UNIVERSITY OF MINNESOTA

This is to certify that I have examined this copy of a doctoral thesis by

Paul Edmund Rybski

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Maria L. Gini

Name of Faculty Adviser(s)

Signature of Faculty Adviser(s)

2003-07-31

Date

GRADUATE SCHOOL

Building Topological Maps using Minimalistic Sensor Models

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Paul Edmund Rybski

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Maria L. Gini, Adviser

July 2003

Acknowledgement

The body of work described herein could never have been achieved without the support of many people. First and foremost, I want to thank my advisor, professor Maria Gini, for giving me the freedom and support to explore my interests in robotics. The many hours I spent discussing research with her were invaluable to my work and were tremendously enjoyable. I will be ever grateful for all of the opportunities that she made available to me.

I would also like to thank each of the other members of my committee for their guidance and support. Professor Nikolaos Papanikolopoulos gave me the opportunity to work on the Distributed Robotics project from which my thesis topic developed. His support and encouragement of my work was greatly appreciated. I am also very grateful to professor Stergios Roumeliotis. The time I spent in his office learning about the subtleties of estimation theory made a huge impact in the quality of my thesis as well as in the improvement of my own skills as a researcher. Professor Richard Voyles inspired me by his incredible enthusiasm and I greatly appreciated being able to tap his knowledge across all aspects of robotics and engineering. Finally, professor Dan Kersten gave me a deep appreciation for Bayesian methods, particularly with their application across the disciplines of computer science, psychology, cognitive science, and biology.

Special thanks goes out to Professor Dan Boley for all the time he spent helping me work out the many numerical analysis, statistics, and LaTeX questions that I encountered during my time in the department.

My achievements were only made possible through the collaboration with and support of many students in the Artificial Intelligence, Robotics, and Vision Laboratory and other labs in the Department of Computer Science and Engineering. In particular, I'd like to thank Ian Burt, Andrew Drenner, Ben Jackson, Mike Janssen, Esra Kadioglu, Brad Kratochvil, Amy Larson, Colin McMillen, Doug Perrin, Sascha Stoeter, Kristen Stubbs, Harini Veeraraghavan, and Franziska Zacharias for our collaboration on research in the lab as well

as the fun we had outside of it. I'd also like to thank Brian Schmalz and Alex Ozerkovsky, two great friends and fellow students from my undergraduate alma mater, for sharing the graduate experience at the University of Minnesota.

Many thanks to my parents, Paul and Sue, for their support and encouragement through my many years in school. Finishing my PhD is the highest tribute that I can pay to two exceptional academics who have given me so much.

Finally, without the kind and loving support of my wife, Elaine, I never could have achieved as much as I have. She came into my life four years ago and taught me, above all else, that *life is the music and nobody can resist dancing.*

Dedication

This dissertation is dedicated to my wife and parents.

Abstract

This dissertation addresses the problem of simultaneous localization and mapping for miniature robots that have extremely poor odometry and sensing capabilities. Existing robotic mapping algorithms generally assume that the robots have good odometric estimates and have sensors that can return the range or bearing to landmarks in the environment. This work focuses on solutions to this problem for robots where the above assumptions do not hold.

A novel method is presented for a sensor poor mobile robot to create a topological estimate of its path through an environment by using the notion of a virtual sensor that equates "place signatures" with physical locations in space. The method is applicable in the presence of extremely poor odometry and does not require sensors that return spatial (range or bearing) information about the environment. Without sensor updates, the robot's path estimate will degrade due to the odometric errors in its position estimates. When the robot re-visits a location, the geometry of the map can be constrained such that it corrects for the odometric error and better matches the true path.

Several maximum likelihood estimators are derived using this virtual sensor methodology. The first estimator uses a physics-inspired mass and spring model to represent the uncertainties in the robot's position and motion. Errors are corrected by relaxing the spring model through numerical simulation to the state of least potential energy. The second method finds the maximum likelihood solution by linearizing a Chi-squared error function. This method has the advantage of explicitly dealing with dependencies between the robot's linear and rotational errors. Finally, the third method employs the iterated form of the Extended Kalman Filter. This method has the advantage of providing a real-time update of the robot's position where the others process all the data at once.

Finally, a method is presented for dealing with multiple locations that cannot be disambiguated because their signatures appear to be identical. In order to decide which sensor

readings are associated with what positions in space, the robot's sensor readings and motion history are used to calculate a discrete probability distribution over all possible robot positions.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This dissertation presents a novel method for a sensor-poor mobile robot to create a topological estimate of its path through an environment. The method is based on the notion of a virtual sensor that equates "place signatures" with physical locations in space. The method is applicable in the presence of extremely poor odometry and does not require sensors that return spatial information (such as range or bearing) about the environment.

Autonomous mobile robots that reason and act in their environments typically need to be able to generate an internal representation of the environment in order to operate effectively. Mobile robot localization is one of the most fundamental problems in robotics and has been examined by many researchers for over a decade. Depending on the environment, the sensors, and the computational power of the robot, this problem can be approached in a number of different ways. This thesis focuses on the problem of robotics localization and map building with small robots that have very limited sensing and/or computational resources.

All robots suffer from the problem of noisy odometry. Slight differences in the speeds of the wheels and small debris or irregularities on the ground will greatly degrade the performance of any dead-reckoning-based position estimate. This becomes more of an issue

for light (less massive) robots that have a lower coefficient of friction between their wheels and the ground. Reduced friction contributes to greater slip between the ground surface and the wheels which in turn causes more uncertainty in dead-reckoning estimates. This physical effect makes any localization or map-construction algorithm that relies exclusively on odometry inappropriate for small robot if they must navigate over long distances. An additional problem with small robot localization comes from the limit on the sensors that can be physically carried. Very accurate sensors, such as commercial laser range finders or stereo camera systems, are generally too large, CPU intensive, and power-hungry for smaller platforms.

Any method for map construction and/or localization must explicitly take into account the large amount of error in the robot's sensing and odometric capabilities. Various methods for dealing with uncertainty have been proposed and studied for some time [9, 14, 22, 30, 55, 58, 70, 91, 93, 97]. These include Bayesian algorithms, such as maximum likelihood estimators (weighted least squares) and the Kalman filter, as well as statistical algorithms such as particle filters. These methods describe the estimates of the robot and/or landmark positions as arbitrary probability densities (not just restricted to Gaussian), where the uncertainty in the robot's sensors and actuators can be directly integrated into the estimate of the system.

In this work, we make the assumption that the robot is capable of obtaining a measurement of its odometry, but that the estimate will be extremely poor. Additionally, we make the assumption that the robot can carry an exteroceptive sensor with which it can obtain a "signature" of features from the outside world. We make no assumptions as to what kinds of features are to be obtained or extracted, but only assume that the landmark signature can be used to compare one physical location of the robot with another. These signatures are not assumed to be unique, nor are they assumed to have any direct correlation with the spatial distribution of the objects in the environment. Unlike laser or sonar sensors, which

return distances to surrounding objects, a feature signature is an abstract representation of the specific position that the robot is in. The only requirement is that the signature does not vary over time and that if the robot returns to nearly the same location, it will encounter the same, or a very similar, sensor signature.

To solve the mapping and localization problem, we propose the construction of a topological "graph"-style map where each node represents a location the robot visited and where it obtained a sensor signature. Initially, the map will contain a node for each sensor snapshot the robot acquired. Thus, if the robot has traversed the same location more than once, there will be multiple nodes in the map corresponding to a single location. In order to obtain an accurate representation of the robot's path through its environment, those nodes that represent the same location in space must be identified and merged. After merging, the constraints that those nodes place on the map must be propagated through the graph such that a structure can be obtained which is consistent with the robot's sensor readings.

This problem can be solved in either a sequential or batch fashion. In a sequential fashion, each pair of nodes is merged on at a time, thus affecting the the relative distance and headings of the nodes around it. Before the next merge, the map must find a stable configuration so that each of the local displacements between the nodes is maintained properly. In contrast, batch methods merge all of the node pairs at once before minimizing the error function.

1.1 Maximum Likelihood Estimator

If the robot's sensors and actuators were perfect, each of these constraints would be satisfied and the estimated path would match the true path perfectly. Since this is not the case, each constraint will have a residual associated with it which represents the mismatch between the true and estimated landmark positions. To find the most likely map, the sum of these constraints (or error terms) must be minimized.

1.1.1 Physics-Based Estimator

A useful analogy to this problem is a physics-based model mass and spring system. Linear distances between each of the nodes can be represented as linear springs while rotational differences between nodes can be represented as torsional springs. The spring constants capture the certainty in the odometry estimates. A very stiff spring represents high certainty in the reading while a very loose spring represents low certainty. This estimator can be run in either a sequential or a batch fashion.

1.1.2 Linearized Maximum Likelihood Estimator

Another method of finding the most likely configuration given the data is to treat all measurements as a sum of quadratic constraint terms and then minimizes the difference between the estimated and measured values for each of the terms. The system of constraints is linearized using a first-order Taylor series expansion around the current state estimate and is iterated until convergence. This is a batch method where all of the constraints are applied before the minimum error state is found.

1.2 Kalman Filter

Solving the simultaneous localization and mapping (SLAM) problem for small, resource-limited robots means doing so without the aid of good odometric estimates and accurate metric range sensors. This causes a problem for traditional solutions which typically require one or both of the above. We propose a modification to the standard solution in which we relax the assumption that the robots can obtain metric distance information to landmarks. We describe a method by which the Iterated form of the Extended Kalman Filter (EKF) processes all measurements, including both actual odometric and inferred relative positions, and estimates the coordinates of the locations where images were recorded along the trajec-

tory of the robot. In this method, landmarks correspond to images taken at various (x, y) positions of the robot.

1.3 Data Association

Perceptual aliasing is the problem of multiple sensor readings that appear to be identical but which correspond to different features or locations in space. Mistakenly identifying two different sets of features or spatial locations as being the same can introduce very large errors into the estimator.

The robot must identify this situation and attempt to correct for it. In this work, the history of the robot's path and previous sensor readings are used to identify locations in space that are likely to be the same. A technique called Markov localization [29] is used to determine the probability of the robot's position at each timestep. These positions must be combined in order to generate a map which correctly matches the topology of the environment. A measure of the entropy of the pose estimate is used to verify the correctness of the merged positions.

1.4 Experimental Validation

The University of Minnesota Distributed Robotics project has developed a group of micro-robots called Scouts [43, 83, 88], shown in Figure 1.1. The robot's small size (11 cm long and 4 cm wide) and light weight (approximately 200 g) allows it to be easily carried by a human or another robot. The Scout is able to navigate through most indoor environments using its differentially-driven wheels and can climb a 20° slope. The Scout can also jump over small obstacles, such as stairs, by winching its leaf-spring tail around its body and snapping it out. Scouts carry a sensor payload, usually a small camera, used to broadcast environmental information over an analog RF transmitter. Scouts can also transmit and

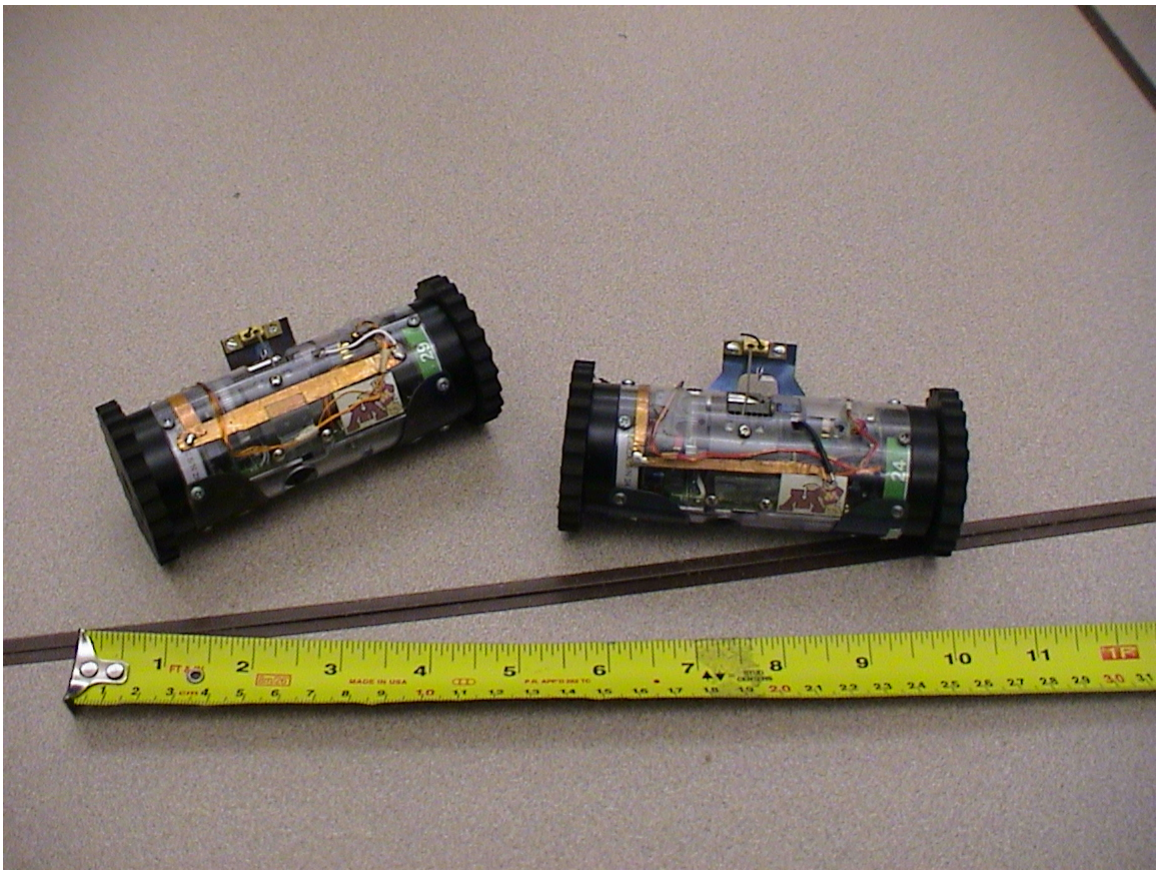


Figure 1.1: Two Scout robots shown next to a ruler for scale (inches on top, centimeters on bottom). The robots are 11 cm long and 4 cm in diameter.

receive digital data over a separate RF modem using a custom network protocol. Due to their small size, Scouts have extremely limited on-board computational power, requiring the full capacity of their two CPUs for network communications and actuator control. Motion commands are transmitted to the robot from a remote source.

These tiny robots can be controlled remotely by a larger all-terrain robot called the Ranger, shown in Figure 1.2. In Rybski *et al.* [86], a group of two Rangers deployed several Scouts into various rooms and controlled them to set up a motion-detection sensor network. In this work, the Scouts used simple visual control, servoing to light and dark areas and using image differencing to tell how far the robot moved.

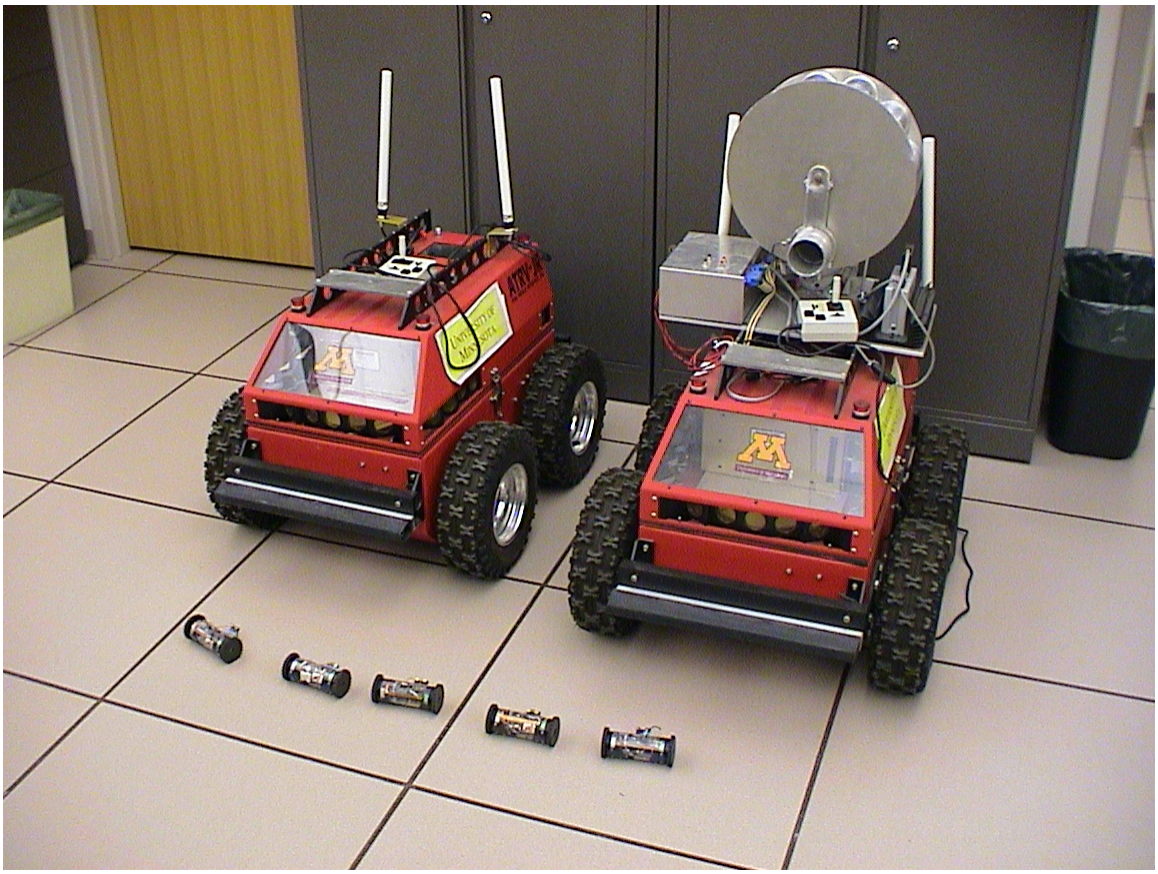


Figure 1.2: Two Rangers and several Scouts. The Ranger on the right is outfitted with a custom launcher which can ballistically deploy a Scout up to 10 m away from it.

In order to use appearance-based mapping with the Scouts, they must be outfitted with a sensor that can adequately generate a signature from each location the robot has visited. An example of a sensor which can be used in this fashion is shown in Figure 1.3. Here, a Scout is equipped with a 190° fish-eye lens which can obtain a full panoramic view of its surroundings. Other sorts of sensors that could be used include: RF signal strength monitors, which are able to sense and return the signal strength from multiple RF beacons; magnetic signatures sensors, which obtain a reading from multiple ferrous objects in the local vicinity; or even chemical composition sensors, which can identify locations by the presence (or lack thereof) of different compounds.

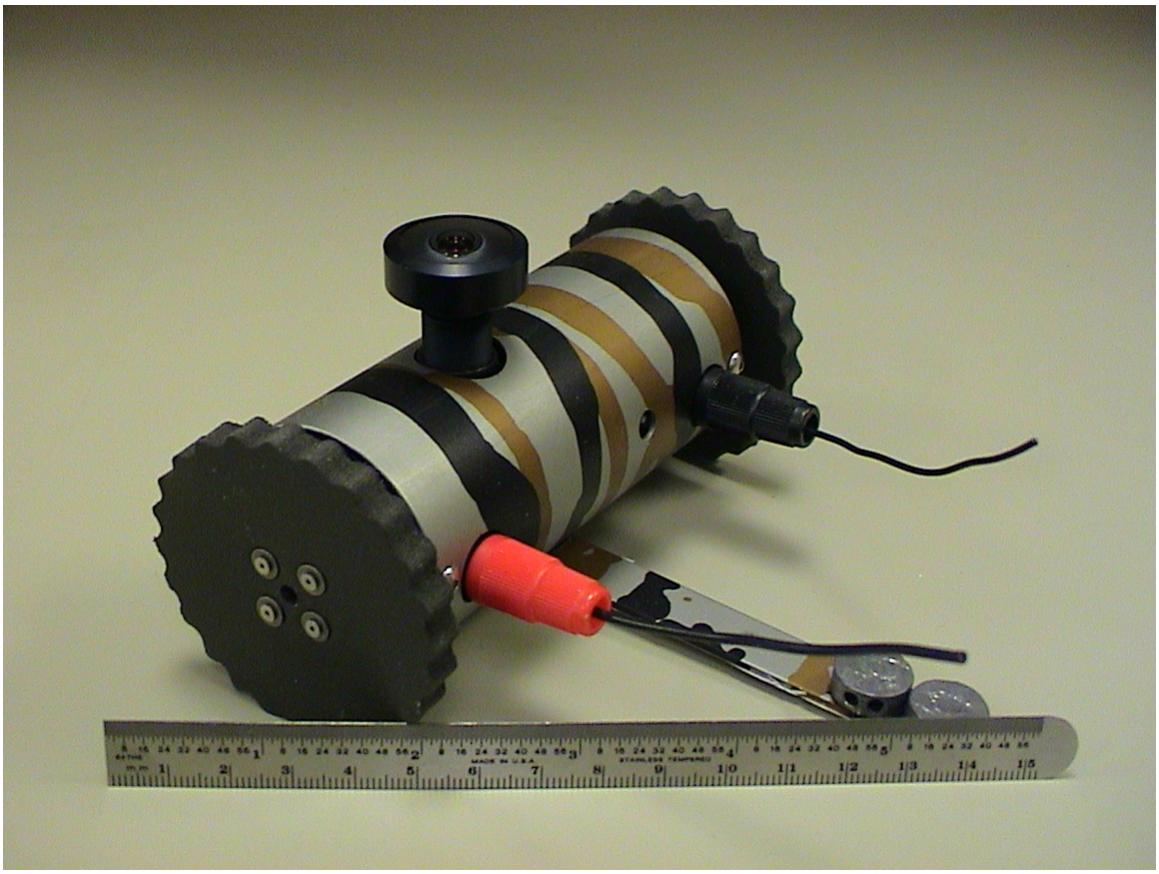


Figure 1.3: A Scout robot with an upward-facing Omnitech 190° fish-eye lens. The robot is 11 cm long and 4 cm in diameter (ruler scale is in cm).

1.5 Contributions

This thesis contributes to the area of miniature robots which have limited hardware capabilities and presents novel spatial reasoning methods for surveillance tasks. The major challenge with miniature robots is their limited hardware capabilities in terms of processing power, odometry, types of sensors, and communication. These limitations make currently available localization and mapping methods completely unusable for small robots and forces the programming of these robots to be limited to reactive methods. This thesis takes the notion of an appearance-based sensor that does not return either bearing or range measure-

ments to landmarks and combines it in a novel way with various estimators that allow the robot to simultaneously localize and map (known as the SLAM problem). This results in a collection of algorithms that extend the applicability of SLAM methods to a whole new class of robots to which they could not be applied before. The use of SLAM methods has the advantage of providing this broad class of robots with a sound and principled way of localization and mapping. The extension of SLAM to robots with limited odometry and no range or bearing sensors opens up new opportunities for other devices, such as sensors with some mobility used in a sensor network, and allows even larger and more capable robots to avoid using precise range sensors when mapping.

A brief analysis of two case studies of small robot teams is presented in Chapter 2. These two groups of robots illustrate the typical complexity of the robot hardware best suited to use the SLAM methods described in this dissertation. A series of experiments are presented that show the utility and limitations of purely reactive robot control methods. These results serve as the primary motivating factor for this research.

Specific contributions include:

- The notion of an appearance-based sensor for SLAM which allows a robot to localize and reconstruct a path estimate without the need for range or bearing information to environmental landmarks. A virtual sensor is assumed which associates purely qualitative measurements of landmarks to robot positions. These measurements are stored as abstract "signatures" that correspond to specific (x, y) positions along the path of the robot. This is the primary contribution of this dissertation.

- A description of a physics-inspired spring/mass method for reconstructing the robot's path from the appearance-based sensor data. This method uses an heuristic based on energy minimization in for approximating the maximum-likelihood estimate of the robot's path through the environment.

- The derivation of linearized maximum likelihood estimator designed to operate over the complete set of sensor data (in a batch rather than sequential fashion). This analytical approach differs from the more empirical spring/mass estimator in that the uncertainty in translation and rotation is handled in a consistent fashion. As a result, this method is more robust but requires higher memory and computation requirements.

- The derivation of an iterated extended Kalman filter that takes into account the actual odometric and inferred relative positions of landmarks, and estimates the coordinates along the robot's trajectory where sensor readings were recorded. The Kalman filter is a sequential estimator which processes sensor readings as they are received by the robot, allowing for more instantaneous position information.

- A method for handling the data association problem, where multiple locations in space have location signatures similar enough that they cannot be disambiguated with a direct comparison of the signatures. The robot's sensor and path history is integrated over time to generate a probability distribution over the space of all possible poses. In environments where the robot's path has significant overlaps, this method identify multiple locations that are the most likely to be the same. A rejection criterion based on entropy is also presented.

Experimental results are presented throughout this dissertation both in simulation and using a miniature mobile robot with an omnicamera in an indoor office environment. As it traverses the environment, the robot's path is reconstructed using the estimators developed in this thesis and the results are compared. The results support the hypothesis that these estimators are capable of reducing the error in the robot estimates of its path even when the odometry is very poor and the only sensory information available is in the form of location signatures. Results show that the linearized maximum likelihood estimator produces the

best results. The spring-based maximum likelihood estimator and the Kalman filter are fairly close in estimate quality until the robot's odometric error exceeds a threshold at which point the estimation quality of the Kalman filter decreases significantly.

1.6 Dissertation Roadmap

This dissertation is organized into the following chapters:

- Chapter 2 describes two case studies involving small resource-limited mobile robots to help motivate the utility of this research.
- Chapter 3 examines and summarizes the related work in the field of mobile robot mapping and localization.
- Chapter 4 defines the specific mapping and localization problem being solved, lists the assumptions made, and describes the introduced framework for localizing robots with poor sensing and odometric estimates.
- Chapter 5 describes two batch maximum likelihood estimator solutions to the problem with simulation results. The first is a physics-inspired model that uses a mass/spring abstraction to represent the robot's knowledge. A numerical simulation of the mass/spring dynamics is used to find the most likely estimate. The second estimator is a linearized version of the non-linear system in which the errors in the sensor readings are taken into account.
- Chapter 6 describes how an extended Kalman filter estimator can be derived to recursively find the maximum likelihood solution to this problem. Simulation results as well as further analysis are provided.
- Chapter 7 describes a method to handle the data association problem, where individual locations in space are not assumed to be uniquely identifiable. Since sensor

readings can be associated with multiple locations in space, information regarding the previous sensor/motion history of the robot and the level of certainty in the robot's position is included.

- Chapter 8 describes experiments using real sensor data and physical robotic hardware. The relative performance of the different estimators operating over the same dataset is also presented.

  The results of the various estimators are described on this dataset.

- Chapter 9 provides a summary of the results and analysis of this research as well as a re-iteration of the contributions.

- Chapter 10 lists the future directions for this research.

- Appendix A is a tutorial showing how a simple 1D Kalman filter and 1D maximum likelihood estimator can be derived.

- Appendix B illustrates the derivation of the dual-pass Markov localization step used in Chapter 7.

Chapter 2

Challenges with Small Robots

This purpose of this chapter is to provide a historical perspective behind the SLAM methods developed in this dissertation. Much of the motivation for this work has stemmed from experiences with robots that have very limited sensing and computational capabilities. Miniature robots, in particular, are interesting because of the challenges in dealing with their limited capabilities.

Why study systems with limited capabilities at all? Moore's law has thus far guaranteed that virtually any algorithm will be able to run on a mobile robot's on-board CPU if one were to simply wait for the market to release the appropriate hardware. [1] Likewise, advances in miniaturization and MEMS-based processes will improve the accuracy of sensors while making them smaller and lighter. This means that mobile robots can easily make use of scanning laser range finders, RF location beacons, GPS, multiple digital cameras, etc. while more advanced sensors are undoubtedly on the way. Fortunately, the answer to this question stems from the same entrepreneurial spirit that drives the advances in CPU and sensing technology. As electromechanical and computational hardware becomes less expensive and more capable, people dream up new ways to make use of it which greatly pushes the

[1] Speculation runs rampant on how long this will remain to be true, but at the time of this publication, speculation suggests that this will remain true for a number of years.

envelope of its capabilities. The distributed robotics project (funded by DARPA) that was responsible for the creation of the Scout robot was one of these examples. To be competitive for winning the contract, all submitted proposals had to describe the creation of a robot that fit into a 5 cm cube. With the technology available in 1999, creating such a robot that fit the size specifications and was still practical and useful was a major challenge.

Additionally, just because one has the capabilities of loading a robot with sensors does not mean that this is an appropriate thing to do. Battery technology is not advancing as fast as the electronics that require it. Thus, robots are still greatly limited by their on-board power requirements. Sensors and the CPU resources required to process them all require power and increases in the former always requires increases in the latter. Thus, there will always be design tradeoffs that must be made in terms of the robot's capabilities vs. its operational lifetime as well as its cost to build and use. Such costs are accentuated in multi-robot systems where there could be hundreds of robots. In these cases, the individual costs of the robots become a major design parameter. These tradeoffs will guarantee that there will always be frontiers to explore with robots for which there is no readily available hardware solution.

There are two lines of previous research that examined some of these issues. The first was the MinDART, a simple multi-robot team designed to solve a foraging task. This work examined the effect of different search strategies, sensing modalities, and communication abilities compared on the performance of the team. The second was previous work with the Scout robots in which the challenges involving their limited bandwidth communication systems were addressed so that they could function together and solve surveillance tasks.

2.1 Foraging with Simple Robots

The MinDART (Minnesota Distributed Autonomous Robot Team), shown in Figure 2.1, is a group of simple homogeneous robot platforms designed to solve a foraging task [84,

85]. Designing a distributed robotic system whereby simple homogeneous units solve a complex task through emergent behavior is an attractive engineering solution for many reasons [8]. Analogous to software design, there are obvious advantages to a modular approach, including reducing a complex task to a number of simpler, more manageable ones. There also arises a natural redundancy in the resultant system, as well as the ability to scale to larger tasks with minimal tractability issues [26]. The key in designing such a distributed system is to create an individual that is just smart enough, or just complex enough, to solve the problem. Complexity comes at a price, both in money and time, and by minimizing the complexity, thus the cost, the hope is to create a simple, efficient, and tractable system.

The robots were designed to solve a foraging task where they search for and locate targets in an enclosed arena and return them at a designated location. Two sets of experiments were run, whereby the task performance relative to team size, target distribution, and control strategy was evaluated. In the first set of experiments, several factors were considered, including the utility of explicit GPS-style localization as well as the performance effects of team size and target distribution. Localization was accomplished by calculating the bearings to landmarks at a known location. In the second set of experiments, control strategies using communication were tested. The communication mechanism consisted of beacons on the robot's chassis that could be lit up. Using this, the robots could make themselves to each other.

The MinDART control software consists of a set of parallel sensory-motor behavior processes, similar to the Subsumption algorithm [11]. Each process is responsible for handling one segment of the robot's control code by mapping sensors to actuators. Their task was to roam a closed environment and search for targets (infrared transmitters) to return to a drop-off location. The drop-off location was defined by a beacon (a light bulb or colored landmark) that the robots could detect. In addition, they could sense collisions with

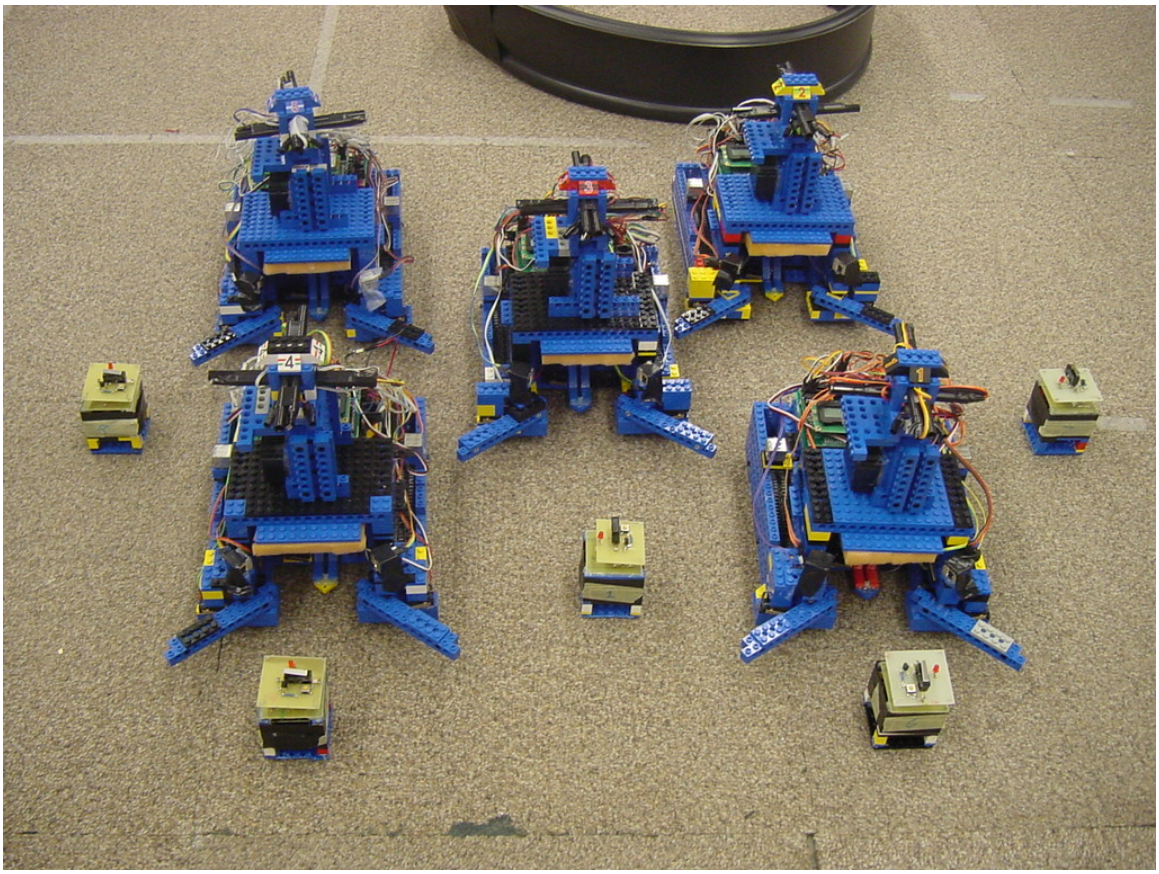


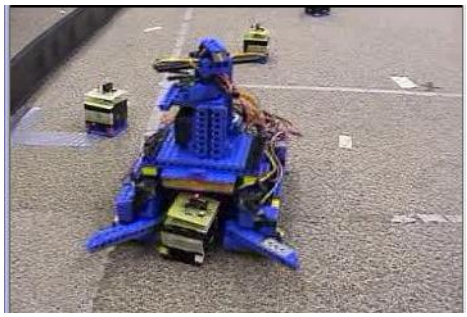
Figure 2.1: The Minnesota Distributed Autonomous Robot Team (MinDART). These robots were used to solve a foraging task and to examine issues regarding solutions that involved robots with limited resources. This image shows the robots equipped for the localization experiments.

obstacles and other robots with their bumpers, and detect the IR targets at a range of approximately 1 m. To solve this task, the robots started from a fixed location, searched an area for targets and returned them to a drop-off zone. Figure 2.2 illustrates a sample run of the robots performing their foraging task.

The localization experiments were run with one-, two-, and four-robot configurations. The robots were not explicitly aware of each other's presence and simply treated each other as obstacles if they collided. Target locations were either distributed uniformly or all

(a) Robot searching for targets

(b) Robot interference



(c) Robot grabbing a target

(d) Robot returning a target

Figure 2.2: An example run of the MinDART illustrating the environment and some of the interactions between the robots. All experiments contained nine targets and eight obstacles. In the localization experiments (shown in these images) three collinear lightbulbs were placed at known locations and were used to determine position and orientation. Obstacles are relatively low and do not block a robot's view of the landmarks but do block a robot's view of the targets.

lumped together into a single location (non-uniformly). Some experiments were run using localization while others were not. Without the ability to localize, the robots only searched randomly.

The experimental setup for the communication experiments was nearly identical to the localization experiment. The main difference was the use of a single colored landmark, which robots identified with their cameras. This is in contrast to the three collinear lightbulb landmarks of the previous experiments, which were identified with CdS photoresistors. The

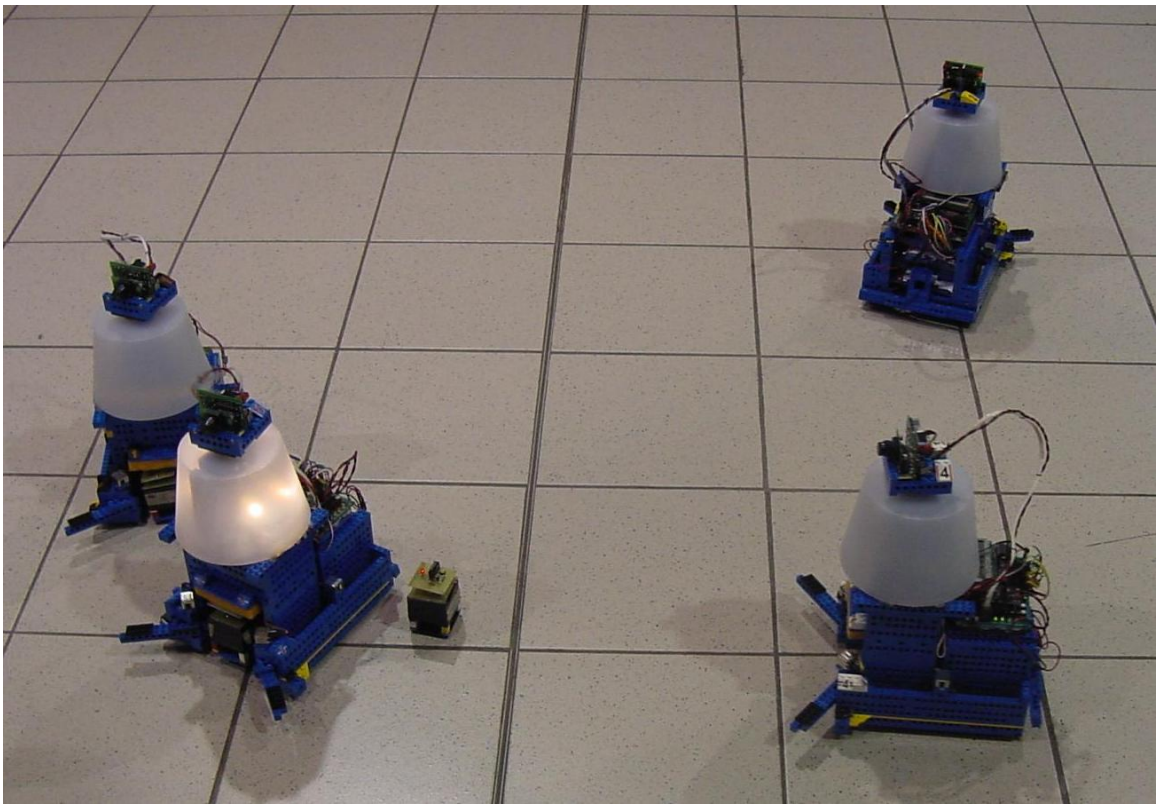


Figure 2.3: The Minnesota Distributed Autonomous Robot Team (MinDART) equipped for the communication experiments. A robot on the left has activated its beacon and is leading another robot to a target.

targets were placed in a single non-uniform distribution in the corner of the environment furthest from the drop-off location. All experiments were run with four robots. Robots communicated with their light-bulb beacons, as shown in Figure 2.3, which could be seen by another robot at a maximum range of 2.9 m.

Each of the 17 different experimental cases was run five times. In the first set of experiments, it was observed (non surprisingly) that localization assisted the robots in the non-uniform target distribution case but not in the uniform target distribution case. However, the overall benefit of doing localization was lost due to the additional time required to infer a position measurement (5-18 seconds.) If the time to localize was completely dis-

counted, the robots were much faster at finding their way back to a new target once one had been dropped off.

Another hypothesis tested by these experiments was that adding more robots would greatly increase the performance of the team, but continually increasing the number of robots would not be as beneficial. This was also observed in that four robots generally did not improve the performance over two robots as much as two robots did over one. Additionally, significant interference was observed between the robots when they tried to obtain targets in the non-uniformly distributed environment, which added further evidence to this claim. However, when analyzing the variances of the task completion times for these experiments, in nearly all cases, the variances decreased as the number of robots increased. Thus, while the average runtime may not improve as much when adding more robots, the performance of the team becomes more consistent.

The communication experiments were designed to test the robot's abilities to lead each other to a single clump of targets. One hypothesis was that once one robot located the group of targets, it would lead the others to that group and thus the overall time to pick up the targets would decrease. However, there was no statistically significant difference in the mean times necessary to complete the entire task between any of the communication experiments and the non-communication experiments. Instead, a trend toward minimization of the variance in the performance was found.

In the final analysis, it was found that increasing the complexity of the robot's search strategies did not help to decrease the mean time to complete the task (in a statistically significant fashion). However, in all cases, increasing the complexity helped to reduce the variance of the team's performance when solving that task. When either localizing or communicating, a robot would have to remain stationary for a fixed amount of time. However, by doing this deliberative behavior, the robots spent less time randomly searching for targets. Random walk introduced a much higher variance in the total time to complete the

task and thus the deliberative approaches made the entire team's performance much more consistant.

2.2 Placing a Sensor Network with Limited Communications

The Scout robots were developed for surveillance and reconnaissance purposes. The initial goal of the project was to develop a portable mobile camera that could be manually controlled in areas that may be hazardous to humans so that video data from those areas could be obtained. Because the robots are so small, several could be carried and deployed by a single person. However, it is a very difficult task for a human to control more than a single robot at a time, and so giving the Scout some autonomy is necessary. The small size of the robot makes this challenging for several reasons.

Due to the Scout's limited volume and power constraints, the two on-board computers are only powerful enough to handle communications and actuator controls. There is very little memory for any high-level decision process and no ability to process video. In order for the Scouts to accomplish anything useful, they must be paired with an off-board computer or a human operator.

Scouts receive command packets transmitted by a radio device connected to a remote computer. Each Scout has a unique network ID, allowing a single radio frequency to carry commands for multiple robots. By interleaving packets destined for the different robots, multiple Scouts can be controlled simultaneously.

Video data is broadcast over a fixed-frequency analog radio link and must be captured by a video receiver and fed into a framegrabber for digitizing. Because the video is a continuous analog stream, only one robot can broadcast on a given frequency at a time. Signals from multiple robots transmitting on the same frequency disrupt each other and become useless.

The RF limitations of the Scout pose two fundamental difficulties when attempting to

control several of them. First, the command radio has a fixed bandwidth. This limits the number of commands it can transmit per second, and therefore limits the number of Scouts that can be controlled simultaneously. Second, the absence of many independent analog video channels reduces the number of robots that can simultaneously transmit video. There are generally not enough commercial frequencies available to allow for a large number of interference-free simultaneous analog transmissions. As a result, if more robots than independent video frequencies are used, video can only be captured by interleaving the time each robot's transmitter is turned on. Thus, an automated scheduling system is required to ensure that the robots share the limited communications resources and do not interfere with each other's transmissions.

2.2.1 Scout Control Architecture

Substantial effort went into designing and implementing a control architecture, shown in Figure 2.4, that would facilitate the operation of multiple Scout robots over a limited bandwidth communications system [66, 87, 94]. This software architecture controls access to hardware resources across a network of computers and manages control requests from client processes. Client processes can consist of user interfaces for teleoperation as well as behaviors for autonomous operation. There are four distinct components of the system: Mission Control, Resource Pool, User Interface, and Backbone.

All behaviors and decision processes are contained and managed from within the *Mission Control*. A complete multi-robot mission can be built out of discrete behaviors and controlled by a single human operator. Behaviors are organized in a hierarchical fashion, where "parent" nodes spawn off "children" to complete subtasks. All behaviors have pre-assigned priorities that determine how they can receive access to the hardware.

Human control of the resources in the system is provided through the *User Interface*. This system allows a human operator to connect to robots directly and command them

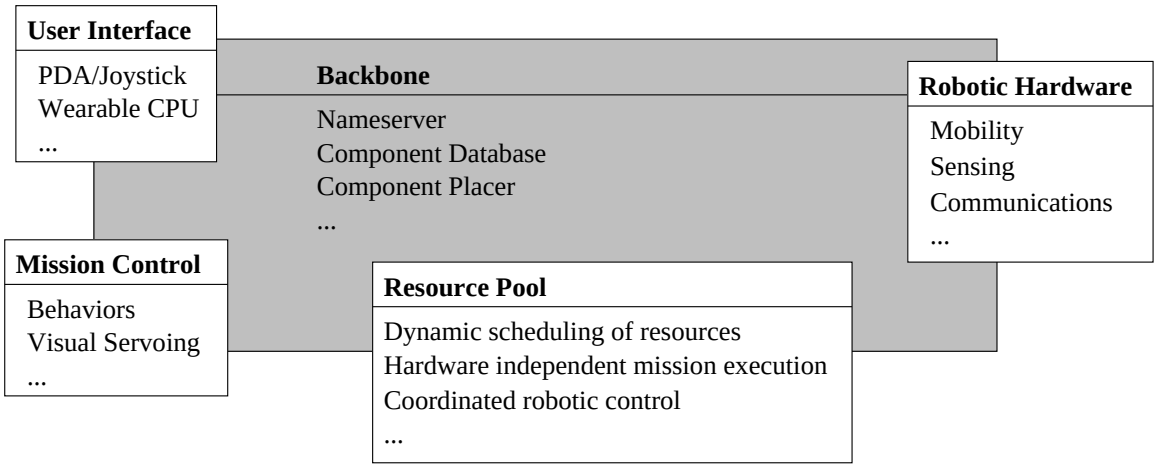


Figure 2.4: High-level diagram of the Scout control architecture. Control processes such as behaviors and user interface consoles connect to robotic hardware after passing through the resource pool. All system components are connected by a backbone consisting of CORBA services.

from a console.

In order to control hardware, behaviors and user interfaces must connect to components in the *Resource Pool* which controls access to robotic hardware and other computational resources. The *Resource Controller Manager* is the central component that oversees the distribution and access to the hardware. This component makes use of a centralized real-time resource scheduler to allocate resources according to the current demand of all running clients.

Connecting everything together is a CORBA-based [39] group of core services called the *Backbone*. The Backbone is responsible for starting and stopping all of the components across the network.

2.2.2 Performance Evaluation of Shared Bandwidth Control

Sharing the bandwidth among different robots affects the performance of the Scouts and thus a series of experiments were performed to identify these effects [88]. In these experi-

ments, the Scouts are used in a distributed surveillance task where they are deployed into an area to watch for motion. This is useful in situations where it is impractical to place fixed cameras because of difficulties relating to power, portability, reconfigurability, or even the safety of the operator. In this task, the Scouts can either be deployed into their environment by a human or another robot, or they can autonomously find their way into useful areas. The Scouts' ability to accomplish the surveillance task was examined with a series of experimental runs. These experiments were designed to test the individual and team performances of the Scouts and the controlling architecture in a number of different situations and group configurations. Of specific interest was how the scheduling system handles multiple Scouts all trying to use the limited RF video channels to detect motion.

Figure 2.5 shows the experimental environment. At the start of each trial, the Scouts were placed in the center starting area. They autonomously located a place along the periphery of the environment to hide themselves, moved there, and then watched for a target that moved into the room and out again. Four different cases were tested in the experiments, including a single Scout using a single video frequency, two Scouts sharing a single video frequency, two Scouts using two different video frequencies, and four Scouts sharing two different video frequencies.

	1 Scout 1 Frequency	2 Scouts 1 Frequency	2 Scouts 2 Frequencies	4 Scouts 2 Frequencies
Area (in m^2) of field of view	$\mu = 4.73$ $\sigma = 2.89$	$\mu = 7.35$ $\sigma = 2.39$	$\mu = 8.09$ $\sigma = 2.34$	$\mu = 10.08$ $\sigma = 2.27$
Time (in s) target within field of view	$\mu = 12.0$ $\sigma = 6.6$	$\mu = 16.6$ $\sigma = 5.1$	$\mu = 18.4$ $\sigma = 4.2$	$\mu = 21.3$ $\sigma = 4.0$
Time (in s) target motion reported	$\mu = 6.5$ $\sigma = 2.8$	$\mu = 1.1$ $\sigma = 0.9$	$\mu = 7.4$ $\sigma = 2.4$	$\mu = 4.5$ $\sigma = 1.4$

Table 2.1: Experimental results (means and standard deviations) from all trials.

As shown in Table 2.1, by having to share the RF bandwidth, the Scouts tended to miss much of the motion. On average, only 29% of all possible motion was detected by the

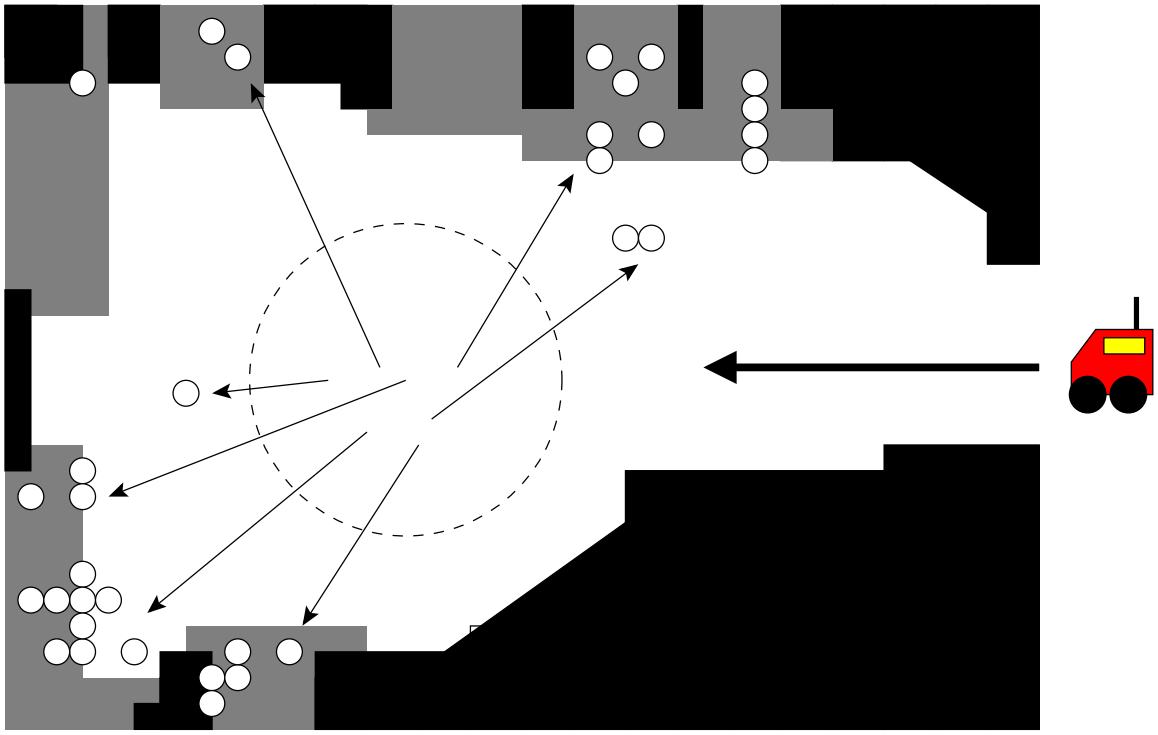


Figure 2.5: Automatic placement of the Scout sensor network. For each of the experiments, the Scouts started somewhere in the center circle and found places to hide. Black areas are impassable obstacles and gray areas are hiding places. Positions that the Scouts found for themselves in the 6 m by 4.2 m are represented as white dots. Once positioned, the Scouts watched for the motion of a larger robot which entered the room from the right.

Scouts across all of the experimental cases. However, this does not mean that the Scouts were unable to succeed in their mission. The robots successfully reported that there was *some* motion in 95% of all of the trials. This indicates that this system is more suited for a binary motion detection task when the system has to share access to the RF bandwidth rather than a continuous motion detection task where the cameras had to be looking for the motion for long periods of time.

2.3 Toward Spatial Reasoning for Small Robots

The key similarity between both sets of research projects is that the variance in the performance of the robotic systems was directly related to the interactions between the robots and their environment. The local behavior strategies were fairly random in nature and thus the final performance of the robots could not easily be predicted. In the first set of MinDART experiments, the robots used features in their environment to localize themselves (somewhat akin to GPS) when returning to a location where they last saw a target. However, they used local control strategies to aim themselves toward their home base when dropping off a target. In the second set of experiments regarding communication, the robots used each other as beacons to home in on.

Similarly, the Scouts used local visual servoing strategies to deploy themselves into their environment. This local control strategy was chosen because of the lack of spatial knowledge and environmental information. If the robots could localize themselves, they could place themselves in a much more deterministic fashion. From the experiments, it was observed that the robots would often end up going to the same location and having completely redundant views. Being able to localize in their environment would address this problem.

The next step of capability development for these and similar sensor-poor robots is to empower them with the ability to localize within the environments that they operate in. However, any localization method must be explicitly modeled in such a way that the limitations of the robot's capabilities are taken into account.

Chapter 3

Related Work

3.1 Map Construction and Localization

A great deal of work has been done in the realm of completely behavior-based or reactive robots, which do not require the use of explicit maps [3, 11, 35, 63]. However, none of these robots are capable of accomplishing structured tasks as complex as giving tours in museums, such as the highly successful RHINO [96] and MINERVA [95].

There are also situations where the robot is required to learn the map directly. In these situations, robots must learn their surroundings by generating a concise and accurate map representation that fits the accuracy of their sensors. Great successes have been made by using statistical methods and representations for these maps [14, 30, 97, 106, 90, 91, 107].

The map-building problem can also be addressed in the framework of multiple robots. A team of multiple robots can be quite successful in cooperatively building environmental maps [14, 19, 98, 106]. This is particularly useful since multiple robots can cover more ground at a given time than a single robot. Another strength of using multiple mobile robots for mapping is that robots which are very certain of their positions can help to localize those that are not [30, 80]. By assisting to localize their teammates, the map built

by a group of mobile robots is likely to be more accurate than a map built by a single robot.

The environments that teams of robots are sent to explore and map are not always very accessible. In fact, very interesting environments, such as in sewer pipes or zero-gravity, often require a very specialized robot to carry out the task [4, 16, 33, 42, 47, 68, 79, 105]. Most existing mapping and localization algorithms have been designed for a particular level of robot complexity. The robots that run these algorithms generally include relatively accurate range sensors and odometry [30, 91, 98]. Very small robots may not have the power or space to be able to carry such devices or have much accuracy in using them. However, the demands on these robots are just as high, if not higher, than their larger counterparts, due to the environments in which they are expected to operate. In order to properly function, very specialized sensor and motion models must be developed for these robots if they are to explore and map their environments. Algorithms inspired from biological sources are being developed which can be applied to these domains. For instance, vision algorithms modeled after the amazing visual abilities of insects are being explored which will allow a very small vision-equipped robot to navigate effectively through a large environment [15, 50, 75, 104].

Closely coupled to the challenge of mapping is the challenge of localization. Once a mobile robot has access to a given map of its environment, it must be able to localize itself with respect to that map so that it can effectively plan where to go next. There are three variations of the localization problem. The first is tracking a robot from a known starting position on the map [9, 18]. The second is attempting to determine the robot's position when no starting position is known [12, 31, 72, 91]. The second problem is far more difficult as the robot's initial starting position is represented as a uniform distribution across all possibilities. Most of the techniques that solve this problem are some varient of Markov localization [29], where a history of the robot's sensor and odometric data is used to create a probability density estimate of the robot's position (see Appendix B for a more complete description of this algorithm). The third problem is to recover the robot's position estimate

after it has been moved from its known position to a completely unknown position. This is sometimes called the "Kidnapped Robot" problem [28, 32] and is very challenging since the robot must recognize that it is no longer in its current position estimate and then must globally relocalize itself.

In this work, we assume that the mapping and localization steps of the robots are tightly correlated. That is, the robot must continually build a map and localize itself in that map as it explores the environment. This is typically refered to as simultaneous localization and mapping (SLAM).

3.2 Grid-Based Mapping Techniques

Mapping techniques that represent the space of the world explicitly as a two- (or three-) dimensional grid have been used with great success [14, 30, 106]. These methods generally make use of a map representation known as a Certainty or Occupancy Grid [27, 52, 70], which explicitly encodes the probability of each cell being occupied by an obstacle or being free of obstructions. Most of these methods rely on Bayesian updating and filtering, where the modes of arbitrary robot pose distributions are computed. Some extensions to this work include a real-time maximum likelihood estimator [40, 98] as well as a more accurate off-line method [97].

These methods typically require very precise sensor modalities (such as laser or stereo camera systems) that allow them to accurately determine the distance to obstacles in the environment. Because most miniature mobile robots such as the Scouts are not equipped with such accurate range sensors, they cannot easily construct geometric maps of an environment. The physical size of most range sensors makes them unusable by robots of the size that we are examining. A notable exception to this rule are the CMU Millibots [5]. These robots are on the same scale as the Scouts and carry ultrasonic range sensors for gathering range measurements, as well as doing inter-robot localization.

3.3 Topological Mapping Techniques

Topological mapping techniques [53, 55, 56, 62, 76, 90, 91, 106] represent the space of the world as a graph. In this paradigm, spatial locations are typically described in a more qualitative fashion rather than given explicit spatial extent. Individual nodes on the graph represent significant locations that are distinguished from other locations by a particular set of sensor readings. As an example, Kortenkamp [53] suggests the use of various kinds of gateways (such as doorways) as distinguishing features. Topological maps are particularly useful because they can deal with arbitrarily large errors in the robot's odometry. However, compared to metric maps, topological maps are only capable of a coarse representation of the environment.

The idea of using topological maps for representing spatial relationships is very useful. Evidence suggests that humans and other biological systems do not store explicit metric maps of their environments in their brains. Instead, cognitive maps consisting of groupings of features, landmarks, and other images are more likely [65]. Ben Kuipers defined a hierarchy of spatial representations called the Spatial Semantic Hierarchy (SSH) [55]. This hierarchy extends from the notion of sensory/motor primitives that define the lowest levels to detailed grid-based representations of free space at the highest. This lowest part of the SSH is called the "Sensor and Control Level" [54]. Specific location in space, called *Distinct Places* (DPs), are associated with a distinctiveness measure or function which takes as input the robot's sensor readings and returns a value which is maximized when the robot is located in the exact coordinates of that place. *Distinct Travel Paths* (DTPs) represent the sensor readings gathered by the robot as it travels along some local feature, such as down a corridor. As an example, assume that a robot equipped with ultrasonic sensors were to traverse a hallway and stay equidistant to the left and right walls. As long as the sensor readings do not change much, the robot will know that it is still in the corridor. As soon as the sensor readings start to change, the robot would assume that it is approaching a

DP and start homing in on it. The next level in the spatial hierarchy is the topological level, which connects the DPs and DTPs into a graph structure of nodes and edges where connectivity and shortest paths can be reasoned about. On top of the topological layer is the geometric layer which adds Euclidean distances between the DPs to the model. Our work is inspired by the SSH philosophy and attempts to wrap it into a more formal and robust representation using the maximum likelihood techniques.

3.4 Batch Maximum Likelihood-Based Estimators

Physics-inspired models that involve spring dynamics have been used quite effectively to find minimum energy states in topological map structures. We have had some success with these methods [89] but have found that the parameter choices for the models tend to be very important, and that numerically solving for the set of non-linear equations can be unstable.

Andrew Howard's work is the closest in flavor to this. In [45, 44], he shows how to define a maximum likelihood estimator (MLE) from the energy equations of a mass and spring system. Tom Duckett [24] uses a combination of local metric maps and connects them by using a MLE to help bound the errors introduced by odometry. He assumes that the data is globally aligned because of additional orientation sensor information such as a compass. In our work, we make no such assumption as it has been our experience that compasses are notoriously inaccurate when used on small robots that are very close to the floor.

Another spring-based MLE is that of Golfarelli *et al.* [36]. A spring model is used to describe the robot's certainty in its own motion. As the robot continues to move between areas it has already visited and the certainty in the estimates becomes greater, the spring constants become stronger. In their work, the robot traverses routes between locations and each time the robot re-visits a location, it is assumed to be traversing the same route to get there. For the methods presented in this dissertation, locations can be visited by an arbitrary number of routes. The specific locations are analyzed to determine whether they

have been re-visited rather than the routes that were taken to get there. Also, we present several different flavors of MLEs in addition to a spring-inspired model and compare the relative performances of each one.

Verena Hafner [41] used a Kohonen self-organising neural network to learn the spatial similarities between places. Euclidean distances between the points are represented as a mass/spring system. The cognitive model generated by the robot depends on the initialization of the neural network and can vary even on multiple runs with the same sensor data. Additionally, the spring constants for the spatial representations are all set to be the same value. In our approach, we strive to reproduce as accurate a reproduction of the spatial relationships in the robot's path as possible rather than the more abstract biologically-inspired representation described in the above work. Addtionally, the spring constants of our physics-based estimator are direct representations of the uncertainty in the robot's position and vary between springs.

Dan Boley *et al.* [7] introduced a recursive total least squares estimator which had better convergence properties to the Extended Kalman Filter. In this work, it was assumed that the robot could track a single feature and compute bearing information to it. Our work does not assume that bearing information is available and operates without it.

3.5 Extended Kalman Filter

The extended Kalman filter (EKF) has been used for localizing [58] and SLAM [93] on mobile robots for at least a decade. Ever since the EKF was adopted by the robotics community, a great deal of work has gone into methods for optimizing its efficiency and memory usage. In previous implementations of SLAM algorithms, it is frequently assumed that the robot is able to measure its relative range and bearing with respect to features/landmarks [23, 71]. However, Deans and Hebert [20] discuss methods in which bearing-only sensors can be used. Analogously, Kantor and Singh [49] discuss methods in which range-only sensors can

be used. Our approach differs from traditional EKF estimators in that we do not have the ability to resolve specific geometric information (bearing or range) about the landmarks we observe in our environment. Instead, the landmark positions are explicitly coupled to the position of the robot.

Another variant of Kalman filter that can be used to handle nonlinear systems is the iterated extended Kalman filter (IEKF) [34], [64]. In this form, the state is updated in an iterative fashion where the Jacobian of the measurement equation is recomputed at each step in order to reduce the errors introduced by the linearization of the system. The IEKF has been used for solving the SLAM problem on a mobile robot equipped with ultrasonic range finders [17], as well as estimating the motion of a camera from a monocular image sequence [10].

3.6 Particle Filters

Particle filters are statistical estimators in which the probability distribution of the robot's position is represented only as samples drawn from that distribution. Thus, they are capable of representing arbitrary distributions where as MLE and EKF represent the pose estimates as Gaussian distributions. Particle filters are of interest to the statistics [61, 92], vision [48], and target tracking [37] communities. In the robotics community, particle filters are commonly referred to as Monte Carlo localization [21, 99] which is very similar in flavor to Markov localization except that instead of representing the robot's pose as a discrete probability mass, i.e. a 1D or 2D histogram, the probability distribution is represented by a set of samples drawn from that distribution. Particle filters have recently been combined with the EKF to construct maps of highly unstructured environments as well [69].

3.7 Structure From Motion

Structure from motion (SFM) [10, 20, 22] algorithms compute the correspondences between features extracted from multiple images to estimate the geometric shape of landmarks as well as to estimate the robot's pose. While we use a vision system to identify locations by their feature signatures, the features are not explicitly tracked over time. We intend to show that one can do away with the assumption that both range and bearing information to landmarks is available. Without bearing information to correlate positions of landmarks, SFM algorithms cannot be employed to reconstruct the shape of the environment. In practice, the vision system used in this work could be replaced by another kind of Boolean sensor modality. Any sensor can be used as long as it can report whether the robot has re-visited a location.

3.8 Insect-inspired navigation and Visual Homing

Much of the mapping literature contains descriptions of motion models for mobile robots with odometry and sensor models for sonars and lasers [13, 27, 70, 97]. However, one of the goals of this thesis is to implement a mapping scheme for robots, which have very poor odometry and only a noisy black & white camera for an environmental sensor. Examples of similar systems can be found in biology, particularly in the navigation techniques of certain kinds of insects. For instance, in [67], a computational model of the *Cataglyphis bicolor* desert ant was implemented. This robot used panoramic visual snapshots of surrounding landmarks to orient itself with respect to those landmarks and was able to return to its home base.

The honeybee is capable of using visual cues from the position of the sun in the sky to landmarks along its path and complete dead reckoning to fly more than 10 kilometers from its nest in search of food [25]. Not only does it accomplish this with a brain the size of

a grass seed, but it is capable of communicating the distances it has traveled to its sister bees as part of the direction-to-honey dance. Experiments with honeybees suggest a robust method of landmark learning and innate optical flow calculation is employed to navigate between the hive and the food source and back again [15]. Models based on the behaviors of these animals can provide some insight into how to develop algorithms that will allow mobile robots to do the same sorts of navigation that insects do with such accuracy. As reported by by Cartwright & Collett [15], a honeybee is capable of recognizing the orientation and size of various landmarks and adjusting its flight path to match a "known path" with respect to these obstacles. From this, they suggest a model in which the honeybees are storing snapshots of the landmarks that they are observing, thus being able to return to them later. This research inspired Kick *et al.* [50] to simulate this approach on a mobile robot. Their explorations suggest that a dynamic model using optical flow calculations is probably more robust.

The idea of using optical flow for navigation has been explored, in Wolfart *et al.* [104], where a robotic system is developed in which a mobile camera tracks landmarks through motion to correct position and orientation errors. A similar idea has been applied by Perkins & Hayes [75] to calculate the distances to obstacles using optic flow and a single camera on a mobile robot.

3.9 Appearance-Based Mapping

In [102], images from an omnidirectional camera are used to construct a topological map of an environment. A color "signature" of the environment is calculated using histograms of the RGB and HSV (Hue, Saturation, and Value) components of the image. A nearest-neighbor learning algorithm is used to construct a map and various histogram matching algorithms are used to calculate how well a particular image matches a location. The individual nodes in the map are chosen manually by a human operator and the system can

return how confident it is of a match. Each node in the generated map represents a single room or corridor and is only capable of recognizing whether the robot is in a particular room rather than a more precise position in that room. This would be problematic for small robots whose operating environments would most likely consist of a single room and would need a position estimates that is finer in granularity.

Pinette [77] described how a robot equipped with an omnidirectional image sensor can navigate to landmarks based on an image-based homing technique. These homing techniques use bearing information and a heuristic for estimating the rough ranges to landmarks. This work also describes how maps can be built and paths followed using additional heuristics. In contrast, our methods deal explicitly with the uncertainty in the robot's position and strive to create an accurate metric representation of the path based on the images in the environment.

Chapter 4

Map Construction

4.1 Sensor and Motion Models

Constructing accurate and useful maps for mobile robots with accurate sensors in static indoor environments is a fairly well understood but nevertheless challenging task. Even with accurate sensors and motion control, without models that accurately model the noise in the system, a map will be inaccurate and generally unusable if the errors are allowed to grow without correction. The problem of representing spatial relationships in a robot's internal map boils down to the problem of deriving the model which properly reflects the granularity and the certainty of the sensors being used. For instance, if the robot has an extremely accurate sensor, like a laser scanner, it can represent the world as a two-dimensional grid with fine granularity. The robot can even do away with accurate odometry information and still be able to construct an accurate map, as reported in Thrun *et al.* [98]. However, if a lower-resolution proximity sensor is used, such as a sonar, the need for having better odometry increases if the robot is required to maintain the same accuracy of map. In fact, if the only sensor were perfect odometry and the robot could detect collisions with obstacles (without introducing any errors in the odometry), it would be able to construct a map

just by bumping into the objects in the environment. It is important to note that not all robots have laser scanners and certainly extremely few robots are capable of successful tactile mapping of an environment that is far larger than the size of the robot. All existing hardware generally falls in the middle of the two extremes.

The small size of the Scout robot gives it particular advantages over larger robotic hardware, specifically from the standpoint of portability, accessibility, and overall cost. However, one of the disadvantages of the Scouts is the inability to carry appropriate sensors and computational hardware necessary for localizing themselves within an environment. If the robots are to be used in any autonomous way, they must be able to sense their environment in an intelligent fashion and construct a spatial representation from those sensor measurements.

As the size and complexity of the robot's sensors and actuators decreases, most common indoor environments start to resemble outdoor kinds of environments. In this respect, all environments must be treated as unstructured and very few assumptions can be made about the terrain that the robot must navigate over and what kinds of obstacles and landmarks exist. Instead of limiting the complexity of the environment to make it easy for the robot to navigate within, the complexity of the robot's sensors and sensing modalities must instead be decreased to make it easier for the robot to interpret the large amount of received environmental.

Estimation of a robot's motion usually depends on modeling a specific sensor modality, such as inertial guidance systems or wheel encoders, that allows the robot to measure how far it has moved. Such sensors are only good for very short distances because it is notoriously difficult to keep these sorts of navigational aids from accumulating significant amounts of error [9]. Using odometry at least in the short run has the advantage that the robot can be fairly sure about its motion for distances traveled in the tens of meters. Beyond this, however, the errors will accumulate so as to make the distance measurements useless. This

problem is compounded if the robot rotates as well as translates. Due to wheel slippage, most robots will have error in the order of several degrees whenever they rotate. When this is added to the translational error, the robot's positional estimate can be extremely poor. Small robots will typically have worse odometry estimates over long distances as compared to larger robots. The mass and inertia of larger robots tends to keep them from being affected by uneven terrain and loss of wheel traction.

4.2 Topological Map Representation

Building maps based on grid representations with small robots is not likely to be successful for two reasons. First, the lack of readily available range sensors suggests that it will be very difficult, if not impossible, for the robot to calculate the freespace between itself and obstacles in the environment. Without this sort of information, the only environmental freespace that can be calculated is the area under the robot as it moves about. Secondly, in order to accurately position the robot, the resolution of the grid must be fairly fine. While the overall range of small robots could be limited to only a few tens of meters, in general, the computational and memory requirements of the grid-based representation may become very challenging. For either of these reasons, grid-based representations are probably not the appropriate choice. Instead, a map based on a topological representation of places and paths is more attractive.

As described earlier, topological maps are low-resolution relational graphs whose nodes represent individual places in the robot's environment and whose edges describe paths that the robot can take to get from one location to another. In general, topological maps are well suited for this kind of problem as they are capable of representing an arbitrary level of robotic positional uncertainty. As long as the places that are in the map are recognizable by the robot's sensors, a qualitative representation of the robot's environment can be maintained. In the topological map described by Ulrich and Nourbakhsh [102], each

node in the map represented a single room and the edges in the graph simply described whether one room was connected to another without actually specifying the exact path to navigate from one point to another. As stated in Thrun *et al.* [100], the ability to handle arbitrary precision makes this kind of map best suited for solving the *global alignment problem*, where locations in the environment are not represented by more than one node in the graph. However, one challenge of this approach occurs when the distinctive places are not unique. If the robot is uncertain as to which location it is at, the robot's position may be extremely uncertain and it may be unable properly localize itself. Kuipers defines a procedure called rehearsal in which robots disambiguate their position estimates in the SSH by actively moving around to test hypotheses [55]. This is more of a heuristic rather than a complete solution due to the fact that traversing to a single distinctive place may not be good enough to resolve the robot's location. In the grid-based paradigm, as reported in Fox *et al.* , [31], active Markov Localization is a method in which a robot actively seeks out areas that will assist in disambiguating its position.

While topological maps that do not rely on any odometric and Euclidean position estimates are useful for high-level qualitative reasoning, at some point, more geometric reasoning is necessary in order to navigate and explore the robot's surroundings. What is required is a method which captures the best features of both topological and grid-based map representations, at least to within the limits of the robot's sensors.

4.3 Appearance-Based Mapping

In the above discussion of the SSH, it was generally assumed that the robot had some sensor that would allow it to follow a gradient to a local distinctive place. However, in unknown environments, it is not always likely that the robot will have a sensor at its disposal that will allow this. At best, the robot will only be able to associate a sensor reading with a specific location in space. Thus, as the robot navigates, it will have no immediate way

to correlate how the sensor readings will change. Thus, the only way for the robot to estimate its position is to integrate a model of its motion, which, as stated previously, will degrade rapidly due to poor odometric measurements. In order to correct the poor odometric estimates, additional information is necessary. In environments where absolute location measurements from the global positioning system (GPS) are not available, a robot will have to use information about its surroundings for this purpose. In order to provide periodic corrections, additional information is necessary.

As stated before, in most mapping algorithms, it is assumed that the robot is able to measure its relative position with respect to features/landmarks [23, 71] or obstacles [97] in the area that it navigates. An alternative approach is to use a small camera and process the relative angular measurements to vertical line features, as described in [6]. However the applicability of this algorithm is conditioned on the existence of a sufficient number of identifiable vertical line segments along the trajectory of the robot. Also it is geared toward position tracking while no attempt is made to construct a map populated by these features.

In the ideal case, a sensor modality that neither relies on any specific type of features, nor requires distance measurements would be preferred. Using the robot's sensor, a unique visual signature for distinct locations along the robot's path can be obtained. These signatures are associated with the estimated pose of the robot at that time instant, and can be retrieved once the robot revisits the same area. Figure 4.1 illustrates an example spatial representation of the robot's environment.

Determining whether the robot is at a certain location for a second time is the key element for providing positioning updates. By correlating any two scenes, we can infer a relative position measurement and use it to update both the current and previous pose estimates (at locations visited in the past) for the robot. This in effect will produce an accurate map of distinct locations within the area that the robot has explored. In effect, the landmarks that the robot detects explicitly represent the specific locations that the

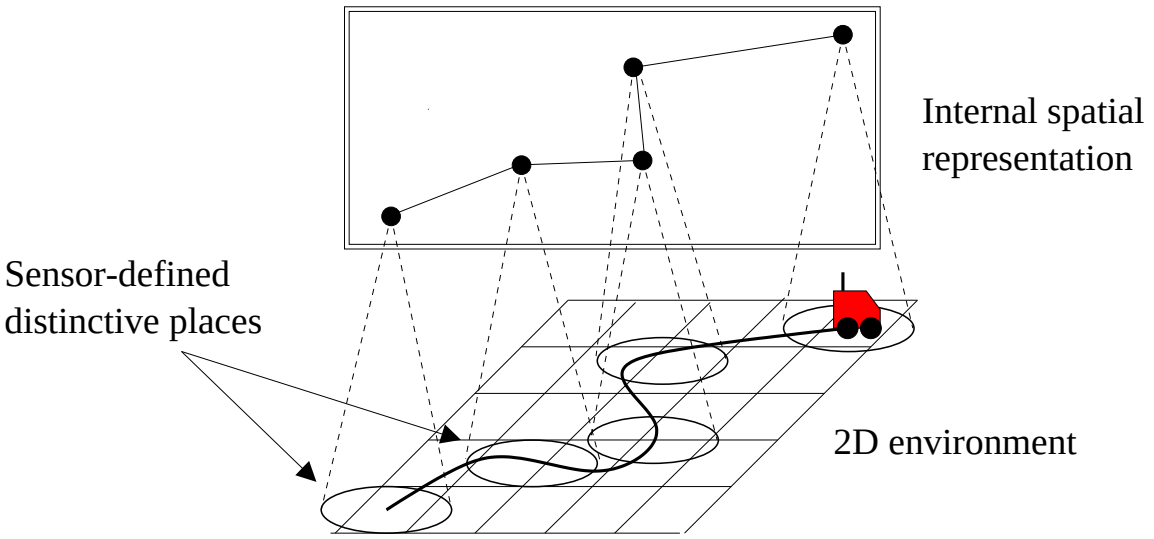


Figure 4.1: Appearance-based mapping

robot has visited.

4.3.1 Cycles as Constraints

Nearly all environments contain some sort of cycle. This can be manifested as a structured loop, such as a corridor of a building, or more problematically, as a completely arbitrary shape which might be found in an unstructured environment. In these situations, a robot exploring its surroundings will have to determine whether an area that it is currently examining is new or whether it has just returned to a previously-explored area. If the sensor information from all areas is unique, then this problem is simplified. However, this is rarely the case, as many environments have areas which look extremely similar.

Since the landmarks in the environment are simply representative of the locations to which the robot has traveled, rather than concrete objects in space, there is no easy way to infer relative distances between them other than through the robot's odometry. If the robot never crosses its own path, then the absolute error in position estimate will grow without bounds. However, if the robot returns to a location that it has already visited, *and*

is able to recognize it as such, then it can place those nodes on top of one another and force them to occupy the same space in the map representation. This will greatly constrain the possible relative positions of the rest of the nodes in the map, and the newly constrained topology will start to resemble the true path. Figure 4.2 illustrates this process.

4.4 Assumptions about the Robot's Capabilities

From the above discussion, we can make the following assumptions about the capabilities of the robot in terms of its sensing and actuation modes as well as the sorts of spatial representations it is capable of generating. These assumptions will direct the formulation of the various mapping/path reconstruction techniques that are described in this research.

- Landmarks are represented by sensory signatures at the location where the reading is taken. Thus, landmark positions and robot positions are identical. This is the lowest level of assumed robot sensory capability. If other sensor information is available, then it can be used to augment the robot's world model.

- Sensor readings are assumed to be orientation independent. Some examples of such sensors include panoramic imaging systems which are capable of obtaining a signature from a full 360°around the robot. Another example of an orientation-independent sensor modality might be an RF signal strength monitor. Such sensors help to reduce the problem of perceptual aliasing caused by the lack of rotational sensory coverage.

- The robot makes relatively small motions between its sensor readings. If the motions are too large, there is a definite chance that the robot will not be aware that it has crossed its own path. The size of each of the motions must be directly related to the region of uncertainty associated with each sensor measurement.

- The robot moves in a two-dimensional flat world typical of most indoor environments.

We also assume that the world is fairly static, that is, there are not a lot of moving objects that could interfere with the detection of landmarks around the robot.

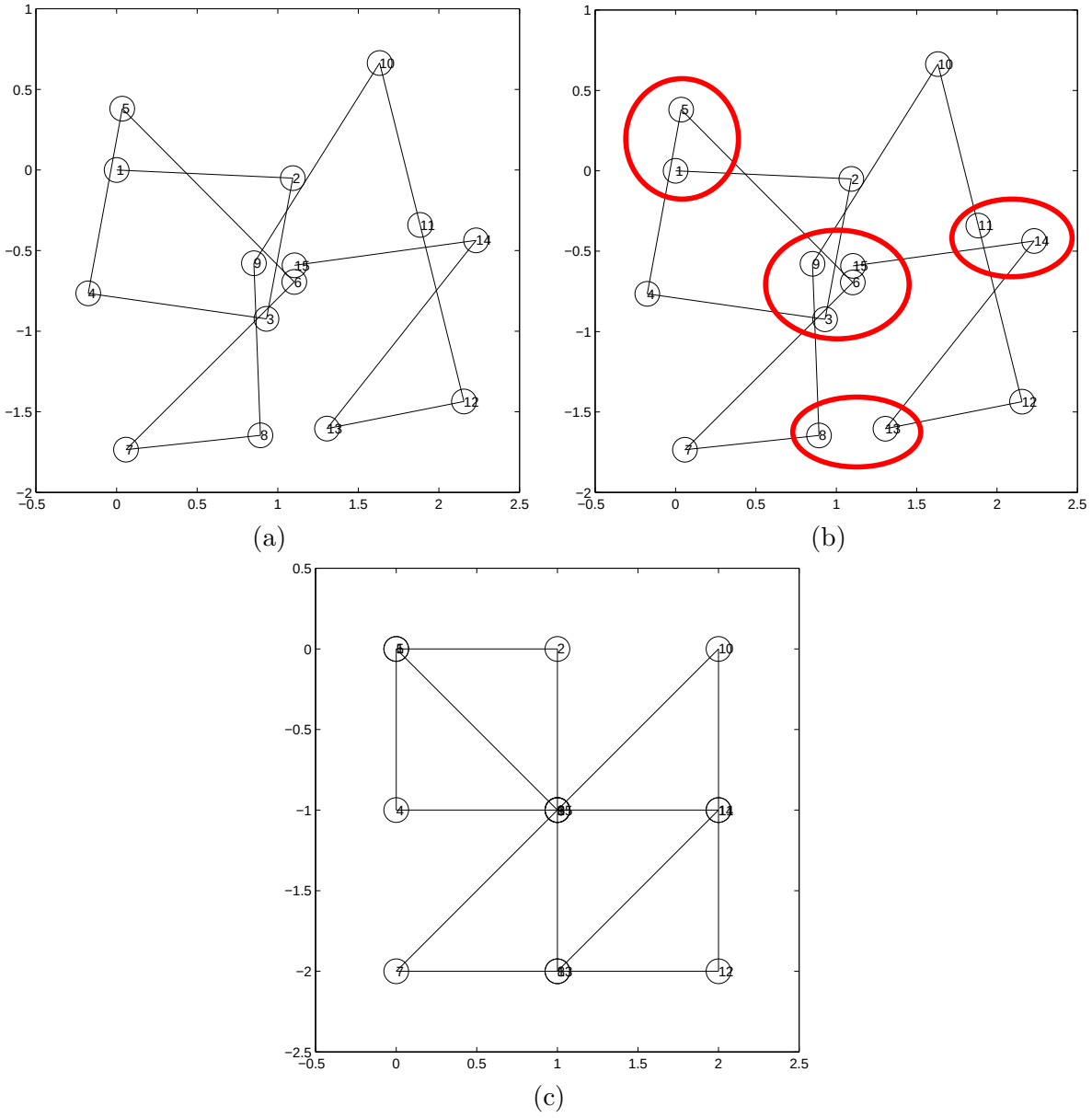


Figure 4.2: Resolving the robot's path. In (a), the robot navigates in an environment and takes sensor readings at each location. In (b), overlaps in the estimated path are identified. Finally, in (c), the topology of the estimated path is constrained.

Chapter 5

Maximum Likelihood Estimator

5.1 Spring and Mass-Inspired Estimator

We are interested in constructing a spatial representation from a set of observations that is topologically consistent with positions in the environment where those observations were made. This means that there exists only one location in the robot's map for each individual location that it has traveled to, regardless of how many times the robot visited that location and took a sensor reading.

More formally, let S be the set of all sensor readings that are obtained by the robot. Each $s_i^t \in S$ represents a single sensor reading taken at a particular location i and at time t. Let D be the set of all unique locations the robot visited. If the robot never traveled to the same location twice, then $|D| = |S|$ (the cardinality of the sets is the same). However, if the robot visits a particular location d_j more than once, then $|D| < |S|$ because multiple sensor readings ($s_i^{t_j}$, $s_i^{t_{j+1}}$, ... ,$s_i^{t_{j+n}}$) were taken at that location. The problem then is to determine from the sensor readings and the sense of self-motion which locations in D are the same. Once identified, these locations are merged in order to create a more accurate map.

The challenge is to determine the most likely map given the sensor readings. The uncertainty in the sensor and odometry measurements can be treated as a cost function to be minimized. This suggests that the problem can be formulated in terms of a maximum likelihood estimation (MLE) problem where the most likely map M is found based on the sensor readings S. This can be described as trying to find the correct parameters that will maximize a probability distribution:

$$P(M|S) = \prod_{s \in S} P(M|s) \tag{5.1.1}$$

However, determining the correct form of this probability distribution can be challenging, and so an interesting simplification is to convert the problem from maximizing a probability distribution into minimizing an energy distribution. By taking the $-log$ of the probability distribution, we have the expression:

$$-log(P(M|S)) = \prod_{s \in S} -log(P(M|s)) = \sum_{s \in S} U(M|s) \tag{5.1.2}$$

If we assume that the probability distribution $P(M|s)$ is Gaussian, then taking the $-log$ of it will return an expression of the form:

$$\frac{1}{2}(s-\mu)^T \sigma (s-\mu) \tag{5.1.3}$$

which in the single-dimensional case has the same form as the potential energy equation of a spring:

$$\frac{k}{2}(e-\hat{e})^2 \tag{5.1.4}$$

In this formulation, e is the current elongation of the spring, $\hat{e}$ is the relaxation length of the spring and k is the spring constant. In order to minimize the system, direct numer-

ical simulation based on the equations of motion (derived by taking the derivative of the equations of potential energy with respect to position) can be employed. An advantage of using MLE to find the best possible map is that computation and storage of the entire distribution is not required. Only the best estimate is maintained at each iteration of the algorithm. This allows for much faster computational updates and much lower memory requirements.

Figure 5.1 shows a simple example of how the linear and torsional springs are used to represent the difference between the current model and the robot's sensor measurements. The values T_i represent the stretch of the torsional springs which signify the difference of the model from the robot's rotational measurements. The values L_i are the stretches of the linear springs which represent the displacement in the model from the robot's translational measurements.

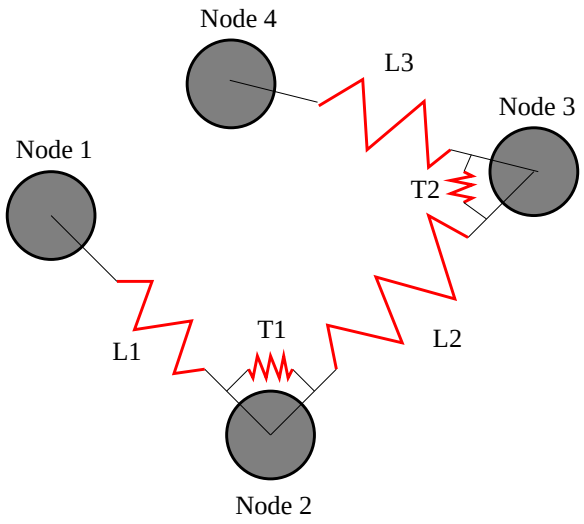


Figure 5.1: Examples of relative poses (Node1-Node4) of the robot connected by linear (L_1-L_3) and torsional (T_1-T_2) springs.

A spring model is a useful way of describing the path of the robot. As the robot moves, it keeps track of its rotational and translational displacements. It is assumed that the robot

moves in a simplified "radial" (or "vector" [38]) manner where rotation is decoupled from translation, i.e. robot motions consist first of a rotation in place followed by a straight-line translation.

Following each motion, a reading from the robot's sensors is obtained. This sequence of motions and sensor observations can be observed as graph where each node has at most two edges attached to it, forming a single chain. In this model, translational displacements in the robot's position can be represented as linear springs and rotational displacements can be represented as torsional springs. The uncertainty in the robot's positional measurements can be represented as the spring constants. For example, if the robot were equipped with high precision odometry sensors, the stiffness in the springs would be very high.

By representing the locations where sensor readings are taken as masses and the sensor-measured distances and angles between those masses as springs, a formulation for how well the model corresponds to the data can be expressed as the potential energy of the system. Andrew Howard *et al.* [45] illustrated the utility of this approach by localizing a group of mobile robots equipped with laser range finders.

Figure 5.2 illustrates how this process would work. In this sequence, a simulated Scout has been teleoperated around a small 4 m x 4 m square and has returned to the same place. Due to the odometry error, the path the robot took is not square. Each circle in the picture represents a position where the motion of the robot was recorded. When the robot returned to its original position, saw that it has returned (due to the matching of landmarks seen there), and the last spring closes the cycle, the potential energy of the stretched springs caused the model to take on a shape which was more indicative of the proper topology of the world.

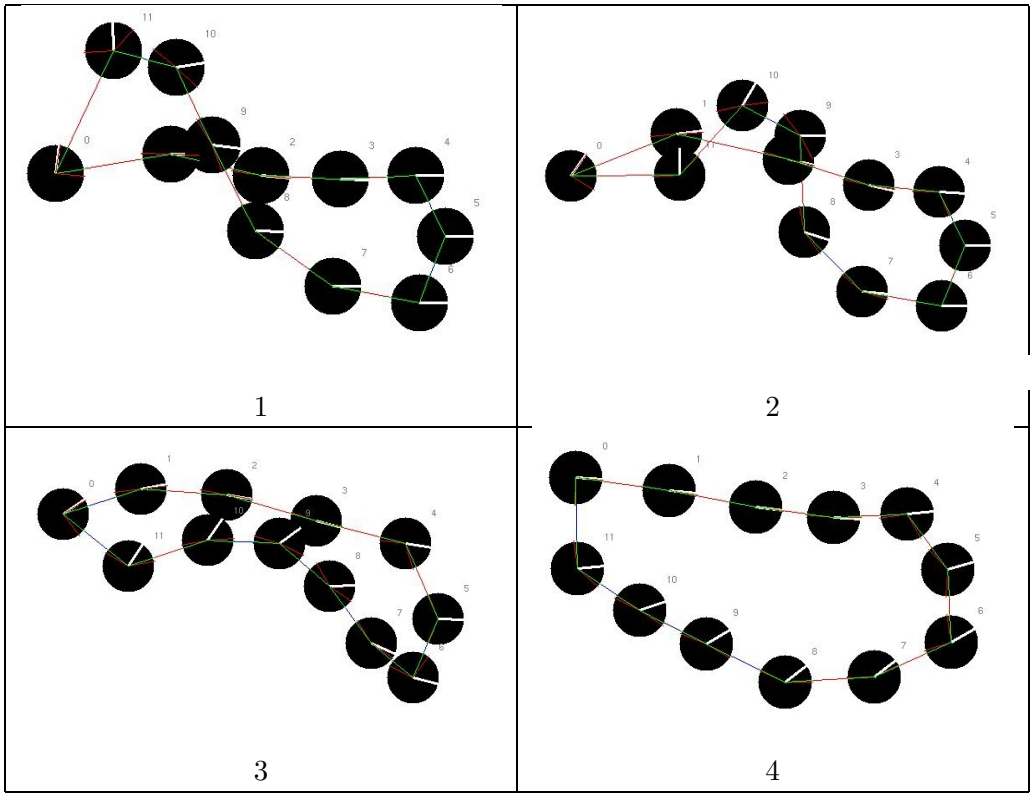


Figure 5.2: This sequence illustrates the process by which a model of the Scout's recorded odometry converges to a shape which best describes the proper topology of the environment. The black circles represent the positions of the Scout as they are recorded by its sensors. The model starts on the top left with raw odometry and iterates through to the bottom right until a rectangular path is produced.

5.1.1 Derivation of Mass/Spring System Dynamics

Linear springs connect masses and apply forces that try to keep the nodes at a constant distance. These represent the measured linear distances between locations where sensor readings were taken. Torsional springs reside on a node j and apply forces that try to maintain a constant angle between the previous node i and the next node k. These represent the measured rotational displacements between nodes as measured by the robot as it traverses the environment.

Table 5.1 lists the terms used to describe characteristics of the masses in the simulation.

p_{x_i}	position of node i along the x axis
p_{y_i}	position of node i along the y axis
v_{x_i}	velocity of node i along the x axis
v_{y_i}	velocity of node i along the y axis
a_{x_i}	acceleration of node i along the x axis
a_{y_i}	acceleration of node i along the y axis
F_{x_i}	force on node i along the x axis
F_{y_i}	force on node i along the y axis

Table 5.1: Terms describing the masses in the derivation of the spring-based maximum likelihood estimator.

5.1.1.1 Linear Springs

Linear springs are defined by their resting length $L_{\hat{e}_{ij}}$, stiffness constant $C_{s_{ij}}$, and damping constant $C_{b_{ij}}$. When the spring's length changes because of a distortion of the surrounding springs, a force is applied to each of the masses that it is connected to. This force is proportional to the change in length (the stretch) $L_{d_{ij}} = L_{e_{ij}} - L_{\hat{e}_{ij}}$ of the spring. Table 5.2 summarizes the terms used to describe the characteristics of the linear springs.

$L_{\hat{e}_{ij}}$	rest length of linear spring connecting nodes i and j
$L_{e_{ij}}$	measured length of linear spring connecting nodes i and j
$L_{d_{ij}}$	stretch ($L_{e_{ij}} - L_{\hat{e}_{ij}}$) of linear spring between nodes i and j
$C_{s_{ij}}$	spring constant of linear spring between nodes i and j
$C_{b_{ij}}$	damping constant of linear spring between nodes i and j

Table 5.2: Terms describing the linear springs in the derivation of the spring-based maximum likelihood estimator.

Opposing this is a damping force that is proportional to the velocity of the mass. The equations are divided through by the mass of the nodes in order to derive expressions for acceleration. The final expressions for the forces (in X and Y) acting on each mass are:

$$a_{x_i} = \frac{C_{s_{ij}} L_{d_{ij}}}{m_i} \sin(\alpha) - \frac{C_{b_{ij}} v_{x_i}}{m_i} = \frac{C_{s_{ij}} L_{d_{ij}}}{m_i} \frac{(p_{x_i} - p_{x_j})}{L_{d_{ij}}} - \frac{C_{b_{ij}} v_{x_i}}{m_i} \tag{5.1.5}$$

$$a_{y_i} = \frac{C_{s_{ij}} L_{d_{ij}}}{m_i} \cos(\alpha) - \frac{C_{b_{ij}} v_{y_i}}{m_i} = \frac{C_{s_{ij}} L_{d_{ij}}}{m_i} \frac{(p_{y_i} - p_{y_j})}{L_{d_{ij}}} - \frac{C_{b_{ij}} v_{y_i}}{m_i} \tag{5.1.6}$$

$$a_{x_j} = -\frac{C_{s_{ij}} L_{d_{ij}}}{m_j} \sin(\alpha) - \frac{C_{b_{ij}} v_{x_j}}{m_j} = -\frac{C_{s_{ij}} L_{d_{ij}}}{m_j} \frac{(p_{x_i} - p_{x_j})}{L_{d_{ij}}} - \frac{C_{b_{ij}} v_{x_j}}{m_j} \tag{5.1.7}$$

$$a_{y_j} = -\frac{C_{s_{ij}} L_{d_{ij}}}{m_j} \cos(\alpha) - \frac{C_{b_{ij}} v_{y_j}}{m_j} = -\frac{C_{s_{ij}} L_{d_{ij}}}{m_j} \frac{(p_{y_i} - p_{y_j})}{L_{d_{ij}}} - \frac{C_{b_{ij}} v_{y_j}}{m_j} \tag{5.1.8}$$

where the angle α, is the angle the spring is from the global Y axis, as shown in Figure 5.3. The mass terms, m_i and m_j, do not have any particular meaning in the framework of this localization context and so are set to 1 for all nodes.

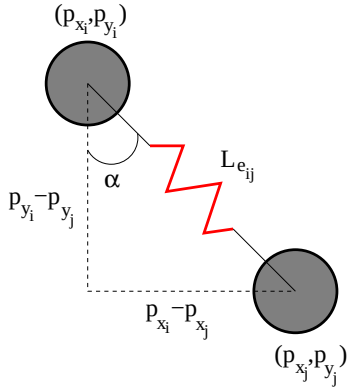


Figure 5.3: Linear spring model where the position of the masses are shown as (p_{x_i}, p_{y_i}) and (p_{x_j}, p_{y_j}), the measured length of the linear spring is $L_{e_{ij}}$, and the angle of the spring from the global Y axis is α.

5.1.1.2 Torsional Springs

Torsional springs represent the angle that the robot turned on its way from node i through node j and onto node k. They are defined by their resting angle $T_{\hat{\phi}_{ijk}}$, and stiffness constant $C_{s_{ijk}}$. The stretch of a torsional spring is the difference between its resting and measured

angle, or $T_{\delta_{ijk}} = T_{\phi_{ijk}} - T_{\hat{\phi}_{ijk}}$. In this model, the torsional spring cannot be "wound up" as it always pulls in the direction of the shortest distance to the resting angle, i.e. the values of $T_{\phi_{ijk}}$ and $T_{\hat{\phi}_{ijk}}$ are always clamped to the range $[-\pi, \pi]$.

$T_{\hat{\phi}_{ijk}}$	rest angle of torsional spring between nodes i and k (residing at node j)
$T_{\phi_{ijk}}$	measured angle of torsional spring between nodes i and k (residing at node j)
$T_{\delta_{ijk}}$	stretch ($T_{\phi_{ijk}} - T_{\hat{\phi}_{ijk}}$) of torsional spring between nodes i and k (residing at node j)
$C_{s_{ijk}}$	spring constant of torsional spring between nodes i and k (residing at node j)
$C_{b_{ijk}}$	damping constant of torsional spring between nodes i and k (residing at node j)

Table 5.3: Terms describing the torsional springs in the derivation of the spring-based maximum likelihood estimator.

Table 5.3 summarizes the terms used to describe the characteristics of the torsional springs. Compressing or extending a torsional spring generates a torque that will try to rotate the spring back to its resting position. This torque translates into forces that are applied to the masses whose linear springs are connected by the torsional spring. The amount of force applied is defined by the expression for torque τ:

$$\tau = -C_{s_{ijk}} T_{\delta_{ijk}} = F_k L_{e_{jk}} = F_i L_{e_{ij}} \tag{5.1.9}$$

where F_i is the instantaneous perpendicular force applied to node i at the end of a moment arm of length $L_{e_{ij}}$, as illustrated in Figure 5.4. Because the linear and torsional springs are specified as being independent of each other, the lengths of the moment arms are ignored when applying the forces to the masses. Thus, the equation for the calculated forces is as follows:

$$-C_{s_{ijk}} T_{\delta_{ijk}} = F_k = F_i \tag{5.1.10}$$

Once computed, all of the linear accelerations are fed into a 4th order Runge-Kutta [78] numerical simulator to determine how their positions change with time. Several tricks are

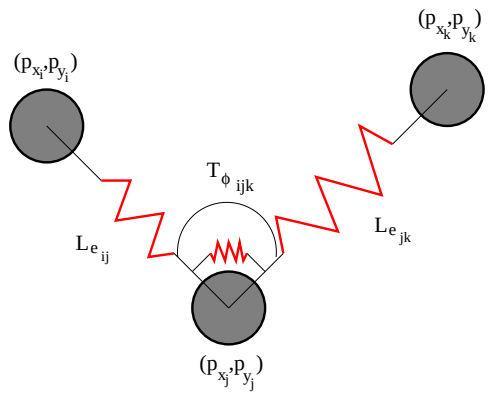


Figure 5.4: Torsional spring models

used to help ensure numerical stability, including using variable-sized timesteps, as well as a global damping term that applies to all nodes equally.

5.1.2 Analysis of the Mass/Spring System

Linear springs represent translational displacements, while torsional springs represent rotational displacements. Because they encapsulate completely different quantities (force and torque) and are assumed to be independent of one another, some care must be taken to ensure that the respective linear and torsional spring constants do not overshadow each other. Figures 5.5, 5.6, and 5.7 illustrate some example potential energy landscapes and that with the selection of proper constants, the linear and torsional components can assert the same level of influence on the energy function.

5.1.2.1 Linear vs. Torsional Springs

Since the linear and torsional springs are separate, their error measurements must be considered individually. The importance of the two kinds of springs should also be considered separately. A set of experiments was performed to analyze the relative importance of the linear and torsional spring strengths. A set of simple three-node paths were generated such

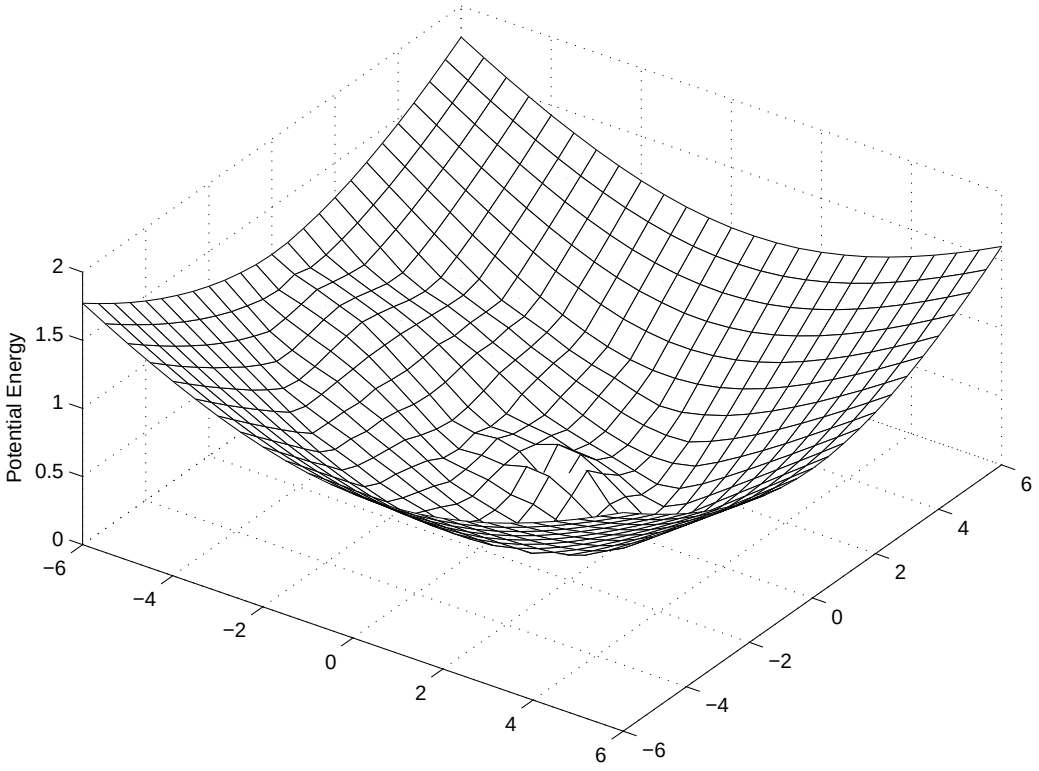


Figure 5.5: Plot showing the linear potential energy contribution. In this plot, a single chain of nodes, anchored at $(0, 0)$, is connected by a set of linear and torsional springs. The end node is stretched to various (x, y) positions, the model is allowed to relax, and the resultant potential energies are computed.

that the robot returned to the starting point after tracing out a regular polygon. The linear and rotational odometry estimates were corrupted by Gaussian random noise with variance ranging from 0 to 1.0. The constants for the linear and torsional springs were set to be the inverse of the noise. Thus, in these experiments, the assumption was made that the robots had a good estimate for the amount of error in both cases.

Figure 5.8 illustrates the process with two different variances. In this figure, the initial true path of the robot is described as a regular polygon where the first and last node close the polygon. The odometric estimates are corrupted by Gaussian noise. The first and last nodes are attached (merged) and the whole spring model is allowed to relax. Finally,

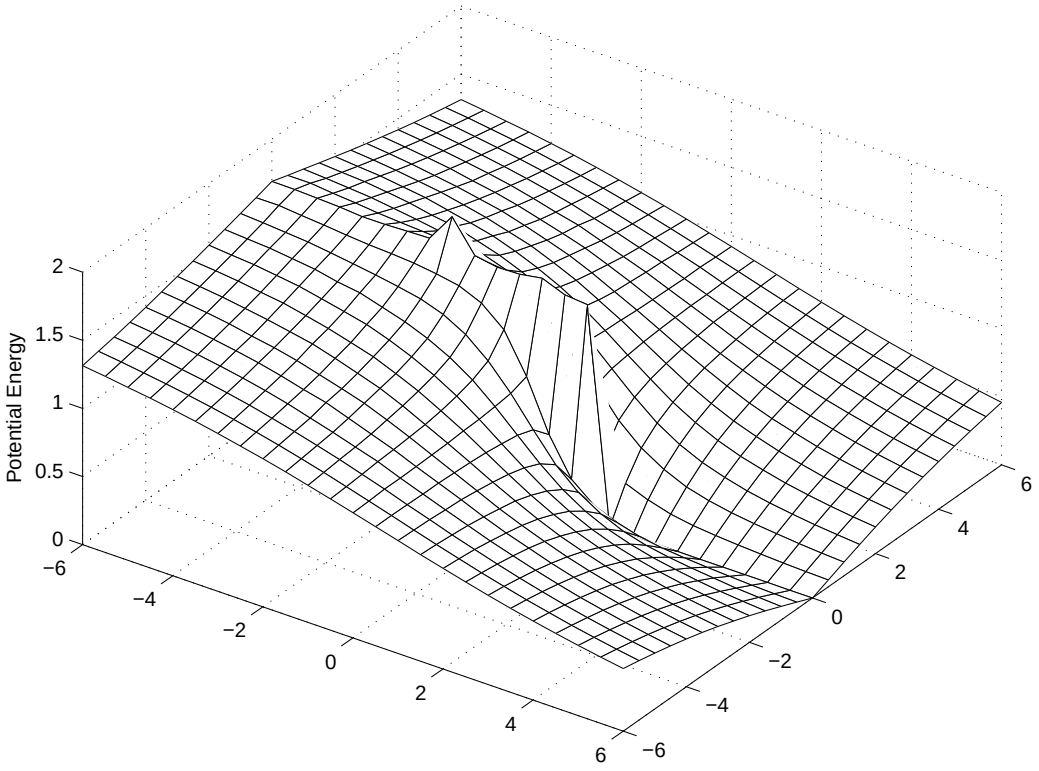


Figure 5.6: Plot showing the torsional potential energy contribution. In this plot, a single chain of nodes, anchored at $(0, 0)$, is connected by a set of linear and torsional springs. The end node is stretched to various (x, y) positions, the model is allowed to relax, and the resultant potential energies are computed.

a transformation is found which minimizes the total distance between the corresponding points in each dataset. This removes errors based on global misalignments and only illustrates the relative errors in displacement between the points in space. As can be seen, the distortion of 0.7 variance Gaussian noise in both linear and torsional springs produces a relaxed path that is very different from the true path and thus would have a very low sum of squared difference match.

The results for the three-node experiment can be seen in Figure 5.9. A similar experiment was run for four- and five-node paths. The resulting curves are extremely similar to the shown three-node path. The results indicate that the torsional spring constant is far

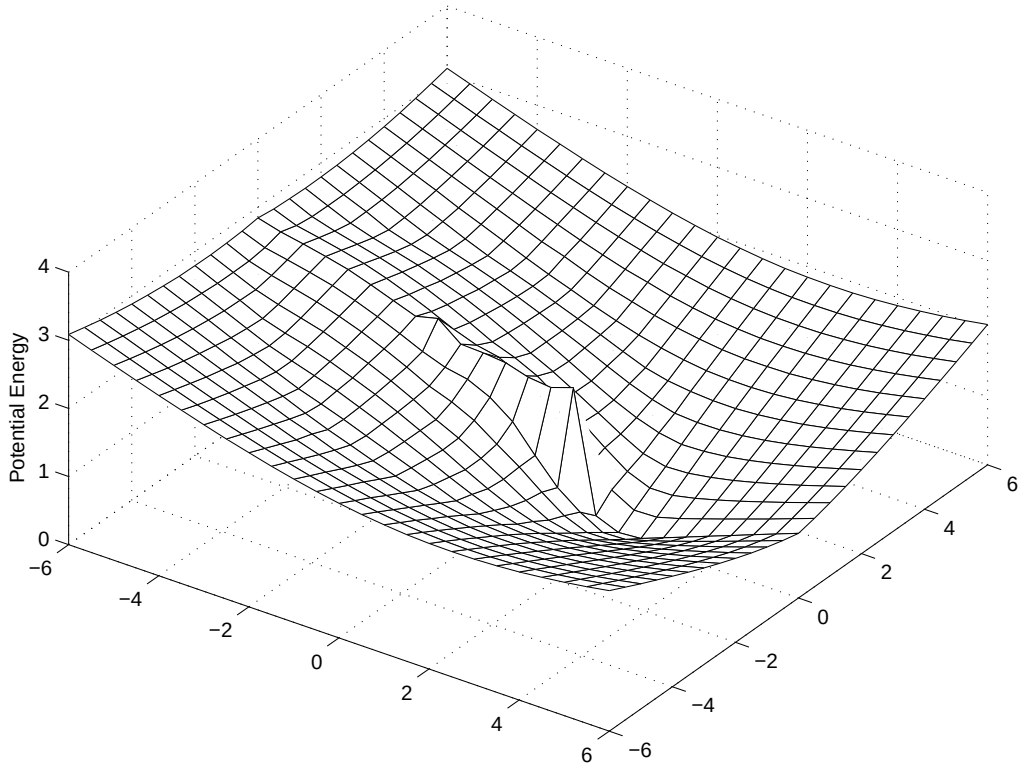


Figure 5.7: Plot showing the integrated (summed) linear and torsional potential energy contributions. In this plot, a single chain of nodes, anchored at $(0, 0)$, is connected by a set of linear and torsional springs. The end node is stretched to various (x, y) positions, the model is allowed to relax, and the resultant potential energies are computed.

more important than the linear spring constant. As long as the torsional spring constant is strong (and thus has a correspondingly low error estimate), the linear spring constant can be very weak (with a correspondingly high error estimate), and the final model will still converge to a shape that is very similar to the original path.

5.1.2.2 Torsional Constants vs. Error

The relative strengths of the spring constants must reflect the certainty of the robot's sensor measurements. The more certain the robot is of its sensor readings, the stronger the spring constants should be. This adds rigidity to the structure of the map and limits the possible

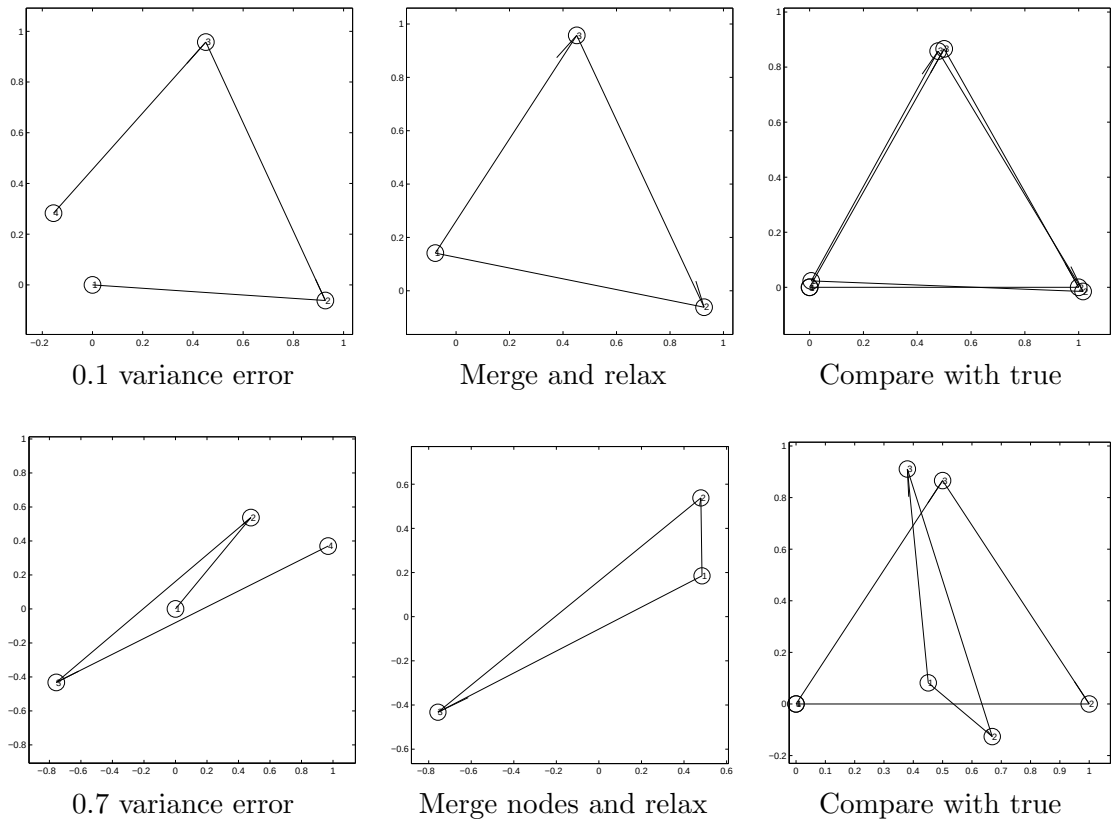


Figure 5.8: Linear vs torsional constant comparison experiment. Two sample variances, 0.1 and 0.7, are shown.

distortions and displacements that could occur.

If the torsional error estimates are very high, then it does not matter how strong the spring constants are. Very large rotational errors introduce too much distortion into the map to be corrected by correspondingly strong spring constants. Thus, it is vital that the robot's rotational estimate errors be low.

Figure 5.10 illustrates the results from this experiment. As can be seen, a good error estimate for the torsional results is absolutely critical. The error estimate completely dominates the accuracy of the final relaxed model, regardless of the strength of the spring.

An interesting conclusion from these experiments is that linear odometry estimates are

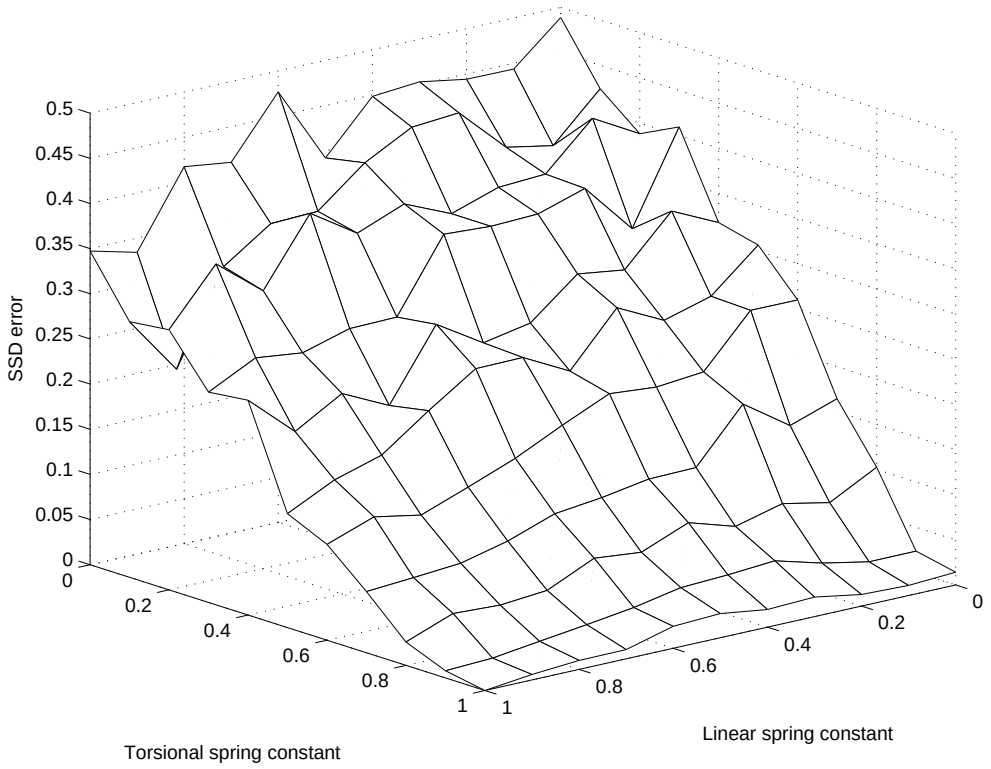


Figure 5.9: Results for the three-node linear vs torsional spring constant experiment.

not nearly as important as rotational odometry estimates. Unfortunately, this is where the majority of the errors in robot odometry propagation estimates occur. Methods for augmenting the robot's odometric estimates such as with visual odometry tracking or with a compass, such as in [24], would thus greatly assist in estimating the robot's position.

5.1.3 Simulation Experiment

This method was tested first on a simulated Scout robot. The standard deviation of the estimated wheel encoder error was 1.4 cm/s. The true path of the simulated robot is shown in Figure 5.11(a) as a square that is traversed twice. Sensor snapshots are taken roughly every 0.5 m as the robot traverses the path. The first time around the loop, the robot is essentially in an exploration mode. Each landmark that it observes is unique and thus it

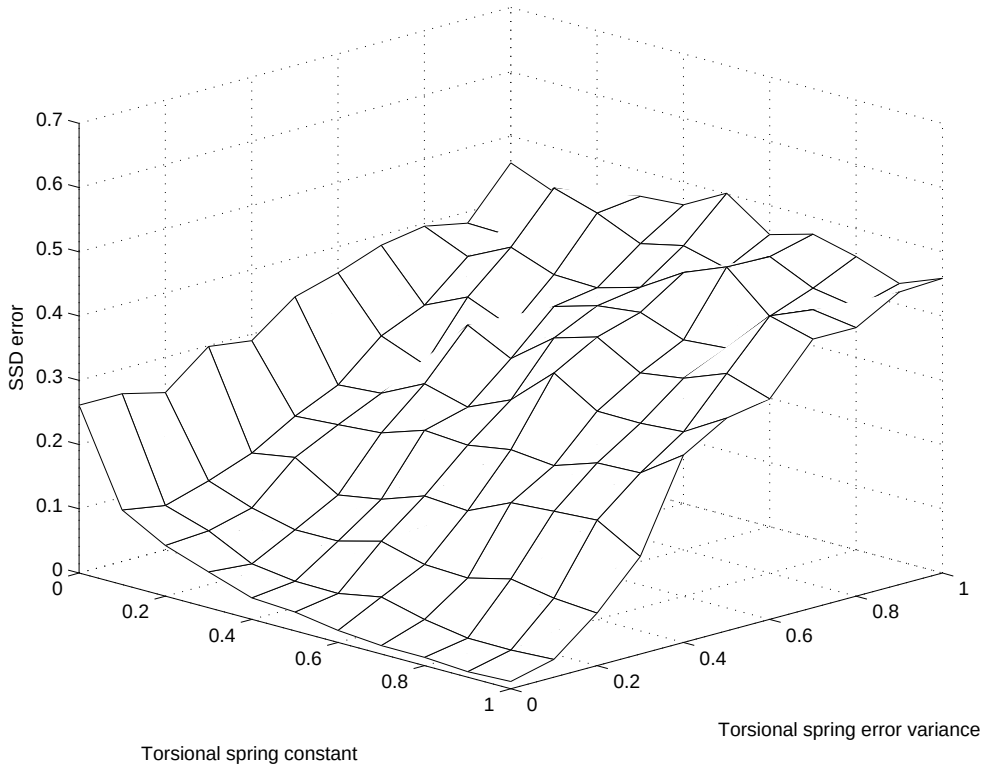
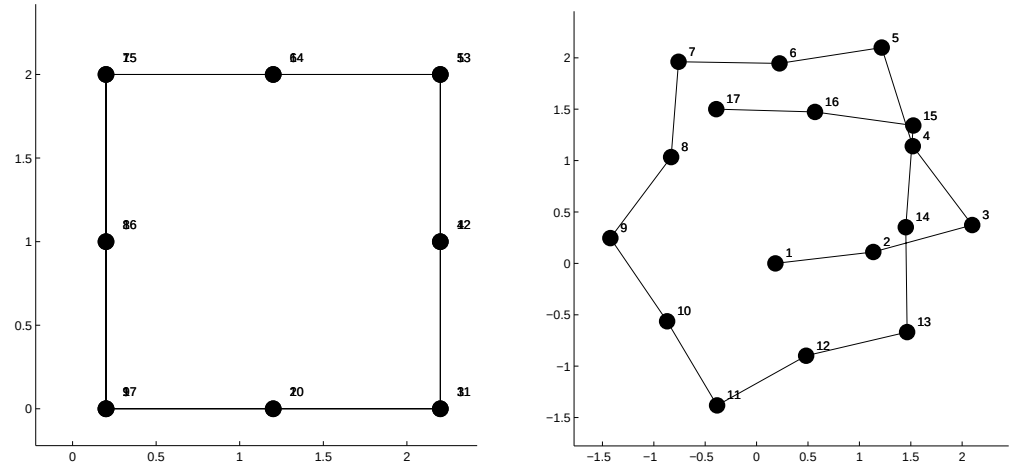


Figure 5.10: Results for the three-node torsional spring constant vs torsional error experiment.

adds the estimated positions of those landmarks directly to the state vector since the robot has no other information on those landmarks. The second time around the loop, the robot re-discovers the landmarks that it saw on the first time around. Without correcting for odometric errors, the robot's path estimate is shown in Figure 5.11(b).

The map convergence process occurs after all of the data has been collected. This process consists of identifying nodes to merge, connecting them together (thus distorting the spring estimates) and then relaxing the model until it converges. Figures 5.12 and 5.13 show how this process works with the simulated data.

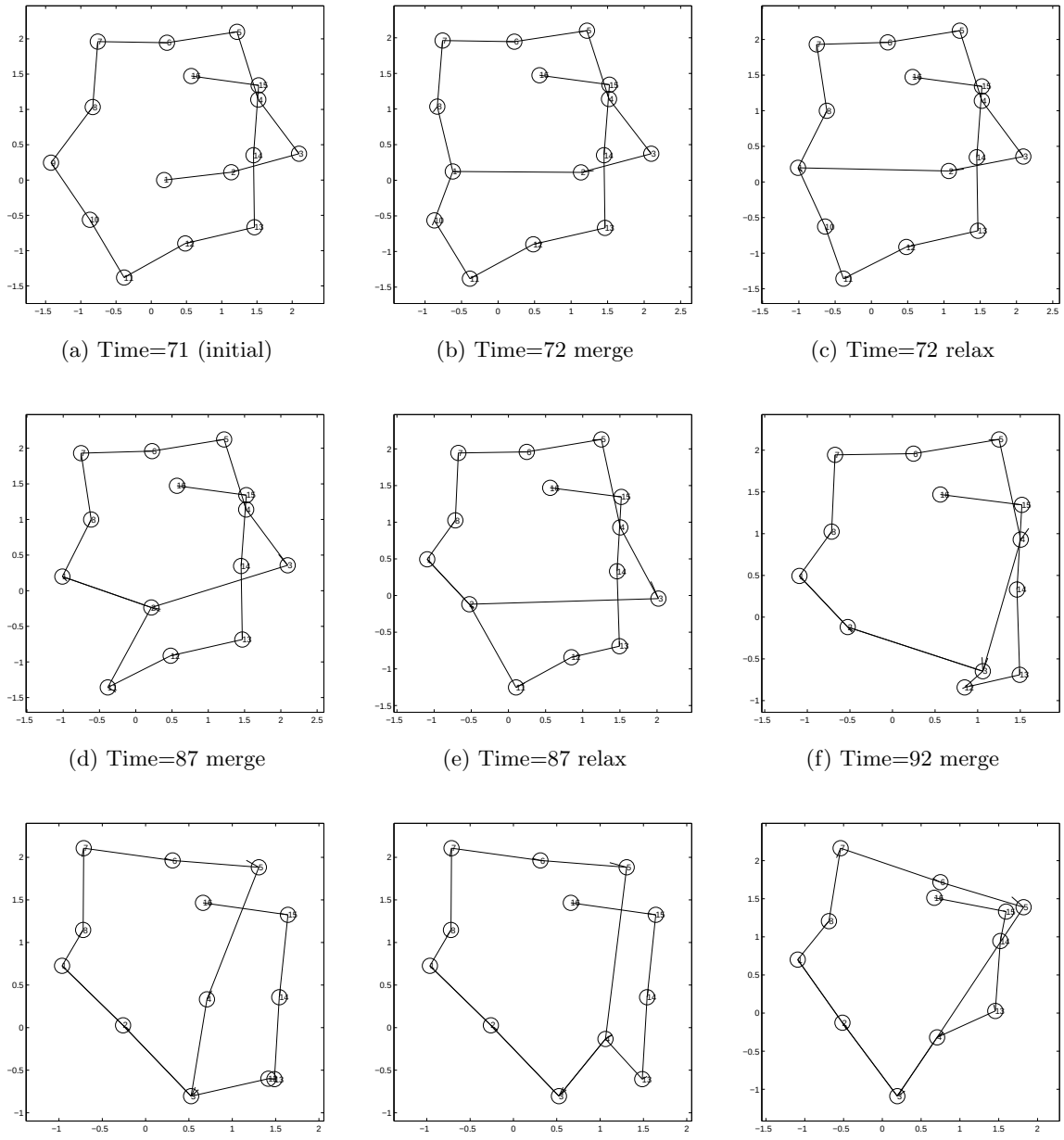
These figures show the process as an iterative process where each node is merged one at a time. The nodes could also be merged all at once, assuming that the robot could



(a) True path (b) Pose estimate

Figure 5.11: True and estimated path for the simulation experiments. The path starts from the lower left, moves counter-clockwise, and is traversed twice. Sensor readings are taken at the corners of the square and at the midpoints of each path leg. The scale is in meters.

afford to wait until it had all of the path information up front. Typically, with nonlinear systems, updating in batch mode tends to produce better estimates than sequential updates. Evidence for this will be seen later when the various estimators are compared to one another.

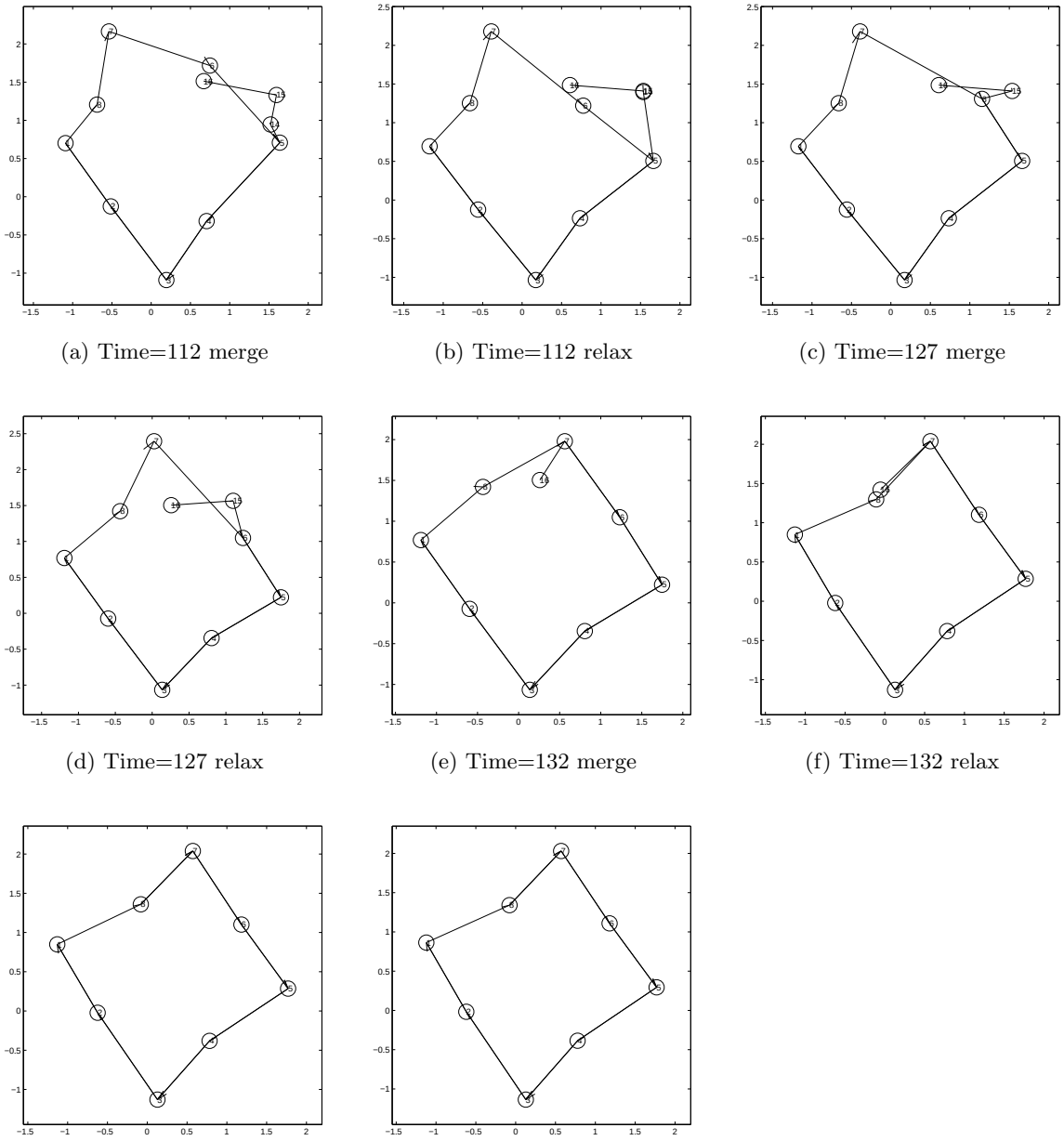


(a) Time=71 (initial) (b) Time=72 merge (c) Time=72 relax

(d) Time=87 merge (e) Time=87 relax (f) Time=92 merge

(g) Time=92 relax (h) Time=107 merge (i) Time=107 relax

Figure 5.12: Merging of locations where sensor readings were taken and the relaxation of the spring equations to obtain the most likely map. This shows the state of the landmark estimates after each pair of nodes were merged and then after the entire system was allowed to relax. Continued in Figure 5.13.



(a) Time=112 merge (b) Time=112 relax (c) Time=127 merge

(d) Time=127 relax (e) Time=132 merge (f) Time=132 relax

(g) Time=147 merge (h) Time=147 relax

Figure 5.13: Merging of locations where sensor readings were taken and the relaxation of the spring equations to obtain the most likely map. This shows the state of the landmark estimates after each pair of nodes were merged and then after the entire system was allowed to relax. Continuation from Figure 5.12.

5.2 MLE with Sensor Cost Functions

In the previous section, a physics-inspired maximum likelihood estimator was derived which used numerical simulation to find the most likely map given a highly non-linear system. However, this estimator made several significant assumptions about the problem. First, it was assumed that when the robot crossed its own path, the nodes would occupy the same physical location in space. Secondly, it was assumed that the linear and torsional motion estimates could be treated separately. While these assumptions might not reduce the quality of the estimates such that they are unusable, they will produce an estimate that is not as accurate as it could be. In order to incorporate this information, the estimator's cost function needs to be formulated in a different way.

The robot's proprioceptive estimate of how far it traveled between one location and another is the same as in the previous estimator. This is computed by integrating a series of readings from the robot's odometric sensors (wheel encoders) to determine the relative displacement (x, y, θ). The cost function for this estimate is called the motion cost function and is described as:

$$(y_i - h_{y_i}(X))^T P_i^{-1} (y_i - h_{y_i}(X)) \tag{5.2.1}$$

where y_i is a vector that describes the measured displacement between the previous position measured at time $i-1$ and the current position measured at time i. The function $h_{y_i}(X)$ computes the predicted displacement of the robot given the current state vector from time $i-1$ to time i. The covariance of this measurement is P_i.

As described in the previous section, the state vector of a maximum likelihood estimator consists of all of the necessary parameters to solve for. For the case of a mobile robot moving on a 2D surface, the variables represent individual locations to which the robot has traveled. In the previous section, this was the set of sensor readings S. It was also assumed that the

robot only traveled to D distinct locations and that $|D| < |S|$. This assumption is not quite true since while the robot may have traveled near the same location several times, those exact positions of the robot were not completely identical. That means that simply merging the nodes as was done previously will not provide the most accurate estimate. Thus, S and D will have the same number of elements and the cost function associated with the sensor readings is:

$$(z_i - h_{z_i}(X))^T R_i^{-1} (z_i - h_{z_i}(X)) \tag{5.2.2}$$

In the sensor cost function, z_i is a vector that describes the measured displacement between a position measured previously at time j (not limited to time $i-1$) and the current position at time i. Using the notion of the appearance-based sensor, the value of z_i will always be 0 since the landmarks correspond directly to the positions of the robot. The function $h_{z_i}(X)$ computes the predicted displacement of the robot given the current state vector from the previously-seen location at time j to the current time i. The covariance of this measurement is R_i.

As the robot discovers new landmarks, it adds their positions to the state vector and marks those variables as the locations of the original sightings. When the robot re-discovers a landmark it also adds this position to its state vector, but flags it as previously-seen. The sensor cost function in Equation. 5.2.2 always compares the current measured position of a landmark against the first discovered position of that landmark.

Combining the motion and sensor cost functions (Equations. 5.2.1 and 5.2.2), the complete cost function is:

$$\sum_i (y_i - h_{y_i}(X))^T P_i^{-1} (y_i - h_{y_i}(X)) + \sum_j (z_j - h_{z_j}(X))^T R_j^{-1} (z_j - h_{z_j}(X)) \tag{5.2.3}$$

The number of motion cost function terms is the number of sensor readings minus one $|S|-1$. The number of sensor cost function terms corresponds to the number of non-unique landmarks the robot has identified.

5.2.1 Linearized Estimator

The non-linear nature of this problem, introduced by the need to handle the rotational component of the robot, means that finding the best solution can be analytically and computationally challenging. In the description of the spring and mass-based estimator mentioned earlier in this chapter, a numerical solution was found for finding the minimum value of the cost functions. Another method for finding the minimum solution is to linearize the system with a first-order linear approximation such as a Taylor series expansion. Thus, the sensor and motion measurement functions take the form:

$$\begin{aligned} h(X) &\approx h(\hat{X}) + \nabla_X h(X)\Big|_{X=\hat{X}}(X-\hat{X}) \\ &\approx h(\hat{X}) + H(X-\hat{X}) \end{aligned} \tag{5.2.4}$$

where X is the true (unknown) state vector and $\hat{X}$ is the robot's estimate of the state vector. Expanding this equation for each of the cost functions and taking the first derivative to solve for its minimum, a recursive formulation of the estimator can be obtained as follows:

$$\sum_{i=1}^{n-1}\left(y_i - h_{y_i}(X)\right)^T P_i^{-1}\left(y_i - h_{y_i}(X)\right) + \sum_{i=1}^{n}\left(z_i - h_{z_i}(X)\right)^T R_i^{-1}\left(z_i - h_{z_i}(X)\right) \approx$$

$$\sum_{i=1}^{n-1}\left(y_i - h_{y_i}(X) - H_{y_i}(X-\hat{X})\right)^T P_i^{-1}\left(y_i - h_{y_i}(X) - H_{y_i}(X-\hat{X})\right) +$$

$$\sum_{i=1}^{n}\left(z_i - h_{z_i}(X) - H_{z_i}(X-\hat{X})\right)^T R_i^{-1}\left(z_i - h_{z_i}(X) - H_{z_i}(X-\hat{X})\right) =$$

$$\sum_{i=1}^{n-1}\left((y_i - h_{y_i}(X) + H_{y_i}(\hat{X})) - H_{y_i}(X)\right)^T P_i^{-1}\left((y_i - h_{y_i}(X) + H_{y_i}(\hat{X})) - H_{y_i}(X)\right) +$$

$$\sum_{i=1}^{n}\left((z_i - h_{z_i}(X) + H_{z_i}(X)) - H_{z_i}(\hat{X})\right)^T R_i^{-1}\left((z_i - h_{z_i}(X) + H_{z_i}(X)) - H_{z_i}(\hat{X})\right) =$$

$$\sum_{i=1}^{n-1}(H_{y_i}X)^T P_i^{-1}(H_{y_i}X) + \sum_{i=1}^{n-1}\left(y_i - h_{y_i}(\hat{X}) + H_{y_i}(\hat{X})\right)^T P_i^{-1}\left(y_i - h_{y_i}(\hat{X}) + H_{y_i}(\hat{X})\right) -$$

$$\sum_{i=1}^{n-1}(H_{y_i}X)^T P_i^{-1}\left(y_i - h_{y_i}(\hat{X}) + H_{y_i}(\hat{X})\right) - \sum_{i=1}^{n-1}\left(y_i - h_{y_i}(\hat{X}) + H_{y_i}(\hat{X})\right)^T P_i^{-1}(H_{y_i}X) +$$

$$\sum_{i=1}^{n}(H_{z_i}X)^T R_i^{-1}(H_{z_i}X) + \sum_{i=1}^{n}\left(z_i - h_{z_i}(\hat{X}) + H_{z_i}(\hat{X})\right)^T R_i^{-1}\left(z_i - h_{z_i}(\hat{X}) + H_{z_i}(\hat{X})\right) -$$

$$\sum_{i=1}^{n}(H_{z_i}X)^T R_i^{-1}\left(z_i - h_{z_i}(\hat{X}) + H_{z_i}(\hat{X})\right) - \sum_{i=1}^{n}\left(z_i - h_{z_i}(\hat{X}) + H_{z_i}(\hat{X})\right)^T R_i^{-1}(H_{z_i}X) \tag{5.2.5}$$

This function is quadratic in X. To minimize the function with respect to X, the first derivative is taken and the equations are set to 0. This results in an equation with the following form:

$$2\sum_{i=1}^{n-1}(H_{y_i}^T P_i^{-1} H_{y_i})X - 2\sum_{i=1}^{n-1} H_{y_i} P_i^{-1}\left(y_i - h_{y_i}(\hat{X}) + H_i\hat{X}\right) +$$

$$2\sum_{i=1}^{n}(H_{z_i}^T R_i^{-1} H_{z_i})X - 2\sum_{i=1}^{n} H_{z_i} R_i^{-1}\left(z_i - h_{z_i}(\hat{X}) + H_i\hat{X}\right) = 0 \Leftrightarrow$$

$$X = \left(\sum_{i=1}^{n-1} H_{y_i}^T P_i^{-1} H_{y_i} + \sum_{i=1}^{n} H_{z_i}^T R_i^{-1} H_{z_i}\right)^{-1}$$

$$\left(\sum_{i=1}^{n-1} H_{y_i}^T P_i^{-1}\left(y_i - h_{y_i}(\hat{X})\right) + \sum_{i=1}^{n} H_{z_i}^T R_i^{-1}\left(z_i - h_{z_i}(\hat{X})\right) + \sum_{i=1}^{n-1}\left(H_{y_i} P_i^{-1} H_{y_i}\right)\hat{X} + \sum_{i=1}^{n}\left(H_{z_i} R_i^{-1} H_{z_i}\right)\hat{X}\right) \Leftrightarrow$$

$$X = \hat{X} + \left(\sum_{i=1}^{n-1} H_{y_i}^T P_i^{-1} H_{y_i} + \sum_{i=1}^{n} H_{z_i}^T R_i^{-1} H_{z_i}\right)^{-1}\left(\sum_{i=1}^{n-1} H_{y_i} P_i^{-1}\left(y_i - h_{y_i}(\hat{X})\right) + \sum_{i=1}^{n} H_{z_i} R_i^{-1}\left(z_i - h_{z_i}(\hat{X})\right)\right) \tag{5.2.6}$$

Where the X on the right-hand side of the equation is the initial estimate of the system. The first value of this estimate can be obtained from the robot's raw odometry, if no other estimate is available. This is a recursive form where the result from the left-hand side of the equation is plugged back into the equation on the right-hand side. This first-order linear approximation of the measurement function is only valid for small errors in the estimate of X. As the equations are iterated, the state estimate should converge to a final solution.

5.2.1.1 Odometry Propagation Measurement

The measurement function for the distance estimates between subsequent nodes based on their odometry is defined as:

$$h_{y_i}(X) = {}^{\mathcal{R}}_{\mathcal{G}}C^T(\phi_R)\left(X_{L_i} - X_{L_{i-1}}\right) \tag{5.2.7}$$

where $X_{L_i} = [x_i \ y_i \ \phi_i]^T$ and $X_{L_{i-1}} = [x_i \ y_i \ \phi_i]^T$ are the positions of the robot at time i and $i-1$, respectively, and

$${}^{\mathcal{R}}_{\mathcal{G}}C(\phi_R) = \begin{bmatrix} \cos\phi_R & -\sin\phi_R \\ \sin\phi_R & \cos\phi_R \end{bmatrix} \tag{5.2.8}$$

is the rotation matrix that relates the orientation of the frame of reference $\mathcal{R}$ on the robot with the global coordinate frame $\mathcal{G}$.

The first-order Taylor approximations of the odometry measurement function is defined as:

$$\begin{aligned}
\tilde{y}_i = & \begin{bmatrix} -C^T(\hat{\phi}_{L_{i-1}}) & -C^T(\hat{\phi}_{L_{i-1}})J\left(\hat{X}_{L_i} - \hat{X}_{L_{i-1}}\right) & \vdots & C^T(\hat{\phi}_{L_{i-1}}) \end{bmatrix} \begin{bmatrix} \tilde{X}_{L_{i-1}} \\ \tilde{X}_{L_i} \end{bmatrix} \\
= & \begin{bmatrix} H_{L_{i-1}} & \vdots & H_{L_i} \end{bmatrix} \begin{bmatrix} \tilde{X}_{L_{i-1}} \\ \tilde{X}_{L_i} \end{bmatrix} \\
= & H_{y_i} \begin{bmatrix} \tilde{X}_{L_{i-1}} \\ \tilde{X}_{L_i} \end{bmatrix}
\end{aligned} \tag{5.2.9}$$

where

$$J = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{5.2.10}$$

and $\tilde{X}$ is the estimated error in the state vector $(X - \hat{X})$. These expressions for the error terms are only important for calculating the Jacobian and are not used for any other part of the estimator.

5.2.1.2 Place Sensor Measurement

The second kind of sensor reading is the estimated distance between two nodes based on the virtual place sensor's reading that are on the same location. Orientations of these landmarks are not tracked as some sensor modalities may not have an orientation associated with its readings. This function is defined as:

$$h_{z_i}(X) = \left(X_{p_i} - X_{p_j}\right) \tag{5.2.11}$$

where $X_{p_i} = [x_i \ y_i]^T$ and $X_{p_j} = [x_j \ y_j]^T$ are the global 2D poses of the robot's position

(orientation is not considered).

Likewise, the first-order Taylor approximations of the place sensor is defined as:

$$\begin{aligned}
\tilde{z}_i &= \begin{bmatrix} -1 & 0 & \vdots & 1 & 0 \\ 0 & -1 & \vdots & 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{X}_{p_j} \\ \tilde{X}_{p_i} \end{bmatrix} \\
&= \begin{bmatrix} H_{p_j} & \vdots & H_{p_i} \end{bmatrix} \begin{bmatrix} \tilde{X}_{p_j} \\ \tilde{X}_{p_i} \end{bmatrix} \qquad (5.2.12) \\
&= H_{z_i} \begin{bmatrix} \tilde{X}_{p_j} \\ \tilde{X}_{p_i} \end{bmatrix} \qquad (5.2.13)
\end{aligned}$$

5.2.2 Simulation Results

This estimator was run on the simulated data shown in Figure 5.11(b). Figure 5.14 shows plots of the covariance matrices associated with each of the individual odometry readings (the y_i values) at each of the locations where sensor readings were taken. Each of the odometric readings are considered to be independent of each other and thus, the covariance matrices are only defined between a single pair of sensor readings. Because of the extreme nonlinearities in the system, this ML algorithm must be iterated several times until convergence. The convergence of this algorithm also depends greatly upon the initial positions of the nodes.

Figure 5.15 illustrates the multi-step process of how the linearized ML estimator converges to a solution. The uncorrected odometric readings are used as the initial estimate for the state vector. The iterative process was stopped when the average landmark update per iteration dropped below 0.001 m. In this experiment, only four iterations of the algorithm were necessary before the stopping condition was reached.

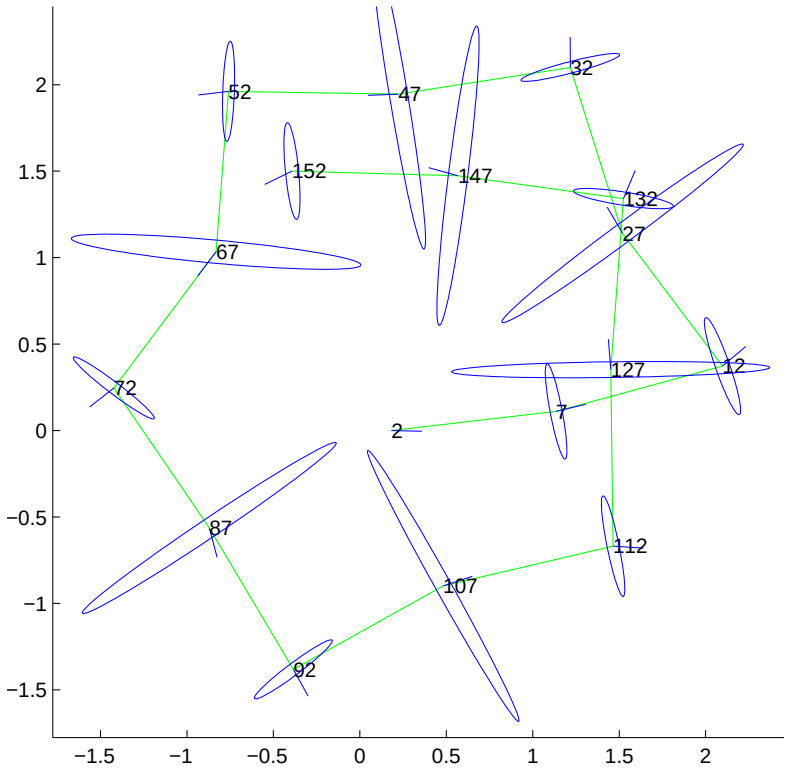
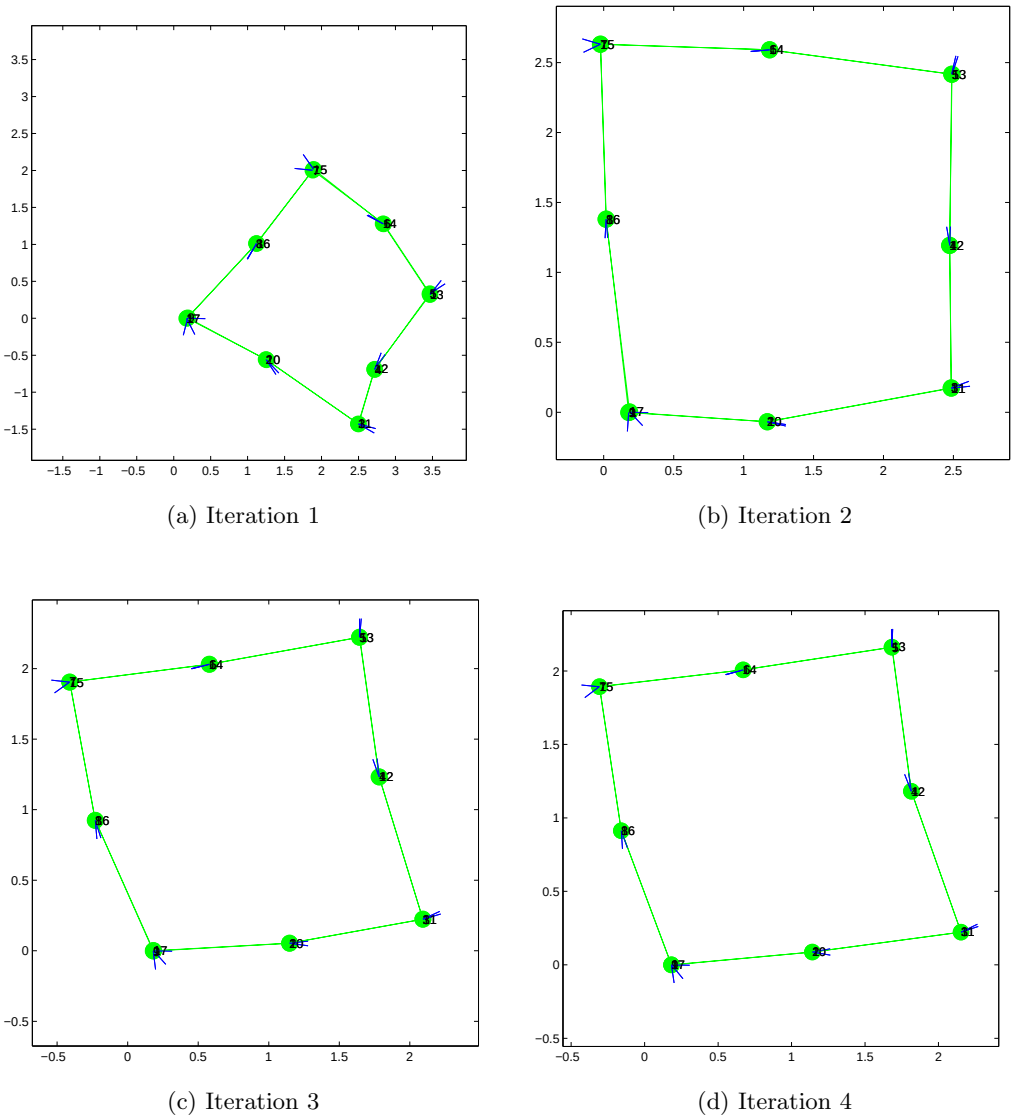


Figure 5.14: Covariance matrices for each of the independent odometric readings used by the linearized maximum likelihood estimator.



(a) Iteration 1

(b) Iteration 2

(c) Iteration 3

(d) Iteration 4

Figure 5.15: Four steps in the convergence of the linearized maximum likelihood estimator. By the fourth step, the estimate is very nearly converged.

Chapter 6

Extended Kalman Filter Estimator

This chapter describes how a Kalman filter estimator for an appearance-based mapping system can be derived. As described previously, such a system uses an environmental sensor that neither relies on any specific type of features, nor takes distance measurements to landmarks. Such a sensor determines a unique sensor signature for distinct locations along the robot's path, stores the signature and the estimated pose of the robot at that time instant, and finally retrieves that information once the robot revisits the same area. Determining whether the robot is at a certain location for a second time is the key element for providing positioning updates. By correlating any two scenes, a relative position measurement can be inferred and be used to update both the current and previous (at locations visited in the past) pose estimates for the robot. This will produce an accurate map of distinct locations within the area that the robot has explored.

6.1 Kalman Filter Derivation

This section describes the derivation of the Kalman filter for estimating the current pose of the robot moving in a 2D environment and the landmarks it observes. Table 6.1 shows the variable notation that will be used in this section.

X	true quantity of a state variable vector
$\hat{X}$	estimated quantity of a state vector
$\widetilde{X}$	accumulated error of a state vector
$X_R = [x_r \ y_r \ \phi_r]$	state vector for the robot
$X_{L_i} = [x_{l_i} \ y_{l_i}]$	state vector of landmark i

Table 6.1: State variable notation

Taking the robot's orientation into account will introduce non-linearities into the system because of the use of transcendental functions to describe the effect of orientation errors on the (x, y) position estimates. To handle these orientation errors, the extended form of the Kalman filter (EKF) must be used. The EKF is a first-order linearized approximation of the nonlinear estimated system and relies on the assumption that errors in the nonlinear parts of the system will be small. When this assumption is violated, as it is in the case of small robot navigation, the state update equations can be iterated multiple times until the effects of the nonlinearities are smoothed out.

6.1.1 Propagation

The extended Kalman filter uses a model of the robots kinematics to compute an estimate of the robot's position at discrete timesteps. Associated with this state estimate is a covariance matrix which measures the uncertainty in the robot's position over time.

A generic set of equations are used for the vehicle's odometry. This allows the method to be easily adapted to different types of vehicles. The continuous time equations for the motion expressed in local coordinates (with respect to a frame of reference $\mathcal{R}$ attached to the robot) are:

$$\dot{x}_r = V \tag{6.1.1}$$

$$\dot{y}_r = 0 \tag{6.1.2}$$

$$\dot{\phi}_r = \omega \tag{6.1.3}$$

where V and ω are the real linear and angular velocity of the robot. The general expressions for the linear and rotational velocities of the robot are described as:

$$x_r(k+1) = x_r(k) + V(k)\delta t \cos\phi_r(k) \tag{6.1.4}$$

$$y_r(k+1) = y_r(k) + V(k)\delta t \sin\phi_r(k) \tag{6.1.5}$$

$$\phi_r(k+1) = \phi_r(k) + \omega(k)\delta t \tag{6.1.6}$$

where δt is the time interval between two consecutive odometric measurements. These equations describe the motion of the robot if the linear and rotational velocities of the robot could be measured perfectly. Since the robot must measure its velocities with some proprioceptive sensor, such as the signals from wheel encoders, these measurements will be corrupted by noise due to the interactions of the robot and the environment (i.e. wheel slippage). Measured in discrete time, these quantities are:

$$V_m(k) = V(k) + w_v(k) \tag{6.1.7}$$

$$\omega_m(k) = \omega(k) + w_\omega(k) \tag{6.1.8}$$

where $w_v(k)$, $w_\omega(k)$ are the zero-mean, white Gaussian noise processes, with known covariance $Q(k) = E\{W_R W_R^T\}$, $W_R = [w_v(k)\ w_\omega(k)]^T$, that contaminate the velocity mea-

surement signals. By integrating the velocity measurements, the estimate of the state variable $\hat{X}_R = \begin{bmatrix} \hat{x}_r & \hat{y}_r & \hat{\phi}_r \end{bmatrix}^T$ is propagated as:

$$\hat{x}_r(k+1) = \hat{x}_r(k) + V_m(k)\delta t \cos\hat{\phi}_r(k) \quad (6.1.9)$$

$$\hat{y}_r(k+1) = \hat{y}_r(k) + V_m(k)\delta t \sin\hat{\phi}_r(k) \quad (6.1.10)$$

$$\hat{\phi}_r(k+1) = \hat{\phi}_r(k) + \omega_m(k)\delta t \quad (6.1.11)$$

When a robot attempts to estimate its position based on its proprioceptive sensors, it does not have an absolute frame of reference to compare against its pose estimate. Thus, the robot must calculate the uncertainty of the estimate based on the difference between the estimates of its unknown true position and the position estimate based on its noisy encoders.

Based on Equations (6.1.3), (6.1.7), (6.1.8), (6.1.11), (6.1.6), the linearized discrete-time error-state propagation equation in global coordinates is the difference between the estimated state and the true state:

$$\begin{bmatrix} \widetilde{x}_r \\ \widetilde{y}_r \\ \widetilde{\phi}_r \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & 0 & -V_m\delta t \sin\phi_r \\ 0 & 1 & V_m\delta t \cos\phi_r \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \widetilde{x}_r \\ \widetilde{y}_r \\ \widetilde{\phi}_r \end{bmatrix}_k + \begin{bmatrix} \delta t\cos\phi_r & 0 \\ \delta t\sin\phi_r & 0 \\ 0 & \delta t \end{bmatrix} \begin{bmatrix} w_v \\ w_\omega \end{bmatrix}_k \quad (6.1.12)$$

or

$$\widetilde{X}_R(k+1) = \Phi_R(k+1)\widetilde{X}_R(k) + G_R(k+1)W_R(k) \quad (6.1.13)$$

with

$$\Phi_R = \begin{bmatrix} 1 & 0 & -V_m \delta t \sin\phi_R \\ 0 & 1 & V_m \delta t \cos\phi_R \\ 0 & 0 & 1 \end{bmatrix}, G_R = \begin{bmatrix} \delta t \cos\phi_R & 0 \\ \delta t \sin\phi_R & 0 \\ 0 & \delta t \end{bmatrix} \tag{6.1.14}$$

Each time the robot propagates the state vector, the estimated errors in the state must also be propagated. Odometric errors will continuously decrease the quality of the robot's estimate which will increase the robot's position uncertainty. The equation for the propagation of the robot's position error covariance matrix is:

$$\begin{aligned} P_R(k+1/k) &= E[\widetilde{X}(k+1)\widetilde{X}^T(k+1)] \\ &= \Phi_R(k) P_R(k/k) \Phi_R^T(k) + G_R(k) Q_R(k) G_R^T(k) \end{aligned} \tag{6.1.15}$$

where

$$P_R(k/k) = \begin{bmatrix} \sigma_{P_x}^2 & \sigma_{P_x}\sigma_{P_y} & \sigma_{P_x}\sigma_{P_\phi} \\ \sigma_{P_y}\sigma_{P_x} & \sigma_{P_y}^2 & \sigma_{P_y}\sigma_{P_\phi} \\ \sigma_{P_\phi}\sigma_{P_x} & \sigma_{P_\phi}\sigma_{P_y} & \sigma_{P_\phi}^2 \end{bmatrix} \tag{6.1.16}$$

is the covariance matrix of the robot's position and orientation. The notation (k/k) refers to the estimate at time k given k measurements. The Q_R matrix from the state error covariance propagation in Equation (6.1.15) represents the covariance of the robot's linear and translational motions. For a differentially-driven robotic platform such as the Scout, where linear and rotational velocities are a function of the left v_l and right v_r wheel speeds, i.e. $V_m = \frac{(v_l+v_r)}{2}$, $\omega_m = \frac{(v_l-v_r)}{\alpha}$, this matrix is defined as:

$$Q(k)_R = \begin{bmatrix} \frac{1}{4}(\sigma_{v_l}^2 + \sigma_{v_r}^2) & \frac{1}{2\alpha}(\sigma_{v_l}^2 - \sigma_{v_r}^2) \\ \frac{1}{2\alpha}(\sigma_{v_l}^2 - \sigma_{v_r}^2) & \frac{1}{\alpha^2}(\sigma_{v_l}^2 + \sigma_{v_r}^2) \end{bmatrix} \tag{6.1.17}$$

where σ_{v_l} and σ_{v_r} are the standard deviations of the wheel speed errors and α is the length of the wheelbase. As can be seen, the linear and rotational velocities are correlated (as long as the standard deviations of the linear and rotational velocity are non-zero and not equal).

6.1.1.1 Augmenting the Propagation Equations

If the robot's position was the only quantity in the state vector, these propagation equations would be sufficient. Since the positions of the landmarks must also be estimated when mapping, these quantities must also be integrated into the state vector. Unlike the robot, the coordinates of the landmark locations X_{L_i} do not change over time. The real, estimated, and error propagation equations for a landmark L_i are:

$$X_{L_i}(k+1) = X_{L_i}(k) \tag{6.1.18}$$

$$\hat{X}_{L_i}(k+1) = \hat{X}_{L_i}(k) \tag{6.1.19}$$

$$\tilde{X}_{L_i}(k+1) = I\tilde{X}_{L_i}(k) + 0W_R(k) \tag{6.1.20}$$

where I is the 2×2 identity matrix and 0 is the 2×2 zero matrix. By augmenting the state vector X with the poses (to be estimated) of all the landmarks X_{L_i} along the pose of the robot X_R,

$$X = \begin{bmatrix} X_R^T & X_{L_1}^T & \dots & X_{L_N}^T \end{bmatrix}^T \tag{6.1.21}$$

and employing Equations (6.1.12) and (6.1.20) we can derive an expression for the error propagation of this augmented state vector and the covariance matrix associated with it:

$$\begin{aligned} \tilde{X}(k+1) &= \Phi(k)\tilde{X}(k) + G(k)W_R(k) \\ P_{k+1/k} &= \Phi(k)P_{k/k}\Phi^T(k) + G(k)Q_R(k)G^T(k) \end{aligned} \tag{6.1.22}$$

where

$$\Phi(k) = \begin{bmatrix} \Phi_R(k) & 0 & \dots & 0 \\ 0 & I & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & I \end{bmatrix}, G(k) = \begin{bmatrix} G_R(k) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{6.1.23}$$

and once again, I is the 2×2 identity matrix.

6.1.2 Update

If the robot were to only propagate its state and state covariance estimates using the above equations, the covariance estimate would increase without bounds. To correct for odometric errors and to reduce the uncertainty, the robot must take a sensor reading and compare that reading with the expected reading given the current state estimate..

Every time the robot takes an image of its surroundings, it employs an algorithm to determine whether the sensor reading corresponds to a previously seen locations X_{L_i}, or to a novel location X_{L_j}. Using the notion of an appearance-based sensor model, the sensor reading will correspond to the quantity:

$$\begin{aligned} Z(k+1) &= 0_{2\times 1} + N_z(k+1) \\ &= {}^{\mathcal{R}}X_{L_i} + N_z \end{aligned} \tag{6.1.24}$$

where ${}^{\mathcal{R}}X_{L_i}$ is the landmark's state vector from the robot's coordinate system $\mathcal{R}$. The $0_{2\times 1}$ value is an *inferred* sensor reading which reflects the robot's assertion that its physical location directly corresponds to the sensor reading. That is, the only way the robot could receive this sensor reading is if ${}^{\mathcal{R}}X_R$ is the same location as ${}^{\mathcal{R}}X_{L_i}$. The robot is assumed not to have any other way to measure distances to landmarks and so any erroneous displacement in this reading is captured by the noise term $N_z(k+1)$.

The estimated sensor reading and estimated sensor reading error are:

$$Z = {}^{\mathcal{R}}_{\mathcal{G}}C^T(\phi_R)\,(X_{L_i} - p_R) + N_z \tag{6.1.25}$$

$$\hat{Z} = {}^{\mathcal{R}}_{\mathcal{G}}C^T(\hat{\phi}_R)\left(\hat{X}_{L_i} - \hat{p}_R\right) \tag{6.1.26}$$

where $p_R = [x_R\ y_R]^T$, $\hat{p}_R = [\hat{x}_R\ \hat{y}_R]^T$ and

$${}^{\mathcal{R}}_{\mathcal{G}}C(\phi_R) = \begin{bmatrix} \cos\phi_R & -\sin\phi_R \\ \sin\phi_R & \cos\phi_R \end{bmatrix} \tag{6.1.27}$$

is the rotation matrix that relates the orientation of the frame of reference $\mathcal{R}$ on the robot with the global coordinate frame $\mathcal{G}$. By subtracting the true sensor reading from the estimated sensor reading, the linearized measurement error is computed as:

$$
\begin{aligned}
\tilde{Z} &= Z - \hat{Z} \\
&\simeq \begin{bmatrix} -C^T(\hat{\phi}_R) & -C^T(\hat{\phi}_R) J \left(\hat{X}_{L_i} - \hat{p}_R \right) & \vdots & C^T(\hat{\phi}_R) \end{bmatrix} \begin{bmatrix} \tilde{p}_R \\ \tilde{\phi_R} \\ \tilde{X}_{L_i} \end{bmatrix} + N_z \\
&= \begin{bmatrix} H_R & \vdots & H_{L_i} \end{bmatrix} \begin{bmatrix} \tilde{X}_R \\ \tilde{X}_{L_i} \end{bmatrix} + N_z
\end{aligned} \tag{6.1.28}
$$

with

$$
\begin{aligned}
H_R &= \begin{bmatrix} -C^T(\hat{\phi}_R) & -C^T(\hat{\phi}_R) J \left(\hat{X}_{L_i} - \hat{p}_R \right) \end{bmatrix} \\
H_{L_i} &= C^T(\hat{\phi}_R) \\
J &= \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \\
\hat{p}_R &= \begin{bmatrix} \hat{x}_R \\ \hat{y}_R \end{bmatrix}
\end{aligned} \tag{6.1.29}
$$

Adding entries for all of the variables, the full equation is expressed as :

$$\begin{aligned}\tilde{Z} &= \begin{bmatrix} H_R & 0 & \dots & 0 & H_{L_i} & 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} \tilde{X}_R \\ \tilde{X}_{L_1} \\ \vdots \\ \tilde{X}_{L_{i-1}} \\ \tilde{X}_{L_i} \\ \tilde{X}_{L_{i+1}} \\ \vdots \\ \tilde{X}_{L_N} \end{bmatrix} + N_z \\ &= H\tilde{X} + N_z \end{aligned} \tag{6.1.30}$$

The $H(k+1)$ matrix is used to update the state estimate for the pose of the robot X_R and the positions of the landmarks X_{L_i} every time an image is recorded. The remainder of the update equations are:

$$r(k+1) = Z(k+1) - \hat{Z}(k+1) \tag{6.1.31}$$

$$S(k+1) = H(k+1)P_{k+1/k}H^T(k+1) + R(k+1) \tag{6.1.32}$$

$$K(k+1) = P_{k+1/k}H^T(k+1)S^{-1}(k+1) \tag{6.1.33}$$

$$\hat{X}_{k+1/k+1} = \hat{X}_{k+1/k} + K(k+1)r(k+1) \tag{6.1.34}$$

$$P_{k+1/k+1} = P_{k+1/k} - K(k+1)S(k+1)K^T(k+1) \tag{6.1.35}$$

The difference between the measured and estimated sensor reading in Equation (6.1.31) is called the residual. The covariance matrix of the residual is shown in Figure (6.1.32). These two values are used to compute the Kalman gain in Equation (6.1.33) which is used to correct the state vector and state covariance in Equation (6.1.34), and Equation (6.1.35),

respectively.

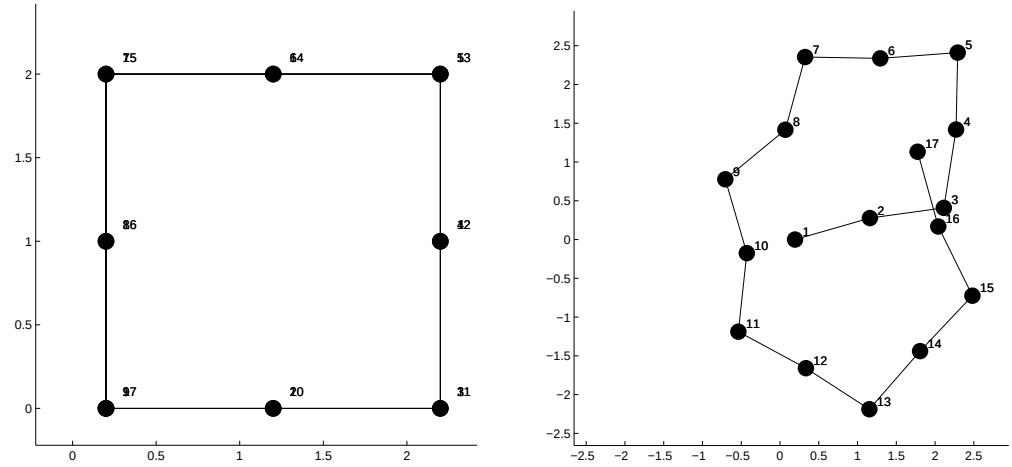
6.1.2.1 Iterative Extended Kalman Filter

Since the accuracy of this update depends on the accuracy of the linearization, we employ the Iterated form of the Extended Kalman filter (IEKF) [34], [64]. First, the IEKF linearizes the measurement equation Equation (6.1.28) around the current estimate $X_{k+1/k}$ of the state and calculates the updated state estimate $X_{k+1/k+1}$ using Equations (6.1.31), (6.1.32), (6.1.33), (6.1.34). Then the filter resets $X_{k+1/k}$ to this updated value and the same process is repeated until it converges (the rate of change in the state estimate drops below a preset threshold). The state covariance $P_{k+1/k}$ is *not* updated with Equation (6.1.35) until after the state estimate has converged. To repeatedly update the covariance on the same measurement would artificially and erroneously reduce the uncertainty in the state vector.

6.2 Simulation Experiment

This method was tested on a simulated Scout robot. The standard deviation of the estimated wheel encoder error was 1.4 cm/s. The true path of the simulated robot is shown in Figure 6.1(a) as a square that is traversed twice. Sensor snapshots are taken roughly every 0.5 m as the robot traverses the path. The first time around the loop, the robot is essentially in an exploration mode. Each landmark that it observes is unique and thus it adds the estimated positions of those landmarks directly to the state vector. Since the robot has no other information on those landmarks, the first sighting is the best information available. The second time around the loop, the robot re-discovers the landmarks that it saw on its first pass. If the Kalman update procedure is used, the odometric error in the robot's position will be adjusted so that they represent a more accurate reading. In addition to correcting the current position estimate, each of the previous landmark positions will also be corrected. If no update step is done, as shown in Figure 6.1(b), the robot's path estimate

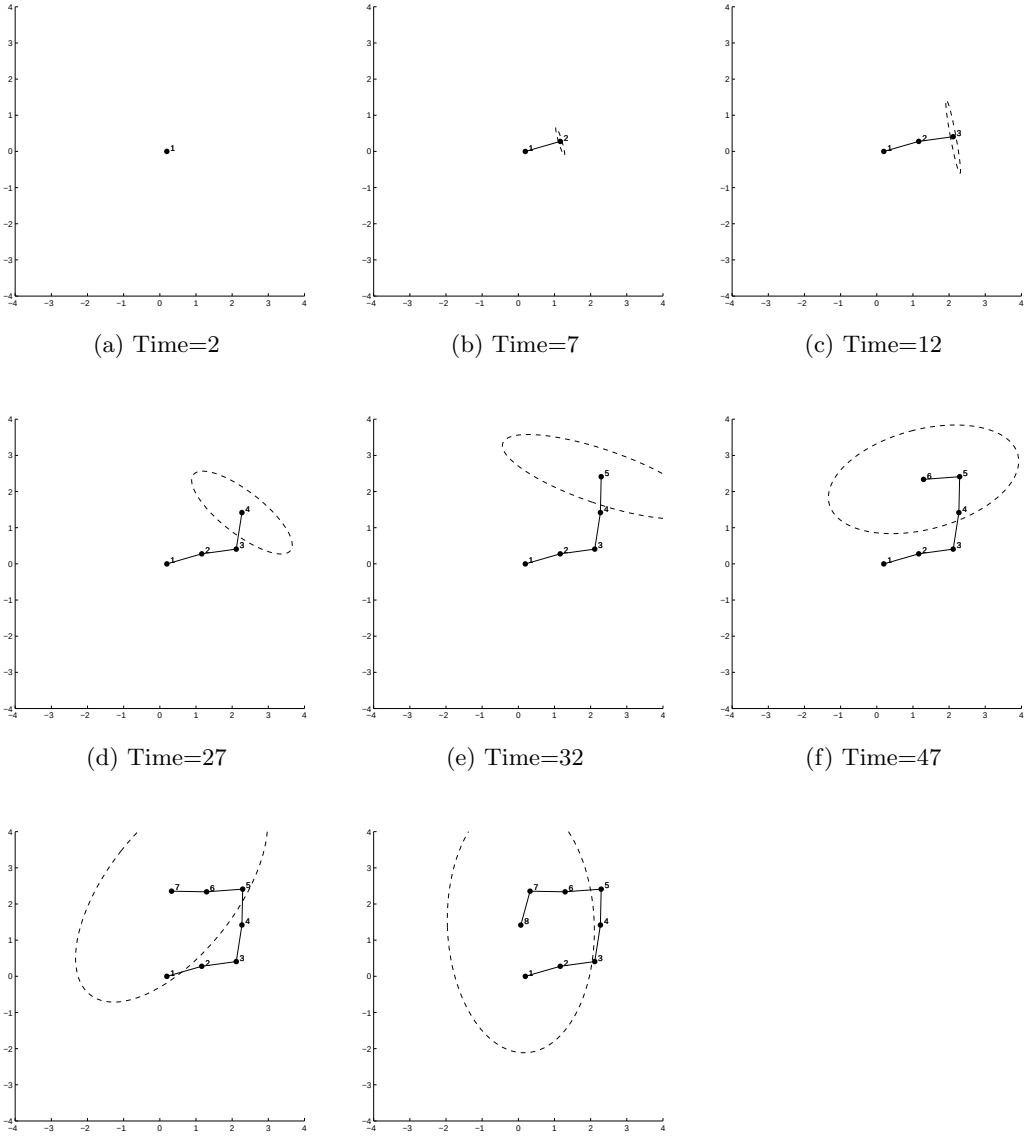
will be very poor and multiple positions will exist for the same landmark.



(a) True path (b) Pose estimate

Figure 6.1: True and estimated path for the simulation experiments. The path starts from the lower left, moves counter-clockwise, and is traversed twice. Sensor readings are taken at the corners of the square and at the midpoints of each path leg. The scale is in meters.

As the robot moves through the environment without any sensor updates, the certainty in its odometric estimate becomes increasingly worse. The covariance of the robot's position with no landmark corrections is shown in Figures 6.2 and 6.3.

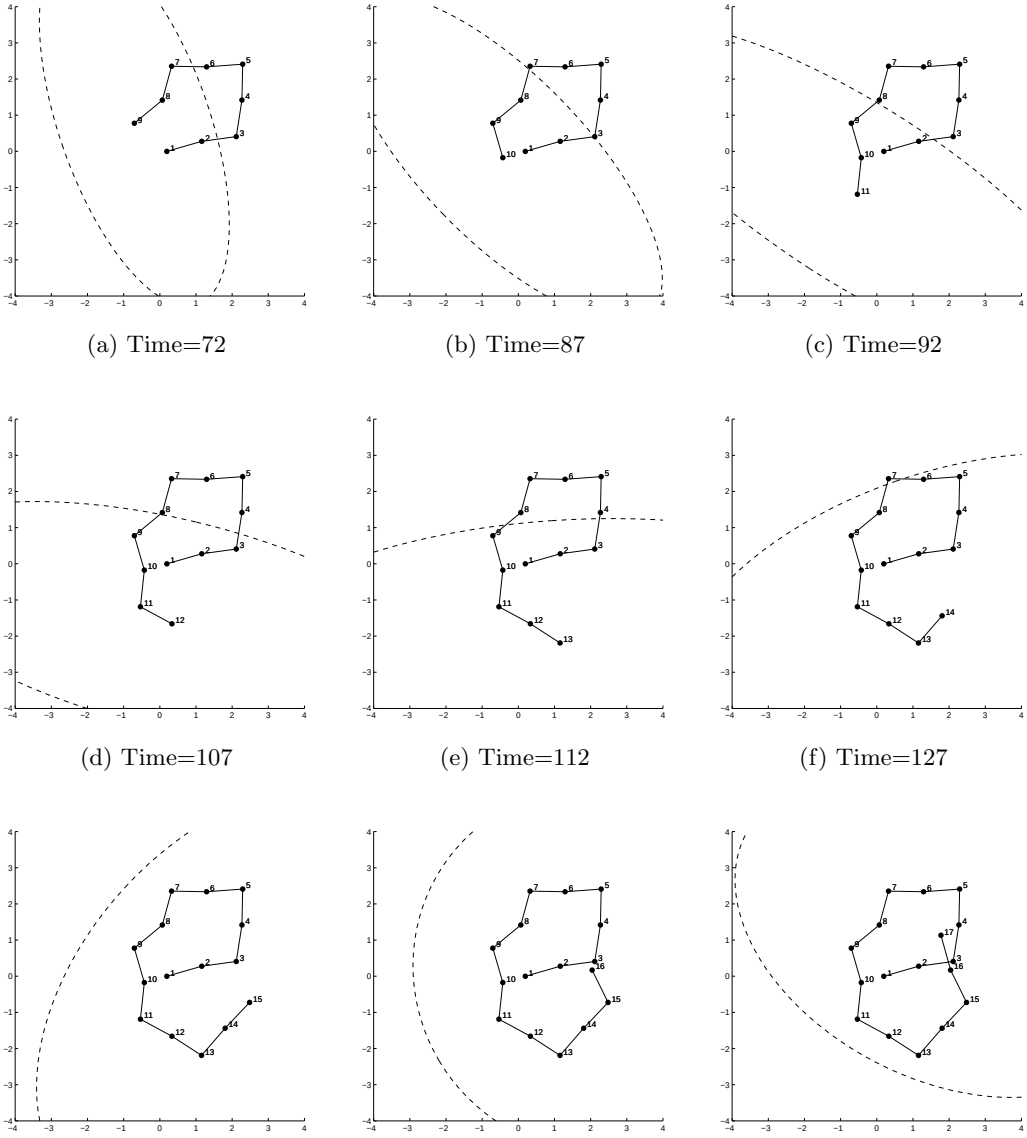


(a) Time=2 (b) Time=7 (c) Time=12

(d) Time=27 (e) Time=32 (f) Time=47

(g) Time=52 (h) Time=67

Figure 6.2: Propagation of uncertainty as the robot traverses its environment during the exploration phase (i.e. no landmark is observed more than once). Each subfigure represents the location where the robot has taken a sensor reading. The 3σ region of uncertainty is shown surrounding the robot's estimated position.



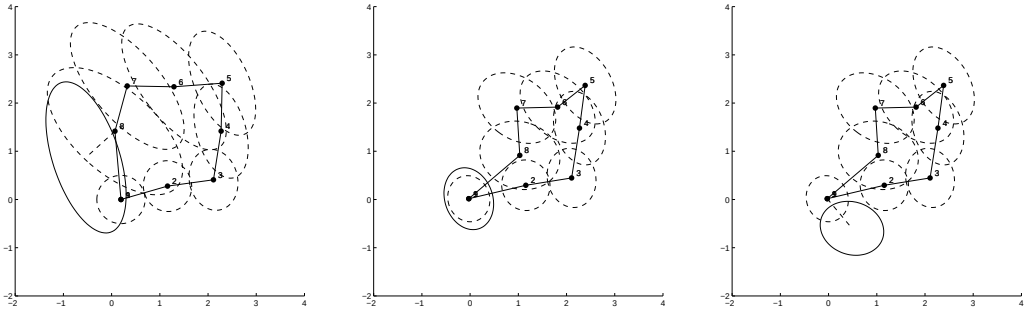
(a) Time=72 (b) Time=87 (c) Time=92

(d) Time=107 (e) Time=112 (f) Time=127

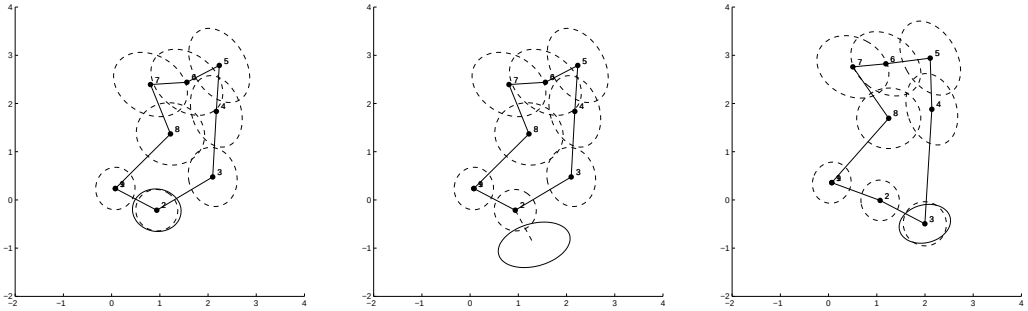
(g) Time=132 (h) Time=147 (i) Time=152

Figure 6.3: Propagation of uncertainty as the robot traverses its environment. No Kalman update step is done and so multiple positions exist for each landmark measurement and the position estimate becomes progressively worse with each step. Each subfigure represents the location where the robot has taken a sensor reading. The 3σ region of uncertainty is shown surrounding the robot's estimated position.

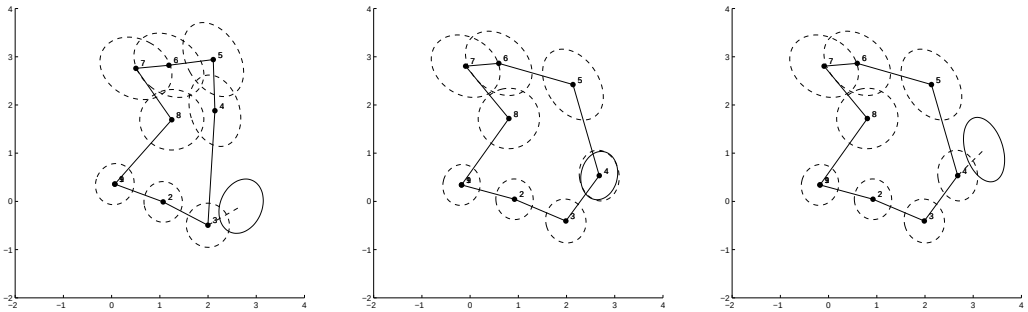
In contrast, Figures 6.4 and 6.5 show the landmark positions and position uncertainty of each location after correction by correlating the robot's position with the sensor readings. After the initial path around the cycle, the first subfigure (timestep 71) shows the large uncertainty accumulated in the robot's position. At timestep 72, the first update step is done and the uncertainty is greatly diminished. This is mostly due to the small covariance of the sensor reading vs. the large covariance of the robot's odometric propagation. After propagating to timestep 86, the error covariance of the robot's path estimate (shown as a dashed line) has generated a somewhat substantial error. This error is once again diminished in timestep 87 when another previously-seen landmark is observed.



(a) Time=71 (b) Time=72 (c) Time=86

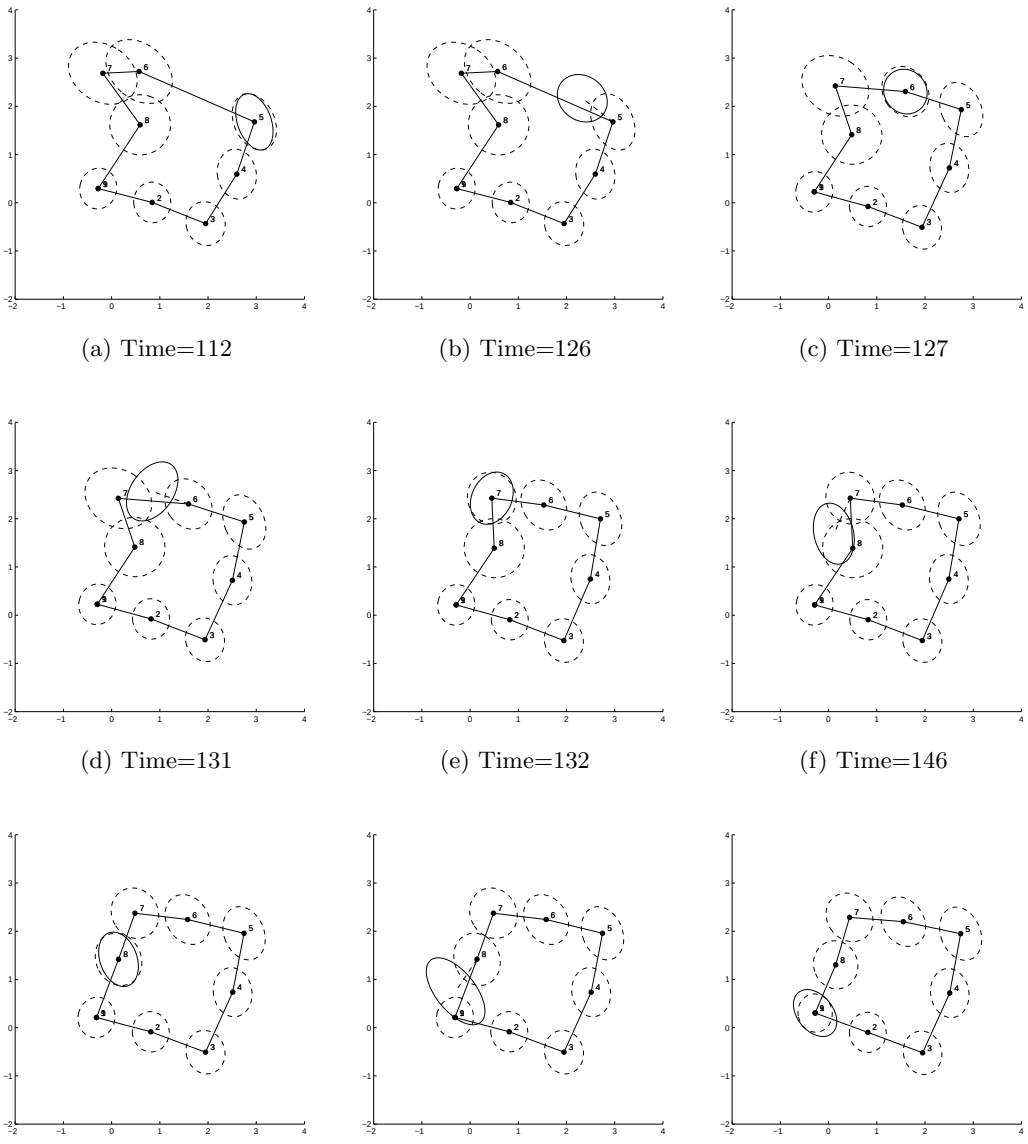


(d) Time=87 (e) Time=91 (f) Time=92



(g) Time=106 (h) Time=107 (i) Time=111

Figure 6.4: Propagation of uncertainty as the robot traverses its environment with Kalman update correction. Each pair of images (continued in Figure 6.5), shows the estimated position of the robot with uncertainty the timestep before and after the sensor reading was taken and the landmark positions were correlated. The estimated path of the robot just before the update is drawn with a dashed ellipse. The 3σ region of uncertainty is shown surrounding the robot's estimated position as a solid ellipse.



(a) Time=112 (b) Time=126 (c) Time=127

(d) Time=131 (e) Time=132 (f) Time=146

(g) Time=147 (h) Time=151 (i) Time=152

Figure 6.5: Propagation of uncertainty as the robot traverses its environment with Kalman update correction. This is continued from Figure 6.4.

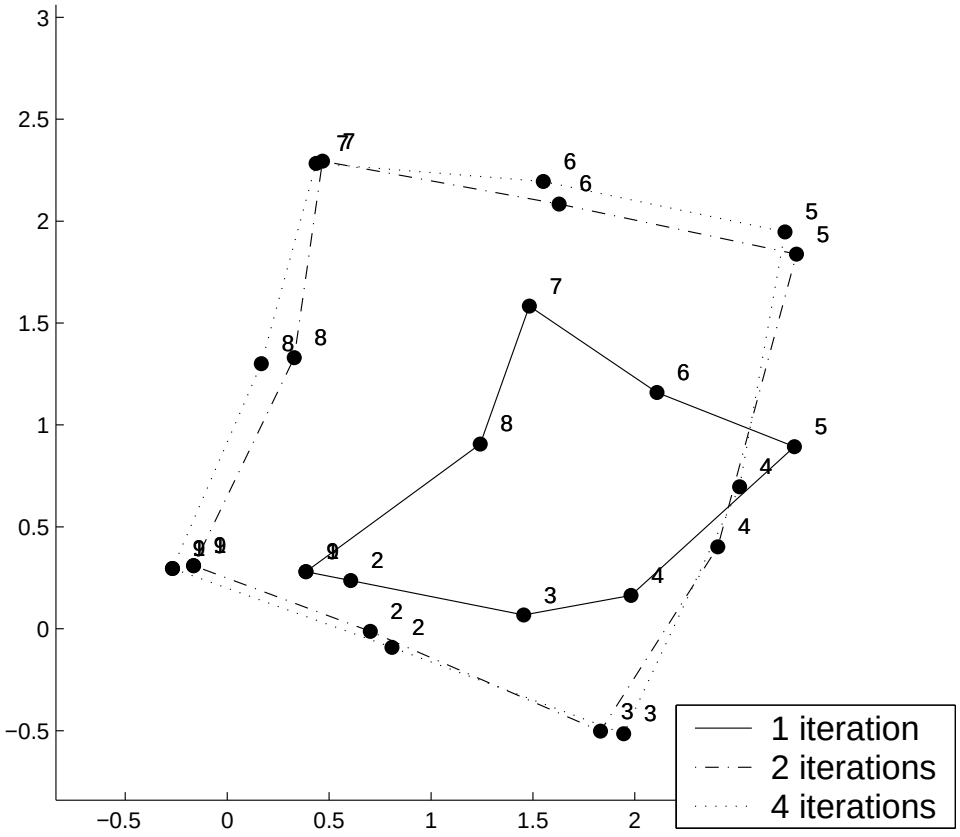


Figure 6.6: Effects of the iterative Kalman Filter on the position estimates. Final landmark positions for 1, 2, and 4 iterations per update step are shown.

Figures 6.6 and 6.7 illustrate how the estimated landmark positions are improved by using the IEKF and how the sensor residual improves with respect to the 3σ upper and lower bounds of the residual covariance estimates.

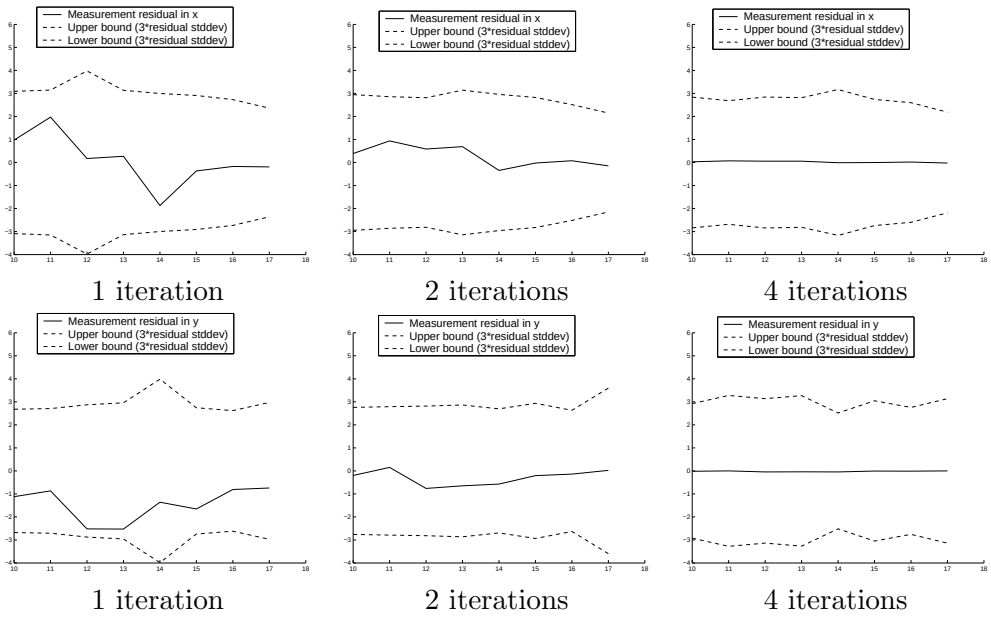


Figure 6.7: The effect of different numbers of iterations in the update step of the IEKF. The plots show the sensor residual $r = z - \hat{z}$ and the 3σ upper and lower bounds of the residual covariance S. The top row shows the residual in x and the bottom row shows the residual in y. These residuals are all for landmark positions that have been visited a second time.

Chapter 7

Data Association

7.1 Perceptual Aliasing

If the landmarks observed at each location were unique, then the task of matching two nodes which represent the same locations would be fairly straightforward. However, in many real world situation, this is unlikely to occur. Without pre-marking the environment and/or without extremely good *a priori* information, robots should assume that there might be multiple locations which appear to have the same sensor reading (a problem called "perceptual aliasing" [103]). Thus, the robot must have the ability to determine their most likely location in a set of sensor readings. To disambiguate its position, the sensor and motion history must be examined very carefully. This is where some knowledge of the spatial or temporal constraints associated with the sensor readings can be useful.

7.2 Markov Localization

One method for disambiguating the robot's position is to treat the set of all possible robot positions as a probability distribution and compute, at each timestep, the probability of being in each location. A recursive Bayesian filter algorithm, known as Markov localiza-

tion [29], can be employed for this purpose. Markov localization is an iterative algorithm by which the robot's believe in its position is gradually refined by filtering out invalid locations. Markov localization will compute, for each timestep, a distribution which shows the probability that the robot was at a particular node at a particular time.

In cases where the probability distribution is multi-modal, or where it is nearly equally likely that the robot was in more than one node at a time, there exists a good chance that those nodes are actually a single node that the robot has visited multiple times. The hypothesis with the highest probability of match from all of the timesteps is selected and those nodes are merged. Following the derivation of Thrun *et al.* [97] (included in Appendix B), this can be described formally as:

$$P\left(L^t|S,M\right) = \eta P\left(L^t|s^1,\ldots,s^t,M\right) P\left(L^t|s^{t+1},\ldots,s^t,M\right) \tag{7.2.1}$$

The first term on the right-hand side of Equation 7.2.1 is referred to as α^t and the second term will be referred to as β^t for notational convenience. Rearranging the conditional probabilities into a recursive Bayesian updating formulation produces the following equations:

$$\alpha^t = \eta P\left(s^t|L^t,M\right) \int P\left(L^t|s^{t-1},L^{t-1}\right) \alpha^{t-1} dL^{t-1} \tag{7.2.2}$$

$$\beta^t = \eta \int P\left(L^{t+1}|s^t,L^t\right) P\left(s^{t+1}|L^{t+1},M\right) \beta^{t+1} dL^{t+1} \tag{7.2.3}$$

The α^t term computes the probability that the robot is in a particular node in the graph at each time step t. This is represented as a probability mass L^t. Initially, at time $t = 0$, the robot's position is uniformly distributed across all nodes. At each timestep, the probability that the robot traversed from one node l_i^{t-1} to another l_j^t is computed. This probability is a function of the probability at each node at time $t-1$, the probability that the robot's

sensors detected the data at l_j, and the probability that the robot's odometry matches the path the robot took to reach l_j from node l_i. The β^t computes the same probability but works backwards in time instead of forward. The backwards step is important because it implicitly allows for the possibility that the robot may have been in a location in more than one timestep and allows the distribution to properly reflect this. Other than computing the values backwards in time, the β distribution is computed in a similar fashion as the α distribution. The values $P\left(s^t|L^t, M\right)$ and $P\left(L^t|s^{t-1}, L^{t-1}\right)$ represent expressions for the robot's sensor and motion models, respectively which will be described next.

7.3 Non-Parametric Sensor Model

The robot's sensor model can be described as $P\left(s^t|L^t, M\right)$. This is an expression for the probability that at time t, the robot's sensors obtain the reading s^t given that the estimate for the robot's position in the map M is given by the probability distribution L^t.

One way to generate this distribution is through a non-parametric method known as Parzen windows [74]. An arbitrary distribution can be approximated by a weighted sum of Gaussian distributions [1]. By placing a Gaussian distribution on each of the sample points, and normalizing each Gaussian by the number of points, an estimate for the probability distribution can be obtained. Figure 7.1 illustrates an example of how an unknown distribution can be estimated by drawing a number of samples from it.

Following the definition of conditional probabilities, the equation for the sensor model can be described as:

$$P\left(s^t|L^t, M\right) = \frac{P\left(s^t, L^t, M\right)}{P\left(L^t, M\right)} \tag{7.3.1}$$

where the joint probability distributions in the above equation are estimated through the sum of Gaussian kernels:

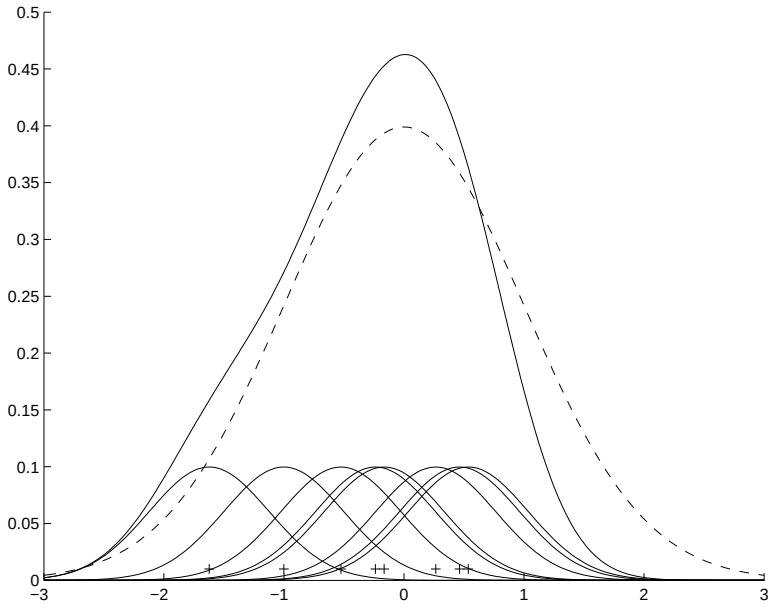


Figure 7.1: Example of a Parzen-window non-parametric density estimator. Samples are drawn from the unknown distribution (rendered with a dashed line). Uniformly-weighted Gaussian distributions are assigned to each of the points and their sum is the estimate of that distribution (rendered with a sold line).

$$\begin{aligned} P\left(s^t|L^t,M\right) =& \frac{P\left(s^t,L^t,M\right)}{P\left(L^t,M\right)} \\ =& \frac{\frac{1}{N}\sum_{n=1}^{N} g_s\left(s^t-s_n^t\right) g_d(d^t-d_n^t)}{\frac{1}{N}\sum_{n=1}^{N} g_d(d^t-d_n^t)} \end{aligned} \tag{7.3.2}$$

where g_s and g_l are Gaussian kernels. In both of these Gaussian kernels, the input parameter is the difference between two sensor readings or two robot positions. Thus, functions for computing these two differences must be computed. The input values to those Gaussians is a distance between sensor readings and locations. The distance between two sensor readings is a function based on the specific sensor modality employed. The distance between any two places in the robot's map is simply the shortest path between those two nodes in the graph.

7.4 Motion Model

The robot's motion model is expressed as $P\left(L^{t+1}|s^t, L^t\right)$, which represents the probability that the robot is in location L^{t+1} at time $t+1$ given that its odometry registered reading s^t after moving from location L^t at time t. This is represented as:

$$P\left(L^{t+1}|s^t, L^t\right) = g_e(e - \hat{e})g_\phi(\phi - \hat{\phi}) \tag{7.4.1}$$

where e and ϕ represent the linear and torsional components of the robot's motion in the current map and $\hat{e}$ and $\hat{\phi}$ represent the originally measured values (see Figure 7.2).

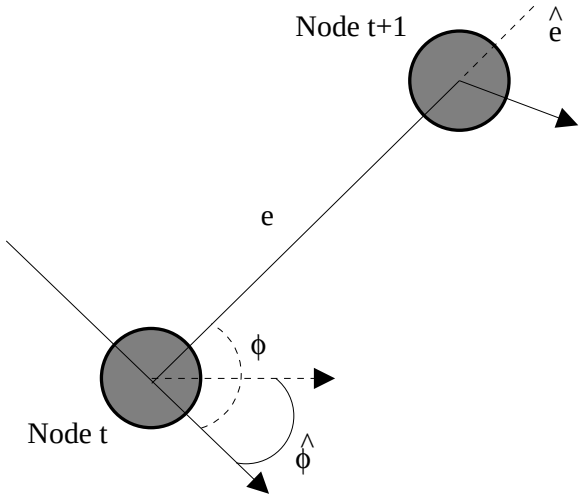


Figure 7.2: Example of the motion model. The current lengths e and angles ϕ between the nodes are represented as straight lines and the expected lengths $\hat{e}$ and angles $\hat{\phi}$ are shown with dashed lines.

7.5 Filtering Erroneous Matches

Positions are chosen for merging based on their relative probabilities. For each timestep t, the probability mass is scanned for nodes that have high probabilities that are similar to each other. The best choice is selected from the set and compared against the best choice

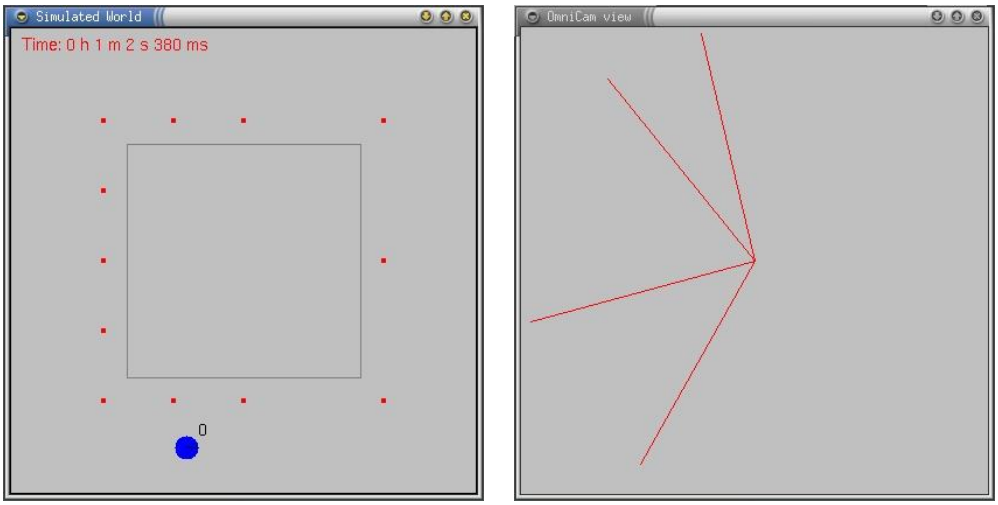
from each of the other n timesteps. The best match from these is chosen and applied to the map. Because this algorithm is something of a greedy search, there is no guarantee that the map will always be optimal. Additional knowledge about the environment can be employed as a consistency check to see whether the map makes sense. As the robot continues to move around, more information about the environment will be gathered and can also be used to get a more accurate estimate of the robot's position.

One such check is done by computing the average entropy

$$-\sum_i p(x_i) \ln p(x_i) \tag{7.5.1}$$

of the discrete probability masses and seeing whether after the merge, the average entropy decreases. A decrease in entropy will indicate an increase in certainty of the robot's position. An increase in entropy will mean that the variables in the probability mass are less distinct from one another and the robot's certainty in its position will decrease. If the robot becomes less certain of its position after a merge, then the node merge was likely to be incorrect and should be undone.

Other consistency checks are based on an evaluation of the specific estimator being used. For instance, the potential energy in the spring model estimator in Chapter 5 can be examined to see if it increases drastically. A drastic increase in potential energy would indicate that the structure has been bent far beyond its expected bounds. For the linearized MLE, the probability for the observed value for χ^2 can be evaluated to determine whether it is likely to be a correct match. For the Kalman filter (chapter 6) estimator, the residual can be evaluated against the residual covariance to see whether it is in the expected bounds.



(a) Simulated world (b) Bearing readings to landmarks

Figure 7.3: A simulated world in which the robot must localize itself based on ambiguous sensor readings.

7.6 Simulation Results

To illustrate how this process works, consider the following example. Figure 7.3 shows a simulated robot moving about a square environment where it takes bearing readings to the square landmarks that it observes. Since each of the landmarks are identical, the sensor reading is a vector of bearing angles between landmarks. The vectors are compared using he *Undirected Hausdorff metric H(A,B)* [46]. Since this metric is sensitive to outliers, the generalized undirected Hausdorff metric is used which looks for the k-th best match (rather than just the overall best match). This is defined as:

$$H(A,B) = \max_{kth}(h(A,B), h(B,A)) \tag{7.6.1}$$

$$h(A,B) = \max_{a \in A} \min_{b \in B} \parallel a_i - b_j \parallel \tag{7.6.2}$$

where $A = \{a_1, a_2, ..., a_m\}$ and $B = \{b_1, b_2, ..., b_n\}$, are two feature sets (differences between bearing measurements) and

$$\| a - b \| = \sum_i \sum_j | a_i - b_j | \tag{7.6.3}$$

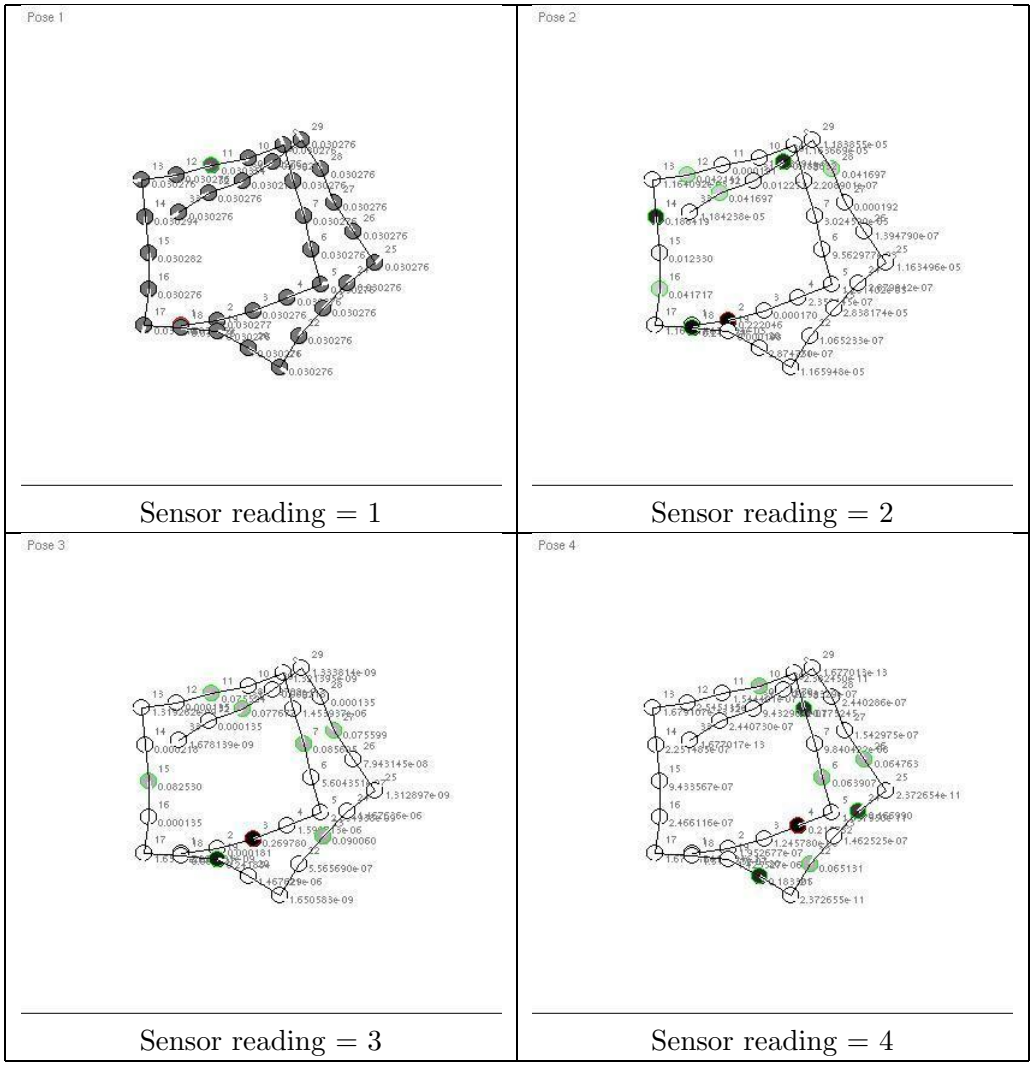


Figure 7.4: The first four probability masses computed from the simulated data set using the forward Markov localization step. The darker the circle, the more likely the probability that the robot is in that location.

The simulated robot travels around the corridor twice taking sensor readings every

several meters. Figure 7.4 illustrates the first four probability masses computed from the simulated dataset using the forward Markov localization step (the α tables). The initial probability distribution is initialized to be uniform and the motion and sensor models are propagated accordingly. There is a high degree of symmetry in this environment and thus each probability mass after the first has multiple places that describe the robot's pose with a very high probability.

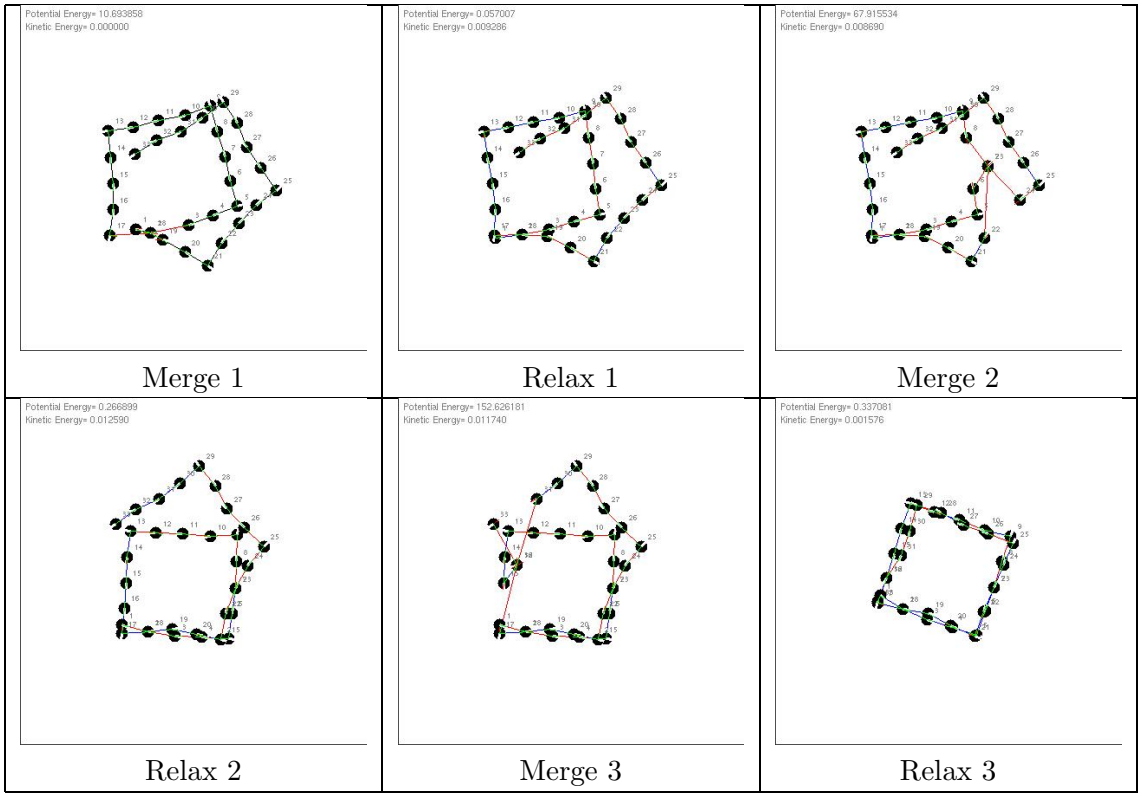


Figure 7.5: Example showing the merging and relaxation process for the first three nodes in the simulated environment. The spring-based estimator is used for this run.

Figure 7.5 shows the spring-based estimator being used to update the positions of the nodes in the map. Only the first three merges are shown in this figure. Figure 7.6 shows the measure of the entropy as a number of different merges are attempted. Not all merges decrease the entropy. Instead, some merges cause an increase in entropy and must be

discarded.

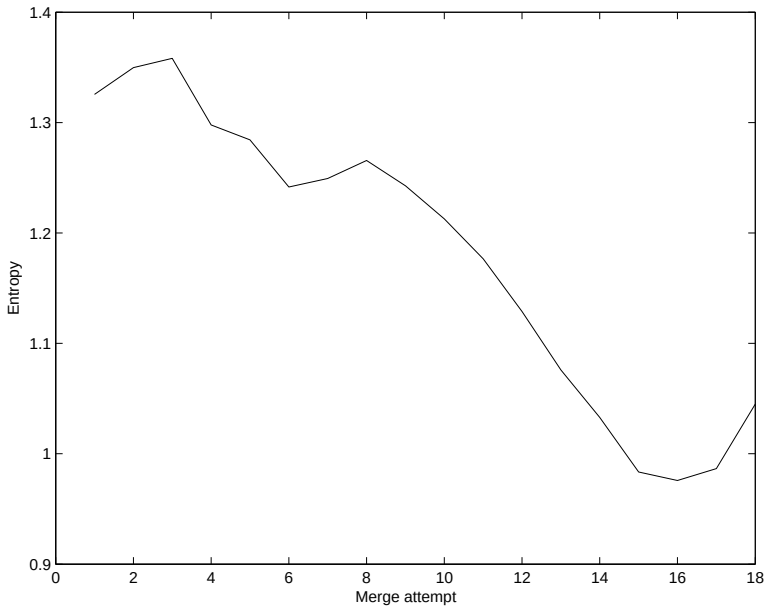


Figure 7.6: The measure of entropy as each merge is evaluated. If the entropy increases, the merge is rejected and the next best pair of nodes is tried. At the end, no more node can be merged and continue to decrease the average entropy of the distribution.

Chapter 8

Experimental Validation

8.1 Vision-Based Features

In their original design, the Scouts were equipped with cameras that have forward-facing lenses and have a field of view that is roughly 65°. For this work a Scout has been equipped with an upward-facing 190° vertical/360° horizontal field of view lens from Omnitech Robotics [73]. An example image taken from this camera and the corresponding de-warped image is shown in Figure 8.1.

In order to compute a signature for each location visited, a set of features must be identified and extracted from the image. However, in the most general case, the robot will be required to explore a completely unknown environment and as such, a specific feature detection algorithm chosen ahead of time could fail to find a distinctive set of features.

For this work, the Lucas-Kanade-Tomasi (KLT) feature tracking algorithm is used to compare on images to determine the degree of match. The KLT algorithm consists of a registration algorithm that makes it possible to find the best match between two images [60] as well as a feature selection rule which is optimum for the associated tracker under pure

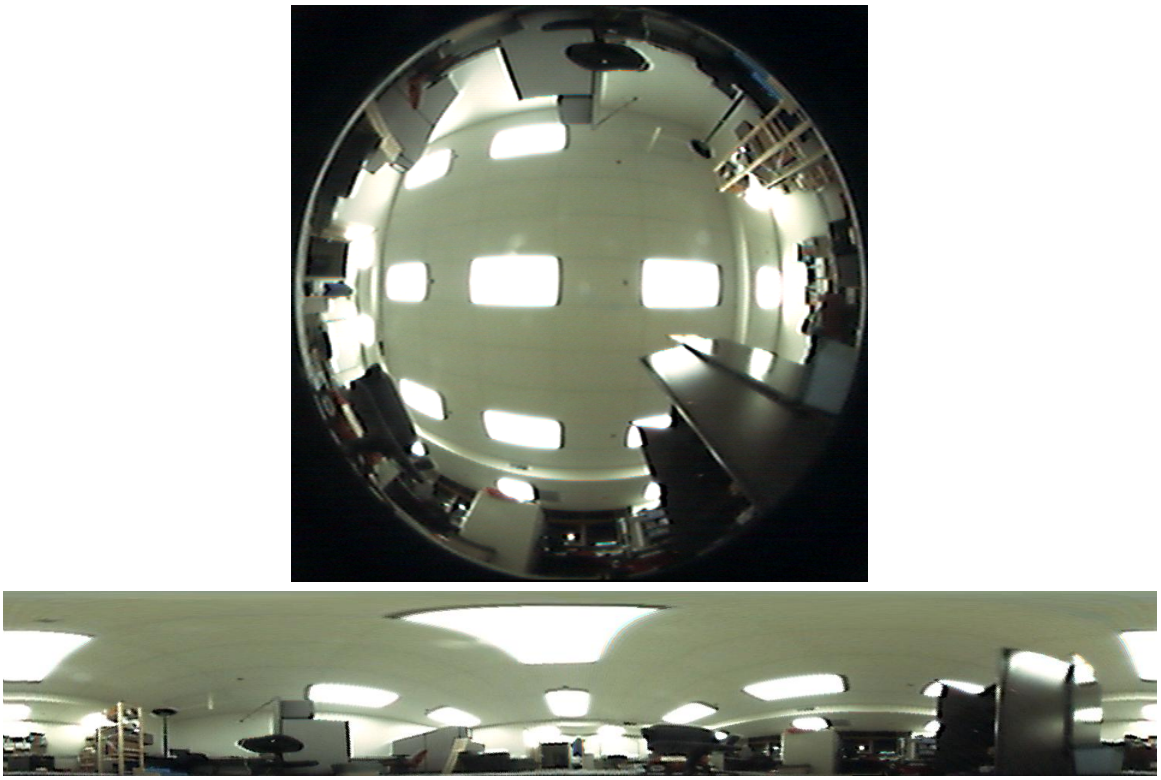


Figure 8.1: A raw and de-warped image taken from the Omnitech 190° lens.

translation between subsequent images [101]. An implementation of the KLT algorithm[1] is used to identify and track features between successive images as a method for determining the match between two images. KLT features are selected from each of the images and are tracked from one image to the next taking into account a small amount of translation for each of the features. The degree of match is the number of features successfully tracked from one image to the next. A total of 100 features are selected from each image and used for comparison. To take into account the possibility that two panoramic images might correspond to the same location but differ only in the orientation of the robot, the test image is rotated until the best match is found. Figure 8.2 shows the 100 best features identified in an image and shows how many of those features are successfully tracked to the

[1]Originally developed by Stan Birchfield at Stanford University [51].

Figure 8.2: The 100 best features selected by the KLT algorithm in the top image are shown as black squares in the top image. The bottom image shows how many features were tracked from the top image to the bottom image (corresponding to a robot translation of approximately 0.6 m.

lower image.

This approach is similar in flavor to [59] in that the image is reduced in resolution for the sake of rapid matching. In that work, a pyramid structure involving several levels of dimensionality reduction is created from each image and different images are matched from the lowest resolution to the highest. In our case, the KLT features serve as a single level of "dimensionality reduction" that is used for matching one image with the next.

It is important to note that no attempt is made to track the features over multiple frames of video. This technique does not attempt to compute structure from motion on this data primarily because the algorithms described in this research will ultimately be run on robots that do not have real-time video processing capability. Additionally, by adding the complexity of tracking features for SFM, the algorithm would not be transferable to classes of sensors that do not return that kind of information.

While mapping, the mobile robot travels around an unknown area and stores images from its camera. KLT is used to compare images recorded at different locations along the trajectory of the robot. When the received image does not match a previously recorded one, it is assumed that this location is novel and is added to the state vector of landmarks.

This constitutes an exploration phase where the robot creates its world model. When the robot encounters an image which matches one that was previously seen, it considers these features to be the same and corrects its estimate of the landmark position.

The KLT algorithm and omnicamera setup is treated as a "virtual sensor" that returns true or false as to whether the robot has returned to a location that it has visited before. This information is given to the estimators and a relative position measurement $Z = 0_{2\times 1} + N_z$ between the current position of the robot and that of the same location visited in the past is inferred. The accuracy of this measurement $R = E\{N_z N_z^T\}$ is inferred by the locus of points (forming an ellipsoid) around a location, with the characteristic that the images recorded at each of them are considered identical by the KLT.

8.2 Office Environment Experiment

The robot was moved around an environment in a path that intersected itself five times and an image was taken from the camera roughly every 0.3 m. The robot's path is shown in Figure 8.3.

The KLT algorithm was used to track features between each pair of images in order to find locations where the robot's path crossed itself. Figure 8.4(a) shows the true path of the robot and the locations where the path crossed itself and landmarks were thus observed. Figure 8.4(b) shows the estimated path of the robot as computed by the robot's noisy odometry readings. The estimated landmark positions observed during the run are shown as well. This figure does not assume that any sensor updates were made.

The different estimators were run on this dataset in order to compare their relative performances. Figure 8.5 shows the relative positions of the landmarks as computed by each of the estimators. The average Euclidean error between the estimated positions and ground truth is also shown.

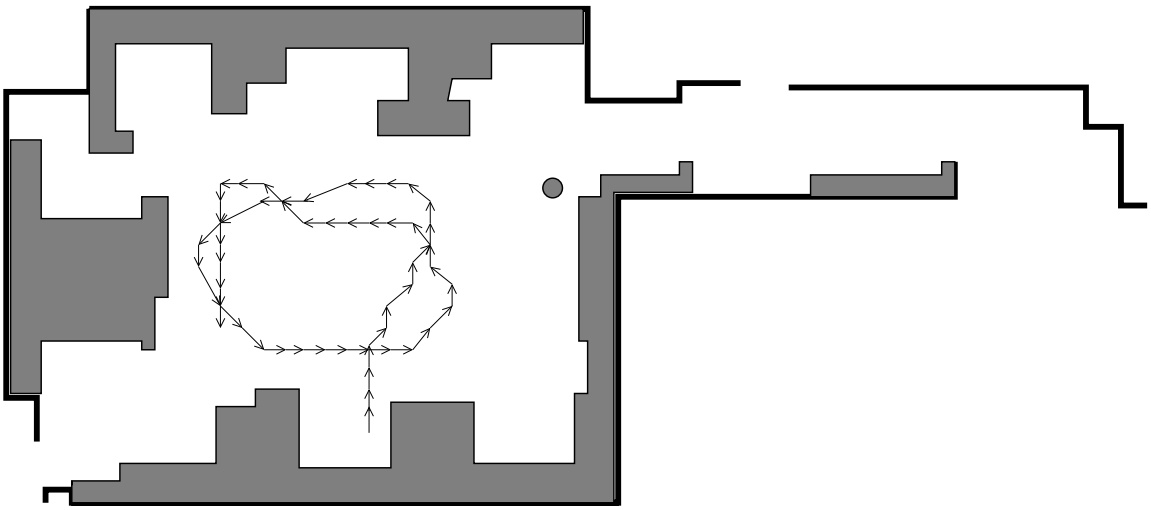
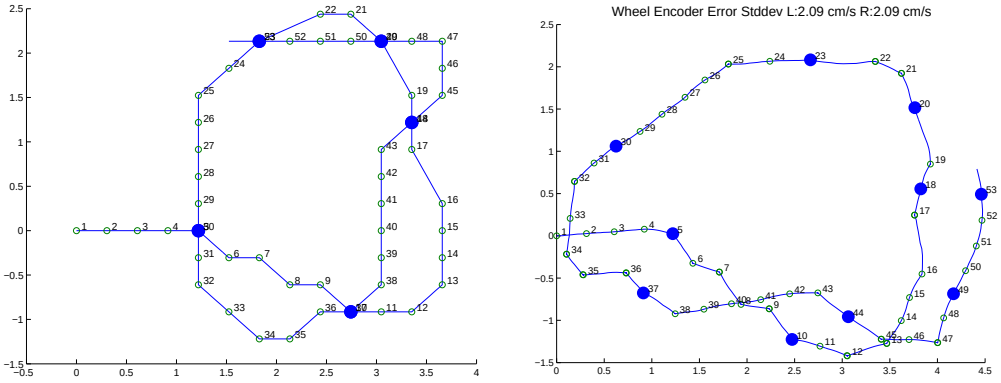


Figure 8.3: The path of the robot through the office environment.



(a) True path of the robot.

(b) Estimated path of the robot.

Figure 8.4: Real world experiments in an indoor environment (scale is in meters). Landmarks in the true path occur wherever there is an intersection in the path. Positions in the path are labeled chronologically.

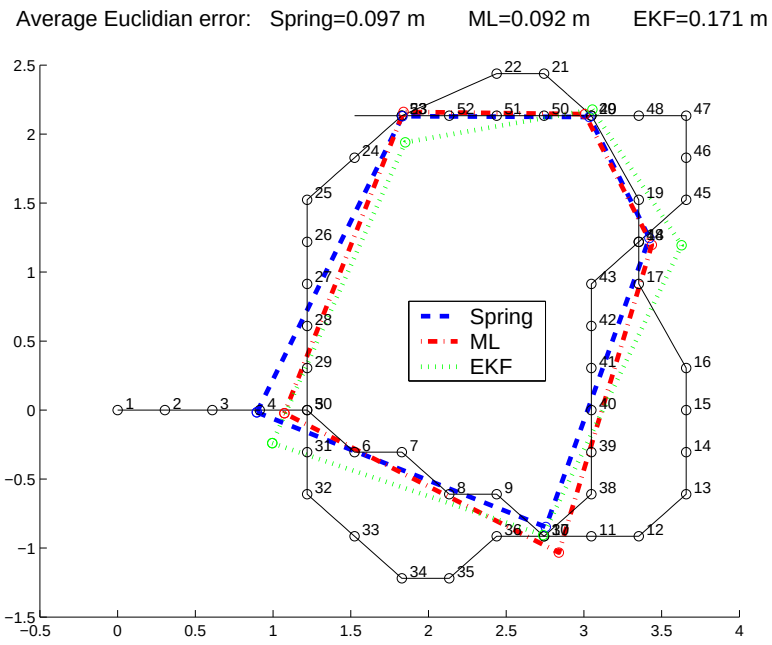
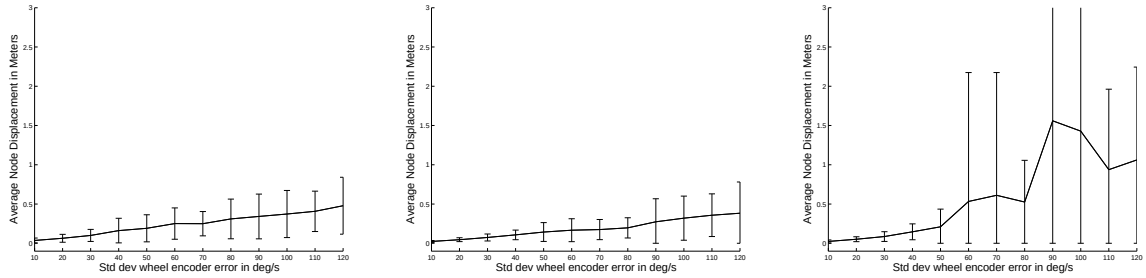


Figure 8.5: Performance of the three estimators on the office environment dataset. The performance of the estimators is described as the average Euclidean error of each of the landmark positions with respect to the true positions. These errors are computed after each dataset has been corrected for global misalignment.

8.2.1 Comparison of Estimators with Varying Noise Models

A series of synthetic paths were generated from the above data set and used to test the performance of each of the estimators using different odometric noise models. The simulated odometric noise ranged from a standard deviation of 10 deg / sec to 120 deg / sec in encoder error (in 10 deg increments). A set of 100 robot paths were created for each noise variance setting. For each path, both of the robot's wheel encoders was corrupted by noise drawn from a distribution with the same variance.

Figure 8.6 shows the results of the different estimators on paths affected by increasing levels of odometric error. The linearized ML estimator had the least amount of error in the placement of the landmarks, followed by the spring-based ML estimator. The performance of IEKF estimator was equivalent to the other estimators up to an error of around 50 deg / sec



(a) Spring ML (b) Linearized ML (c) IEKF

Figure 8.6: Comparison of the means and standard deviations of the three estimators on datasets with varying degrees of encoder error. Standard deviation of errors ranged from 10 deg / sec to 120 deg / sec.

but rapidly diminished in accuracy as the odometric errors increased.

8.3 Data Association Experiment

In the previous experiments, the office was cluttered enough such that 100 KLT features were sufficient to disambiguate all of the locations where the robot visited. A set of 320 images was taken at 0.3 m intervals in the office environment used for these experiments. Figure 8.7 shows a plot of the Euclidean distance estimate between each pair of locations as a function of the number of features that the KLT algorithm can track between the respective images. As can be seen, until the number of features tracked drops between 40-50, the likelihood that the two images are within 0.5 m of each other is extremely high. With fewer features, it becomes extremely hard to tell whether a location is the same or not. In this graph, there were no values of matched features of 60 and higher. A match of 100 features would indicate that the the robot was in exactly the same location.

To evaluate the data association algorithm on real data, a more ambiguous data set was created in which perceptual aliasing was much more of a problem. A set 26 of panoramic

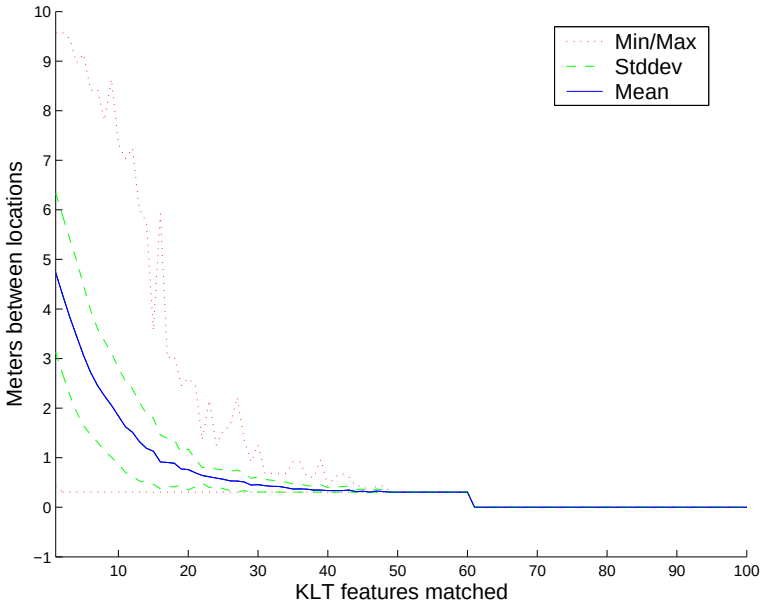


Figure 8.7: Comparison of the number of features tracked vs. the Euclidean distance between locations where the features were obtained.

images were obtained in an office environment shown in Figure 8.8. The dotted lines show the outline of the office and the furniture within it while the solid lines show the path along which the images were taken. Images were taken at 1.07 m increments. The heading of the robot is indicated by the arrows. In this data set, the size of the path traversed was increased relative to the earlier experiments in this chapter. Additionally, the distance between locations where sensor readings were taken was also increased while the number of features tracked by the KLT algorithm was decreased.

A total of 15 features are selected from each image and used for comparison. To take into account the possibility that two panoramic images might correspond to the same location but differ in rotation, the test image was rotated to eight different angles to see which would match the best. Noisy odometry estimates were assigned to each of the paths between images in the training set. As before, the KLT feature tracking algorithm was used to compute the differences between the images.

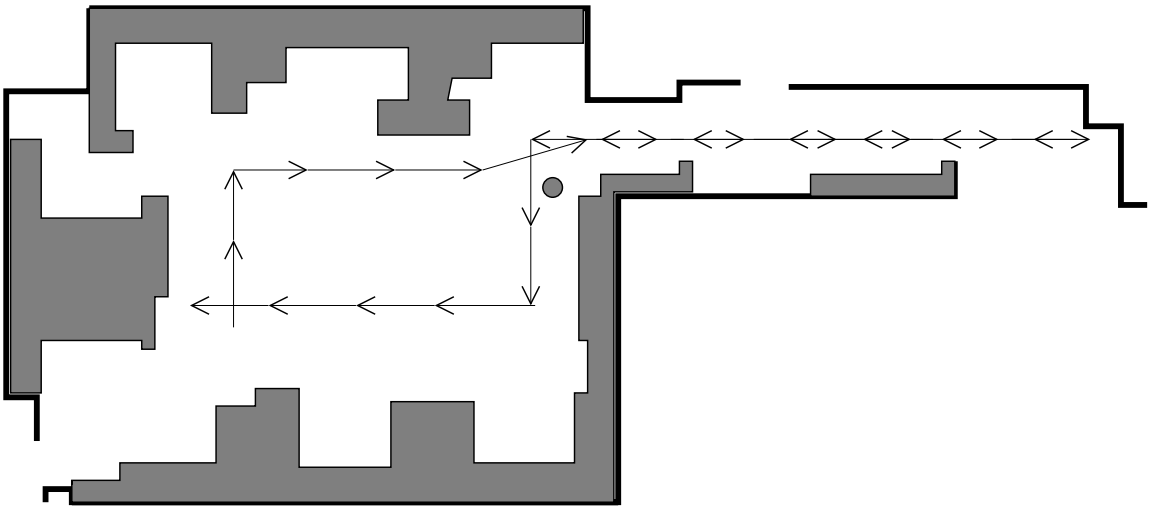


Figure 8.8: The path of the robot through the office environment for the data association algorithm test.

Figure 8.9 illustrates the process of how the algorithm proceeded. The original data reflects the errors in the odometric readings of the robot. The Markov localization step identifies a high probability of the robot's position in nodes at timesteps 10 and 22, as well as at timesteps 14 and 17. These pairs are merged and the spring model was allowed to relax. By this point, the map has obtained a shape which better matches the environment along the corridor that the robot traversed. Since no nodes were matched within the open area of the environment, the starting and ending point of the robot's path do not connect. This example suggests that the localization heuristic will work much better along areas where the robot has traversed large quantities of its own path. Point intersections along the path may not provide enough data to allow the Markov localization step to identify nodes that should be merged.

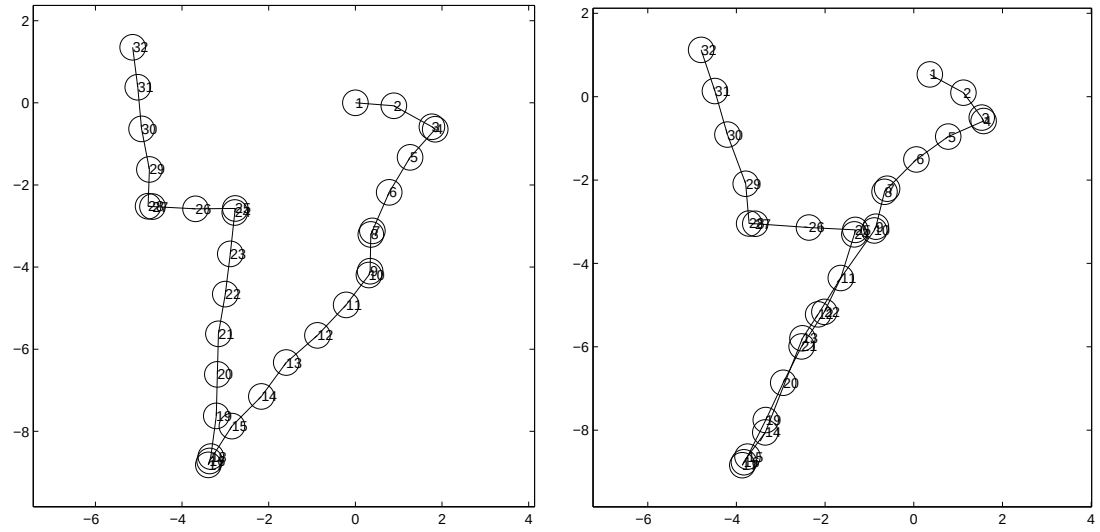


Figure 8.9: Convergence of map after two sets of nodes were selected for merging.

Chapter 9

Summary and Contributions

9.1 Summary

Localization and mapping is a challenge for all mobile robots. Existing methods which work well on large robots do not necessarily scale well as the size of the robot decreases. Sensors typically used in mapping algorithms, such as sonars, laser, and stereo camera pairs, are not appropriate for many miniature robots. Additionally, odometric estimates tend to get worse as the robot becomes smaller since its wheels are likely to slip more as well as being severely affected by distortions in the surface that it travels over.

A method for performing localization and map construction with sensor-poor robots has been proposed in which several maximum likelihood-based estimators, such as batch methods and the recursive Kalman filter, have been formulated to relax the assumption that our sensors return metric distance information to landmarks. To accomplish this, a conventional sensor modality is converted into a"virtual sensor" which is used to determine whether the robot has returned to a location that it has visited before. Using this methodology, landmarks are designated by their sensor signatures and indicate locations the robot has visited. The virtual sensor is both the strength and the weakness of the

method as it allows correlations to be found between locations that the robot has visited, but global metric information, such as orientation, can be difficult to capture. As shown in the experimental results, the local structure of the landmarks can be recreated, but there can be global misalignments in rotation that can be corrected by incorporating additional information such as the known global position of one of the landmarks. The effectiveness of this algorithm has been illustrated on simulated and real world data.

Finally, a method for attempting to address the data association problem has been described. When sensor readings are not distinct and perceptual aliasing is a problem, a Markov localization step is employed to integrate the history of sensor and motion estimates into a discrete probability mass. A greedy search is performed to choose the best matches between the set of all possible matches. Several heuristics are invoked to determine whether or not the match is correct. This approach makes the most sense when all of the data is available to the robot, such for the batch maximum likelihood estimators. Additionally, areas where the robot has crossed significant ports of its own path are much more likely to be recognized as regions that should be matched. If the robot crosses its own path at a single point, this method will have a harder time identifying the locations as being the same.

9.2 Contributions

This thesis contributes to the area of miniature robots which have limited hardware capabilities and presents novel spatial reasoning methods for surveillance tasks. The major challenge with miniature robots is their limited hardware capabilities in terms of processing power, odometry, types of sensors, and communication. These limitations make currently available localization and mapping methods completely unusable for small robots and forces the programming of these robots to be limited to reactive methods. This thesis takes the notion of an appearance-based sensor that does not return either bearing or range measure-

ments to landmarks and combines it in a novel way with various estimators that allow the robot to simultaneously localize and map (known as the SLAM problem). This results in a collection of algorithms that extend the applicability of SLAM methods to a whole new class of robots to which they could not be applied before. The use of SLAM methods has the advantage of providing this broad class of robots with a sound and principled way of localization and mapping. The extension of SLAM to robots with limited odometry and no range or bearing sensors opens up new opportunities for other devices, such as sensors with some mobility used in a sensor network, and allows even larger and more capable robots to avoid using precise range sensors when mapping.

A brief analysis of two case studies of small robot teams is presented in Chapter 2. These two groups of robots illustrate the typical complexity of the robot hardware best suited to use the SLAM methods described in this dissertation. A series of experiments are presented that show the utility and limitations of purely reactive robot control methods. These results serve as the primary motivating factor for this research.

Specific contributions include:

- The notion of an appearance-based sensor for SLAM which allows a robot to localize and reconstruct a path estimate without the need for range or bearing information to environmental landmarks. A virtual sensor is assumed which associates purely qualitative measurements of landmarks to robot positions. These measurements are stored as abstract "signatures" that correspond to specific (x, y) positions along the path of the robot. This is the primary contribution of this dissertation.

- A description of a physics-inspired spring/mass method for reconstructing the robot's path from the appearance-based sensor data. This method uses an heuristic based on energy minimization in for approximating the maximum-likelihood estimate of the robot's path through the environment.

- The derivation of linearized maximum likelihood estimator designed to operate over the complete set of sensor data (in a batch rather than sequential fashion). This analytical approach differs from the more empirical spring/mass estimator in that the uncertainty in translation and rotation is handled in a consistent fashion. As a result, this method is more robust but requires higher memory and computation requirements.

- The derivation of an iterated extended Kalman filter that takes into account the actual odometric and inferred relative positions of landmarks, and estimates the coordinates along the robot's trajectory where sensor readings were recorded. The Kalman filter is a sequential estimator which processes sensor readings as they are received by the robot, allowing for more instantaneous position information.

- A method for handling the data association problem, where multiple locations in space have location signatures similar enough that they cannot be disambiguated with a direct comparison of the signatures. The robot's sensor and path history is integrated over time to generate a probability distribution over the space of all possible poses. In environments where the robot's path has significant overlaps, this method identify multiple locations that are the most likely to be the same. A rejection criterion based on entropy is also presented.

Experimental results are presented throughout this dissertation both in simulation and using a miniature mobile robot with an omnicamera in an indoor office environment. As it traverses the environment, the robot's path is reconstructed using the estimators developed in this thesis and the results are compared. The results support the hypothesis that these estimators are capable of reducing the error in the robot estimates of its path even when the odometry is very poor and the only sensory information available is in the form of location signatures. Results show that the linearized maximum likelihood estimator produces the

best results. The spring-based maximum likelihood estimator and the Kalman filter are fairly close in estimate quality until the robot's odometric error exceeds a threshold at which point the estimation quality of the Kalman filter decreases significantly.

Chapter 10

Future Research

Several methods have been presented for a single robot to localize itself using an appearance-based sensor model that does not return explicit metric information about features in the environment. These methods have shown their utility in estimating the robot's path by recognizing locations that the robot has visited before and suggest a number of interesting extensions for future work.

10.1 Data Association

The presented data association method will work the best only when the complete set of sensor data is available to the robot before it attempts to correct for odometric errors. This is sufficient for the maximum likelihood estimators which typically operate in batch mode on the data after it has all been obtained. However, the extended Kalman filter has the added benefit of being an anytime algorithm which operates on the data as it receives it and can produce the robot's best estimate at any time. If the data association problem is to be handled in this fashion, then a different kind of data association model must be used.

The multiple hypothesis tracking extension to the EKF [81] is one possible solution to this problem. When the robot encounters a situation in its map where there are two or

more hypothesis that seem equally likely, the state vector can be duplicated where each copy represents a separate hypothesis. Each of these hypotheses is propagated and updated separately, and they are evaluated after each new update for its likelihood. The less likely hypothesis can be pruned from the tracked set until only one hypothesis is left.

10.2 Multi-Robot Mapping

A natural extension of this work is to apply it to the case of multi-robot map building where data from multiple robots must be fused into a single global representation. The major challenge of this problem is that the uncertainties in an individual robot's position will be multiplied by the other robot's positional uncertainty. In order to successfully use the combined sum of all the locally-defined maps as a single unit, the robots must define a global map space in which they will combine their maps.

Multiple robot map building has been addressed in several ways. If the two robots know their relative positions with high accuracy, they can merge the maps that they create through a simple overlaying of their respective maps. This is the approach used in [14, 106]. Other systems make extremely restrictive assumptions about the environment (as all walls are straight and all intersections are perpendicular) and reason explicitly about the error in classifying environmental features, such as in [19]. A method for using the EKF across multiple robots is described in [82]. In this representation, the state vector and covariance matrix is shared between multiple robots such that each robot only has to propagate its own information until two robots need to share an update of information.

10.3 Particle Filter Estimators

Bayesian methods for map construction such as batch MLE and the EKF are only one family of estimators that can be used for map construction. Another class of estimators that would

be interesting to explore are the particle filters [21, 99]. One challenge with particle filters is that while they can represent arbitrary probability distributions, they currently do not have any way of representing the shared information between degrees of freedom in the estimator such as which is stored in the EKF's covariance matrix. A recent attempt to bridge this gap is with an algorithm called FASTSlam [69] where particles represent the joint probability of the individual landmarks and each landmark has an EKF associated with it to track its position. To use particle filters as an estimator for appearance-based mapping, the state of each position visited by the robot would have to be represented by its own set of particles. Some care will need to be taken when merging the density estimates for multiple locations that correspond to the same location such that the joint probability is represented correctly.

10.4 Alternative Sensor Modalities

If the Scout had a non-vision virtual sensor sensor whose returned signatures could be transmitted on a digital frequency, this sensor could be used for localization instead. This would allow more robots to operate in the same video broadcasting range.

All of the experiments in this research have been done using the Scout's cameras. Since Scouts only have analog video transmitters, a workstation must have a dedicated video capture unit with a tuned receiver. This also means that only one robot can transmit on a single frequency at a time to avoid interference in the video signals. Processing of the Scout's video data must be done off-line because of the computational complexity. As described in Chapter 2, when the Scouts use their cameras, access to those video frequencies must be scheduled meaning that some robots must wait for resources from other robots before they are able to operate.

By minimizing the reliance on their cameras for navigation, more robots would be able to navigate in parallel. Another complexity with using video data to localize the robots is that moving objects in the environment will increase the likelihood that features will be

missed when using the KLT algorithm to match images. Additional filtering must be done to ensure that false positives can be avoided. This is not a problem that is relegated only to the Scouts as any robot using a camera will encounter these difficulties.

An alternative sensor modality for localization is to make use of wireless signal strength from their communication links. Most robots make use of some sort of wireless communication system and systems like WiFi Ethernet (802.11b) have the ability to monitor the signal strength from other Ethernet nodes. Due to problems with signal propagation issues in complex and unknown environments, such as multipath reflection and destructive interference with other nodes, there might not be a linear relationship between the strength of the received signal and the distance to the wireless node [2]. However, if the robot's network device can detect the signal strengths to a number of other devices, then this vector can be used as the input to the appearance-based localization mechanisms. The larger the number of nodes in the environment, the more distinctive each location is likely to be. One challenge with this approach is that WiFi signal strength seems to depend on the orientation of the antenna to the receiver [57].

Bibliography

[1] B. D. O. Anderson. *Optimal Filtering*. Prentice-Hall, June 1979.

[2] P. Bahl and V. N. Padmanabhan. Radar: An in-building rf-based user location and tracking system. In *Proceedings of IEEE INFOCOM 2000*, Tel-Aviv, Israel, 2000.

[3] T. Balch and R. C. Arkin. Motor schema-based formation control for multiagent robot teams. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 10–16, 1995.

[4] E. Baumgartner, B. Wilcox, R. Welch, and R. Jones. Mobility performance of a small-body rover. In *International Symposium on Robotics with Applications, World Automation Congress*, May 1998.

[5] C. Bererton, L. Navarro-Serment, R. Grabowski, C. J. Paredis, and P. K. Khosla. Millibots: Small distributed robots for surveillance and mapping. In *Government Microcircuit Applications Conference*, Anaheim, CA, Mar. 2000.

[6] M. Betke and L. Gurvits. Mobile robot localization using landmarks. *IEEE Transactions on Robotics and Automation*, 13(2):251–263, April 1997.

[7] D. L. Boley, E. S. Steinmetz, and K. T. Sutherland. Robot localization from landmarks using recursive total least squares. In *Proceedings of the IEEE International*

Conference on Robotics and Automation, pages 1381–1386, Minneapolis, MN, April 1996. IEEE.

[8] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, 1999.

[9] J. Borenstein, B. Everett, and L. Feng. *Navigating Mobile Robots: Systems and Techniques*. A.K. Peters, Ltd., Wellesley, MA, 1996.

[10] T. J. Broida, S. Chandrashekhar, and R. Chellappa. Recursive 3-D motion estimation from a monocular image sequence. *IEEE Transactions on Aerospace and Electronic Systems*, 26, 1990.

[11] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14–23, Mar. 1986.

[12] W. Burgard, D. Fox, D. Hennig, and T. Schmidt. Estimating the absolute position of a mobile robot using position probability grids. In *Proceedings of the National Conference on Artificial Intelligence*, 1996.

[13] W. Burgard, D. Fox, H. Jans, C. Matenar, and S. Thrun. Sonar-based mapping with mobile robots using EM. In *Proceedings of the 16th International Conference on Machine Learning*, pages 67–76. Morgan Kaufmann, San Francisco, CA, 1999.

[14] W. Burgard, D. Fox, M. Moors, R. Simmons, and S. Thrun. Collaborative multi-robot exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2000. To appear.

[15] B. Cartwright and T. Collet. Landmark learning in bees. *Journal of Comparative Physiology, A*, 151:521–543, 1983.

[16] B. Chemel, E. Mutschler, and H. Schempf. Cyclops: miniature robotic reconnaissance system. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2298–2303, 1999.

[17] K. S. Chong and L. Kleeman. Sonar based map building for a mobile robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1700–1705, 1997.

[18] I. Cox and G. Wilfong, editors. *Autonomous Robot Vehicles*. Springer Verlag, 1990.

[19] R. L. de Màntaras, J. Amat, F. Esteva, M. López, and C. Sierra. Generation of unknown environment maps by cooperative low-cost robots. In *Proceedings of the International Conference on Autonomous Agents*, pages 164–169, Marina Del Rey, California, February 1997.

[20] M. Deans and M. Hebert. Experimental comparison of techniques for localization and mapping using a bearingonly sensor. In *International Conference on Experimental Robotics*, Honolulu Hawaii, Dec. 2000.

[21] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1322–1328, 1999.

[22] F. Dellaert and A. Stroupe. Linear 2D localization and mapping for single and multiple robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, May 2002.

[23] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241, June 2001.

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

[25] F. C. Dyer. Spatial memory and navigation by honeybees on the scale of the foraging range. *Journal of Experimental Psychology*, 199:147–154, 1996.

[26] K. Easton and A. Martinoli. Efficiency and optimization of explicit and implicit communication schemes in collaborative robotics experiments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2795–2800, 2002.

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

[28] S. Engelson and D. McDermott. Error correction in mobile robot map learning. In *ICRA*, pages 2555–2560, Nice, France, May 1992.

[29] D. Fox. *Markov Localization: A Probabilistic Framework for Mobile Robot Localization and Navigation*. PhD thesis, Institute of Computer Science III, University of Bonn, Germany, December 1998.

[30] D. Fox, W. Burgard, H. Kruppa, and S. Thrun. Collaborative multi-robot localization. *Autonomous Robots*, 8, 2000.

[31] D. Fox, W. Burgard, and S. Thrun. Active Markov localization for mobile robots. *Robotics and Autonomous Systems*, 25:195–207, 1998.

[32] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *JAIR*, 11:391–427, 1999.

[33] T. Fukuda, A. Kawamoto, and K. Shimojima. Acquisition of swimming motion by RBF fuzzy neuro with unsupervised learning. In *ALIFE V*, pages 31–37, 1996.

[34] A. Gelb. *Applied Optimal Estimation*. MIT Press, 1994.

[35] D. Goldberg and M. J. Mataric. Interference as a tool for designing and evaluating multi-robot controllers. In *Proceedings of the National Conference on Artificial Intelligence*, July 1997.

[36] M. Golfarelli, D. Maio, and S. Rizzi. Elastic correction of dead-reckoning errors in map building. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1998.

[37] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings on Radar and Signal Processing*, 104:107–113, 1993.

[38] R. Grabowski, L. E. Navarro-Serment, C. J. J. Paredis, and P. K. Khosla. Heterogeneous teams of modular robots for mapping and exploration. *Autonomous Robots*, 8(3):293–308, 2000.

[39] O. M. Group. *The Common Object Request Broker: Architecture and Specification*. Object Management Group, 1998.

[40] J. 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)*, 2000.

[41] V. V. Hafner. Learning places in newly explored environments. In *SAB2000 Proceedings Supplement Book*, Paris, France, 2000. International Society for Adaptive Behavior.

[42] A. Halme, T. Schönberg, and Y. Wang. Motion control of a spherical mobile robot. In *4th International Workshop on Advanced Motion Control*, 1996.

[43] D. Hougen, J. Bonney, J. Budenske, M. Dvorak, M. Gini, D. Krantz, F. Malver, B. Nelson, N. Papanikolopoulos, P. Rybski, S. Stoeter, R. Voyles, and K. Yesin. Reconfigurable robots for distributed robotics. In *Government Microcircuit Applications Conference*, pages 72–75, Anaheim, CA, March 2000.

[44] A. Howard, M. Matarić, and G. Sukhatme. Localization for mobile robot teams using maximum likelihood estimation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, EPFL Switzerland, Sept. 2002.

[45] A. Howard, M. J. Matarić, and G. S. Sukhatme. Relaxation on a mesh: a formalism for generalized localization. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1055–1060, 2001.

[46] D. Huttenlocher, D. Klanderman, and A. Rucklige. Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, September 1993.

[47] M. Inaba, S. Kagami, F. Kanechiro, K. Takeda, O. Tetsushi, and H. Inoue. Vision-based adaptive and interactive behaviors in mechanical animals using the remote-brained approach. *Robotics and Autonomous Systems*, 17:35–52, 1996.

[48] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[49] G. A. Kantor and S. Singh. Preliminary results in range-only localization and mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation*, May 2002.

[50] A. Kick, A. Walker, and R. Fisher. A dynamic model for autonomous vehicle navigation. Department of Artificial Intelligence Research Paper 691, University of Edinburgh, April 1994.

[51] KLT: An implementation of the Kanade-Lucas-Tomasi feature tracker. http://robotics.stanford.edu/~birch/klt/.

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

[53] D. Kortenkamp and T. Weymouth. Topological mapping for mobile robots using a combination of sonar and vision sensing. In *Proceeding of Twelfth National Conference on Artificial Intelligence*, pages 979–984, Menlo Park, July 1994. AAAI, AAAI Press/MIT Press.

[54] B. Kuipers. The spatial semantic hierarchy. Technical report, University of Texas at Austin Artificial Intelligence Lab, 1999. TR AI99-281A.

[55] B. 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.

[56] C. Kunz, T. Willeke, and I. R. Nourbakhsh. Automatic mapping of dynamic office environments. *Autonomous Robots*, 7:131–142, 1999.

[57] A. M. Ladd, K. E. Bekris, G. Marceau, A. Rudys, D. S. Wallach, and L. E. Kavraki. Using wireless Ethernet for localization. In *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, Sept. 2002.

[58] J. J. Leonard and H. F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(3):376–382, 1991.

[59] D. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, pages 1150–7, 1999.

[60] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[61] J. Lue and R. Chen. Sequential monte carlo method for dynamic systems. *Journal of the American Statistical Association*, 93:1031–1044, 1998.

[62] M. Matarić. A distributed model for mobile robot environment-learning and navigation. Master's thesis, MIT, Cambridge, MA, January 1990.

[63] M. Matarić. Issues and approaches in the design of collective autonomous agents. *Robotics and Autonomous Systems*, 16(2-4):321–331, Dec. 1995.

[64] P. S. Maybeck. *Stochastic Models, Estimation and Control*, volume 141–142 of *Mathematics in Science and Engineering*. Academic Press, 1982.

[65] T. P. McDonald and J. W. Pellegrino. Psychological perspectives on spatial cognition. In T. Gärling and R. Golledge, editors, *Behavior and Environment: Psychological and Geographical Approaches*, chapter 3, pages 47–81. Elsevier Science Publishers B.V., 1993.

[66] C. McMillen, K. Stubbs, P. E. Rybski, S. A. Stoeter, M. Gini, and N. Papanikolopoulos. Resource scheduling and load balancing in distributed robotic control systems. In *Proceedings of the International Conference on Intelligent Autonomous Systems*, pages 223–230, Mar. 2002.

[67] R. Möller, D. Labrinos, R. Pfeifer, and R. Wehner. Insect strategies of visual homing in mobile robots. In *Proceedings of the Computer Vision and Mobile Robotics Workshop, CVMR 98*, pages 37–45, Heraklion, Greece, 1998.

[68] F. Mondada, E. Franzi, and P. Ienne. Mobile robot miniaturisation: A tool for investigation in control algorithms. In *Experimental Robotics III, Proceedings of the 3rd International Symposium on Experimental Robotics*, pages 501–513, Kyoto, Japan, Oct. 1993. Springer Verlag, London.

[69] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada, 2002. AAAI.

[70] H. P. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 9(2):61–74, 1988.

[71] J. Neira and J. Tards. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, 17(6):890–897, Dec. 2001.

[72] I. Nourbakhsh, R. Powers, and S. Birchfield. DERVISH an office-navigating robot. *AI Magazine*, 16(2):53–60, 1995.

[73] Omnitech Robotics International, LLC, 2640 South Raritan Circle, Englewood, CO, 80110. *ORIFL190-3: 190 Degree Field of View Fisheye Lens for 1/3" Image Sensor Cameras*, 2002.

[74] E. Parzen. On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 3:1065–1076, 1962.

[75] S. Perkins and G. Hayes. Real-time optical flow based range sensing on mobile robots. In A. Borkowski and J. Crowley, editors, *Proceedings of the International Symposium on Intelligent Robotic Systems*, pages 303–310, LIFIA-IMAG, Grenoble, July 1994.

[76] D. Pierce and B. J. Kuipers. Map learning with uninterptreted sensors and effectors. *Artificial Intelligence*, 92:169–227, 1997.

[77] B. Pinette. *Image-based Navigation through Large scale Environments.* PhD thesis, University of Massachusetts, Computer Science Department, 1994.

[78] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C : The Art of Scientific Computing.* Cambridge University Press, 2nd edition, 1993.

[79] http://ic-www.arc.nasa.gov/ic/psa/.

[80] I. Rekleitis, G. Dudek, and E. Milios. Multi-robot exploration of an unknown environment, efficiently reducing the odometry error. In *International Joint Conference in Artificial Intelligence*, pages 1340–1345, Nagoya, Japan, August 1997.

[81] S. I. Roumeliotis and G. A. Bekey. Bayesian estimation and kalman filtering: A unified framework for mobile robot localization. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2985–2992, San Francisco, CA, April 2000.

[82] S. I. Roumeliotis and G. A. Bekey. Collective localization: A distributed kalman filter approach to localization of groups of mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2958–2965, San Francisco, California, Apr. 2000.

[83] P. E. Rybski, I. Burt, T. Dahlin, M. Gini, D. F. Hougen, D. G. Krantz, F. Nageotte, N. Papanikolopoulos, and S. A. Stoeter. System architecture for versatile autonomous and teleoperated control of multiple miniature robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Seoul, Korea, May 2001.

[84] P. E. Rybski, A. Larson, M. Lindahl, and M. Gini. Performance evaluation of multiple robots in a search and retrieval task. In *Workshop on Artificial Intelligence and Manufacturing*, pages 153–160, Albuquerque, NM, Aug. 1998.

[85] P. E. Rybski, A. Larson, A. Schoolcraft, S. Osentoski, and M. Gini. Evaluation of control strategies for multi-robot search and retrieval. In *Proceedings of The 7th International Conference on Intelligent Autonomous Systems (IAS-7)*, pages 281–288, Mar. 2002.

[86] P. E. Rybski, S. A. Stoeter, M. D. Erickson, M. Gini, D. F. Hougen, and N. Papanikolopoulos. A team of robotic agents for surveillance. In *Proceedings of the International Conference on Autonomous Agents*, pages 9–16, Barcelona, Spain, June 2000.

[87] P. E. Rybski, S. A. Stoeter, M. Gini, D. F. Hougen, and N. Papanikolopoulos. Effects of limited bandwidth communications channels on the control of multiple robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 369–374, Hawaii, USA, Oct. 2001.

[88] P. E. Rybski, S. A. Stoeter, M. Gini, D. F. Hougen, and N. Papanikolopoulos. Performance of a distributed robotic system using shared communications channels. *IEEE Transactions on Robotics and Automation*, 22(5):713–727, Oct. 2002.

[89] 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 IEEE International Conference on Robotics and Automation*, 2003.

[90] H. Shatkay and L. Kaelbling. Learning topological maps with weak local odometric information. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 920–927, San Mateo, CA, 1997. Morgan Kaufmann.

[91] R. Simmons and S. Koenig. Probabilistic robot navigation in partially observable environments. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1080–1087, San mateo, CA, 1995. Morgan Kaufmann.

[92] A. Smith and A. E. Gelfand. Bayesian statistics without tears: A sampling-resampling perspective. *American Statistician*, 46(2):84–88, 1992.

[93] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In I. J. Cox and G. T. Wilfong, editors, *Autonomous Robot Vehicles*, pages 167–193. Springer-Verlag, 1990.

[94] S. A. Stoeter, P. E. Rybski, K. N. Stubbs, C. P. McMillen, M. Gini, D. F. Hougen, and N. Papanikolopoulos. A robot team for surveillance tasks: Design and architecture. *Robotics and Autonomous Systems*, 40(2-3):173–183, Sept. 2002.

[95] S. Thrun, M. Bennewitz, W. Burgard, A. Cremers, F. Dellaert, D. Fox, D. Haehnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. MINERVA: A second generation mobile tour-guide robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1999–2005, 1999.

[96] S. Thrun, A. Bücken, W. Burgard, D. Fox, T. Fröhlinghaus, D. Hennig, T. Hofmann, M. Krell, and T. Schmidt. Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R. P. Bonasso, and R. Murphy, editors, *AI-based Mobile Robots: Case studies of successful robot systems*. MIT Press, 1998.

[97] S. Thrun, W. Burgard, and D. Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, 31:29–53, 1998.

[98] S. Thrun, W. Burgard, and D. Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2000.

[99] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 101:99–141, 2000.

[100] S. Thrun, J.-S. Gutmann, D. Fox, W. Burgard, and B. Kuipers. Integrating topological and metric maps for mobile robot navigation: A statistical approach. In *Proceedings of the National Conference on Artificial Intelligence*, pages 989–995, 1998.

[101] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical report, School of Computer Science, Carnegie Mellon University, April 1991.

[102] I. Ulrich and I. Nourbakhsh. Appearance-based place recognition for topological localization. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1023–1029, San Francisco, CA, April 2000.

[103] S. D. Whitehead and D. H. Ballard. Active perception and reinforcement learning. In *Proceedings, Seventh International Conference on Machine Learning*, Austin, TX, 1990.

[104] E. Wolfart, R. B. Fisher, and A. Walker. Position refinement for a navigating robot using motion information based on honey bee strategies. In *Proceedings of the Symposium on Intelligent Robotic Systems (SIRS95), Pisa*, pages 257–264, July 1995.

[105] A. S. Wu, A. C. Schultz, and A. Agah. Evolving control for distributed micro air vehicles. In *Proceedings of the IEEE 1999 Internationall Symposium on Computational Intelligence in Robotics and Automation*, Monterey, CA, Nov. 1999.

[106] B. Yamauchi. Frontier-based exploration using multiple robots. In *Proceedings of the International Conference on Autonomous Agents*, pages 47–53, 1999.

[107] B. Yamauchi and R. Beer. Spatial learning for navigation in dynamic environments. In *IEEE Transactions on Systems, Man, and Cybernetics*, volume 26, 1996.

Appendix A

Derivation of 1D Linear Estimators

A.1 Derivation of the 1D Kalman Filter

As a simple example, a Kalman Filter is derived for estimating the position of the degenerate case of a robot moving only in 1D (no orientation).

A.1.1 Propagation

In the 1D case, the robot has a single dimension of motion and thus will have a velocity $\dot{x} = v$ that is the same in both robot-centric ${}^{\mathcal{R}}x$ and global ${}^{G}x$ coordinates.

At every discrete timestep k, the true position of the robot is defined as the previous position plus some displacement based on its current velocity:

$$x(k+1) = x(k) + v(k)\Delta t \tag{A.1.1}$$

where $v(k)$ is the 1D velocity of the robot at the (discrete) time interval k.

The estimated position from the encoders is defined as:

$$\hat{x}(k+1) = \hat{x}(k) + v_m(k)\Delta t \tag{A.1.2}$$

where $v_m = v + \omega_v$ and ω_v is the error (assumed to be from a white zero-mean Gaussian distribution) in the robot's odometric estimate.

The derivation of the position error is as follows:

$$\begin{aligned}
\widetilde{x}(k+1) &= x(k+1) - \hat{x}(k+1) \\
&= x(k) + v(k)\Delta t - \hat{x}(k) - (v(k) + w_v)\Delta t \\
&= x(k) + v(k)\Delta t - \hat{x}(k) - v(k)\Delta t - w_v\Delta t \\
&= x(k) - \hat{x} - w_v\Delta t \\
&= \widetilde{x}(k) - w_v\Delta t
\end{aligned} \tag{A.1.3}$$

This can also be represented as:

$$\widetilde{x}(k+1) = F\widetilde{x}(k) + Gw_v \tag{A.1.4}$$

where $F = 1$ and $G = -\Delta t$.

From this, the covariance estimate at each timestep can be derived as:

$$\begin{aligned} P_{k+1/k} &= E[\widetilde{x}(k+1)\widetilde{x}^T(k+1)] \\ &= E[(F\widetilde{x}(k)+Gw_v)(F\widetilde{x}(k)+Gw_v)^T \\ &= F(k)E[\widetilde{x}(k)\widetilde{x}^T(k)]F^T(k)+2F(k)E[\widetilde{x}(k)w(k)]G(k)+G(k)E[w(k)w(k)^T] \\ &= F(k)E[\widetilde{x}(k)\widetilde{x}^T(k)]F^T(k)+G(k)E[w(k)w(k)^T]G^T(k) \\ &= F(k)P_{k/k}F^T(k)+G(k)Q_kG^T(k) \end{aligned} \tag{A.1.5}$$

where $Q_k = \sigma_v^2 = E\{\omega_k^2\}$.

A.1.2 Update

When deriving the update part of the system, the robot is assumed to have a "virtual" sensor which returns true or false if the robot is at a position that it was in before. Thus, if z is the measurement of the distance to the landmark, then $z = 0$ in all cases. Formally, the measurement and the expectation of the measurement are defined as:

$$z = {}^R x_L = {}^G x_L - {}^G x_R + n_z \tag{A.1.6}$$

$$\hat{z} = {}^R \hat{x}_L = {}^G \hat{x}_L - {}^G \hat{x}_R \tag{A.1.7}$$

The measurement error can be defined as:

$$\begin{aligned}
\widetilde{z} &= z - \hat{z} \\
&= {}^{R}x_L + n_z - {}^{R}\hat{x}_L \\
&= {}^{G}x_L - {}^{G}x_R + n_z - {}^{G}\hat{x}_L + {}^{G}\hat{x}_R \\
&= {}^{G}x_L - {}^{G}\hat{x}_L - {}^{G}x_R + n_z + {}^{G}\hat{x}_R + n_z \\
&= {}^{G}\widetilde{x}_L - {}^{G}\widetilde{x}_R + n_z
\end{aligned} \tag{A.1.8}$$

or, expressed in matrix form:

$$\widetilde{z} = \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_R \\ \hat{x}_L \end{bmatrix} + n_z \tag{A.1.9}$$

$$= H\hat{X} + n_z \tag{A.1.10}$$

where $H = \begin{bmatrix} -1 & 1 \end{bmatrix}$.

A.2 Derivation of the 1D Maximum Likelihood Estimator

As a simple example, a simple linear estimator is derived for a robot moving in 1D (on a single line), as shown in Figure A.1.

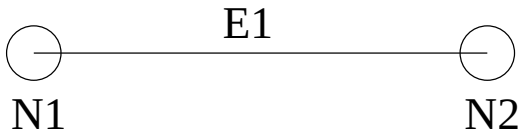


Figure A.1: Single-dimensional maximum likelihood example. The robot moves back and forth between position $N1$ and $N2$. At each position, the robot takes a reading from its sensors and recognizes that location as a particular landmark. The estimate of the robot's travel distance is uncertain and errors accumulate as the robot continues to travel between the two nodes.

In this example, the purpose of the estimator is to compute, from the collection of all its sensor readings, the best estimate for the positions of the landmarks. Let y_i be the estimate for the distance that the robot computes as it travels between the landmarks. Let x_j be the collection of places in space where the robot took a landmark reading, where a single location in space might have multiple such readings. Initially, the robot has no idea where the landmarks are and must explore the environment to locate them. When the robot first discovers a landmark and takes a sensor reading, it assumes that the position of the landmark is perfect since it has no previous reading to compare the data against.

The robot is assumed to have an appearance-based sensor of the kind described earlier in this thesis. With this sensor, the robot can determine whether it is at a particular location. The returned value from this sensor is z_j. When the robot returns to a previously-explored landmark, it stores the inferred distance to the landmark z_k, and the distance it traveled from the previous landmark. These readings are used in the cost function that is minimized.

To illustrate how this process works, Figure A.2 shows an example set of data taken from the robot as it travels back and forth between the two landmarks.

The set of all sensor data can be described as pairs of sensor readings (y_i, z_j) and expected values for those quantities, given the current estimate of the state vector ($\hat{y_i}$, $\hat{z_j}$). The estimator attempts to estimate the correct values for the robot positions $x_i \in X$ such that the residuals ($y_i - \hat{y_i}$ and $z_i - \hat{z_i}$) are minimized.

The cost function χ^2 can be written as:

$$\chi^2 = \sum_{i=1}^{n-1}(y_i - \hat{y_i})P_i^{-1}(y_i - \hat{y_i}) + \sum_{i=1}^{n}(z_i - \hat{z_i})R_i^{-1}(z_i - \hat{z_i})$$

where the cost is the square of the residual of the measured vs. the estimated values multiplied by the covariance for that measurement. The measurement covariances are P_i for the odometric estimates and R_i for the sensor measurements. In the single-dimensional case, sensor measurement functions are linear:

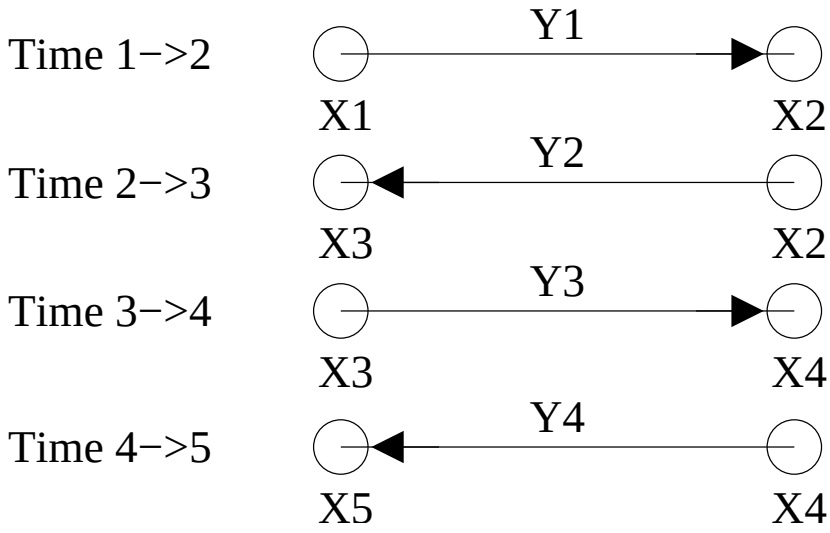


Figure A.2: Visualization of the data history returned from the 1D robot's travels. Every time the robot travels between the two landmarks, it stores the length of the path as y_i. Each time it visits a landmark, it stores that position as x_j.

$$\begin{aligned} \hat{y}_i &= h_y(\hat{X}) = \hat{x}_i - \hat{x}_{i-1} \\ \hat{z}_i &= h_z(\hat{X}) = \hat{x}_i - \hat{x}_j \end{aligned}$$

Thus, the individual terms of the cost function are as follows:

$$
\begin{aligned}
y_1 &= x_2 - x_1 + w_1 \\
\hat{y}_1 &= \hat{x}_2 - \hat{x}_1 \\
y_2 &= x_3 - x_2 + w_2 \\
\hat{y}_2 &= \hat{x}_3 - \hat{x}_2 \\
y_3 &= x_4 - x_3 + w_3 \\
\hat{y}_3 &= \hat{x}_4 - \hat{x}_3 \\
y_4 &= x_5 - x_4 + w_4 \\
\hat{y}_4 &= \hat{x}_5 - \hat{x}_4
\end{aligned}
\qquad
\begin{aligned}
z_1 &= x_1 - x_1 + n_1 \\
\hat{z}_1 &= \hat{x}_1 - \hat{x}_1 \\
z_2 &= x_2 - x_2 + n_2 \\
\hat{z}_2 &= \hat{x}_2 - \hat{x}_2 \\
z_3 &= x_3 - x_1 + n_3 \\
\hat{z}_3 &= \hat{x}_3 - \hat{x}_1 \\
z_4 &= x_4 - x_2 + n_4 \\
\hat{z}_4 &= \hat{x}_4 - \hat{x}_2 \\
z_5 &= x_5 - x_1 + n_4 \\
\hat{z}_5 &= \hat{x}_5 - \hat{x}_1
\end{aligned}
$$

In the previous equations, each noise term w_i and n_i is an independent zero-mean Gaussian. In order to solve this, the partial derivatives with respect to each of the variables x_1, x_2, x_3, x_4, and x_5 must be taken and the value is set to zero.

$$
\begin{aligned}
\frac{\partial \chi^2}{\partial \hat{x}_1} &= 2(y_1 - \hat{x}_2 + \hat{x}_1)P_1^{-1} + 2(z_3 - \hat{x}_3 + \hat{x}_1)R_3^{-1} + 2(z_5 - \hat{x}_5 + \hat{x}_1)R_5^{-1} \\
\frac{\partial \chi^2}{\partial \hat{x}_2} &= -2(y_1 - \hat{x}_2 + \hat{x}_1)P_1^{-1} + 2(y_2 - \hat{x}_3 + \hat{x}_2)P_2^{-1} + 2(z_4 - \hat{x}_4 + \hat{x}_2)R_4^{-1} \\
\frac{\partial \chi^2}{\partial \hat{x}_3} &= -2(y_2 - \hat{x}_3 + \hat{x}_2)P_2^{-1} + 2(y_3 - \hat{x}_4 + \hat{x}_3)P_3^{-1} + -2(z_3 - \hat{x}_3 + \hat{x}_1)R_3^{-1} \\
\frac{\partial \chi^2}{\partial \hat{x}_4} &= -2(y_3 - \hat{x}_4 + \hat{x}_3)P_3^{-1} + 2(y_4 - \hat{x}_5 + \hat{x}_4)P_4^{-1} + -2(z_4 - \hat{x}_4 + \hat{x}_2)R_4^{-1} \\
\frac{\partial \chi^2}{\partial \hat{x}_5} &= -2(y_4 - \hat{x}_5 + \hat{x}_4)P_4^{-1} + -2(z_5 - \hat{x}_5 + \hat{x}_1)R_5^{-1}
\end{aligned}
$$

The terms are rearranged so that the equation takes the form $AX - b = 0$, as shown below.

$$\begin{bmatrix} P_1^{-1} + R_3^{-1} + R_5^{-1} & -P_1^{-1} & -R_3^{-1} & 0 & -R_5^{-1} \\ -P_1^{-1} & P_2^{-1} + P_1^{-1} + R_4^{-1} & -P_2^{-1} & -R_4^{-1} & 0 \\ -R_3^{-1} & -P_2^{-1} & P_2^{-1} + P_3^{-1} + R_3^{-1} & -P_3^{-1} & 0 \\ 0 & -R_4^{-1} & -P_3^{-1} & P_3^{-1} + P_4^{-1} + R_4^{-1} & -P_4^{-1} \\ -R_5^{-1} & 0 & 0 & -P_4^{-1} & P_4^{-1} + R_5^{-1} \end{bmatrix} \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \\ \hat{x}_4 \\ \hat{x}_5 \end{bmatrix} - \begin{bmatrix} y_1 P_1^{-1} + z_3 R_3^{-1} + z_5^{-1} R_5^{-1} \\ -y_1 P_1^{-1} + z_2 P_2^{-1} + z_4^{-1} R_4^{-1} \\ -y_2 P_2^{-1} + y_3 P_3^{-1} + z_3^{-1} R_3^{-1} \\ -y_3 P_3^{-1} + y_4 P_4^{-1} + z_4^{-1} R_4^{-1} \\ -y_4 P_4^{-1} + z_5^{-1} R_5^{-1} \end{bmatrix} = 0$$

Solving this equation is simply a matter of adding b to both sides of the equation and then pre-multiplying both sides by the inverse of A. However, A does not have a unique inverse (it is rank deficient) given that there is an infinite number of solutions for the vector X which will satisfy the system of equations. The solution is to provide a hard-coded value to x_1 (such as 0 for example) and modify the above equations accordingly. Since the robot is constructing the map from its own coordinate system, it can choose to place the origin anywhere and thus be able to properly constrain the system of equations to a unique solution.

Appendix B

Markov Localization

B.1 Derivation of the Dual-pass Markov Localization Step

The following derivation was originally reported in [97] and is included here for the sake of completeness.

Let the sensor stream S from the robot be defined as:

$$S = \left\{ o^{(1)}, u^{(1)}, o^{(2)}, u^{(2)}, \ldots, o^{(n)}, u^{(n)} \right\} \tag{B.1.1}$$

The sensor stream S consists of sensor observations that the robot makes in its environment as well as a movements that the robot makes. Each $o^{(t)}$ is a sensor observation made at time t. Each $u^{(t)}$ is a motion command that was made between time t and $t+1$.

Each robot has a motion model which is defined for it, consisting of:

$$P(L'|u, L) \tag{B.1.2}$$

This refers to the probability that the robot's pose is L' after executing action u at previous pose L.

Each sensor has a perceptual model, defined as follows:

$$P(o|L, M) \tag{B.1.3}$$

which is the likelihood of observing o in situations where both the world map M and the robot's position L are known.

The estimate of the robot's position given all of the data and the map can be described as:

$$P\left(L^{(t)}|S, M\right) = P\left(L^{(t)}|o^{(1)}, \ldots, o^{(t)}, u^{(t)}, \ldots, o^{(T)}, M\right) \tag{B.1.4}$$

Applying Bayes' rule, this can be re-written as:

$$P\left(L^{(t)}|S, M\right) = \eta_1 P\left(o^{(1)}, \ldots, o^{(t)}|L^{(t)}, u^{(t)}, \ldots, o^{(T)}, M\right) P\left(L^{(t)}|u^{(t)}, \ldots, o^{(T)}, M\right) \tag{B.1.5}$$

An additional assumption can be made regarding the independence of past data from future data. Applying this first-order Markov assumption:

$$P\left(L^{(t)}|S, M\right) = \eta_1 P\left(o^{(1)}, \ldots, o^{(t)}|L^{(t)}, M\right) P\left(L^{(t)}|u^{(t)}, \ldots, o^{(T)}, M\right) \tag{B.1.6}$$

A second application of Bayes' rule yields:

$$P\left(L^{(t)}|S, M\right) = \eta_2 P\left(L^{(t)}|o^{(1)}, \ldots, o^{(t)}, M\right) P\left(o^{(1)}, \ldots, o^{(t)}|M\right) P\left(L^{(t)}|u^{(t)}, \ldots, o^{(T)}, M\right) \tag{B.1.7}$$

The $P\left(o^{(1)}, \ldots, o^{(t)}|M\right)$ term can be absorbed into the normalizing constant, yielding:

$$P\left(L^{(t)}|S,M\right) = \eta_3 P\left(L^{(t)}|o^{(1)},\ldots,o^{(t)},M\right) P\left(L^{(t)}|u^{(t)},\ldots,o^{(T)},M\right) \tag{B.1.8}$$

Following the literature on hidden Markov models, these two right-hand terms can be defined as $\alpha^{(t)}$ and $\beta^{(t)}$.

Computation of the α Values

Initially, the robot is assumed to be in a known position, the origin of its coordinate system. This is represented as a Dirac delta function, where

$$\alpha^{(1)} = P\left(L^{(1)}|o^{(1)},M\right) = \begin{cases} 1 & \text{if } L^{(1)} = (0,0), \\ 0 & \text{otherwise.} \end{cases} \tag{B.1.9}$$

Then, for each value of $t = \{2,\ldots,n\}$,

$$\alpha^{(t)} = P\left(L^{(t)}|o^{(1)},\ldots,o^{(t)},M\right) \tag{B.1.10}$$

By Bayes' rule:

$$\alpha^{(t)} = \eta P\left(o^{(t)}|L^{(t)},\ldots,o^{(t-1)},M\right) P\left(L^{(t)}|o^{(1)},\ldots,u^{(t-1)},M\right) \tag{B.1.11}$$

Then, due to the first-order Markov independence assumption about the perceptual model, the first term can be re-written as:

$$\alpha^{(t)} = \eta P\left(o^{(t)}|L^{(t)},M\right) P\left(L^{(t)}|o^{(1)},\ldots,u^{(t-1)},M\right) \tag{B.1.12}$$

The second term can be broken into two components, the first of which is the motion model:

$$\alpha^{(t)} = \eta P\left(o^{(t)}|L^{(t)}, M\right) \int P\left(L^{(t)}|u^{(t-1)}, L^{(t-1)}\right) P\left(L^{(t-1)}|o^{(1)}, \ldots, o^{(t-1)}, M\right) dL^{(t-1)} \tag{B.1.13}$$

Finally, the third term is just the definition of $\alpha^{(t-1)}$. Thus, the equation can be written as:

$$\alpha^{(t)} = \eta P\left(o^{(t)}|L^{(t)}, M\right) \int P\left(L^{(t)}|u^{(t-1)}, L^{(t-1)}\right) \alpha^{(t-1)} dL^{(t-1)} \tag{B.1.14}$$

Computation of the β Values

$$\beta^{(t)} = P\left(L^{(t)}|u^{(t)}, \ldots, o^{(T)}, M\right) \tag{B.1.15}$$

This can be broken into two parts, the first being the robot's motion model:

$$\begin{aligned}\beta^{(t)} &= \int P\left(L^{(t)}|u^{(t)}, L^{(t+1)}\right) P\left(L^{(t+1)}|o^{(t+1)}, \ldots, o^{(T)}, M\right) dL^{(t+1)} \\ &= \int P\left(L^{(t+1)}|u^{(t)}, L^{(t)}\right) P\left(L^{(t+1)}|o^{(t+1)}, \ldots, o^{(T)}, M\right) dL^{(t+1)}\end{aligned} \tag{B.1.16}$$

The latter term can be segmented into two parts.

$$\begin{aligned}\beta^{(t)} = \int & P\left(L^{(t+1)}|u^{(t)}, L^{(t)}\right) \eta P\left(o^{(t+1)}, u^{(t+1)}, \ldots, o^{(T)}, M\right) \\ & P\left(L^{(t+1)}|o^{(t+1)}, \ldots, o^{(T)}, M\right) dL^{(t+1)}\end{aligned} \tag{B.1.17}$$

The middle term can be collapsed into the perceptual model, given the first order Markov

assumption, and the third term is the definition of $\beta^{(t+1)}$, yielding:

$$\beta^{(t)} = \eta \int P\left(L^{(t+1)}|u^{(t)}, L^{(t)}\right) P\left(o^{(t+1)}|L^{(t+1)}, M\right) \beta^{(t+1)} dL^{(t+1)} \tag{B.1.18}$$

$\beta^{(T)}$ represents the final probability of the robot's position. It is uniformly distributed as it does not depend on data.