by the beam, a false positive is registered with likelihood ε_{F}. A false negative occurs for an occupied cell with likelihood ε_{E}. The sensor returns the range $\hat{r}$ to the closest cell $m_{\min}$ which is detected as occupied. The range is corrupted by uniformly distributed noise, i.e., if the robot is in cell m_{ij}, $\hat{r} = ||m_{\min} - m_{ij}|| + U[-\sigma_r, \sigma_r]$.

In our analysis of mapping, we focus on the sensor characteristics described above and as such, we assume the trajectory of the robot is perfectly known. In Section 3.6 we discuss extensions to the model which may enable analysis in the SLAM case.

3.3.3 Mapping

The robot builds an occupancy grid map. Updates are applied to a cell when it intersects the sensor beam. The map updates employ a bounded uncertainty model: the beam width is inflated from $2\Delta_\beta$ to $2(\Delta_\beta + \sigma_\beta)$ to account for bearing error, and the beam range is inflated from $\hat{r}$ to $\hat{r} + \sigma_r$ to account for range error. The beam is divided into regions as shown in Figure 3.1; we further define $\mathcal{C}_{\mathtt{E}}^{\Delta} = \mathcal{C}_{\mathtt{E}}^{\Delta,1} \cup \mathcal{C}_{\mathtt{E}}^{\Delta,2}$ and $\mathcal{C}_{\mathtt{E}} = \mathcal{C}_{\mathtt{E}}^{*} \cup \mathcal{C}_{\mathtt{E}}^{\Delta}$; define $\mathcal{C}_{\mathtt{F}}^{\Delta}$ and $\mathcal{C}_{\mathtt{F}}$ similarly; and define $\mathcal{C} = \mathcal{C}_{\mathtt{E}} \cup \mathcal{C}_{\mathtt{F}}$.

Based on the returned range $\hat{r}$ of a beam, we determine a measurement $z_{ij} \in \{\mathtt{E}, \mathtt{F}\}$ for each cell m_{ij} contained within the beam. If $m_{ij} \in \mathcal{C}_{\mathtt{F}}$ the measurement is $z_{ij} = \mathtt{F}$; if $m_{ij} \in \mathcal{C}_{\mathtt{E}}$ it is $z_{ij} = \mathtt{E}$. We use a simple additive update model. If $z_{ij} = \mathtt{F}$, the *belief* $\hat{p}(m_{ij} = \mathtt{F})$ in the occupancy of a cell is incremented by p_0, i.e., $\hat{p}(m_{ij}{=}\mathtt{F} | z_{ij}{=}\mathtt{F}) = \hat{p}(m_{ij}{=}\mathtt{F}) + p_0$; similarly, if $z_{ij} = \mathtt{E}$, the belief is decremented by the same p_0. This update model is simplistic and in practice one might instead use Bayesian or Dempster-Shafer updates (Pagac et al., 1998) along with a more complicated sensor model (Konolige, 1997; Thrun, 2003). However, our focus is on analysis and the additive approach is both convenient and effective in practice.

3.4 Bound on cell updates

In this section we prove a lower bound on the expected number of times an update is applied to each cell in the grid. In the next section we use this result to establish a bound on the expected error in the map.

For convenience, we define $\mathcal{E}_{\mathtt{E}} = ((1-d)(1-\varepsilon_{\mathtt{E}}) + d\varepsilon_{\mathtt{F}})$ and $\mathcal{E}_{\mathtt{F}} = (d(1-\varepsilon_{\mathtt{F}}) + (1-d)\varepsilon_{\mathtt{E}})$ as the probabilities that some cell in a beam registers as empty or full, respectively.

Theorem 3.1. *In T time steps, for a sensor as described in Section 3.3, the expected number of updates of any cell $m_{ab} \in \mathcal{M}$ is bounded by:*

$$E[o_{ab}] \geq \frac{2TF\rho(\Delta_\beta + \sigma_\beta)}{M^2} \sum_{\tau=0}^{\left\lceil \frac{r^+ + \sigma_r}{\delta} \right\rceil} \tau \cdot p_{\text{obs}} \tag{3.1}$$

where:

$$p_{\text{obs}} \geq \mathcal{E}_{\mathtt{E}}^{\Delta_\beta \tau^2} \tag{3.2}$$

Proof. Let n_{ij} be the number of times the sensor is fired when the robot is in cell m_{ij}. Furthermore, define $\text{obs}(m_{ab}, k)$ as the event that m_{ab} is *observed* by beam k, $1 \leq k \leq \rho$, and define $B(m_{ab}, k)$ as the event that m_{ab} intersects the maximum-range beam k. Then:

$$E[o_{ab}] = \sum_{i,j} E[n_{ij}] \sum_{k=1}^{\rho} \underbrace{p(\text{obs}(m_{ab}, k) | B(m_{ab}, k), \mathbf{x}_t^r \in m_{ij})}_{p_{\text{obs}}} \times \underbrace{p(B(m_{ab}, k) | \mathbf{x}_t^r \in m_{ij})}_{p_{\text{beam}}} \tag{3.3}$$

Under our assumption of uniformly distributed poses and since the beam angles are all uniformly distributed with the same variance, the terms in the internal summation are identical for all beams. Thus:

$$E[o_{ab}] = \frac{TF\rho}{M^2} \sum_{i,j} p_{\text{obs}} \cdot p_{\text{beam}} \tag{3.4}$$

We now turn to p_{beam}. Note that for m_{ab} such that $||m_{ab} - m_{ij}|| > r^+ + \sigma_r$, this likelihood is zero. Define $\mathcal{N}(m_{ab}) = \{m_{ij} : ||m_{ab} - m_{ij}|| \leq r^+ + \sigma_r\}$. We ignore cells not in $\mathcal{N}(m_{ab})$. Now it is easily seen that the likelihood that m_{ab} intersects beam k is simply the ratio of cells in $\mathcal{N}(m_{ab})$ intersecting a beam of width $\Delta_\beta + \sigma_\beta$ and range $r^+ + \sigma_r$ to the total number of cells in $\mathcal{N}(m_{ab})$, i.e.:

$$p(B(m_{ab},k)|\mathbf{x}_t^r \in m_{ij}) = \Omega\left(\frac{\Delta_\beta + \sigma_\beta}{\pi}\right) \tag{3.5}$$

With this we can rewrite (3.4):

$$E[o_{ab}] \geq \frac{TF\rho(\Delta_\beta + \sigma_\beta)}{\pi M^2} \sum_{m_{ij}\in\mathcal{N}(m_{ab})} p_{\text{obs}} \tag{3.6}$$

Reparameterizing the summation by the integral range τ to the cells yields:

$$\geq \frac{TF\rho(\Delta_\beta + \sigma_\beta)}{\pi M^2} \sum_{\tau=0}^{\left\lceil \frac{r^+ + \sigma_r}{\delta} \right\rceil} \left\lceil \frac{2\pi\tau\delta}{\delta} \right\rceil p_{\text{obs}} \tag{3.7}$$

where $\frac{\lceil 2\pi\tau\delta \rceil}{\delta}$ is the number of cells at integral range τ from m_{ab}. Simplifying:

$$\geq \frac{2TF\rho(\Delta_\beta + \sigma_\beta)}{M^2} \sum_{\tau=0}^{\left\lceil \frac{r^+ + \sigma_r}{\delta} \right\rceil} \tau \cdot p_{\text{obs}} \tag{3.8}$$

Only p_{obs}, the probability that we actually *update* our estimate of the occupancy of m_{ab}, remains. The cell will not be updated only if it is occluded by another cell detected as occupied, closer to the robot. Let $\tau\delta$ be the distance $||m_{ab} - m_{ij}||$. Then m_{ab} is observed only if (a) no cell m_{qr} such that $||m_{qr} - m_{ij}|| < \tau\delta - 2\sigma_r$ is detected as being occupied, *and* for all cells such that $\tau\delta - 2\sigma_r \leq ||m_{qr} - m_{ij}|| < \tau\delta$, either (b) none of the cells is detected as occupied; or (c) a cell is detected as occupied but the (noisy) reported range is less than $\tau\delta - \sigma_r$. In all other cases, *some* update will be applied to cell m_{ab}. We focus only on (a) and (b) since our objective is simply to bound the number of observations.

Define p_a and p_b as the probabilities of events (a) and (b), respectively, and define $W_{\text{c}} = \{m_{qr} : B(m_{qr}, k) \wedge ||m_{qr} - m_{ij}|| < \tau\delta - 2\sigma_r\}$ and $W_{\text{n}} = \{m_{qr} : B(m_{qr}, k) \wedge \tau\delta - 2\sigma_r \leq ||m_{qr} - m_{ij}|| \leq \tau\delta\}$. It is straightforward to see that:

$$|W_{\text{c}}| = O\left(\frac{\Delta_\beta}{\delta^2} \max\{0, \tau\delta - 2\sigma_r\}^2\right) \tag{3.9}$$

and:

$$|W_{\text{n}}| = O\left(\frac{\Delta_\beta}{\delta^2}\left((\tau\delta)^2 - \max\{0, \tau\delta - 2\sigma_r\}^2\right)\right) \tag{3.10}$$

The probability that a cell $m_{qr} \in W_{\text{c}}$ is detected as *unoccupied* is just $\mathcal{E}_{\text{E}}$, so $p_a \geq \mathcal{E}_{\text{E}}^{\frac{\Delta_\beta}{\delta^2}(\tau\delta - 2\sigma_r)^2}$ if $\tau\delta > 2\sigma_r$, and $p_a = 1$ otherwise. Similarly, the probability that all cells in W_{n} are detected as unoccupied is $p_b \geq \mathcal{E}_{\text{E}}^{4\frac{\Delta_\beta}{\delta^2}(\tau\delta\sigma_r - \sigma_r^2)}$ if $\tau\delta > 2\sigma_r$, and $p_b = \mathcal{E}_{\text{E}}^{\Delta_\beta \tau^2}$ otherwise. Multiplying p_a and p_b yields the lower bound on p_{obs} stated in (3.2). We omit a derivation here but it is also straightforward to obtain a bound on p_c. Specifically, for the case $\tau\delta > 2\sigma_r$:

$$p_c \geq \frac{1}{2} \min\left\{1, \frac{4\mathcal{E}_{\text{F}}\Delta_\beta}{\delta^2}(\tau\delta\sigma_r - \sigma_r^2)\right\} \tag{3.11}$$

In practice we have found that p_c generally has little effect on $E[o_{ab}]$ so we do not include it in the main bound.

Combining the above results yields the bound. □

3.4.1 Discussion

The key result we have shown so far is that some sensing parameters have diminishing returns. For example, the improvements obtained by increasing sensing radius or beam width decay exponentially after a certain point (as seen in Equation 3.2). On the other hand, increasing the number of readings per scan always increases the number of updates, as does increasing the firing frequency. Of course, increased sensor noise also causes us to make more updates, but these updates will frequently be incorrect. We therefore now turn to a more definitive metric, the error in the resulting map, and show how it is related to the various sensor parameters.

3.5 Bounds on map error

In this section we prove an upper bound on the expected error in the maximum likelihood map estimate and give conditions on the sensor for convergence to the correct map. The maximum likelihood map estimate for cell m_{ij} is defined as $m_{ij}^{\max} = \text{argmax}_{s\in\{\texttt{E},\texttt{F}\}}\, \hat{p}(m_{ij} = s)$. The error ν_{ij} for the cell is 0 if $m_{ij}^{\max} = m_{ij}$ and 1 otherwise, and the error for the entire map is $\nu = \sum_{i,j} \nu_{ij}$. We begin by characterizing the probability that a particular update of a cell is incorrect.

Lemma 3.2. *If cell m_{ij} is* unoccupied, *the probability $p(inc|m_{ij} = \texttt{E})$ that an update to m_{ij} is incorrect (i.e., $\hat{p}(m_{ij} = \texttt{F})$ is increased) is at most:*

$$\sum_{\tau=0}^{\left\lceil \frac{r^+ + \sigma_r}{\delta} \right\rceil} p_{\text{obs}} \cdot p_{\text{f}} \cdot \frac{(\tau\delta + \sigma_r)^2 - \max\{0, \tau\delta - \sigma_r\}^2}{(\tau\delta + \sigma_r)^2} \tag{3.12}$$

where:

$$p_{\text{f}} = \min\left\{1, \frac{\Delta_\beta \mathcal{E}_{\texttt{F}}}{\delta^2}\left((\tau\delta + \sigma_r)^2 - \max\{0, \tau\delta - \sigma_r\}^2\right)\right\} \tag{3.13}$$

Proof. An update to an unoccupied cell is incorrect if the cell falls in $\mathcal{C}_{\texttt{F}}$ in some sensor beam. It is easily seen that:

$$p(\text{inc}|m_{ij} = \texttt{E}) \le \sum_{\tau=0}^{\left\lceil \frac{r^+ + \sigma_r}{\delta} \right\rceil} p\left(\lceil \tau\delta - \delta \rceil \le \hat{r} \le \lceil \tau\delta + \delta \rceil\right) \frac{|\mathcal{C}_{\texttt{F}}^{\tau\delta}|}{|\mathcal{C}^{\tau\delta}|} \tag{3.14}$$

where the first term in the summation is the probability of obtaining a set of particular range measurements, and $|\mathcal{C}^{\tau\delta}|$ indicates the number of cells in the beam of range $\tau\delta$. A simple bound can be obtained based on p_{obs} from the preceding section:

$$p(\text{inc}|m_{ij} = \texttt{E}) \le \sum_{\tau=0}^{\left\lceil \frac{r^+ + \sigma_r}{\delta} \right\rceil} p_{\text{obs}} |\mathcal{C}_{\texttt{F}}^{*,\tau\delta}| \mathcal{E}_{\texttt{F}} \frac{|\mathcal{C}_{\texttt{F}}^{\tau\delta}|}{|\mathcal{C}^{\tau\delta}|} \tag{3.15}$$

Setting $p_{\text{f}} = |\mathcal{C}_{\texttt{F}}^{*,\tau\delta}|\mathcal{E}_{\texttt{F}}$ we obtain the bound. □

Lemma 3.3. *If m_{ij} is* occupied*:*

$$p(inc|m_{ij} = \texttt{F}) \le \sum_{\tau=0}^{\left\lceil \frac{r^+ + \sigma_r}{\delta} \right\rceil} p_{\text{obs}} \cdot p_{\text{f}} \cdot \frac{\max\{0, \tau\delta - \sigma_r\}^2}{(\tau\delta + \sigma_r)^2} \tag{3.16}$$

Proof. The proof is identical to that of Lemma 3.2, replacing $\frac{|\mathcal{C}_{\texttt{F}}^{\tau\delta}|}{|\mathcal{C}^{\tau\delta}|}$ with $\frac{|\mathcal{C}_{\texttt{E}}^{\tau\delta}|}{|\mathcal{C}^{\tau\delta}|}$. □

Theorem 3.4 (Convergence conditions)**.** *Let* $p_{\text{inc}} = d \cdot p(inc|m_{ij}{=}\texttt{F}) + (1-d) \cdot p(inc|m_{ij}{=}\texttt{E})$*. If* $p_{\text{inc}} > \frac{1}{2}$ *the maximum likelihood map error diverges.*

Proof. Note that p_{inc} is the probability that an update of an arbitrary cell is incorrect. If $p_{\text{inc}} > \frac{1}{2}$, the frequency of incorrect updates outnumbers that of correct updates. Since we use an additive approach to update the map, the error in the maximum likelihood estimate only increases with more measurements. □

Theorem 3.5 (Error bound)**.** *If* $p_{\text{inc}} \leq \frac{1}{2}$*, the expected error* ν *of the maximum likelihood map is bounded from above by:*

$$E[\nu] \leq M^2 \exp\left\{-2E[o_{ab}]\left(\frac{1}{2} - p_{\text{inc}}\right)^2\right\} \tag{3.17}$$

Proof. The result is a direct application of the Chernoff bound. From Theorem 3.1 we have an expression for the expected number of times an update is applied to a cell in the map. Lemmas 3.2 and 3.3 are combined to obtain p_{inc}, the probability that any particular update is incorrect. The Chernoff bound states that likelihood of error in the maximum likelihood estimate of a cell decreases exponentially in the number of updates, at a rate inversely proportional to the probability of an incorrect update. The total error is over M^2 cells, yielding the bound. □

Corollary 3.6. *If* $p_{\text{inc}} > \frac{1}{2}$ *the expected error is bounded from below by:*

$$E[\nu] \geq M^2 \cdot \left(1 - \exp\left\{-2E[o_{ab}]\left(p_{\text{inc}} - \frac{1}{2}\right)^2\right\}\right) \tag{3.18}$$

3.5.1 Discussion

An interesting consequence of Theorem 3.4 is that it reveals one way in which our mapping approach may be unrealistic. In particular, consider a bearing-only sensor, such as a monocular camera, for which $\sigma_r \approx r^+$ since a camera cannot detect depth. According to Theorem 3.4, the map diverges for such a sensor — i.e., the error in the maximum likelihood map increases with more observations. Nevertheless, a number of researchers have demonstrated mapping with cameras. The difference between the occupancy-based technique we employ and some practical mapping algorithms is that the occupancy approach does not rely on (or make use of) correspondence information, e.g., a "landmark id" assigned to a particular measurement. Our results suggest that without such correspondence information, or some other source of knowledge, it is impossible for bearing-only sensors to build a correct map. This is similarly the case for range-only sensors in certain environments, since p_{f} is proportional to the beam width Δ_β.

3.6 Extensions

While we have focused on a particular scenario — occupancy mapping with random poses in unstructured environments — the approach we have taken has implications for a variety of scenarios and can be extended in several ways.

Feature-based mapping, with known or unknown correspondences, can be analyzed in our framework by maintaining separate grids for each landmark ID, updating a grid only when the associated landmark is observed (and with unknown correspondences, scaling the update magnitudes by a correspondence likelihood). We additionally note that Theorem 3.1 has direct implications for feature-based mapping. Specifically, for a given "minimum feature size," the bound on $E[o_{ab}]$ is also a bound on the expected number of single-beam observations of each feature in the environment. Coupled with an analysis similar to that in Section 3.5 (specific to the type of features being estimated), a bound on feature-based map error can be obtained in terms of the robot's sensing capabilities.

	Explanation	SONAR	Laser	Bearing	Range
ρ	readings/scan	16	180	60	1
r^+	max. range	10	10	10	10
Δ_β	beam width	17°	0.3°	0.3°	360°
σ_β	bearing unc.	0.06°	0.06°	0.3°	0°
σ_r	range unc.	0.1	0.02	8	0.1
ε_{E}	false neg.	0.01	0.001	0.001	0.01
ε_{F}	false pos.	0.02	0.001	0.01	0
d	env. density	0.01	0.01	0.001	0.001

Table 3.1: Synthetic sensor parameters

Empirical evidence presented in the following section suggests that, at least in unstructured environments, our assumption of random poses may not be a bad one since adding correlation between mapping poses changes performance only slightly in most cases. (This is less the case in structured environments where the "view" from one instant to the next changes only slightly.) A first step toward analyzing constrained trajectories would be to incorporate a simple random walk process model in the framework.

Finally, pose uncertainty can be included in our model by maintaining a grid-based representation of the uncertainty and convolving the grid with map updates. Our conditions for convergence do not apply in this "SLAM"-like case, but we believe a similar approach may yield useful results.

3.7 Experimental results

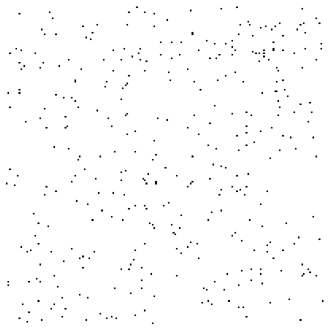
To verify our results, we implemented simulations of the environment, sensor, and map update models from Section 3.3. The simulations take place on a 200 × 200 cell grid with $\delta = 0.1$ m. Most of our simulations were with sparse environments ($d = 0.01$ or less) since denser worlds do not exhibit much variation in map quality for different sensors.

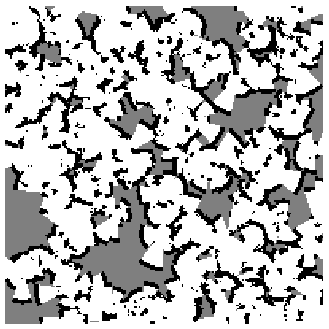
We simulated several synthetic "base" sensor models, varying individual parameters for each experiment. Here we mostly present results from a SONAR-like sensor array; we also include some interesting results from other types of sensors. The parameters for each of the sensors we tested are shown in Table 3.1. Distances are given in meters and angles in degrees. All of the sensors fired at $F = 5$ times per second and ran for $T = 25$ sec. Figure 3.2 shows an environment with density $d = 0.01$ along with typical maps produced by each of the (base) sensors.

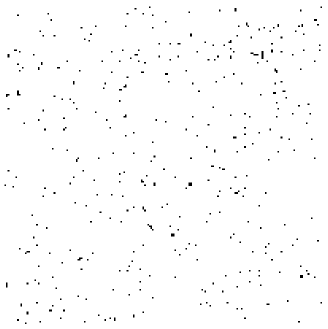
The left-hand side of Figure 3.3 shows plots of the mean cell update count for the SONAR-like sensor as each sensor parameter is individually varied. Each plot contains both the simulated result and the predicted upper bound from Theorem 3.1. We additionally show the update count obtained from simulations with a realistic trajectory with velocity and turning constraints. The right hand side of Figure 3.3 shows similar plots for the error in the maximum-likelihood maps. In computing the empirical error, a fair coin was flipped to determine the occupancy of unobserved cells.

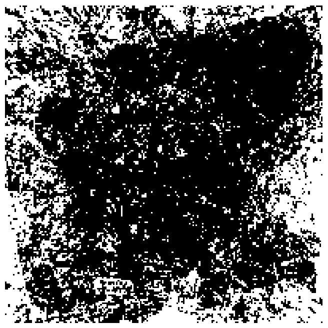
In nearly all cases, the predicted observation counts match very well with the empirical results. The upper bound on map error is less tight but is still reasonably close to the empirical result for every sensor parameter.

Figure 3.4 shows two interesting cases illustrating map divergence for bearing- and range-only sensors, as described in Section 3.5.1. Increasing the range uncertainty σ_r of a range-only sensor handicaps its ability to make any useful observations about the environment, and because the full 360° of the sensor beam is updated for each measurement, as σ_r increases so does $|\mathcal{C}_{\mathrm{F}}|$ until most map updates are incorrect. As discussed previously, a bearing-only sensor has little hope of building a correct occupancy map to begin with; increasing the beam width of the sensor only exacerbates the issue.


(a) Ground truth


(b) SONAR-like sensor


(c) Laser-like sensor


(d) Bearing-only sensor

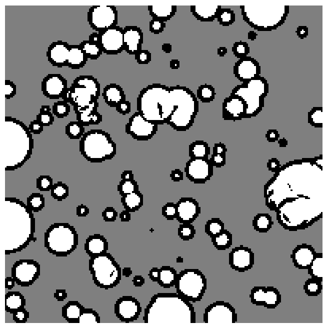

(e) Range-only sensor

Figure 3.2: An example world with $d = 0.01$ and typical maximum likelihood maps produced by each of the sensors. Occupied cells are black, empty cells white, and unobserved cells are shown here as gray for clarity. For an explanation of the bearing-only map, see Section 3.5.1.

3.7.1 Structured environments

In our analysis, we assumed unstructured environments (cells are independent). In practice, nearly every environment exhibits some kind of structure. While it may be possible to address these issues analytically, a good first step is to examine them empirically to improve our understanding of how structure affects map quality for different sensors.

To obtain structured simulated environments, we generated worlds based on a pairwise Markov random field (MRF) model using an approach similar to that of Rachlin et al. (2005). In a pairwise MRF, the occupancies of a cell m_{ab} in the "neighborhood" $N(m_{ij})$ of m_{ij} are related to that of m_{ij} through a *compatibility function* $\psi_{ij,ab}(m_{ij}, m_{ab})$; for our purposes, the compatibility function can be thought of as a likelihood function for m_{ij} conditioned on the occupancy of m_{ab}. The distribution of environments in our MRF model can be written as:

$$p(\mathcal{M}) = \eta \prod_{i,j} d \prod_{m_{ab} \in N(m_{ij})} \psi_{ij,ab}(m_{ij}, m_{ab}) \tag{3.19}$$

where d is the base occupancy likelihood of a cell. Intuitively, a pairwise MRF model generates an environment with local structure which depends on the definition of $N(m_{ij})$ and the compatibilities $\psi_{ij,ab}$. In the results given here, we used $d = 0.5$ and simple rectilinear 4-neighborhoods, with the likelihood of two adjacent cells having the same occupancy set to 0.9, and the likelihood of different occupancies set to 0.1. We obtained simulated environments by applying 20 iterations of Gibbs sampling to (3.19). Figure 3.5 shows a typical MRF-sampled environment and maps produced with the SONAR-like and laser-like sensors.

The results of simulations in MRF-generated environments with the SONAR-like sensor and

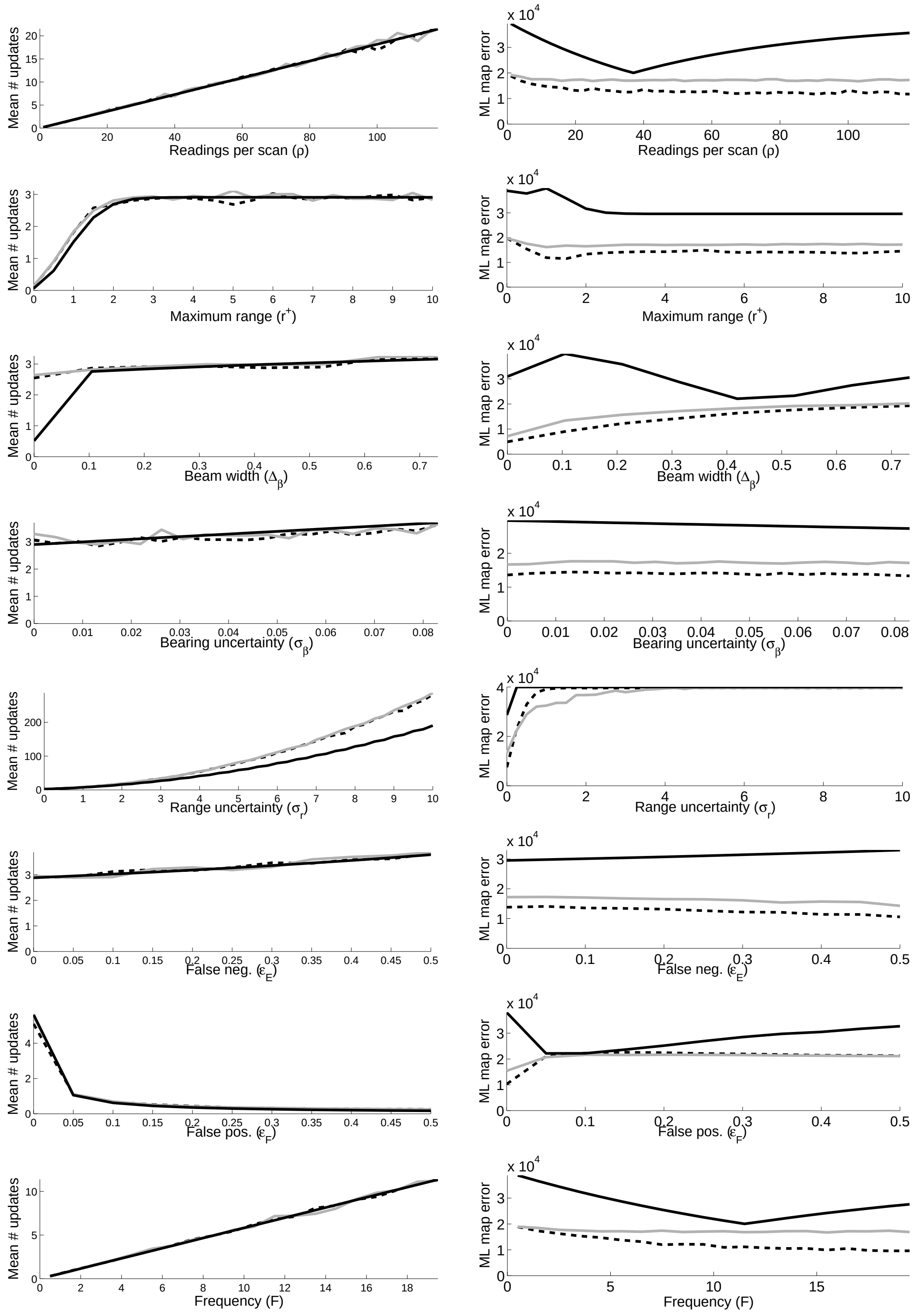
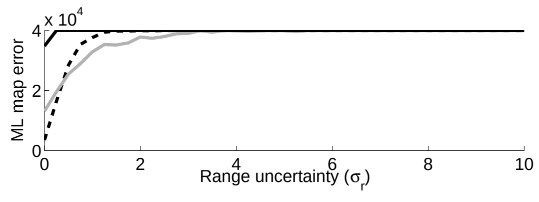


Figure 3.3: Left: mean cell update count for the SONAR-like sensor while varying individual parameters. Right: maximum likelihood map error. The solid black line is the analytical bound; the dashed line is the empirical value for random poses; and the gray line is the empirical value for realistic trajectories.



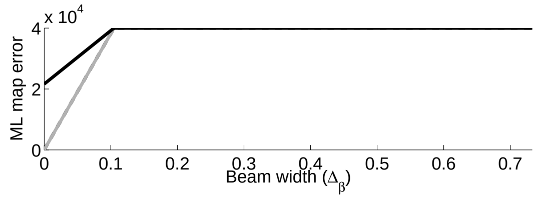
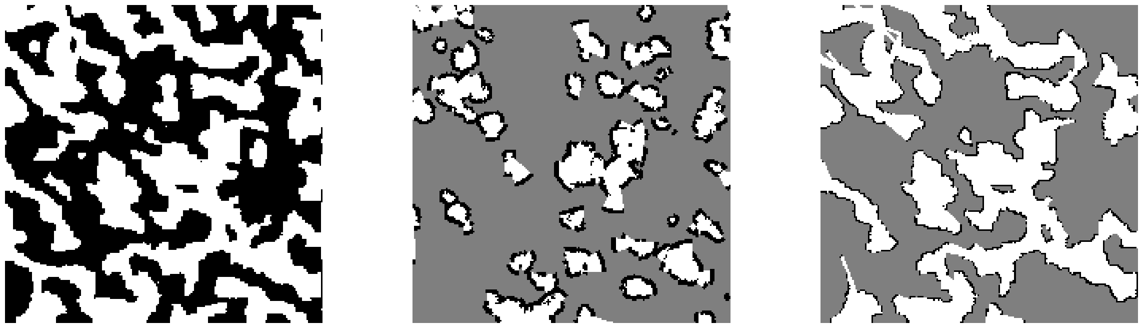


Figure 3.4: Map divergence as σ_r increases for a range-only sensor (left), and as Δ_β increases for a bearing-only sensor (right).



(a) Ground truth (b) SONAR-like (c) Laser-like

Figure 3.5: A MRF-generated environment and maps.

random poses are given in Figure 3.6. The main observation is that structure in the environment clearly invalidates our bound on observation count. This we might easily expect since we make heavy use of the independence assumption, particularly in computing p_{obs}. An interesting approach might be to incorporate the MRF model directly in our framework, i.e., computing the marginals of (3.19) as a step in determining the likelihoods $\mathcal{E}_{\mathbb{E}}$ and $\mathcal{E}_{\mathbb{F}}$.

3.7.2 An analytical comparison of real sensors

In addition to the experiments with synthetic sensors, we obtained parameters for our model for three real mapping sensors: the SICK LMS 200-30106 scanning laser rangefinder (SICK AG, 2003), the Polaroid 6500 series SONAR ranging module (SensComp/Polaroid, Inc., 2003), and the Sharp GP2D12 infrared rangefinder (Sharp Corp., 2005). Using our results, we analytically examined the relative mapping capabilities of sensors in the following configurations:

- Laser: two "back to back" rangefinders each covering 180° for total coverage of 360°
- SONAR-16, SONAR-24: an array of 16 (resp. 24) SONAR sensors at evenly spaced angles
- IR-5, IR-16: an array of 5 (resp. 16) IR sensors at evenly spaced angles

The model parameters for each configuration, gleaned mainly from the product datasheets, are listed in Table 3.2.

3.7.2.1 Basic comparison

First we apply the model assuming each robot has the same amount of time for exploration and mapping. Figure 3.7 shows the expected error versus time for each sensor in several environments. In most cases, the laser rangefinder is dramatically better than any of the SONAR or IR configurations. The improvement obtained from a 24-sensor SONAR ring over a 16-sensor ring is relatively small; this is similarly the case for a 16-sensor IR ring versus a 5-sensor ring, particularly in the sparse environments. Clearly, the range limitation and smaller beam width of the infrared

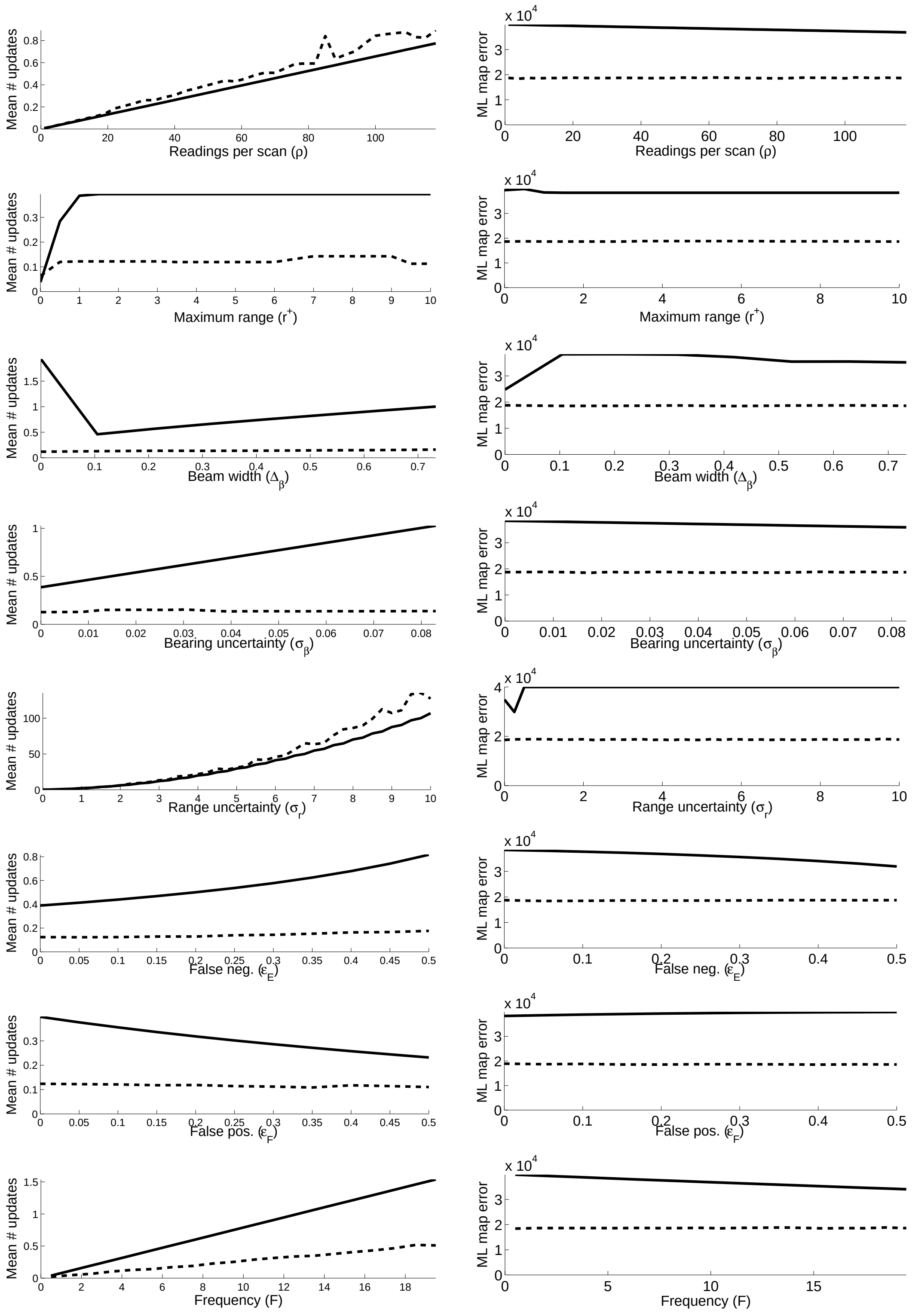


Figure 3.6: Mean cell update count (left) and map error (right) for the SONAR-like sensor in MRF-generated structured environments. The solid black line is the analytical bound; the dashed line is the empirical value for random poses.

	Explanation	Laser	SONAR-16	SONAR-24	IR-5	IR-16
F	firing frequency[a] (Hz)	20	12	8	25	10
ρ	readings/scan	360	16	24	5	16
r^+	sensing radius (m)	10	10	10	0.8	0.8
Δ_β	half beam width (°)	0.6	15	15	1.8	1.8
σ_β	bearing uncertainty[b] (°)	0.06	0.06	0.06	0.06	0.06
σ_r	range uncertainty (m)	0.015	0.1	0.1	0.015	0.015
ε_E	false negative rate	0.02	0.08	0.08	0.02	0.02
ε_F	false positive rate	0.001	0.01	0.01	0.001	0.001

[a]Most SONAR or IR arrays rely to some extent on sequential collection of data from each sensor, due to bus limitations. Thus, adding more sensors reduces the effective firing frequency.

[b]This is mainly a function of accuracy in mounting the sensors.

Table 3.2: Parameters for real sensors in several configurations

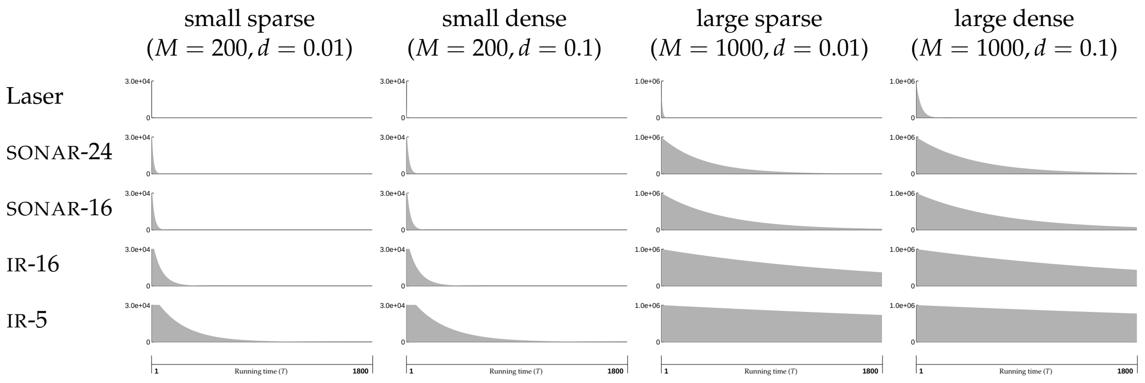


Figure 3.7: Expected maximum likelihood map error for the different sensors in several environments. In all cases, the cell size is $\delta = 0.1$ m.

array is a major handicap versus the SONAR array, despite the SONAR sensor's less favorable error rates.

In sparse environments, all of the sensors converge to essentially the correct map within a reasonable time limit. To examine this more closely, Figure 3.8 depicts the running times required by each sensor to achieve a given bound on the expected map error. The error bound is specified in terms of the "quality" Q of the map, expressed as the percentage of correct cells, i.e., $Q = (M^2 - E[\nu])/M^2$.

In effect, the plots show how much "work" must be done (i.e., how much time is required) by a robot equipped with each sensor to build a map with the desired quality. Our results indicate, for example, that a robot with a 16-SONAR array must do (approximately) between 19 and 62 times more work than a laser-equipped robot to build a comparable map in fairly sparse environments. A robot with only a 5-sensor infrared ring must do between 189 and 707 times more work. Interestingly, the ratio of work between different sensors can be seen to depend on the density of of the environment, but not the size of the environment or the desired map quality. A summary of the work required to attain several map error values in different environments appears in Table 3.3.

3.7.2.2 Laser-normalized running time

As mentioned above, the relative work required by different sensors to attain a particular map quality is independent of the size of the environment and the desired quality. This suggests a useful notion of relative sensor capability that depends only upon the complexity of the environment (density in our model) and the characteristics of the sensors.

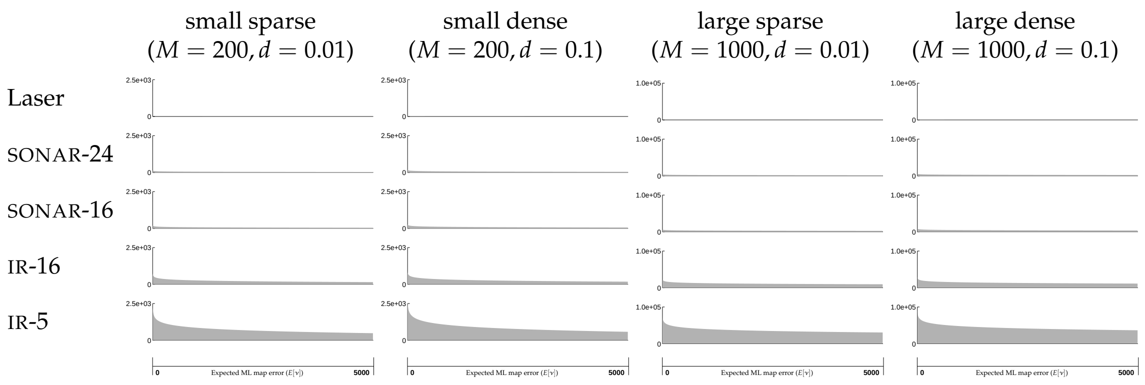


Figure 3.8: Running time required to achieve a desired expected map error in several environments. In all cases, the cell size is $\delta = 0.1$ m.

d	M	Sensor	T/T_{laser}	$T(0.999)$	$T(0.99)$	$T(0.9)$	$T(0.7)$
0.01	200	Laser	**1**	2.26	1.50	0.75	0.39
	1000			56.42	37.61	18.81	9.83
	200	SONAR-16	**61.61**	139.05	92.70	46.35	24.23
	1000			3476.26	2317.51	1158.75	605.89
	200	SONAR-24	**41.07**	92.70	61.80	30.90	16.16
	1000			2317.51	1545.00	772.50	403.92
	200	IR-5	**706.47**	1594.44	1062.96	531.48	277.90
	1000			39861.05	26574.04	13287.02	6947.50
	200	IR-16	**220.77**	498.26	332.18	166.08	86.84
	1000			12456.58	8304.39	4152.19	2171.09
0.1	200	Laser	**1**	10.09	6.72	3.36	1.76
	1000			252.14	168.09	84.05	43.95
	200	SONAR-16	**18.90**	190.67	127.11	63.56	33.23
	1000			4766.67	3177.78	1588.89	830.80
	200	SONAR-24	**12.60**	127.11	84.74	42.37	22.15
	1000			3177.78	2118.52	1059.26	553.86
	200	IR-5	**189.05**	1906.70	1271.13	635.57	332.32
	1000			47667.56	31778.37	15889.19	8308.12
	200	IR-16	**59.08**	595.84	397.23	198.61	103.85
	1000			14896.11	9930.74	4965.37	2596.29

Table 3.3: Work $T(Q)$ (time in seconds spent on exploration and mapping) to attain a given expected map quality $Q = (M^2 - E[v])/M^2$; in all cases, $\delta = 0.1$. Note that the laser-normalized running times (T/T_{laser}) are independent of Q.

Specifically, we normalize the required work by each sensor with respect to that for a "reference" sensor, using arbitrary values of M and Q, to obtain the work required by the sensors relative to the reference sensor for all environments of a particular density. For example, the column T/T_{laser} in Table 3.3 does this for two different environment densities using the laser as the reference sensor. We term this notion of normalized work *laser-normalized running time* since it specifies the time for each sensor, relative to the laser sensor model, required to build a map of equal expected quality to that built by the laser. (One could just as easily define the normalized running time with respect to any other sensor in our model. We have chosen the laser since it is the most commonly used sensor for mapping.)

Figure 3.9 shows the laser-normalized running times for each sensor with respect to the density of the environment. As the density of the environment increases, the relative capabilities

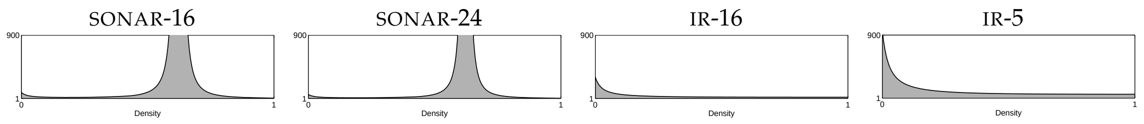


Figure 3.9: Analytically determined laser-normalized running times for the different sensors, versus density (d) of the environment. (The laser-normalized running time for the laser is always, of course, 1.)

of a laser rangefinder and an infrared array converge. This is reasonable since the laser's longer range and higher spatial resolution are less useful in highly cluttered environments where many nearby cells are occupied; aside from those characteristics, the capabilities of a laser rangefinder and infrared array are mostly the same.

The behavior of the SONAR arrays is more complex, with the work required relative to the laser rangefinder increasing dramatically between environment densities of around $d = 0.5$ to $d = 0.8$. This increase is due to the beam width of the SONAR sensors. Specifically, for environments with densities in this "middle" range, empty cells frequently fall into the region $\mathcal{C}_{\mathrm{F}}$ and are updated incorrectly; similarly, occupied cells frequently fall into $\mathcal{C}_{\mathrm{E}}^{\Delta}$. Our model thus suggests that occupancy mapping with SONAR sensors should be avoided in particularly cluttered environments. (While the model shows improvement in extremely dense environments since beam width is less of a factor, such environments are mostly impossible to map in practice anyway since a robot cannot move inside an obstacle.)

3.8 Summary

In this chapter we have described a sensor model which is detailed enough to capture much of the interesting behavior of real-world mapping sensors, and we have applied the model with a basic occupancy grid mapping algorithm, under several simplifying assumptions, to prove three main results. First, we derived a lower bound on the number of map updates applied to each cell in the grid, in terms of the sensor characteristics. Second, we gave conditions which, if met, guarantee the eventual convergence of the maximum likelihood map to the ground truth; if not met, the map diverges. Finally, we used the preceding two results to obtain an upper bound on the expected error in the maximum likelihood map for a given sensor and trajectory length. We implemented the sensor model in simulation with various synthetic sensors and found that our analytical bounds match well with the empirical results. In addition, we applied the model with real-world sensors, including a scanning laser rangefinder, SONAR arrays, and infrared rangefinder arrays, to analytically compare the mapping capabilities of the sensors.

There are several shortcomings to our approach. Most importantly, our bounds assume unstructured environments and "trajectories" consisting of random poses. Our empirical evidence suggests that in unstructured environments, the difference in map quality between realistic trajectories and random poses is slight. However, structure in the environment has a significant effect on the ability of a robot to build a map, motivating further study of how to analyze sensing capabilities in the presence of structure. Finally, we believe our framework may also be useful in studying feature-based mapping, sensor-based exploration strategies, and SLAM.

The theoretical analysis presented in this chapter shows that, when faced with sensing limitations, one must do more "work" — i.e., take more measurements — to build a high-quality map. In the following chapters, we develop practical algorithms for feature-based mapping with limited sensors similar to the small infrared arrays described earlier. These sensors are only able to partially observe the parameters of environment features with a single scan, but by doing more work — collecting data from several scans at different poses — we are able to obtain enough information to do feature extraction.

4 SLAM WITH SPARSE SENSING

This chapter examines the simultaneous localization and mapping (SLAM) problem (see Section 2.4 and Appendix B) for robots with low spatial resolution range-bearing sensors, such as infrared rangefinder arrays. These sensors provide too little data to extract features of the environment from a single scan, so we develop a particle filtering SLAM algorithm which groups several scans taken as the robot moves into *multiscans*, which contain sufficient information for extraction and data association. We formulate a full system model for this approach, and then introduce simplifications that enable efficient implementation using a Rao-Blackwellized particle filter. We additionally describe a simple algorithm for feature extraction of line features from multiscans, and enhancements to our approach for trading off accuracy and computation, and then present experimental results using real data from several environments.

4.1 Overview

Figure 1.1 on page 7 compares the data density provided by a single scan from a typical scanning laser rangefinder with that of a small infrared rangefinder array. As we have noted in the previous chapters, laser data is sufficient for extracting meaningful features of the environment such as lines or corners, or for direct "scan matching" against a map. In contrast, it is much more difficult to extract features from a single sparse scan. Thus, using a sparse array demands an approach for SLAM that is somewhat different from the traditional sequence of moving, sensing, extracting features, finding correspondences, and then updating the map and pose estimates.

Our approach is to group consecutive sparse scans into *multiscans* so the data from multiple frames can be used to extract good features. The tradeoff is that uncertainty in the robot's motion contributes to noise in feature extraction. This approach requires the robot's trajectory (or "pose history") to be kept in order to process the multiscan. While others, e.g., Leonard et al. (2002) have explicitly stored the pose history in the system state, we instead separate the pose history from the system state, enabling feature extraction to be performed only once per multiscan in a particle filtering implementation. Our approach yields an efficient SLAM algorithm for robots with sparse sensing.

We note that because our algorithm is passive, we cannot make any guarantees about the quality or completeness of the data produced by a multiscan. For example, a robot that moves through empty spaces away from landmarks will obtain no useful information for mapping. Furthermore, our approach requires a robot's trajectory to be (effectively) continuous — collecting

m	multiscan size	$\mathbf{z}_{t-m+1:t}$	multiscan	C	num. stratification chunks
$\mathbf{x}^r_{t-m+1:t}$	multiscan trajectory	$\mathbf{P}_{t-m+1:t}$	multiscan covariance	$\overline{\mathbf{x}}^r_{t-m+1:t}$	mean intermediate trajectory

Notation for this chapter

individual scans from spatially close-by vantage points enables us to garner sufficient information to extract parameters for individual features. However, despite these limitations, the multi-scan approach works well in practical scenarios because mobile robots indeed follow continuous trajectories, which typically produce consecutive scans that give multiple distinct observations of environment features.

4.2 SLAM with partially observable features

In Chapter 1 we briefly discussed previous research on mapping with limited sensing (Section 1.5). Others have recognized that the primary issue in building maps with low-resolution sensors is that the sensors are incapable of instantaneously observing all parameters of environment features — i.e., features are only *partially observable*. Partial observability arises in several contexts.

Bearing-only sensors (Deans and Hebert, 2000a; Bailey, 2003; Solá et al., 2005; Lemaire et al., 2005) are one such case. Bearing-only SLAM is typically performed with monocular cameras. Cameras offer dense coverage of the environment over their field of view, which enables accurate "discovery" and data association of features in images. Thus, it is reasonably straightforward to fully initialize landmarks given good image registration techniques, e.g., (Lowe, 2004), and sufficient baseline (e.g., via motion of the robot).

In contrast, sparse range-bearing sensor arrays, such as SONAR and infrared rangefinder arrays, are very low-resolution over the field of view. A single scan from such a sensor is insufficient to initialize line or point features, and due to the low data density, data association between individual scans is essentially impossible. This motivates an approach in which a robot's proprioceptive odometry information is incorporated into the feature extraction process, e.g., by gathering measurements from blocks of poses along the robot's trajectory and extracting features from the aggregate data.

Several researchers have examined the problem of using SONAR data from multiple poses to extract features and perform SLAM (Wijk and Christensen, 2000; Zunino and Christensen, 2001; Tardós et al., 2002; Leonard et al., 2002). An advantage of SONAR sensors is that, while they provide only imprecise bearing information, the wide beam width of a SONAR sensor enables full range-only coverage of a large field of view. Together with Hough transform methods for feature extraction using data from several poses, SONAR sensors can observe line or point landmarks with reasonable accuracy.

We instead study the SLAM problem for robots with sparse arrays of *narrow* beam width sensors, e.g., infrared rangefinder arrays. While these sensors give precise bearing information, they only sparsely cover the field of view of the array. Thus, both range and bearing information about features can only be observed through motion of the robot.

4.3 SLAM with a pose history

When features can only be detected from multiple poses, one approach to SLAM is to incorporate the pose history into the state vector. This is the method used by Leonard et al. (2002), who apply an EKF to the resulting system. In this section, we give a brief description of this approach and how one might apply a particle filter to this system. This sets the context for our simplified approach to the problem, described in Section 4.4.

Suppose we keep the most recent m robot poses in the system state vector. Then the state is:

$$\mathbf{x}_t = [\mathbf{x}^r_{t-m+1:t} \;\; \mathbf{x}^m_t]^T = [\mathbf{x}^r_{t-m+1} \;\; \mathbf{x}^r_{t-m+2} \;\; \ldots \;\; \mathbf{x}^r_t \;\; \mathbf{x}^m_t]^T \tag{4.1}$$

where $\mathbf{x}^r_t$ is the robot pose at time t and $\mathbf{x}^m_t$ is a vector of n landmarks. This fits into the standard system model:

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_t) + \mathbf{v}_t \tag{4.2}$$

$$\mathbf{z}_t = h(\mathbf{x}_t) + \mathbf{w}_t \tag{4.3}$$

where $\mathbf{u}_t$ is the input, $\mathbf{z}_t$ the measurement, and $\mathbf{v}_t$ and $\mathbf{w}_t$ are system and measurement noise, respectively.

4.3.1 Using the EKF

Leonard et al. (2002) apply the EKF to this system, so they maintain an estimate of the state vector as well as its full covariance:

$$\mathbf{P}_{\mathbf{x}} = \begin{bmatrix} \mathbf{P}^{r}_{t-m+1:t} & \mathbf{P}^{r,m}_{t-m+1:t} \\ \left(\mathbf{P}^{r,m}_{t-m+1:t}\right)^{T} & \mathbf{P}^{m} \end{bmatrix} \tag{4.4}$$

where $\mathbf{P}^{r}_{t-m+1:t}$, $\mathbf{P}^{r,m}_{t-m+1:t}$, and $\mathbf{P}^{m}$ are covariance matrices for the robot pose history and the landmark locations. The key points in applying the EKF to this system are:

- When the state is projected forward, the new robot pose $\mathbf{x}^{r}_{t}$ is inserted into $\mathbf{x}_t$ and the oldest pose is discarded.
- Feature extraction is done using the last m pose estimates and the corresponding m measurements.

(For a detailed discussion of the EKF for SLAM, see Appendix B.) Since the full covariance (and cross covariance) of the past m robot poses are maintained, feature extraction can account for the uncertainty of (and correlation between) poses from which the sensor measurements were taken.

However, the computation of the full covariance matrix is very expensive (at least $O((m+n)^2)$ complexity), and it must be performed at each time step. Furthermore, EKF-SLAM is unable to represent multiple data association hypotheses or multimodal distributions over the state space. Recently, particle filtering techniques have been used to overcome these limitations in traditional SLAM formulations. Appendix B introduces the basic particle filtering approach for SLAM.

4.3.2 Using a particle filter

A reasonably straightforward adaptation of the above model to a standard Rao-Blackwellized particle filter is:

Algorithm 4.1 RBPF with a pose history

```
1: for each particle $\phi^i = \{\mathbf{x}^i_{t-1}, \omega^i_{t-1}\}$, $i = 1 \ldots N$ do
2:     Project the state forward by drawing a new robot pose $\mathbf{x}^{r,i}_t$ from the distribution of $\mathbf{v}_t$ centered at $f(\mathbf{x}^{r,i}_{t-1}, \mathbf{u}_t)$
3:     Insert $\mathbf{x}^{r,i}_t$ in $\mathbf{x}^i_t$ and discard the oldest robot pose
4:     Extract features from the measurements using the last $m$ poses and perform data association
5:     Update the map and initialize new features
6:     Compute a new weight $\omega^i_t$ equal to the likelihood of the data association
7: end for
8: If necessary, resample the particles with probabilities proportional to $\omega^i_t$
```

Note that a "lag-m" pose history is stored with each particle, so collectively the particles sample the space of the previous m poses. This approach avoids the expensive computation of $(m+n)$-dimensional covariance matrices as required by the EKF, but potentially, many particles are required to adequately sample the pose history space. Perhaps more importantly, feature extraction is required for every particle at each time step because of their unique pose histories. The computation for feature extraction is generally fairly expensive, so this straightforward adaptation is infeasible in practice.

4.4 SLAM using multiscans

Our approach is based on grouping sparse scans from m consecutive robot poses into a *multiscan* $\mathbf{z}_{t-m+1:t} = [\mathbf{z}_{t-m+1} \;\; \mathbf{z}_{t-m+2} \;\; \ldots \;\; \mathbf{z}_t]^T$. We formulate a system model in which a SLAM update is performed only after each m steps, reducing the required computation. A further simplification enables a particle filter implementation where features are extracted only once per multiscan (instead of once per particle per multiscan).

4.4.1 System model

The state vector is $\mathbf{x}_t = [\mathbf{x}^r_t \;\; \mathbf{x}^m_t]^T$, i.e., $\mathbf{x}_t$ contains only the single pose $\mathbf{x}^r_t$. Our system model is:

$$\mathbf{x}_t = F(\mathbf{x}_{t-m+1}, \mathbf{u}_{t-m+1:t}) \tag{4.5}$$
$$\mathbf{Z}_t = g(\mathbf{z}_{t-m+1:t}, \mathbf{x}^r_{t-m+1:t}, \mathbf{x}^m_{t-m}) \tag{4.6}$$

where $\mathbf{u}_{t-m+1:t}$ is the vector of control inputs $[\mathbf{u}_{t-m+1} \;\; \mathbf{u}_{t-m+2} \;\; \ldots \;\; \mathbf{u}_t]^T$.

The system function F is defined as:

$$F(\mathbf{x}_{t-m}, \mathbf{u}_{t-m+1}) = \begin{bmatrix} \mathbf{x}^r_t \\ \mathbf{x}^m_{t-m} \end{bmatrix} \tag{4.7}$$

where the pose $\mathbf{x}^r_t$ is modeled recursively as:

$$\mathbf{x}^r_t = f(\mathbf{x}^r_{t-1}, \mathbf{u}_t) + \mathbf{v}_t \tag{4.8}$$

The pose $\mathbf{x}^r_{t-m}$ is taken from the state vector $\mathbf{x}_{t-m}$ after the previous SLAM update.

The function g computes a feature vector $\mathbf{Z}_t$ containing the parameters of features extracted from the multiscan $\mathbf{z}_{t-m+1:t}$, which is acquired from the intermediate robot poses $\mathbf{x}^r_{t-m+1:t} = [\mathbf{x}^r_{t-m+1} \;\; \ldots \;\; \mathbf{x}^r_t]^T$. Each scan $\mathbf{z}_t$ in a multiscan is modeled by:

$$\mathbf{z}_t = h(\mathbf{x}^r_t, \mathbf{x}^m_{t-m}) + \mathbf{w}_t \tag{4.9}$$

where $\mathbf{w}_t$ is the measurement noise and $\mathbf{x}^m_{t-m}$ is the landmark vector (map) from the most recent SLAM update.

4.4.2 Multiscan particle filter

We apply a Rao-Blackwellized particle filter to this system, assuming the landmarks are independent when conditioned on the robot's trajectory:

4.4.3 Feature extraction

Because our features are extracted from sensor readings taken from different poses, both the measurement noise and the odometry error contribute to uncertainty in the extracted features. In our particle filter implementation, however, we do not maintain a pose history of the intermediate states for each particle. Instead, we use the expected intermediate pose history $\overline{\mathbf{x}}^r_{t-m+1:t} = [\overline{\mathbf{x}}^r_{t-m+1} \;\; \ldots \;\; \overline{\mathbf{x}}^r_t]^T$, calculated from the robot's odometry as:

$$\overline{\mathbf{x}}^r_t = f(\overline{\mathbf{x}}^r_{t-1}, \mathbf{u}_t) \tag{4.10}$$

Without loss of generality, we assume $\overline{\mathbf{x}}^r_{t-m}$ is the origin.

We perform feature extraction using this expected pose history, and then transform the features for each particle p^i so that the pose $\overline{\mathbf{x}}^r_t$ coincides with $\mathbf{x}^{r,i}_t$ to find correspondences. While this approach does not precisely account for the correlation between robot movement and feature parameters and increases feature extraction uncertainty, it avoids performing feature extraction separately with the intermediate pose history of every particle. Section 4.6 describes an approach for trading off computation with feature extraction accuracy by performing feature extraction

Algorithm 4.2 Multiscan RBPF SLAM

```
1: loop
2:   for $m$ time steps do
3:     Move and collect sparse scans
4:   end for
5:   Extract features by considering the data from the previous $m$ scans simultaneously as a
     single multiscan.
6:   for each particle $\phi^i = \{\mathbf{x}^i_{t-m}, \omega^i_{t-m}\}$, $i = 1 \ldots N$ do
7:     for $k = t - m + 1$ to $t$ do
8:       Draw $\mathbf{x}^{r,i}_k \sim p(\mathbf{x}^r_k | \mathbf{x}^{r,i}_{k-1}, \mathbf{u}_k)$ to project the pose forward
9:     end for
10:    Find correspondences between extracted features and the landmark locations $\mathbf{x}^{m,i}_{t-m}$
11:    Update the map and initialize new landmarks
12:    Compute a new weight $\omega^i_t$ equal to the likelihood of the data association
13:  end for
14:  If necessary, resample the particles with probabilities proportional to $\omega^i_t$
15: end loop
```

with the expected trajectories of "clusters" of particles, instead of for the complete set of particles at once.

4.4.4 Innovation covariance

We consider $\mathbf{Z}_t$ to be the measurement for SLAM updates, and thus the innovation for a measurement lies in the feature parameter space. The innovation covariance $\mathbf{S}$ is required for both maximum likelihood data association and to update landmark locations. To simplify our explanation, assume that all features in $\mathbf{Z}_t$ are associated with landmarks whose parameters are in $\tilde{\mathbf{x}}^m_t = \mathbf{M}\mathbf{x}^m_t$, where $\mathbf{M}$ is a matrix that simply selects the appropriate landmarks from $\mathbf{x}^m_t$. The innovation is then $\boldsymbol{\nu} = \mathbf{Z}_t - \tilde{\mathbf{x}}^m_t$, and its covariance is:

$$\mathbf{S} = \mathbf{H}_g \mathbf{P}_{t-m+1:t} \mathbf{H}_g^T + \mathbf{M}\mathbf{P}_{\mathbf{x}_t}\mathbf{M}^T \tag{4.11}$$

where $\mathbf{H}_g$ is the Jacobian of the feature extraction process with respect to the multiscan, $\mathbf{P}_{\mathbf{x}_t}$ is the covariance of the system state vector (which is readily available), and $\mathbf{P}_{t-m+1:t}$ is the covariance of the multiscan and pose history:

$$\mathbf{P}_{t-m+1:t} = \begin{bmatrix} \mathbf{R}_{t-m+1:t} & \mathbf{P}^{r,z}_{t-m+1:t} \\ \left(\mathbf{P}^{r,z}_{t-m+1:t}\right)^T & \mathbf{P}^r_{t-m+1:t} \end{bmatrix} \tag{4.12}$$

Feature extraction can be performed using the full $\mathbf{P}_{t-m+1:t}$. A further approximation that yields acceptable results is to represent $\mathbf{P}_{t-m+1:t}$ using only the block diagonal portions of $\mathbf{R}_{t-m+1:t}$ and $\mathbf{P}^r_{t-m+1:t}$, i.e., to assume that measurements are independent and that although pose uncertainty compounds, multiscan poses are independent.

For complicated feature extraction methods, $\mathbf{H}_g$ is difficult to compute analytically. However, it is well-known that maximum likelihood estimation gives good estimates for the covariance of the parameters being estimated, even for an approximately specified model such the block diagonal version of $\mathbf{P}_{t-m+1:t}$ (White, 1982). Thus, by using maximum likelihood estimation as a feature extractor, we can obtain a good approximation to the innovation covariance $\mathbf{S}$ without computing $\mathbf{H}_g$. We use this approach to extract lines from multiscan data with a procedure described in Section 4.5.

Algorithm 4.3 EXTRACT-LINES($\mathbf{x}^r_{t-m+1:t}, \mathbf{z}_{t-m+1:t}, \mathbf{P}_{t-m+1:t}$)

```
1: Compute points $\mathbf{p}_1 \dots \mathbf{p}_n, n = |\mathbf{z}_{t-m+1:t}|$, from raw scan data in $\mathbf{z}_{t-m+1:t}$
2: Sort $\mathbf{p}_1 \dots \mathbf{p}_n$ radially with respect to $\mathbf{x}^r_t$ (alternatively, w.r.t. $\text{mean}_{k \in t-m+1:t}\, \mathbf{x}^r_k$)
3: $r_{\min} = 1, r_{\max} = 2$
4: repeat                                                    // initial breakpoint detection
5:   if $||\mathbf{p}_{r_{\max}} - \mathbf{p}_{r_{\max}-1}|| > D_{\max}$ or $r_{\max} = n$ then   // consecutive points too far apart?
6:     if $r_{\max} - r_{\min} > P_{\min}$ then               // enough points to estimate a line?
7:       $\mathcal{R} \leftarrow \mathcal{R} \cup \{[r_{\min}, r_{\max}]\}$            // add this cluster
8:     $r_{\min} \leftarrow r_{\max}$
9:   end if
10:  $r_{\max} \leftarrow r_{\max} + 1$
11: until $r_{\max} > n$
12: while $\mathcal{R} \neq \emptyset$ do                      // repeat until all clusters converge
13:   $\mathcal{R}' \leftarrow \emptyset$
14:   for all $[r_{\min}, r_{\max}] \in \mathcal{R}$ do
15:     Compute least squares line parameters $\boldsymbol{\ell} = [r \;\; \theta]^T$ from $\mathbf{p}_{r_{\min}} \cdots \mathbf{p}_{r_{\max}}$
16:     Let $k^+ = \text{argmax}_{k \in [r_{\min}, r_{\max}]} ||\mathbf{p}_k - \boldsymbol{\ell}||$
17:     if $||\mathbf{p}_{k^+} - \boldsymbol{\ell}|| < T_{\max}$ then      // is max-distance point close to line?
18:       $\mathcal{S} \leftarrow \mathcal{S} \cup \{[r_{\min}, r_{\max}]\}$       // done splitting this cluster
19:     else                                             // split the cluster at the farthest point
20:       if $k^+ - r_{\min} > P_{\min}$ then                 // first subcluster — enough points?
21:         $\mathcal{R}' \leftarrow \mathcal{R}' \cup \{[r_{\min}, k^+]\}$
22:       if $r_{\max} - k^+ - 1 > P_{\min}$ then             // second subcluster — enough points?
23:         $\mathcal{R}' \leftarrow \mathcal{R}' \cup \{[k^+ + 1, r_{\max}]\}$
24:     end if
25:   end for
26:   $\mathcal{R} \leftarrow \mathcal{R}'$
27: end while
28: for all $[r_{\min}, r_{\max}] \in \mathcal{S}$ do                // compute lines and extents
29:   Compute full max. likelihood parameters/covariance $\boldsymbol{\ell}, \mathbf{P}_{\boldsymbol{\ell}}$ from $\mathbf{p}_{r_{\min}} \cdots \mathbf{p}_{r_{\max}}$
30:   Let $\mathbf{p}^-, \mathbf{p}^+ \in \{\mathbf{p}_{r_{\min}} \cdots \mathbf{p}_{r_{\max}}\}$ be the two extremal points when projected onto $\boldsymbol{\ell}$
31:   $\mathcal{R} \leftarrow \mathcal{R} \cup \{[\boldsymbol{\ell} \;\; \mathbf{p}^- \;\; \mathbf{p}^+]^T\}$          // store line and endpoints
32: end for
33: $\mathcal{S} \leftarrow \emptyset$
34: repeat                                                   // merge similar line segments
35:   Compute max. likelihood pairings of all segments in $\mathcal{R}$
36:   for all ML pairings $(\mathbf{L}, \mathbf{L}')$ do
37:     if likelihood$(\mathbf{L} = \mathbf{L}') > H_{\min}$ then
38:       $\mathcal{R} \leftarrow (\mathcal{R} \setminus \{\mathbf{L}, \mathbf{L}'\}) \cup \{\text{merge}(\mathbf{L}, \mathbf{L}')\}$ // remove, then merge and re-insert the two segments
39:     else                                             // keep the segments separate
40:       $\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathbf{L}, \mathbf{L}'\}$
41:   end for
42: until $\mathcal{R} = \emptyset$
43: for all $[\boldsymbol{\ell} \;\; \mathbf{p}^- \;\; \mathbf{p}^+]^T \in \mathcal{S}$ do
44:   if $||\mathbf{p}^+ - \mathbf{p}^-|| < L_{\min}$ then
45:     $\mathcal{S} \leftarrow \mathcal{S} \setminus [\boldsymbol{\ell} \;\; \mathbf{p}^- \;\; \mathbf{p}^+]^T$          // discard too-short segments
46: end for
47: return $\mathcal{S}$
```

4.4.5 Co-dependence on odometry

Notice that there is an issue of independence in the multiscan formulation. The robot's odometry is used twice, first to update the pose estimate, and second to extract features. The multiscan approach makes an implicit assumption that the co-dependence of the robot's pose and the measurements can be ignored. For this to be true, m, the number of readings per multiscan, must be kept small enough that the odometry error experienced over the course of the multiscan is small. One strategy is to make m a function of the pose uncertainty.

4.5 Feature extraction with multiscan data

We have implemented sparse sensing SLAM with line features. The features are parameterized by distance r and angle θ to the origin (see Section 2.7). For purposes of data association and display, basic extent information is kept for the features, but is not formally incorporated into the SLAM filtering process. Line features have been used for SLAM on a number of occasions, e.g., (Leonard et al., 2002; Tardós et al., 2002; Rodriguez-Losada and Matia, 2003; Yuen and MacDonald, 2003), and methods for performing line extraction and data association with (r, θ) lines are readily available (Pfister et al., 2003; Nguyen et al., 2005).

Before line parameters can be estimated, the data from a multiscan must be segmented. Since our data are sparse and from different poses, most common segmentation methods, such as the adaptive breakpoint detector of Borges and Aldon (2004), cannot be directly employed because they assume ordered measurements from a single pose. One solution is to cluster the points from the multiscan using, e.g., an agglomerative clustering technique or a more involved principle components analysis based approach like that of Brunskill and Roy (2005).

A simpler approach which we have employed is to select a single reference configuration, e.g., $\mathbf{x}_t^r$ (the final pose in the multiscan) or $\overline{\mathbf{x}}_{t-m+1:t}^r$ (the average of the poses in the multiscan). Then, the multiscan data points are sorted radially with respect to the reference pose. The resulting ordering is used initially with an iterative endpoint filter (IEPF) like algorithm (Duda and Hart, 1973) to segment the points into clusters. Lines are fit to each cluster using simple least squares, and recursive splitting and refitting of the lines is performed as necessary. Clusters with too few points are discarded. Once the clusters have converged, a full maximum likelihood estimation is performed to obtain line parameters and covariances, using the approach of Pfister et al. (2003), and extent information is computed by projecting extremal points onto the lines. Finally, extracted lines with similar parameters are merged using well-known techniques for merging Gaussians.

The entire approach is an instance of the split-merge framework described by Nguyen et al. (2005). While the computation of a radial ordering of the data points with respect to a single reference configuration is an approximation, the resulting feature extraction is both efficient and accurate in practice for reasonably short multiscan trajectories. Pseudocode for our feature extraction algorithm is shown in Algorithm 4.3.

Maximum likelihood data association and map updates for line features are straightforward since the parameters of lines can be directly compared and merged.

4.5.1 An alternative: segment features

We have also implemented SLAM with line segment features. An advantage of segment features is that they explicitly encode extent information in the state. In order to extract segments from multiscan data, we employ the same procedure as for line features, and then project the clustered points onto their associated lines. The two extremal projected points $\mathbf{p}_i$ and $\mathbf{p}_j$ are used as the segment parameters, and the parameter covariance is computed based on the covariances of the line parameters and of the projected endpoints, i.e.:

$$\mathbf{P_s} = \mathbf{H}_\ell \mathbf{P}_\ell \mathbf{H}_\ell^T + \mathbf{H}_{\mathbf{p}_i} \mathbf{P}_{\mathbf{p}_i} \mathbf{H}_{\mathbf{p}_i}^T + \mathbf{H}_{\mathbf{p}_j} \mathbf{P}_{\mathbf{p}_j} \mathbf{H}_{\mathbf{p}_j}^T \tag{4.13}$$

where $\mathbf{H}_\ell, \mathbf{H}_{\mathbf{p}_i}$, and $\mathbf{H}_{\mathbf{p}_j}$ are the derivatives of $\mathbf{s}$ with respect to ℓ, $\mathbf{p}_i$, and $\mathbf{p}_j$, respectively.

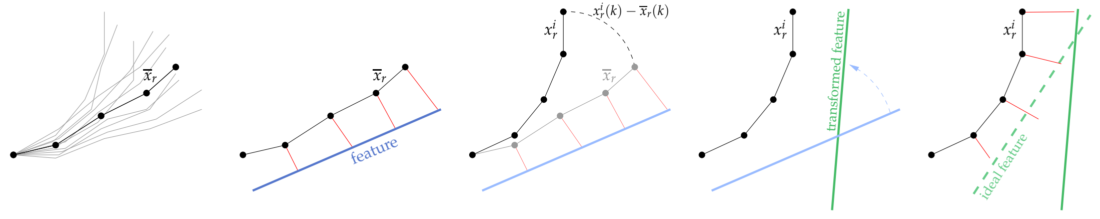


Figure 4.1: Extracting features using the expected trajectory and transforming for each particle. While this approach works well in many cases, it can yield poor results for trajectories with noisy odometry. In this case, the ideal and transformed features are significantly different.

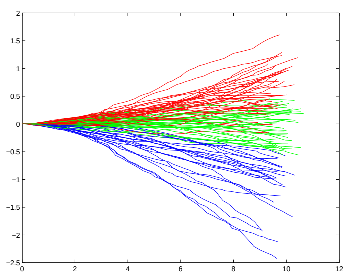
A complication of line segment features is that it is impossible to directly compare the parameters of two segments for data association since they may actually represent different portions of the same feature. A simple solution is to "extend" the segments so that their endpoints can be directly compared. We extend two segments so that their projections onto the angular bisector are the same. Care must be taken to update the endpoint covariances accordingly. Unfortunately the extension procedure represents a complicated function with convoluted Jacobians, so updating the covariance is hard. A simple approximation is to assume that the lengths each endpoint is extended, d_0 and d_1, are known parameters of a function that transforms a single segment, i.e., $E(\mathbf{s}, d_0, d_1)$, which has a simpler Jacobian, $\mathbf{H_s} = \partial E/\partial \mathbf{s}$, that can be used to transform the segment covariance. In practice, this simplification works reasonably well and is much easier to compute.

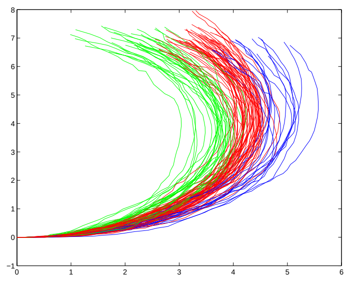
4.6 Per-stratum feature extraction

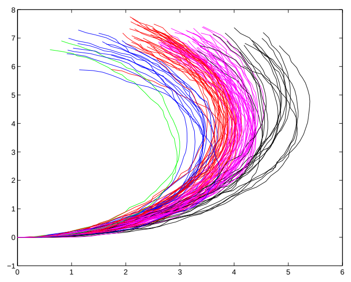
The approach of extracting features based on the expected intermediate trajectory and then transforming the features using a single reference configuration for each particle is computationally efficient. However, for longer intermediate trajectories or very noisy odometry, the results can be significantly different than those obtained by performing extraction conditioned on the separate trajectories of each particle. Figure 4.1 shows a synthetic example where the ideal feature extraction is much different than the feature obtained by the extract-transform approach.

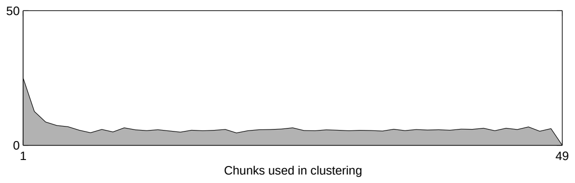
Algorithm 4.4 CHUNKED-TRAJECTORY-CLUSTER($\{\mathbf{x}^{r,1}_{t-m+1:t}, \ldots, \mathbf{x}^{r,N}_{t-m+1:t}\}$)

```
1: for $i = 1 \ldots N$ do                                  // iterate over particles
2:   for $c = 1 \ldots C$ do                                // initialize chunk means for this particle
3:     $\boldsymbol{\mu}_i[c] \leftarrow \mathbf{0}$
4:   end for
5:   $c \leftarrow 1$
6:   $\mathbf{x}_0 \leftarrow [x^{r,i}_{t-m+1} \; y^{r,i}_{t-m+1}]^T$             // start pose for the multiscan
7:   for $k = t-m+1 \ldots t$ do                             // iterate over time steps in multiscan
8:     if $k > c\frac{m}{C}$ then
9:       $c \leftarrow c+1$                                  // increment to next chunk
10:    $\hat{\mathbf{x}} \leftarrow [x^{r,i}_k \; y^{r,i}_k]^T - \mathbf{x}_0$           // point relative to multiscan start
11:    $\boldsymbol{\mu}_i[c] \leftarrow \boldsymbol{\mu}_i[c] + \frac{m}{C}\hat{\mathbf{x}}$        // update the mean for chunk c
12:  end for
13: end for
14: Perform $K$-means clustering of the $2C$-dimensional vectors $\{\boldsymbol{\mu}_i\}$ and store the cluster indices
    for the particles in $\{\kappa_i\}$
15: return $\{\kappa_i\}$
```


(a) Clustering of 50 sampled trajectories for a straight motion. 3 clusters, $C = 5$.


(b) Clustering of 50 sampled trajectories for a left turning motion. 3 clusters, $C = 5$.


(c) Clustering of 50 sampled trajectories for a left turning motion. 5 clusters, $C = 15$.



(d) Number of particles assigned to the same cluster as for $C = 50$, as C varies. For $C > 5$, the quality of the clustering is approximately the same. Data generated over 1000 Monte Carlo trials.

Figure 4.2: Stratification of intermediate trajectories. Features can be extracted separately using the mean trajectory of each cluster (or "stratum"), and then transformed for each particle in the stratum. The clustering is indicated by color.

This tradeoff between accuracy — extracting features separately for every particle — and efficiency — extracting features once for all particles and transforming — can be managed by clustering (or "stratifying") the intermediate trajectories into groups with similar pose histories relative to the starting pose, and using the extract-transform approach within each stratum of particles. Increasing the number of strata yields better features at the expense of computation time.

Clustering of similar trajectories is a difficult problem because, even for short trajectories, clustering must take place in a high-dimensional space (three dimensions per pose). Our approach exploits properties of the trajectories being clustered to simplify the problem somewhat.

To cluster intermediate trajectories over a short time span, we employ a simple "chunking" dimensionality reduction technique. We divide the intermediate trajectory into C chunks of consecutive poses, computing the mean pose for each chunk. A K-means algorithm is applied to cluster the poses in the resulting $3C$-dimensional space. A further simplification, useful because it is difficult to define the "distance" between poses with orientation, is to cluster only in the $2C$-dimensional space of the mean chunk (x, y) positions. A partial justification for this approach with differential drive robots is that, under the assumption that the robot drives in a direction related to its orientation, the robot's heading is at least partially encoded in the consecutive relative (x, y) components of the trajectory.

Pseudocode for this intermediate trajectory clustering technique is given in Algorithm 4.4. We have applied the algorithm to a variety of short trajectory samples with positive results. Several examples of the clusters resulting from applying the algorithm to different intermediate trajectories are given in Figure 4.2. In our experiments, we have found that only a few chunks are generally sufficient to obtain clusters that are nearly identical to those obtained by K-means on the full trajectory dimensionality, enabling efficient stratification of particles for multiscan feature extraction. Figure 4.2(d) shows the differences in cluster assignments as the number of chunks

Table 4.1: Sparse sensing SLAM statistics

	USC SAL	CMU NSH	Stanford
Dimensions	39m $\times$ 20m	25m $\times$ 25m	64m $\times$ 56m
Particles	100	600	1000
Sensing range	5 m	3 m	5 m
Path length	122 m	114 m	517 m
Path rotation	450 rad	133 rad	495 rad
Scans per multiscan	50	40	18
Total multiscans	89	118	1151
Avg. MS translation	1.37 m	0.97 m	0.45 m
Avg. MS rotation	3.80 rad	1.12 rad	0.43 rad
Feature type	Lines	Segments	Segments
Num. landmarks	88	168	750

increases for a particular example.

4.7 Experimental results

We have implemented our sparse sensing SLAM approach and tested it on a variety of data sets. Our Rao-Blackwellized particle filter is implemented as shown in Algorithm 4.2; we also use the adaptive sampling approach described by Grisetti et al. (2005).

Most aspects of our implementation have been discussed in previous sections. In our feature extraction, a simple weighted least squares estimator was used instead of a full maximum likelihood approach, for simplicity and efficiency. Also, the experiments presented here with line segment features estimated covariance using only the two segment endpoints rather than the full data.

Our experiments used data from Radish (Howard and Roy, 2003), an online repository of SLAM data sets. Most of the available data sets use scanning laser rangefinders with 1° spacing. In order to test SLAM with sparse sensing, we simply discarded most of the data in each scan, using only the five range measurements at $0^\circ, 45^\circ, 90^\circ, 135^\circ$, and 180°. Additionally, we restricted the maximum sensor range to at most 5 m, and in some cases less.

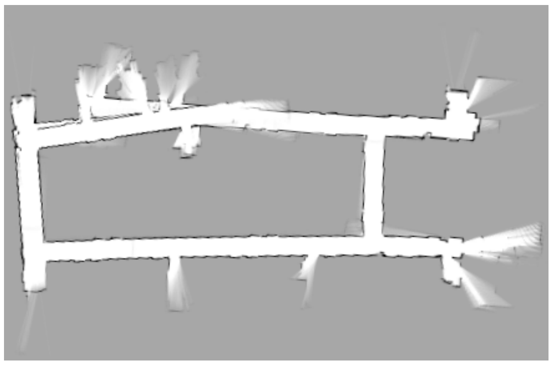
4.7.1 Experiments

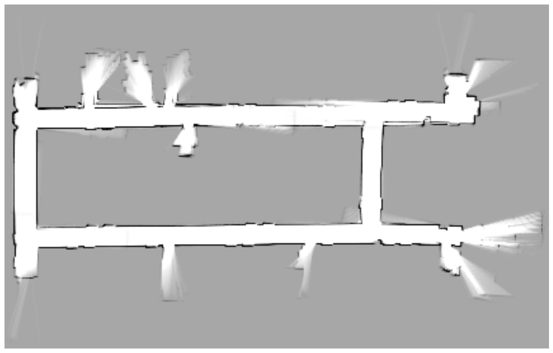
Figures 4.3, 4.4, and 4.5 show the results of sparse sensing SLAM on several data sets. (The occupancy grid images were generated using corrected trajectories from sparse sensing SLAM, but with the original dense laser data for better clarity.) Detailed statistics for these data sets are shown in Table 4.1.

USC SAL Building (Figure 4.3) This data set consists of a primary loop and several small excursions. The robot closed the loop properly in its line-based landmark map despite only a small overlap between the loop's start and end. Notice that some slight aliasing exists in the occupancy grid map: this was a common occurrence since most features have high uncertainty due to the use of odometry to augment the sparse sensing.

CMU Newell-Simon Hall (Figure 4.4) Because of the maximum sensing range of 3 m for this experiment, the fairly large initial loop (bottom) could not be closed until after the robot finished exploring the upper hallway.

Stanford Gates Building (Figure 4.5) This long data set has three large and several small loops. Figure 4.5(a) shows the uncorrected data, which exhibits significant error, particularly with respect to the robot's orientation. At several points the robot spins in place, a difficult situation


(a) Occupancy data for the raw uncorrected trajectory


(b) Occupancy data for the corrected trajectory (generated using the full laser data for clarity)

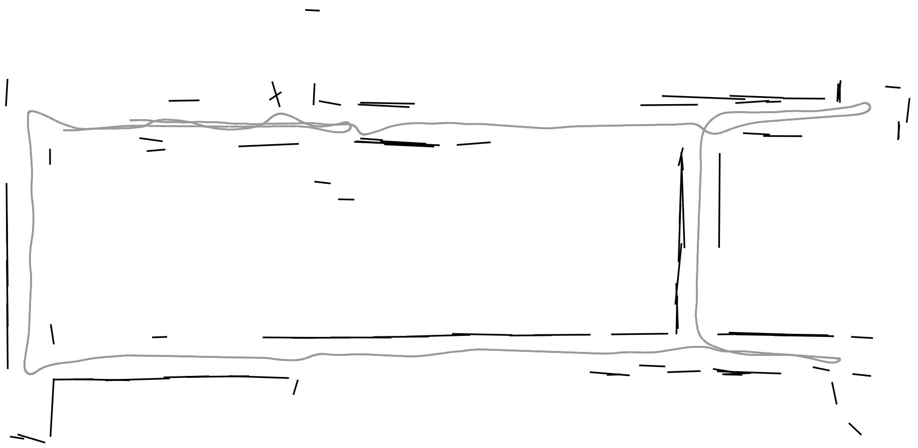

(c) Estimated line landmark map (black) and trajectory (gray)

Figure 4.3: The USC SAL Building, second floor. Data set courtesy of Andrew Howard via Radish (Howard and Roy, 2003). The landmark map shows only the original observed extent of the line features.

for sparse sensing because rotation dramatically increases the pose uncertainty and decreases the scan density due to high rotational velocities. Note that some landmarks are spurious, a result of poor data association due to large uncertainty. No detection of spurious landmarks was implemented, so these landmarks remained in the map. Again, although there is some aliasing in the results, the environment's structure and the robot's trajectory were properly recovered.

4.7.2 Discussion

These results show that performing SLAM in large indoor environments is feasible even with minimal sensing. All of our tests used only five sensors with restricted range, but even large loops were closed correctly. However, there are tradeoffs to using such limited sensing:

- More particles are required since the parameters of landmarks are more uncertain due to the use of odometry to augment sensing.
- The success of SLAM is sensitive to the amount of pose uncertainty accumulated during a multiscan.
- The size of multiscans (m) is a parameter that must be determined, either by selecting a constant or computing a value based on pose uncertainty. Choosing m is a complex problem

given all of the factors — error models, environment complexity, robot behaviors, etc. — that affect SLAM.

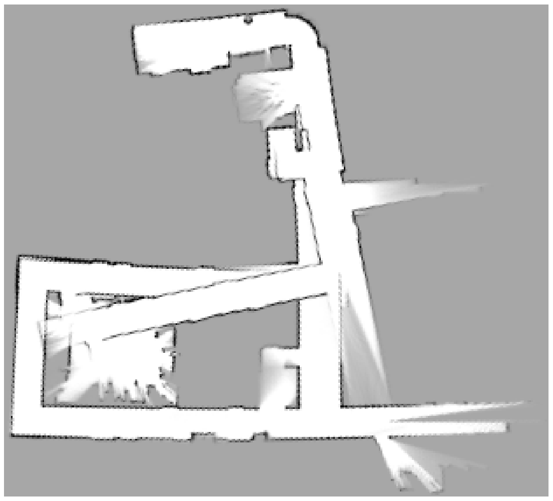
- Computationally motivated approximations (such as those made in extracting features and computing the innovation covariance) can lead to poor data association, as exhibited by the existence of spurious landmarks.

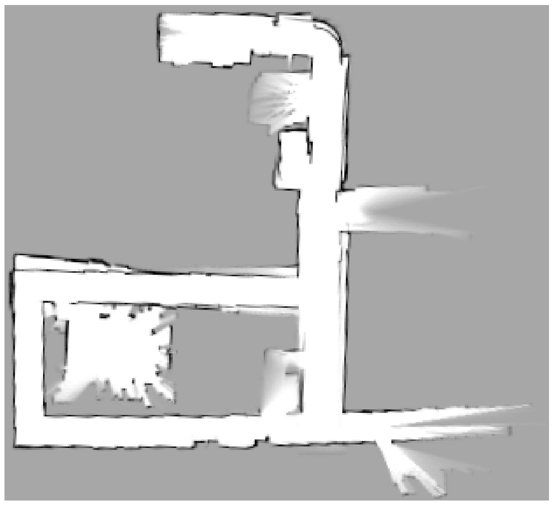
In Chapter 5 we describe a method for incorporating prior knowledge about the environment that alleviates many of these concerns. Chapter 6 introduces new sampling techniques for particle filters that incorporate "future" information to draw better pose samples.

4.8 Summary

In this chapter, we have presented a strategy for implementing particle filtering SLAM for a robot with very sparse sensing. Rather than performing feature extraction on every frame of scan data, we group the data into "multiscans" consisting of several consecutive frames of data. Feature extraction, data association, and state updates are then performed based on multiscan data as if it were data from a single scan. We formally specified a system model for this approach that maintains multiscan information separately from the system state, allowing the efficient application of particle filtering methods for SLAM. We then discussed properties of the innovation covariance for features extracted from multiscans, and presented a simplified representation for which the computation costs are significantly reduced. We also described simple approaches for extracting line features from multiscan data. Finally, we described an extension to enable an accuracy/efficiency tradeoff in feature extraction from multiscan data by clustering similar trajectories.

In our experiments using measurements from only five one-dimensional range sensors, the robot was able to close loops and recover the correct map and trajectory in several large real environments. While the experiments uncovered several tradeoffs to using limited sensing, the success of our approach with real data shows that it is possible to implement SLAM efficiently with sparse sensors.


(a) Occupancy data for the raw uncorrected trajectory


(b) Occupancy data for the corrected trajectory (generated using the full laser data for clarity)

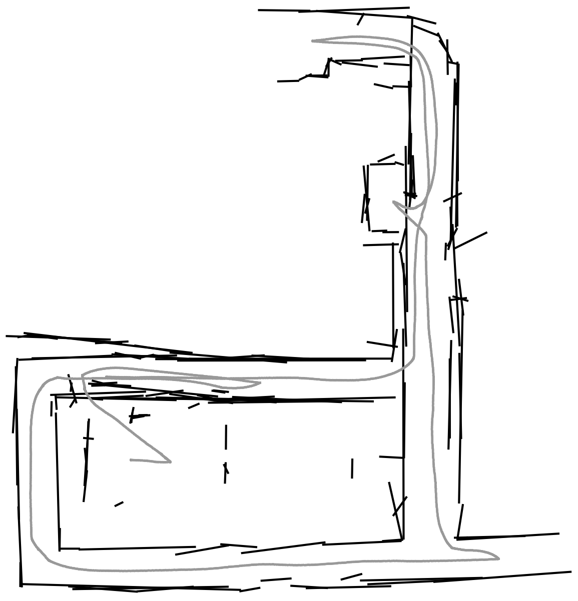
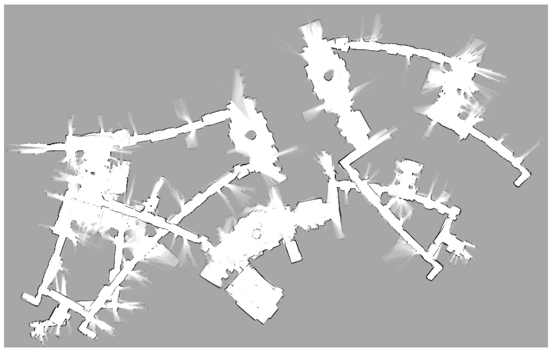
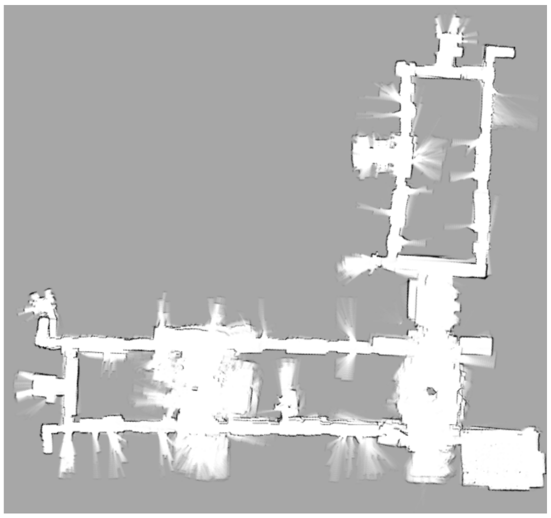

(c) Estimated line landmark map (black) and trajectory (gray)

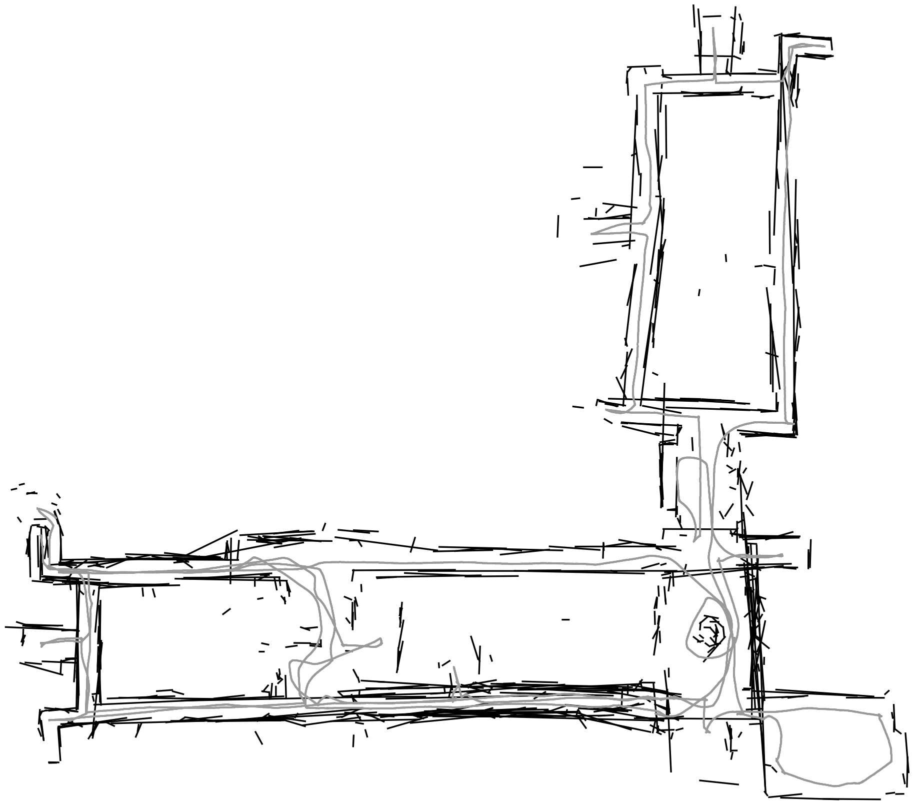
Figure 4.4: A partial map of Newell-Simon Hall Level A at CMU. Data set courtesy of Nicholas Roy via Radish (Howard and Roy, 2003).



(a) Occupancy data for the raw uncorrected trajectory (not to scale with the corrected data)



(b) Occupancy data for the corrected trajectory (generated using the full laser data for clarity)



(c) Estimated line landmark map (black) and trajectory (gray)

Figure 4.5: The Gates Building at Stanford, first floor. Data set courtesy of Brian Gerkey via Radish (Howard and Roy, 2003).

5 INCORPORATING RELATIVE MAP CONSTRAINTS

Most algorithms for simultaneous localization and mapping (SLAM) do not incorporate prior knowledge of structural or geometrical characteristics of the environment. In some cases, such information is readily available and making some assumptions is reasonable. For example, one can often assume that many walls in an indoor environment are rectilinear. Intuitively, exploiting prior knowledge should lead to reduced uncertainty about the state of the robot and the world — a desirable property for robots with limited or even powerful sensing.

In this chapter of the thesis, we develop a SLAM algorithm that incorporates prior knowledge of relative constraints between landmarks. We describe a "Rao-Blackwellized constraint filter" that infers applicable constraints and efficiently enforces them in a particle filtering framework. We have implemented our approach with rectilinearity constraints. Results from simulated and real-world experiments show the use of constraints leads to consistency improvements and a reduction in the number of particles needed to build maps.

5.1 Overview

In many cases structural or geometrical assumptions can be represented as information about relative constraints between landmarks in a robot's map, which can be used in inference to determine which landmarks are constrained and the parameters of the constraints. In the rectilinearity example, such a formulation can be used to constrain the walls of a room separately from, say, the boundary of a differently-aligned obstacle in the center of the room.

Given relative constraints between landmarks, they must be enforced. Some previous work has enforced constraints on maps represented using an extended Kalman filter (EKF) (Durrant-Whyte, 1988; Wen and Durrant-Whyte, 1992; Rodriguez-Losada et al., 2006). In this chapter, we develop techniques to instead enforce constraints in maps represented by a Rao-Blackwellized particle filter (RBPF). The major difficulty is that RBPF SLAM relies on the conditional independence of landmark estimates given a robot's pose history, but relative constraints introduce correlation between landmarks. (See Appendix B or the previous chapter for more background on

$\mathbf{c}_{i,j}$	pairwise constraint parameters for $\mathbf{x}_i^m, \mathbf{x}_j^m$	$\mathbf{x}_i^m, \mathbf{P}_i$	ith landmark state/cov.	$L_i \in \mathcal{L}$	ith superlandmark
$g_i(\mathbf{c}_{j,i}; \mathbf{x}_{j,\rho}^m)$	map $\mathbf{x}_{j,\rho}^m \to \mathbf{x}_{i,\rho}^m$	$\mathbf{x}_{i,\rho}^m$	constrained vars. of ith landmark	$\boldsymbol{\beta}_i, \boldsymbol{\Lambda}_i$	"backup" state/cov.
$h_i(\mathbf{c}_{j,i}; \mathbf{x}_{i,\rho}^m)$	map $\mathbf{x}_{i,\rho}^m \to \mathbf{x}_{j,\rho}^m$	$\mathbf{x}_{i,\bar{\rho}}^m$	unconstrained vars. of ith landmark	$\mathbf{Z}_i, \mathbf{Q}_i$	accumulated measurements/cov.

Notation for this chapter

particle filtering SLAM.)

The approach presented in this chapter exploits a property similar to that used in the standard SLAM Rao-Blackwellization: conditioned on values of constrained state variables, unconstrained state variables are independent. We use this fact to incorporate per-particle constraint enforcement into RBPF SLAM. We have also developed a technique to address complications which arise when initializing a constraint between groups of landmarks that are already separately constrained; the technique efficiently recomputes conditional estimates of unconstrained variables when modifying the values of constrained variables.

Incorporating constraints can have a profound effect on the computation required to build maps. A motivating case is the problem of mapping with sparse sensing. The previous chapter of this thesis showed that particle filtering SLAM is possible with limited sensors such as small arrays of infrared rangefinders, but that many particles are required due to increased measurement uncertainty. By extending sparse sensing SLAM to incorporate constraints, an order-of-magnitude reduction in the number of particles can be achieved.

The rest of this chapter proceeds as follows. We first discuss previous work on constrained SLAM. Then, in Section 5.2, we discuss the assumption of unstructured environments made by most SLAM algorithms. In Section 5.3 we formalize the idea of SLAM with relative constraints and describe a simple but infeasible approach. We then introduce the Rao-Blackwellized constraint filter: Section 5.4 describes an RBPF-based algorithm for enforcing constraints, and Section 5.5 incorporates inference of constraints. Finally, in Section 5.6 we describe the results of simulated and real-world experiments with a rectilinearity constraint.

5.1.1 Related work

Most work on SLAM focuses on building maps using very little prior information about the environment, aside from assumptions made in feature extraction and data association. See Chapter 2 for a reasonably thorough coverage of much of the state-of-the art in unconstrained SLAM.

The problem of inferring when constraints should be applied to a map is largely unexplored. Rodriguez-Losada et al. (2006) employ a simple thresholding approach to determine which of several types of constraints should be applied.

On the other hand, several researchers have studied the problem of enforcing *a priori* known constraints in SLAM. In particular, Durrant-Whyte (1988) and Wen and Durrant-Whyte (1992) have enforced constraints in EKF-based SLAM by treating the constraints as zero-uncertainty measurements. A similar approach was also described in the seminal SLAM work of Smith et al. (1990). More recently, Csorba and Durrant-Whyte (1997); Newman (1999); and Deans and Hebert (2000b) have built maps where the state consists of relationships between landmarks; they apply constraints on the relationships to enforce map consistency. From a consistent relative map an absolute map can be estimated.

Finally, others have studied general constrained state estimation using the EKF. Simon and Chia (2002) derive Kalman updates for linear equality constraints (discussed in detail in Section 5.3.1) that are equivalent to projecting the unconstrained state onto the constraint surface. This approach is extended in (Simon and Simon, 2003) to deal with linear inequality constraints.

5.2 Structured environments

In this chapter, like the last, we consider SLAM with landmark based maps. Figure 5.1 depicts a Bayesian network encoding the usual SLAM assumptions. (For a detailed discussion of the different variables in SLAM, see Appendix B. Briefly, $\mathbf{x}_t^r$ is the robot pose at time t, $\mathbf{x}^m$ is the map, and $\mathbf{u}_t$, $\mathbf{z}_t$, and $\mathbf{n}_t$ are the control input, observations, and correspondences at time t, respectively.) Most importantly, particle filtering SLAM relies on the assumption that the environment is *unstructured*, i.e., that landmarks are randomly and independently distributed in the environment. Under this assumption (and the additional assumption that the environment is static), landmarks are conditionally independent given the robot's trajectory, since correlation between landmark variables arises only through robot pose uncertainty (Murphy, 2000). In Figure 5.1, the high-

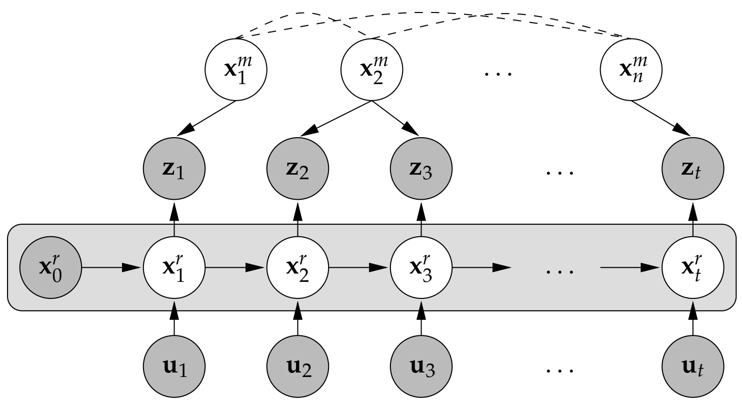


Figure 5.1: A Bayes network showing common SLAM model assumptions. Input variables are represented by shaded nodes; the objective of SLAM is to estimate values for the unshaded nodes. Arcs indicate causality or correlation between variables. (Correspondence variables $\{\mathbf{n}_{1:t}\}$ are omitted for clarity — observations are connected directly to the corresponding landmarks.) Correlations between landmarks due to structure in the environment (dashed arcs) are typically ignored in SLAM.

lighted variables (the robot's trajectory) *d-separate* the landmark variables. This d-separation is critical to the application of the particle filter in SLAM.

The structured environment assumption, however, is often not met, as in the case of indoor environments where landmarks are placed methodically. Thus, some correlation exists between landmarks, due not to the robot's pose uncertainty, but rather to the *structure* in the environment. (This is represented by the dashed arcs in Figure 5.1.)

Correlation between landmarks can arise in many ways, making it difficult to include in the SLAM model. In this chapter of the thesis, we assume that structure in the environment takes on one of a few forms — i.e., that the space of possible (structured) landmark relationships is small and discrete. When this is the case, the model shown in Figure 5.2 can be used. Here, arcs indicating correlation between landmarks are parameterized. The parameters $\mathbf{c}_{i,j}$ indicate the constraints (or lack thereof) between landmarks $\mathbf{x}_i^m$ and $\mathbf{x}_j^m$. We perform inference on the constraint parameter space, and then enforce the constraints. In this paper we focus on the pairwise case, but more complicated relationships can in principle be exploited.

5.3 SLAM with relative constraints

We begin by addressing the issue of efficiently *enforcing* known relative constraints. Parallel to this is the problem of *merging* constraints when new relationships are found between separate groups of already constrained landmarks.

Throughout the rest of this chapter we omit time indices for clarity. Variables are vectors unless otherwise noted. We use $\mathbf{P}_i^m$ to represent the covariance of the landmark estimate $\mathbf{x}_i^m$. We assume that measurements of a landmark are in the parameter space of the landmark (i.e., measurements are of the landmark state). Measurements that do not meet this condition can easily be transformed. Finally, while we present our formulation for a single constraint, the approach can be applied in parallel to several types of constraints.

5.3.1 The superlandmark filter

Recall the Rao-Blackwellized particle filter for SLAM (derived in Appendix B):

$$p(\mathbf{x}_{1:t}^r, \mathbf{x}^m | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) = p(\mathbf{x}_{1:t}^r | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \prod_{i=1}^{n} p(\mathbf{x}_i^m | \mathbf{x}_{1:t}^r, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \tag{5.1}$$

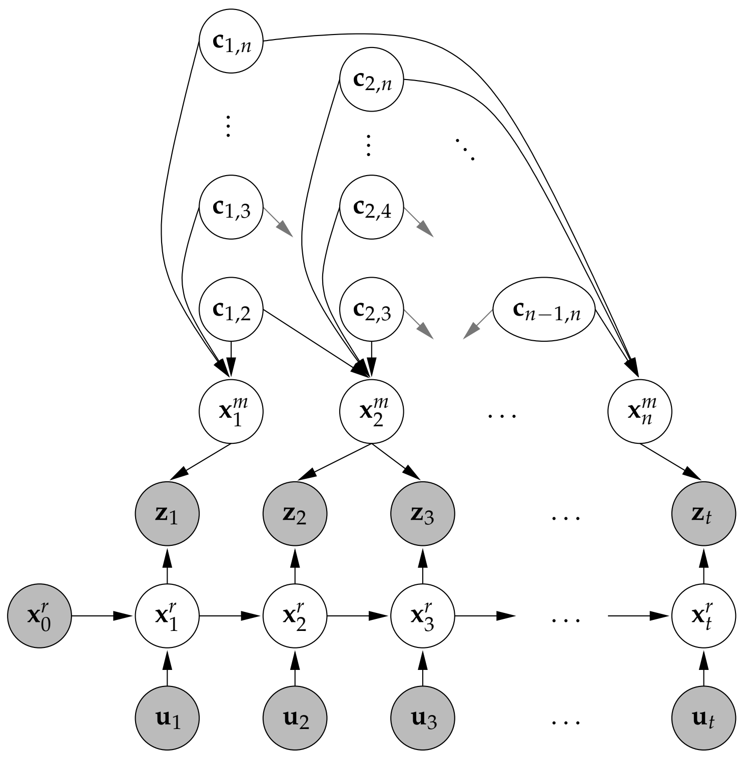


Figure 5.2: Bayes network for a SLAM model that incorporates pairwise constraints between landmarks, parameterized by the variables $\mathbf{c}_{i,j}$. Inference in the space of relationship parameters can be used to determine correlations between landmark parameters; relative constraints on the landmarks enforce inferred relationships.

As discussed above, RBPF relies on the independence of landmark estimates when conditioned on the robot's trajectory. There is an immediate problem with SLAM when the environment is structured: landmark correlations lead to interdependencies that break the factorization utilized in Equation 5.1, which assumes correlation arises only through robot pose uncertainty. We first describe a simple (but ultimately impractical) approach to deal with the correlation, which leads to an improved technique in Section 5.4. Note that the RBPF factorization still holds for unconstrained landmarks; we rewrite the filter, grouping constrained landmarks. Formally, partition the map into groups:

$$\mathcal{L} = \Big\{ \{\mathbf{x}^m_{a_1}, \mathbf{x}^m_{a_2}, \ldots\}, \{\mathbf{x}^m_{b_1}, \mathbf{x}^m_{b_2}, \ldots\}, \{\mathbf{x}^m_c\}, \ldots \Big\} \tag{5.2}$$

Each group ("superlandmark") $L_i \in \mathcal{L}$ contains landmarks constrained with respect to each other; correlation arises only among landmarks in the same group. We immediately have the following version of the RBPF SLAM filter:

$$p(\mathbf{x}^r_{1:t}, \mathbf{x}^m | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) = p(\mathbf{x}^r_{1:t} | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \prod_{i=1}^{|\mathcal{L}|} p(L_i | \mathbf{x}^r_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \tag{5.3}$$

We can still apply a particle filter to estimate the robot's trajectory. Each superlandmark is estimated using an EKF, which accounts for correlation due to constraints since it maintains full covariance information.

There are several ways to enforce constraints on a superlandmark. One approach is to treat the constraints as zero-uncertainty measurements of the constrained landmarks (Durrant-Whyte, 1988; Wen and Durrant-Whyte, 1992; Rodriguez-Losada et al., 2006). An alternative is to directly incorporate constrained estimation into the Kalman filter. Simon and Chia (2002) have derived a version of the EKF that accounts for equality constraints of the form

$$\mathbf{D}L_i = \mathbf{d} \tag{5.4}$$

where L_i represents the superlandmark state with k variables, $\mathbf{D}$ is an $s \times k$ constant matrix of full rank, and $\mathbf{d}$ is a $s \times 1$ vector; together they encode s constraints. In their approach, the unconstrained EKF estimate is computed and then repaired to account for the constraints. Given the unconstrained state L_i and covariance matrix $\mathbf{P}_{L_i}$, the constrained state and covariance are computed as follows (see (Simon and Chia, 2002) for the derivation):

$$\tilde{L}_i \leftarrow L_i - \mathbf{P}^m_{L_i}\mathbf{D}^T(\mathbf{D}\mathbf{P}^m_{L_i}\mathbf{D}^T)^{-1}(\mathbf{D}L_i - \mathbf{d}) \tag{5.5}$$

$$\tilde{\mathbf{P}}^m_{L_i} \leftarrow \mathbf{P}^m_{L_i} - \mathbf{P}^m_{L_i}\mathbf{D}^T(\mathbf{D}\mathbf{P}^m_{L_i}\mathbf{D}^T)^{-1}\mathbf{D}\mathbf{P}^m_{L_i} \tag{5.6}$$

i.e., the unconstrained estimate is projected onto the constraint surface.

If a constraint arises between two superlandmarks they are easily merged:

$$L_{ij} \leftarrow \begin{bmatrix} L_i \\ L_j \end{bmatrix} \quad , \quad \mathbf{P}^m_{ij} \leftarrow \begin{bmatrix} \mathbf{P}^m_i & \mathbf{P}^m_i \left(\frac{\partial L_j}{\partial L_i}\right)^T \\ \left(\frac{\partial L_j}{\partial L_i}\right)\mathbf{P}^m_i & \mathbf{P}^m_j \end{bmatrix} \tag{5.7}$$

Unfortunately, the superlandmark filter is too expensive unless the size of superlandmarks (i.e., the number of constrained landmarks in a group) can be bounded by a constant. In the worst case the environment is highly constrained and, in the extreme, the map consists of a single superlandmark. EKF updates for SLAM take at least $O(k^2)$ time and constraint enforcement using Equations 5.5 and 5.6 requires $O(k^3)$ time for a superlandmark of size k. If $k \to n$ (the size of the entire map) and the particle filter has N particles, the superlandmark filter requires $O(Nn^3)$ time for a single update. We thus require a better solution.

5.3.2 Reduced state formulation

A simple improvement can be obtained by noting that maintaining the full state and covariance for each landmark in a superlandmark is unnecessary. Constrained state variables are redundant: given the value of the variables from one "representative" landmark, the values for the remaining landmarks in a superlandmark are determined. In the rectilinearity example, with landmarks represented as lines parameterized by distance r and angle θ to the origin, a full superlandmark state vector has the form:

$$[r_1 \;\; \theta_1 \;\; r_2 \;\; \theta_2 \;\; \ldots \;\; r_k \;\; \theta_k]^T \tag{5.8}$$

If the $\{\theta_i\}$ are constrained the state can be rewritten as:

$$[r_1 \;\; \theta_1 \;\; r_2 \;\; g_2(\mathbf{c}_{1,2};\theta_1) \;\; \ldots \;\; r_k \;\; g_k(\mathbf{c}_{1,k};\theta_1)]^T \tag{5.9}$$

Thus, filtering of the superlandmark need only be done over the reduced state:

$$[r_1 \;\; r_2 \;\; \ldots \;\; r_k \;\; \theta_1]^T \tag{5.10}$$

The function $g_i(\mathbf{c}_{j,i};\mathbf{x}^m_{j,\rho})$ with parameters $\mathbf{c}_{j,i}$ maps the constrained variables $\mathbf{x}^m_{j,\rho}$ of the representative landmark $\mathbf{x}^m_j$ to values for $\mathbf{x}^m_{i,\rho}$. In the rectilinearity case, $\mathbf{c}_{j,i} \in \{0, 90, 180, 270\}$ and:

$$g_i(\mathbf{c}_{j,i};\theta_j) = \theta_j - \mathbf{c}_{j,i} \tag{5.11}$$

We assume the constraints are invertible: the function $h_i(\mathbf{c}_{j,i}; \mathbf{x}^m_{i,\rho})$ represents the reverse mapping, e.g.:

$$h_i(\mathbf{c}_{j,i}; \theta_i) = \theta_i + \mathbf{c}_{j,i} \tag{5.12}$$

We sometimes refer to the unconstrained variables of landmark $\mathbf{x}^m_i$ as $\mathbf{x}^m_{i,\bar{\rho}}$.

5.4 Rao-Blackwellized constraint filter

From the reduced state formulation we see it is easy to separate the map state into constrained variables $\mathbf{x}^{m,\mathrm{c}} = \{\mathbf{x}^m_{1,\rho}, \ldots, \mathbf{x}^m_{n,\rho}\}$, and unconstrained variables $\mathbf{x}^{m,\mathrm{f}} = \{\mathbf{x}^m_{1,\bar{\rho}}, \ldots, \mathbf{x}^m_{n,\bar{\rho}}\}$. By the same reasoning behind Equation 5.1, we factor the SLAM filter as follows:

$$p(\mathbf{x}^r_{1:t}, \mathbf{x}^m | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) = p(\mathbf{x}^r_{1:t}, \mathbf{x}^{m,\mathrm{c}} | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \prod_{i=1}^{|\mathbf{x}^{m,\mathrm{f}}|} p(\mathbf{x}^m_{i,\bar{\rho}} | \mathbf{x}^r_{1:t}, \mathbf{x}^{m,\mathrm{c}}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \tag{5.13}$$

In other words, conditioned on *both* the robot's trajectory and the values of all constrained variables, free variables of separate landmarks are independent.

Equation 5.13 suggests that we can use a particle filter to estimate both the robot trajectory and the values of the constrained variables. We can then use separate small filters to estimate the unconstrained variables conditioned on sampled values of the constrained variables. The estimation of representative values for the constrained variables is accounted for in the particle filter resampling process, where particles are weighted by data association likelihood.

There are two issues to resolve:

- We should use all available information in selecting (per-particle) values of constrained variables on which to condition constrained landmarks.

- When initializing a new constraint between already separately constrained groups of landmarks, the constraints need to be merged and a single set of representative values for the constrained variables must be selected.

5.4.1 Particlization of landmark variables

We first discuss initialization of constraints between previously unconstrained landmarks. Given a set $\mathcal{R} = \{\mathbf{x}^m_1, \mathbf{x}^m_2, \ldots, \mathbf{x}^m_k\}$ of landmarks to be constrained, along with constraint parameters $\mathbf{c}_{1,i}$ for each $\mathbf{x}^m_i \in \mathcal{R}, i = 2 \ldots k$ (i.e., with $\mathbf{x}^m_1$ as the "representative" landmark — see Section 5.3.2), we form a superlandmark from $\mathcal{R}$. Then, we perform a *particlization procedure*, sampling the constrained variables from the reduced state of the superlandmark. Conditioning of the unconstrained variables of every landmark in the superlandmark is performed using the sampled values. We are left with an EKF for each landmark that estimates only the values of unconstrained state variables.

In selecting values of the constrained variables on which to condition, we should take into account all available information, i.e., the estimates of the constrained variables from each landmark. We compute the maximum likelihood estimate of the constrained variables:

$$\mathbf{P}^m_{\hat{\rho}} \leftarrow \left(\sum_{\mathbf{x}^m_j \in \mathcal{R}} (\mathbf{P}^m_{j,\rho})^{-1} \right)^{-1} \quad , \quad \hat{\rho} \leftarrow (\mathbf{P}^m_{\hat{\rho}})^{-1} \left(\sum_{\mathbf{x}^m_j \in \mathcal{R}} h_j(\mathbf{c}_{1,j}; \mathbf{x}^m_{j,\rho}) (\mathbf{P}^m_{j,\rho})^{-1} \right) \tag{5.14}$$

To choose values for ρ, we can either sample, e.g., according to $\mathcal{N}\left(\hat{\rho}, \mathbf{P}^m_{\hat{\rho}}\right)$; or we can simply pick $\hat{\rho}$, which is the approach we take in our implementation.

Once values of constrained variables are selected, we condition the unconstrained variables on the selected values. To condition $\mathbf{x}^m_i$ with covariance $\mathbf{P}^m_i$ on values for $\mathbf{x}^m_{i,\rho}$, we first partition

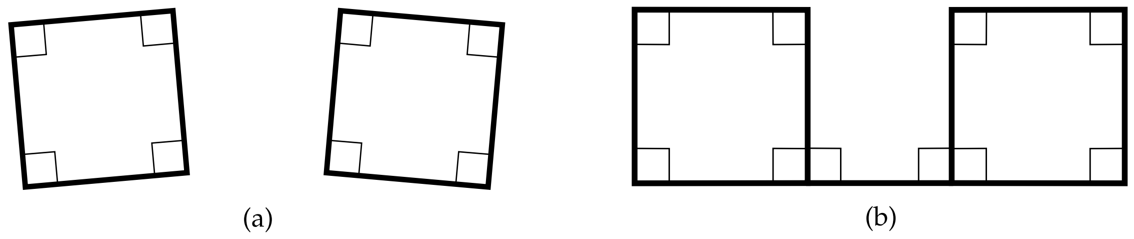


Figure 5.3: Merging groups of constrained landmarks. (a) Two constrained groups of landmarks. (b) After finding a new landmark constrained with respect to both groups, all landmarks are constrained together.

the state and covariance:

$$\mathbf{x}_i^m = [\mathbf{x}_{i,\bar{\rho}}^m \;\; \mathbf{x}_{i,\rho}^m]^T \quad , \quad \mathbf{P}_i^m = \begin{bmatrix} \mathbf{P}_{i,\bar{\rho}}^m & \mathbf{P}_{i,\bar{\rho}\rho}^m \\ \mathbf{P}_{i,\rho\bar{\rho}}^m & \mathbf{P}_{i,\rho}^m \end{bmatrix} \tag{5.15}$$

Then given $\mathbf{x}_{i,\rho}^m = g_i(\mathbf{c}_{1,i}; \hat{\rho})$ and since landmark state is estimated by an EKF, the standard procedure for conditioning the Normal distribution yields:

$$\tilde{\mathbf{x}}_{i,\bar{\rho}}^m \quad \leftarrow \quad \mathbf{x}_{i,\bar{\rho}}^m + \mathbf{P}_{i,\bar{\rho}\rho}^m (\mathbf{P}_{i,\rho}^m)^{-1} (g_i(\mathbf{c}_{1,i}; \hat{\rho}) - \mathbf{x}_{i,\rho}^m) \tag{5.16}$$

$$\tilde{\mathbf{P}}_{i,\bar{\rho}}^m \quad \leftarrow \quad \mathbf{P}_{i,\bar{\rho}}^m - \mathbf{P}_{i,\bar{\rho}\rho}^m (\mathbf{P}_{i,\rho}^m)^{-1} (\mathbf{P}_{i,\bar{\rho}\rho}^m)^T \tag{5.17}$$

For purposes of data association it is convenient to retain the full state and covariance, in which case $\tilde{\mathbf{x}}_{i,\rho}^m = g_i(\mathbf{c}_{1,i}; \hat{\rho})$ and $\tilde{\mathbf{P}}_{i,\rho}^m = \tilde{\mathbf{P}}_{i,\bar{\rho}\rho}^m = \tilde{\mathbf{P}}_{i,\rho\bar{\rho}}^m = [\mathbf{0}]$.

5.4.2 Reconditioning

Particlization is straightforward if none of the landmarks is already constrained. This is not the case when a new landmark is added to a superlandmark or when merging several constrained superlandmarks. Since the values of unconstrained state variables are already conditioned on values of the constrained variables, we cannot change constrained variables without invalidating the conditioning. Such a situation is depicted in Figure 5.3.

One solution is to "rewind" the process to the point when the landmarks were first constrained and then "replay" all of the measurements of the landmarks, conditioning on the new values of the constrained variables. This is clearly infeasible. However, we can achieve an equivalent result efficiently because the order in which measurements are applied is irrelevant. Applying m measurements to the landmark state is equivalent to merging $m+1$ Gaussians. Thus, we can "accumulate" all of the measurements in a single Gaussian and apply this instead, in unit time.

From this, we obtain the following reconditioning approach:

1. Upon first constraining a landmark $\mathbf{x}_i^m$, store its pre-particlization unconstrained state $\boldsymbol{\beta}_i = \mathbf{x}_i^m, \boldsymbol{\Lambda}_i = \mathbf{P}_i^m$, initialize the "measurement accumulator" $\mathbf{Z}_i = [\mathbf{0}], \mathbf{Q}_i = [\infty]$, and particlize the landmark.

2. For a measurement $\mathbf{z}$ with covariance $\mathbf{R}$ of the constrained landmark update both the conditional state and the measurement accumulator:

$$\mathbf{x}_i^m \quad \leftarrow \quad \mathbf{x}_i^m + \mathbf{P}_i^m (\mathbf{P}_i^m + \mathbf{R})^{-1} (\mathbf{z} - \mathbf{x}_i^m) \tag{5.18}$$

$$\mathbf{P}_i^m \quad \leftarrow \quad \mathbf{P}_i^m - \mathbf{P}_i^m (\mathbf{P}_i^m + \mathbf{R})^{-1} (\mathbf{P}_i^m)^T \tag{5.19}$$

$$\mathbf{Z}_i \quad \leftarrow \quad \mathbf{Z}_i + \mathbf{Q}_i (\mathbf{Q}_i + \mathbf{R})^{-1} (\mathbf{z} - \mathbf{Z}_i) \tag{5.20}$$

$$\mathbf{Q}_i \quad \leftarrow \quad \mathbf{Q}_i - \mathbf{Q}_i (\mathbf{Q}_i + \mathbf{R})^{-1} \mathbf{Q}_i^T \tag{5.21}$$

3. When instantiating a new constraint on $\mathbf{x}_i^m$, recondition $\mathbf{x}_i^m$ on the new constrained variable values by rewinding the landmark state ($\mathbf{x}_i^m = \boldsymbol{\beta}_i$, $\mathbf{P}_i^m = \boldsymbol{\Lambda}_i$), computing the conditional distribution $\tilde{\mathbf{x}}_i^m, \tilde{\mathbf{P}}_i^m$ of the state (Equations 5.16-5.17), and replaying the measurements since particlization with:

$$\mathbf{x}_i^m \quad \leftarrow \quad \tilde{\mathbf{x}}_i^m + \tilde{\mathbf{P}}_i^m(\tilde{\mathbf{P}}_i^m + \mathbf{Q}_i)^{-1}(\mathbf{Z}_i - \tilde{\mathbf{x}}_i^m) \tag{5.22}$$

$$\mathbf{P}_i^m \quad \leftarrow \quad \tilde{\mathbf{P}}_i^m - \tilde{\mathbf{P}}_i^m(\tilde{\mathbf{P}}_i^m + \mathbf{Q}_i)^{-1}(\tilde{\mathbf{P}}_i^m)^T \tag{5.23}$$

The reconditioning technique can be extended to handle multiple types of constraints simultaneously by separately storing the pre-particlization state and accumulated measurements for each constraint. Only completely unconstrained state variables should be stored at constraint initialization, and only the measurements of those variables need be accumulated.

5.4.3 Discussion

A potential issue with our approach is that reconditioning neither re-evaluates data associations nor modifies the trajectory of a particle. In practice we have observed that the effect on map estimation is negligible.

Computationally, the constrained RBPF approach is a significant improvement over the superlandmark filter, requiring only $O(Nn)$ time per update.* At first it appears that more particles may be necessary since representative values of constrained variables are now estimated by the particle filter. However, incorporating constraints often leads to a significant reduction in required particles by reducing the degrees of freedom in the map. In a highly constrained environment, particles only need to filter a few constrained variables using the reduced state, and the EKFs for unconstrained variables are smaller since they filter only over the unconstrained state. By applying strong constraint priors where appropriate, the number of particles required to build maps is often reduced by an order of magnitude, as can be seen in Section 5.6.

5.4.4 Inequality constraints

So far we have only considered equality constraints, whereas many useful constraints are inequalities. For example, we might specify a prior on corridor width: two parallel walls should be at least a certain distance apart. In (Simon and Simon, 2003), the authors apply inequality constraints to an EKF using an active set approach. At each time step, the applicable constraints are tested. If a required inequality is violated, an equality constraint is applied, projecting the unconstrained state onto the boundary of the constraint region.

While this approach appears to have some potential problems (e.g., it ignores the landmark PDF over the unconstrained half-hyperplane in parameter space), a similar technique can be incorporated into the Rao-Blackwellized constraint filter. After updating a landmark, applicable inequality constraints are tested. Constraints that are violated are enforced using the techniques described in Section 5.4. The unconstrained state is accessible via the measurement accumulator, so if the inequality is later satisfied, the parameters can be “de-particlized” by switching back to the unconstrained estimate.

5.5 Inference of constraints

We now address the problem of deducing the relationships between landmarks, i.e., deciding when a constraint should be applied. A simple approach is to just examine the unconstrained landmark estimates. In the rectilinearity case, we can easily compute the estimated angle between two landmarks. If this angle is “close enough” to one of $0°, 90°, 180°$, or $270°$, the constraint is applied to the landmarks. (A similar approach is used by Rodriguez-Losada et al. (2006).) However, this technique ignores the confidence in the landmark estimates.

*We note that while the data structures that enable $O(N \log n)$ time updates for FastSLAM (Montemerlo, 2003) can still be applied, they do not improve the complexity of constrained RBPF since the reconditioning step is worst-case linear in n.

We instead compute a PMF over the space $\mathcal{C}$ of pairwise constraint parameters; the PMF incorporates the landmark PDFs. In the rectilinearity example, $\mathcal{C} = \{0, 90, 180, 270, \star\}$, where $\star$ is used to indicate that landmarks are unconstrained. Given a PMF over $\mathcal{C}$, we sample constraint parameters for each particle to do inference of constraints. Particles with incorrectly constrained landmarks will yield poor data associations and be resampled.

We compute the PMF of the "relationship" of landmarks $\mathbf{x}_i^m$ and $\mathbf{x}_j^m$ using:

$$p(\mathbf{c}_{i,j}) = \int p(\mathbf{x}_{i,\rho}^m) \int_{h_j(\mathbf{c}_{i,j};\mathbf{x}_{j,\rho}^m)-\delta}^{h_j(\mathbf{c}_{i,j};\mathbf{x}_{j,\rho}^m)+\delta} p(\mathbf{x}_{j,\rho}^m)\, d\mathbf{x}_{j,\rho}^m\, d\mathbf{x}_{i,\rho}^m \tag{5.24}$$

for all $\mathbf{c}_{i,j} \in \mathcal{C} \setminus \star$. Then:

$$p(\star) = 1 - \sum_{\mathbf{c}_{i,j} \in \mathcal{C} \setminus \star} p(\mathbf{c}_{i,j}) \tag{5.25}$$

The parameter δ encodes "prior information" about the environment: the larger the value of δ, the more liberally we apply constraints. A benefit of this approach is that the integrals can be computed efficiently from standard approximations to the Normal CDF since the landmarks are estimated by EKFs.

In the rectilinearity case, given orientation estimates described by the PDFs $p(\theta_i)$ and $p(\theta_j)$, for $\mathbf{c}_{i,j} \in \{0, 90, 180, 270\}$, we have:

$$p(\mathbf{c}_{i,j}) = \int_{-\infty}^{\infty} p(\theta_i) \int_{\theta_i + \mathbf{c}_{i,j} - \delta}^{\theta_i + \mathbf{c}_{i,j} + \delta} p(\theta_j)\, d\theta_j\, d\theta_i \tag{5.26}$$

which gives a valid PMF as long as $\delta \leq 45°$.

5.6 Results

We have now described the complete approach for implementing constrained RBPF SLAM. Algorithm 5.1 gives pseudocode for initializing a landmark $\mathbf{x}_{n+1}^m$ given the current set of superlandmarks $\mathcal{L}$. Algorithm 5.2 shows how to update a (possibly constrained) landmark given a measurement of its state. The algorithms simply collect the steps described in detail in Sections 5.4 and 5.5.

We have implemented the Rao-Blackwellized constraint filter for the rectilinearity constraint described earlier, on top of our algorithm for RBPF SLAM with sparse sensing as described in Chapter 4, which extracts features using data from multiple poses. Because of the sparseness of the sensor data, unconstrained SLAM typically requires many particles to deal with high uncertainty. We performed several experiments, using both simulated and real data, which show that incorporating prior knowledge and enforcing constraints leads to a significant improvement in the resulting maps and a reduction in estimation error.

5.6.1 Simulated data

We first used a simple kinematic simulator based on an RWI MagellanPro robot to collect data from a small simulated environment with two groups of rectilinear features. The goal was to test the algorithm's capability to infer the existence of constraints between landmarks. Only the five range sensors at $0°, 45°, 90°, 135°$, and $180°$ were used (i.e.,). Noise was introduced by perturbing measurements and motions in proportion to their magnitude. For a laser measurement of range r, $\sigma_r = 0.01r$; for a motion consisting of a translation d and rotation ϕ, the robot's orientation was perturbed with $\sigma_\theta = 0.03d + 0.08\phi$, and its position with $\sigma_x = \sigma_y = 0.05d$.

Figure 5.4 shows the results of RBPF SLAM with a rectilinearity prior (as described in Section 5.5, with $\delta = \frac{\pi}{10}$). The filter contained 20 particles and recovered the correct relative constraints. The edges of the the inner "box" were constrained, and the edges of the boundary were separately constrained.

Algorithm 5.1 INITIALIZE-LANDMARK($\mathbf{x}^m_{n+1}, \mathbf{P}^m_{n+1}, \mathcal{L}$)

```
1: $\boldsymbol{\beta}_{n+1} \leftarrow \mathbf{x}^m_{n+1}; \quad \boldsymbol{\Lambda}_{n+1} = \mathbf{P}^m_{n+1}$    // initialize backup state
2: $\mathbf{Z}_{n+1} \leftarrow [\mathbf{0}]; \quad \mathbf{Q}_{n+1} \leftarrow [\infty]$    // initialize measurement accumulator
3: $\mathcal{R} \leftarrow \{\}$    // initialize constraint set
4: for all $L_i \in \mathcal{L}$ do    // previously constrained groups
5:   $\mathbf{c}_{n+1,j} \sim p(\mathbf{c}_{n+1,j}), \ \forall \mathbf{x}^m_j \in L_i$    // draw constraint parameters
6:   if $\exists \mathbf{x}^m_j \in L_i$ such that $\mathbf{c}_{n+1,j} \neq \star$ then    // constrained?
7:     for all $\mathbf{x}^m_j \in L_i$ do
8:       $\mathcal{R} \leftarrow \mathcal{R} \cup \{\mathbf{x}^m_j\}$    // add $\mathbf{x}^m_j$ to constraint set
9:     end for
10:    $\mathcal{L} \leftarrow \mathcal{L} \setminus L_i$    // remove old superlandmark
11:  end if
12: end for
13: if $\mathcal{R} = \emptyset$ then
14:   return    // no constraints on $\mathbf{x}^m_{n+1}$
15: end if
16: $\mathcal{R} \leftarrow \mathcal{R} \cup \{\mathbf{x}^m_{n+1}\}$    // add new landmark to constraint set
17: $\mathcal{L} \leftarrow \mathcal{L} \cup \{\mathcal{R}\}$    // add new superlandmark
18: for all $\mathbf{x}^m_j \in \mathcal{R}$ do    // for all constrained landmarks
19:   $\hat{\mathbf{x}}^m_j \leftarrow \boldsymbol{\beta}_j + \boldsymbol{\Lambda}_j \mathbf{Q}_j^{-1}(\mathbf{Z}_j - \boldsymbol{\beta}_j)$    // compute unconstrained state estimate
20:   $\hat{\mathbf{P}}^m_j \leftarrow \boldsymbol{\Lambda}_j - \boldsymbol{\Lambda}_j \mathbf{Q}_j^{-1}(\boldsymbol{\Lambda}_j)^T$    // compute unconstrained covariance
21: end for
22: $\mathbf{P}^m_{\hat{\rho}} \leftarrow \left(\sum_{\mathbf{x}^m_j \in \mathcal{R}} (\hat{\mathbf{P}}^m_{j,\rho})^{-1}\right)^{-1}$    // covariance of ML estimate of $\rho$
23: $\hat{\rho} \leftarrow (\mathbf{P}^m_{\hat{\rho}})^{-1} \left(\sum_{\mathbf{x}^m_j \in \mathcal{R}} h_j(\mathbf{c}_{n+1,j}; \hat{\mathbf{x}}^m_{j,\rho})(\hat{\mathbf{P}}^m_{j,\rho})^{-1}\right)$    // ML estimate of $\rho$
24: for all $\mathbf{x}^m_j \in \mathcal{R}$ do    // for all constrained landmarks
25:   $\mathbf{x}^m_j \leftarrow \boldsymbol{\beta}_j; \quad \mathbf{P}^m_j \leftarrow \boldsymbol{\Lambda}_j$    // "rewind" state to pre-particlized version
26:   $\mathbf{x}^m_{j,\overline{\rho}} \leftarrow \mathbf{x}^m_{j,\overline{\rho}} + \mathbf{P}^m_{j,\overline{\rho}\rho}(\mathbf{P}^m_{j,\rho})^{-1}\left(g_j(\mathbf{c}_{n+1,j}; \hat{\rho}) - \mathbf{x}^m_{j,\rho}\right)$    // conditional mean given $\rho$
27:   $\mathbf{P}^m_{j,\overline{\rho}} \leftarrow \mathbf{P}^m_{j,\overline{\rho}} - \mathbf{P}^m_{j,\overline{\rho}\rho}(\mathbf{P}^m_{j,\rho})^{-1}(\mathbf{P}^m_{j,\overline{\rho}\rho})^T$    // conditional covariance
28:   $\mathbf{x}^m_{j,\rho} \leftarrow g_j(\mathbf{c}_{n+1,j}; \hat{\rho}); \quad \mathbf{P}^m_{j,\rho} \leftarrow [\mathbf{0}]; \quad \mathbf{P}^m_{j,\overline{\rho}\rho} \leftarrow [\mathbf{0}]$    // fix constrained variables
29:   $\mathbf{x}^m_j \leftarrow \mathbf{x}^m_j + \mathbf{P}^m_j(\mathbf{P}^m_j + \mathbf{Q}_j)^{-1}(\mathbf{Z}_j - \mathbf{x}^m_j)$    // "replay" meas. since particlization
30:   $\mathbf{P}^m_j \leftarrow \mathbf{P}^m_j - \mathbf{P}^m_j(\mathbf{P}^m_j + \mathbf{Q}_j)^{-1}(\mathbf{P}^m_j)^T$
31: end for
```

Algorithm 5.2 UPDATE-LANDMARK($\mathbf{x}^m_j, \mathbf{P}^m_j, \mathbf{z}, \mathbf{R}$)

```
1: $\mathbf{x}^m_j \leftarrow \mathbf{x}^m_j + \mathbf{P}^m_j(\mathbf{P}^m_j + \mathbf{R})^{-1}(\mathbf{z} - \mathbf{x}^m_j)$    // update state
2: $\mathbf{P}^m_j \leftarrow \mathbf{P}^m_j - \mathbf{P}^m_j(\mathbf{P}^m_j + \mathbf{R})^{-1}(\mathbf{P}^m_j)^T$    // update covariance
3: if $\exists L \in \mathcal{L}, \mathbf{x}^m_k \in L$ such that $\mathbf{x}^m_j \in L$ and $\mathbf{x}^m_j \neq \mathbf{x}^m_k$ then    // is $\mathbf{x}^m_j$ constrained?
4:   $\mathbf{Z}_j \leftarrow \mathbf{Z}_j + \mathbf{Q}_j(\mathbf{Q}_j + \mathbf{R})^{-1}(\mathbf{z} - \mathbf{Z}_j)$    // update measurement accumulator
5:   $\mathbf{Q}_j \leftarrow \mathbf{Q}_j - \mathbf{Q}_j(\mathbf{Q}_j + \mathbf{R})^{-1}\mathbf{Q}_j^T$    // update accumulator covariance
6: else    // not constrained
7:   $\boldsymbol{\beta}_j \leftarrow \mathbf{x}^m_j; \boldsymbol{\Lambda}_j \leftarrow \mathbf{P}^m_j$    // update backup state/covariance
8: end if
```

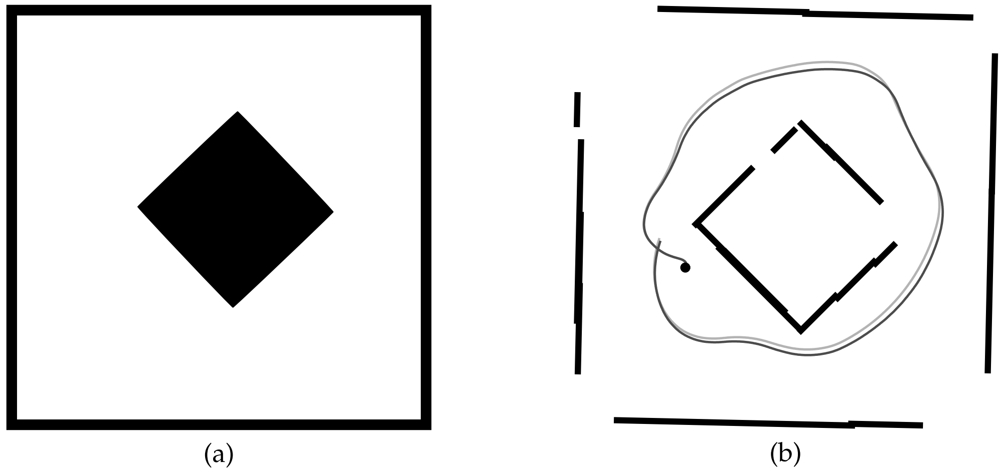


Figure 5.4: (a) Simulated environment (ground truth). (b) Results of applying constrained SLAM. The dark curved line is the trajectory estimate, the light curved line is the ground truth trajectory, and the dot is the starting pose. The landmarks on the boundary form one constrained group; those in the interior form the other.

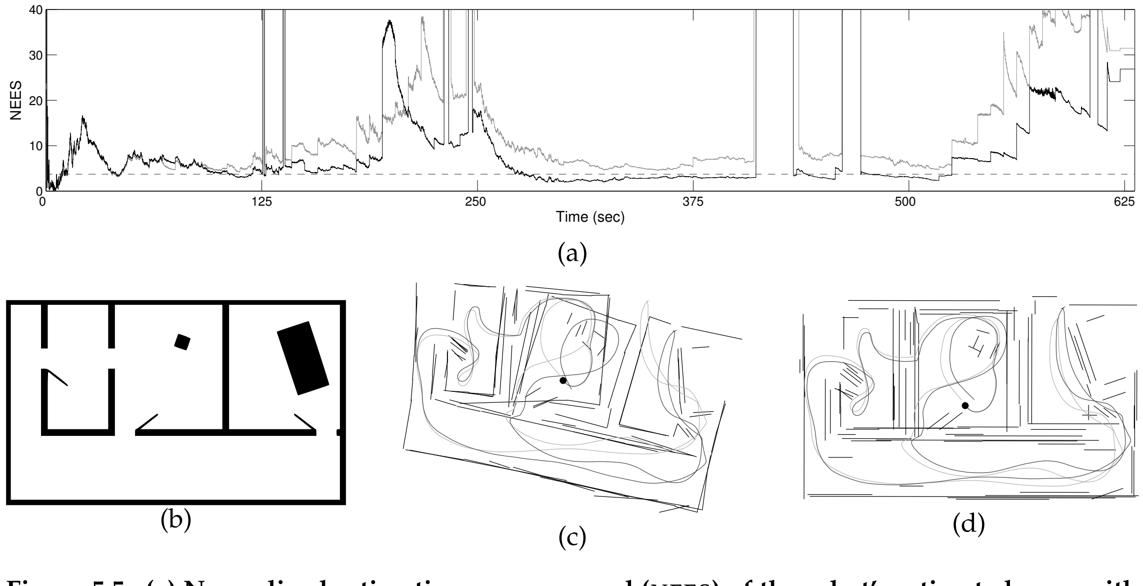


Figure 5.5: (a) Normalized estimation error squared (NEES) of the robot's estimated pose with respect to the ground truth, computed over 50 Monte Carlo trials for the environment in (b). The gray plot is the error for standard (unconstrained) RBPF SLAM. The black plot is the error for our algorithm with rectilinearity constraints. Error significantly above the dashed line indicates an optimistic (inconsistent) filter. Our approach is less optimistic. (Sharp spikes correspond to degeneracies due to resampling upon loop closure.) (c) A typical map produced by unconstrained sparse sensing SLAM. (d) A typical rectilinearity-constrained map.

	USC SAL		CMU NSH	
Dimensions	39×20	m^2	25×25	m^2
Particles (constrained)	20		40	
Particles (unconstrained)	100		600	
Avg. Runtime (constrained, 30 runs)	11.24	s	34.77	s
Avg. Runtime (unconstrained, 30 runs)	32.02	s	268.44	s
Sensing range	5	m	3	m
Path length	122	m	114	m
Num. landmarks	162		219	
Constrained groups	3		3	

Table 5.1: Constrained SLAM experiment statistics

A separate experiment compared the consistency of the rectilinearity-constrained filter and the unconstrained filter. (All other filter parameters were kept identical, including number of particles.) A filter is inconsistent if it significantly underestimates its own error. It has been shown that RBPF SLAM is generally inconsistent (Bailey et al., 2006); our experiments indicate that using prior knowledge and enforcing constraints improves (but does not guarantee) consistency. (Chapter 6 also addresses the consistency issue.)

Figure 5.5 depicts the consistency analysis. The ground truth trajectory from the simulation was used to compute the normalized estimation error squared (NEES) (Bar-Shalom et al., 2001; Bailey et al., 2006) of the robot's trajectory estimate. For ground truth pose $\mathbf{x}_t^r$ and estimate $\hat{\mathbf{x}}_t^r$ with covariance $\hat{\mathbf{P}}_t^r$ (estimated from the weighted particles assuming they are approximately normally distributed), the NEES is:

$$(\mathbf{x}_t^r - \hat{\mathbf{x}}_t^r)(\hat{\mathbf{P}}_t^r)^{-1}(\mathbf{x}_t^r - \hat{\mathbf{x}}_t^r)^T \tag{5.27}$$

For more details of how NEES can be used to examine SLAM filter consistency, see (Bailey et al., 2006). The experiment used 200 particles for each of 50 Monte Carlo trials, with a robot model similar to the previous simulation.

5.6.2 Real-world data

Our real-world experiments used data from Radish (Howard and Roy, 2003), an online repository of SLAM data sets. Most of the data sets use scanning laser rangefinders. Since our goal is to enable SLAM with limited sensing, we simply discarded most of the data in each scan, keeping only the five range measurements at $0°, 45°, 90°, 135°$, and $180°$. We also restricted the sensor range (see Table 5.1). We used the same rectilinearity prior as for the simulated examples ($\delta = \frac{\pi}{10}$).

Figure 5.6 shows the results of our algorithm for two data sets. The *USC SAL* data set consists of a primary loop and several small excursions. Most landmarks are constrained, in three separate groups. For the *CMU NSH* experiment, the maximum sensing range was restricted to 3 m, so the large initial loop (bottom) could not be closed until the robot finished exploring the upper hallway. Aside from several landmarks in the curved portion of the upper hallway, most landmarks are constrained.

Table 5.1 gives mapping statistics. Also included is the number of particles required to successfully build an unconstrained map, along with running times for comparison. (The complete results for unconstrained sparse sensing SLAM can be found in the preceding chapter.) All tests were performed on a P4-1.7 GHz computer with 1 GB RAM. Incorporating constraints enables mapping with many fewer particles — about the same number as needed by many unconstrained SLAM algorithms that use full laser rangefinder information. This leads to significant computational performance increases when constraints are applicable.

One caveat is that the conditioning process is sensitive to the landmark cross-covariance estimates. (The cross-covariances are used in Equations 5.16-5.17 to compute a "gain" indicating how to change unconstrained variables when conditioning on constrained variables.) Because we

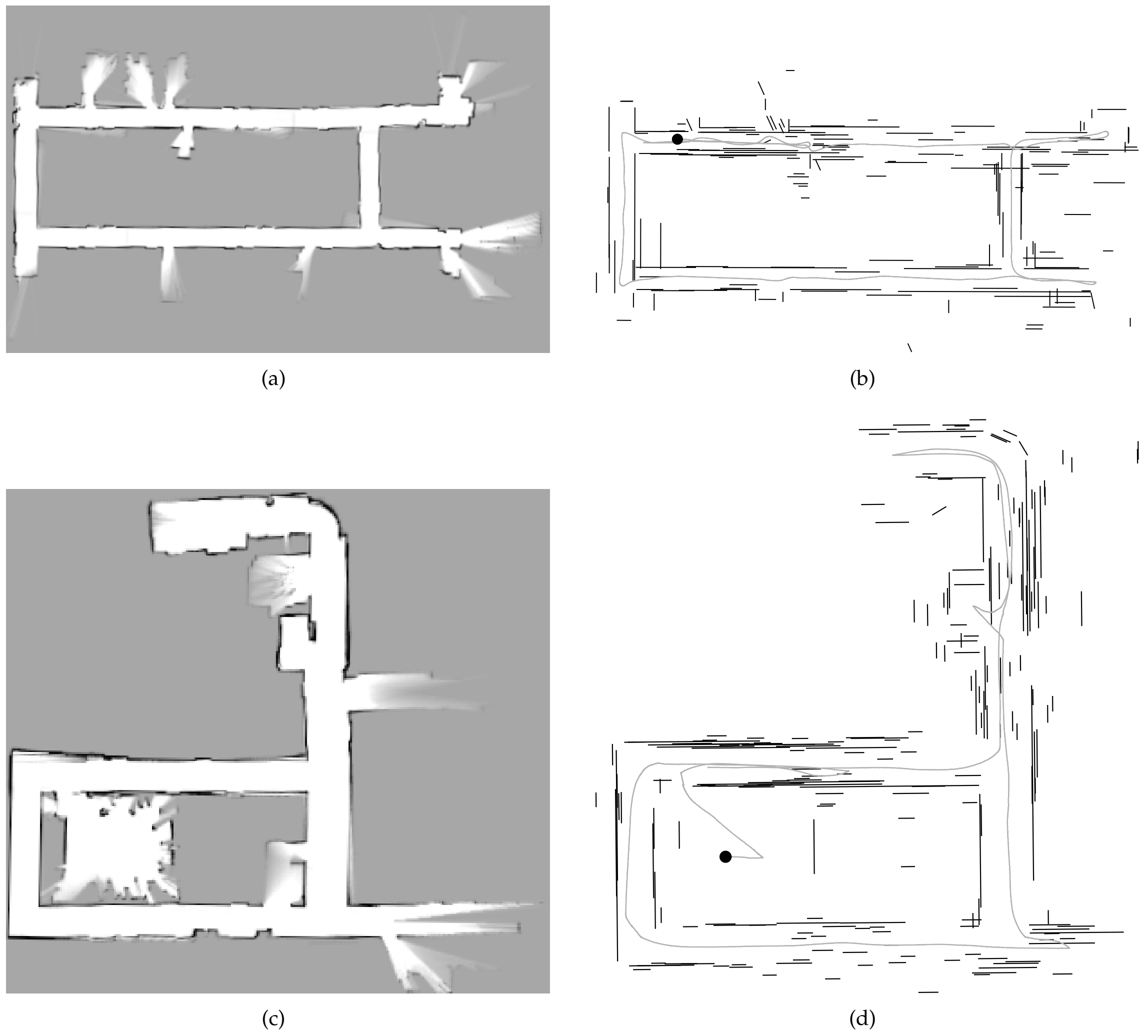


Figure 5.6: (a) and (b) show the USC SAL Building, second floor (data set courtesy of Andrew Howard). (c) and (d) show Newell-Simon Hall Level A at CMU (data set courtesy of Nicholas Roy). (a) and (c) Occupancy data for the corrected trajectories (generated using the full laser data for clarity). (b) and (d) The estimated landmark maps (black) and trajectories (gray).

use sensors that give very little data for feature extraction, the cross-covariance of $[r\ \ \theta]^T$ features is only approximately estimated. This leads to landmark drift in highly constrained environments since landmarks are frequently reconditioned, as can be seen in, e.g., the upper right corner of the NSH map in Figure 5.6(d). Future research will examine alternative feature estimators and map representations (e.g., relative maps (Newman, 1999; Deans and Hebert, 2000b)) that may alleviate this issue.

5.7 Summary

This chapter has described a Rao-Blackwellized particle filter for SLAM that exploits prior knowledge of structural or geometrical relationships between landmarks. Relative constraints between landmarks in the map of each particle are automatically inferred based on the estimated landmark state. By partitioning the state into constrained and unconstrained variables, the constrained variables can be sampled by a particle filter. Conditioned on these samples, unconstrained variables are independent and can be estimated by EKFs on a per-particle basis.

We have implemented our approach with rectilinearity constraints and performed experiments on simulated and real-world data. For SLAM with sparse (low spatial resolution) sensing, incorporating constraints significantly reduced the number of particles required for map estimation.

Most of this work has focused on linear equality constraints. While we have described a way to extend the approach to inequality constraints, this remains an area for future work. Also, while constraints clearly help in mapping with limited sensing, they do not significantly improve data association inaccuracies related to sparse sensing, another potential avenue for improvement.

6 SAMPLING STRATEGIES

This chapter of the thesis addresses the issue of filter consistency in Rao-Blackwellized particle filtering SLAM with the development of several new sampling strategies for use in the filtering process. A SLAM filter is inconsistent if it significantly underestimates the uncertainty in the robot and map state. Because this uncertainty is frequently quite large in robot systems with limited sensing, its accurate estimation is paramount to the ability of such robots to build accurate maps.

Two of the sampling strategies developed in this chapter, *fixed-lag roughening* and the *block proposal distribution*, exploit "future" information, when it becomes available, to improve the filter's pose and map estimation for previous time steps. Fixed-lag roughening perturbs trajectory samples over a fixed lag time according to a Markov Chain Monte Carlo kernel. The block proposal distribution directly samples poses over a fixed lag time from their fully joint distribution conditioned on all the available data. Our experimental results indicate that these strategies, especially the block proposal, yield significant improvements in filter consistency and a reduction in particle degeneracies compared to standard sampling techniques such as the improved proposal distribution of FastSLAM 2.

We also examine the effectiveness of two alternative resampling techniques, *residual resampling* and *generalized resampling*, as applied to RBPF SLAM. These drop-in-place techniques are simple to use and (in the case of residual resampling) computationally cheaper than the standard random resampling approach. However, our results show that they offer no real improvement in performance over random resampling in SLAM.

6.1 Overview

Simultaneous localization and mapping (SLAM) algorithms based on particle filters, like the algorithms described in the preceding two chapters, have gained exceptional popularity in the last few years due to their relative simplicity, computational properties, and experimental successes. However, recent work such as that by Bailey et al. (2006) and others has shown that current particle filtering SLAM algorithms are susceptible to substantial estimation inconsistencies because they generally significantly underestimate their own error. In large part this is due to degeneracies in the particle filter sampling process caused by frequent resampling such that most samples

$\hat{N}_{\text{eff}}$	effective sample size	$h()$	measurement model	FS2	FastSLAM 2
L	lag time	$g()$	forward motion model	FLR(L)	fixed-lag roughening, lag L
$\mathbf{x}^r_{t-L+1:t}$	last L poses	$s()$	backward motion model	BP(L)	block proposal, lag L
a^i_t	generalized weight	$\mathcal{N}(\mu, \mathbf{P})$	Normal distribution	RES	FS2, residual resampling
α	generalized resampling parameter	$\tilde{\mu}, \tilde{\mathbf{P}}^{r,i}$	forward-filtered dist.	GEN(α)	FS2, generalized resampling (param. α)
		$\mathbf{R}_k$	measurement covariance		

Notation for this chapter

become identical for much of the robot's trajectory.

For a complete introduction to particle filtering SLAM, see Appendix B. Briefly, a particle filter for SLAM represents the posterior distribution of the robot's trajectory using a set of samples, or "particles." Conditioned on each particle is a map, estimated using a series of small extended Kalman filters for each landmark. At each time step, particles are extended according to a motion model and maps are updated based on sensor observations. The particles are weighted according to the likelihood of the observations given the sampled poses and previous observations. Finally, particles are resampled (with replacement) according to to their weights in order to give more presence to highly-weighted trajectories.

Particle degeneracies and filter inconsistencies occur when the weights of particles in the filter are highly skewed. In this case, the resampling step selects many copies of a few highly weighted particles. Since resampling is repeated often (in the basic particle filter, at every time step), the sampled poses representing past portions of the robot's trajectory tend to become degenerate (i.e., all or mostly all identical) so that they insufficiently encode uncertainty in the estimation. These degeneracies can have significant consequences if the robot revisits the poorly estimated region, such as when closing a loop, because the map estimate may be incorrect and overcertain, preventing good data association.

Some previous work has addressed the issue of particle diversity. The improved proposal distribution of FastSLAM 2 (Montemerlo, 2003) seeks to sample "better" poses so that particle weights remain relatively uniform. The adaptive resampling technique employed by Grisetti et al. (2005), and used in all of the particle filtering algorithms described in this thesis, performs resampling only when the weights become skewed, rather than at every time step. (We further discuss these and other approaches in Section 6.2.)

In this chapter we describe two new sampling strategies for particle filtering SLAM, inspired in part by the tracking literature (Gilks and Berzuini, 2001; Doucet et al., 2006), which improve the consistency of the filter's estimation and the diversity of the trajectory samples. The first approach, termed *fixed-lag roughening*, incorporates a Markov Chain Monte Carlo (MCMC) move step, perturbing pose samples over a fixed lag time according to an MCMC kernel to combat particle degeneracy. The second technique employs a *block proposal distribution* which directly samples poses over a fixed lag time from their fully joint distribution conditioned on all of the available data. The main idea behind both methods is to exploit "future" information to improve the estimates of past portions of the trajectory, in the sense that the information becomes available only *after* initial estimation of the poses.

The new sampling techniques lead to significant reductions in estimation error over previous approaches. For example, in our experiments, estimation error using fixed-lag roughening was as little as 30% that of FastSLAM 2 on average, and error using the block proposal was as little as 12%, in both cases at the expense of a constant factor increase in computation time. Furthermore, trajectory samples from the block proposal exhibit better diversity than those from FastSLAM 2, with the filter maintaining multiple samples over most of the pose history for reasonably long trajectories.

The primary complication of both approaches is in designing and drawing from the sampling distributions — i.e., the MCMC kernel for fixed-lag roughening, and the fully joint pose distribution for the block proposal. We derive conditional distributions for a Gibbs sampler for the roughening case, and employ a well-known algorithm to simulate from joint distributions in state-space models for the block proposal. In addition, we show how to apply the techniques in practice using the standard Gaussian approximations to the motion and measurement models of the robot.

Other work in the statistics literature has pinpointed several alternatives to the usual random resampling strategy employed in particle filtering techniques (Liu, 2001). For purposes of comparison, we have implemented two of the approaches, residual resampling and generalized resampling, in the context of RBPF SLAM. Residual resampling deterministically selects a number of copies of each particle proportional to the weights. A benefit of this approach is that it requires access to many fewer random numbers than the standard approach. Generalized resampling draws particles with probability proportional to a *function* of the weights. The function can be tailored to balance the need for particle diversity with the need for sample effectiveness. Unfortunately our results show that neither technique yields much improvement over random resampling in terms of filter performance.

In the next section describe previous work on improving consistency and particle diversity. We introduce fixed-lag roughening in Section 6.3 and the block proposal distribution in Section 6.4. The two resampling strategies are discussed in Section 6.5. Finally, Section 6.6 presents the results of experiments comparing our techniques to previous approaches.

6.2 Related work

Several researchers have addressed consistency in the context of RBPF SLAM; we describe two well-known approaches and a third recent development.

6.2.1 Improved proposal distribution

For robots with very accurate sensors such as scanning laser rangefinders, the measurement likelihood $p(\mathbf{z}_t|\mathbf{x}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t})$ is highly peaked, whereas the motion model based proposal distribution $p(\mathbf{x}^r_t|\mathbf{x}^{r,i}_{t-1}, \mathbf{u}_t)$ is not. Thus, the proposal distribution samples many robot poses that are assigned low weights, so only a few samples survive the resampling step. This can quickly lead to particle degeneracies.

An alternative is to incorporate the current sensor measurement $\mathbf{z}_t$ into the proposal distribution, i.e., to sample robot poses according to:

$$p(\mathbf{x}^r_t|\mathbf{x}^{r,i}_{1:t-1}, \mathbf{u}_{1:t}, \mathbf{z}_t, \mathbf{n}_t) \tag{6.1}$$

Using this approach, many more samples are drawn for robot poses that match well with the current sensor measurement and particles are more evenly weighted, so more particles are likely to survive resampling. This "improved proposal distribution" has been used in both landmark-based SLAM (Montemerlo, 2003), where it is often called "FastSLAM 2," and in occupancy grid scan-matching SLAM (Grisetti et al., 2005).

6.2.2 Effective sample size

The basic particle filtering algorithm resamples particles according to their weights at every iteration, i.e., for every SLAM update step. It can be shown that if the weights of particles are approximately the same, resampling only *decreases* the efficiency of the sampled representation (Liu, 2001). The *effective sample size* is a useful metric to determine whether resampling is necessary (Liu and Chen, 1995). It can be approximated as:

$$\hat{N}_{\text{eff}} = \frac{1}{\sum_{i=1}^{N} \left(\omega^i_t\right)^2} \tag{6.2}$$

If the effective sample size is large, say, $\hat{N}_{\text{eff}} > N/2$, resampling is undesirable since the PDF over robot trajectories is well represented. This technique, originally developed by Liu and Chen (1995), was first applied to SLAM by Grisetti et al. (2005).

6.2.3 Recovering diversity through stored state

The preceding methods focus on preventing loss of particle diversity. Another approach is to attempt to "recover" diversity. Stachniss et al. (2005b) store the "state" of the particle filter (i.e., the current set of pose samples) upon detecting the robot's entry into a loop. The robot repeatedly traverses the loop to improve the map, a process normally resulting in loss of particle diversity. Finally, the diverse pose samples are restored by splicing the loop trajectory estimate onto each saved particle, effectively eliminating any degeneracies that arose during mapping of the loop.

6.3 Fixed-lag roughening

Resampling leads to degeneracies because multiple copies of the same highly-weighted particles survive. In this section we describe a modification of the particle filter sampling process that incorporates "roughening" of the sampled trajectories over a fixed lag so that the posterior over trajectories is better estimated. A similar approach termed "resample-move" has been described in the statistical literature in the context of target tracking (Gilks and Berzuini, 2001; Doucet et al., 2006), and has been mentioned (but not pursued) in the context of SLAM by Bailey (2002).

The basic idea is to incorporate a post-SISR Markov Chain Monte Carlo (MCMC) step to "move" the trajectory of each particle over a fixed lag time L after the usual RBPF update is complete. Specifically, for each particle $\phi_t^i, i = 1 \ldots N$, we sample:

$$\mathbf{x}_{t-L+1:t}^{r,i} \sim q(\mathbf{x}_{t-L+1:t}^r) \tag{6.3}$$

where q is an MCMC kernel with invariant distribution

$$p(\mathbf{x}_{t-L+1:t}^r | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \tag{6.4}$$

After the move, the particles are still approximately distributed according to the desired posterior but degeneracies over the lag time L have been averted. Furthermore, some "future" information is used in drawing new values for previously sampled poses. The samples are already approximately distributed according to the desired posterior before the move, so the usual burn-in time of MCMC samplers can be avoided. The MCMC move can be repeated to obtain better samples, although in our implementation we only perform a single move at each time step.

There are two main difficulties in implementing the approach. First, an appropriate kernel q and method for sampling from it must be devised. Second, care must be taken to avoid bias from counting the same measurement twice, leading to a need for a simple mechanism to manage incremental versions of the map.

6.3.1 Fixed-lag Gibbs sampler for SLAM

An effective approach for sampling from the joint MCMC kernel $q(\mathbf{x}_{t-L+1:t}^r)$ is to employ *Gibbs sampling*, which samples each component of $\mathbf{x}_{t-L+1:t}^r$ in turn from its conditional distribution given the values of other components. Specifically, we sample each component in turn according to the following scheme, for every particle ϕ_t^i:

$$\begin{aligned}
\mathbf{x}_{t-L+1}^{r,i} &\sim p(\mathbf{x}_{t-L+1}^r | \mathbf{x}_{1:t-L,t-L+2:t}^{r,i}, \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \\
&\ldots \\
\mathbf{x}_k^{r,i} &\sim p(\mathbf{x}_k^r | \mathbf{x}_{1:k-1,k+1:t}^{r,i}, \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \\
&\ldots \\
\mathbf{x}_t^{r,i} &\sim p(\mathbf{x}_t^r | \mathbf{x}_{1:t-1}^{r,i}, \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t})
\end{aligned} \tag{6.5}$$

The last distribution (6.5) is equivalent to the usual (improved) proposal distribution. The other intermediate distributions are of the form:

$$\mathbf{x}_k^{r,i} \sim p(\mathbf{x}_k^r | \mathbf{x}_{1:k-1,k+1:t}^{r,i}, \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \tag{6.6}$$

at a particular lag time k. We will concentrate on manipulating (6.6) into a form from which we can easily sample. We can first rewrite this distribution using Bayes' rule as follows:

$$p(\mathbf{x}_k^r|\mathbf{x}_{1:k-1,k+1:t}^{r,i}, \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \\ = \frac{p(\mathbf{z}_k|\mathbf{x}_{1:t}^{r,i}, \mathbf{u}_{1:t}, \mathbf{z}_{1:k-1,k+1:t}, \mathbf{n}_{1:t})\, p(\mathbf{x}_k^r|\mathbf{x}_{1:k-1,k+1:t}^{r,i}, \mathbf{u}_{1:t}, \mathbf{z}_{1:k-1,k+1:t}, \mathbf{n}_{1:t})}{p(\mathbf{z}_k|\mathbf{x}_{1:k-1,k+1:t}^{r,i}, \mathbf{u}_{1:t}, \mathbf{z}_{1:k-1,k+1:t}, \mathbf{n}_{1:t})} \tag{6.7}$$

The denominator can be subsumed into a normalization constant η:

$$= \quad \eta\, p(\mathbf{z}_k|\mathbf{x}_{1:t}^{r,i}, \mathbf{u}_{1:t}, \mathbf{z}_{1:k-1,k+1:t}, \mathbf{n}_{1:t})\, p(\mathbf{x}_k^r|\mathbf{x}_{1:k-1,k+1:t}^{r,i}, \mathbf{u}_{1:t}, \mathbf{z}_{1:k-1,k+1:t}, \mathbf{n}_{1:t}) \tag{6.8}$$

Applying the Markov assumption and factoring, we obtain:

$$= \quad \eta\, p(\mathbf{z}_k|\mathbf{x}_{1:t}^{r,i}, \mathbf{u}_{1:t}, \mathbf{z}_{1:k-1,k+1:t}, \mathbf{n}_{1:t})\ \underbrace{p(\mathbf{x}_k^r|\mathbf{x}_{k-1}^{r,i}, \mathbf{u}_k)}_{\text{forward model}}\ \underbrace{p(\mathbf{x}_k^r|\mathbf{x}_{k+1}^{r,i}, \mathbf{u}_{k+1})}_{\text{backward model}} \tag{6.9}$$

We now have three terms. The forward model is the distribution of robot poses given the previous pose and the current control input; the backward model is the distribution of poses given the next pose and control input. Finally, marginalizing the first term yields:

$$p(\mathbf{z}_k|\mathbf{x}_{1:t}^{r,i}, \mathbf{u}_{1:t}, \mathbf{z}_{1:k-1,k+1:t}, \mathbf{n}_{1:t}) = \\ \int \underbrace{p(\mathbf{z}_k|\mathbf{x}_k^{r,i}, \mathbf{n}_k, \mathbf{x}_{\mathbf{n}_k}^{m,i})}_{\text{measurement likelihood}}\ \underbrace{p(\mathbf{x}_{\mathbf{n}_k}^{m,i}|\mathbf{x}_{1:k-1,k+1:t}^{r,i}, \mathbf{z}_{1:k-1,k+1:t}, \mathbf{n}_{1:t})}_{\text{landmark distribution}}\ d\mathbf{x}_{\mathbf{n}_k}^{m,i} \tag{6.10}$$

Substituting this back into (6.9) we obtain:

$$p(\mathbf{x}_k^r|\mathbf{x}_{1:k-1,k+1:t}^{r,i}, \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) = \\ \eta \int p(\mathbf{z}_k|\mathbf{x}_k^{r,i}, \mathbf{n}_k, \mathbf{x}_{\mathbf{n}_k}^{m})\, p(\mathbf{x}_{\mathbf{n}_k}^{m}|\mathbf{x}_{1:k-1,k+1:t}^{r,i}, \mathbf{z}_{1:k-1,k+1:t}, \mathbf{n}_{1:t})\, d\mathbf{x}_{\mathbf{n}_k}^{m} \\ p(\mathbf{x}_k^r|\mathbf{x}_{k-1}^{r,i}, \mathbf{u}_k)\, p(\mathbf{x}_k^r|\mathbf{x}_{k+1}^{r,i}, \mathbf{u}_{k+1}) \tag{6.11}$$

6.3.2 Practical implementation

In practice we approximate the measurement likelihood, the landmark distributions, and the forward and backward models by Gaussians. Replacing the terms of (6.11) accordingly, we obtain the convolution of two Gaussians multiplied by two more Gaussians, i.e.:

$$\int \mathcal{N}\Big(h(\mathbf{x}_{\mathbf{n}_k}^{m,i}, \mathbf{x}_k^{r,i}), \mathbf{R}_k\Big)\, \mathcal{N}\Big(\mathbf{x}_{\mathbf{n}_k}^{m,i}, \mathbf{P}_{\mathbf{n}_k}^{m,i}\Big)\ d\mathbf{x}_{\mathbf{n}_k}^{m,i} \\ \times\ \mathcal{N}\Big(g(\mathbf{x}_{k-1}^{r,i}, \mathbf{u}_k), \mathbf{V}_k\Big)\ \mathcal{N}\Big(s(\mathbf{x}_{k+1}^{r,i}, \mathbf{u}_{k+1}), \mathbf{V}_{k+1}\Big) \tag{6.12}$$

where the functions h, g, and s represent the measurement model, forward motion model, and backward motion model, respectively.

Typically, the measurement model must be linearized in order to compute the integral, e.g., by taking the first-order Taylor expansion:

$$h(\mathbf{x}_{\mathbf{n}_k}^{m}, \mathbf{x}_k^{r}) \approx h(\mathbf{x}_{\mathbf{n}_k}^{m,i}, g(\mathbf{x}_{k-1}^{r,i}, \mathbf{u}_k)) + \mathbf{H}_m(\mathbf{x}_{\mathbf{n}_k}^{m} - \mathbf{x}_{\mathbf{n}_k}^{m,i}) + \mathbf{H}_r(\mathbf{x}_k^{r} - \mathbf{x}_k^{r,i}) \tag{6.13}$$

where $\mathbf{H}_m$ and $\mathbf{H}_r$ are the derivatives of h with respect to the observed landmark and the robot's pose, respectively, evaluated at the expected values.

Note that aside from the backward motion model term, the distribution (6.12) is exactly the approximation of the improved proposal described by Montemerlo (2003), who has shown that the resulting distribution is Gaussian with mean $\tilde{\mu}_k^{r,i}$ and covariance $\tilde{\mathbf{P}}_k^{r,i}$ as follows:

$$\begin{aligned}\tilde{\mu}_k^{r,i} &= \tilde{\mathbf{P}}_k^{r,i}\mathbf{H}_r^T\mathbf{S}_k^{-1}(\mathbf{z}_k - h(\mathbf{x}_{\mathbf{n}_k}^{m,i}, g(\mathbf{x}_{k-1}^{r,i}, \mathbf{u}_k))) + g(\mathbf{x}_{k-1}^{r,i}, \mathbf{u}_k) && (6.14)\\ \tilde{\mathbf{P}}_k^{r,i} &= \left(\mathbf{H}_r^T\mathbf{S}_k^{-1}\mathbf{H}_r + \mathbf{H}_u^T\mathbf{V}_k^{-1}\mathbf{H}_u\right)^{-1} && (6.15)\end{aligned}$$

where:

$$\mathbf{S}_k = \mathbf{R}_k + \mathbf{H}_m\mathbf{P}_{\mathbf{n}_k}^{m,i}\mathbf{H}_m^T \tag{6.16}$$

It remains to incorporate the backward motion model, which is straightforward since we can just merge the backward model distribution with the distribution (6.14-6.15), i.e.:

$$\begin{aligned}\mu_k^{r,i} &= \tilde{\mu}_k^{r,i} + \tilde{\mathbf{P}}_k^{r,i}(\tilde{\mathbf{P}}_k^{r,i} + \mathbf{H}_u\mathbf{V}_{k+1}\mathbf{H}_u^T)^{-1}(s(\mathbf{x}_{k+1}^{r,i}, \mathbf{u}_{k+1}) - \tilde{\mu}_k^{r,i}) && (6.17)\\ \mathbf{P}_k^{r,i} &= \left((\tilde{\mathbf{P}}_k^{r,i})^{-1} + \mathbf{H}_u^T\mathbf{V}_{k+1}^{-1}\mathbf{H}_u\right)^{-1} && (6.18)\end{aligned}$$

The fixed-lag roughening algorithm computes the distribution $\mathcal{N}\left(\mu_k^{r,i}, \mathbf{P}_k^{r,i}\right)$ for each time step $k = t - L + 1 \dots t$ for every particle ϕ_t^i to draw the new poses $\mathbf{x}_{t-L+1:t}^{r,i}$, and then updates the maps conditioned on the new trajectories.

6.3.3 Incremental map management

To avoid bias the intermediate map estimate $p(\mathbf{x}_{\mathbf{n}_k}^m | \mathbf{x}_{1:k-1,k+1:t}^{r,i}, \mathbf{z}_{1:k-1,k+1:t}, \mathbf{n}_{1:t})$ used in (6.11) should incorporate all available information *except* the measurement $\mathbf{z}_k$ from the time step being moved. Thus, it is necessary to store "incremental" versions of the map over the lag time so that the intermediate map distributions can be computed. A simple strategy which we use in our implementation is to store the map from time $t - L$ and the measurements $\mathbf{z}_{t-L+1:t}$. The intermediate map distributions are computed on the fly by applying EKF updates to the map using the observations from all but the kth time step.

To avoid storing multiple complete copies of the map of each particle, the binary tree data structure of log N FastSLAM (Montemerlo, 2003) can be used to store only the differences between maps from each time step.*

6.3.4 Discussion

Note that (6.11) is nearly identical to the result of similar manipulations of the improved proposal distribution (6.1) as described by Montemerlo (2003). The primary difference is the inclusion of the "backward model" $p(\mathbf{x}_k^r | \mathbf{x}_{k+1}^{r,i}, \mathbf{u}_{k+1})$ since we are sampling a pose in the midst of the trajectory rather than simply the most recent pose.

Note also that we do not reweight the particles after performing the MCMC roughening step. This is because the particles before the move are asymptotically drawn from the same distribution as those after the move.

6.4 Block proposal distribution

An alternative approach for exploiting "future" information in drawing trajectory samples over a fixed lag time L is to draw new samples for the last L poses directly from the joint "L-optimal

*We note that while the tree described in (Montemerlo, 2003) stores landmarks only at the leaf nodes and thus requires $O(n \log n)$ memory for n landmarks, it is possible to store landmarks at every node in the tree, which requires $O(n)$ memory.

block proposal distribution," i.e.:

$$p(\mathbf{x}^r_{t-L+1:t}|\mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}, \mathbf{x}^{r,i}_{t-L}) \tag{6.19}$$

The basic idea is to sample from (6.19) at each time step, replacing the most recent L poses of each particle with the newly sampled ones. The result is a particle filter that is "current" in that its samples are always distributed according to the desired posterior:

$$p(\mathbf{x}^r_{1:t}, \mathbf{x}^m|\mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \tag{6.20}$$

The filter's results can therefore be used in, e.g., planning, but the samples are much better than those from the usual particle filter since future information is directly exploited by the joint proposal. Thus, degeneracies in the weights of particles are much less likely to occur. A related technique was recently described by Doucet et al. (2006) in the general particle filtering context. One can think of the standard improved proposal distribution (6.1) as a "1-optimal" version of the block proposal.

The main difficulty in employing the block proposal is in drawing samples from the joint distribution (6.19). Our approach relies on the factorization due to Chib (1996):

$$\begin{aligned} p(\mathbf{x}^r_{t-L+1:t}|\mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}, \mathbf{x}^{r,i}_{t-L}) = p(\mathbf{x}^r_{t}|\mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}, \mathbf{x}^{r,i}_{t-L}) \times \\ p(\mathbf{x}^r_{t-1}|\mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}, \mathbf{x}^{r,i}_{t-L}, \mathbf{x}^{r,i}_{t}) \times \cdots \times p(\mathbf{x}^r_{k}|\mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}, \mathbf{x}^{r,i}_{t-L}, \mathbf{x}^{r,i}_{k+1:t}) \times \cdots \times \\ p(\mathbf{x}^r_{t-L+1}|\mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}, \mathbf{x}^{r,i}_{t-L}, \mathbf{x}^{r,i}_{t-L+2:t}) \end{aligned} \tag{6.21}$$

Here, the typical term is:

$$p(\mathbf{x}^r_{k}|\mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}, \mathbf{x}^{r,i}_{t-L}, \mathbf{x}^{r,i}_{k+1:t}) \tag{6.22}$$

and application of Bayes' rule and the Markov assumption leads to (Chib, 1996):

$$\begin{aligned} & p(\mathbf{x}^r_{k}|\mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}, \mathbf{x}^{r,i}_{t-L}, \mathbf{x}^{r,i}_{k+1:t}) \\ & \quad \propto p(\mathbf{x}^r_{k}|\mathbf{u}_{1:k}, \mathbf{z}_{1:k}, \mathbf{n}_{1:k}, \mathbf{x}^{r,i}_{t-L})\ p(\mathbf{x}^r_{k+1:t}, \mathbf{z}_{k+1:t}|\mathbf{x}^{r,i}_{k}, \mathbf{x}^{r,i}_{t-L}, \mathbf{u}_{1:t}, \mathbf{z}_{1:k}, \mathbf{n}_{1:t}) && (6.23) \\ & \quad = p(\mathbf{x}^r_{k}|\mathbf{u}_{1:k}, \mathbf{z}_{1:k}, \mathbf{n}_{1:k}, \mathbf{x}^{r,i}_{t-L})\ p(\mathbf{x}^r_{k+1}|\mathbf{x}^{r,i}_{k}, \mathbf{u}_{k+1}) \\ & \qquad \times p(\mathbf{x}^r_{k+2:t}, \mathbf{z}_{k+1:t}|\mathbf{x}^{r,i}_{k}, \mathbf{x}^{r,i}_{k+1}, \mathbf{u}_{1:t}, \mathbf{z}_{1:k}, \mathbf{n}_{1:t}, \mathbf{x}^{r,i}_{t-L}) && (6.24) \\ & \quad \propto p(\mathbf{x}^r_{k}|\mathbf{u}_{1:k}, \mathbf{z}_{1:k}, \mathbf{n}_{1:k}, \mathbf{x}^{r,i}_{t-L})\ p(\mathbf{x}^r_{k+1}|\mathbf{x}^{r,i}_{k}, \mathbf{u}_{k+1}) && (6.25) \end{aligned}$$

where the final step follows because $p(\mathbf{x}^r_{k+2:t}, \mathbf{z}_{k+1:t}|\mathbf{x}^{r,i}_{k}, \mathbf{x}^{r,i}_{k+1}, \mathbf{u}_{1:t}, \mathbf{z}_{1:k}, \mathbf{n}_{1:t}, \mathbf{x}^{r,i}_{t-L})$ is independent of $\mathbf{x}^r_k$.

The idea is to first *filter forward* over the robot's trajectory by computing the distributions $\{p(\mathbf{x}^r_{k}|\mathbf{u}_{1:k}, \mathbf{z}_{1:k}, \mathbf{n}_{1:k}, \mathbf{x}^{r,i}_{t-L})\}$ using alternating prediction and update steps (e.g., with an EKF), and then *sample backward*, first drawing:

$$\mathbf{x}^{r,i}_t \sim p(\mathbf{x}^r_{t}|\mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}, \mathbf{x}^{r,i}_{t-L}) \tag{6.26}$$

and then sampling the poses from the preceding time steps in reverse order using the distributions that arise from substituting the sampled values into (6.25). This process is repeated for every particle, and the corresponding maps are updated conditioned on the sampled trajectories.

Once new samples have been drawn for $\{\mathbf{x}^{r,i}_{t-L+1:t}\}$, the particles are reweighted according to the usual technique, i.e.:

$$\omega^i_t = \omega^i_{t-1} \frac{\text{target distribution}}{\text{proposal distribution}} \tag{6.27}$$

The optimal weight update is given by:

$$\begin{aligned}
\omega_t^i &= \omega_{t-1}^i \, \frac{p(\mathbf{x}^r_{1:t-L} | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t})}{p(\mathbf{x}^r_{1:t-L} | \mathbf{u}_{1:t-1}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t-1})} && (6.28) \\
&= \omega_{t-1}^i \, \frac{p(\mathbf{z}_t | \mathbf{x}^{r,i}_{1:t-L}, \mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t}) \; p(\mathbf{x}^r_{1:t-L} | \mathbf{u}_{1:t-1}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t-1})}{p(\mathbf{x}^r_{1:t-L} | \mathbf{u}_{1:t-1}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t-1}) \; p(\mathbf{z}_t | \mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t})} && (6.29) \\
&\propto \omega_{t-1}^i \, p(\mathbf{z}_t | \mathbf{x}^{r,i}_{1:t-L}, \mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t}) && (6.30)
\end{aligned}$$

Thus, the weight update is proportional to the likelihood of the current measurement using the forward-filtered pose and map distribution.

6.4.1 Practical implementation

As with fixed-lag roughening, we implement the necessary models as Gaussians in practice. The algorithm for sampling from (6.19) proceeds in two steps: forward filtering and backward sampling.

6.4.1.1 Forward filtering

The forward filtering step estimates the intermediate pose distributions conditioned upon past and present data, but not on future data. Note that the distributions must be computed separately for each particle since they are also conditioned on the "starting pose" $\mathbf{x}^{r,i}_{t-L}$. We implement the step using an extended Kalman filter (EKF) with the usual alternation between predictions and updates, for $k = t - L + 1 \ldots t$. We first compute the prediction:

$$\begin{aligned}
\tilde{\mu}_k^{r,i} &= g(\tilde{\mu}_{k-1}, \mathbf{u}_k) && (6.31) \\
\tilde{\mathbf{P}}_k^{r,i} &= \tilde{\mathbf{P}}_{k-1}^{r,i} + \mathbf{H}_u \mathbf{V}_k \mathbf{H}_u^T && (6.32)
\end{aligned}$$

and then apply the measurement(s) to improve the model using (6.14-6.15). The resulting distribution is used recursively to compute the distribution for the next time step. The process is initialized with $\tilde{\mathbf{P}}^{r,i}_{t-L} = \mathbf{0}$ since the pose $\mathbf{x}^{r,i}_{t-L}$ is already sampled.

Note that during the forward filtering step, a temporary version of the map *must be updated* with the measurements from each time step to obtain the correct forward-filtered distributions. The ideal approach is to apply the EKF to the full state vector $[\mathbf{x}^r \;\; \mathbf{x}^m]^T$ over the lag time. If the number of observed landmarks in a time step is less than a constant m (rather than a function of the size of the map), then the total cost of forward filtering is $O(NLm^3)$, i.e., asymptotically constant time to draw N samples at each time step.

An alternative approach is to assume the landmarks are independent and apply the usual RBPF updates to the landmarks during forward filtering, inflated by the uncertainty of the intermediate pose distributions computed by the EKF, i.e.:

$$\begin{aligned}
\mathbf{x}^{m,i}_{\mathbf{n}_k} &= \mathbf{x}^{m,i}_{\mathbf{n}_k} + (\mathbf{P}^{m,i}_{\mathbf{n}_k} \mathbf{H}_m^T \mathbf{S}^{-1})(\mathbf{z}_k - h(\mathbf{x}^{m,i}_{\mathbf{n}_k}, \tilde{\mu}_k)) && (6.33) \\
\mathbf{P}^{m,i}_{\mathbf{n}_k} &= \left((\mathbf{P}^{m,i}_{\mathbf{n}_k})^{-1} + \mathbf{H}_m^T \mathbf{R}_k^{-1} \mathbf{H}_m + \mathbf{H}_r^T (\tilde{\mathbf{P}}_k^{r,i})^{-1} \mathbf{H}_r \right)^{-1} && (6.34)
\end{aligned}$$

6.4.1.2 Backward sampling

Once the forward filtering step is complete, we draw samples for each intermediate pose $\mathbf{x}^r_k$, starting with $k = t$ and decrementing until $k = t - L + 1$. The first sample is drawn directly from the (approximately) optimal forward-filtered distribution:

$$\mathbf{x}_t^{r,i} \sim \mathcal{N}\left(\tilde{\mu}_t^{r,i}, \tilde{\mathbf{P}}_t^{r,i}\right) \qquad (6.35)$$

The remaining samples are conditioned on the poses drawn for the succeeding time steps by applying the same backward model used in fixed-lag roughening, i.e., we apply the steps (6.17-6.18) with $\tilde{\mu}_k^{r,i}$ and $\tilde{\mathbf{P}}_k^{r,i}$ as computed by during forward filtering, and draw samples from the resulting Gaussians.

6.4.1.3 Reweighting

After drawing the samples $\{\mathbf{x}_{t-L+1:t}^{r,i}\}$, the particles must be reweighted to approximate the desired posterior. From (6.30), it can easily be seen that the appropriate weight update is:

$$\omega_t^i \quad = \quad \omega_{t-1}^i \times |2\pi \mathbf{L}_t^i|^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{z}_t - h(\mathbf{x}_{\mathbf{n}_t}^{m,i}, \tilde{\mu}_t^{r,i}))^T (\mathbf{L}_t^i)^{-1} (\mathbf{z}_t - h(\mathbf{x}_{\mathbf{n}_t}^{m,i}, \tilde{\mu}_t^{r,i}))\right) \tag{6.36}$$

with:

$$\mathbf{L}_t^i = \mathbf{H}_r \tilde{\mathbf{P}}_t^{r,i} \mathbf{H}_r^T + \mathbf{H}_m \mathbf{P}_{\mathbf{n}_t}^{m,i} \mathbf{H}_m^T + \mathbf{R}_t \tag{6.37}$$

where $\mathbf{x}_{\mathbf{n}_t}^{m,i}$ and $\mathbf{P}_{\mathbf{n}_t}^{m,i}$ are as computed during the forward filtering step.

6.4.2 Discussion

At first it may appear that the samples drawn using the block proposal distribution are no different from those obtained with fixed-lag roughening. In fact, while the samples are asymptotically from the same distribution (the desired posterior), those obtained from the block proposal will generally be better. This is because poses over the lag time are drawn *directly* from the joint distribution that incorporates future information. On the other hand, in fixed-lag roughening, poses are originally drawn using past and present information only, and then are gradually moved as future information becomes available. Only through the application of many MCMC moves at each time step will the samples obtained by fixed-lag roughening be as good as those from the block proposal.

6.5 Resampling strategies

In addition to fixed-lag roughening and the block proposal distribution described above, we have examined the use of alternative resampling methods in RBPF SLAM. To our knowledge, all published RBPF SLAM algorithms employ the random resampling approach, which resamples particles with probability proportional to their importance weights. We briefly describe two alternative techniques from the statistical literature (Liu, 2001), termed *residual resampling* and *generalized resampling*, and apply them to particle filtering SLAM. Our results indicate that the performance of these strategies in the context of SLAM is not appreciably better than random resampling. However, they offer flexibility and, in the case of residual resampling, may be computationally beneficial.

6.5.1 Residual resampling

Residual sampling is a mostly-deterministic approach that enforces the number of copies of a particle retained during resampling to be (approximately) proportional to the weight of the sample. (Note that this is the expected result of random resampling.) The technique is shown in Algorithm 6.1.

Algorithm 6.1 Residual resampling

```
1: Let $\tilde{\omega}_t^j = \omega_t^j / \sum_{i=1}^N \omega_t^i$
2: Retain $k_j = \lfloor N\tilde{\omega}_t^j \rfloor$ copies of $\phi_t^j$
3: Let $N_r = N - k_1 - \ldots - k_N$
4: Obtain $N_r$ i.i.d. draws (with replacement) from $\{\phi_t^1, \ldots, \phi_t^N\}$ w.p. proportional to $N\tilde{\omega}_t^j - k_j$
5: $\forall j,\ \omega_t^j = 1/N$
```

Since the number of deterministically selected copies of all particles, $\sum_{j=1}^{N} k_j$, may be less than N, random resampling is performed according to the residuals $N\tilde{\omega}_t^j - k_j$ in Step 4 to prevent bias.

According to Liu (2001), residual resampling can be shown to "dominate" random resampling in that it yields more accurate PDF approximations and is comparable or better in terms of computation. Perhaps the primary benefit is that residual resampling gives comparable performance to random resampling while consuming many fewer random numbers, which may be costly to generate, particularly in embedded scenarios.

6.5.2 Generalized resampling

The idea of generalized resampling is to resample according to alternative probabilities $\{a_t^i\}$ instead of the usual importance weights $\{\omega_t^i\}$. The intuition behind this approach, which is depicted in Algorithm 6.2, is that $\{a_t^i\}$ can be used to "modify" the weights of particles, balancing focus (giving more presence to particles with high weights) with particle diversity.

Algorithm 6.2 Generalized resampling

```
1: for $j = 1 \ldots N$ do
2:     Draw $k$ from $\{1, \ldots, N\}$ according to $a_t^i$, $i = 1 \ldots N$
3:     Let $\tilde{\phi}_t^j = \phi_t^k$
4:     Let $\tilde{\omega}_t^j = \omega_t^k / a_t^k$
5: end for
6: return the $\tilde{\phi}_t^i$s and $\tilde{\omega}_t^i$s
```

Liu (2001) suggests assigning generalized weights according to:

$$a_t^i = \left(\omega_t^i\right)^\alpha \tag{6.38}$$

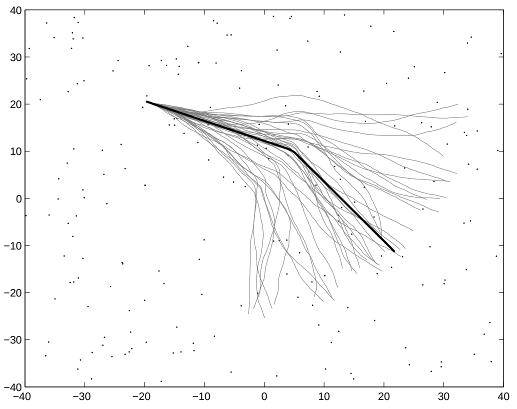
with $0 < \alpha \leq 1$. By picking $\alpha < 1$, the weight of seemingly poor particles is slightly amplified, giving them a "second chance." (Note that the a_t^is should be monotone in ω_t^i since we generally want to discard bad samples and duplicate good ones.) The weights are reset nonuniformly after resampling to prevent bias.

6.6 Results

Our experiments compared the standard FastSLAM 2 algorithm, the fixed-lag roughening (FLR) algorithm from Section 6.3, the block proposal (BP) distribution from Section 6.4, and FastSLAM 2 with residual (RES) and generalized (GEN) resampling as described in Section 6.5. For the roughening and block proposal approaches we tested the algorithms with several values for the lag time L, and generalized resampling was tested with several values of the parameter α. All experiments used $N = 500$ particles and resampling was performed only when $\hat{N}_{\text{eff}} < N/2$.

Our experiments were in simulation since comparing the estimation error of the filters requires ground truth. We assumed known data associations to prevent poor correspondence-finding from influencing the comparison between filtering algorithms. Noise was introduced by perturbing odometry and range-bearing measurements. The observation model used $\sigma_r = 5$ cm and $\sigma_b = 0.3°$ with a sensing radius of 10 m, and the motion model used $\sigma_x = 0.12d\cos\theta$, $\sigma_y = 0.12d\sin\theta$ and $\sigma_\theta = 0.12d + 0.24\phi$ for translation d and rotation ϕ.

Experiments were performed in a variety of simulated environments consisting of point features. We present results from two representative cases with randomly placed landmarks: a "sparse" map with a simple 27 sec. trajectory (no loops) and a "dense" map with a 63 sec. loop trajectory. The environments, ground truth trajectories, and typical raw odometry estimates are shown in Figure 6.1. All results were obtained by averaging 50 Monte Carlo trials of each simulation.


(a) Sparse environment

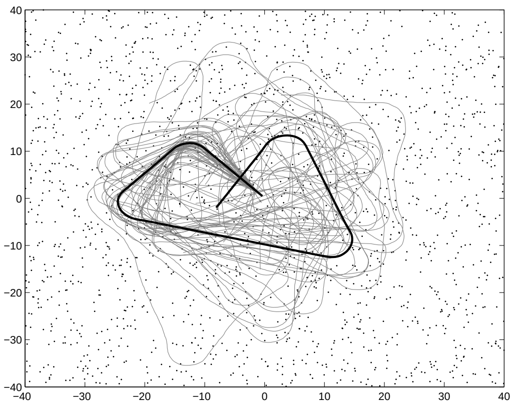

(b) Dense environment

Figure 6.1: Simulated environments used to compare RBPF SLAM algorithms. The environments consist of point landmarks placed uniformly at random. The solid dark lines represent the ground truth trajectories of the robot, which were kept the same for all simulations. The lighter gray lines depict several typical uncorrected odometry estimates of the robot's trajectory.

6.6.1 NEES comparison

We begin by comparing the normalized estimation error squared (NEES) (Bar-Shalom et al., 2001; Bailey et al., 2006) of the trajectory estimates produced by each algorithm. The NEES is a useful measure of filter consistency since it estimates the *statistical* distance of the filter estimate from the ground truth, i.e., it takes into account the filter's estimate of its own error. Recall from Chapter 5 that for a ground truth pose $\mathbf{x}_t^r$ and an estimate $\hat{\mathbf{x}}_t^r$ with covariance $\hat{\mathbf{P}}_t^r$ (estimated from the weighted particles assuming they are approximately Gaussian), the NEES is:

$$(\mathbf{x}_t^r - \hat{\mathbf{x}}_t^r)(\hat{\mathbf{P}}_t^r)^{-1}(\mathbf{x}_t^r - \hat{\mathbf{x}}_t^r)^T \tag{6.39}$$

The recent paper by Bailey et al. (2006) gives more details about using NEES to measure RBPF SLAM consistency.

We computed the NEES at each time step using the current particle set. Figures 6.2-6.3 and 6.4-6.5 show the resulting errors from each of the algorithms in the sparse and dense examples, respectively. In the sparse environment, NEES grows steadily for FS2, FLR, RES, GEN, and for BP with small lag times. Increasing the lag time for FLR has relatively little effect on NEES because "future" information is exploited slowly (see Section 6.4.2). FLR's NEES is approximately 33% that of FS2 on average for $L = 5$ and $L = 10$. On the other hand, increasing the lag time for BP dramatically reduces NEES. The NEES of BP(1) is roughly 73% that of FS2 on average; for BP(5), 22%; and for BP(10), 12%. RES and GEN fail to improve on random resampling and in fact — aside from GEN(0.5) — do noticeably worse than FS2 in the sparse environment.

For the dense case the results are mostly similar. Note that FLR avoids degeneracies (manifested as spikes in the NEES plots) by moving particles after resampling. Interestingly, increasing L in a dense environment appears to slightly *increase* the NEES of FLR, a subject warranting further investigation. RES and GEN do better in the dense case, with performance comparable to that of FS2. In the dense environment, increasing α (placing more emphasis on representation accuracy than on particle diversity) leads to improved performance, a clue that generalized resampling is offering little benefit in this case aside from the expected slight reduction in degeneracies.

Note that the range of the NEES plots is quite large — none of the filters is truly consistent. (A consistent filter over 50 Monte Carlo trials should have NEES less than 3.72 with 95% probability (Bailey et al., 2006).) While the estimation error using fixed-lag roughening and the block proposal is significantly reduced, these strategies alone do not guarantee a consistent filter, at least with reasonably small lag times. In fact it is likely that guaranteeing consistent SLAM estimation (with high probability) while representing the trajectory posterior by samples requires drawing the full dimensionality of the samples from a distribution conditioned on all the measurements, e.g., with MCMC, since particle filtering is always susceptible to resampling degeneracies depending on the environment and trajectory.

6.6.2 $\hat{N}_{\text{eff}}$ comparison

The effective sample size $\hat{N}_{\text{eff}}$ is also a useful statistic in examining filter consistency. If $\hat{N}_{\text{eff}}$ is high, the weights of particles are relatively unskewed, i.e., all particles are contributing to the estimate of the trajectory posterior. Furthermore, since $\hat{N}_{\text{eff}}$ dictates when resampling occurs, high values of $\hat{N}_{\text{eff}}$ indicate less chance for degeneracies in past portions of the trajectory estimate because resampling occurs infrequently.

Figures 6.6-6.7 show $\hat{N}_{\text{eff}}$ as computed at each time step in the simulations. In the sparse case, FLR exhibits no significant improvement over FS2, but in the dense environment FLR(5) and FLR(10) have about 72% higher $\hat{N}_{\text{eff}}$ than FS2 on average. Again, BP exhibits stronger results, with BP(10) more than 1000% better than FS2 in the dense case, and 340% better in the sparse case. This can be attributed to the direct use of future information by the block proposal, which leads to better samples for essentially the same reason FastSLAM 2's samples are better than those of FastSLAM 1. Residual resampling gives slightly better performance than FS2 (a result of its deterministic approach). GEN yields worse $\hat{N}_{\text{eff}}$ than FS2 in all cases, with the ratios approximately proportional to the parameter α. Again, this is the expected result since generalized resampling trades off representation accuracy (which is measured by $\hat{N}_{\text{eff}}$) for particle diversity.

6.6.3 Particle diversity

Finally, we examine particle diversity for each of the different filters. Figures 6.8 and 6.10 show the variance of the pose histories of all the particles, computed at the end of SLAM. Figures 6.9 and 6.11 show the number of unique particles representing each pose. For all of the algorithms, the end of the trajectory is better represented than earlier portions. FLR extends the representation over the lag time but the typical quick dropoff remains. BP avoids the loss of diversity in the sparse case, maintaining non-zero variance over most of the trajectory, as one would expect since little resampling occurs due to the high effective sample size. In a denser environment a significant amount of resampling still occurs, reducing the benefit somewhat. RES gives slightly worse performance than FS2. GEN very slightly improves on FS2, with about 1% more unique samples of each pose on average for $\alpha = 0.2$ and $\alpha = 0.5$.

6.7 Summary

We have described two new sampling strategies for particle filtering SLAM. The first method, fixed-lag roughening, applies an MCMC move to the trajectory samples over a fixed lag at each time step. The second approach, the block proposal distribution, draws new samples for all poses in a fixed-lag portion of the trajectory from their joint distribution. Both techniques exploit "future" information to improve the estimation of past poses.

Our results show that the new algorithms lead to substantial improvements in SLAM estimation. Fixed-lag roughening and the block proposal yield samples which exhibit less statistical estimation error than those of FastSLAM 2. Furthermore, the samples drawn by the block proposal distribution tend to have much more uniform importance weights (and thus higher "effective sample sizes"), leading to less need for resampling and consequently, improved particle diversity.

We have also examined the utility of two alternative resampling algorithms. Residual resampling deterministically selects a number of copies of each particle proportional to the weights, consuming many fewer random numbers than the usual random resampling. Generalized resampling selects samples according to some function of their importance weights, giving flexibility in balancing particle diversity with representation accuracy. While our results have shown that neither technique significantly improves upon the estimation performance of basic random resampling, the approaches expand the resampling toolkit; in particular, residual resampling may be useful in computationally restricted scenarios.

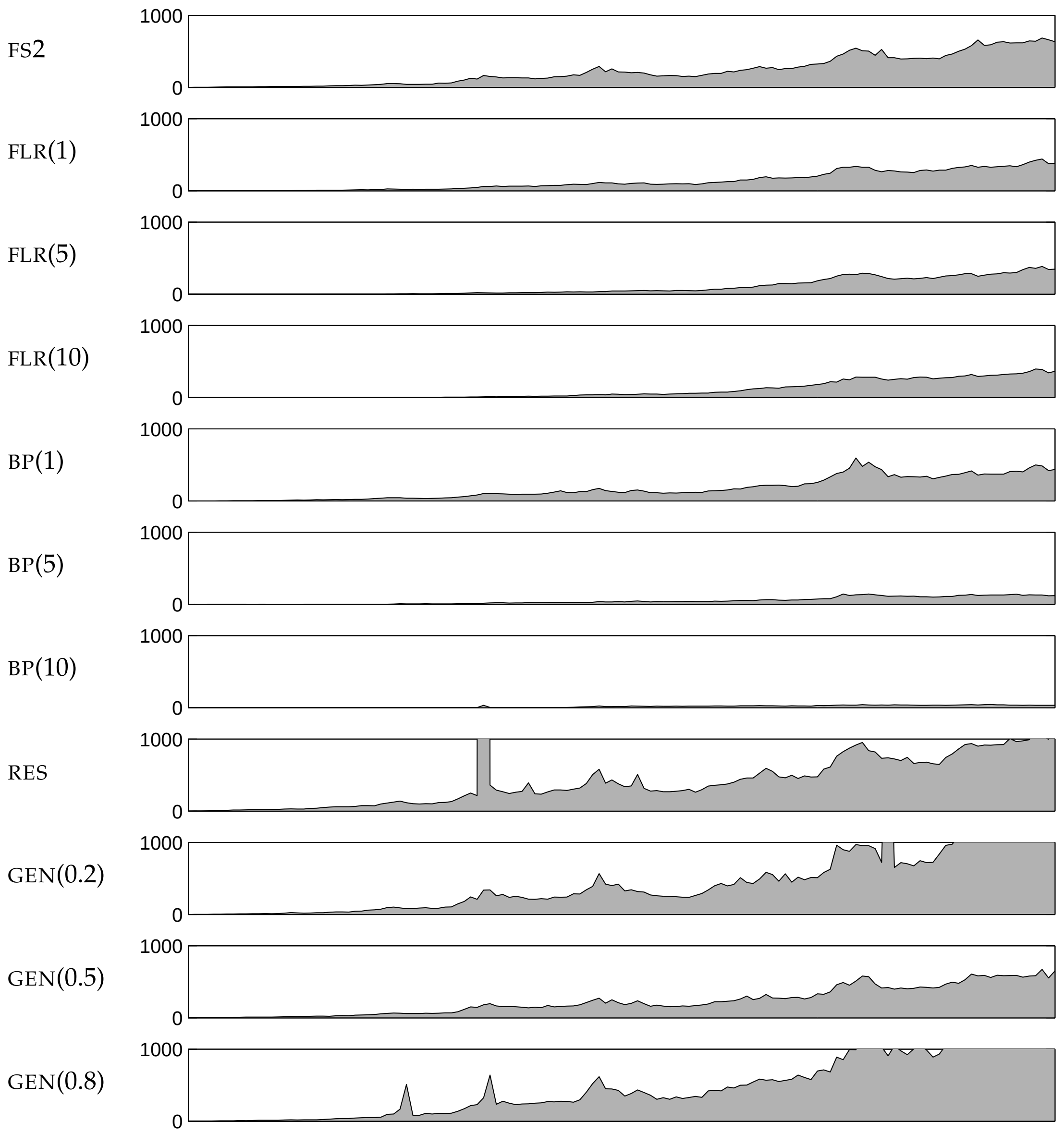


Figure 6.2: NEES for the sparse environment, versus the simulation time on the x-axis. We use the following abbreviations: FS2 for FastSLAM 2 with random resampling, FLR(L) for fixed-lag roughening with lag time L, BP(L) for the block proposal with lag time L, RES for FastSLAM 2 with residual resampling, and GEN(α) for generalized resampling with parameter α.

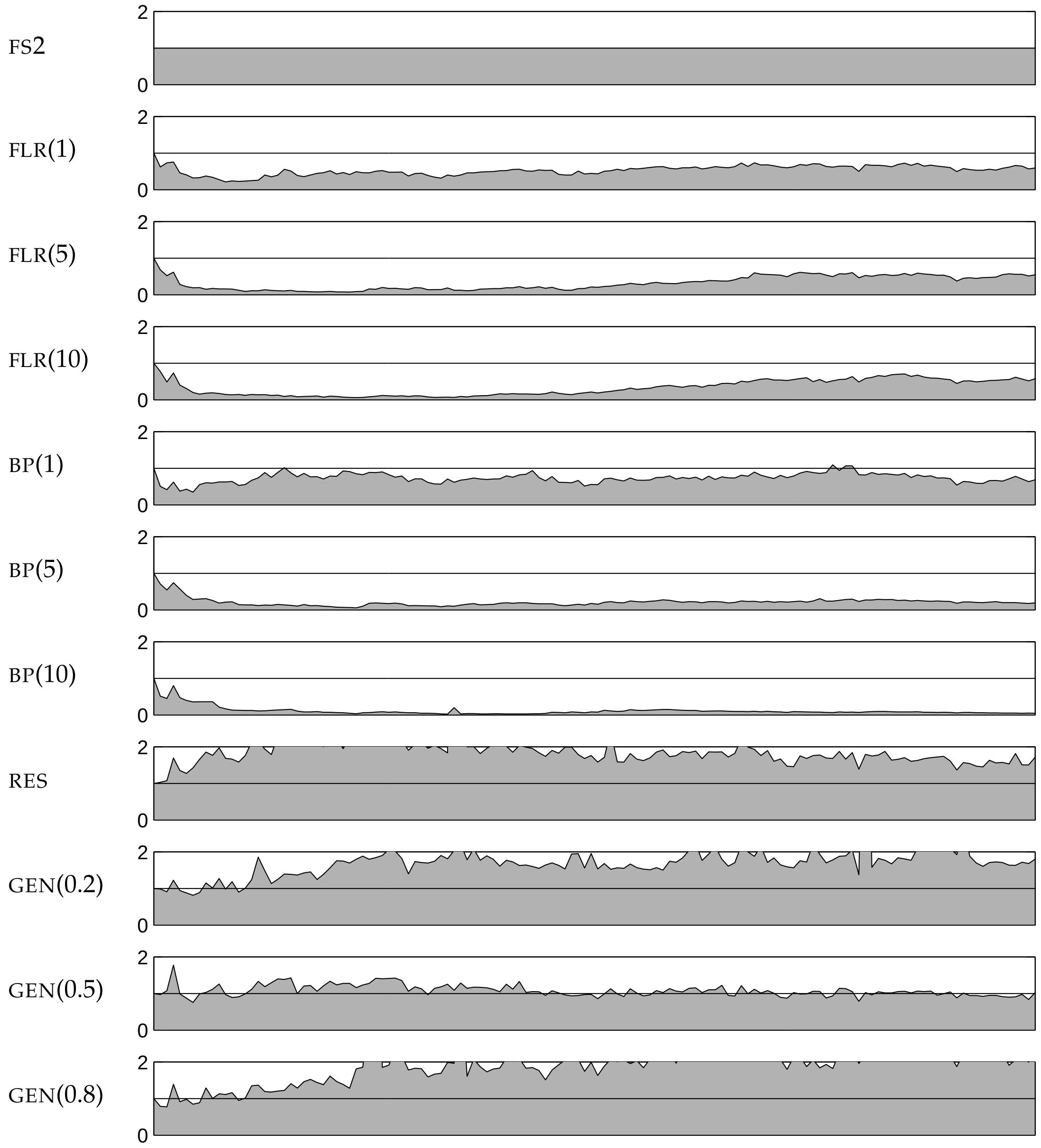


Figure 6.3: Ratio of NEES to that of FastSLAM 2 for the sparse environment.

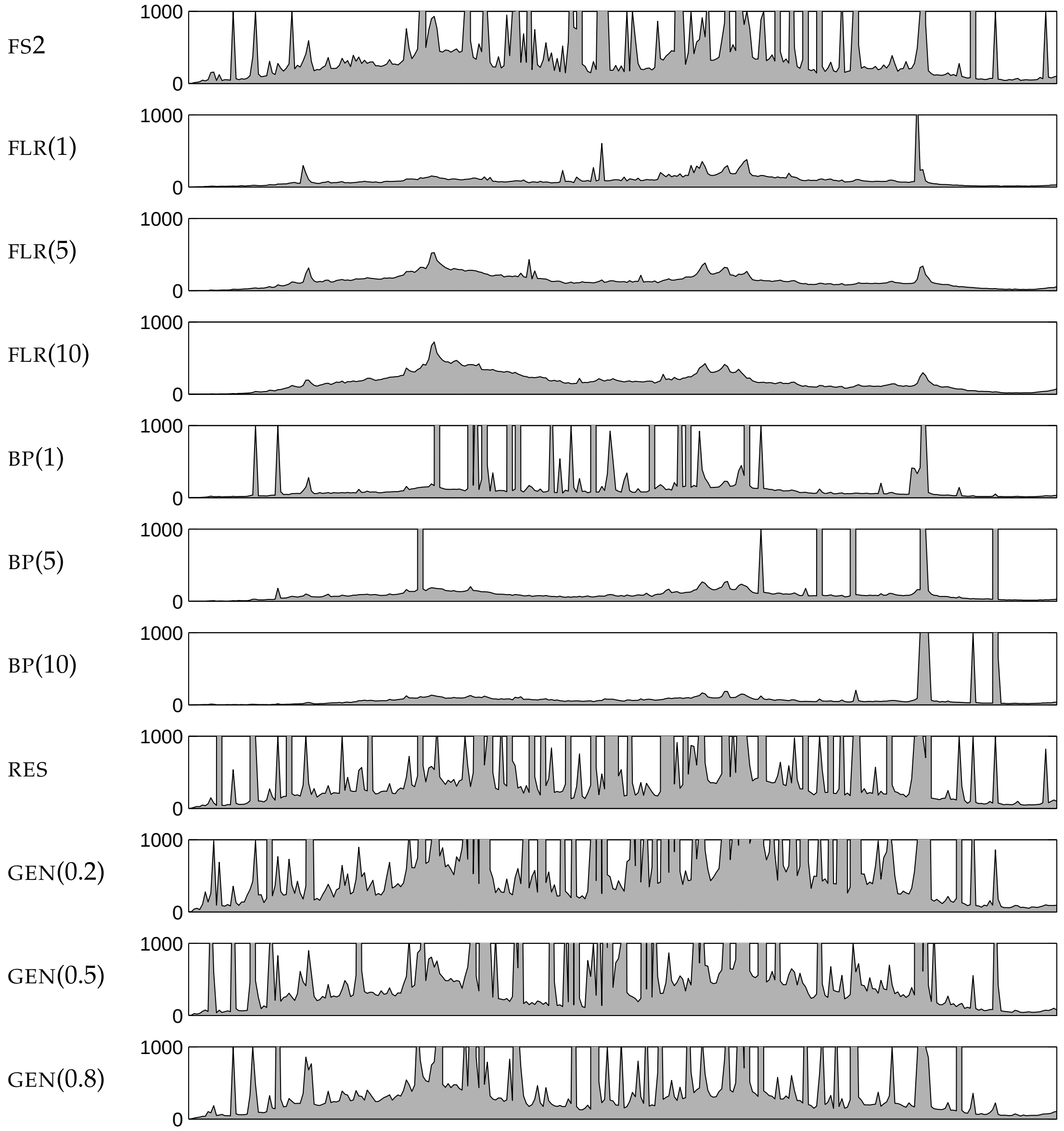


Figure 6.4: NEES for the dense environment.

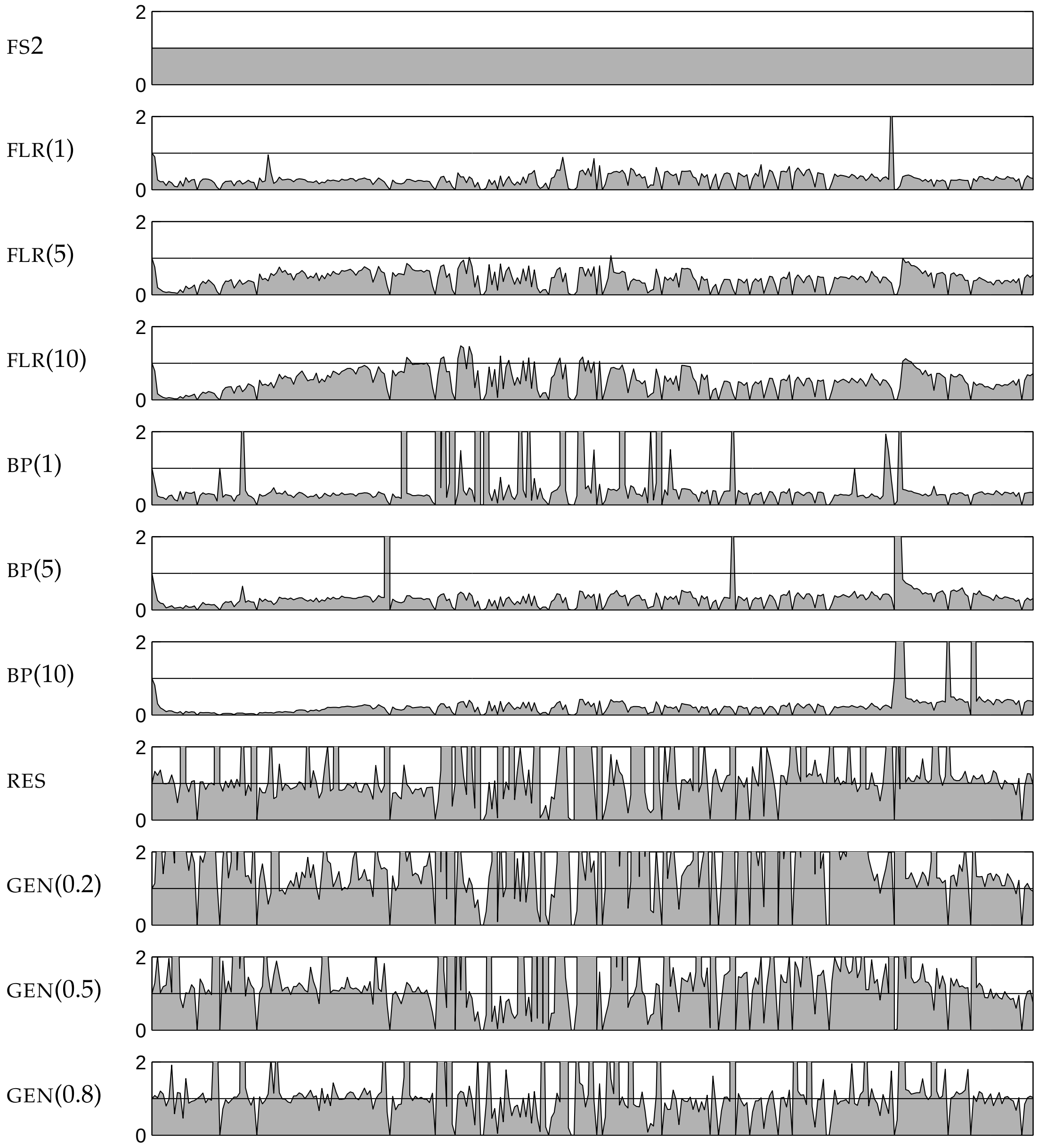


Figure 6.5: Ratio of NEES to that of FastSLAM 2 for the dense environment.

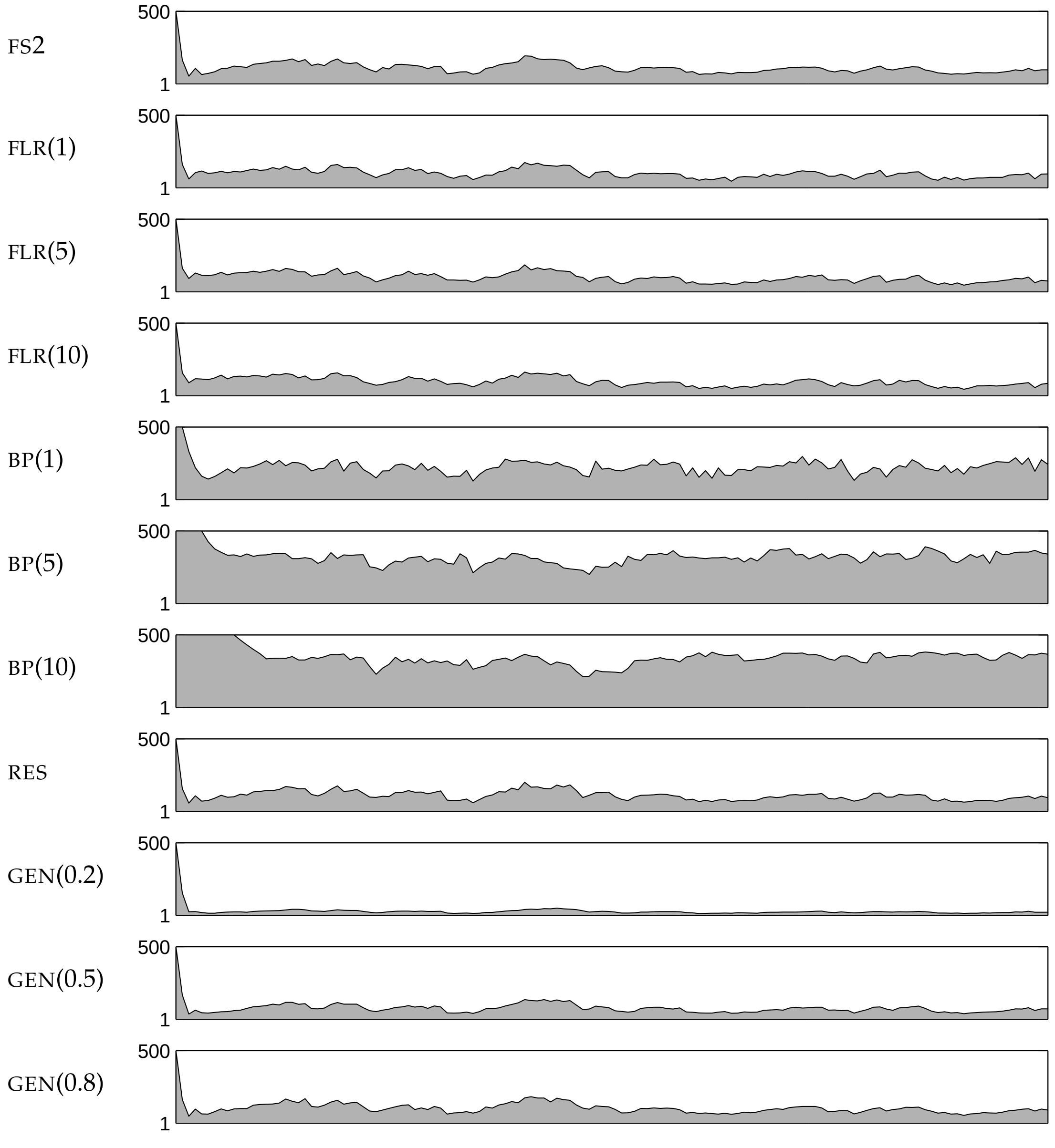


Figure 6.6: $\hat{N}_{\text{eff}}$ **for the sparse environment, versus the simulation time on the** x**-axis.**

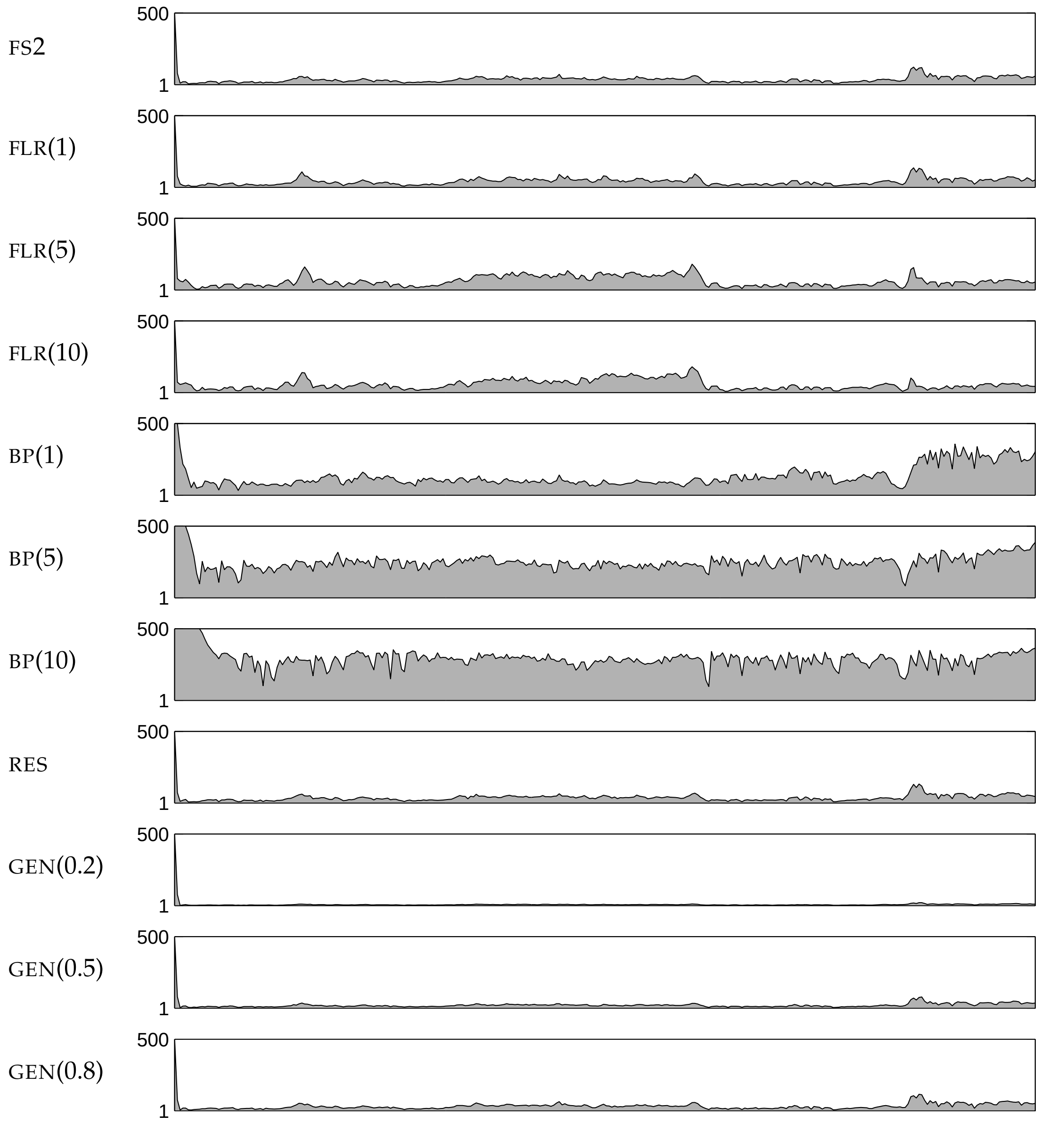


Figure 6.7: $\hat{N}_{\mathbf{eff}}$ for the dense environment.

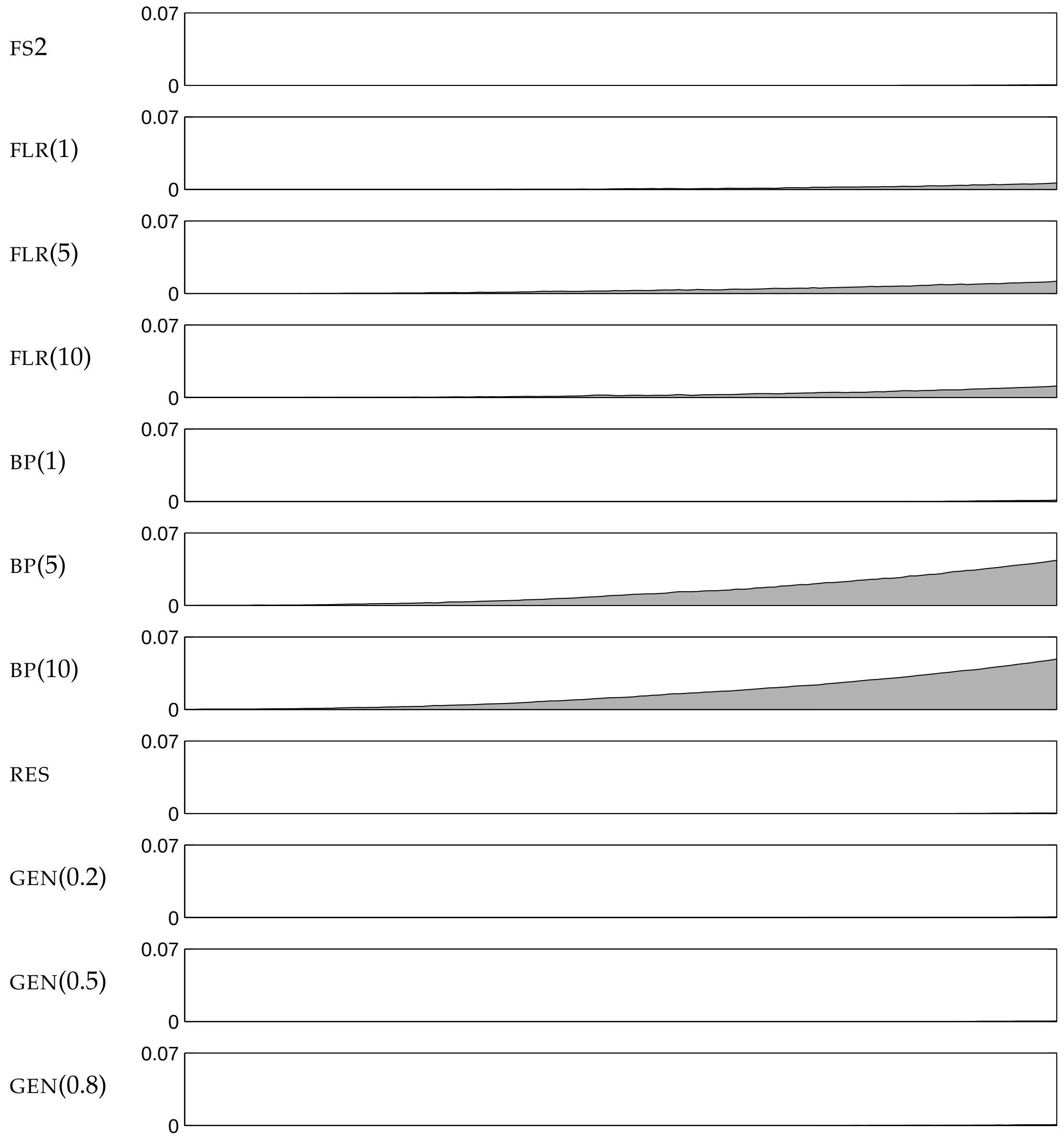


Figure 6.8: Sample variance of the final trajectories for the sparse environment. The plots shown here are computed using the trajectory samples at the end of SLAM, i.e., $\{\mathbf{x}_{1:t}^{r,i}|\mathbf{u}_{1:t},\mathbf{z}_{1:t},\mathbf{n}_{1:t}\}$.

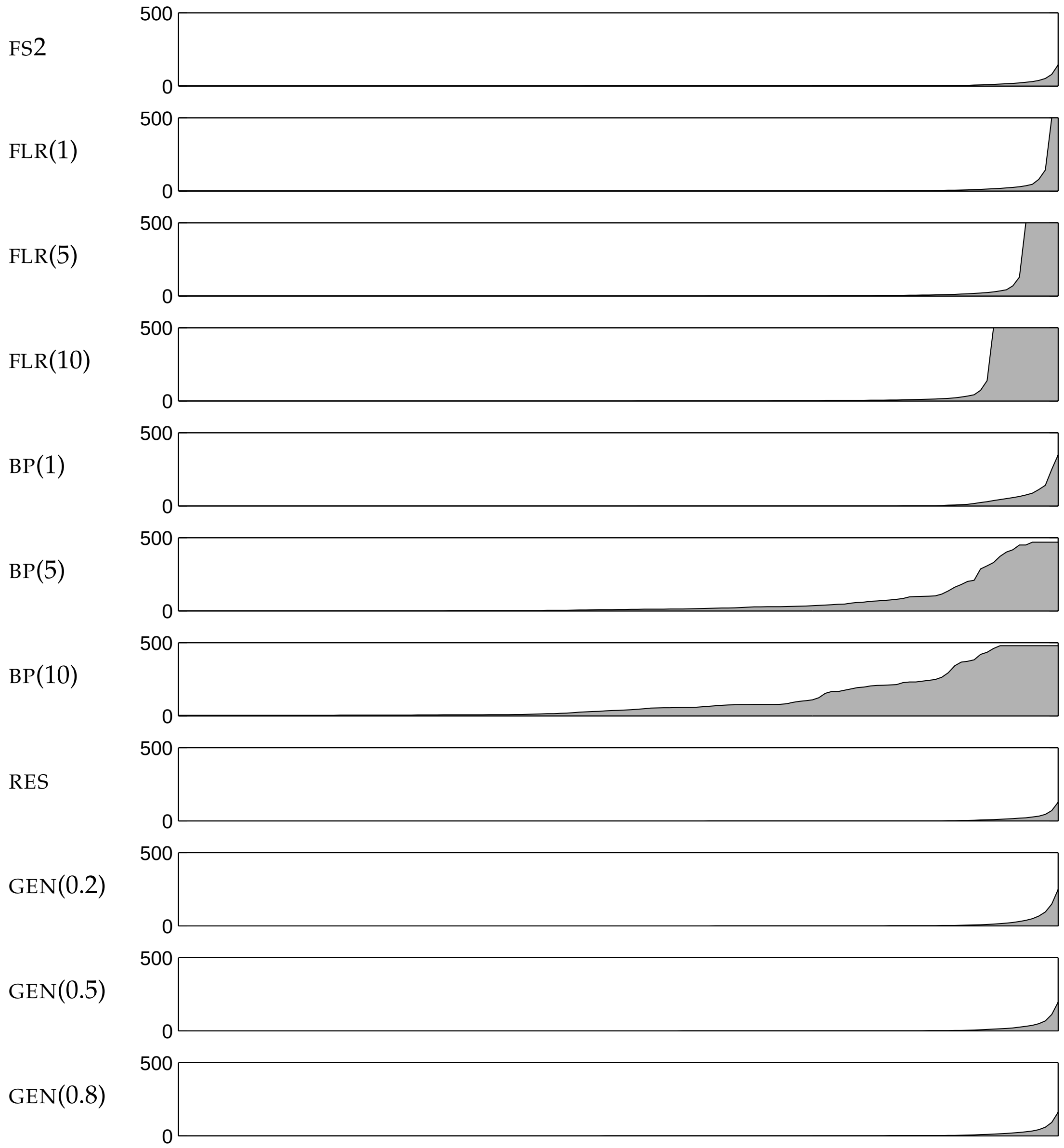


Figure 6.9: Number of unique samples representing each pose in the trajectory at the end of SLAM, for the sparse environment. Here, the x-axis is the time step for each pose in the final trajectory estimate.

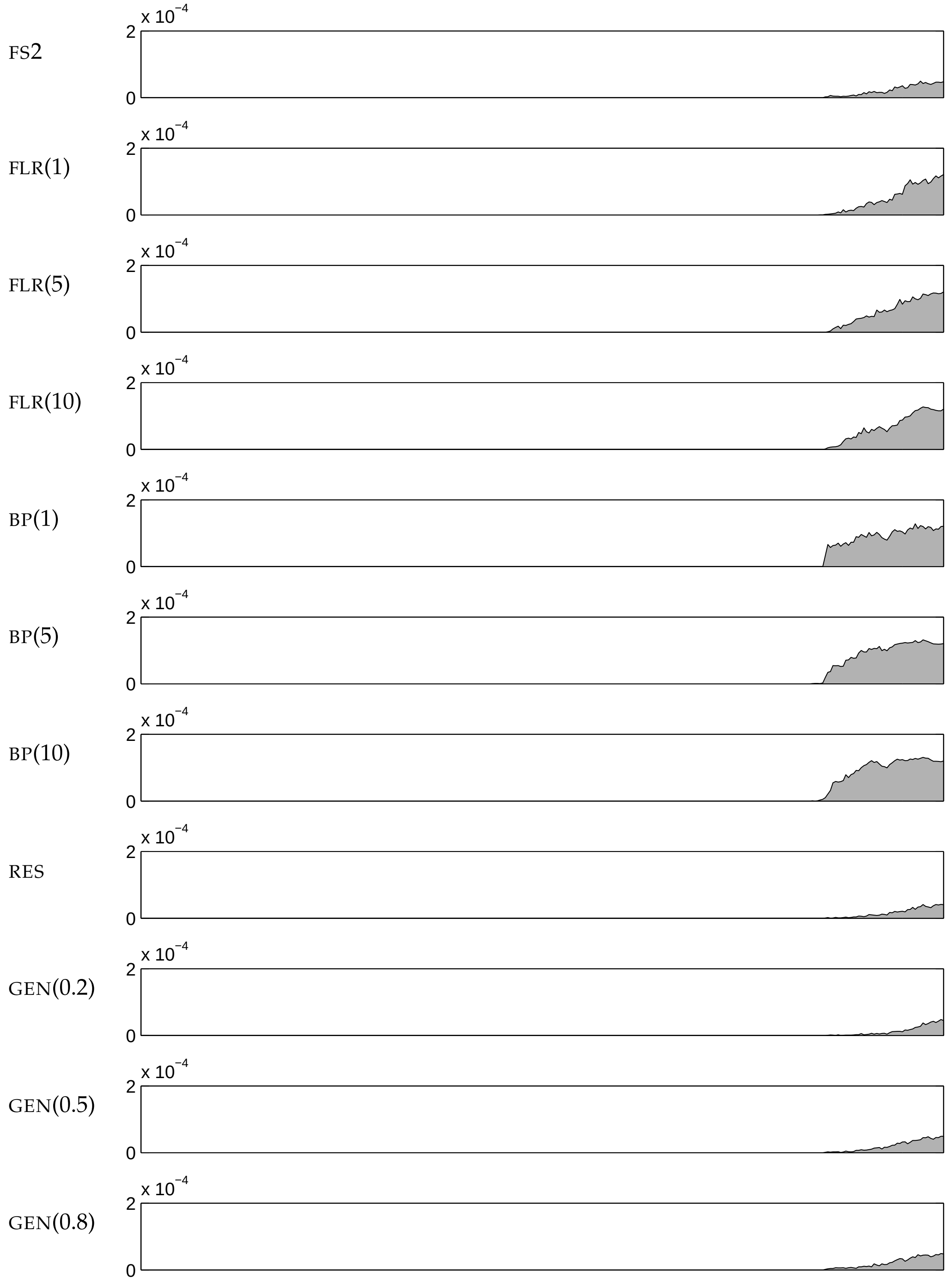


Figure 6.10: Sample variance of the final trajectories for the dense environment.

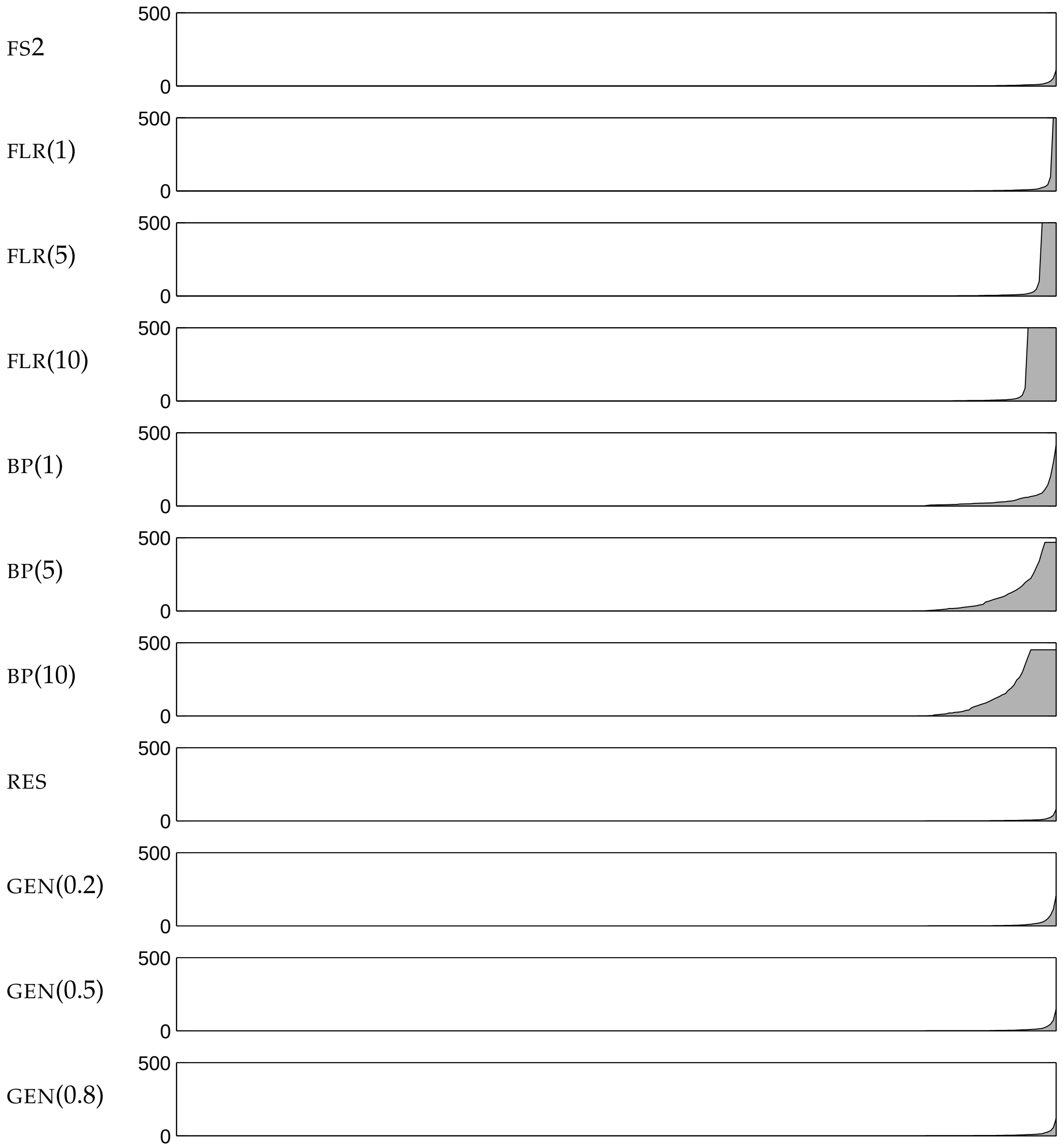


Figure 6.11: Number of unique samples representing each pose in the trajectory at the end of SLAM, for the dense environment. Here, the x-axis is the time step for each pose in the final trajectory estimate.

7 PRACTICAL SLAM FOR LOW-COST ROBOTS

In practice, restrictions on the sensing capabilities of a robot are likely to come hand-in-hand with limitations on the available computation. For example, just as a reasonably inexpensive consumer robot cannot be equipped with a costly scanning laser rangefinder, neither can it be outfitted with an expensive and power-draining full-blown, modern computer and large amounts of storage. This chapter of the thesis shows that an implementation of SLAM on a very minimal real robot is feasible despite computational restrictions. We describe a fixed-point implementation of the multiscan SLAM algorithm from Chapter 3 on an 8-bit, 16 MHz Atmel ATMEGA64 microcontroller with 128 KB storage. To our knowledge this is the first complete and working SLAM implementation on a microcontroller. The implementation is tested with simulated and real data and runs in on the ATMEGA64 with relatively fast execution times. Our results show that SLAM can be done on the current generation of basic embedded microprocessors, opening the way for inexpensive, mass-produced robots capable of mapping their environments.

7.1 Overview

For SLAM to be deployed on real inexpensive robots, algorithms that are robust enough to deal with minimal sensing but that are simple enough to be deployed with minimal computing and storage must be developed. In the previous chapters, we have focused mainly on sensing limitations and developed algorithms that we have implemented on modern PC hardware. For applications where sensing capabilities are restricted, e.g., because of cost, it is likely that the robot in question will also be handicapped in its processing and storage capabilities. For example, a robot meant to be sold inexpensively on the mass market may be equipped with only a microcontroller and a few kilobytes of memory.

In this chapter we describe a fleet of five robots, dubbed "Ratbots" (collectiveley, the "Rat Pack"), named Rat, Pig, Goat, Zebra, and Angry Bob, after characters in the "Pearls Before Swine" comic strip (Pastis, 2003). The Ratbots, designed and built at Rensselaer Polytechnic Institute from 2004–2006, are equipped with extremely limited sensing and computing, but can be built for only a few hundred US$ (and likely much less in large quantities). We have implemented a version of particle filtering SLAM that runs on board the Ratbots. The source code for our implementation is available (see Appendix C).

N_L	left encoder count	c	encoder conversion factor	r^-	IR minimum range
N_R	right encoder count	b	wheel base	r^+	IR maximum range

Notation for this chapter

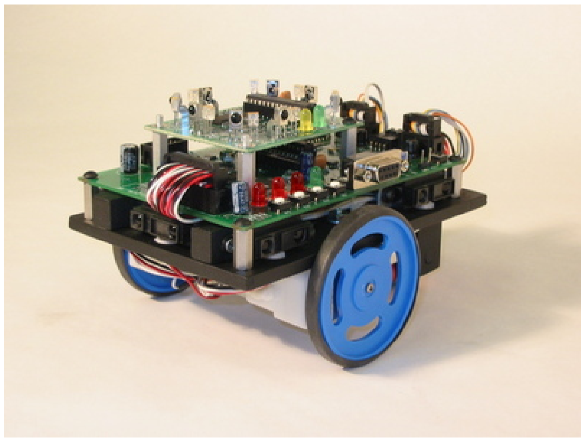


Figure 7.1: The Ratbot platform.

In the next section, we introduce the Ratbots, detail their hardware capabilities, and describe a simple kinematic model. Next, in Section 7.3, we describe some of the interesting details of our particle filtering SLAM implementation that enable the filter to run at reasonably fast speeds on a simple microcontroller. Finally, in Section 7.4, we describe the results of experiments with the implementation that indicate the feasibility of SLAM on the Ratbots.

7.2 The Ratbot platform

The Ratbots are small differential drive mobile robots. They were designed and built in the Algorithmic Robotics Laboratory at RPI, as a simple and inexpensive mobile platform for use in mobile robotics and sensor network applications. A complete Ratbot consists of several hundred US$ in parts, including:

- An Atmel ATMEGA64 8-bit, 16 MHz microcontroller (Atmel Corp., 2006) as the main computer. The ATMEGA64 uses a Harvard architecture, i.e., separate program and data memory, and has 64 KB of nonvolatile flash program memory and 4 KB of on-chip SRAM, and 60 KB of external SRAM.
- A PIC 16F873 20 MHz microcontroller (Microchip Technology, Inc., 2001) dedicated to motor control and encoder counting
- A Honeywell HRF-ROC09325XM 19.2 Kb/sec. radio module (Honeywell, 2003) with approximate range of 30 m (indoors)
- Five Sharp GPD12 infrared rangefinders (Sharp Corp., 2005) with minimum 10 cm and maximum 80 cm range
- Two US Digital E4 256 CPR / 1024 PPR encoders (US Digital, 2006)

Figure 7.1 shows a photograph of a Ratbot, with an additional “robot detector” module mounted on top of the main board. The overhead schematic in Figure 7.2 depicts the layout of the components relevant to mapping. There are two side-facing infrared rangefinders on each side of the robot, and an additional forward-facing rangefinder. The wheel axle and robot reference frame are slightly forward of the center of the actual platform, and the back of the robot is additionally supported by a roller wheel caster. Table 7.1 gives the measured values for the physical parameters shown in the schematic, along with some other parameters relevant to mapping.

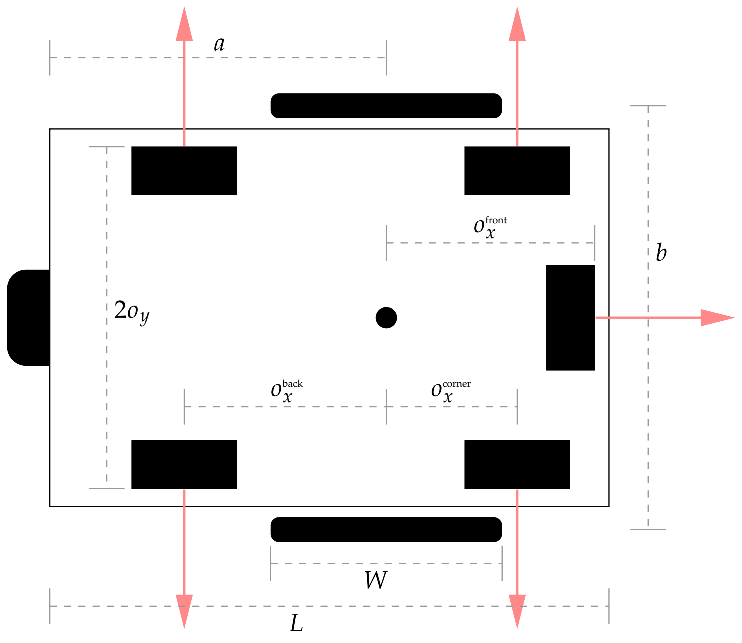


Figure 7.2: Physical Ratbot parameters.

Parameter	Explanation	Measured value	
L	chassis length	15.7	cm
W	wheel diameter	6.8	cm
a	chassis back x-offset	9.45	cm
b	wheel base	11.9	cm
o_y	side IR y-offset from centroid	4.8	cm
o_x^{back}	back IR x-offset	5.675	cm
o_x^{corner}	corner IR x-offset	3.675	cm
o_x^{front}	front IR x-offset	5.85	cm
C	encoder pulses per revolution	1024	
c	encoder conversion factor	$\pi W/C$	
r^-	IR minimum range	10	cm
r^+	IR maximum range	80	cm

Table 7.1: Physical Ratbot parameter values.

7.2.1 Kinematic model

The values from Table 7.1 and sampled encoder counts can be used to write the kinematic model for the Ratbots using a reasonably standard odometry based differential drive model (see, e.g., Borenstein et al. (1996)). For an input configuration $\mathbf{x}_{t-1}^r = [x_{t-1} \;\; y_{t-1} \;\; \theta_{t-1}]^T$ and encoder data $\mathbf{u}_t = [N_L \;\; N_R]^T$ (with N_L and N_R being the left and right encoder counts, respectively), the new configuration $\mathbf{x}_t^r$ is computed as:

$$\mathbf{x}_t^r \overset{g}{\leftarrow} \mathbf{x}_{t-1}^r + \begin{bmatrix} \left(\frac{cN_R+cN_L}{2}\right) \cos\left(\theta_{t-1} + \frac{cN_R-cN_L}{2b}\right) \\ \left(\frac{cN_R+cN_L}{2}\right) \sin\left(\theta_{t-1} + \frac{cN_R-cN_L}{2b}\right) \\ \frac{cN_R-cN_L}{2b} \end{bmatrix} \tag{7.1}$$

To perform SLAM prediction steps we need the derivatives of (7.1) with respect to $\mathbf{x}_{t-1}^r$ and

$\mathbf{u}_t$, which are, respectively:

$$\mathbf{H_x} = \frac{\partial g}{\partial \mathbf{x}^r_{t-1}} = \begin{bmatrix} 1 & 0 & -\left(\frac{cN_L+cN_R}{2}\right)\sin\theta_t \\ 0 & 1 & \left(\frac{cN_L+cN_R}{2}\right)\cos\theta_t \\ 0 & 0 & 1 \end{bmatrix} \tag{7.2}$$

and:

$$\mathbf{H_u} = \frac{\partial g}{\partial \mathbf{u}_t} = \begin{bmatrix} \frac{c}{2}\cos\theta_t - \frac{c^2N_L+c^2N_R}{4b}\sin\theta_t & \frac{c}{2}\cos\theta_t - \frac{c^2N_L+c^2N_R}{4b}\sin\theta_t \\ \frac{c}{2}\sin\theta_t + \frac{c^2N_L+c^2N_R}{4b}\cos\theta_t & \frac{c}{2}\sin\theta_t + \frac{c^2N_L+c^2N_R}{4b}\cos\theta_t \\ -\frac{c}{2b} & \frac{c}{2b} \end{bmatrix} \tag{7.3}$$

(Note the use of the new orientation θ_t as computed in (7.1).)

In our implementation the left and right encoder counts were assumed to be independent with variance proportional to the magnitude of the count. We used the experimentally determined value of $\sigma_N = 0.03N$, i.e., 3% of the reported encoder count.

7.3 Implementation details

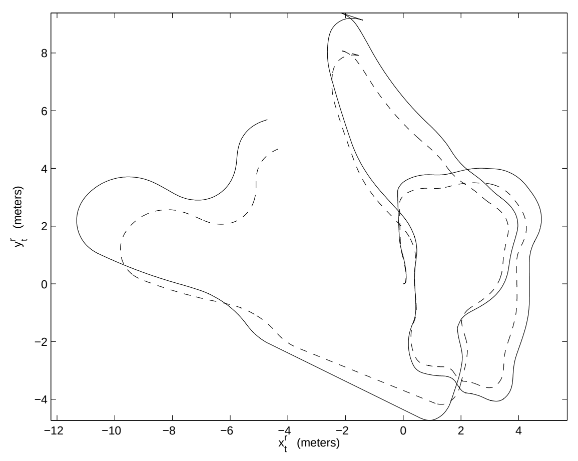
We implemented the basic Rao-Blackwellized particle filter for SLAM onboard the Ratbot platform, using the multiscan approach described in Chapter 4. (For background on RBPF SLAM see Appendix B.) While improvements such as those described in Chapters 5-6 would lead to better maps, implementing them in an extremely resource-constrained environment such as the Ratbot platform is difficult because of book-keeping overhead. Furthermore, our focus was on a simple proof-of-concept to show that SLAM on a microcontroller is indeed feasible.

7.3.1 Fixed point SLAM

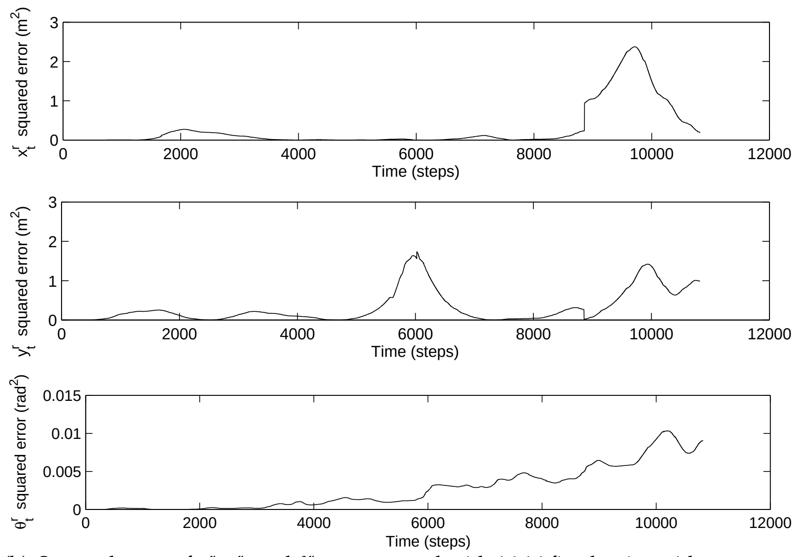
The ATMEGA64, like most microcontrollers, has no hardware floating point capabilities. This is a severe disability for implementing SLAM, which requires *en masse* processing of large amounts of real-valued data. Floating point operations can be emulated in software but the resulting code is prohibitively slow for real-time SLAM implementation.

An alternative is to trade off some accuracy, in exchange for efficient storage and fast operations with real-valued numbers, by using a fixed point number representation. In our implementation we employed a "16.16" representation, with 16 bits representing the whole part, and also 16 bits representing the fractional part. The range of the resulting numbers is thus $[-32767.999985, 32767.999985]$ with a resolution of 1.52588×10^{-5}. The ATMEGA64 operates natively on 16-bit integers so there is some overhead in using the 32-bit fixed point representation. Unfortunately, the magnitudes of the numbers that appear in SLAM vary widely, from particle weights (which can be very small), to statistical distances (which can be very large), to landmark parameters (which depend on the dimensions of the environment and the units of measure). Thus, a representation using only 16 bits, e.g., 6.10 or 8.8, is insufficient. In some cases it may be desirable to use customized fixed point representations for different purposes in SLAM, but for simplicity our implementation used 16.16 numbers throughout. Another potential enhancement is to re-work the units in SLAM computations like the motion model or feature extraction to obtain numbers that fit well into a particular number representation. However, our implementation is mostly a direct translation of particle filtering into the 16.16 representation.

The limited resolution of fixed point number representations introduces error into the SLAM computations beyond that of normal SLAM implementations based, typically, on double-precision floating point numbers. While we do not perform a complete analysis of the effects of this error, our experiments have shown that the error due to limited precision is mostly absorbed by the SLAM filtering process (e.g., it can be treated as odometry error). For example, Figure 7.3 shows a trajectory computed from raw odometry data using both double precision and 16.16 fixed point numbers. Clearly, the fixed point computations introduce some error, but for this particularly lengthy trajectory the fixed point pose does not significantly diverge from the double



(a) Computed trajectories for the same raw odometry data. Solid line: trajectory computed using double precision floating point numbers. Dotted line: trajectory computed using 16.16 precision fixed point numbers.



(b) Squared error of x_t^r, y_t^r, and θ_t^r as computed with 16.16 fixed point with respect to those computed with double precision floating point.

Figure 7.3: Comparison of predicted trajectories obtained using double precision floating point computations and 16.16 fixed point computations.

precision computed pose. Without analyzing the filter implementation to determine the precise effects of this error, we can still accommodate it in practice by slightly inflating the odometry model uncertainty.

7.3.2 Computational shortcuts

In addition to the considerations described above, our RBPF implementation on the ATMEGA64 employed a number of simple shortcuts and other techniques to reduce computation and memory demands without requiring significant sacrifices in filtering accuracy. Some of the shortcuts are relatively standard for implementing intensive algorithms on limited hardware:

- Lookup tables were used for trigonometric functions such as sine, cosine, and tangent, and computation of arctangents was done via a binary search on the tangent table. In our imple-

Start	End	Bytes	Usage
0x0000	0x1100	4096	Internal register and SRAM (stack, dynamic memory)
0x1100	0x11A4	165	Pose history mean (m poses)
0x11A5	0x12EE	330	Pose history covariance (m poses)
0x12EF	0x1514	550	Multiscan ranges ($5 \times m$ ranges)
0x1515	0x43FF	12011	Working buffers (feature extraction, etc.)
0x4400	0xF000	44032	Particle data

Table 7.2: Fixed-point SLAM memory allocation.

mentation the trigonometry tables had an angular resolution of 0.45 degrees. Lookup tables were also used in an implementation of the Box-Muller transformation (Box and Muller, 1958) for generating Gaussian random numbers. All of the lookup tables are stored in program memory (on the ATMEGA64's flash) to conserve space in RAM.

- Rather than employing generic implementations of linear algebraic operations like matrix multiplication and inversion, the operations were written out from scratch. Doing so enabled substantial savings in memory (e.g., covariance matrices are symmetric) and computation (by reordering operations where appropriate, sharing terms, exploiting multiplications by zero, etc.). This approach is especially feasible in the context of particle filtering SLAM since nearly all matrix operations are with low-dimensional (2×2 or 3×3) matrices.

Other shortcuts we employed are more specific to the context of particle filtering SLAM:

- Correspondence tests were performed only after thresholding to obtain landmarks within the sensing range of the robot.
- While our other implementations performed full maximum likelihood estimation of line feature parameters and covariances (see, e.g., Algorithm 4.3 on page 52), we used simple unweighted least squares for parameter estimation in the ATMEGA64 implementation. Covariances were estimated using the closed-form formulae described by Pfister et al. (2003).
- An important computational advantage of the multiscan approach described in Chapter 4 is that, since SLAM updates are performed only every m time steps, the update calculations can consume more than one SLAM time step of CPU time. On slower processors, this allows for simultaneous accumulation of multiscan data and processing of the data from the preceding multiscan.
- We did not implement an efficient (but complex) tree-based map management scheme (Montemerlo, 2003) on the ATMEGA64. It was thus necessary to minimize the cost of resampling, which requires significant copying of particle data. Algorithm 7.1 shows a simple in-place random resampling technique. When a particle index j is "selected" during resampling, a counter c_j for that index is incremented. Particles with $c_j = 1$ are left as they are. Particles with $c_j > 1$ are copied $c_j - 1$ times into the storage of particles with $c_j = 0$. An important advantage of this technique is that no temporary copies of particles need be made, so the entire memory can be allocated to active particles.

7.3.3 Memory usage

In our implementation, the 4 KB on-chip SRAM was used for the stack and for dynamic memory allocation. Lookup tables were stored in the 64 KB program memory. The remaining 60 KB of (slower) external SRAM was used for working buffers and particle data. The memory allocation was divided as shown in Table 7.2 (assuming the multiscan size parameter $m = 55$).

The particles were pre-allocated with sufficient storage for up to 125 landmarks, with each landmark requiring 22 bytes (11 16-bit fixed point numbers: two for line parameters, three for

Algorithm 7.1 In-place random resampling

```
1:  $c_j \leftarrow 0$ for all $j \in 1 \ldots N$                    // initialize copy-counts to zero
2:  for $i = 1 \ldots N$ do                                          // select N particles
3:    Select $j \in [1, N]$ w.p. proportional to $\omega_j$
4:    $c_j \leftarrow c_j + 1$                                       // increment copy-count for selected particle
5:  end for
6:  $z \leftarrow 1$
7:  for $i = 1 \ldots N$ do
8:    while $c_i > 1$ do                                             // repeat until copy-count is at most one
9:      while $c_z \neq 0$ do                                        // find next unselected particle
10:       $z \leftarrow z + 1$
11:     end while
12:     $\phi_z \leftarrow \phi_i$                                   // replace unselected particle with a copy of a multiply-selected one
13:     $c_i \leftarrow c_i - 1$                                     // decrement copy-count of multiply-selected particle
14:     $z \leftarrow z + 1$
15:   end while
16: end for
```

covariance information, and six for line segment extent parameters). With some extra overhead, each particle required 3088 bytes of storage, leaving enough space for around 13-14 particles. The maximum number of landmarks can be traded off with the number of particles.

7.4 Experiments

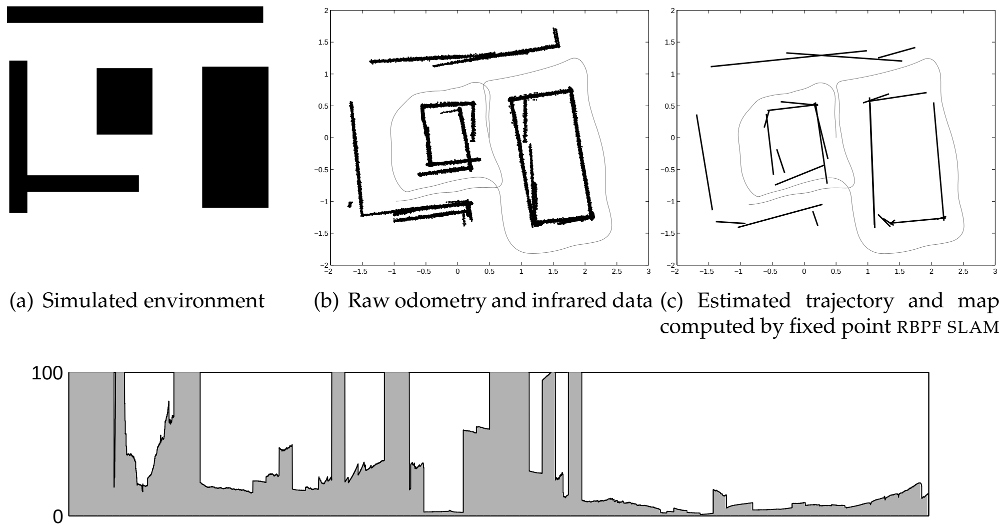
This section describes the results of some simple proof-of-concept experiments with our fixed point RBPF SLAM implementation. We tested the implementation by building maps with both simulated data, for which ground truth information was available, and with real data collected from the Ratbots.

7.4.1 Simulated data

Figure 7.4 shows the results of fixed point RBPF SLAM on a simulated dataset. The primary purpose of this experiment was to test the fixed point SLAM implementation using data for which ground truth information was available. The SLAM code was run on a PC (not on a Ratbot) for this experiment.

The simulator was a version of the `mpro` software (see Appendix C), modified to simulate the Ratbot platform. The simulated trajectory was 15.5 m long with a total rotation of 21 rad. The fixed point particle filter used $N = 100$ particles. As the simulator generated scan data at a fairly high frequency relative to the robot's velocity (about 708 frames of data per meter of travel), a large multiscan size of $m = 180$ was used.

The fixed point SLAM implementation successfully closed a fairly large loop in the course of the experiment. However, as with the results in earlier chapters, the use of simple nearest-neighbor data association resulted in a map containing several spurious landmarks. Figure 7.4(d) shows the normalized estimation error squared (NEES) for the fixed point particle filter trajectory estimator with respect to the ground truth trajectory. (For an explanation of NEES and several other NEES plots for comparison, see, e.g., Chapters 5 and 6.) Because only a basic particle filter was implemented — without enhancements like the improved proposal distribution or the techniques of Chapters 5–6 — estimation degeneracies occurred somewhat frequently. Despite this, the filter was able to consistently build a reasonably correct map, closing the loop properly in 91 out of 100 trials.


(a) Simulated environment (b) Raw odometry and infrared data (c) Estimated trajectory and map computed by fixed point RBPF SLAM

(d) Normalized estimation error squared (NEES) of the trajectory for the fixed-point estimator

Figure 7.4: Simulated experiment for `sim-world1`.

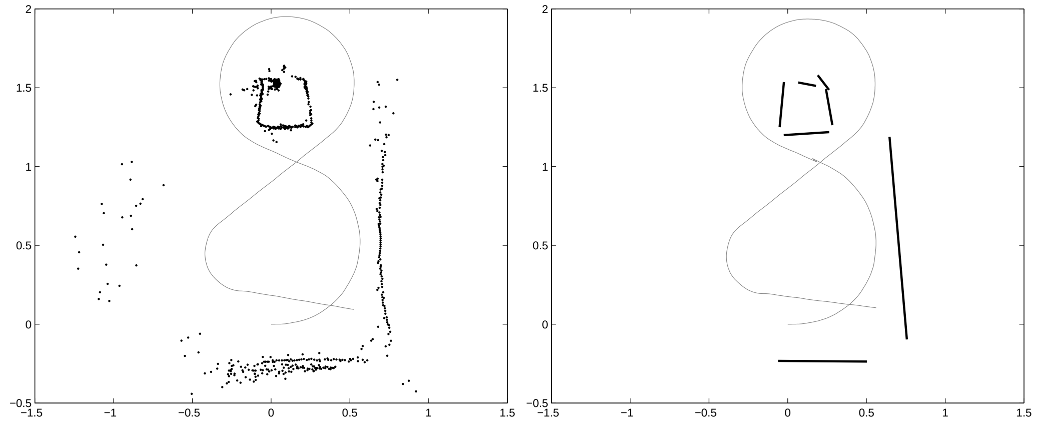

(a) (b)

Figure 7.5: (a) Ratbot dataset and; (b) the resulting map and trajectory produced by the fixed point SLAM implementation.

7.4.2 Ratbot data

We performed a small experiment using a Ratbot that followed a pre-planned trajectory. A dataset was collected using the Ratbot, and then fed (offline) to the SLAM code running on the ATMEGA64. The Ratbot traced a figure eight in a 2 m $\times$ 2.5 m area with a partial perimeter wall and a small rectangular obstacle. The experiment lasted approximately 35 seconds, with a trajectory length of about 6.25 m and a total rotational change of about 11.75 rad. In this experiment, only the data from a single infrared sensor (the back right sensor) was used to build a map. Figure 7.5(a) shows the raw odometry estimate and infrared data points for the experiment, and Figure 7.5(b) shows the trajectory and line segment map obtained using the fixed point SLAM implementation.

The experiment used $N = 13$ particles, with multiscans consisting of $m = 55$ individ-

Task	Total time	Iterations	Time per iteration
Motion model	285.19 sec	861	0.33 sec
SLAM update	3.62 sec	15	0.24 sec

Table 7.3: Fixed-point SLAM execution time breakdown for an experiment running on the ATMEGA64.

ual scans. (The sensing frequency was much lower than in the simulated experiment — about 138 frames of data per meter of travel.) While the odometry data was fairly accurate, the infrared data was very sparse. The SLAM implementation successfully closed a loop during the experiment, properly associating features corresponding to the wall at the bottom of the map.

The fixed point SLAM implementation was executed on the Ratbot's ATMEGA64 processor to obtain timing information, which is summarized in Table 7.3. The entire SLAM process for the experimental data from Figure 7.5 required 289 seconds to produce the resulting trajectory and map. The data consisted of 861 frames of infrared measurements and encoder counts. Using the multiscan approach with $m = 55$, 15 SLAM updates were performed, but the filter prediction step was applied for every frame of encoder data. Thus, the prediction step consumed about 285 seconds of the total running time, or 0.33 seconds per time step. The RBPF updates consumed 3.62 seconds in total, or 0.24 seconds per update, for the small map shown in Figure 7.5(b). The intensiveness of the prediction computations, due mainly to the many multiplications required to obtain the covariance matrix for the motion model for every particle, are the main obstacle to deploying the SLAM implementation in real time. (There were about 500 fixed point multiplications per particle per prediction step.) Simplifying the motion model (e.g., using a uniform distribution rather than a Gaussian) would likely reduce the prediction cost to the point of enabling real time implementation, at the cost of some reduction in filtering accuracy.

7.5 Summary

In this chapter we have described an implementation of particle filtering SLAM with multiscans on a simple robotic platform. The robot's sensing consists of a small array of infrared rangefinders, and SLAM is implemented on an Atmel ATMEGA64 8-bit microcontroller with 128 KB total memory. The particle filter is implemented using fixed point numbers since the processor lacks floating point capability. We have also described a number of simple implementation shortcuts that make SLAM on a microcontroller computationally feasible without significantly sacrificing estimation quality. The resulting particle filter executes on board the robot. While the execution is not in real time, it is fast enough to prove that with computers in the same class, or with slightly more optimization, SLAM can be implemented in real time on commodity hardware. Our experiments with simulated and real data show that the fixed point particle filtering implementation can successfully build maps with range data from a small infrared array.

Our implementation shows that SLAM is feasible for inexpensive robots built from off-the-shelf parts, such as the Ratbots described in this chapter. Despite restrictions on sensing, computation, and storage, the multiscan particle filter is simple and efficient enough to yield positive results. Future work will focus on applying some of the other SLAM improvements described in this thesis, such as the constraints of Chapter 5 and the sampling improvements of Chapter 6, to the Ratbot and similar platforms. As with the implementation described in this chapter, the primary challenge is in simplifying the techniques so they can be feasibly applied despite limited computation, without sacrificing too much estimation accuracy.

8 CONCLUSIONS

This thesis has examined the robot mapping problem in several contexts. Primarily, the thesis has proposed mapping algorithms for robots with limited sensing capabilities. It has also examined theoretically the sensing requirements for mapping; described algorithms that improve mapping not just with limited sensing, but also for the well-equipped robots more common in robotics research; and discussed an implementation of a simultaneous localization and mapping algorithm on inexpensive robots with limited computing and storage.

In this chapter, we will review the specific contributions of the thesis, and then address future challenges in mapping, particularly with respect to sensing and computational limitations. Lastly, we offer some final thoughts on the state of mapping research and its practical applications.

8.1 Contributions

As described in Section 1.6, this thesis has made contributions to robot mapping research in several contexts.

8.1.1 Theoretical contributions

Chapter 3 employed a novel approach to examine the theoretical sensing requirements for mapping and the relative mapping abilities of different sensors. The chapter developed a generalized model of mapping sensors that incorporates aspects such as range, spatial resolution, beam width, noise, and more. The model is detailed enough to encapsulate most of the range-bearing sensors used to build maps. Coupled with a generic occupancy grid mapping technique, and several assumptions on the trajectory of the robot, the correlation between sensor observations, and structure in the environment, the model was used to obtain bounds on the number of observations of each cell in the grid, and on the maximum likelihood map error, in terms of the sensor characteristics. This is in contrast to other investigations of the sensing requirements for tasks like localization, which have instead focused on designing custom mapping algorithms for every sensor configuration.

We used the bounds to examine the influence of different sensor parameters on the quality of the maps produced with the sensor. In addition, we employed the bounds in comparing the theoretical mapping capabilities of several realistic sensors, including a scanning laser rangefinder and different configurations of SONAR and IR sensor arrays.

8.1.2 Algorithms for mapping with limited sensing

The main contribution of the thesis is a set of particle filtering algorithms for simultaneous localization and mapping with sparse arrays of range sensors, such as infrared rangefinders. While particle filtering SLAM algorithms have gained wide acceptance in the SLAM community in recent years, the standard techniques are poorly suited to robots with sparse sensing, which can only partially observe features of the environment at any given time. Furthermore, current particle filtering SLAM algorithms, while easy to implement, do not take advantage of prior knowledge or exploit other techniques for managing uncertainty and improving estimation accuracy.

In Chapter 4, we developed a simple particle filtering algorithm for mapping with sparse sensing. The algorithm groups multiple consecutive sensor scans into "multiscans," and then performs feature extraction as if the multiscan data came from a single sensor scan. This approach exploits the motion of the robot to improve the observability of features, with the tradeoff that uncertainty in the motion contributes to feature extraction error. A major difficulty of using a particle filtering algorithm with multiscans is that a full implementation must perform feature extraction separately for every particle. We instead described an approximation that extracts features only once, and transforms them for each particle. Furthermore, using a simple clustering technique, the accuracy of the approximation can be traded off with its efficiency.

If prior knowledge about the environment is available, it can be exploited to reduce uncertainty and improve the map estimation process. In Chapter 5, we described a new particle filter based algorithm that incorporates prior knowledge in the form of constraints on landmarks in the map. For example, if it is known that two walls in the environment are rectilinear with respect to each other (e.g., indoors), the algorithm enforces the rectilinearity as a constraint on the landmark parameters. The filter does inference in the space of constraints to determine if landmarks should be constrained. One of the main challenges we addressed was to develop an efficient method for enforcing constraints on sampled maps in a particle filtering scheme, using a novel technique for modifying the maps upon inferring that landmarks should be constrained.

Particle filtering algorithms for SLAM are inherently sequential Monte Carlo techniques, and a number of advances in SMC have appeared in the statistics literature in recent years. Inspired by some of these advances, Chapter 6 developed two new particle filtering SLAM algorithms: fixed-lag roughening and the block proposal distribution. The main idea behind both techniques is to take advantage of "future" information to improve the estimation of the robot's trajectory (and the map) over a fixed lag time, as opposed to using the current sensor measurements only in estimating the current pose. Fixed-lag roughening incorporates a Markov Chain Monte Carlo move step to update particles based on future information, and the block proposal directly samples poses over the fixed lag from their joint distribution conditioned on all the available measurements. In addition to developing these two techniques, Chapter 6 also examined the effectiveness of two alternative particle resampling algorithms.

Importantly, the ideas described in Chapters 5 and 6 are applicable not only in the limited sensing case, but also in more typical SLAM scenarios, where the robots are equipped with high-fidelity sensors such as scanning laser rangefinders.

8.1.3 Practical mapping for inexpensive robots

For SLAM to be useful outside of the research community on a large scale, we must develop practical algorithms that can be implemented on reasonably inexpensive hardware, with sensing and computational limitations. Chapter 7 described an implementation of a basic particle filtering SLAM algorithm with multiscans, on board a very restricted platform called the Ratbot. The Ratbot is equipped with a five-sensor infrared rangefinder array, an 8-bit, 16 MHz main computer, and a total of 128 KB of storage. Implementing SLAM in this environment required a number of tradeoffs. Our implementation exploited a fixed point number representation and a variety of simplifications to improve computational efficiency without significantly sacrificing accuracy. To our knowledge, ours is the first working SLAM implementation on a microcontroller.

8.2 Future directions

A principle aim of this thesis has been to bridge the gap between modern techniques for mapping, which typically rely on high-fidelity sensing and computing, and practical deployment using hardware suitable for use in inexpensive robots. Toward this end we have addressed theoretical issues, and also developed algorithms which work with practical inexpensive sensors such as small infrared rangefinder arrays. While this thesis has shown that mapping with limited sensing (and to some extent, limited computing) is possible, a number of questions remain to be answered before mapping can be realistically deployed both inexpensively and reliably.

8.2.1 Sensing requirements for mapping

Chapter 3 addressed the problem of determining the sensing requirements for mapping. However, in doing so, the chapter made several assumptions that must be addressed in future work. Modifying the analysis to incorporate realistic trajectories and pose uncertainty is perhaps the most pressing issue, since until this is done, the bounds derived in our work are only somewhat applicable in realistic mapping scenarios. (We note, however, that the simulations in Chapter 3 show the difference between continuous and random trajectories to be relatively small in terms of the effect on map quality.)

Most environments in which mapping robots are deployed also exhibit some structure. Chapter 5 exploits structure in the environment to improve map and trajectory estimation, but our analysis in Chapter 3 does not account for structure. The results of our simulations using structured environments, generated using simple Markov random fields, indicate that the existence of structure has a significant effect on the observability of the environment for different sensors. Incorporating structure, perhaps in the form of MRFs, into our analysis may give us much more realistic estimates of the relative capabilities of mapping sensors.

8.2.2 Exploration

The focus of this thesis has been on *passive* mapping algorithms, rather than on *active* algorithms which direct the exploration of the robot. A completely autonomous mapping robot must, of course, implement an exploration algorithm, particularly if it is equipped with range-limited sensors. Most recent work on active mapping algorithms has focused on the information gain metric for determining exploration goals (Stachniss et al., 2005a), or on the development of other more ad-hoc cost functions for exploration (Tovar et al., 2006). While these techniques may be useful in the context of limited sensing, we believe that exploration algorithms for robots like the Ratbot must more explicitly take into account the observability of the environment by the robot's mapping sensors, rather than simply seeking to build accurate maps as quickly as possible. Future research should ask the question: what is the best exploration strategy for a given sensor? One promising approach may be to incorporate reinforcement learning techniques in the exploration strategy. Recent work by Kollar and Roy (2006) has employed reinforcement learning to learn exploration strategies for robots with different odometer characteristics.

8.2.3 Filtering algorithms

The core of most SLAM algorithms is the statistical filtering algorithm that estimates the robot's trajectory and the map. In this thesis, we have employed particle filters for SLAM, and have developed enhancements to the standard particle filtering techniques in Chapters 4-6 that improve the estimation process. Other mapping algorithms make heavy use of the extended Kalman filter (see Appendix B). Many recent approaches are hybrid in nature, using statistical filtering techniques to build local maps, and global optimization strategies for recovering the wider topology of the environment.

Chapter 6 of this thesis draws on new particle filtering techniques developed in the general statistical literature. The relationship between SLAM and the general parameter estimation problem is obvious, but the connection has been under-exploited. Future research on SLAM should continue to utilize new results in general filtering. For example, the "marginal particle filter" described by Klaas et al. (2005) may be promising in the SLAM context.

In the context of minimalist robots with limited sensing and computing, the ability to trade off efficiency and accuracy in a SLAM algorithm is paramount. Of the SLAM techniques currently in wide use, particle filters are clearly the most flexible in this sense, because the number of particles can be adjusted as necessary. Hybrid algorithms based on local maps are also reasonably flexible, since the maximum size of a local map can be tailored to the available computation. Future research should investigate ways to improve the flexibility of SLAM algorithms in this sense, so the algorithms can be deployed on a wide variety of robotic platforms.

8.2.4 Practical implementations

Chapter 7 of this thesis described a simple implementation of particle filtering SLAM on a microcontroller. Simplifying SLAM algorithms so they can be deployed on such minimal computing is important, since more capable computers are prohibitively expensive and consume too much power for use in mass-produced, reasonably simple robots such as those meant for the consumer market. This issue is likely to persist at least over the coming few years, whereas the market for consumer robots is already growing rapidly. No consumer robot, however, is currently capable of SLAM.

Unfortunately, SLAM is an inherently computationally intensive process and simplifying it so it can be done with limited computing and limited sensing requires tradeoffs in accuracy. An important challenge is thus finding ways to make such tradeoffs without handicapping the SLAM algorithm to the point of failure. As algorithms for SLAM become more complicated, like those of Chapters 5 and 6 in this thesis, the computational requirements for implementations will unavoidably increase, but by carefully examining the requirements of these implementations and making simplifications as we have done in Chapter 7, the goal of robust but economical SLAM on real robots can be reached.

8.3 Final thoughts

Robot mapping is a very well-studied problem. At any large robotics conference, one could ask a room full of researchers, "Is the robot mapping problem solved?" and they would likely answer "Yes," with some conditions. In a mid-sized, uncluttered indoor environment, with a scanning laser rangefinder and sufficient computing, modern EKF-based and particle filtering SLAM algorithms can consistently produce very accurate maps.

Nevertheless, there are still important questions that must be addressed. The fundamental sensing and computational requirements for mapping remain poorly understood. In cluttered, unstructured environments like those encountered outdoors, mapping is much more difficult. And for robots equipped with only limited sensing and computing like those described in this thesis, new approaches like the multiscan technique described in Chapter 4 must be employed.

As robots move out of the laboratory and into real environments — the home, the office, hospitals, and so on — cost considerations become important to address. Practical mapping algorithms that can deal with limitations in sensing, computing, memory, and power usage will need to be employed if inexpensive, useful robots are to find their way into wider use and acceptance in the next several years.

A
SAMPLING TECHNIQUES

Many of the techniques in the proposed thesis are based on ideas from the statistics literature. Specifically, nonparametric sample-based representations of complicated PDFs are used heavily in particle filtering SLAM. This appendix briefly introduces techniques for drawing samples according to a complicated distribution. Particular applications of these techniques for mapping are described in Appendix B and in Chapters 4-6.

For a detailed introduction to the methods described below, see the book by Liu (2001).

A.1 Importance sampling

Suppose we wish to use samples to approximate some complicated distribution $p(\mathbf{x})$. There are many techniques to draw samples from $p(\mathbf{x})$, as long as the PDF can be evaluated. *Importance sampling* is perhaps the most widely employed.

The basic idea of importance sampling is to draw samples $\mathbf{x}^i$ from an easy-to-sample PDF, and then weight the samples according to their likelihood $p(\mathbf{x}^i)$. The weighted samples are approximately distributed according to $p(\mathbf{x})$.

More formally, given a *proposal distribution* $g(\mathbf{x})$, we draw samples according to:

$$\mathbf{x}^i \sim g(\mathbf{x}) \tag{A.1}$$

Then, the *importance weight* of each sample is computed as:

$$\omega^i = \frac{\text{target distribution}}{\text{proposal distribution}} = \frac{p(\mathbf{x}^i)}{g(\mathbf{x}^i)} \tag{A.2}$$

The weighted samples approximately represent $p(\mathbf{x})$ and can be used, for example, to compute an estimate of an expected value of some function f with respect to the target PDF. For N samples:

$$\mathbf{E}_p\left[f(\mathbf{x})\right] \approx \frac{1}{N}\sum_{i=1}^{N} \omega^i f(\mathbf{x}^i) \tag{A.3}$$

Importance sampling is most efficient if the proposal distribution approximates the target distri-

$\mathbf{x}^i$	ith sample from $p(\mathbf{x})$	ω^i	weight of ith sample	$\mathbf{x}_{1:t}$	first t components of $\mathbf{x}$
$p(\mathbf{x})$	target distribution	$\mathbf{x}_t$	tth component of $\mathbf{x}$	N	number of samples
$g(\mathbf{x})$	proposal distribution				

Notation for this chapter

Algorithm A.1 Sequential importance sampling-resampling (SISR)

```
1: for $t = 1 \ldots T$ do
2:   for all particles $\phi_{t-1}^i = \{\mathbf{x}_{1:t-1}^i, \omega_{t-1}^i\}$, $i = 1 \ldots N$ do
3:     $\mathbf{x}_t^i \sim g_t(\mathbf{x}_t | \mathbf{x}_{1:t-1}^i)$
4:     $\mathbf{x}_{1:t}^i = \left[\mathbf{x}_{1:t-1}^i \;\; \mathbf{x}_t^i\right]^T$
5:     $\omega_t^i \propto \frac{p_t(\mathbf{x}_t^i | \mathbf{x}_{1:t-1}^i)}{g_t(\mathbf{x}_t | \mathbf{x}_{1:t-1}^i)}$
6:   end for
7:   Resample with replacement from $\{\phi_t^1, \ldots, \phi_t^N\}$ according to $\omega_t^i$
8: end for
```

bution closely.

A.2 Sequential importance sampling and particle filtering

For high-dimensional problems it is often difficult to design a proposal distribution that closely approximates $p(\mathbf{x})$. One approach is to build up the proposal sequentially, i.e.:

1. Decompose $\mathbf{x}$ into $\mathbf{x} = [\mathbf{x}_1 \;\; \ldots \;\; \mathbf{x}_T]^T$; then

2. Construct the proposal density $g(\mathbf{x})$ from the product of its marginals:

$$g(\mathbf{x}) = g_1(\mathbf{x}_1) g_2(\mathbf{x}_2 | \mathbf{x}_1) \ldots g_T(\mathbf{x}_T | \mathbf{x}_1, \ldots, \mathbf{x}_{T-1}) \tag{A.4}$$

The importance weight ω can be similarly decomposed but the target density $p(\mathbf{x})$ cannot. However, the marginal target densities $p_t(\mathbf{x}_t)$ can often be approximated by "auxiliary distributions" up to a normalizing constant. In nonlinear filtering scenarios, $p_t(\mathbf{x}_t)$ is usually chosen to be the "current" posterior distribution. (Appendix B describes this approach for particle filtering SLAM.)

The weight can be written recursively as:

$$\omega_t = \omega_{t-1} \frac{p_t(\mathbf{x}_t | \mathbf{x}_{1:t-1})}{g_t(\mathbf{x}_t | \mathbf{x}_{1:t-1})} \tag{A.5}$$

This sequential approach has an important advantage: if the weight of a sample ever becomes "too small" we can stop generating the sample to save computation. To replace the sample, we could either regenerate it from the beginning (which is potentially expensive), or resample from among the current sample set. The latter approach tends to work very well in practice.

This technique for generating samples from high-dimensional distributions is variously known as "sequential importance sampling" (SIS), "bootstrap filtering," or "particle filtering." Algorithm A.1 gives the algorithm for generating N samples sequentially, with resampling. This approach is the basis for particle filtering SLAM, described in detail in Appendix B. For an in-depth introduction to sequential sampling techniques and applications, see the books by Liu (2001) and Doucet et al. (2001).

A.3 Markov chain Monte Carlo

An alternative to sequential sampling techniques is to instead generate samples from $p(\mathbf{x})$ iteratively based on Markov chains. A variety of methods exist for doing this "Markov chain Monte Carlo" (MCMC) sampling, but most of them are variations of the Metropolis-Hastings algorithm (Metropolis et al., 1953; Hastings, 1970), shown in Algorithm A.2.

In Metropolis-Hastings, the proposal distribution $q(\mathbf{x}^i, \hat{\mathbf{x}})$ need not be related to $p(\mathbf{x})$ (although it can be and often is), unlike the proposal in importance sampling. The basic idea is that the proposal, coupled with the acceptance probability a, form a transition kernel in a Markov

Algorithm A.2 Metropolis-Hastings (MCMC)

```
1: Draw a state $\mathbf{x}^0$ from some initial distribution
2: loop
3:     Draw a new sample from the proposal distribution: $\hat{\mathbf{x}}^{i+1} \sim q(\mathbf{x}^i, \hat{\mathbf{x}})$
4:     Compute the acceptance probability:
           $$a(\mathbf{x}^i, \hat{\mathbf{x}}^{i+1}) = \frac{p(\hat{\mathbf{x}}^{i+1})}{p(\mathbf{x}^i)} \frac{q(\hat{\mathbf{x}}^{i+1}, \mathbf{x}^i)}{q(\mathbf{x}^i, \hat{\mathbf{x}}^{i+1})}$$
5:     $\mathbf{x}^{i+1} = \begin{cases} \hat{\mathbf{x}}^{i+1} & \text{w.p. } \min\{1, a(\mathbf{x}^i, \hat{\mathbf{x}}^{i+1})\} \\ \mathbf{x}^i & \text{otherwise} \end{cases}$
6: end loop
```

chain. It can be shown that the invariant (or steady-state) distribution of the chain is $p(\mathbf{x})$. For a detailed discussion of the properties of the algorithm, see, e.g., the thesis by Neal (1993).

By running the Markov chain over N iterations, the samples $\mathbf{x}^i$, $i = 1 \ldots N$, can be used to represent $p(\mathbf{x})$. Frequently, Metropolis-Hastings is performed "component-wise," i.e., only a single variable or block of variables $\mathbf{x}_t \in \mathbf{x}$ is modified at a time. An important variant is *Gibbs sampling*, which replaces each component, usually in turn, by sampling from its conditional distribution given the values of other components, i.e.:

$$\begin{array}{rcl} \mathbf{x}_1^{i+1} & \sim & p(\mathbf{x}_1 | \mathbf{x}_2^i, \mathbf{x}_3^i, \ldots, \mathbf{x}_T^i) \\ \mathbf{x}_2^{i+1} & \sim & p(\mathbf{x}_2 | \mathbf{x}_1^{i+1}, \mathbf{x}_3^i, \ldots, \mathbf{x}_T^i) \\ & \ldots & \\ \mathbf{x}_t^{i+1} & \sim & p(\mathbf{x}_t | \mathbf{x}_1^{i+1}, \ldots, \mathbf{x}_{t-1}^{i+1}, \mathbf{x}_{t+1}^i, \ldots, \mathbf{x}_T^i) \\ & \ldots & \\ \mathbf{x}_T^{i+1} & \sim & p(\mathbf{x}_T | \mathbf{x}_1^{i+1}, \mathbf{x}_2^{i+1}, \ldots, \mathbf{x}_{T-1}^{i+1}) \end{array} \tag{A.6}$$

Note that the new value $\mathbf{x}_{t-1}^{i+1}$ is used immediately in sampling $\mathbf{x}_t^{i+1}$.

There are several important caveats when generating samples using MCMC:

- Samples are only *asymptotically* from $p(\mathbf{x})$, since a Markov chain only asymptotically approaches its invariant distribution. Thus, some samples should be discarded from the beginning of the MCMC process, which is often called the "burn-in period." The length of the burn-in period depends on the time required to explore the state space (the *mixing time*), which is generally shortest if the proposal approximates $p(\mathbf{x})$ well.

- Samples are correlated through the proposal distribution, unlike samples drawn by importance sampling, which are i.i.d.

- *Entire* samples are generated at a time, so MCMC approaches are not suitable for use in recursive estimation scenarios.

Nevertheless, MCMC can be useful in the context of mapping. For example, Ranganathan et al. (2005) use MCMC to do offline topological mapping. Chapter 6 describes another application: "roughening" of trajectories sampled using particle filtering techniques in order to increase particle diversity.

B SLAM

A majority of recent robot mapping research, beginning with the seminal work of Smith et al. (1990), falls into the framework of *simultaneous localization and mapping* (SLAM). The SLAM problem is to concurrently estimate both a map of the environment and the robot's pose with respect to the map. The problem can be posed probabilistically, with the goal of estimating the posterior distribution:

$$p(\mathbf{x}_t^r, \mathbf{x}^m | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \tag{B.1}$$

where $\mathbf{x}_t^r$ is the robot's pose at time t, $\mathbf{x}^m$ is the map, $\mathbf{u}_{1:t}$ is the history of control inputs given to the robot, $\mathbf{z}_{1:t}$ is the history of sensor measurements, and $\mathbf{n}_{1:t}$ is the history of correspondences between measured features and landmarks in the map, i.e., the history of data associations.

In SLAM, it is desirable to estimate (B.1) recursively as new information arrives, so that the robot's pose and map can be updated online. Given a recursive estimation technique, it must also be computationally efficient so that it can be executed in realtime.

B.1 Bayesian formulation

We first formulate SLAM as a Bayesian filtering problem. Much of this formulation is based on that of Thrun et al. (2005), who offer a very thorough discussion of SLAM.

Let the system state vector be:

$$\mathbf{x}_t = \begin{bmatrix} \mathbf{x}_t^r \\ \mathbf{x}^m \end{bmatrix} \tag{B.2}$$

The SLAM posterior distribution is then:

$$p(\mathbf{x}_t | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \tag{B.3}$$

Applying Bayes' rule, we obtain:

$$p(\mathbf{x}_t | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) = \frac{p(\mathbf{z}_t | \mathbf{x}_t, \mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t}) p(\mathbf{x}_t | \mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t})}{p(\mathbf{z}_t | \mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t})} \tag{B.4}$$

η	normalization constant	$\mathbf{P}_{\mathbf{x}_t^r\mathbf{x}^m}$	pose/map cross-covariance	$\mathbf{x}_{1:t}^{r,i}$	robot trajectory for the *i*th particle
$\mathbf{x}_t^r$	robot pose at time t	$\mathbf{V}_t$	control input covariance	$\mathbf{x}^{m,i}$	map for the *i*th particle
$\mathbf{x}^m$	map	$\mathbf{R}_t$	measurement covariance	ω_t^i	weight of the *i*th particle
$\mathbf{P}_{\mathbf{x}_t^r}$	robot pose covariance	ϕ_t^i	*i*th particle at time t	N	number of particles
$\mathbf{P}_{\mathbf{x}^m}$	map covariance				

Notation for this chapter

The denominator can be subsumed into a normalization constant:

$$\eta p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t}) p(\mathbf{x}_t|\mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t}) \tag{B.5}$$

To proceed, we make a simplifying assumption known as the *Markov* or *complete state assumption*, i.e., we assume that the state $\mathbf{x}_t$ completely represents all relevant components of the robot and environment such that given $\mathbf{x}_t$, future measurements are independent of past states and measurements. For the purposes of SLAM, this assumption typically equates to an assumption that the environment is static. Section 2.9.4 discusses the Markov assumption in more detail.

Assuming the system is Markovian, we can employ conditional independence to simplify the first term of Equation B.5:

$$p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t}) = p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{n}_t) \tag{B.6}$$

This term represents a probabilistic model of the robot's sensor, i.e., the *measurement model*.

We now marginalize the second term of Equation B.5:

$$p(\mathbf{x}_t|\mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t}) p(\mathbf{x}_{t-1}|\mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t})\, d\mathbf{x}_{t-1} \tag{B.7}$$

Again applying the Markov assumption, the first term inside the integrand can be simplified:

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t}) = p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) \tag{B.8}$$

This term represents a probabilistic model of the robot's motion, i.e., the *motion model*.

Combining the above terms and dropping noninformative variables, we obtain the basic SLAM Bayes filter:

$$\underbrace{p(\mathbf{x}_t|\mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t})}_{\text{posterior}} = \eta p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{n}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) \underbrace{p(\mathbf{x}_{t-1}|\mathbf{u}_{1:t-1}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t-1})}_{\text{previous posterior}} d\mathbf{x}_{t-1} \tag{B.9}$$

As can be seen in Equation B.9, the Bayes filter for SLAM recursively estimates the desired posterior based on the previous posterior, the motion model, and the measurement model. Given a robot motion command and a sensor measurement, the Bayes filter can be implemented in two steps: a *prediction* step, which projects the state forward based on the previous posterior and the motion model, and an *update* step, which modifies the predicted posterior according to the measurement model.

B.2 EKF SLAM

One way to implement Bayes filtering SLAM is with an extended Kalman filter (EKF). A thorough introduction to the EKF and its application to SLAM is available in (Choset et al., 2005) or (Thrun et al., 2005). The EKF approximates the posterior (B.3) as a Gaussian, i.e.:

$$p(\mathbf{x}^r, \mathbf{x}^m) = \mathcal{N}(\mathbf{x}_t, \mathbf{P}_{\mathbf{x}_t}) = \mathcal{N}\left(\begin{bmatrix} \mathbf{x}_t^r \\ \mathbf{x}^m \end{bmatrix}, \begin{bmatrix} \mathbf{P}_t^r & \mathbf{P}_t^{r,m} \\ (\mathbf{P}_t^{r,m})^T & \mathbf{P}_t^m \end{bmatrix} \right) \tag{B.10}$$

Algorithm B.1 shows the high-level EKF SLAM algorithm.

In the algorithm, Steps 3–4 implement the prediction step, i.e., they compute:

$$p(\mathbf{x}_t|\mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t-1}) \tag{B.11}$$

In the linear Gaussian model of the EKF, this is done by linearizing the process model f to update

Algorithm B.1 EKF SLAM

```
1:  loop
2:      Move according to the control $\mathbf{u}_t$
3:      $\hat{\mathbf{x}}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_t)$                         // apply process model
4:      $\hat{\mathbf{P}}_{\mathbf{x}_t} = \mathbf{H}_\mathbf{x}\mathbf{P}_{\mathbf{x}_{t-1}}\mathbf{H}_\mathbf{x}^T + \mathbf{H}_\mathbf{u}\mathbf{V}_t\mathbf{H}_\mathbf{u}^T$     // dilate uncertainty
5:      Obtain a measurement $\mathbf{z}_t$ and compute correspondences $\mathbf{n}_t$
6:      $\boldsymbol{\nu} = \mathbf{z}_t - h(\hat{\mathbf{x}}_t)$                              // compute innovation
7:      $\mathbf{S} = \mathbf{H}_h\hat{\mathbf{P}}_{\mathbf{x}_t}\mathbf{H}_h^T + \mathbf{R}_t$                  // compute innovation covariance
8:      $\mathbf{K} = \hat{\mathbf{P}}_{\mathbf{x}_t}\mathbf{H}_h^T\mathbf{S}^{-1}$                       // compute Kalman gain
9:      $\mathbf{x}_t = \hat{\mathbf{x}}_t + \mathbf{K}\boldsymbol{\nu}$                             // update state
10:     $\mathbf{P}_{\mathbf{x}_t} = \hat{\mathbf{P}}_{\mathbf{x}_t} - \mathbf{K}\mathbf{H}_h\hat{\mathbf{P}}_{\mathbf{x}_t}$                 // update covariance
11:     If necessary, add new landmarks to the state
12: end loop
```

the covariance $\mathbf{P}_{\mathbf{x}_t}$ of the robot pose and map, i.e., using the Jacobian:

$$\mathbf{H}_\mathbf{x} = \frac{\partial f(\mathbf{x}_{t-1})}{\partial \mathbf{x}} \tag{B.12}$$

and the Jacobian:

$$\mathbf{H}_\mathbf{u} = \frac{\partial f(\mathbf{x}_{t-1})}{\partial \mathbf{u}} \tag{B.13}$$

The noisy control input is represented by the distribution:

$$\mathcal{N}(\mathbf{u}_t, \mathbf{V}_t) \tag{B.14}$$

Upon taking a sensor measurement, the measurement model is used to update the robot pose and map, i.e., to compute:

$$p(\mathbf{x}_t | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \tag{B.15}$$

The *innovation* $\boldsymbol{\nu}$ is the difference between the measurement $\mathbf{z}_t$ and the predicted measurement according to the model, h. The measurement model is linearized with the Jacobian $\mathbf{H}_h$, which, for a measurement of a single landmark $\mathbf{x}_i^m$, is:

$$\mathbf{H}_h = \frac{\partial h(\hat{\mathbf{x}}_t)}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial h(\hat{\mathbf{x}}_t)}{\partial \mathbf{x}^r} & \mathbf{0} & \ldots & \mathbf{0} & \frac{\partial h(\hat{\mathbf{x}}_t)}{\partial \mathbf{x}_i^m} & \mathbf{0} & \ldots & \mathbf{0} \end{bmatrix}^T \tag{B.16}$$

The sensor noise is represented by the distribution:

$$\mathcal{N}(\mathbf{z}_t, \mathbf{R}_t) \tag{B.17}$$

The innovation and its covariance $\mathbf{S}$ are used to compute the Kalman gain $\mathbf{K}$. If the state estimate is very uncertain and the measurement is very certain, the Kalman gain will be large; if the state estimate is certain and the measurement is not, the gain will be small.

Extracted features for which data association finds no match in the map do not contribute to the Kalman update, but are added to the state after the update.

There are several problems with EKF SLAM:

- Computation of the Kalman gain requires a matrix inverse and matrix multiplications for matrices that are bounded only by the size of the map. For maps with n landmarks, every EKF update step costs $O(n^3)$ with a naïve implementation; by exploiting structural properties of the SLAM problem, $O(n^2)$ updates can be obtained (Newman, 1999). This is still often too expensive for realtime implementation in complicated environments.

- The linear Gaussian approximation is often a bad one. Neither the motions nor measurements are truly Gaussian. Furthermore, the true posterior distribution is often highly multimodal.
- The EKF update requires that a single data association hypothesis be chosen. While multiple hypothesis approaches have been investigated (Cox and Leonard, 1994), they are even more expensive.

B.3 RBPF SLAM

A recent approach that improves significantly on many aspects of EKF SLAM is based on a slightly different problem formulation. The goal is to estimate the "full" SLAM posterior (Thrun et al., 2004, 2005):

$$p(\mathbf{x}^r_{1:t}, \mathbf{x}^m | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) \tag{B.18}$$

i.e., to estimate the entire *pose history* of the robot in addition to the map. At first this appears only to make SLAM more complicated, but Murphy (2000) observed that, under the Markov assumption and conditioned on the entire trajectory of the robot, landmarks in the map are independent. This is because correlation between landmark estimates arises only through robot pose uncertainty.[*] Thus, the posterior can be *factored*. Applying conditional probability and dropping noninformative variables:

$$\begin{aligned}
p(\mathbf{x}^r_{1:t}, \mathbf{x}^m | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) &\overset{\text{cond. prob.}}{=} p(\mathbf{x}^r_{1:t} | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) p(\mathbf{x}^m | \mathbf{x}^r_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) && \text{(B.19)}\\
&= p(\mathbf{x}^r_{1:t} | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) p(\mathbf{x}^m_1, \ldots, \mathbf{x}^m_n | \mathbf{x}^r_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t}) && \text{(B.20)}\\
&\overset{\text{cond. indep.}}{=} \underbrace{p(\mathbf{x}^r_{1:t} | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t})}_{\text{posterior over trajectories}} \prod_{i=1}^{n} \underbrace{p(\mathbf{x}^m_i | \mathbf{x}^r_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t})}_{\text{posterior over landmark } i} && \text{(B.21)}
\end{aligned}$$

The posterior over robot trajectories can be represented using samples, or "particles." Conditioned on each sample, a map with n landmarks can be represented by n separate small filters. Typically these are EKFs. This type of factorization is known in the statistics literature as *Rao-Blackwellization* (Liu, 2001).

Offline and given full control and measurement data, it may be reasonable to draw samples from the trajectory posterior using iterative techniques like MCMC. To implement SLAM online, we can instead employ sequential techniques like those described in Appendix A, i.e., we can do *Rao-Blackwellized particle filtering* (RBPF). The motion model is employed as a proposal distribution for sequential importance sampling. Given a control input, a particle $\phi^i_{1:t-1} = \left\{\mathbf{x}^{r,i}_{1:t-1}, \mathbf{x}^{m,i}\right\}$ is projected forward by sampling:

$$\mathbf{x}^{r,i}_t \sim p(\mathbf{x}^r_t | \mathbf{x}^{r,i}_{t-1}, \mathbf{u}_t) \tag{B.22}$$

After projecting every particle forward, the "prediction" step is complete and the posterior:

$$p(\mathbf{x}^i_{1:t} | \mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t-1}) \tag{B.23}$$

is represented by the particles.

Following the approach described in Appendix A, the importance weight of a sample is

[*]This independence assumption is in fact a major weakness of the approach and can lead to inconsistent estimation. An argument against the assumption is given in Chapter 5 of this thesis.

Algorithm B.2 RBPF SLAM

```
1: loop
2:   Move according to the control $\mathbf{u}_t$ and obtain a measurement $\mathbf{z}_t$
3:   for all particles $\phi^i_{t-1}$, $i = 1 \dots N$ do
4:     $\mathbf{x}^{r,i}_t \sim p(\mathbf{x}^r_t | \mathbf{x}^{r,i}_{t-1}, \mathbf{u}_t)$    // sample new pose
5:     Compute correspondences $\mathbf{n}^i_t$
6:     for each observed landmark $\mathbf{x}^{m,i}_j$ do
7:       $\boldsymbol{\nu}_j = \mathbf{z}_{t,j} - h_j(\mathbf{x}^{r,i}_t)$    // compute innovation
8:       $\mathbf{S}_j = \mathbf{H}_{h,j}\mathbf{P}^i_j\mathbf{H}^T_{h,j} + \mathbf{R}_{t,j}$    // compute innovation covariance
9:       $\mathbf{K}_j = \mathbf{P}^i_j\mathbf{H}^T_{h,j}\mathbf{S}^{-1}_j$    // compute Kalman gain
10:      $\mathbf{x}^{m,i}_j = \mathbf{x}^{m,i}_j + \mathbf{K}_j\boldsymbol{\nu}_j$    // update landmark
11:      $\mathbf{P}^i_j = \mathbf{P}^i_j - \mathbf{K}_j\mathbf{H}_{h,j}\mathbf{P}^i_j$    // update landmark covariance
12:      $\omega^i_t = \omega^i_t \times p(\mathbf{z}_{t,j} | \mathbf{x}^{r,i}_t, \mathbf{x}^{m,i}, \mathbf{n}^i_{t,j})$    // update importance weight
13:    end for
14:    If necessary, add new landmarks to $\mathbf{x}^{m,i}$
15:  end for
16:  Draw with replacement $N$ times from $\{\phi^1_t, \dots, \phi^N_t\}$ according to $\omega^i_t$    // resample
17:  for $j = 1 \dots N$ do
18:    $\omega^j_t = \frac{1}{N}\sum_{i=1}^N \omega^i_t$    // reset weights
19:  end for
20: end loop
```

computed as:

$$\omega^i_t = \frac{\text{target distribution}}{\text{proposal distribution}} \tag{B.24}$$

$$= \frac{p(\mathbf{x}^i_{1:t} | \mathbf{u}_{1:t}, \mathbf{z}_{1:t}, \mathbf{n}_{1:t})}{p(\mathbf{x}^i_{1:t} | \mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t-1})} \tag{B.25}$$

$$\overset{\text{marginalization}}{=} \eta \frac{p(\mathbf{z}_t | \mathbf{x}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t}) p(\mathbf{x}_{1:t} | \mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t-1})}{p(\mathbf{x}_{1:t} | \mathbf{u}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t-1})} \tag{B.26}$$

$$\overset{\text{cancellation}}{=} \eta p(\mathbf{z}_t | \mathbf{x}_{1:t}, \mathbf{z}_{1:t-1}, \mathbf{n}_{1:t}) \tag{B.27}$$

In other words, the importance weight is proportional to the measurement likelihood. Frequently, a Gaussian measurement model is assumed and the importance weight is computed as:

$$\omega^i_t = |2\pi\mathbf{S}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\boldsymbol{\nu}\mathbf{S}^{-1}\boldsymbol{\nu}^T\right) \tag{B.28}$$

The importance weight is used to perform a resampling step to improve the estimate of the posterior given the current measurement.

For per-particle maps represented by a series of EKFs, the standard EKF update can be applied to a landmark upon observing it. The cost is proportional to the size of the landmark state, which is a (typically small) constant.

The basic RBPF SLAM algorithm is given in Algorithm B.2. The RBPF approach improves upon EKF SLAM in several respects:

- A straightforward implementation requires only $O(Nn)$ time per update, where N is the number of particles and n is the number of landmarks in the map of each particle. Under reasonable assumptions and by using a tree-based map representation, $O(N \log n)$ updates are possible (Thrun et al., 2004).

- The nonparametric representation of the trajectory posterior enables RBPF to capture the multimodal nature of the distribution.
- Data association is done separately for each particle, since every particle has its own map. Thus, each particle represents a data association hypothesis.

There are also weaknesses in the technique, most importantly the landmark independence assumption (see Chapter 5) and the inconsistency and degeneracy issues discussed in Chapter 6.

C
SOFTWARE

Most of the software used to generate the results in this thesis is available online at the following website:

```
http://www.cs.rpi.edu/~beevek/thesis/
```

Most of the algorithms presented in the thesis were implemented in C++ on a Linux 2.6.x system. All of the software was implemented by the author of this thesis unless otherwise noted (e.g., in source code comments). Each of the packages contains a `README` file with additional notes, including necessary libraries and configuration details. The following packages are available:

gridsim2 A simulation of the sensor, environment, and mapping models described in Chapter 3, used to verify the theoretical results detailed in the chapter and examine alternative cases such as MRF-based structured environments that were not investigated in the analysis.

slam3 A complete implementation of particle filtering SLAM, used to obtain the results in Chapters 4–6. Implemented SLAM algorithms include: FastSLAM 1, FastSLAM 2, FastSLAM 1 with rectilinearity constraints (see Chapter 5), fixed-lag roughening, and the block proposal distribution (see Chapter 6), all with support for multiscans as described in Chapter 4. Supports point features (pre-extracted from the raw data) and line segment features (pre-extracted, or extracted using the split-merge algorithm described in Chapter 4). Supports known correspondences or nearest neighbor data association. Implements a fast binary tree based landmark map representation (a.k.a. "$\log N$" RBPF) and simple spurious landmark detection and removal. Uses a simple odometry-based motion model. Implements random, residual, and generalized resampling as described in Chapter 6, and adaptive resampling based on $\hat{N}_{\text{eff}}$. Also includes a basic implementation of scan-matching RBPF SLAM and a tool for simulating a robot among point features. Various Matlab utilities and Perl scripts for parsing and visualizing input and output files are provided. Supports data sets in RPI, RPI binary, Player/Stage, and CARMEN file formats.

ratbot-slam A fixed point implementation of FastSLAM 1 with multiscans as described in Chapter 7, meant to run on the Ratbot platform with an ATMEGA64 microcontroller as the main CPU.

fixedpoint A simple fixed point math library for numbers in 16.16 or 6.10 format, with support for basic conversions, random number generation, trigonometric functions, and exponential functions. (Used by ratbot-slam.)

ktracker A simple OpenCV (Intel Corp., 2006) based vision tracker. Tracks a single mobile robot's position and heading, using two colored markers mounted on the robot and a single calibrated monocular camera. Supports Firewire and video4linux cameras, and the FreeBSD `meteor` video capture driver. Also implements Bayer decoding for use with the Point Grey Research Dragonfly camera.

mpro A set of programs and libraries implementing remote control, visualization, simulation, and data retrieval/logging for the iRobot MagellanPro research robot. Used to obtain simulated results in Chapters 5 and 7 (with modifications to simulate the Ratbot platform).

BIBLIOGRAPHY

E.U. Acar, H. Choset, and P.N. Atkar. Complete sensor-based coverage with extended-range detectors: A hierarchical decomposition in terms of critical points and Voronoi diagrams. In *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots & Systems*, pages 1305–1311, October 2001.

Atmel Corp. *Atmel 8-bit AVR microcontroller with 64 K bytes in-system programmable flash (ATmega64(L)) 2490K-AVR-04/06*. San Jose, CA, 2006.

F. Aurenhammer. Voronoi diagrams — A survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345–405, September 1991.

T. Bailey. *Mobile robot localisation and mapping in extensive outdoor environments*. PhD thesis, Australian Center for Field Robotics, University of Sydney, August 2002.

T. Bailey. Constrained initialisation for bearing-only SLAM. In *Proceedings of the 2003 IEEE International Conference on Robotics & Automation*, pages 1966–1971, September 2003.

T. Bailey, J. Nieto, and E. Nebot. Consistency of the FastSLAM algorithm. In *IEEE Intl. Conf. on Robotics and Automation*, pages 424–427, 2006.

Y. Bar-Shalom, X. Rong Li, and T. Kirubarajan. *Estimation with applications to tracking and navigation*. Wiley, New York, 2001.

Kristopher R. Beevers. Topological mapping and map merging with sensing-limited robots. Master's thesis, Rensselaer Polytechnic Institute, Troy, NY, April 2004.

Kristopher R. Beevers. Sampling strategies for particle filtering SLAM. Technical Report 06-11, Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, September 2006.

Kristopher R. Beevers and Wesley H. Huang. Loop closing in topological maps. In *2005 International Conference on Robotics and Automation (ICRA 2005)*, pages 4378–4383, Barcelona, Spain, April 2005.

Kristopher R. Beevers and Wesley H. Huang. SLAM with sparse sensing. In *2006 International Conference on Robotics and Automation (ICRA 2006)*, pages 2285–2290, Orlando, FL, May 2006a.

Kristopher R. Beevers and Wesley H. Huang. Inferring and enforcing relative constraints in SLAM. In *2006 Workshop on the Algorithmic Foundations of Robotics (WAFR 2006)*, New York, NY, July 2006b.

Kristopher R. Beevers and Wesley H. Huang. Fixed-lag sampling strategies for particle filtering SLAM. *2007 International Conference on Robotics and Automation (ICRA 2007)*, to appear, April 2007a.

Kristopher R. Beevers and Wesley H. Huang. How sensing capabilities influence map quality. Submitted to *2007 International Conference on Robotics and Automation (ICRA 2007)*, April 2007b.

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.

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.

P. Biber and T. Duckett. Dynamic maps for long-term operation of mobile service robots. In *Proceedings of Robotics: Science and Systems*, Cambridge, MA, June 2005.

J. Borenstein, H.R. Everett, and L. Feng. *"Where am I?" Sensors and methods for mobile robot positioning*. University of Michigan, April 1996.

G.A. Borges and M.-J. Aldon. Line extraction in 2D range images for mobile robotics. *Journal of Intelligent and Robotic Systems*, 40:267–297, 2004.

M. Bosse, P.M. Newman, J.J. Leonard, and S. Teller. SLAM in large-scale cyclic environments using the Atlas framework. *International Journal of Robotics Research*, 23(12):1113–1139, December 2004.

G.E.P. Box and M.E. Muller. A note on the generation of random normal deviates. *Annals of Mathematics and Statistics*, 29:610–611, 1958.

K. Briechle and U.D. Hanebeck. Localization of a mobile robot using relative bearing measurements. *IEEE Transactions on Robotics and Automation*, 20(1):36–44, February 2004.

Emma Brunskill and Nicholas Roy. SLAM using incremental probabilistic PCA and dimensionality reduction. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, pages 344–349, 2005.

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.

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.

S. Chib. Calculating posterior distributions and modal estimates in Markov mixture models. *Journal of Econometrics*, 75(1):79–97, 1996.

H. Choset. Incremental construction of the generalized Voronoi diagram, the generalized Voronoi graph, and the hierarchical generalized Voronoi graph. In *Proceedings of the First CGC Workshop on Computational Geometry*, Baltimore, MD, October 1997.

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.

H. Choset, K.M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun. *Principles of robot motion*. MIT Press, Cambridge, MA, 2005.

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

M. Csorba and H.F. Durrant-Whyte. New approach to map building using relative position estimates. *SPIE Navigation and Control Technologies for Unmanned Systems II*, 3087(1):115–125, 1997.

M. Deans and M. Hebert. Experimental comparison of techniques for localization and mapping using a bearings only sensor. In *Proceedings of the Seventh International Symposium on Experimental Robotics (ISER)*, pages 395–404. Springer-Verlag, December 2000a.

M.C. Deans and M. Hebert. Invariant filtering for simultaneous localization and mapping. In *Proc. IEEE Intl. Conf. Robotics and Automation*, pages 1042–1047, 2000b.

G. Dedeoglu and G.S. Sukhatme. Landmark-based matching algorithm for cooperative mapping by autonomous robots. In *Proceedings of the 2000 International Symposium on Distributed Autonomous Robotic Systems (DARS)*, pages 251–260, Knoxville, Tennessee, October 2000.

C. Detweiler, J. Leonard, D. Rus, and S. Teller. Passive mobile robot localization within a fixed beacon field. In *2006 Workshop on the Algorithmic Foundations of Robotics (WAFR 2006)*, New York, July 2006.

J. Djugash, S. Singh, and P. Corke. Further results with localization and mapping using range from radio. In *Proceedings of the Fifth International Conference on Field and Service Robotics*, Australia, July 2005.

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

A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo methods in practice*. Springer, New York, 2001.

A. Doucet, M. Briers, and S. Sénécal. Efficient block sampling strategies for sequential Monte Carlo methods. *Journal of Computational and Graphical Statistics*, to appear, 2006.

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

R.O. Duda and P.E. Hart. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

R.O. Duda and P.E. Hart. *Pattern classification and scene analysis*. Wiley, New York, USA, 1973.

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.

H.F. Durrant-Whyte. Uncertain geometry in robotics. *IEEE Journal of Robotics and Automation*, 4 (1):23–31, 1988.

M.A. Erdmann. Understanding action and sensing by designing action-based sensors. *International Journal of Robotics Research*, 14(5):483–509, 1995.

M.A. Erdmann and M.T. Mason. An exploration of sensorless manipulation. *IEEE Transactions on Robotics and Automation*, 4(4):369–379, August 1988.

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.

J. Folkesson and H. Christensen. Graphical SLAM – a self-correcting map. In *Proceedings of the 2004 IEEE International Conference on Robotics & Automation*, pages 383–390, New Orleans, April 2004.

D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391–427, November 1999.

U. Frese, P. Larsson, and T. Duckett. A multilevel relaxation algorithm for simultaneous localization and mapping. *IEEE Transactions on Robotics*, 21(2):196–207, April 2005.

W.R. Gilks and C. Berzuini. Following a moving target — Monte Carlo inference for dynamic Bayesian models. *Journal of the Royal Statistical Society B*, 63(1):127–146, 2001.

K.Y. Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, 10:201–225, 1993.

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.

Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improving grid-based SLAM with Rao-Blackwellized particle filters by adaptive proposals and selective resampling. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, pages 2443–2448, 2005.

A. Großman and R. Poli. Robust mobile robot localisation from sparse and noisy proximity readings using Hough transform and probability grids. In *Proceedings of the IJCAI-99 Workshop on Reasoning with Uncertainty in Robot Navigation (RUR-99)*, 1999.

J. Guivant, J. Nieto, F. Masson, and E. Nebot. Navigation and mapping in large unstructured environments. *International Journal of Robotics Research*, 23(4-5):449–472, 2004.

J.E. Guivant and E.M. Nebot. Optimization of the simultaneous localization and map building algorithm for real time implementation. *IEEE Transactions on Robotics and Automation*, 17(3): 242–257, 2000.

J.-S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, number November, pages 318–325, Monterey, CA, 1999.

Joseph Y. Halpern. *Reasoning about uncertainty*. MIT Press, Cambridge, MA, 2003.

W.K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97–109, 1970.

Honeywell. *HRF-ROC09325XM radio module 2003 09325XM*. Plymouth, MN, October 2003.

Andrew Howard and Nicholas Roy. The Robotics Data Set Repository (Radish), 2003. URL `http://radish.sourceforge.net/`.

Wesley H. Huang and Kristopher R. Beevers. Complete topological mapping with sparse sensing. Technical Report 05-06, Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, March 2005a.

Wesley H. Huang and Kristopher R. Beevers. Topological map merging. *International Jounal of Robotics Research*, 24(8):601–613, August 2005b.

Wesley H. Huang and Kristopher R. Beevers. Topological mapping with sensing-limited robots. In M. Erdmann et al., editors, *Algorithmic Foundations of Robotics VI*, pages 235–250. Springer, 2005c.

Intel Corp. Open source computer vision library (OpenCV), 2006. URL `http://www.intel.com/research/mrl/research/opencv`.

V. Isler and R. Bajcsy. The sensor selection problem for bounded uncertainty sensing models. In *Fourth International Symposium on Information Processing in Sensor Networks (IPSN 2005)*, pages 151–158, April 2005.

G.A. Kantor and S. Singh. Preliminary results in range-only localization and mapping. In *Proceedings of the 2002 IEEE International Conference on Robotics & Automation*, volume 2, pages 1818–1823, May 2002.

D.G. Kirkpatrick. Efficient computation of continuous skeletons. In *Proceedings of the 20th IEEE Annual Symposium on Foundations of Computer Science*, pages 18–27, 1979.

M. Klaas, N. de Freitas, and A. Doucet. Toward practical n^2 Monte Carlo: the marginal particle filter. In *Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pages 308–315, 2005.

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.

S. Koenig and R.G. Simmons. Unsupervised learning of probabilistic models for robot navigation. In *Proceedings of the 1996 IEEE International Conference on Robotics & Automation*, 1996.

T. Kollar and N. Roy. Using reinforcement learning to improve exploration trajectories for error minimization. In *2006 International Conference on Robotics and Automation (ICRA 2006)*, pages 3338–3343, Orlando, FL, May 2006.

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

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.

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

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

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.

B. Kuipers, J. Modayil, P. Beeson, M. MacMahon, and F. Savelli. Local metrical and global topological maps in the hybrid spatial semantic hierarchy. In *Proceedings of the 2004 IEEE International Conference on Robotics & Automation*, pages 4845–4851, New Orleans, April 2004.

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.

D. Kurth. Range-only robot localization and SLAM with radio. Master's thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 2004.

U. Larsson, J. Forsberg, and A. Wernersson. Mobile robot localization: integrating measurements from a time-of-flight laser. *IEEE Transactions on Industrial Electronics*, 43(3):422–431, June 1996.

J.-C. Latombe. *Robot motion planning*. Kluwer Academic Publishers, Norwell, MA, 1991.

D.T. Lee. Two-dimensional Voronoi diagrams in the L_p-metric. *Journal of the Association for Computing Machinery*, 27(4):604–618, October 1980.

D.T. Lee and R.L. Drysdale. Generalization of Voronoi diagrams in the plane. *SIAM Journal of Computing*, 10(1):73–87, February 1981.

T. Lemaire, S. Lacroix, and J. Solá. A practical 3D bearing-only SLAM algorithm. In *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots & Systems*, pages 2449–2454, August 2005.

J. Leonard and P. Newman. Consistent, convergent, and constant-time SLAM. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI03)*, pages 1143–1150, 2003.

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.

J.J. Leonard, R. Rikoski, P. Newman, and M. Bosse. Mapping partially observable features from multiple uncertain vantage points. *Intl. Journal of Robotics Research*, 21(10):943–975, October 2002.

J.S. Liu and R. Chen. Blind deconvolution via sequential imputations. *Journal of the American Statistical Association*, 90(430):567–576, June 1995.

Jun S. Liu. *Monte Carlo strategies in scientific computing*. Springer, New York, 2001.

D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

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

A. Martinelli, A. Tapus, K.O. Arras, and R. Siegwart. Multi-resolution SLAM for real world navigation. In *Proceedings of the11th International Symposium of Robotics Research (ISRR)*, Siena, Italy, October 2003.

N.C. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1091, 1953.

Microchip Technology, Inc. *PIC16F87X data sheet DS30292C*. Chandler, AZ, 2001.

J. Modayil, P. Beeson, and B. Kuipers. Using the topological skeleton for scalable global metrical map-building. In *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots & Systems*, pages 1530–1536, 2004.

M. Montemerlo. *FastSLAM: a factored solution to the simultaneous localization and mapping problem with unknown data association*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 2003.

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.

O. Martinez Mozos, C. Stachniss, and W. Burgard. Supervised learning of places from range data using Adaboost. In *Proceedings of the 2005 IEEE International Conference on Robotics & Automation*, pages 1742–1747, 2005.

K.P. Murphy. Bayesian map learning in dynamic environments. In *Advances in Neural Information Processing Systems*, volume 12, pages 1015–1021. MIT Press, 2000.

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

R. Neal. Probabilistic inference using Markov Chain Monte Carlo methods. Technical Report CRG-TR-93-1, University of Toronto, 1993.

P.M. Newman. *On the structure and solution of the simultaneous localization and mapping problem*. PhD thesis, University of Sydney, Australia, 1999.

V. Nguyen, A. Martinelli, N. Tomatis, and R. Siegwart. A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics. In *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots & Systems*, pages 1929–1934, 2005.

J. O'Kane and S. LaValle. On comparing the power of mobile robots. In *Robotics: Science and Systems 2006*, August 2006a.

J.M. O'Kane and S.M. LaValle. Localization with limited sensing. *Submitted to IEEE Transactions on Robotics*, 2006b.

E. Olson, J. Leonard, and S. Teller. Robust range-only beacon localization. In *Proceedings of IEEE Autonomous Underwater Vehicles (AUV-2004)*, 2004.

E. Olson, J. Leonard, and S. Teller. Fast iterative optimization of pose graphs with poor initial estimates. In *2006 International Conference on Robotics and Automation (ICRA 2006)*, pages 2262–2269, Orlando, FL, May 2006.

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.

Stephan Pastis. *Pearls Before Swine: BLTs Taste So Darn Good*. Andrews McMeel Publishing, 2003.

S.T. Pfister, S.I. Roumeliotis, and J.W. Burdick. Weighted line fitting algorithms for mobile robot map building and efficient data representation. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, Taipei, Taiwan, 2003.

Y. Rachlin, J.M. Dolan, and P. Khosla. Efficient mapping through exploitation of spatial dependencies. In *IROS 2005*, 2005.

A. Ranganathan and F. Dellaert. A Rao-Blackwellized particle filter for topological mapping. To appear in *2006 International Conference on Robotics and Automation (ICRA 2006)*, May 2006.

A. Ranganathan, E. Menegatti, and F. Dellaert. Bayesian inference in the space of topological maps. *IEEE Transactions on Robotics*, 2005. in press.

E. Remolina and B. Kuipers. Towards a general theory of topological maps. *Artificial Intelligence*, 152:47–104, 2004.

D. Rodriguez-Losada and F. Matia. Integrating segments and edges in feature-based SLAM. In *Proc. of the 11th Intl. Conf. on Advanced Robotics*, pages 1717–1722, 2003.

D. Rodriguez-Losada, F. Matia, A. Jimenez, and R. Galan. Consistency improvement for SLAM – EKF for indoor environments. In *IEEE Intl. Conf. on Robotics and Automation*, pages 418–423, 2006.

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

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.

F. Savelli and B. Kuipers. Loop-closing and planarity in topological map building. In *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots & Systems*, pages 1511–1517, 2004.

Fransesco Savelli. *Topological mapping of ambiguous space: combining qualitative biases and metrical information*. PhD thesis, Department of Computer and Systems Science, University of Rome "La Sapienza", Rome, Italy, 2005.

SensComp/Polaroid, Inc. *600 series environmental transducer datasheet*. Livonia, MI, July 2003.

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

Sharp Corp. *Sharp GP2D12 optoelectronic device datasheet SMA05006*. Osaka, Japan, 2005.

H. Shatkay and L.P. Kaelbling. Learning geometrically-constrained Hidden Markov Models for robot navigation: Bridging the topological-geometrical gap. *Journal of Artificial Intelligence Research*, 16:167–207, 2002.

SICK AG. *LMS 2XX Laser Measurement System Technical Description 8 008 970/06-2003*. Germany, 2003.

D. Simon and T.L. Chia. Kalman filtering with state equality constraints. *IEEE Transactions on Aerospace and Electronic Systems*, 39:128–136, 2002.

D. Simon and D.L. Simon. Aircraft turbofan engine health estimation using constrained Kalman filtering. In *Proc. ASME Turbo Expo 2003*, number GT2003-38584, 2003.

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.

J. Solá, A. Monin, M. Devy, and T. Lemaire. Undelayed initialization in bearing only SLAM. In *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots & Systems*, pages 2499–2504, August 2005.

C. Stachniss, G. Grisetti, and W. Burgard. Information gain-based exploration using Rao-Blackwellized particle filters. In *Proc. Robotics: Science and Systems (RSS05)*, Cambridge, MA, June 2005a.

C. Stachniss, G. Grisetti, and W. Burgard. Recovering particle diversity in a Rao-Blackwellized particle filter for SLAM after actively closing loops. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, pages 667–672, 2005b.

E. Stump, B. Grocholsky, and V. Kumar. Extensive representations and algorithms for nonlinear filtering and estimation. In *2006 Workshop on the Algorithmic Foundations of Robotics (WAFR 2006)*, New York, NY, July 2006.

J.D. Tardós, J. Neira, P.M. Newman, and J.J. Leonard. Robust mapping and localization in indoor environments using SONAR data. *Intl. Journal of Robotics Research*, 21(4):311–330, April 2002.

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

S. Thrun. Learning occupancy grid maps with forward sensor models. *Autonomous Robots*, 15(2): 111–127, September 2003.

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.

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, 1998a.

S. Thrun, D. Fox, and W. Burgard. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, 31:29–53, 1998b.

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)*, 1998c.

S. Thrun, M. Montemerlo, D. Koller, B. Wegbreit, J. Nieto, and E. Nebot. FastSLAM: An efficient solution to the simultaneous localization and mapping problem with unknown data association. *Journal of Machine Learning Research*, 2004. to appear.

S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. MIT Press, Cambridge, MA, 2005.

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 2002a.

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*, pages 2749–2754, Washington, DC, May 2002b.

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.

B. Tovar, L. Munoz-Gómez, R. Murrieta-Cid, M. Alencastre-Miranda, R. Monroy, and S. Hutchinson. Planning exploration strategies for simultaneous localization and mapping. *Robotics and Autonomous Systems*, 54(4):314–331, April 2006.

US Digital. *E4 miniature optical kit encoder*. Vancouver, WA, September 2006.

W. Wen and H.F. Durrant-Whyte. Model-based multi-sensor data fusion. In *Proc. IEEE Intl. Conf. Robotics and Automation*, pages 1720–1726, 1992.

H. White. Maximum likelihood estimation of misspecified models. *Econometrica*, 50(1):1–26, January 1982.

O. Wijk and H.I. Christensen. Triangulation based fusion of sonar data with application in robot pose tracking. *IEEE Transactions on Robotics and Automation*, 16(6):740–752, 2000.

D.C.K. Yuen and B.A. MacDonald. Line-base SMC SLAM method in environments with polygonal obstacles. In *Proc. of the Australasian Conf. on Robotics and Automation*, 2003.

X. Yun, K. Latt, and J.S. Glennon. Mobile robot localization using the Hough transform and neural networks. In *Proceedings of the 1998 IEEE International Symposium on Intelligent Control (ISIC/CIRA/ISAS Joint Conf.)*, pages 393–400, Gaithersburg, MD, 1998.

G. Zunino and H.I. Christensen. Navigation in realistic environments. In M. Devy, editor, *9th Intl. Symp. on Intelligent Robotic Systems*, Toulouse, France, 2001.