INFERRING ENVIRONMENTAL REPRESENTATIONS THROUGH LIMITED SENSORY DATA WITH APPLICATIONS TO SENSOR NETWORK SELF-CALIBRATION

Dimitrios Paul Marinakis

School of Computer Science
McGill University, Montréal

November 2008

A Thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfilment of the requirements for the degree of
Doctor of Philosophy

ABSTRACT

This thesis addresses the problem of using distributed sensing for automatically inferring a representation of the environment, *i.e.* a map, that can be useful for the self-calibration of intelligence systems, such as sensor networks. The information recovered by such a process allows typical applications such as data collection and navigation to proceed without labour intensive input from a human technician. Simplifying the deployment of large scale sensor networks and other intelligent systems will effectively reduce their cost and improve their widespread availability and hence aid their practical application to tasks such as the monitoring of carbon emissions and greenhouse gases.

In our research we focus on algorithms and techniques for recovering two types of information from the immediate environment: topology information that indicates physical connectivity between regions of interest from the point of view of a navigating agent; and a probability distribution function (PDF) describing the position of components of the intelligent system. We consider situations where data is collected from systems that comprise of: a number of stationary network components; stationary network components augmented with a mobile robot; or a mobile robot only. Our approaches are, for the most part, based on statistical methods that employ stochastic sampling techniques to provide approximate solutions to problems for which computing the optimal or exact solution is intractable. Numerical simulations and experiments conducted on hardware suggest that this research has promising real world applications in the area of sensor network self-configuration.

RÉSUMÉ

Ce thèse s'adresse au problème de l'emploi de la détection dispersée pour déduire automatiquement une représentation de l'environnement, c'est-à-dire une carte, qui peut servir dans l'autocalibrage des systèmes intelligents tels que les réseaux des capteurs. L'information récupérée par un tel processus permet aux applications typiques telle que la collecte des données et la navigation de continuer sans une contribution de main d'oeuvre de la part d'un technicien humain. Simplifier la répartition en grand des réseaux de capteurs et d'autres systèmes intelligents réduira effectivement leur coût et améliora leur disponibilité répandue, donc il facilitera leur application pratique aux tâches comme le contrôle des émissions de carbone et les gaz à effet de serre.

Dans nos recherches nous nous concentrons sur les algorithmes et les techniques pour récupérer deux types d'information de l'environnement immédiat : l'information topologique qui indique la connectivité physique entre les régions d'intérêt du point de vue d'un agent navigateur; et une fonction de dispersion de probabilité (PDF) qui décrit la position des élément du système intelligent. Nous considérons les situation où les données se recueillent des systèmes composés de: plusieurs éléments fixes du réseau; des éléments fixes du réseau augmentés d'un robot mobile; un robot mobile seulement. Nos approches sont, pour la plupart, fondées sur des méthodes statistiques qui emploient des techniques stochastiques d'échantillonnage pour fournir des solutions approximatives aux problèmes dont le calcul d'une solution exacte ou optimale reste réfractaire. Les simulations numériques et les expériences exécutées au matériel suggèrent que ces recherches promettent des applications actuelles et pratiques dans le domaine d'autocalibrage des réseaux de capteurs.

ACKNOWLEDGEMENTS

There are many people who helped and supported me during my graduate work towards this thesis. First, I must thank my supervisor Professor Gregory Dudek for his insights, ideas, optimism and perhaps most importantly for his infectious enthusiasm. Additionally, I would like to thank my committee members Doina Precup and Joelle Pineau for their ideas, help and guidance.

I must also thank several colleagues who, in addition to my supervisor and committee members, have made intellectual contributions to the research presented in this thesis. David Fleet is one of the participants of the research that led ultimately to the topology inference work presented in Chapter 3. Philippe Giguère made significant contributions to the work presented in Chapter 4 and Ketan Dalal should also be acknowledged for his help on this Chapter. David Meger and Ioannis Rekleitis have also contributed significantly to the work presented in Chapter 7.

Also deserving of thanks are Matt Garden, Eric Bourque, Junaed Sattar, Saul Simhon, Paul Di Marco, Luz Abril Torres-Méndez, Dan Burfoot, and others of the Mobile Robotics lab, along with Cynthia Davison, Jan Binder, Marlene Gray and others of the CIM administration and SOCS system staff. These people all made various contributions of technical help, administrative help, equipment loans, proof-reading, experimental assistance, trajectory data, ideas, and good company. I should also acknowledge my primary funding source, the Natural Sciences and Engineering Research Council of Canada.

Finally, I would like thank my family for their continual support and confidence, and especially Michelle for the photos, proof reading, valuable assistance with experiments, patience, and encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

1. Inferring an Environmental Representation through Limited Sensory Data

This thesis presents methods of inferring a representation of the environment given limited sensory data. That is, it addresses the issue of forming a *map* given observations collected from an intelligent system made up of one or more spatially distributed sensing components. Understanding the spatial relationships or correspondences between either the individual components of such a system or key landmarks in the environment allows typical applications such as data collection or navigation. Indeed almost any application of an intelligent system made up of physical components, sensors and actuators requires at least some rudimentary notion of its own embedding in order to serve its purpose. For example, consider trying to usefully apply temperature and humidity observations collected from a sensor network deployed in a grape field without knowing anything regarding the locations of the individual components from which the measurements were collected. We are interested in automating all or portions of the process of constructing this requisite spatial information, and we are interested in doing so with observational data of the quality that could be affordably obtained from a typical low-powered, resource-poor, sensor platform. Ideally, we are able to utilize data opportunistically collected by the sensors already provided for the purpose of the final application.

The primary focus of this research is on the topic of *sensor network self-calibration*. We use the term sensor network self-calibration to refer to the process of automatically

obtaining spatial information regarding the network and its environment necessary for the system to carry out its assigned tasks. During our investigations, we have considered the self-calibration problem for systems made up only of stationary network components, and also for systems in which additional information is gathered by a mobile robot. Additionally, one chapter considers the system calibration problem for a single mobile robot; albeit with extremely limited sensory capabilities.

We begin this introductory chapter by describing the issue of sensor network calibration which is the primary motivator for our research. We then we describe some specifics regarding the problem we are interested in solving and provide a high level overview of our approach. This is followed by a discussion of some motivation for the work. Finally, we end the chapter with some comments on originality and provide an outline for the entire thesis.

2. Sensor Network Self-Calibration

Advances in computing hardware are making the deployment of networks of sensing and computing devices practical for a variety of control and information gathering purposes. This recent technology, termed *sensor networks*, becomes increasingly relevant as continuous improvements are made to the processing power, sensing ability, and wireless communication range and bandwidth available on various computing platforms. The components of such networks can include emplaced motion sensors, emplaced cameras, robots, or even cell phones.

As stated earlier, we define *sensor network self-calibration* as the process of automatically obtaining the spatial information necessary for the network to carry out its assigned task. Our definition of this term is slightly broader than the phrase *sensor network self-localization* which is commonly used in the literature to refer to the process of determining the location and orientation of each sensor after deployment and does not encompass topological correspondences between the sensors. An even broader term in common use which we will employ later in this thesis is *sensor network self-configuration* which includes other automated network tasks such as setting up and maintaining communications.

For many sensor network applications, the self-calibration problem as we have defined it is seen as a critical issue. For example, research by Correal and Patwari [**21**] identify position location as a key application enabler for sensor networks. Similar claims have been

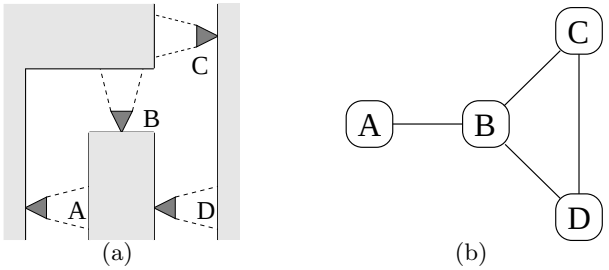


FIGURE 1.1. Example of a sensor network layout (a) and corresponding topology (b) where the labels A through D denote sensing nodes.

put forth by a number of other sensor network researchers including Akyildiz *et al.* [**1**], Bulusu *et al.* [**13**], and Teller *et al.* [**125**].

Once calibrated, the network should also have the capability of adjusting for dynamic changes both in the environment and in the network. The ability to self-calibrate becomes especially relevant if the network is large and sensors are to be deployed in an ad-hoc manner; *i.e.* distributed through some technique that does not return the exact locations of the sensors.

In our research we address several aspects of this problem of self-calibrating a sensor network. The ultimate goal of this portion of the work is to infer a representation of the sensor network and its relationship with the surrounding environment. Specifically, we consider methods of recovering the relative metric locations of the sensor nodes in a network and their physical connectivity from the point of view of an agent navigating the environment; *i.e.* a map of the network embedded region that contains both metric spatial information and topological connectivity information.

Recovering the relative positions of the components of a network has been well explored in sensor network research, however, there has been little work done in augmenting this metric spatial representation with a topological description. The problem of inferring the topology of a sensor network is closely related to that of metric self-localization. In self-localization, the goal is to recover the relative locations of the nodes independent of the layout of the space in which the network is embedded. Topology inference as we define it, however, must take into account the spatial constraints of the environment since these constraints determine the inter-node connectivity parameters (Figure 1.1). These two tasks can complement one other.

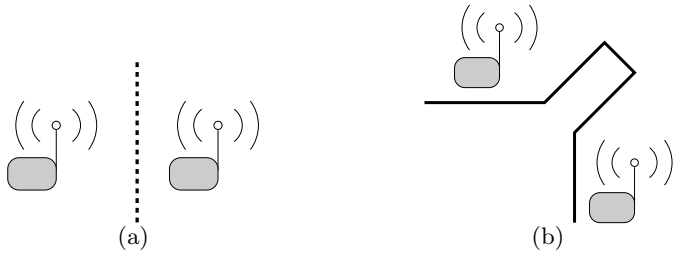


FIGURE 1.2. Examples where communication signal strength is misleading: a) thin interior wall prevents passage but signal is strong b) blocking exterior wall prevents signal but nodes are topologically adjacent

Information regarding the spatial locations of the nodes as well as their communication connectivity can make it easier to determine topologically adjacent nodes and *vice versa*, although, in many cases, the information can be misleading. Spatial proximity does not necessarily imply a topological connection in some other medium such as navigation. Likewise, two nodes that are topologically adjacent do not have to be physically close to each other. For example, consider Figure 1.2, which depicts two scenarios in which received signal strength, a rough indicator of distance, gives misleading information. In the first scenario, two nodes are proximal, but are separated by a thin wall that blocks direct navigation between the two nodes. In the second scenario, the nodes can be considered topological neighbours since there exists a navigable path between them, but the signal strength is reduced due to a significant line-of-sight obstruction.

Topological information can improve on a solely metric representation which identifies the relative locations of the sensors but does not provide information about the layout of the region. By considering both the metric data and topology of the surrounding environment, information regarding obstructions and motion corridors can be inferred. For example, two spatially proximal nodes that are not topologically adjacent suggests a barrier at a particular location, perhaps an interior wall or a river (in an outdoor deployment). Recovering a useful representation of the surrounding environment can be considered an important step in the overall goal of developing self-adapting and self-configuring networks.

Our mapping approach requires only poor quality sensor and range data and employs statistical and probabilistic techniques to extract meaningful information. We assume that we have no prior knowledge of the relative locations of the sensors and that we have only a limited knowledge of the type of activity present in the environment. To infer the topological

data, we use observational data returned from our sensors to build up an understanding of motion patterns present in the environment. In the case of the metric data, we utilize low quality range estimates to build up a representation of a probability distribution function (PDF) for the pose of the network; *i.e.* the configurations of all the nodes. In the next section we will provide some more details describing this approach.

3. Overview of Approach

We will begin this section by employing a simplified abstraction that illustrates a typical problem we are interested in solving. Figure 1.3(a) depicts a sensor network distributed within an indoor environment. Let us assume that this network has been deployed for some purpose such as surveillance and requires knowledge of both the relative spatial positioning of its network components and also the topological relationship between the network components. During some initial calibration period the network collects observations of agents passing by each sensor (Figure 1.3(b)) and additionally estimates inter-sensor ranges through the use of received communication signal strength. The problem we are trying to solve is how to use these collected observations to construct a representation of the environment that contains both a topological and metric description of the relationships between the network components (Figure 1.3(c)). This type of network might arise, for example, if wireless cameras were deployed in a workplace environment.

3.1. The Topology Inference Problem. Our approach to inferring a topological representation of the network is based on exploiting existing motion in the environment. We use observational data returned from our sensors to understand traffic dynamics formed by sources of motion present in the environment. By inferring underlying patterns in these motions we can then recover the relationships between the sensors of our network. The algorithm we employ is formulated using Monte Carlo Expectation Maximization (MCEM), but also depends on selecting the simplest explanation that described the majority of the data collected (the principle of Occam's Razor [1]). Essentially, our approach attempts to infer likely sets of trajectories taken by sources of motion (agents) in the environment. These agents might be people in a building or packets in a data network. Understanding the relative order in which sensors are visited in trajectories taken by these agents gives us

[1] Occam's Razor is the principle enunciated by William of Occam that the simplest explanation is the best.

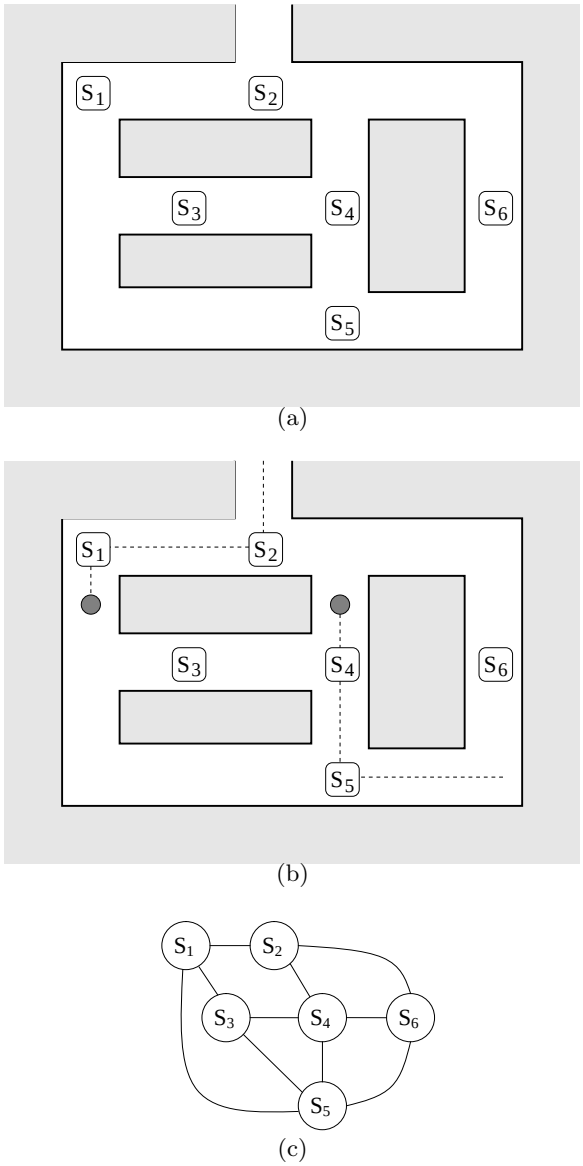


FIGURE 1.3. An example of a sensor network which we wish to calibrate. a) The original ad-hoc deployment. The lettered positions on the map indicate the placement of the nodes. b) An example of agent motion observed and exploited by the calibration process. c) The desired map of the network where edges denote traversability but not necessarily a straight-line path.

clues regarding the topology of the environment. The final result is a probabilistic model of the sensor network connectivity graph and other information describing the underlying traffic trends.

3.2. The Probabilistic Sensor Localization Problem. To infer the relative metric relationships between the sensors, our approach distinguishes itself from existing techniques by attempting to recover not only a maximum likelihood estimate (MLE) for the pose of the network, but also a probability distribution function (PDF) for each node. This probabilistic representation allows the information to be presented in the same framework as the topological data. We define the problem of obtaining and representing arbitrary distributions for the sensor locations as the *Probabilistic Sensor Localization Problem (PSLP).*

Computing an arbitrary PDF entails more computational cost than obtaining the MLE, and for the most part has been overlooked by previous authors. Providing simplistic parametric uncertainty estimates is fairly common in the literature, however, the characterization of more complex uncertainty functions appears rare and to the best of our knowledge has only been previously addressed to some degree by Ihler *et al.* with a mixture of Gaussians [**53**] and by Peng and Sichitiu with an occupancy grid [**97**]. Ours is the first work that we are aware of that attempts to directly obtain the underlying PDF for the sensor locations and is capable of accurately representing completely arbitrary distributions. Like the topological inference problem, we apply computationally sophisticated statistical techniques such as Markov Chain Monte Carlo (MCMC) to solve the constraints and provide estimates of their uncertainty inherent in the final results.

3.3. Final Outcome. Our intention is to create a probabilistic representation of the environment containing both metric and topological data that is of a form suitable for higher level inference tasks. These higher level tasks, such as navigation and planning, are themselves presumably based, at least partially, on probabilistic reasoning and therefore are able to exploit the uncertainty estimates present in our map.

4. Motivation

As sensor networks are established in more locations for monitoring and surveillance purposes, there will be a demand for algorithms and software approaches that can make inferences about the environment based on large quantities of highly distributed and possibly low quality sensing information. This is especially true in areas where we are unable to venture ourselves, or unwilling to venture for fear of influencing the data we are collecting. One

example of this type of network is the proposed underwater observing system NEPTUNE [**5**], which plans to wire the entire Juan de Fuca tectonic plate off the coast of the North-West Pacific ocean. The network will generate a vast amount of observational data from a variety of distributed sensors. These data could be used to infer additional information about the ocean environment that would be difficult to collect directly for logistical and financial reasons. The project will doubtlessly lead to a number of research efforts based on various interpretations of the sensor data. Another large and current network is used to monitor bird activities on Great Duck Island, Maine [**67, 18**]. Here the use of a sensor network allows a great deal of environmental information regarding the nesting sites of the animals to be collected without causing distress or disturbance through human presence.

Our work addresses specific aspects of the more general problem of inferring information about the environment given distributed and potentially poor quality observational data. We can envision applications utilizing the spatial data inferred by our approach both for self-configuration purposes and also as an end product on its own right. For example, a typical data collection application requires the localization of its sensors for calibration purposes, while a vehicle monitoring network might be deployed specifically for the purpose of obtaining connectivity information between key intersections. Similarly, consider sensors distributed at key points throughout a national park. The system, after localizing its components, could recover connectivity patterns giving researchers clues about wildlife behaviours.

One potential application of our work is in the construction of a *smart house* in which actuators and sensors respond to the behaviour of the resident in order to help them with their daily activities [**11**]. Such a technology could allow aging seniors more independence and therefore a better quality of life. Furthermore, it could allow them to stay longer in their own residence than might otherwise be possible, alleviating pressure on public institutions. A mobile robot combined with a sensor network in such a system could aid the resident in many tasks, but must deal with navigation and localization. Furthermore, many of the sensors distributed throughout the house might require localization and information about the topology of the region in order to carry out their assigned task. Our approach could help decrease the complexity of the installation process, and therefore help realize the practical deployment of such a system.

Another potential application for our research is in the calibration of sensor networks deployed for the purpose of monitoring carbon and other greenhouse gas (GHG) emissions. There is consensus in the scientific community that atmospheric concentrations of GHGs have caused changes in global climate patterns as a result of human activity [**120**]. International and local efforts to manage GHG emissions will require methods of measuring and detecting these substances and sensor networks are a suitable technology for this task. Simplifying the processes involved in the deployment and installations of detection and monitoring networks should effectively reduce their cost and improve their widespread availablity and practical application.

5. Contributions

In this section we identify the key intellectual contributions made in this thesis. They focus on the development and validation of techniques for the self-calibration of intelligent systems. Listed in point form, the major contributions are:

- The general formulation of a statistical approach for accurately inferring the topology of a network using signature free observations from the distributed nodes. This approach has been shown superior to previous related techniques.
- The application of the above approach to a sensor network self-calibration problem. A complete system is described which includes event detection and collection from a heterogeneous sensor network.
- The first formalization of the Probabilistic Sensor Localization Problem (PSLP).
- The presentation and experimental validation *via* simulations and hardware of two MCMC-based algorithms for solving variants of the PSLP problem.

Additional secondary contributions include:

- The first theoretical analysis of the topology inference problem with signature free observations for the special case in which the observations are ordered, but time-stamp free.
- The presentation and simulation-based validation of an algorithm for solving the marker-less graph exploration problem in robotics. This algorithm out performs previous work in this area.

6. Statement of Originality

Various portions of the research presented in this thesis have been previously published [**76**] [**79**] [**78**] [**36**] [**75**] [**74**] [**72**] [**73**] [**70**] [**77**] [**71**] or are in the process of being submitted for publication. There are a number of colleagues who, in addition to my supervisor and committee members, have made key intellectual contributions to the research presented in this thesis. David Fleet is one of the participants of the research, in collaboration with my supervisor and myself, that led ultimately to the topology inference work presented in Chapter 3. Philippe Giguère made significant contributions to the work presented in Chapter 4 and Ketan Dalal should also be acknowledged for his help on this Chapter. David Meger and Ioannis Rekleitis have also contributed significantly to the work presented in Chapter 7.

7. Outline

In this thesis, we present and validate algorithms related to inferring environmental representations through the exploitation of limited sensory data. These efforts are focused, for the most part, on sensor network self-calibration. In this introductory chapter, we have attempted to give a general description and motivation for the types of problems we are interested in solving. In the chapter that follows, we will discuss related work that has addressed similar types of problems, and additionally, provide some background material on some of the techniques utilized by our approach.

In Chapters 3, 4 and 5 we will present algorithms based on recovering a topological map of the environment based on limited sensory data. Chapter 3 will consider inferring this topological data based solely on motion detections observed from distributed sensors. This chapter includes results from both simulations and experiments and therefore includes some systems building notes related to the implementation of the network used for collecting the experimental data. Chapter 4 provides an analysis of a version of the sensor network topological mapping problem in which only ordering information, and not timing information, is available from the observations. Chapter 5 considers the topological mapping problem from the perspective of a single mobile robot with extremely limited sensory abilities. In both Chapters 4 and 5 only simulation results are presented.

In Chapters 6 and 7 we consider the problem of obtaining a metric map of the environment including accurate uncertainties; *i.e.* the probabilistic sensor localization problem (PSLP). Like the body of work considered in Chapters 3 through 5 we again assume that we have only limited sensory data. First in Chapter 6, we consider and assess with simulations the case of localizing a network of stationary components in which only range data is available. Then in Chapter 7, we consider the case that the network is augmented with a mobile robot. Here we consider the problem of localizing the network based only on the odometry information collected by the robot. Both simulation and experimental results are presented for this portion of the work. Finally, in Chapter 8 we wrap up with some concluding remarks and consider directions for future work.

CHAPTER 2

Background

In this chapter, we describe related work and give some background on the techniques employed by our approach. First, we review work on sensor network self-configuration and network calibration in general. Then, we focus in on calibrating network topology based on the exploitation of motion in the environment and existing techniques for topology calibration in sensor networks. This is followed by a review of some related work in the field of multi-target tracking, since our approach shares some methods employed in this research area. We then turn to the closely related field of robotics and give some background on topological mapping and simultaneous localization and mapping (SLAM) in this area. Finally, we provide some background information on Expectation Maximization, Markov Chain Monte Carlo, and other statistical techniques used in our work.

1. Related Work

1.1. Network Self-Configuration. It is recognized that self-calibration and other more general self-configuration algorithms are important for sensor networks [**21, 125**]. The ability of a network to automatically adapt to varying conditions is essential if large numbers of sensors are to be rapidly deployed in an ad-hoc manner [**13**]. Since it is not practical for a technician to tune individual parameters on each network component, the system should be able to operate as autonomously as possible even in a dynamic environment. Ideally, the network should be capable of re-organizing itself to handle changes such as individual node failures, changes in the layout of the network components, and shifting communication

ranges. Additionally it should be able to track and respond to relevant changes in the surrounding environment.

Much of the research conducted on self-configuration efforts is based on developing distributed, computationally efficient algorithms appropriate for low-power sensor network platforms. Work by Estrin *et al.*, for example, has focused on developing a distributed, query-based algorithm model for information dissemination [**38, 54**]. Called directed diffusion, it has been an influential concept in self-configuration research.

Other research on self-configuration has looked at self-organizing for routing and networking efficiency purposes [**1, 132, 60, 3, 22**]. For example, Arici and Altunbasak [**3**] have looked at the issue of reducing overall resource usage by only activating a subset of the available nodes for some task while Couture *et al.* [**22**] have considered the issue of finding a subset of nodes in the network that can be used for effective communication. This area of wireless sensor research generally focuses on communication related issues among the network components and uses simpler and less complete spatial models than our work.

As sensor network research has matured, there has been a shift towards more complex approaches incorporating advanced probabilistic techniques and graphical models [**53, 94**]. For example, work by Paskin *et al.* [**94**] considers the problem of constucting a data structure called a junction tree in a sensor network in order to solve inference problems *via* message passing. In work by Dantu and Sukhatme [**23**] it is argued that for some applications tiered network architectures that incorporate components of some computational sophistication are more appropriate than traditional systems of impoverished, homogenous nodes. While practical implementation is a concern, many sensor network researchers now consider using computationally sophisticated techniques in the processing of distributed observations. This is especially true for sensor networks made up of vision-based sensors, *e.g.* the work of Rahimi *et al.* [**100**] or the work of Javed *et al.* [**55**].

1.2. Network Self-Localization. A key self-configuration requirement for many network applications is the ability to self-localize [**1**]; *i.e.* recover the relative metric positions of the individual sensors in the network. The majority of self-localization efforts to date have focused on recovering the relative locations of the sensors in situations where satellite-based Global Positioning System locators are too expensive, not available, or otherwise impractical [**16**] [**112**]. Localization efforts are usually based on methods for estimating

the distances between sensors and then integrating these measurements across the network. Common techniques include the use of received communication signal strength in radio networks, *e.g.* the work of Bulusu *et al.* [**14**], or time-of-arrival ranging using ultrasound, *e.g.* the work of Niculescu and Nath [**92**]. Many approaches assume that several of the sensors in the network have a known location and act as beacons or anchor nodes for their neighbors; the work of Patwari *et al.* [**96**] is an example where this common idea is applied. Range estimation techniques typically have limited accuracy and localization algorithms must be able to handle some degree of noise in the range data. For example, the work of Moore *et al.* [**89**] considers a localization algorithm based on quadrilateral formations of nodes that are robust to ambiguities common when there is significant noise in range estimations. In additional related work, Boukerche *et al.* [**9**] consider the vulnerability of localization approaches to attack or disruption.

When addressing sensor network self-localization, most authors consider only the problem of estimating the maximum likelihood network configuration and do not consider the problem of accurately characterizing the uncertainty in the estimates. Even in the case that uncertainties are considered, the models employed are usually not expressive enough to represent arbitrary or multi-modal distributions. Two exceptions that we are aware of are the work of Ihler *et al.* [**53**] and the work of Peng and Sichitiu [**97**]. In both cases, the researchers involved have considered self-localization approaches that result in an expressive representation of uncertainties.

The sensor network self-localization method described by Ihler *et al.* [**53**] models the conditional independence of the various range estimates as a graphical model and then solves the graphical model using nonparametric belief propagation. Each node in the graph corresponds to a single network component, and maintains an estimate of the posterior marginal distribution for the location of that component. This estimate, or 'belief', is then communicated to all of its neighbours in the graphical model during one iteration of the belief propagation process. Each component updates its belief based on the information received from its neighbours and the process continues. Internal to each node, the beliefs are represented by a number of samples, but when the estimates are communicated they are converted to a mixture of Gaussians, or in some cases, an analytic function. One of the strengths of this approach is the manner in which the algorithm can be distributed

among the various network components, however, the uncertainties represented suffer from the approximations made during the message passing portion of the belief propagation, and there are no guarantees of correctness. Additionally, the computational responsibility of each individual sensor is significant and could be challenging to implement without a floating point processor and only limited amounts of RAM. Experimental results presented are from simulations only and it is unclear whether this technique is practical for certain sensor networks, such as those containing low-powered, resource-limited devices.

Peng and Sichitiu [**97**] consider the localization of a sensor network for outdoor environments where received signal strength (RRS) can be used to provide an model of range between sensors. Beacon nodes broadcast position estimates which are propagated throughout the network along with the RRS values recorded by intermediate sensor nodes. Each sensor maintains its own position estimate internally in the form of an occupancy grid that encompasses the local environment. When a node receives a new message it uses the information gained to update its own location estimate and then broadcasts the updated position estimate to its neighbours. A number of communication and processing optimizations are discussed, but like the work of Ihler *et al.*, experiments are restricted to simulations and it appears the approach could have difficulties if implemented on typical resource-limited devices.

Both the sensor network self-localization work of Ihler *et al.* and Peng and Sichitiu do not claim to accurately obtain the underlying PDF for the pose of the network. Instead, their work focuses on a mechanism for the efficient estimation of the locations and uncertainties of the sensors in a manner suitable for distributed processing. In our work, we assume the existence of a powerful processing unit on which centralized processing can take place and focus on accurately representing the underlying PDF in a principled manner.

1.3. Techniques Exploiting Motion in the Environment. Some recent work has looked at the self-configuration of multi-sensors networks by exploiting motion in the environment [**122, 39, 121, 100**]. These efforts generally assume vision-based sensors and place less emphasis on the traditional sensor network concerns of efficiency and distributed processing. Instead, they focus on research issues regarding the processing of observations collected from distributed sensors.

Stein [**122**] considered the problem of self-calibrating multiple cameras which were far apart but had some overlap in their field of view. The method used moving objects observed in pairs of cameras to determine a rough alignment of the ground plane. An error minimization technique was used to select a homography matrix computed from a set of matching motion features in the two sequences. The matrix uniquely maps a point in one image to a point in the second. The homography estimate was then improved using static objects in the two overlapping scenes. Using internal camera parameters, the relative location of the cameras could then be recovered by decomposing the homography matrix defined by the planar alignment.

Also relying on overlapping fields of view between cameras placed at adjacent sensing locations, Stauffer and Tieu [**121**] described a method for building a tracking correspondence model based solely on observational data. Their work focused on determining which tracking data resulted from observations of the same objects in sensors with overlapping views. Their approach was based on probabilistically determining correspondences between cameras and ultimately using this information to calibrate a camera network to better track objects between fields of view. They verified their method with experiments conducted on a five camera network.

Fisher [**39**] explored a self-localization approach for networks of cameras without overlapping fields of view. This method exploited the motion of distant moving objects such as stars. The objects were assumed to have well-behaved linear or parabolic trajectories, and it was necessary that the observed objects could be uniquely identified across separate cameras.

In a more recent effort, Rahimi *et al.* [**100**] described a simultaneous calibration and tracking algorithm that uses a velocity extrapolation technique to self-localize a network of non-overlapping cameras based on the motion of a single target. Their work avoided the difficult problem of associating observations with different targets by assuming only one source of motion with a long trajectory.

1.4. Network Topology Calibration. Some previous work has considered the problem of inferring *topology* or connectivity information in networks of sensors. In contrast to our approach, these efforts either address a slightly different problem [**55, 99**] or they employ considerably different methods [**68, 37**].

To track multiple agents across disjoint fields of view, Javed *et al.* [**55**] first calibrated the connectivity information of their surveillance system using observational data. To learn the probability of correspondence (transition probabilities) and inter-camera travel times (delay distributions), they assumed a training period in which the data association between observations and agents was known. Given this observation ownership information, they employed a Parzen window based technique that looks for correspondences in agent velocity, inter-camera travel time, and the location of agent exit and entry in the fields of view of the camera.

The complete system that they developed is a multi-target tracking application that incorporates object signatures and the learned network connectivity parameters into a Bayesian framework. Their method was successfully verified on small networks of two and three cameras. However, by assuming the data association problem to be solved, Javed *et al.* addressed a different version of the network topology problem than the one we are interested in.

Focusing on camera network calibration, Ellis *et al.* [**68, 37**], presented a technique for topology recovery based on event detection only. They outlined an approach in which they first identified entrance and exit points in camera fields of view and then attempted to find correspondences between these entrance and exit based on video data. The method could then automatically determine the topology of the camera network by assuming links between cameras with corresponding entrance and exit zones.

Their approach relies on exploiting temporal correlation in observations of agent movements as they enter and leave the field of view of different cameras. They do not rely on object correlation across specific cameras. Instead, they consider whether there is a strong temporal correlation in delay times between each pair of entrance and exit zones. To determine if a correlation exists between a particular entrance and exit, all entrance appearances occurring within a time window from the exit event are collected in a discrete-time buffer; *i.e.* a histogram of delay times is constructed. This histogram corresponds to a transition time probability distribution function for the entrance-exit pair. The detection of a peak in this temporal distribution of travel times between the pair suggests that a correspondence or topological link exists. Two methods were considered for detecting a peak in the distribution of delay times: a threshold-based heuristic and a technique based on a mixture of k

Gaussians. The heuristic approach proved more accurate. It looked for peaks in the distribution of delay times between pairs of entrance and exit zones by setting a threshold based on a weighted sum of the mean and standard deviation of the transition time probability distribution function.

The technique gave promising results on experiments carried out on a six camera network. Although it requires a large number of observations, the method does not rely on object correlation across specific cameras. Thus, the approach can be used to efficiently produce an approximate network connectivity graph. However, in contrast to our more computationally sophisticated algorithm, when the network dynamics are complex or the traffic distribution exhibits substantial variation the accuracy of the technique suffers.

1.5. Multi-Target Tracking. Much of the work on network calibration through the exploitation of motion is motivated by or incorporated into research conducted on multi-target tracking. The work of Javed *et al.* [**55**] and that of Stauffer and Tieu[**121**] for example, is directly related to the development of multi-target tracking systems. Similarly, one of the stated goals of Ellis *et al.* [**68, 37**] is to enhance the tracking performance of surveillance systems. Since our approach relies on recovering plausible trajectories of individual agent motion, we address some of the same problems faced in this area of research.

Multi-target tracking is a well established research area in sensor networks [**4, 65**] and multi-robot systems [**109**]. One of the key difficulties faced is that of maintaining target identities during periods when two or more targets move close together or are unobserved for a period of time. Probabilistic techniques such as Identity Mass Flow as described by Shin *et al.* [**116**] have been devised to handle this situation. Other work poses the target identity problem as a data association problem; *e.g.* work by Rasmussen and Hager [**104**], and work by Huang and Russell [**51, 52**]. The core idea is to account for the net mass (or number of agents) in the system and postulate models for how to account for it when there are insufficient observations (*e.g.* by assuming one agent hides another).

Pasula *et al.* [**95**] successfully approached a traffic monitoring problem from the data association perspective through a stochastic sampling technique, although only in very simple networks. Given known sensor positions and topology, the goal of the work was to track multiple objects passing through the network and recover their long-range origin/destination information. An iterative Expectation Maximization algorithm was employed that assigned

probable trajectories to each vehicle. These samples were then used to update model parameters such as link-travel time and vehicle characteristics. New trajectory samples were generated from existing samples by swapping vehicle assignments between pairs of adjacent sensors. A new sample was accepted based on its relative probability to the existing sample.

The approach was verified using a freeway simulator that modeled one-hundred cars of different colors passing through a network of nine cameras. The algorithm remained robust when the color discrimination capability of the cameras was reduced, however, no results are presented for spurious or missing observational data.

Our method of generating trajectory samples in our topology inference algorithm is close in spirit to that used by Pasula *et al.* [**95**]. Our implementation differs, however, due to the specifics of the problem. Additionally, we use the trajectory samples for a very different purpose. While they inferred motion parameters given a known network, we address the opposite problem: inferring information about the network given motion in the environment.

Statistical tracking based on network tomography has also been used to recover environmental information based on large amounts of sparse data. Vardi [**128**] introduced a statistical method for estimating node-to-node (source-destination) traffic flow based on traffic counts at inter-node links. The technique, which was applied to strongly connected networks of known topology, assumes that network traffic can be modeled through a Poisson distribution. The Poisson assumption results in a set of linear equations that can be solved through the method of moments. This technique has been evaluated on traffic surveillance video data [**10**] and the results were comparable to conventional tracking algorithms.

1.6. Topology Inference. Although investigations into topological mapping have been relatively recent in the area of sensor networks, the area has been well explored in the mobile robotics community. There are similarities in the type of data that must be processed and hence the techniques employed. In the context of robotics, a topological map provides: first, a useful representation of the environment that allows robot navigation without necessarily requiring the maintenance of the robot's pose in a global reference frame; and second, an abstraction provided by the topology information that can aid higher level planning and inference tasks. The first is true of any sensor network or other intelligent system that incorporates one or more mobile components, and the second is true regardless.

Early work in this area by Kuipers and Byun [**62**] constructed a topological network description of the environment by identifying and then linking distinctive places and paths based on the sensory input and control strategies of the robot. Later work by Dudek, [**32**] describes an approach for building a hierarchy of representations of an unknown environment. At the lowest level, the hierarchy begin with geometric data obtained from the processing of sensory data gathered by the robot. The final abstraction is a topological map with attached semantic labels that could be used for higher level tasks. This concept is refined in later work by Simhon and Dudek [**117**] with the idea of *islands of reliability* which link locally understood metric maps in a larger topological representation of the global environment.

Work such as [**115**] by Shatkay and Kaelbling addressed the topological mapping problem with statistical formulations and techniques. They model the robot's interaction with the world as a Hidden Markov Model and employ an extended Baum-Welch algorithm to recover its parameters. Their approach incorporates odometry data and abstracted sensory information collected by the robot. The outcome of these approaches is generally a graph where vertices represent distinct locations or landmarks in the region and edges indicate navigability.

Practical applications of topological mapping must provide a method for the robot to reliably identify a topological node, (or landmark) in the world being explored. For example, in [**19**] Choset and Nagatani use sonar data to identify and position the robot on a Voronoi graph of the environment. The vertices of the graph are then used as symbols which are combined with odometry data to create a topological map that is used for localization. In work by Kuipers and Beeson [**61**], place recognition is achieved through a multi-process bootstrapping technique that includes sensory clustering and probabilistic inference. Other approaches consider the extraction of features from vision or other sensory data (*e.g.* [**114**] [**110**] [**44**]).

A more recent approach in topological mapping by Ranganthan and Dellaert [**101**] [**103**] [**102**] relies almost exclusively on odometry data. In addition to collecting imperfect odometry data, their approach requires only that the robot have the ability to detect signature-free landmarks; *i.e* the robot can detect a proximal landmark, but can not differentiate between the different landmarks it encounters. The final outcome is a probabilistic

distribution over potential topologies that describe the obtained observations. The authors refer to this result as a Probabilistic Topological Map (PTM) and explore various inference techniques for generating the PTMs including Markov Chain Monte Carlo (MCMC) [**101**] [**102**] and a Rao-Blackwellized particle filter [**103**]. One aspect of our work in this thesis considers the problem of inferring the topology of the environment in the case that our 'sensor network' consists of a single mobile robot. In this area our approach is similar in concept to this work by Ranganathan and Dellaert [**101**] [**103**]. The weighted partial world models we maintain as part of the inference technique we apply to this problem have some similarity to the concept of a PTM as defined by these authors. In both our technique and theirs, a multi-hypothesis, topological representation is maintained. The distinguishing difference is that, while we only apply a ranking heuristic function, they use odometry measurements to assign relative probabilities to each of the potential world models.

In topological mapping, and in mapping in general, one of the issues of applying the observations collected from the world in the map making process is identifying *where* the observations were collected. Of course, without a map or prior knowledge, localizing the sensor can be difficult. This issue has been considered extensively in mobile robotics where it is known as the simultaneous localization and mapping problem (SLAM). We will briefly discuss this related area in the next section.

1.7. Simultaneous Localization and Mapping (SLAM). Inferring a representation of the environment from sensors without prior knowledge of how the sensors are distributed has some relation to the simultaneous localization and mapping (SLAM) problem in mobile robotics. In traditional SLAM, a robot of uncertain pose uses observations collected over time to determine both its own location and a map of the environment. Generally, the solutions for SLAM and related problems employ a complex probabilistic framework and use sophisticated inference techniques that are computationally intensive in comparison to approaches designed to run on a typical sensor network platform. There are a number of techniques we can borrow from this research, especially in the case that our sensor network includes mobile components.

The extended Kalman filter, as was pioneered by Smith *et al.* [**119**] [**118**], and modern variants of this original technique are widely employed in approaches to SLAM; (*e.g.* in work by Wolf and Sukhatme [**131**]). The use of the Expectation Maximization algorithm is

also often employed; for example by Shatkay and Kaelbling [**115**] and by Thrun *et al.* [**126**]. Efficient recent methods such as the FastSLAM algorithms of Montemerlo *et al.* [**86**] [**88**] employ a combination of filtering and sampling-based statistical techniques.

Instead of filtering, or in combination with filtering, global approaches are sometimes applied in which the entire set of maintained poses is considered, this global approach is also referred to as *smoothing*. Early work in this area by Lu and Milios [**66**] considered an energy minimization approach for comparing sets of range scan measurements and obtaining a globally consist map of the environment. A number of researchers have considered enhancements to the Lu and Milios style of approach; *e.g.* the work of Gutmann and Konolige [**46**] which considers techniques for dealing with cyclical structures in the environment. A recent example of smoothing is the work of Dellaert and Kaess [**24**] who use sparse linear algebra techniques based on matrix square roots to efficiently incorporate the entire trajectory of the robot into the map making process.

These later global methods are similar in concept to our approach to metric self-localization for sensor networks, however, they do not return a full representation of the underlying distribution. Furthermore they are targeted at robotic mapping, whereas our work targets the more sparse observations obtained when mapping a sensor network. Additional examples of sensor network applications involving mobile components employing SLAM-like techniques include the work of Reklietis *et. al* [**105**] in their use of an extended Kalman filter for the self-calibration of a hybrid robot/camera-network system, Coates [**20**], who employs a particle filter for distributed state estimation and Djugash *et. al* [**30**] who consider the localization problem in the case where only range data to sensor nodes is available.

Although some aspects of our network localization problem in which we incorporate mobile components can be viewed in the context of SLAM, there are significant differences. First, our landmarks are actually deployed sensors and can be considered uniquely identifiable, so there is no correspondence issue. Second, we assume that in our system, any mobile component (robot) will operate for the most part within the confines of sensor-network deployed region and will ultimately visit the local area of each stationary network component many times. Given such behaviour, it will be desirable to know not only the most likely location for each stationary sensor, but with what confidence or certainty such a location is

known. Finally, in the scenario we consider, we assume that mobile robots in our deployed network are only able to detect or be detected by the stationary sensors. We also assume that these robots do not have sensing capabilities that could be used to identify, and localize additional landmarks in the environment.

2. Background on Statistical Techniques Employed

This section will provide some brief background information on some of the main statistical techniques employed by our approach. See Tanner [**124**] or Gilks [**45**] for more detailed explanations.

2.1. Expectation Maximization (EM). Expectation Maximization (EM) addresses the problem of fitting a model to data in cases where the solution can not be determined analytically. The technique augments the existing data with additional *hidden data* which can then be used to incrementally improve the model estimate. The hidden data gives values to the unobserved, latent variables required by the model. This approach is commonly used for parameter estimation in problems like ours in which there are only incomplete data models. EM has been shown to converge to a set of model parameters that locally maximize the likelihood [**27**].

EM iterates over two steps: the *E Step* and the *M Step*. In the *E Step*, an expression for the expected likelihood of the *complete data* is calculated given the current estimate of the model parameters θ. The *complete data* is made up of both the *observed data* y and a probability distribution function over possible values for the *hidden data* z. The hidden data allows the latent variables to be treated as if they were observed. In the second *M Step* the set of parameters determining the model estimate are updated to maximize the expected likelihood calculated in the first step. It should be noted that in many EM applications these two 'steps' do not lead to two separate processes that one would follow as in an algorithm. For example, when both of the equations resulting from the two steps have an easily calculated closed form there can be just one update step *per* iteration. Conceptually, however, the *E Step* and the *M Step* are repeated in turn until the process converges, as measured by the size of successive changes in the model parameters:

(i) *The E-Step:*

$$Q\big(\theta, \theta^{(i-1)}\big) = E\Big[\log p(z, y|\theta)|y, \theta^{(i-1)}\Big]$$

(ii) *The M-Step:*

$$\theta^{(i)} = \underset{\theta}{\operatorname{argmax}}\, Q\big(\theta, \theta^{(i-1)}\big)$$

where θ^{i-1} is the parameter estimation obtained during the last iteration and p gives the likelihood of the data given the model.

The term EM was coined by Dempster *et al.* in a 1977 paper [**27**] in which they formalized the algorithm and provided a proof of convergence. However, the technique had been applied earlier in specific problem domains. For example, the Baum-Welch algorithm [**7**] applies the EM principle to Hidden Markov Models, and has been used in the speech recognition community since the 1960s[1]. EM is currently a well established and much used statistical technique. It has been applied, for example, to mapping in robotics [**15, 126, 115**], and to problems in bio-informatics [**127**].

2.2. Monte Carlo Expectation Maximization (MCEM). One problem with EM, however, is that the *E Step* can be difficult or impossible to calculate exactly. For example, consider the case in which there is no closed form for the expected value and there are arbitrary distributions over the potential values that could be assigned to the latent variables. This situation could make integration intractable, leading to no exact formulation of the expected value. However, stochastic versions of the EM algorithm expand its applicability by executing the *E Step* through a Monte-Carlo estimation process. In MCEM, the expected value of the *hidden data* is estimated using a number of samples of potential values for the latent variables. A common approach for computing the estimate is to use Markov Chain Monte Carlo (MCMC) sampling [**130**]. In this case, the *E Step* is now calculated as follows:

$$Q\big(\theta, \theta^{(i-1)}\big) = \frac{1}{M}\sum_{m=1}^{M} \log p(z^{(m)}, y|\theta)$$

where $z^{(m)}$ is drawn using MCMC sampling from the previously estimated $\theta^{(i-1)}$.

[1]Lloyd Welch and Leonard Baum developed the solution circa 1962 but it was classified for many years.

The technique expands the scope of the basic EM algorithm to problems for which it is impossible to obtain a closed form equation for the expected value of the *hidden data*, but for which the distribution can be estimated through random sampling. In recent years, MCEM has been successfully applied to a number of areas such as tracking [**95**] and structure from motion [**25**].

2.3. Markov Chain Monte Carlo (MCMC). A precursor to many methods, including MCEM is fair sampling. This, in itself, can be a difficult problem. The Markov Chain Monte Carlo method provides a mechanism for drawing representative samples from the probability distribution π of a very large (but finite) state space Ω. The technique was first used computationally in the field of statistical physics, but since then its application has grown to include combinatorial enumeration and optimization problems. For many of these problems it is the only known method to provide a polynomial time approximation to the desired probability distribution [**56**].

Many approaches to sampling depend on the *ergodicity* of the underlying process. Recall that a Markov Chain is a memory-less process defined by a state space Ω, a transition matrix P, and initial probabilities. Ergodicity implies that there exists a unique stationary distribution π over Ω such that $\pi = \pi P$, and that it will be reached from any initial state in the limit:

$$\lim_{k\to\infty} p_{ij}^k = \pi_j, \forall i, j \in \Omega$$

where p_{ij}^k refers to the probability of transition from state j to state i after k time steps and π_j specifies the density of state j in the stationary distribution.

For a finite state Markov chain, ergodicity is implied by an *irreducible* and *aperiodic* transition matrix P [**40**]. Irreducibility implies that any two states $i, j \in \Omega$ are in the same communication class; *i.e.* s_j can be reached from s_i. Aperiodicity implies that for at least one state $i \in \Omega$, there does not exist an integer $r_i > 1$ for which $p_{ii}^k > 0$ only when k is a multiple of r_i. In other words, a Markov chain is ergodic if the transition graph is connected and the chain can not get trapped in cycles.

An additional attribute of an MCMC sampler that allows improved performance guarantees is *reversibility* or *detailed balance*. Informally, reversibility means that the average density flowing between any two states is equal in both directions. For a Markov Chain,

reversibility is specified by the following detailed balance equation:

$$\pi_i p_{ij} = \pi_j p_{ji}, \forall i, j \in \Omega \tag{2.1}$$

For an ergodic Markov Chain, reversibility is a sufficient (but not necessary) condition for ensuring that the unique stationary distribution (guaranteed to exist because of the ergodic property) is specified by the target distribution [**2**]. Essentially, the relative probabilities of the states, as controlled through the detailed balance equation, determines the final steady state distribution π.

2.4. Metropolis-Hastings Algorithm. The Metropolis-Hastings (MH) algorithm is an established MCMC sampling technique first proposed by Metropolis *et al.* [**84**] in 1953 and later generalized by Hastings [**49**]. The technique constructs a Markov Chain by accepting proposed transitions $R = \{r_{ij}\}$ from the current state s_i to a new state s_j based on a probability determined according to:

$$\alpha = \min(1, \frac{\pi_j r_{ij}}{\pi_i r_{ji}}) \tag{2.2}$$

The original algorithm described by Metropolis used a symmetric proposal function in which $r_{ij} = r_{ji}$. This results in the acceptance test:

$$\alpha = \min(1, \frac{\pi_j}{\pi_i}) \tag{2.3}$$

If one follows the Metropolis or Metropolis-Hastings algorithm for constructing a Markov Chain, then it has been shown that: (1), the transition matrix P is reversible; and (2), if the proposal matrix R is ergodic then so is P [**40**]. Therefore, ergodicity can be ensured through a well chosen proposal method, and reversibility is inherently assured by the nature of the Metropolis-Hastings method. Given ergodicity, and sufficient simulation time, the resulting Markov Chain should yield samples representative of the distribution π.

Despite theoretical guarantees, determining a 'sufficient simulation time' can be challenging and is related to the mixing rate of the chain; *i.e.* how quickly the chain tours the target distribution. However, there are a number of practices commonly followed when using MH to construct a Markov chain which helps yield representative samples. The proposal method $r(.|s)$ should be crafted such that the *acceptance rate* of new proposals, as

determined by equation 2.2, is neither too high nor too low as both these extremes will lead to a reduced mixing rate. As well, some inital number of samples drawn from the chain should be discarded; this is refered to as a 'burn-in' time. Informally, the burn-in period gives the chain a chance to 'forget' its starting position and reach a high probability state representative of the target distribution. Additionally when assessing if enough samples have been drawn in order to adequately characterize the target distribution, one looks for statistical similarities between sets of samples drawn from different portions of the chain or from seperate runs of the chain based on different starting positions. See Gilks *et al.* [**45**] for more MCMC implementation details.

CHAPTER 3

Learning Sensor Network Topology

As discussed earlier, our goal is to allow a sensor network or intelligent system to self-calibrate given limited observational data. In this chapter and the two that follow, we will consider the problem of obtaining a topological map of the environment. First, in this chapter we will consider the problem of automatically determining the topology and connectivity information of a network of sensors based on a statistical analysis of observed motion in the environment. The detection of activity proximal to a sensor can be a relatively simple task, and should be possible with various degrees of accuracy and range using a variety of sensors; *e.g.* an infrared detector, a microphone, or an accelerometer could all be utilized for this task. We assume this detection can be achieved in some manner, although there may be occasional errors in the detection process which can be modeled probabilistically. The key factor that makes the this problem difficult is data association: we do not know which source of motion in the environment induces a given measurement.

Our approach to solving this problem employs a two-level reasoning system. The first level is made up of our fundamental topology inference algorithm that takes the sensor observations and environmental assumptions as inputs and returns the network parameters as an output. The algorithm is formulated using Monte Carlo Expectation Maximization (MCEM), but it depends on fixed values for certain numerical parameters that represent *a priori* knowledge regarding traffic patterns in the environment. The second level searches over the input parameter space of the first level algorithm to find a global solution that optimizes a more abstract objective function based on the principle of Occam's Razor. This portion of our work addresses sensor network self-calibration, but has techniques in common

with multi-target tracking, SLAM, and other problem domains where data association is an issue. The algorithm uses only detection events from the deployed sensors and is based on reconstructing plausible trajectories for the agents through statistical techniques. We require no prior knowledge of the relative locations of the sensors and only weak assumptions regarding environmental conditions.

The final output of the two-level approach is a probabilistic model of the sensor network connectivity graph and the underlying traffic trends. It is worth noting that the technique recovers a much more complete description of network connectivity than just a topological map of the environment. We infer information about the number of agents in the system, inter-node delay distributions, inter-node transition likelihoods, and additional statistics regarding motion activity.

In the remainder of the this chapter we will first give a formal definition of the problem we are attempting to solve. We will then provide the details of our methodology followed by an assessment of the method through results obtained from both simulations and experiments.

1. Problem Description

We describe the problem of topology inference in terms of the inference of a weighted directed graph which captures the spatial relationships between the positions of the sensor nodes. The motion of multiple agents moving asynchronously through a sensor network embedded region can be modeled as a semi-Markov process. The network of sensors is described as a directed graph $G = (V, E)$, where the vertices $V = v_i$ represent the locations where sensors are deployed, and the edges $E = e_{i,j}$ represent the connectivity between them; an edge $e_{i,j}$ denotes a path from the position of sensor v_i to the position of sensor v_j. The motion of each of the N agents in this graph can be described in terms of their transition probability across each of the edges $A_n = \{a_{ij}\}$, as well as a temporal distribution indicating the duration of each transition D_n. The observations $O = \{o_t\}$ are a list of events detected at arbitrary times from the various vertices of the graph, which indicate the likely presence of one of the N agents at that position at that time.

The goal of our work is to estimate the parameters describing this semi-Markov process; *i.e.* the transition probabilities A and associated temporal distributions D. From these, we

can infer the underlying topological map G of the environment. Our approach is based on a number of assumptions. We assume that each observation was generated by exactly one agent and furthermore that the behavior of all the agents in the system can be approximated as being homogeneous; *i.e.* the motion of all agents are described by the same A and D. In addition, we must make some assumptions about the distribution of the inter-vertex transition times. Generally, we make the assumption that the delays fit some family of distributions and are bounded within a fixed range. We will show later, however, that we can relax this assumption in some situations.

In the approach we have outlined above, we are making some inherent assumptions about the behavior and quality of our sensors. We assume that an individual sensor generates only a single observation for an agent despite the fact that the agent will spend some finite amount of time within the detection range. This assumption can usually be guaranteed in practice through post processing of raw sensor measurements; a technique sometimes referred to as 'debouncing'. More difficult in practice is the assumption that, ideally, this sensor can generate a second observation when a second agent enters its detection range, even if the first agent is still detectable. A more easily satisfied assumption is that agents do not travel close enough together for this duplicate detection situation to be encountered. If this assumption is only occasionally violated, the system can model one of these events as a missing observation (*i.e* a false-negative). As we will show in later sections, our technique is robust to moderate levels of sensor error. As long as the assumptions we have outlined above are approximately correct, *i.e.* are violated infrequently and in a non-systematic manner, the inference process produces accurate results.

Given the observations O and the vertices V, the problem is to estimate the network connectivity parameters A and D, subsequently referred to as θ.

2. The First Level: Topology Inference through Expectation Maximization

The algorithm that makes up the first level of our technique infers the connectivity of a sensor network given non-discriminating observations. It assumes knowledge of the number of agents in the environment and attempts to augment the given observations with an additional data association that links each observation to an individual agent. The approach is based on the statistical technique of Expectation Maximization (EM) [**27**].

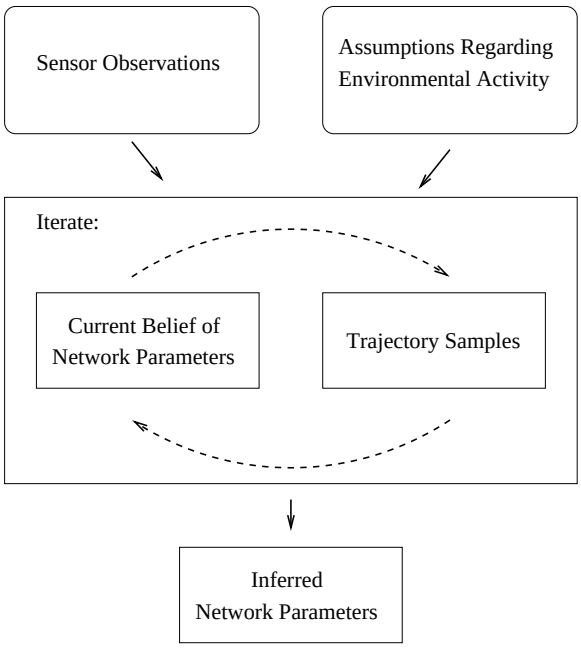


FIGURE 3.1. A block diagram of Level One of the Two-Level Approach where the blocks indicate algorithmic components and the arrows indicate the transfer of data.

The algorithm iterates over constructing plausible trajectories of agent motions based on current estimates of connectivity parameters (E Step), and then updating the parameters to maximum likelihood estimates based on the sampled trajectories (M Step). Figure 3.1 shows a block diagram illustrating the control form of the inference algorithm.

2.1. Expectation Maximization. We use the Expectation Maximization (EM) algorithm to solve the connectivity problem by simultaneously converging toward high likelihood observation data correspondences and network parameters values. For our problem instance, the *E Step* and *M Step* which we iterate over take the following form:

(i) *The E-Step:* which calculates the expected log likelihood of the complete data given the current parameter guess:

$$Q\big(\theta, \theta^{(i-1)}\big) = E\Big[\log p(O, Z|\theta)|O, \theta^{(i-1)}\Big]$$

where O is the vector of binary observations collected by each sensor, and Z represents the hidden variable that determines the data correspondence between the observations and agents moving throughout the system.

(ii) *The M-Step:* which then updates our current parameter guess with a value that maximizes the expected log likelihood:

$$\theta^{(i)} = \underset{\theta}{\operatorname{argmax}}\, Q\big(\theta, \theta^{(i-1)}\big)$$

We employ Monte Carlo Expectation Maximization [**130**] to calculate the E-Step because of the intractability of summing over the high dimensional data correspondences. Note that there is one dimension for each element of the observation vector O. We approximate $Q\big(\theta, \theta^{(i-1)}\big)$ by drawing M samples of an ownership vector $L^{(m)} = \{l_i^m\}$ (an instance of Z) which uniquely assigns the agent i to the observation o_i in sample m:

$$\theta^{(i)} = \underset{\theta}{\operatorname{argmax}} \left[\frac{1}{M} \sum_{m=1}^{M} \log p(L^{(m)}, O|\theta) \right]$$

where $L^{(m)}$ is drawn using the previously estimated $\theta^{(i-1)}$ according to a Markov Chain Monte Carlo (MCMC) sampling technique that will be explained in the next section.

In order to ensure an adequate burn-in time for the Markov Chain, a number of initial samples of the ownership vector are discarded. A simple heuristic is employed in which samples are discarded until their computed likelihood stops increasing.

At every iteration we obtain M samples of the ownership vector L, which are then used to re-estimate the connectivity parameter θ (the M-Step). We continue to iterate over the E-Step and the M-Step until we obtain a final estimate of θ. At every iteration of the algorithm the likelihood of the ownership vector tends to increase, and the process is terminated when subsequent iterations result in sufficiently small changes to θ. The following pseudo code outlines the algorithm:

```
WHILE (θ^i − θ^{i−1}) > Threshold
 Draw sample L until p(L, O|θ) stops increasing
 Draw K samples L^(k)
 Update θ^i given {L^(1) ... L^(K)}
END WHILE
```

In general, we make the assumption that the inter-vertex delays fit some family of distributions and determine the maximum likelihood parameters for each of the inter-vertex

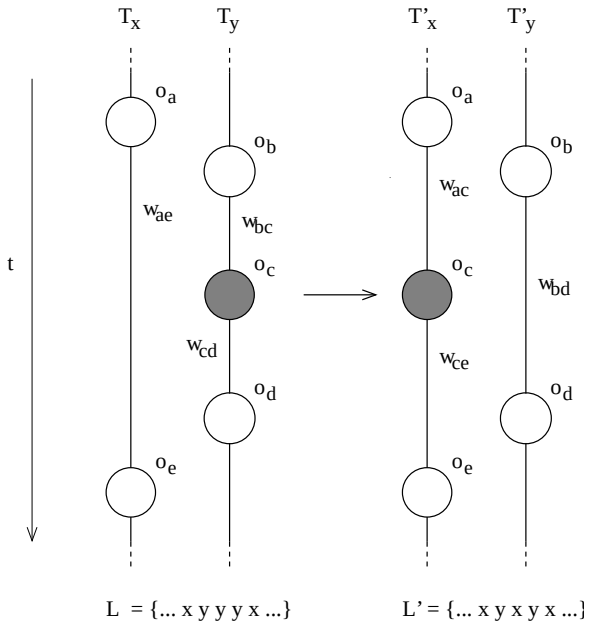


FIGURE 3.2. An example of a proposed Markov Chain transition resulting from the application of an Observation Exchange Proposal. The ownership assigned to o_c has been shifted from agent y to agent x. To evaluate this transition, the probability of the edge traversals w_{ac}, w_{ce}, w_{bd} must be compared to the original traversals w_{ae}, w_{bc}, w_{cd}.

distributions. In a subsequent section, we will describe how we occasionally reject outlying low likelihood delay data and omit it from the parameter update stage.

2.2. Trajectory Sampling. We use Markov Chain Monte Carlo sampling to assign each of the observations to one of the agents, thereby breaking the multi-agent problem into multiple versions of a single-agent problem. In the single agent case, the observations O specify a single trajectory through the graph which can be used to obtain a maximum likelihood estimate for θ. Therefore, we look for a data association that breaks O into multiple single agent trajectories. We express this data association as an ownership vector L that assigns each of the observations to a particular agent.

Given some guess of the connectivity parameter θ, we can obtain a likely data association L using the Metropolis algorithm; an established method of MCMC sampling [**124**]. See Section 2.4 of Chapter 2 for additional background on the Metropolis algorithm. From our current state in the Markov Chain specified by our current observation assignment L,

we propose a symmetric transition to a new state by reassigning a randomly selected observation to a new agent selected uniformly at random. We define this proposal scheme as an *Observation Exchange Proposal*. This new data association L' is then accepted or rejected based on an acceptance probability which is defined by the relative probabilities of L and L' according to the Metropolis selection test:

$$\alpha = \min\left(1, \frac{p(L', O|\theta)}{p(L, O|\theta)}\right) \tag{3.1}$$

from Equation 2.3.

2.2.1. *Observation Exchange Proposal.* The acceptance probability α shown in Equation 3.1 can be expressed in a simple form since the trajectories described by L' differ from those in L by only a few edge transitions. Consider L as a collection of ordered non-intersecting sets containing the observations assigned to each agent $L = (T_1 \cup T_2 \cup \ldots \cup T_N), T_n = \{w_{jk}\}$ where w_{jk} refers to the edge traversal between vertices j and k. The probability of a single agent trajectory is then the product of all of its edge transitions probabilities:

$$p(T|\theta) = \prod_{w \in T} p(w|\theta)$$

Therefore, a change in state suggested by the application of the Observation Exchange Proposal that reassigns the observation o_n from agent y to agent x must remove an edge traversal w from T_y and add it to T_x. Only the change in the trajectories of these two agents need be considered, since all other transitions remain unchanged. In the example shown in Figure 3.2:

$$\begin{aligned} \alpha &= \min\left(1, \frac{p(T'_x, T'_y|\theta)}{p(T_x, T_y|\theta)}\right) \\ &= \min\left(1, \frac{p(w_{ac}, w_{ce}, w_{bd}|\theta)}{p(w_{ae}, w_{bc}, w_{cd}|\theta)}\right) \end{aligned} \tag{3.2}$$

In between each complete sample of the ownership vector L, each of the observations are tested for a potential transition to an alternative agent assignment. This testing is accomplished in random order and should provide a large enough spacing between realizations

of the Markov Chain that we can assume some degree of independence in between samples. Although our method of proposing transitions is simple and does not result in large jumps through the state space, the acceptance test can be evaluated efficiently and we can thus afford to test many proposals.

THEOREM 1. *Applying the Metropolis algorithm with the Observation Exchange Proposal, which proposes reassigning the ownership of a single element in L, results in an ergodic and reversible Markov chain. Therefore, given adequate simultation time, the method is guaranteed to produce samples representative of the true probability distribution for the ownership vector L.*

2.2.2. *Proof of Theorem 1.* Ergodicity and reversibility in a Markov Chain are sufficient conditions to ensure that there exists a unique and *specified* stationary distribution [**2**]. In our case, reversibility is guaranteed through our use of the Metropolis algorithm; *i.e.* adhering to the Metropolis acceptance test (Equation 2.3) results in a reversible Markov chain which satisfies the detailed balance equation (Equation 2.1).

Furthermore, if our chain is ergodic, then the detailed balance equation specifies the stationary distribution π [**40**]. In our case, each state of our chain represents an instance of the ownership vector. It has a stationary distribution specified by:

$$\begin{aligned} \pi_j &= \lim_{t\to\infty} p_{ij}^t, \forall i, j \in Z \\ &\propto p(L^{(j)}|O,\theta) \end{aligned}$$

where Z represents the set of all possible realizations of the ownership vector L and p_{ij}^t gives the probability of reaching $L^{(j)}$ from $L^{(i)}$ in t steps. It remains, however, to show that our Markov Chain is ergodic.

For a finite state Markov chain, ergodicity is implied by an *irreducible* and *aperiodic* transition matrix P [**40**]. However, because we determine state transition probabilities based on the Metropolis algorithm, we are ensured that if the proposal matrix R is ergodic then so is P [**40**]. Therefore, showing that the proposal matrix R which results from our application of the Observation Exchange Proposal is irreducible and aperiodic is a sufficient condition for demonstrating the ergodicity of the resulting Markov Chain.

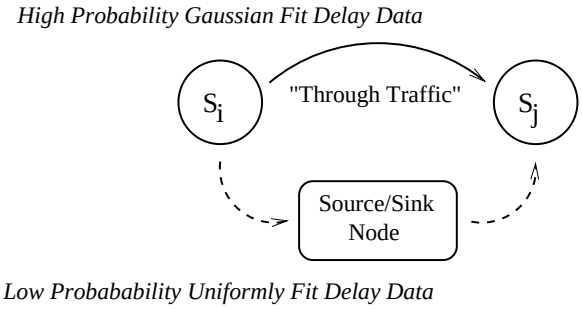


FIGURE 3.3. Graphical description of the algorithm delay model.

That our proposal matrix R is ergodic can be demonstrated by considering both a single step transition and a k step transition where k is the number of observations $|O|$. Our proposal matrix $R = \{r_{ij}\}$ gives the probability of a proposed transition from $L^{(i)}$ to $L^{(j)}$. For a single step, $r_{ij} > 0$ if $L^{(i)}$ and $L^{(j)}$ are *exactly the same* or differ by only one ownership assignment to a single observation. All other elements of the one step proposal matrix are zero. In the worst case, $L^{(i)}$ and $L^{(j)}$ can differ from each other by k ownerships assignments where $k = |O|$. In this case, the two states require k transitions to be communicable since each transition is capable of swapping one of the ownership assignments. Therefore $(r_{ij})^k > 0, \forall i, j \in Z$ as long as $k > |O|$. Since $(R)^1$ has non-zero diagonal elements, it is aperiodic and since there exists a finite k such that $(R)^k$ has all positive entries, it is irreducible $\square$.

2.3. The Delay Model. To make the algorithm more robust to realistic traffic patterns, we have introduced an inter-vertex delay model that allows for the possibility of agent transitions to and from sources and sinks. This makes the algorithm more robust both to shifting numbers of agents in the environment and to agents that pause or delay their motion in between sensors. Additionally, assuming the existence of sources and sinks, we can recover their connectivity to each of the sensors in our network.

In addition to maintaining a vertex that represents each sensor in our network, we introduce an additional vertex that represents the greater environment outside the monitored region and any phenomena that could remove or insert an agent: a *source/sink node*. If we have M sensors in our network, then our model uses one vertex $v_1 \dots v_M \in V$ for each sensor and an additional vertex $v_{M+1} \in V$ for the source/sink node. The corresponding additions are also make to the transition matrix A and temporal distributions D in our

model. Essentially, we treat this source/sink node as a potential location in the physical environment and attempt to infer associated traffic patterns for it as we do with each of the real sensor locations.

To decide when a transition to the source/sink node has occurred we use a mixture model during the E-Step of our iterative EM process in which we evaluate potential changes to agent trajectories. An inter-vertex delay time is assumed to arise from some specified family of distributions (*e.g.* a gamma distribution or a truncated normal) or else from a uniform distribution of fixed likelihood (Figure 3.3). This model allows for low probability jumps of almost arbitrary length. The data assigned to the inter-node delay distribution are assumed to be generated by direct transitions between nodes and are used during the M-Step to update our belief of the inter-node delay times and transition likelihoods. On the other hand, the data fit to the uniform distribution are used to model transitions from the first vertex into the sink/source node and then from the sink/source node to the second vertex. Therefore they are not used for updating inter-vertex delay parameters of the two nodes, but rather are considered outliers and are used only for updating the belief of transitions to and from the source/sink node for the associated vertices.

Note that no parameters are used to characterize the distribution suggested by the outlying data points; *i.e* we are not attempting to learn the delay distribution between any particular node in the system and the sink/source node. Instead, we specify when a data point should be considered an outlier given only our current belief of the parameters for the associated delay distribution. This value can not be estimated explicitly without attempting to parameterize a second distribution which would not be consistent with our model.

While the data assigned to the inter-node delay distributions are expected to be within a realistic temporal range for direct agent transitions, the delay data fit to the uniform distribution are more loosely bounded. This gives the inference technique a mechanism for temporarily removing agents from the system by assigning them to long transitions, or to explain events that would otherwise seem extremely unlikely such as the disappearance of an agent from one node and its almost immediate appearance at a second.

The delay model provides robustness to noise by discarding outliers in the delay data assigned to each pair of vertices and explaining their existence as transitions to and from

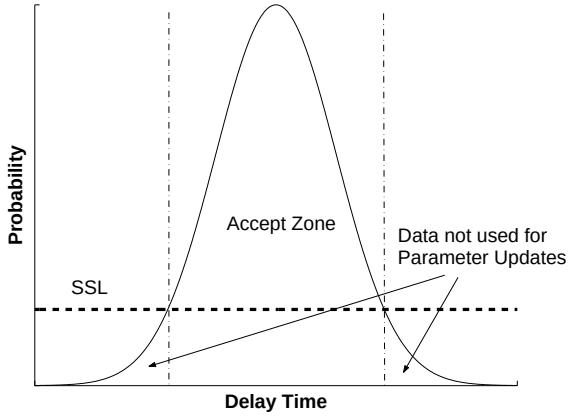


FIGURE 3.4. Graphical description of the Source Sink Likelihood (SSL) Parameter.

a source/sink node. The key to this process is determining whether or not a delay value should be considered an outlier. This is implemented through a tunable parameter, called Source Sink Likelihood (SSL), that determines the threshold probability necessary for the delay data to be incorporated into parameter updates (Figure 3.4). The probability for an inter-vertex delay is first calculated given the current belief of the delay distribution. If this probability is lower than the SSL then this motion is interpreted as a transition made *via* the source/sink node. The delay is given a probability equal to the SSL, and the transition is not used to update the network parameters associated with the origin and destination vertices.

The value assigned to the SSL parameter determines how easily the algorithm discards outliers and, hence, provides a compromise between robustness to observational noise and a tendency to discard useful data.

3. Level Two: Network Parameter Evaluation

The first stage just described has a serious limitation: it takes the number of agents N in the environment as a known input. The second level of our approach treats the topology inference algorithm described in the previous section as a 'black box' and attempts to search over its input parameter space to find reasonable solutions (Figure 3.5). We construct a heuristic evaluation function that quantitatively assesses a potential solution based on the principle of Occam's Razor. The first level topology inference algorithm takes the following inputs: the observations O; the assumed number of agents in the environment N; and the

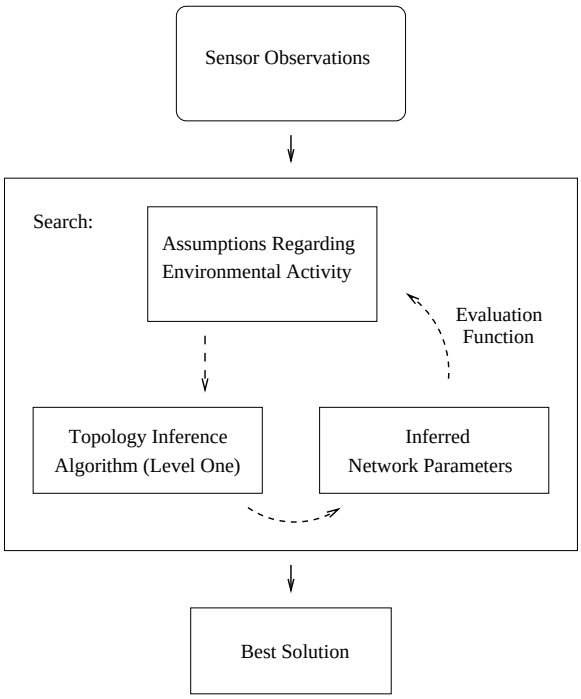


FIGURE 3.5. A block diagram of Level Two of the Two-Level Approach where the blocks indicate algorithmic components and the arrows indicate the transfer of data.

SSL parameter. The outputs of the algorithm are the network parameters θ and the *ratio* of data R_{data} incorporated into the parameter updates:

$$(\theta, R_{data}) \leftarrow alg(O, N, SSL)$$

Different input values result in different environmental assumptions and, hence, produce different outputs.

We have created a metric that attempts to assess the validity of a solution by making the assumption that a good solution both explains the majority of the data and is as *simple* as possible. This principle, known as Occam's razor, states, "if presented with a choice between indifferent alternatives, then one ought to select the simplest one." The concept is a common theme in computer science and underlies a number of approaches in AI; *e.g.* hypothesis selection in decision trees and Bayesian classifiers [**85**].

Our simplicity metric incorporates a measure of the simplicity of the transition matrix and the amount of data explained by the solution. We measure the simplicity of a transition matrix by rewarding it in inverse proportion to how close it is to a uniform belief of transition

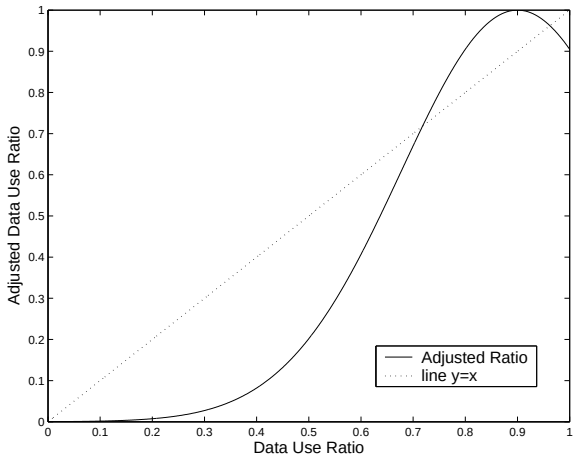


FIGURE 3.6. Example relationship between R_{data} and R_{adj} with $\gamma = 0.9$ and $\tau = 0.1$.

probabilities:

$$A_{simp} = \sum_{a_i \in A} (a_i)^{\beta}$$

where β determines the degree of the reward. We measure the utility of a given data use ratio by constructing an adjusted data ratio that attempts to reflect our belief in the solution as a function of the data used. The adjusted data ratio should incorporate the fact that some small portion of discarded data is actually optimal, but that our belief tails off rapidly as the discarded portion grows:

$$R_{data} = \frac{|\text{Explained Obs}|}{|\text{Obs}|}$$

$$R_{adj} = \exp^{-\frac{1}{\tau}(R_{data}-\gamma)^2}$$

where γ and τ describe the shape of the belief curve (Figure 3.6). The final simplicity metric incorporates a weighted combination of A_{simp} and R_{adj}:

$$Q_{simp} = (A_{simp})^{\kappa} * (R_{adj})^{\lambda}$$

where κ and λ reflect the relative weights assigned to the two portions.

With the construction of the simplicity metric Q_{simp}, we have shifted our dependence from specific *a priori* assumptions that must be made on a case to case basis. Instead, we depend on more general assumptions regarding the attributes of a believable solution for our problem domain.

Instead of the two level approach outlined in this and the previous section, an alternative approach to recovering the network connectivity parameters would be to stay within the EM framework of the fundamental algorithm. To do this, one could attempt to infer the MAP solution for a particular problem using the Q_{simp} metric as a Bayesian prior for favoring appropriate models. There would, however, be some difficulties with this approach. The first disadvantage is that, although varying the number of agents at the MCMC proposal level is possible and is related to the work of Oh *et al.* [**93**], one must invent a suitable model capable of preventing the algorithm from improving configuration likelihoods through over-fitting. For example, the almost certainly incorrect assumption that there is one agent per measurement will yield a model with an extremely high likelihood without the use of counter balancing priors. The specification of the priors could be sufficiently challenging to make this approach difficult in practice. Additionally, incorporating an arbitrary prior into the Q function of the M step of the EM loop prevents a closed form solution for maximizing the parameter values, hence forcing the use of a numerical estimation method. A potential danger here is that by attempting to infer the number of agents within the EM framework, we risk destabilizing and substantially slowing the convergence of the algorithm, which is not guaranteed under all conditions for stochastic variants; *e.g.* [**26**]. Instead, we have chosen to clearly delineate between the inner, fundamental algorithm which, in our investigations, has shown robust dependable behavior, and a second higher level component which attempts to enforce the priors we desire.

4. Simulation Results

In this section, we examine the performance of our algorithm through a number of experiments conducted in simulation. We begin with a description of our simulator. Then we assess the operation and performance of the first level topology inference algorithm and examine the effect of varying the input parameters. Finally, we discuss the ability of our technique to correctly tune these input parameters in order to achieve a minimum error solution and justify our tuning of the parameters shaping the Q_{simp} metric.

4.1. The Simulator. We have developed a tool that simulates agent traffic through an environment represented as a planar graph. Our simulation tool takes as input the number of agents in the system and a weighted graph where the edge weights are proportional

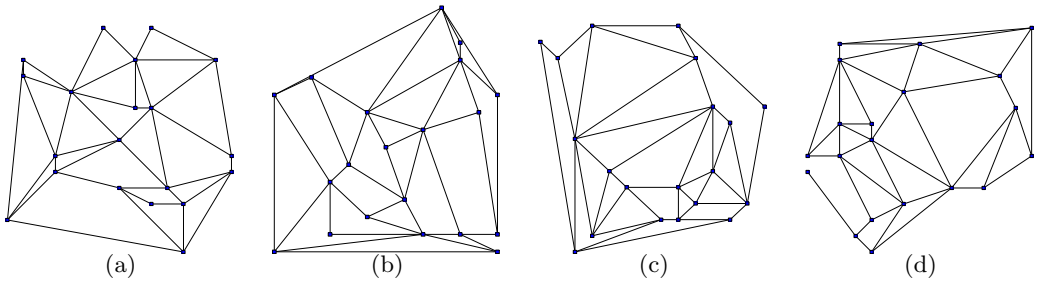


FIGURE 3.7. Examples of randomly created 20 node, 80 directed edge graphs.

to mean transit times between the nodes. All connections are considered bidirectional; *i.e.* each connection is made up of two unidirectional edges. The output is a list of observations generated by randomly walking the agents through the environment. When arriving at a new vertex during its random walk, an agent selects its next edge to traverse uniformly at random from those connected to that vertex. Inter-node transit times are determined based on a truncated normal distribution with a standard deviation equal to the square root of the mean transit time. (Negative transit times are rejected).

Two types of noise were modeled in order to assess performance using data that we believe more closely reflects observations collected from realistic traffic patterns. First, a 'white' noise was generated by removing a percentage of correct observations and replacing them with randomly generated spurious observations. Second, a more systematic noise was generated by taking a percentage of inter-vertex transitions and increasing the Gaussian distributed delay time between them by an additional delay value selected uniformly at random. The range of this additional delay time was selected to be from 0 to 20 times the average normal delay time. The hope is that small values of these types of noise simulate the effects of both imperfect sensors and also the tendency for agents to stop occasionally along their trajectories; *e.g.* to talk, use the water fountain, or enter an office for an period.

A number of experiments were run using the simulator on randomly generated planar, connected graphs. The graphs were produced by selecting a connected sub-graph of the Delaunay triangulation [**98**] of a set of randomly distributed points (Figure 3.7). In Delaunay triangulation, no edges cross and the minimum angle between edges is maximized; *i.e.* 'sharp turns' are minimized. A connected sub-graph of such a triangulation is suitable as a generic topological map of an arbitrary environment. This technique has been used before

is the field of robotics to generate random planar graphs; (see Rekleitis *et al.* [**107**] for a complete description).

For each experiment, the results were obtained by comparing the final estimated transition matrix A' to the real transition matrix A. A graph of the inferred environment was obtained by thresholding A'. The Hamming error was then calculated by measuring the distance between the true and inferred graphs normalized by the number of directed edges m in the true graph:

$$HamErr_A = \left(\frac{1}{m}\right) \sum_{a_{ij} \in A, a'_{ij} \in A'} \left[\psi(a_{ij}) - \psi(a'_{ij})\right]^2$$

where $\psi(a) = \lceil a_{ij} - \theta \rceil$.[1] Additionally, the squared error between the true and inferred transition matrix was calculated:

$$Err_A = \sum_{a_{ij} \in A, a'_{ij} \in A'} (a_{ij} - a'_{ij})^2$$

4.2. Performance under Noise Free Conditions. When operating with noise-free data and knowledge of the correct number of agents in the environment, the results show that problems involving a limited number of agents were easy to solve given an adequate number of observations (Figure 3.8). For 95 *per cent* of the generated 12 node graphs the topology was perfectly inferred with zero Hamming error for simulations with 4 agents. For simulations with 4 agents and larger graphs of 20 nodes the topology was perfectly inferred for over 50 *per cent* of the trials and for those trial in which there were errors, the resulting graph was generally within one or two directed edges of the correct graph. The problem became more difficult to solve when the number of agents in the simulation was increased from 4 to 10. In all simulations with 10 agents, however, the majority of the structure of each of the graphs considered was recovered. In the worst case, 15 *per cent* of the directed edges differed between the recovered graph and the true graph; *i.e.* the fraction of Hamming error to directed edges was at most 0.15.

[1] A threshold value of $\theta = 0.1$ was selected for our experiments.

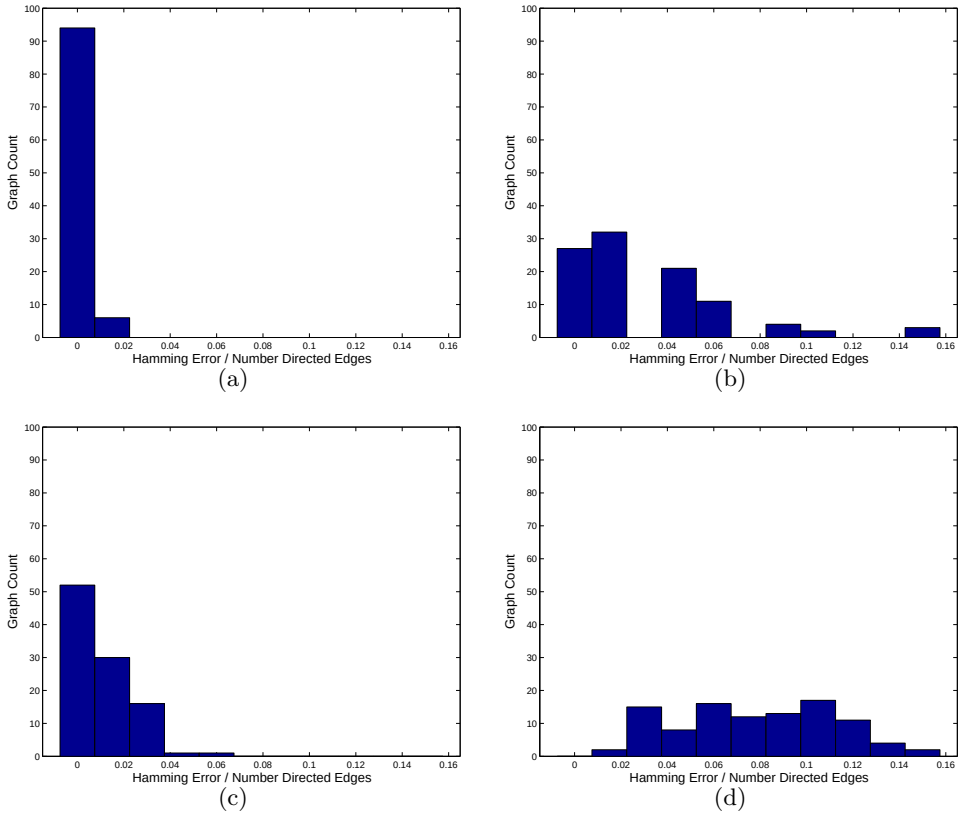


FIGURE 3.8. A histogram of Hamming error per edge using the simulator with 8000 observations on 100 randomly produced graphs for: a) 12 nodes and 4 agents, b) 12 nodes and 10 agents, c) 20 nodes and 4 agents, and d) 20 nodes and 10 agents. A directed edge to vertex ratio of $4:1$ was selected for the random graphs used in these experiments.

4.2.1. *Convergence and Implementation Assessment.* Recall that at each iteration of the algorithm, we gather a number of samples of the ownership vector using MCMC. These samples are discarded until a burn-in period, and are then collected and used to represent plausible agent trajectories through the environment given the belief of the network parameters computed during the last iteration. The network parameters are then updated to maximize the likelihood these of trajectories and a new iteration begins. This process continues until the updated parameters are similar enough to those found during the last iteration. See Section 2.1 for more details.

In most cases, we found that the algorithm converged quickly, finding most of the coarse structure of the graph in the first few iterations and making incrementally smaller changes

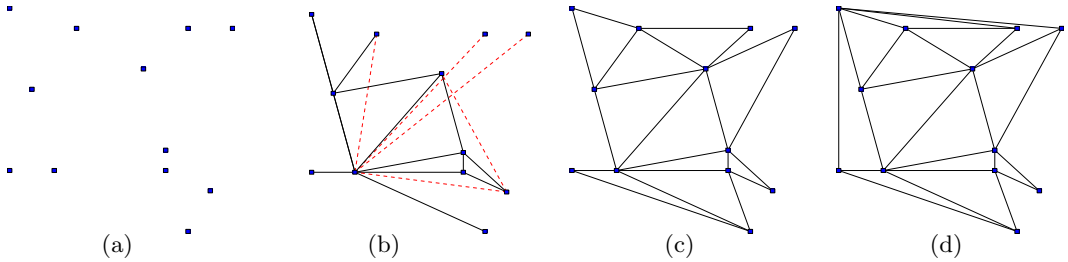


FIGURE 3.9. Incremental belief of the topology of a 12 node, 48 (directed) edge graph using 4 simulated agents on 8000 observations: a) initially b) after 1 iteration, c) after 2 iterations, d) after 3 iterations (the true graph). Dotted lines indicate incorrect transitions.

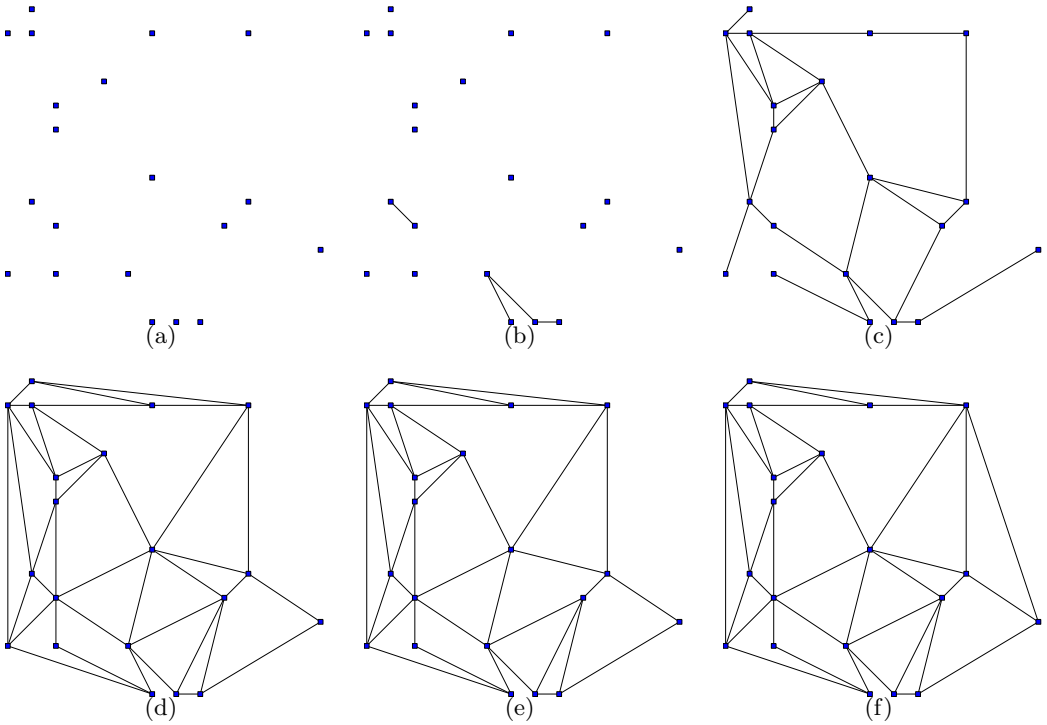


FIGURE 3.10. Incremental belief of the topology of a 20 node, 80 (directed) edge graph using 4 simulated agents on 8000 observations: a) initially b) after 1 iteration, c) after 2 iterations, d) after 3 iterations e) after 4 iterations f) after 5 iterations (the true graph).

until convergence (Figures 3.9, 3.10). After every new iteration of the MCEM process, the set of sampled ownership vectors generally increased in likelihood on average in comparison with samples gathered during the last iteration. An example of the progression in sample likelihood for a simulation can be seen in Figure 3.11. It can be noticed that most of the

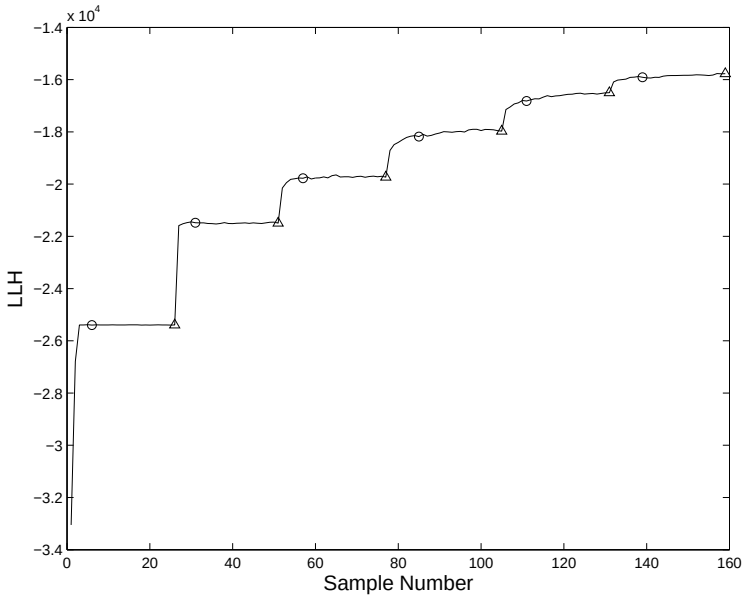


FIGURE 3.11. The log likelihood of samples of the ownership vector for an example run of the algorithm using 4 simulated agents on a 12 node, 48 edge random graph with 4000 observations, (same graph as for Figure 3.12). The horizontal axis gives the sample number (across all iterations). For each iteration, only the samples shown between the circle and the triangle are used for updating network parameters (the M Step).

likelihood gain occurred in the first few steps of an iteration, which was typical in these experiments. Figure 3.12 gives a closer look at how the sample likelihood progressed *with in* each iteration for the same example. As the number of iterations increased, the change in likelihood immediately after a parameter update decreased.

There appears to be a tradeoff between frequent parameter updates based on a small number of trajectory samples and a smaller number of updates each based on a large number of samples. In terms of the underlying EM algorithm, the number of samples (K) taken before each parameter update (M Step), corresponds to how much effort is spent on estimating the expected value of the log likelihood of the complete data (E Step). Modifying the number of samples K of the ownership vector drawn during each iteration affected the performance of the algorithm; (see Section 2.1). As the value of K was increased, the convergence time increased and the error of the final solution decreased (Table 3.1, Figure 3.13). For the easier problems, however, frequent parameter updates resulted in the algorithm terminating quickly with zero Hamming error. For example, in the 4 agent

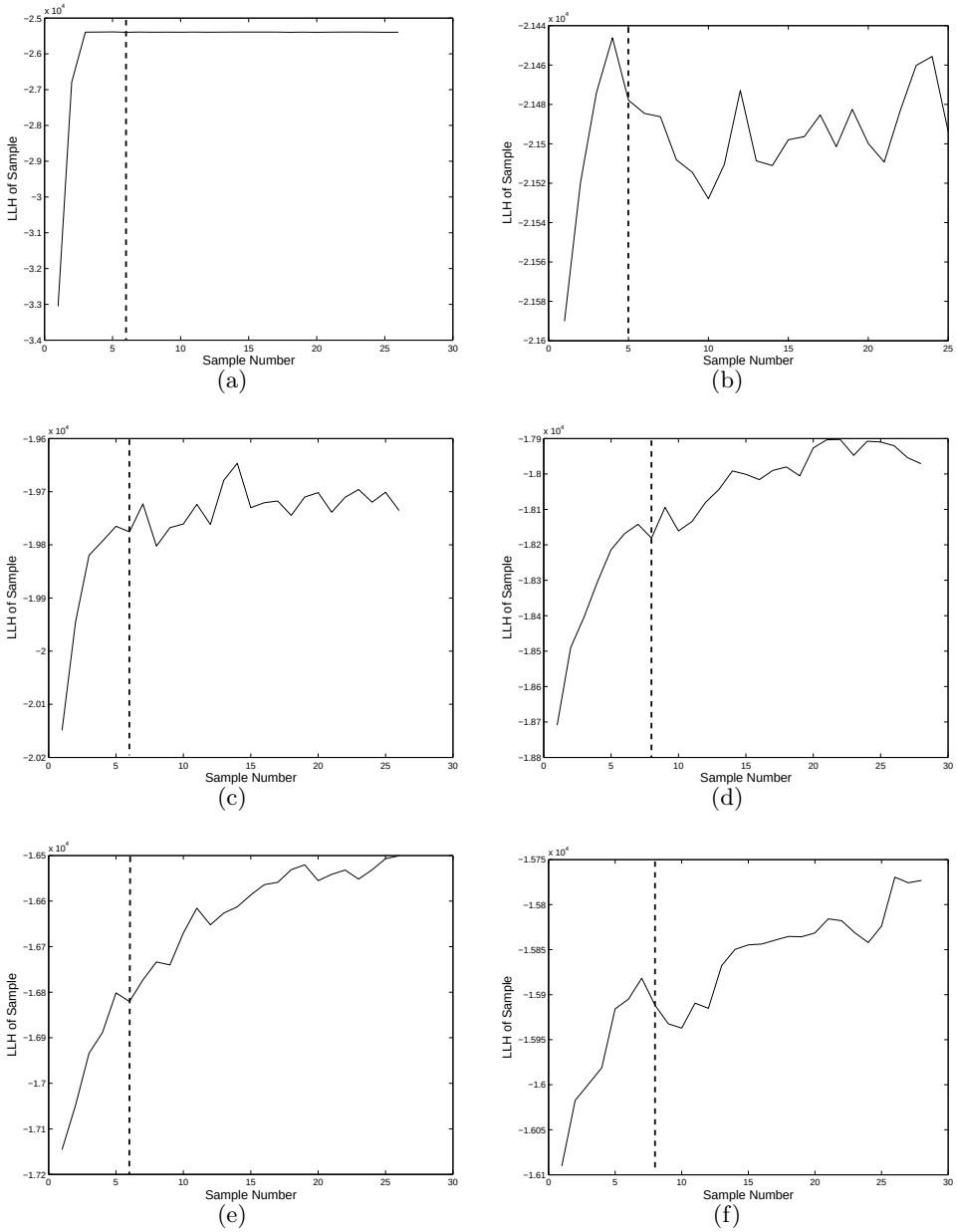


FIGURE 3.12. The log likelihood of samples of the ownership vector for each iteration of the algorithm: a) initially, b) after 1 iteration, c) after 2 iterations, d) after 3 iterations, e) after 4 iterations, f) after 5 iterations. The results were produced using 4 simulated agents on a 12 node, 48 edge random graph with 4000 observations ($K = 20$). The horizontal axis indicates the sample number for each iteration. The dotted horizontal line indicates the heuristic-estimated burn-in position (see Section 2.1). Samples taken after this point in each iteration are used in parameter updates.

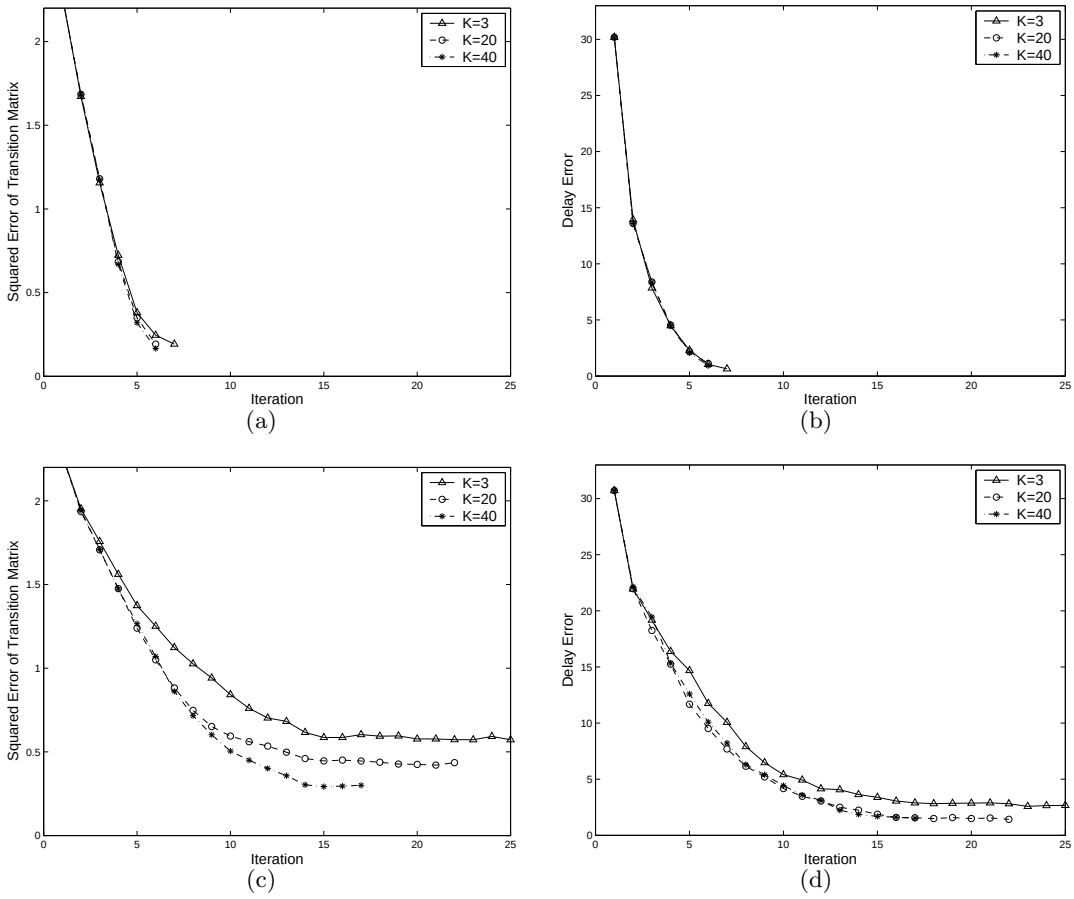


FIGURE 3.13. A comparison of algorithm performance per iteration as a function of K. Results were obtained using the simulator on a 12 node, 48 edge random graph with 4000 observations with: a) and b) 4 agents; c) and d) 10 agents.

case shown in Table 3.1, the algorithm terminated with less than one quarter of the samples required when K was assigned a value of 3 as opposed to 40 and the final squared error of the transition matrix was less than 0.2 (resulting in a Hamming error of zero), regardless of the K value used. Presumably, this result was because the Markov chain both quickly reached the stationary distribution, and also because the distribution was easy to characterize with only a few samples. It seemed that the more difficult problems, however, such as those involving a large number of agents, required a greater effort during each iterative E Step in order to produce accurate results (Figures 3.13(c), 3.13(d)).

Eventually, we will look at finding a method of automating the effort placed in each iteration based on an analysis of the likelihood trends of the sampled ownership vectors.

	4 agents			10 agents		
K	**Err_A**	**Err_D**	**Total Samples**	**Err_A**	**Err_D**	**Total Samples**
3	0.187	0.592	70.5	0.555	3.470	221.4
20	0.141	0.593	179.1	0.399	2.747	424.8
40	0.121	0.454	314.3	0.381	3.011	630.4

TABLE 3.1. Comparison of performance and computational effort until convergence as a function of K averaged over 10 graphs of 12 nodes, 24 edges.

However, for the moment, we currently set the number of samples K used in each iteration to an experimentally determined intermediate value. For the remainder of this thesis, all runs of the topology inference algorithm are conducted with $K = 20$.

4.2.2. *Comparison to Existing Method.* In a number of experiments, we compared our algorithm for topology inference to an implementation of the threshold-based approach presented by Ellis *et al.* in [**37**]. Figure 3.14 shows a histogram comparing the outcome 100 trials of the two approaches for two different problem types. In these experiments the threshold-based method did not perform as well as the method described in this thesis. For 100 trials on 20 node graphs with 10 agents, our algorithm achieved a mean Hamming error per directed edge of more than four times lower than the threshold-based method. A similar performance difference was seen for 100 trials on 12 node, 10 agent simulations. Furthermore, for each individual trial conducted in this experiment, our algorithm achieved an equal or lower error value than the threshold-based approach. Although shown to be less accurate in our simulations, the heuristic threshold-based approach of Ellis *et al.* is very fast and does not need to make an assumption regarding the number of agents in the system.

4.2.3. *Significance of Graph Size and the Number of Agents.* A critical parameter is the number of agents moving in the system relative to the number of vertices. Clearly, under noise free conditions, if there is only one agent in the network the problem is straightforward since its event sequence can simply be "traced out". However, in the case of multiple agents, the events generated by a given agent's movements in the network risk being incorrectly associated with those of any other agents'. It is the relative density of the correct pairings relative to the incorrect ones that makes the problem more or less easy to solve.

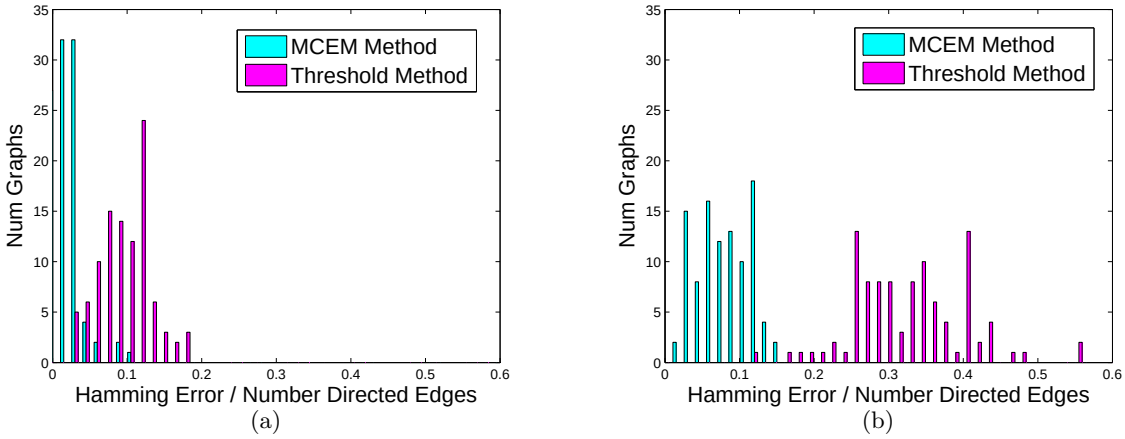


FIGURE 3.14. Histograms of Hamming error per edge using both the threshold method described by Ellis *et al.* [**37**] and our MCEM method. The techniques were tested using 10 simulated agents with 8000 observations on 100 randomly produced graphs of size: a) 12 nodes, 48 edges; and b) 20 nodes, 80 edges.

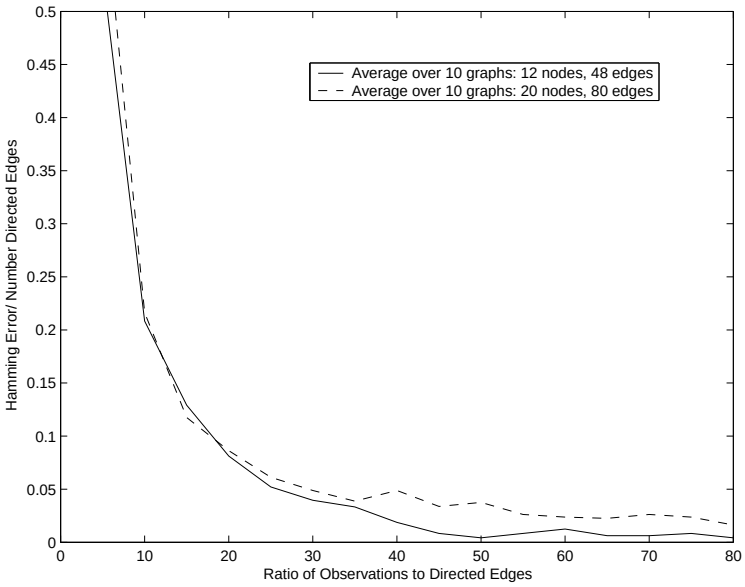


FIGURE 3.15. Hamming error per edge as a function of the ratio of observations to true (directed) edges using 4 simulated agents.

In our experiments, we found that increasing either the number of agents present in the environment or the size of the graph made the problem more difficult to solve, albeit for rather different reasons. While increasing the number of agents allowed a greater number of probable trajectories, and was analogous to decreasing the signal to noise ratio in the

system, increasing the graph size while holding the number of observations steady reduced the expected number of observations per edge in the graph. Experiments support the idea that the accuracy of our approach increases for a particular number of agents when the ratio of observations to edges is increased (Figure 3.15). In the extreme case, if there are some edges that have no observations recorded along them at all, our approach will not have enough information to infer the correct graph. At the minimum, an observed agent must traverse each edge at least once.

4.3. Effects of Observational Noise. While the algorithm is robust to moderate levels of 'white' observational noise, its sensitivity to systematic noise depends on the tuning of the delay model. (See Section 4.1 for a description of the simulated noise.) The delay model is controlled by the SSL parameter which determines the probability threshold for including delay data in the update of the network connectivity parameters; (see Section 2.3). For purposes of brevity, the value assigned to the SSL parameter is reported in natural logarithm form in the results we present here.[2] Figure 3.16 shows the result of varying the value assigned to the SSL parameter for different types of noise. Figure 3.17 shows the ability of the delay model to successfully identify and discard low probability transitions and explain them as transitions to and from the source/sink node.

When assigned a high SSL value, the use of a mixture model for modeling delays was successful at minimizing the effects of systematic noise. When 10 *per cent* of the transitions were perturbed by large delay errors, the Hamming error of the inferred transition matrix was near zero (Figure 3.16(a)). The reduction in error for inferred mean delay times was especially dramatic in comparison to results obtained with a low SSL value (Figure 3.16(b)). When the SSL parameter was assigned a value of zero, the algorithm had no method of discarding spurious delay data and had to update its network parameters given all the observations. Hence, it was heavily effected by biased delay times. The vast improvement in estimates of mean delay times for simulations with a reasonably selected SSL parameter values demonstrates the ability of the delay model to successfully identify and discard the 'non-through-traffic' data; *i.e.* data which was not apparently generated by agent motion between nodes of the graph.

[2]For example, instead of SSL=0, we report ln(SSL)=-inf.)

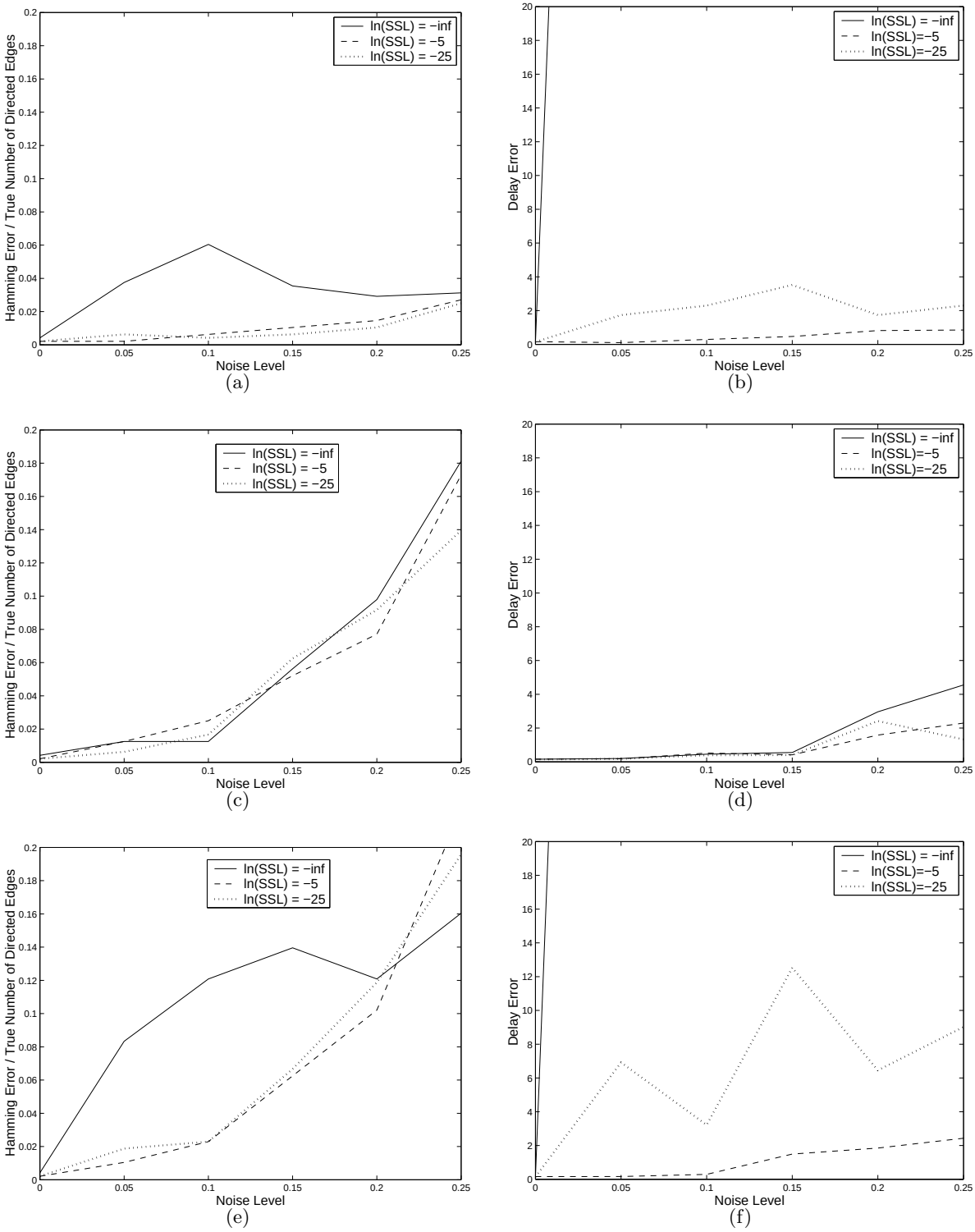


FIGURE 3.16. Hamming and delay error as a function of observational noise. The results are averaged over 10 graphs using 4 simulated agents on 12 node, 48 edge graphs with 4000 observations. The horizontal axis indicates proportions of: a,b) systematic noise; c,d) white noise; e,f) both systematic and white noise.

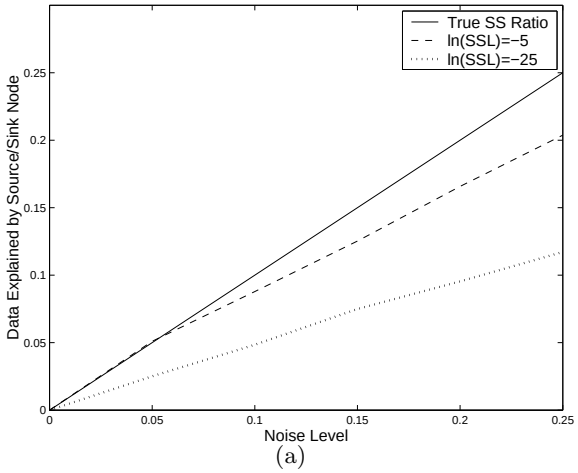
It seemed that moderate amounts of 'white', un-biased observational noise can be handled by the algorithm regardless of the tuning of the delay distribution mixture. (Figures 3.16(d), 3.16(c)). It was the inferred transition belief and not the mean delay times that were most effected by large amounts of this type of noise. This is because the effect of randomly inserting and deleting observations is to skew the distribution of likely sampled trajectories. Hence, the inference technique develops an incorrect belief of the underlying network and its inter-sensor transition probabilities. Since determining the correlation between the various sensor observations is key to our approach, it is unsurprising that after about 10 *per cent* of both missing and spurious observations the performance of the algorithm drops significantly.

When moderate levels of both types of noise were present, the delay model was still able to reduce the effects of the biased delay data (Figures 3.16(f), 3.16(e)). However, this ability seems to decrease as the noise level is increased. This effect can be seen in Figure 3.16(e) where at extreme levels of noise, *i.e.* 25 *per cent* of both white and systematic noise, the best performance was actually obtained with a SSL value of zero. As the white noise was increased along with the systematic noise it became harder for the algorithm to distinguish between the two types of noise. The distribution of delay times flattened out and hence a larger proportion of the data was better fit to the uniform distribution than to the inter-vertex distribution of the delay mixture model.[3] Under these conditions, the algorithm had difficulty identifying peaks or correlations in the delay data and incorrectly assumed additional transitions to and from sources and sinks; (compare Figures 3.17(c) and 3.17(a)). Hence, the method had less data with which to determine the relative strengths of the inter-sensor transitions.

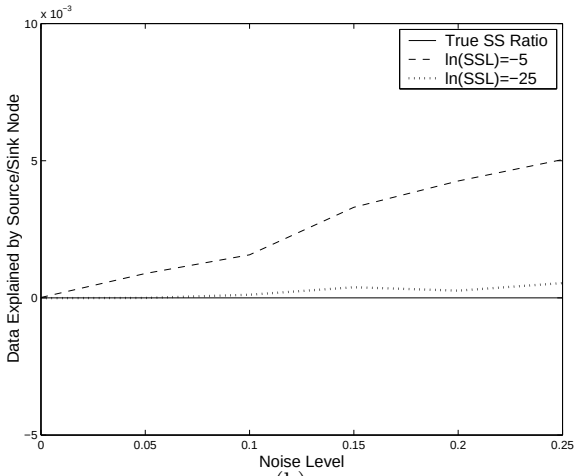
The robust behavior of the algorithm under noisy conditions demonstrates both the general stability of the sampling-based approach and the success of the delay model. With an appropriately selected value assigned to the SSL parameter, the technique can infer highly accurate connectivity information even with moderate levels of both systematic and white noise.

4.4. Automatic Parameter Selection (Level Two). In this section, we attempt to validate our general approach for selecting nearly optimal input parameters for the first level topology inference algorithm by assessing the quality of the solution it produces.

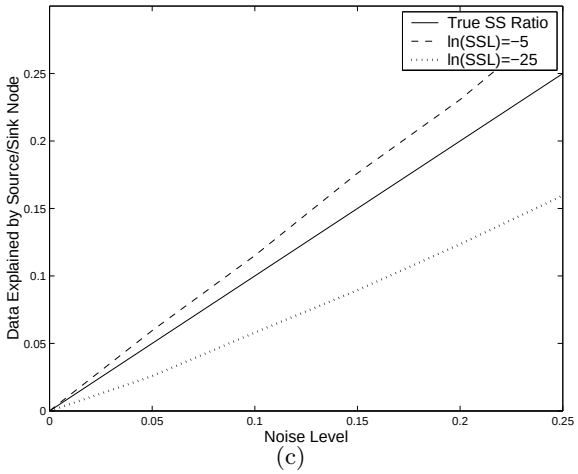
[3] A Guassian distribution was employed for these experiments.



(a)



(b)



(c)

FIGURE 3.17. A plot of the proportion of delay data rejected as a function of observational noise. The results were averaged over 10 graphs using 4 simulated agents on 12 node, 48 edge graphs with 4000 observations. The horizontal axis indicates proportions of: a) systematic noise; b) white noise; c) both systematic and white noise.

β	γ	τ	κ	λ
2	0.9	0.2	2	1

TABLE 3.2. Table of values used to shape the simplicity quotient Q_{simp}. (See Section 3 for the definitions of these parameters.)

We select parameters defining the Q_{simp} metric based both on domain knowledge and experimental methods (Table 3.2). In order to determine these parameter values and to assess the effectiveness of this approach, we conducted a number of simulations in which we varied the input parameters and looked for a correlation between the performance of the algorithm and the simplicity metric.

4.4.1. *The Effect of Input Parameters.* Input parameters that resulted in good algorithm performance also resulted in solutions that generated high Q_{simp} quotient values. Figure 3.18 shows the mean error and corresponding Q_{simp} value obtained as a result of running the fundamental topology inference algorithm with different inputs. In can be seen in this experiment that the lowest error was obtained when the assumed number of agents in the system was set to the value of four (Figure 3.18(a) and Figure 3.18(c)). Likewise, a value of four for the assumed number of agents in the system resulted in the highest calculated value for the Q_{simp} quotient (Figure 3.18(e)). A similar relationship can be seen between the value selected for the SSL parameter, (Figure 3.18(b) and Figure 3.18(d)), and the corresponding Q_{simp} quotient (Figure 3.18(f)). In general, we observed that under noise free operation, the most accurate solutions also generated the highest Q_{simp} values. This result gives support for our adoption of Occam's Razor as a mechanism for selecting input parameters.

The accuracy of the solution we obtain depends heavily on the assumed number of agents in the environment. The lowest error was consistently observed when the assumed number of agents was set to the actual value, and generally, the closer to the correct value this parameter was set, the better the results. Over-estimating the assumed number of agents greatly impacted the accuracy of the estimated mean delay, but had less effect on the accuracy of the inferred transition matrix. The opposite effect occurred when the value for this parameter was underestimated.

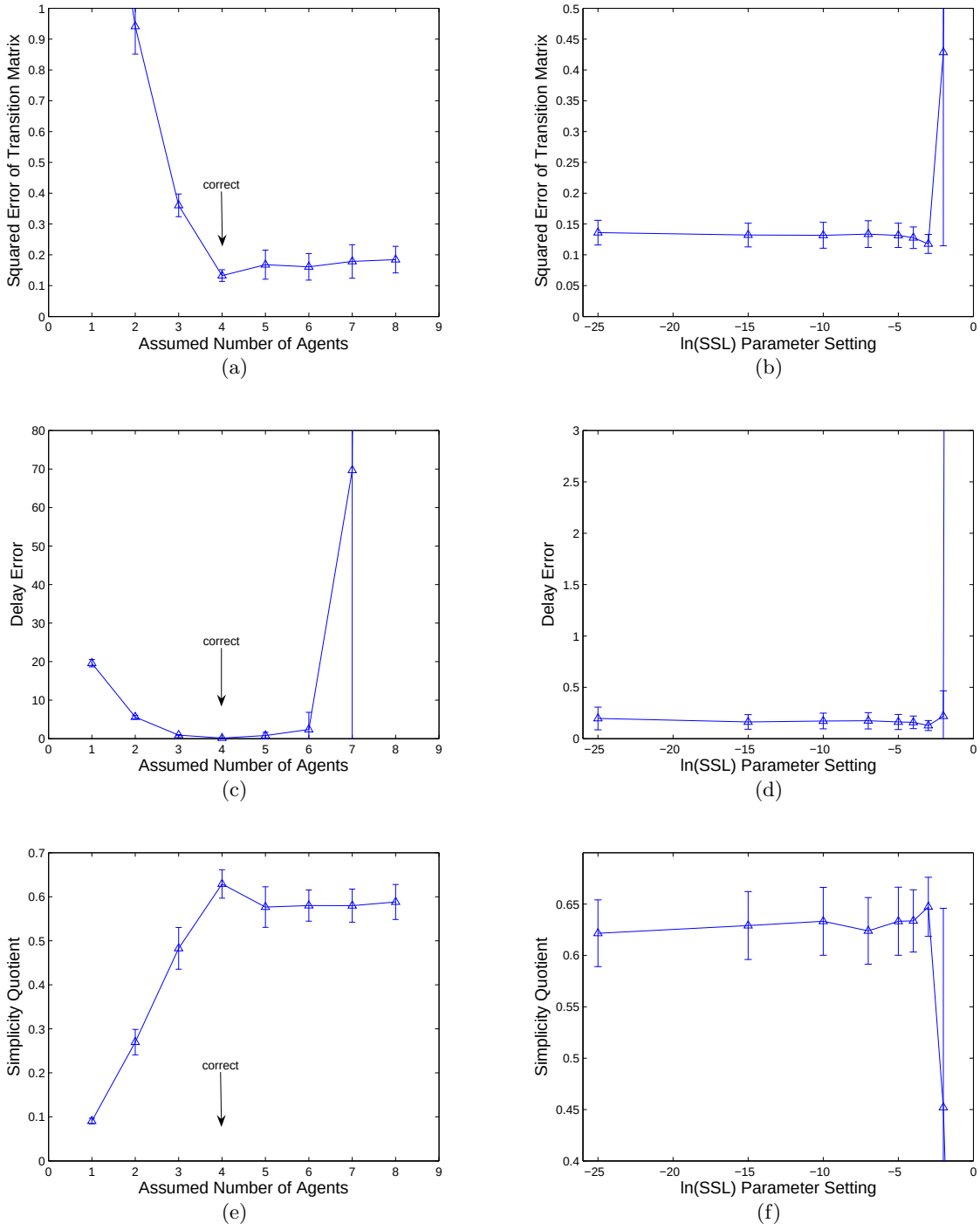


FIGURE 3.18. The effect of varying assumed numbers of agents and the value of the SSL parameter on performance and the simplicity quotient. Results are averaged over 20 graphs using 4 simulated agents on 12 node, 48 edge graphs with 4000 observations; error bars show one standard deviation.

A correctly tuned SSL parameter was also important to the accuracy of the final solution. As the value for this parameter was increased, there appeared to be a "phase transition" in the accuracy of the results. Past a certain threshold, the error suddenly increased dramatically. Interestingly, the best results for both the inferred mean delay times and transition likelihoods seems to be obtained just before this sudden degradation in performance; *e.g.* see Figure 3.18(b) and Figure 3.18(d).

4.4.2. *Direct Correlation between Performance and the Simplicity Quotient.* When the error in the inferred transition matrix was plotted against the value obtained for the simplicity quotient Q_{simp} for a number of simulations, there was evidence of a definite correspondence (Figure 3.19). The effect appeared robust to moderate levels of observational noise and different sizes of graphs. While, the shaping of the Q_{simp} metric is ongoing work, the current parameter values are adequate to demonstrate the correlation between the correctness and simplicity of the inferred transition matrix. In our experimental work, described in the next section, we took advantage of this correlation to select appropriate input parameters since the 'correct' values were unknown.

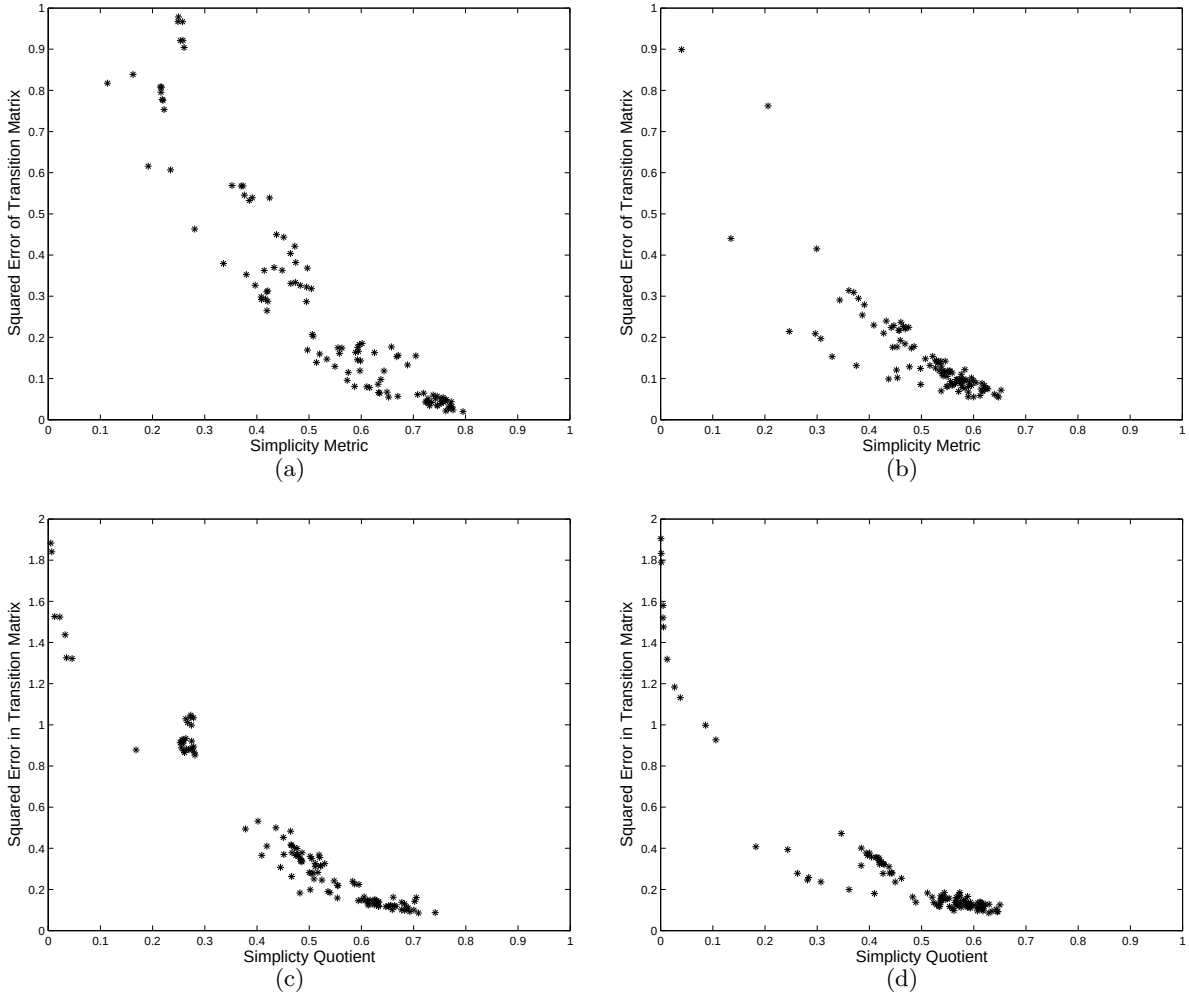
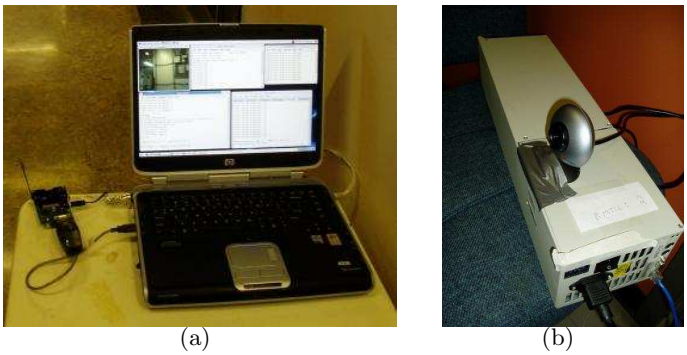


FIGURE 3.19. The mean error in the inferred transition matrix elements plotted against Q_{simp} for data obtained from the simulator with 4 true agents. Input parameters to the algorithm were varied: assumed number of agents from 2 to 7; and $ln(SSL)$ from -2 to -7. The results are obtained using the simulator on: a) 4 random graphs of 6 nodes, 14 edges with 2000 noise-free observations (144 trials); b) 4 random graphs of 6 nodes, 14 edges with 2000 observations containing 5 *per cent* white and systematic noise (144 trials); c) 4 random graphs of 12 nodes, 48 edges with 4000 noise-free observations (144 trials); d) 4 random graphs of 12 nodes, 48 edges with 4000 observations containing 5 *per cent* white and systematic noise (144 trials). Observe that the solutions obtaining high simplicity quotient values are consistently among those with the lowest transition matrix error.

5. Experiments Conducted on a Heterogeneous Sensor Network

To assess the performance of our technique under real-world conditions, we conducted experiments using a sensor network deployed in an office building. In this section we first



(a) (b)

FIGURE 3.20. Laptop used as the central server and an example of a vision-based sensor node.

describe some system and implementation details and then present results from two experiments. As part of this project we discuss custom data acquisition and analysis software for a Linux network and as firmware for embedded systems.

5.1. System Description. The sensor network was made up of two types of devices: vision-based sensors running on PC hardware; and photocell-based sensors running on low-powered commercial devices. Both types of sensors were programmed to act as simple motion detectors sending event messages to a central server, which logged the origin and time of the activity (Figure 3.20).

The vision based sensor nodes were constructed of inexpensive PC hardware networked together over Ethernet using custom software. A single node consisted of a 352x292 pixel resolution Labtech USB webcam connected to a Flexstar PEGASUS single board computer. The operating system used was Redhat Linux based on kernel 2.4 (Figure 3.20). The sensor nodes contained an Intel Celeron 500Hhz CPU and 128 MB of RAM. They were disk-less and had to netboot from a central server which they were connected to either *via* a wireless bridge or a standard Ethernet cable.

A standard client/server architecture was implemented over TCP/IP using linux sockets in the C language. Each sensor runs an identical copy of the client program while a single copy of the server application runs on a central computer.

The client software functions as a motion detector based on the Labtech webcam. During an initial period, a background image is captured from the camera and the method for triggering an event detection is calibrated. An intensity threshold is calibrated for

each colour channel by calculating the standard deviation from the background based on a number of captured frames:

$$\theta_c = C * std\{f_0 - \Gamma, \ldots, f_n - \Gamma\}$$

where f is a captured frame, Γ is the background frame and C is a constant determining the sensitivity of the system. The sensor then enters an armed state in which captured frames are compared to the background image, and any difference exceeding the threshold triggers a detection event (Figure 3.21). A frame rate of approximately 10Hz is obtained. Once triggered, the sensor re-arms itself after a couple of seconds of inactivity. The background is slowly updated to account for gradual changes in the scene; *e.g.* changes in lighting or a re-positioned object such as a door:

$$\Gamma' = \alpha * f + (1 - \alpha) * \Gamma$$

where α is a constant determining how quickly the background is updated.

Events are transmitted over TCP/IP to a central server where they are time-stamped and logged for offline analysis. The server is multi-threaded and allows control of the system through a command line interface. In addition to detection events, the application allows either a full resolution capture or a low-resolution streaming of images from any sensor to the server.

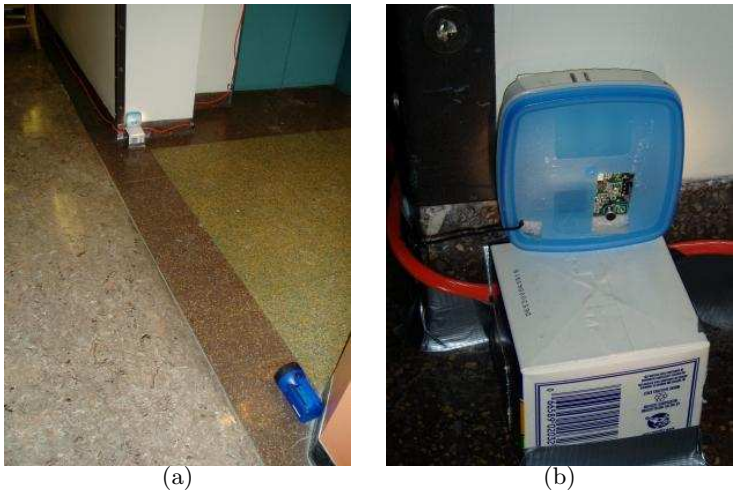
Offline experimentation suggested that while a correctly calibrated vision-based sensors rarely missed events, it occasionally generated false positives. Changes in brightness sometimes triggered subsequent events following a real event.

Additionally, lighting and contrast conditions during the calibration of the pixel intensity thresholds were important factors. A very uniform background scene could result in a threshold value that was too high to detect subtle activities. However, the sensors generally calibrated effectively, and hence performed well when they were given a background image containing varied colors.

5.1.1. *Low-powered Photo-cell Based Sensors.*

(a) (b)

FIGURE 3.21. An example of images captured from a vision sensor: a) the background image; b) a frame triggering an event detection.

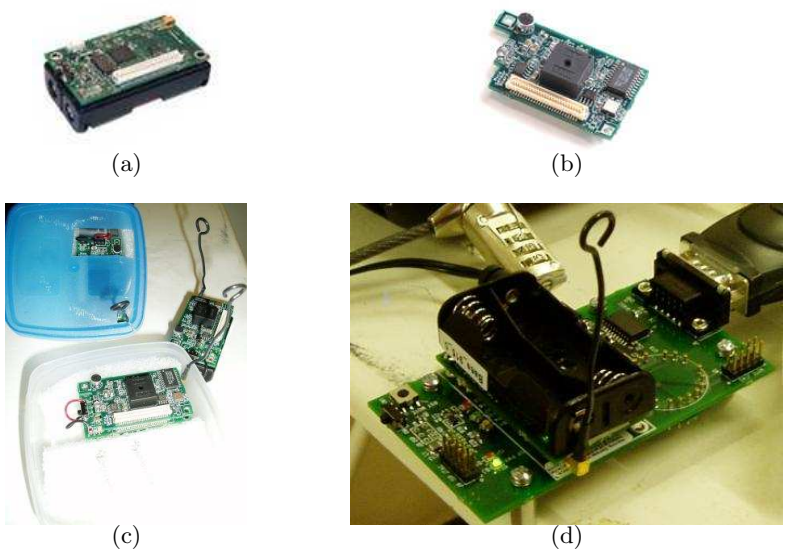


(a) (b)

FIGURE 3.22. a) Complete setup and, b) close up of a deployed photocell-based sensor constructed out of a flashlight and a Crossbow wireless sensor. (Plastic containers were used as protective covering during experiments.)

The second type of motion sensor consisted of of a flashlight and a photocell equipped low-powered device (Figure 3.22). The flashlight beam was focused on the photocell. Any decrease in the intensity of the light was detected by custom firmware which sent an event message to a central server (Figure 3.24).

The low-power devices used were MICA2 Crossbow wireless sensors with MTS310CA sensor boards (Figure 3.23). The CPU on this model was an 8 bit Atmel ATmega128L with 4K of RAM and 128K of flash memory. The devices were equipped for low-data rate (19.2K baud) RF communication on the 916 MHz band. A single MICA2 mote connected to a MIB510 Serial Interface Board was used as a base-station to communicate events back



(a) (b)

(c) (d)

FIGURE 3.23. Crossbow hardware used in the experiments: a) MICA2 Processor/Radio module (image from http://www.xbow.com); b) MICA2 Multi-Senor Module (image from http://www.xbow.com); a) MICA2 motes with plastic containers used as a protective casing; b) base-station used to communicate to the central server over a serial port.

to the central server (Figure 3.23(d)). The photocell was one of several sensors provided which could be sampled by an 8 channel 10 bit Analog-to-Digital converter.

The MICA2 Crossbow motes come with a software development package that includes a RTOS called TinyOS [**63, 41**] written in necC [**42**]. TinyOS and the nesC language provide a component-based event driven framework for developing networked embedded wireless applications. They were both originated at the University of California at Berkeley [**50**], and the source code and software are publicly available online.[4]

TinyOS is a small, energy efficient, soft real-time operating system. The kernel implements a two-level priority scheme. A round robin scheduler is provided for low priority tasks which are interrupted by higher priority asynchronous events whenever they occur. Idle CPU cycles are automatically spent in a sleep mode leading to efficient power usage. TinyOS includes a multi-hop communication protocol and other components specifically designed for sensor network projects.

[4]http://www.tinyos.net/

FIGURE 3.24. An example of motion triggering a detection event by the photocell-based sensor.

The language necC is a modified version of C which formalizes an event driven and component-based approach to firmware development. TinyOS, although originally written in C, was constructed in this component-based fashion and was later re-written in necC. Programs written in necC are constructed of *components* which are wired together through *interfaces*. The components contain the actual functional code, while the interfaces define a two way flow of control and data between the components. Interfaces specify *commands* and *events* which, in C, are similar to standard function calls and call-back routines assigned through function pointers. For one component to be wired to another, they must share a common interface. Development using necC can be relatively fast since it is easy to wire together a new application using existing components.

MICA2 Crossbow sensors running TinyOS are commonly employed in sensor network research. They have been used in work ranging from monitoring applications [**67**] to robot navigation [**6**] and information propagation investigations [**64**].

The firmware used on the MICA2 devices was developed for this project and implements a motion detector using the photocell. During sensing operation, the photocell is sampled at roughly 800 Hz. Any significant reduction in light intensity compared to a previously calibrated background level triggers an event. Events are sent to the central server via the TinyOS multi-hop communication protocol where they are time-stamped and logged for offline analysis. The multi-hop routing for the nodes is established before the experiment begins.

Experimentation suggests that the photocell-based sensors are extremely reliable at detecting events, but have several weaknesses. It is possible for an event to occur during the moment that the device is re-calibrating the photocell. This will disrupt the calibration

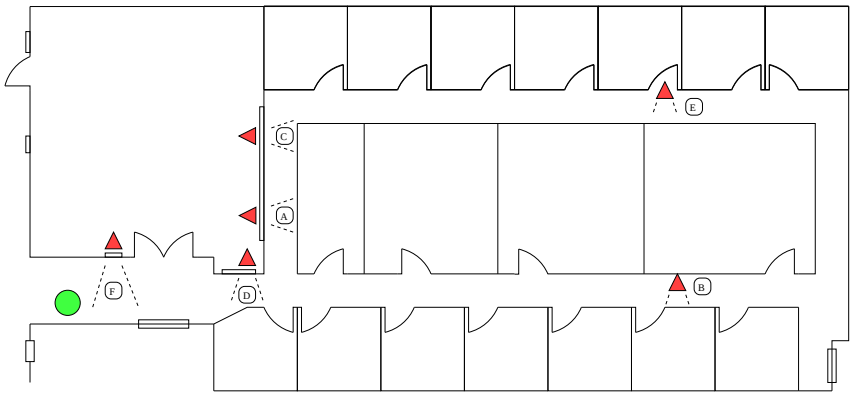


FIGURE 3.25. The layout of the six camera sensor network used for experiment. Labeled triangles represent sensor positions, and the circle represents the location of the central server.

process and could result in an inability to sense events until the next calibration, a period of roughly one minute.

A more serious limitation is the poor communication range. At distances beyond about 8 or 9 metres transmission reliability decreases significantly and often results in lost packets, and therefore, missed event detections. It is likely that the level of interference is much higher than normal in the research building where the experiments were conducted due to the large amount of wiring, wireless subnets, and electronic equipment.

5.2. Experiment with a Six Node Vision-Based Sensor Network. As the first test of our technique under real-world conditions, we setup an experiment using a medium sized network of vision-based sensors.

5.2.1. *Data Collection.* The experiment was conducted in the hallways of one wing of an office building (Figure 3.25). The data were collected during a typical weekday for a period of five hours from 10:00am to 2:30 pm. In addition to the normal traffic one or two subjects were encouraged to stroll about the region from time to time during the collection period in order to increase the density of observations. A total of approximately 1800 events were collected.

5.2.2. *Ground Truth.* Ground truth values were calculated in order to assess the results inferred by the approach. A topological map of the environment (Figure 3.27(a)) was determined based on an analysis of the sensor network layout shown in Figure 3.25. (Note that we have not attempted to analytically determine reasonable connections to

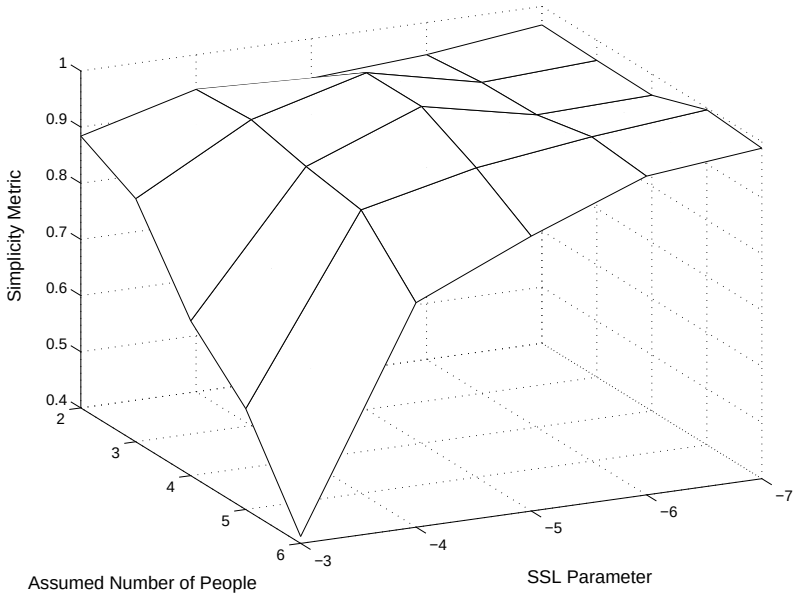


FIGURE 3.26. A plot of the Q_{simp} metric as a function of input parameters.

sources or sinks in the environment.) Additionally, inter-vertex transition times for the connected sensors were recorded with a stopwatch for a typical subject walking at a normal speed (Table 3.4).

5.2.3. *Selection of Input Parameters.* In order to determine appropriate input parameters for our inference algorithm we conducted an exhaustive search over a range of possible values (layer two of the two-layer approach described in Section 3). We ran the first level topology inference algorithm on the experimentally collected data for inputs varying from $N = 2, .., 6$ and $ln(SSL) = -7, .., -3$ (Figure 3.26). We then selected the output values that maximized our Q_{simp} metric; *i.e.* producing the simplest model. (We used the same shaping parameters for the Q_{simp} metric that were verified through simulations; see Section 4.4.) The parameters that provided this optimum were: $N = 4$ and $ln(SSL) = -5$.

5.2.4. *Assessment of Results.* The network parameters inferred by our topology inference algorithm closely corresponded to the ground truth values. Table 3.3 shows the transition matrix output by the algorithm, and Figure 3.27 compares the analytically determined and inferred topological maps. Disregarding reflexive links, the difference between the inferred and determined matrices amounts to a Hamming error of 1. The inferred connection from D to B was not given a transition probability large enough to be detected

	A	**B**	**C**	**D**	**E**	**F**	**SS**
A	0.05	**0.16**	**0.28**	**0.32**	0.02	0.09	0.08
B	**0.28**	0.08	0.01	**0.12**	**0.41**	0.04	0.06
C	**0.40**	0.05	0.05	0.05	**0.32**	0.05	0.08
D	**0.22**	0.08	0.05	0.07	0.01	**0.43**	**0.13**
E	0.04	**0.39**	**0.40**	0.04	0.04	0.03	0.06
F	0.06	0.03	0.08	**0.34**	0.00	**0.28**	**0.22**
SS	0.08	0.07	0.09	**0.25**	0.03	**0.49**	0.00

TABLE 3.3. The transition matrix inferred from the experimental data. SS refers to the source/sink node introduced by the algorithm. Bold values over the threshold $\theta = 0.1$ are interpreted as one way edges. The underlined values were not directly predicted by the ground truth analysis.

Connection	**Timed**	**Inferred**
A,B	16	15 / 16
A,C	3	3 / 3
A,D	4	3 / 3
B,D	15	16 / 17
B,E	16	15 / 15
C,E	14	15 / 14
D,F	5	5 / 3

TABLE 3.4. A comparison of timed and inferred delay times (both ways) between sensors. All values are rounded to the nearest second.

based on our thresholding technique. However, the opposite edge from B to D was correctly inferred. Of course, it would be easy to build into the algorithm the assumption that all edges must be two ways. A strong belief in an edge in one direction would dictate that the opposite edge must also exist.

The mean transition times produced by the algorithm were also consistent to those determined by stopwatch (Table 3.4). Some examples of inferred delay distributions are shown in Figure 3.28.

Sensor F marks the only heavily used entrance and exit to the region monitored by the network. The self-connection inferred to this node is due to a detected correlation in the delay between exit times and subsequent re-entry times for agent motion. In fact, this correlation is due to the tendency of subjects to re-enter the system after roughly the same

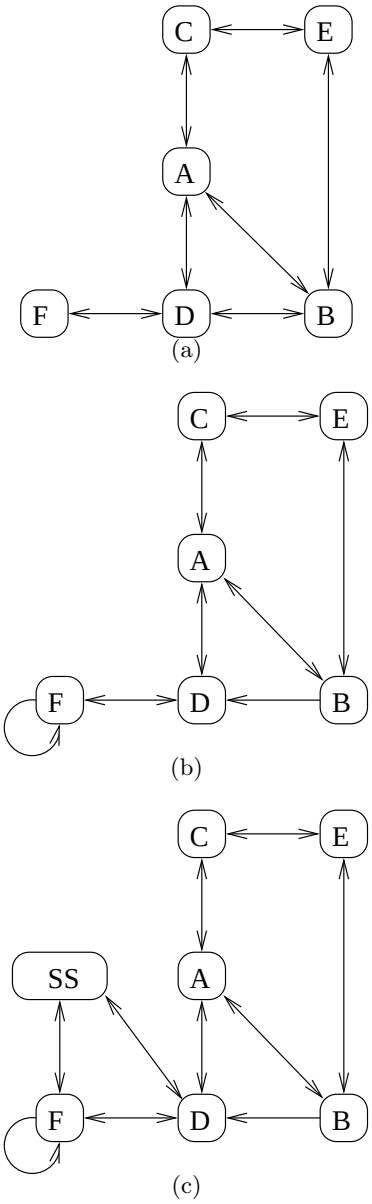


FIGURE 3.27. Topological maps of the environment that were: a) analytically determined based on the layout; b) inferred by the algorithm; c) inferred by the algorithm including the source/sink node.

time period (*e.g.* to use the washroom or photocopier). Therefore, the detection of this connection was actually a correct inference on the part of the algorithm.

It is interesting to note that two-way connections were inferred to the source/sink node from both sensors D and F (Figure 3.27(c)). It was possible for subjects to pass by either

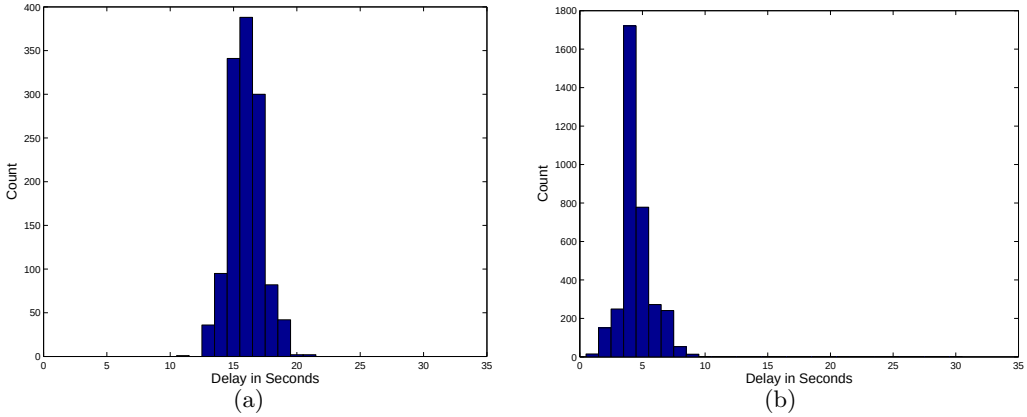


FIGURE 3.28. Two examples of delay distributions inferred for: a) sensor A to sensor B; b) sensor D to sensor F.

of these sensors on their way into or out of the monitored region. (The exit to the far right of the area, shown in Figure 3.25, was little used.) This demonstrates the function of the source/sink node as a method for the algorithm to explain sudden appearances and disappearance of agents in the system.

5.3. Results from a Nine Sensor Heterogeneous Network. In our second experiment, we tested the performance of our technique on a larger sensor network that contained both vision-based sensors and the smaller photocell-based sensors.

5.3.1. *Data Collection.* In a manner similar to the first experiment, a sensor network was set up in the hallways of one wing of an office building (Figure 3.29). The data were collected during a six and a half hour period from 10:00am to 4:30 pm on a weekday. Like the first experiment, subjects were encouraged to stroll about the region from time to time during the collection period in order to increase the density of observations. In total, approximately 4700 time stamped events were collected.

The three low-powered sensors were placed close to the central server to accommodate their shorter communication range. Despite this layout, the furthest low-powered sensor, I, was only able to communicate to the central server *via* intermediate sensor, H, using the multi-hop protocol.

5.3.2. *Ground Truth.* Like the previous experiment, ground truth values were calculated in order to assess the results inferred by the approach. A topological map of the environment was determined (Figure 3.31(a)) based on an analysis of the sensor network

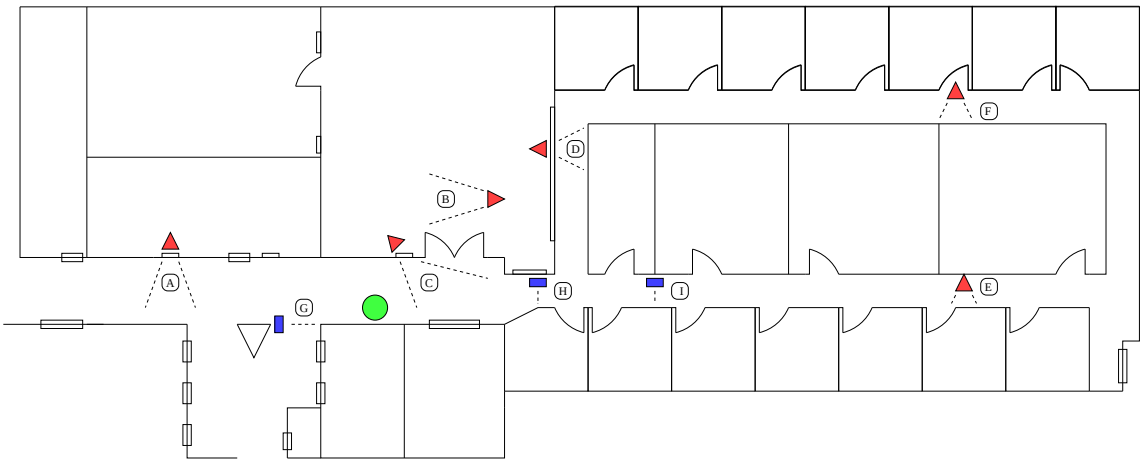


FIGURE 3.29. The layout of the nine senor (heterogeneous) network used for the experiment. Labeled triangles represent vision-based sensor positions (A-F) and labeled rectangles represent low-powered photo-based sensors (G-I). The circle represents the location of the central server.

layout. Inter-vertex transition times for the connected sensors were recorded with a stopwatch for a typical subject walking at a normal speed (Table 3.6).

5.3.3. *Selection of Input Parameters.* To determine appropriate input parameters for our inference algorithm we conducted an exhaustive search over the range of $N = 2, .., 6$ and $ln(SSL) = -7, .., -3$ (Figure 3.30). The values $N = 5$ and $ln(SSL) = -5$ gave a slightly higher Q_{simp} value than the surrounding parameter space.[5] Therefore, we selected the solution generated by these parameter values as our inferred network.

5.3.4. *Assessment of Results.* The network parameters inferred by our topology inference algorithm closely corresponded to the ground truth values. Disregarding reflexive links, the difference between the inferred and 'ground truth' results amounted to a Hamming error of 2. Table 3.5 shows the transition matrix output by the algorithm and Figure 3.31 compares the analytically determined and inferred topological maps. The two significant errors are: an extra edge found between sensors A and B; and a missing one-way edge from sensor D to I.

The missing edge from D to I is likely due to the tendency of people to go straight rather than turn right when navigating the corridor on the bottom right of the region (heading left) as shown in Figure 3.29. The inferred transition probability of 0.06 seems low (Table

[5]We used the same simulation verified shaping parameters for the Q_{simp} metric as the previous experiment.

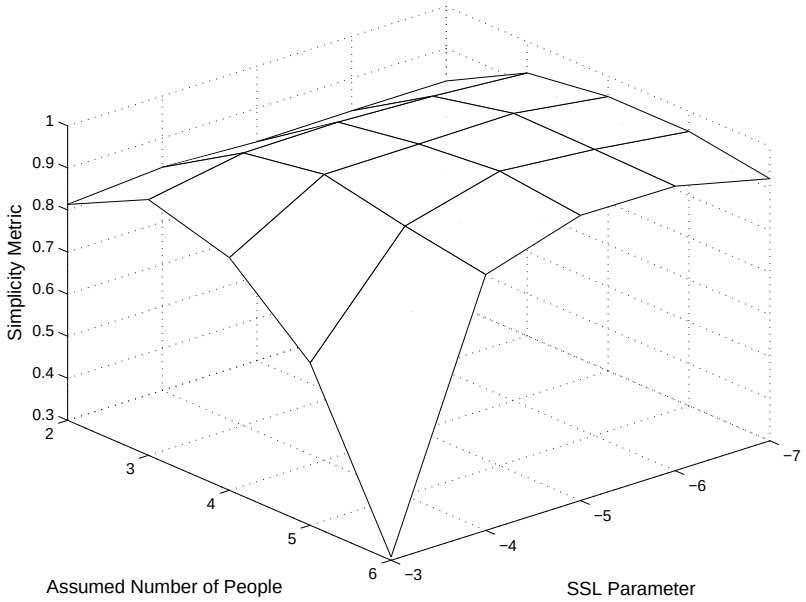


FIGURE 3.30. A plot of the Q_{simp} metric as a function of input parameters.

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**SS**
A	**0.19**	**0.11**	**0.29**	0.00	0.00	0.02	**0.13**	0.03	0.05	**0.19**
B	0.07	**0.63**	**0.11**	0.02	0.01	0.01	0.04	0.04	0.01	0.07
C	**0.20**	**0.12**	0.08	0.03	0.01	0.00	**0.17**	**0.26**	0.03	**0.10**
D	0.05	0.04	0.07	**0.10**	0.02	**0.27**	0.02	**0.31**	0.06	0.05
E	0.02	0.02	0.02	0.02	0.08	**0.39**	0.01	0.04	**0.35**	0.05
F	0.01	0.03	0.03	**0.43**	**0.37**	0.05	0.01	0.00	0.03	0.03
G	**0.31**	0.03	**0.34**	0.03	0.01	0.00	**0.12**	0.04	0.01	**0.11**
H	0.04	0.05	**0.35**	**0.15**	0.02	0.00	0.02	0.04	**0.25**	0.07
I	0.04	0.02	0.05	**0.11**	**0.32**	0.02	0.02	**0.33**	0.02	0.07
SS	**0.28**	**0.17**	**0.23**	0.05	0.02	0.01	**0.13**	0.08	0.04	0.00

TABLE 3.5. The transition matrix inferred from the experimental data. SS refers to the source/sink node introduced by the algorithm. Bold values over the threshold $\theta = 0.1$ are interpreted as directed edges. The underlined values were not directly predicted by the ground truth analysis.

3.5), but might actually reflect reality; *i.e.* only about 6 *per cent* of trajectories turn right at the intersection between sensors H and I. This missing inferred connection demonstrates a limitation in the approach of exploiting motion in the environment. Our technique can only learn traffic patterns common enough to be easily recognized and distinguished.

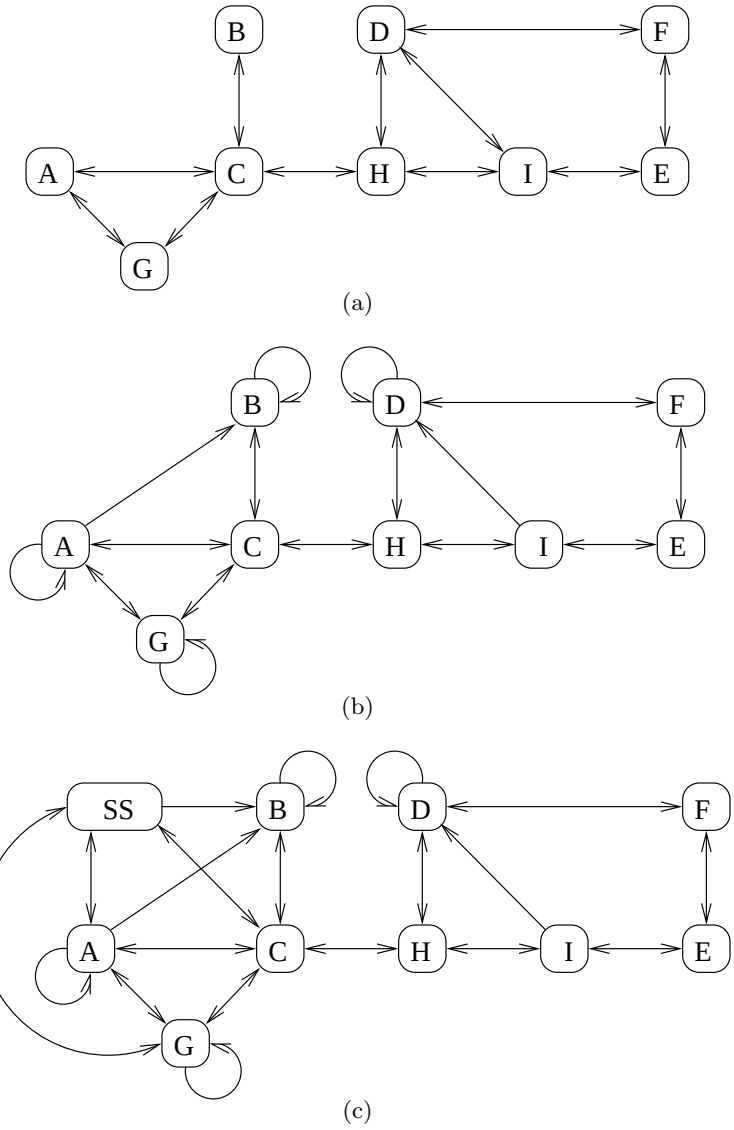


FIGURE 3.31. Topological maps of the environment that were: a) analytically determined based on the layout; b) inferred by the algorithm; c) inferred by the algorithm including the source/sink node.

The extra edge found leading from sensor A to sensor B is likely due to a correlation in the detection intervals between these two nodes. Since both sensors are in boundary locations, they are likely to receive events caused by people that then leave the monitored region for some time. Both of these areas see significant traffic, much of which does not directly lead to another monitored area. Figure 3.32(d) shows the inferred delay distribution

Connection	**Timed**	**Inferred**
A,G	6	8 / 11
A,C	9	12 / 10
B,C	5	6 / 8
C,G	5	5 / 5
C,H	5	6 / 6
D,F	14	15 / 17
D,H	5	5 / 6
D,I	6	7 / 7
E,F	13	13 / 13
E,I	13	15 / 14
H,I	4	4 / 4

TABLE 3.6. A comparison of timed and inferred delay times (both ways) between sensors. All values are rounded to the nearest second.

between these two nodes; the distribution is far from what would be expected from 'through-traffic'. It is possible that erroneous edges of this type could be eliminated based on the shape of their associated delay distribution. This could be done probabilistically using a prior, or as a post processing step.

It should be noted that in this work a truncated normal was employed to model the delay distributions, however, results were also obtained using a gamma distribution. Interestingly, better results were obtained using the truncated normal. It is possible that when using this distribution family, the algorithm is better at symmetrically rejecting outliers on both sides of the mean, and as a consequence finds parameters that form tighter more decisive inter-vertex distributions. Presumably this has the effect of improving the accuracy of the inference process.

The mean transition times produced by the algorithm were consistent to those determined manually (Table 3.6, Figure 3.32). In general, however, the inferred delay value were on average longer than the measured values. This is probably due to the fact that it is much easier to lengthen a delay time than shorten it; *i.e.* a person can, in practice, be arbitrarily slow. For example, people stop and exchange a few short words with someone as they pass on route to their destination more often than they break into a jog. Figure 3.32(b) shows a distribution that might be explained by this effect.

The connections to the source/sink node occur only for boundary nodes (Figure 3.31(c)) and are therefore consistent with an analytical assessment of the traffic patterns. Since

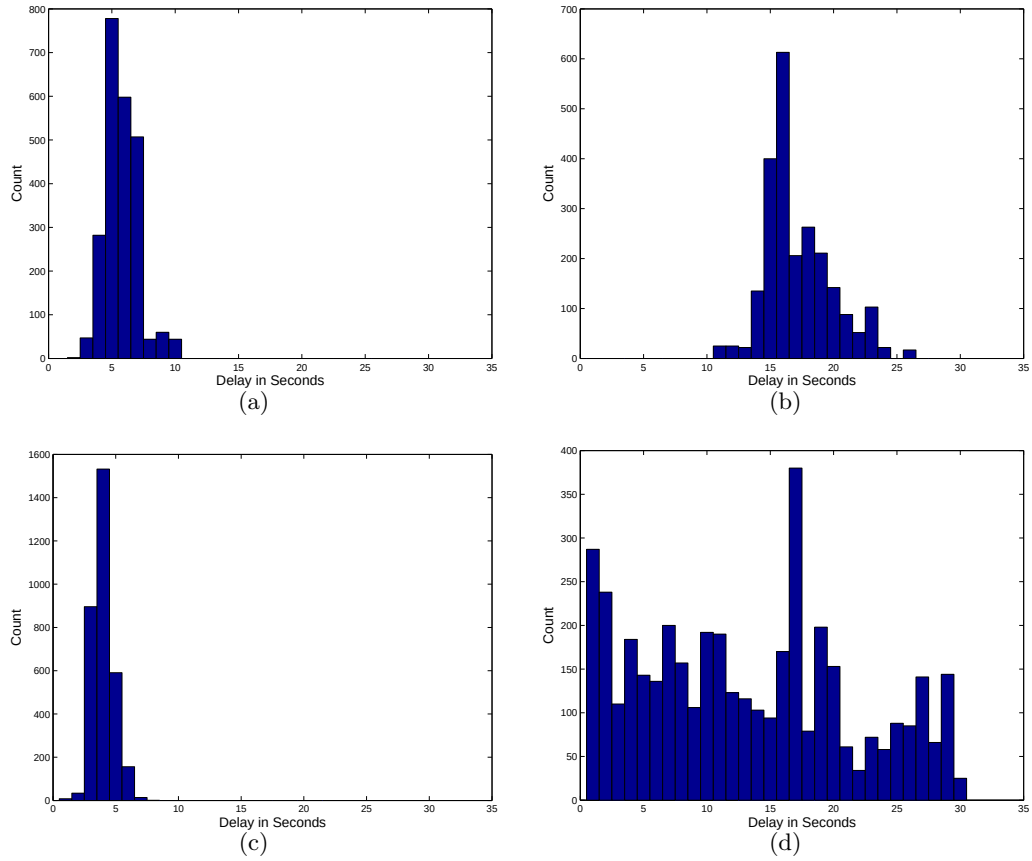


FIGURE 3.32. Examples of delay distributions inferred for: a) sensor D to sensor H; b) sensor F to sensor D; c) sensor H to sensor I; d) sensor A to sensor B (an erroneously inferred edge).

traffic commonly enters and exits the monitored region *via* one of the boundary nodes, the inference algorithm should commonly employ the source/sink node in order bring the agent back into the system.

6. Discussion

In this chapter we presented a method for inferring the topology of a sensor network given non-discriminating observations of activity in the monitored region. Our technique recovers the network connectivity information opportunistically through the exploitation of existing motion. This task is accomplished based on no prior knowledge of the relative locations of the sensors and only a limited knowledge of the type of activity present in the environment.

We described a formulation of the problem such that it could be iteratively solved with a stochastic Expectation Maximization algorithm. The method uses the observational data and Markov Chain Monte Carlo sampling to construct likely trajectories describing the motion of agents present in the environment. By inferring underlying patterns in their motions, the technique recovers the connectivity relationships between the sensors and constructs a Markov model describing their behavior. From this information, a topological description of the network can be constructed.

Results from numerical simulations verified the feasibility our approach. A simulator modeling the problem was constructed, and the technique was tested on a number of random networks of different sizes and under a number of different conditions. The technique demonstrated a high degree of accuracy and was both robust to noise and to complex traffic patterns. It appeared that the results obtained by our method compared favorably to related work by Ellis *et al.* [**68, 37**], although their approach was much less computationally intensive.

Our approach was then further examined with experiments carried out using a heterogeneous sensor network. The network was constructed using two types of sensors: vision-based sensors using PC hardware and webcams, and photocell-based sensors using low-powered MICA2 devices. Some implementation details were non-trivial due to the limitations of the low-powered platforms.

Data collected under these real world conditions varied considerably from data generated by the simulator. The imperfect, hardware implemented, sensors were occasionally subject to both missing and spurious observations. These errors often occurred in an unpredictable manner. Additionally, the patterns of motion through the environment were complex and did not consist of only 'through traffic'.

The performance of our technique on the experimental data was satisfying. The inferred results closely matched analytically determined 'ground truth' values and were consistent with empirical assessments. However, the results were poorer than what would be expected on data produced from similar graphs with the simulator. Since the inference algorithm was primarily developed and tested under simulated conditions, the drop in performance under real world conditions presumably indicates both shortcomings in our algorithm when

processing real data, and also some significant differences between the data produced by our simulator and the real world.

Although efforts were taken to model real world effects such as spurious and missing observations, and biased delay times, realistic traffic patterns are considerably more complex than the ones we were capable of generating in our simulator or capturing in the model used by our algorithm. Some of the more critical attributes present in the real world that are missing in our simulated environment and could be better handled by our model are the following:

(i) real traffic tends to be 'bursty' with either many or very few people in the region at any one time;
(ii) there is no upper bound on the number of people in the region;
(iii) real people walk at different speeds, and the same person will walk at different speeds at different times;
(iv) there are significant differences in traffic flow among different areas in a region;

Closing some of the gaps between the real world and the simulated environment could be a step toward eventually improving the performance of the algorithm under real-world conditions. Although realistic traffic patterns are complex, they can also give additional clues about the environment. For example, prior belief could be placed on the assumption that heavily used areas are adjacent to other heavily used areas. Another possibility could be to exploit the velocity of a particular agent as a probabilistic method of identifying them throughout the system.

Results from both simulations and experiments have shown the ability of the algorithm presented in this chapter to generate accurate results under conditions of sensor noise and complex traffic patterns. The technique compares favorably to related approaches and could have promising real world applications in the area of sensor network calibration and self-configuration. We will explore potential future directions for the work in the next section.

7. Future Work

The work presented in this chapter suggests some open problems. We assume that agents in the system tend to transit the sensors separately. While we can tolerate some

violation of this assumption, an explicit model might be required to deal with an environment in which this happens with high frequency. We also assume that the behavior of the agents in the environment are statistically independent. Dealing explicitly with correlated behavior is an interesting problem and is related to the work of Haigh [**47**].

A somewhat related issue that potentially effects the accuracy of our technique is the fact that traffic patterns change over larger scales of time. A potential improvement to the technique could be made by attempting to model these changes to some degree. The technique could then take advantage of the information rich times in which only a few agents are in the region. Our current approach models fluctuations in the number of agents through the use of the source/sink node. An alternative technique could segment the observations into time-windows and then choose traffic assumptions appropriate for each window based on the nature of the data and the current belief of the network parameters. For example, the number of agents present during a time-window could be estimated based on the density of observations in that window given current network parameters. The algorithm could then further refine its estimate by generating trajectory samples based on these temporally local estimates of the number of agents.

Another interesting area of future investigation would be the incorporation of more detailed sensor information into our algorithm. This should result in faster convergence and higher accuracy with fewer observations. In this chapter, we restricted the sensor observations to be non-discriminating in order to demonstrate our technique under worst-case possibilities. However, even poor quality, noisy features extracted from the observations should improve the accuracy of the system as long as they are incorporated in a probabilistic manner.

The easiest way to include these additional event signatures into our probabilistic framework would be to calibrate the sensors prior to deployment. The goal of the calibration would be to have the same agent generate similar event signatures at each of the sensors that employ the same sensing modality. However, this prior calibration step is somewhat against the spirit of sensor network self-configuration.

A more interesting way to incorporate event signatures into our approach would be to learn the correspondences between observation features and individual agents at each sensor. This information would become part of the learned network parameters and would

influence the trajectory samples obtained at each iteration. For example, the system might learn to associate a particular agent with the colour red at sensor one, and low frequency audio at sensor two.

In the next chapter we consider a variant of the problem we have investigated in this chapter in which even less observational data is available for inference purposes.

CHAPTER 4

Learning Network Topology from Simple Sensor Data

In this chapter, we consider an approach for recovering a topological map of the environment using detection events from a deployed sensor network for the case in which the observational data is *time-stamp free*. The probabilistic inference process discussed in the last chapter relies on the timing information in the observations to build up a delay model which aids the convergence and final accuracy of the approach. The removal of the timing information makes this problem different, and much harder than the one we considered in the previous chapter.

In the time-stamp free version of the sensor network topology inference problem, we assume not only that the agents moving though the environment are indistinguishable, but that there are no temporal clues that can be used to aid the inference process. In other words, the detection events are correctly ordered but are not *time-stamped* with a synchronized time value that temporally relates the time an observation was collected with those collected by other sensors. However, by employing a sliding window over the observations, we will show that the problem can be re-formulated as a version of the well understood set-covering problem and accurate results can be obtained without timing information. Therefore, when the inference algorithm presented in this chapter is employed, the time-stamp data can be discarded or simply not collected in the first place.

In order to exploit timing information in the observational sequence some model of agent motion in the environment needs to be either constructed based on prior assumptions, or

learned from the data. The technique we present in this chapter, however, allows the correct edges in the graph to be inferred while avoiding the prior domain knowledge or algorithmic complications involved in constructing an adequately accurate motion model.

In the remainder of this chapter we first formally define this version of the problem, and then give a theoretical analysis of the timestamp free version of the sensor network topology inference problem. Then, we present two heuristics based on the set-covering formulation of the problem and finally evaluate them with numerical simulations.

1. Problem Definition

We formulate the problem of learning the topology of the network as the inference of a directed graph $G = (V, E)$, where the vertices $V = \{v_i\}$ correspond to deployed sensors and the edges $E = \{e_{i,j}\}$ correspond to connectivity between them; *i.e.* an edge $e_{i,j}$ denotes a path from the position of sensor v_i to the position of sensor v_j. The sources of motion with in the sensor network are modeled as some number of agents N moving asynchronously through the graph. Each agent generates an observation every time it visits a vertex. This corresponds to an agent passing near a particular sensor which then detects the presence of motion in its region.

The input to the problem is the number $|V|$ of sensors deployed and an ordered list of observations $O = \{o_t\}$ where t is an index which, for convenience, we define as an integer from 1 to $|O|$. Each of the observations o_t is identifiably generated by one of the sensors; *i.e.* each $o_t \in [1, |V|]$. The goal is to find the correct underlying graph G explaining this observational sequence.

2. Algorithm Formulation

2.1. Smallest Graph is Correct Answer. The key idea behind our approach is to find the *smallest*[1] graph that successfully explains, or is *consistent* with, the observed data. Leaving aside for the moment the implementation details, let us consider this idea in more depth by proposing the existence of an algorithm A that takes as an input the assumed number of agents N in the environment and the observational sequence and returns as an output the smallest graph consistent with the observations O.

[1]The graph with the smallest number of edges.

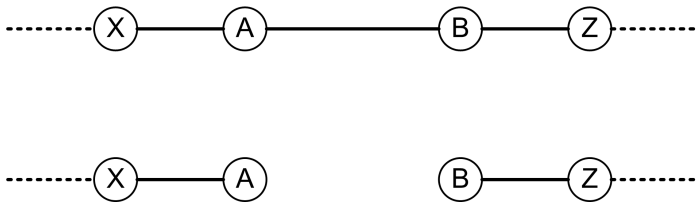


FIGURE 4.1. Example of removing edge AB from graph G_c, (shown partially on top), to create graph G'_c, (shown partially below).

Our algorithm A considers each of the possible trajectories that could be taken by these N agents given the observational sequence O and then selects the trajectory set that requires the smallest number of inter-vertex traversals which are consistent with the observations. The algorithm then returns the graph populated only with edges that correspond to the inter-vertex traversals required by this chosen trajectory set; *i.e.* $G = A(O, N)$.

The concept that the simplest solution explaining the data is probably the correct solution was used successfully in the version of the topology inference problem considered in the last chapter. This can be viewed as a specific instance of the general principle known as Occam's razor. We will show in the next section that under certain assumptions, we can prove that an algorithm that finds the smallest consistent graph will return the correct answer.

THEOREM 2. *A graph G that is consistent with the observations O and any bounded value for the assumed number of agents N must have the correct solution G_c as a subgraph given the following assumptions:*

(i) *There are an infinite number of observations, O.*

(ii) *The transit time between nodes may be longer than the time between the first and last observation in the sequence O.*

(iii) *The motion of each of the agents is random.*

(iv) *the true number of agents N_c in the system is bounded.*

(v) *There are no self-referential connections in the true graph G_c;* i.e. *no agent may trigger two observations by one passage through the region of a single sensor.*

Proof:

We will show that it is possible to have sequences generated by the true graph G_c that cannot be explained by a smaller graph G'_c. Let us consider a graph G'_c created by removing a single edge from G_c, as illustrated in Figure 4.1. In this case, we remove the edge AB from graph G_c. Let us now create a valid observational sub-sequence $K = ABABABAB...AB$ which was created in truth by a single agent traversing back and forth on the edge AB. Note that this sequence generated by a single agent can be arbitrarily long given the assumption of unbounded transit times. The only way agents in a graph G'_c could generate exactly this observational sequence would be if some number of them were 'stationed' at node X, and some number 'stationed' at node Z, and alternatively one agent from X traversed the edge to A, and then one from Z traversed the edge to B. However, if the length, $|K|$ of the observational sub-sequence is larger than the maximum number of assumed agents N, then there will not be enough agents in G'_c to generate K. Therefore, the edge AB must be present in any consistent solution. Applying this to all the edges in G_c, we see that any consistent solution that can explain all the transitions implied by the observations must have G_c as a subgraph $\square$.

Note that this analysis requires that there be both an infinite number of observations and random motion on the part of the agents in order to allow such very rare observational sequences to exist. However, this concept holds probabilistically with bounded sequences of observations. As the number of observations grows, it becomes increasingly unlikely that a consistent solution can be found that is missing portions of the real graph. This concept is defined in the following corollary:

COROLLARY 1. *A graph G that is consistent with the observations O and any bounded value for the assumed number of agents N will have as a subgraph the correct solution G_c with a probability that approaches one as the number of observations $|O|$ approaches infinity given the following assumptions:*

(i) *The transit time between nodes may be longer than the time between the first and last observation in the sequence O.*

(ii) *The motion of each of the agents is random.*

(iii) *the true number of agents N_c in the system is bounded.*

(iv) *There are no self-referential connections in the true graph G_c;* i.e. *no agent may trigger two observations by one passage through the region of a single sensor.*

From Theorem 2 and our definition of A we can construct some simple lemmas:

LEMMA 1. *G_c will be a subgraph of $A(O, N)$ $\forall$ finite values of N*

Proof:

By definition, $G = A(O, N)$ is a consistent graph for any bounded value of N which, by Theorem 2 can not be smaller than G_c $\square$.

LEMMA 2. $A(O, N_c) = G_c$

Proof:

By definition, $A(O, N_c)$ returns the smallest consistent graph for N_c and O. G_c, must be consistent with O and N_c by virtue of being the true graph. Since from Theorem 2, there can not be a consistent graph smaller than G_c, the smallest consistent graph found by $A(O, N_c)$ is the correct graph G_c $\square$.

THEOREM 3. *For $G_1 = A(O, N_1)$ and $G_2 = A(O, N_2)$, if $N_2 > N_1$ then G_2 will be a subgraph of G_1.*

Proof:

To show that for $N_2 > N_1$, the smallest consistent graph returned by $G_2 = A(O, N_2)$ is no larger than for $G_1 = A(O, N_1)$, we will demonstrate that a path generated by a single agent can be *spliced* between two agents using a 'tag team' method, and yet will still yield the same set of edges. Consider a sequence of vertex traversals S generated by a single agent on the graph G_1. First, without loss of generality, select any vertex $v_{splice} \in S$. We can now pair any two agents together to jointly generate this traversal sequence S in the following way. Let one of the agents be initially stationed at v_{splice}. When the other agent enters this vertex it will exchange its role with the first agent, as in a game of tag team wrestling. The other agent will now leave the vertex v_{splice}, generating a sub-sequence of S

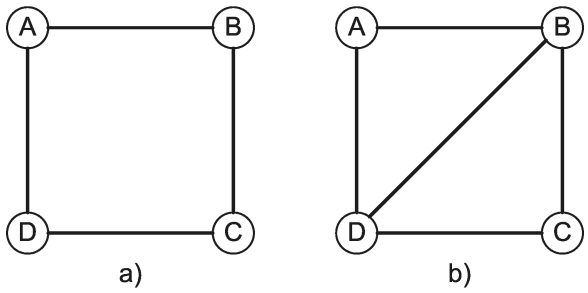


FIGURE 4.2. a) The correct graph G_c b) an incorrect graph

until it re-enters v_{splice}, where again they will switch roles. In this manner, all superfluous agents used in the algorithm can be 'hidden' by splicing a valid path repeatedly □.

It directly follows from Lemma 2 and Theorem 3 that we can assume the existence of more agents than the actual number that generated an observational stream and still produce paths that are consistent with the correct graph. This applies even for non-stationary agents. For example, let us consider the vertex sequence $S = ABCDADCDABCBA$ generated by an agent in G_c of Figure 4.2(a.). We choose $v_{splice} = C$. Now the vertex sequence S assumed to come from a single agent looks like the following: $AB\mathbf{cdad}CDAB\mathbf{cba}$ where capital letters are used for the path P_1 of agent one, and small bold letters are used for the path P_2 of agent two. The individual sequences $P_1 = ABCDAB$ and $P_2 = \mathbf{cdadcba}$ are both valid sequences in the graph G_c.

On the other hand, some sequences in O can only be consistently explained by assuming that the number of agents in the system is at least N_c. Let us consider again the graph depicted in figure 4.2(a.) and let us assume an observation sequence generated on this graph by the motion of two agents. Agent one follows a clockwise trajectory: $ABCD\ldots$, and agent two follows a counter clockwise trajectory $\mathbf{ADCB}\ldots$. By combining the two paths, it is possible at some point to get the subsequence: $K{=}AB\mathbf{DC}$. If we assumed the existence of only one agent, then we would be forced to assume the existence of an extra edge between B and D as shown in figure 4.2(b.).

The analysis in this section suggests that an algorithm A that returns the consistent graph with the smallest number of edges given the observations O and assumed number of agents N could be a powerful tool for finding the correct underlying graph even if the number of agents generating the observations were unknown. For example, one could run

the algorithm for larger and larger values of N until the graph that is computed stops decreasing in size. Of course, some of the assumptions made in this analysis, such as an infinite number of observations, will not hold in practice.

In the next section, we draw on the theoretical analysis of this section to motivate a pragmatic approach for topology inference. In particular, we will consider methods for estimating the smallest consistent graph given an observation sequence.

3. The Sliding Window Approach

We now present an algorithm for estimating the smallest possible graph G given an observation sequence O and the number of agents in the system N. Our approach is based on the following lemma:

LEMMA 3. *In any given continuous sequence of $|O| > N$ observations generated by N agents in the graph G, at least $(|O| - N)$ transitions between observed vertices correspond to edges in the graph G.*

For example, let $|O| = 4$ and $N = 3$ and the recorded observational sequence be $ABCD$. Since there is one more observation than there are agents in this example, at least two of the observations must have been generated by the same agent. Therefore, at least one of the following transitions between nodes must have occurred: AB, AC, AD, BC, BD or CD. In general, the number of potential transitions generated with a sequence of $|O| > N$ observations is:

$$Q = \frac{(|O| - 1)|O|}{2} \tag{4.1}$$

Our approach is to consider in turn small contiguous subsequences of the entire observation sequence O. We refer to these successive sets as a 'sliding window'. We use a sliding window of size $W = N + 1$ so that each of the subsequences that we consider gives rise to a list of candidate edges L_i, one of which must be present in the graph G. From Equation

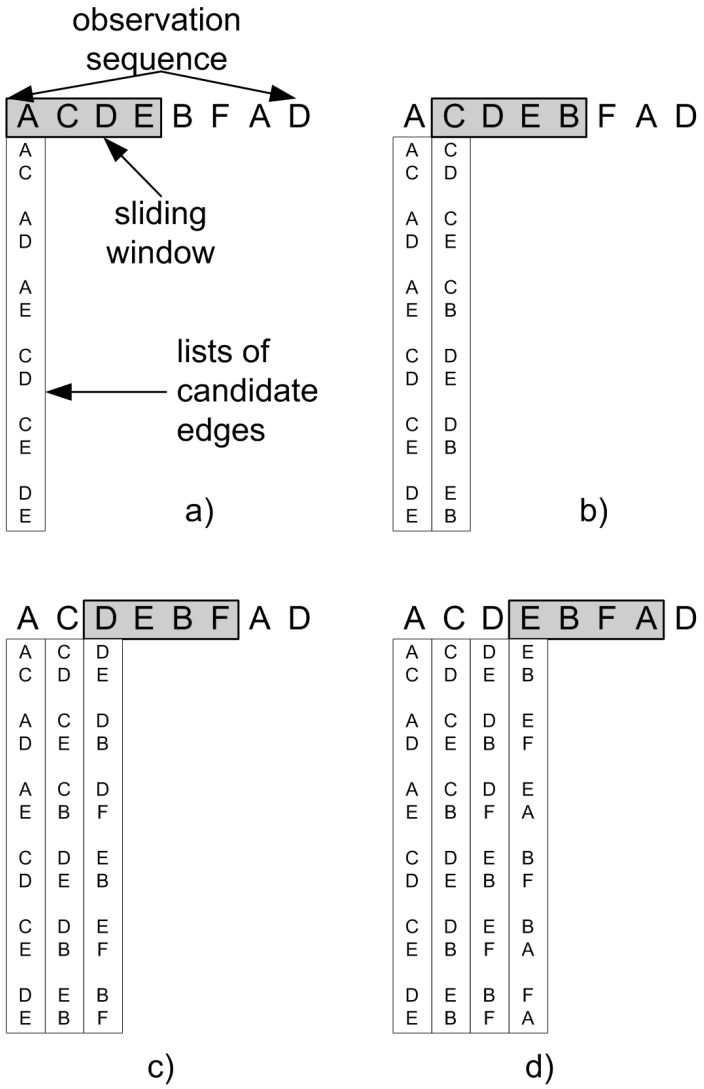


FIGURE 4.3. Example of generating candidate edges for each sliding window position. The window is moved to the right from a) to d).

4.1, in can be seen that the number of candidate edges will be:

$$\begin{aligned} Q &= \frac{(W-1)W}{2} \\ &= \frac{(N+1-1)N+1}{2} \\ &= (N^2+N)/2 \end{aligned}$$

Once the window has moved over the complete observation sequence O, there will be $K = |O| - W$ lists generated. Figure 4.3 shows an example of generating candidate edges using

the sliding window approach. Our approach is to find the smallest graph that can explain at least one edge in each of these candidate lists: $L_1, L_2, ...L_K$.

This problem of selecting the smallest number of edges which contain at least one member of each candidate list is equivalent to the well known set-covering problem. In the set covering problem, the input is a number of sets, each of which might have some elements in common. The desired output is the minimum number of sets required such that each element is represented; *i.e.* the union of the sets selected is the same as the union of all the input sets. To formulate our problem as a set covering problem, one can consider there to be one set for each edge that is in at least one candidate list. Each set contains each of the candidate lists which have this edge as an element. The input to our problem is now a number of sets, each corresponding to a potential edge, and the desired output is the minimum number of sets which contain in their union each of the candidate lists. The set covering problem is NP-complete [**58**], however, several heuristics can be employed to provide a good solution. We will consider two heuristic approaches in the next sections.

3.1. A Greedy Approach. One method of obtaining a solution to the sliding window problem posed above, is to adopt a greedy algorithm. This is a standard, locally optimal heuristic often used with good results for set-covering problems. In our domain, the greedy algorithm would work as follows:

(i) Begin by marking all candidate lists $L_1, L_2, ...L_K$ unexplained and initialize a list of edges E to be empty.

(ii) Find the edge e that is present in the greatest number of currently unexplained candidate lists.

(iii) Remove from consideration those candidate lists which contain edge e by marking them explained, and add e to E.

(iv) Repeat steps 2 to 3 until all lists are marked explained. Return the graph corresponding to our list of edges E as the final solution.

3.2. A Statistical Approach. A statistical approach could also be used to determine the correct edges in G_c. The number of times a given edge has been seen in any candidate list could be tallied up. Those edges that occur with a frequency greater than some threshold T could then be selected.

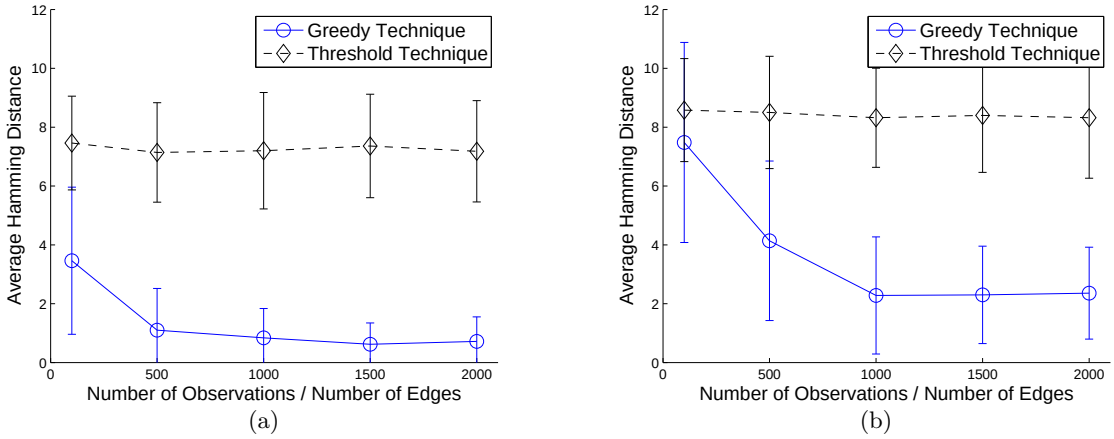


FIGURE 4.4. Mean Hamming distance obtained from the two techniques for various numbers of observations averaged over 50 randomly produced graphs. (Error bars show one standard deviation in the Hamming distance.) Results obtained from 4 agents and 10 node graphs with: a) 12 edges b) 20 edges

Let us consider a suitable value for the threshold T. If G_c corresponded to a fully connected undirected graph, the average tally of each edge would be:

$$\mu = \frac{KQ}{|E|}$$

where Q is the number of candidate edges generated *per* window, K is the number of candidate lists (windows), and $|E|$ is the number of potential edges in the graph. Replacing K with $|O| - W$, Q with $(N^2 + N)/2$, and $|E|$ with $V(V-1)$, we arrive at:

$$\mu = \frac{(|O| - W)(W-1)W}{V(V-1)}$$

Since we expect G_c to contain less edges than its fully connected counterpart, $T = \mu$ can be expected to be a suitable threshold.

Both the threshold method and the greedy algorithm often seem to produce acceptable solutions. In the next section, we evaluate their performance rigorously.

4. Performance Evaluation

4.1. Simulator. We have examined the sliding window approach with a number of experiments conducted in simulation. We have constructed a simulation tool that takes

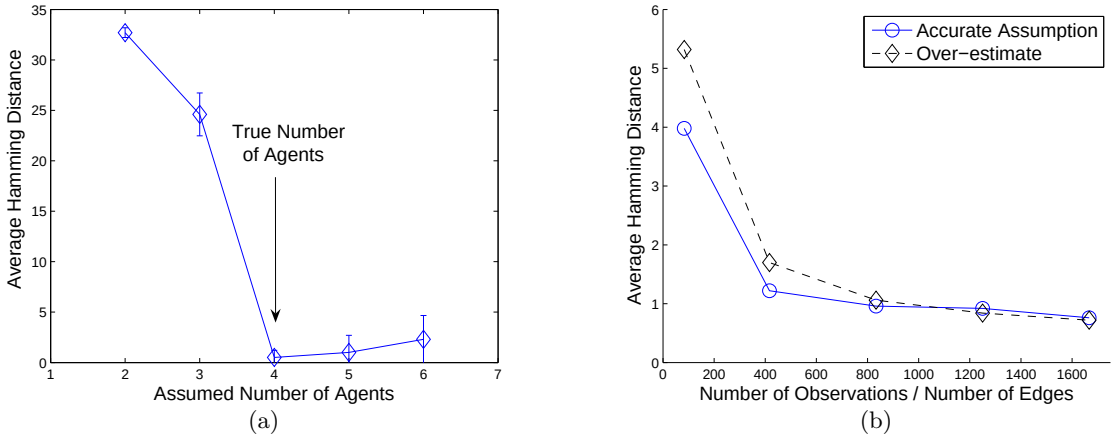


FIGURE 4.5. Results obtained by differing the assumed number of agents for graphs of size 10 nodes and 12 edges. a.) Hamming distance as a function of the assumed number of agents for the greedy algorithm. Results obtained with 10000 observations generated from 4 agents and averaged over 10 graphs. (Error bars show one standard deviation). b.) Mean Hamming distance as a function of observations for an accurate assumption of 4 agents and an over-estimate of 5 agents. Results averaged over 50 graphs.

as input a graph and the number of agents in the environment and outputs a list of observations generated by randomly walking the agents through the environment. A number of experiments were run using this simulator on randomly generated planar connected graphs. The graphs were produced by selecting a connected sub-graph of the Delaunay triangulation of a set of randomly distributed points. (Examples of graphs produced by this technique were shown in the last chapter; Figure 3.7.) For each experiment, a performance metric was computed using the Hamming distance between the true and inferred graph as was described in Section 4.1 of Chapter 3.

4.2. Assessment of Results. The greedy approach was capable of producing accurate results for moderately sized graphs. For example, when given an adequate number of observations, graphs with 10 nodes and 4 agents were consistently solved by the greedy approach with an average Hamming distance of less than one for sparse graphs and less than three for dense graphs. Although not as accurate on average as the greedy approach, the threshold-based approach was also capable of producing a solution near the true answer. Figure 4.4 compares the accuracy of theses two approaches over 50 randomly produced

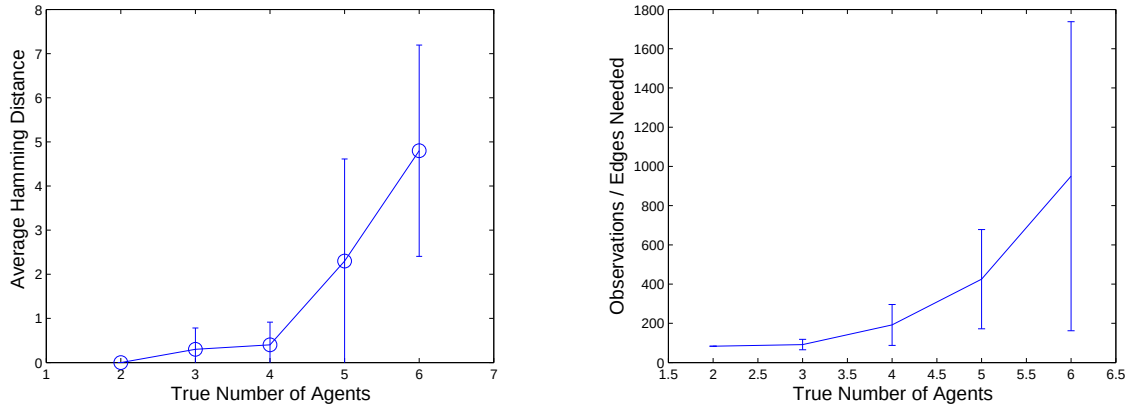


FIGURE 4.6. Performance of algorithm as a function of the true number of agents for the greedy algorithm where the assumed number of agents is set to the correct number. Results averaged over 10 graphs of size 10 nodes and 12 edges; (error bars show one standard deviation). a.) Hamming distance obtained with 10000 observations. b.) Number of observations required to obtain a result with a Hamming distance of 2 or less.

graphs of 10 nodes and two different edge densities. Note that the accuracy of both approaches tended to increase as the number of observations increased. It also appeared that denser graphs required larger numbers of observations to obtain the same accuracy level than that obtained in sparser graphs. Additionally, it was observed that the greedy approach obtained a lower Hamming distance on average for less dense graphs. However, when the *proportion* of the true graph structure recovered was considered, this effect was lessened. For example, for the experiment shown in figure 4.4, the Hamming distance divided by the true number of edges in the graph was approximately double for denser graph, while the Hamming distance alone was approximately triple.

Unsurprisingly, the accuracy of the threshold-based approach was very sensitive to the value of the threshold selected. Experiments not shown here verified that the value for T selected above was generally suitable for graphs of various densities and sizes, although often better results could be obtained for any specific graph type through careful tuning. This approach requires relatively little computational effort and might have value as a bootstrapping technique for more complex approaches such as the one presented in the previous chapter.

As predicted by Theorem 3, the error induced by over-estimating the number of agents in the environment was less severe than that of under-estimating the number of agents. Figure 4.5 shows the result of assuming various numbers of agents in one situation for the greedy approach. As the over-estimation increases, the required number of observations needed to solve the problem also increases. The problem of topology inference becomes more difficult as more agents are added to the system. Figure 4.6 shows the correspondingly poorer performance obtained with the greedy approach on the same set of graphs with observations generated from larger numbers of agents. Even if the correct number of agents is known, we conjecture that less information is available as the number of agents increases. As the size of the sliding window increases, so does the number of candidate edges generated by each sliding window. Therefore, the ratio of known correct to incorrect edges decreases, and hence, more observations are needed to obtain the same level of error. A similar effect was noticed and discussed in the last chapter, in Section 4.2.3.

5. Discussion

In this chapter, we have described a technique for learning the *topology* of a sensor network, using only event ordering information. We presented a theoretical analysis of the problem, and re-formulated it as a set-covering problem. Two methods were presented to solve this problem, one based on a statistical measure, and one based on a sliding window technique. The work has raised some open problems. Although it seems clear that increasing only the number of agents in an instant of the problem tends to dilute the information gained per each observation, it is not obvious whether it is possible to derive a formal relationship for the information gained in the context of the sliding window approach. Additionally, it would be of interest to find an analytical relationship between the number of observations needed to solve a problem and the corresponding density of agents in the system; *i.e.* the ratio of agents to edges in the true graph.

The method we have presented in this chapter could potentially be useful in a data collection system in which ordering information is available, but accurately time stamped information is not. For example, one could envision a network in which neighbouring nodes exchange localized communications for ordering purposes when observations are recorded, but do not undergo the effort of maintaining network wide synchronized clocks. This could

be the case, for example, in a large scale wildlife monitoring network distributed in a park. Individual sensor nodes might only communicate briefly with immediate neighbours when an observation occurs in order to obtain a correct local ordering and otherwise remain radio silent for power conservation purposes. The distributed data could be periodically sent to a gateway and a global observation ordering could be reconstructed centrally before processing.

The analysis of the problem is also useful in its own right from a theoretical point of view and suggests approaches that could be incorporated into more general techniques for topology inference such as the one considered in the last chapter. In the next chapter we again consider topology inference, but this time from the point of view of a mobile agent. Unlike this chapter and the last, in which stationary components exploit mobile entities for inference purposes, we next consider a mobile entity exploiting stationary features in the environment for inference purposes.

CHAPTER 5

Topological Mapping with Weak Sensory Data

In this chapter, we consider the topological inference problem from the perspective of a single mobile sensor. This is as opposed to a network of many stationary sensors as has been considered in the last two chapters. Like our investigations up to now, however, we will assume that the mobile sensor, or robot, has minimal sensing abilities and, as in the last two inference techniques, we will rely on the principle of Occam's razor to help select from among the potential topological explanations for the environment that match our observational data. We represent the world as an undirected graph in which vertices represent discrete places and edges navigable paths between them. We assume that the robot can consistently assign a cyclic ordering to the edges leaving a vertex with reference to the edge it arrived from, however, it is unable to associate a unique label with any place or edge. Given this limited sensing capability, and without the use of any markers or additional information, we will show that the construction of a topological map is still feasible.

The work we present in this chapter addresses a fundamental problem in mobile robotics: the mapping of an unknown environment. As the wealth of literature addressing the simultaneous localization and mapping (SLAM) problem in mobile robotics suggests, this problem of mapping a previously unknown environment in the face of imperfect sensory data has proved to be a challenging task. Some important research in this area that has emerged in the first decade of this century includes the FastSLAM work of Montemerlo *et al.* [**87**] [**88**], work by Wolf and Sukhatme [**131**], and work by Dellaert and Kaess [**24**].

One of the key problems in SLAM is that of *closing the loop* or determining whether a currently observed landmark or region corresponds to a previously visited location or a new portion of the world being explored; (*e.g.* work by Newman and Ho [**91**] or the work of Martinelli *et. al* [**80**]). The question we consider in this chapter can be considered an extreme case of the loop closing problem in SLAM in which the robot has almost no ability to characterize its surroundings or obtain meaningful odometry measurements. While most SLAM techniques are based on local, incremental localization, loop closing depends on global localization that takes into account the full history of the robot's motion.

In the remainder of this chapter we first provide some background on this class of problems, and then give a definition of the specific topological mapping problem we are interested in. We then introduce our methodologies and present an evaluation of their performance through simulations. Finally, we give some discussion of the results.

1. Background on Graph Exploration

The study of a robot equipped only with the sensing ability to assign a consistent cyclic ordering to edges in a graph-like world has been examined previously in [**34**] and [**33**] by Dudek and colleagues. In this work, a mapping strategy is presented in which the robot constructs an *exploration tree* that enumerates consistent world hypotheses at each step of an exploration process. The authors classified the potential correspondence errors that could be made during the construction of this tree into three classes. One, errors of type OLD-LOOKS-NEW, in which the current location is assumed to be newly explored, but was actually visited earlier; two, errors of type MIS-CORRESPONDENCE in which the current location is thought to be a certain previously visited area, but is actually a different previously visited area; and three, errors of type NEW-LOOKS-OLD, in which a location is assumed to have been previously visited, but is actually new.

The authors discussed the fact that in a complete hypothesis tree, there will always exist a model which assumes that each place visited is a new location; *i.e* multiple errors of the type OLD-LOOKS-NEW. Among the three types, this class of errors is unique since the models they generate can never be shown inconsistent given the local sensing capabilities considered. The work concludes by suggesting a heuristic to be used during the exploration

process that prunes all models of size greater than:

$$T = \gamma s + C \tag{5.1}$$

where s is the number of nodes in the current smallest incomplete model, and γ and C are constants. We will refer to this threshold as the *Dudek size threshold* (DST) in the remainder of this chapter.

Later work such as that conducted by Dudek *et al.* [**35**], by Rekleitis *et al.* [**108**], and by Deng *et al.* [**29**], [**28**] considered a version of the problem in which the robot with the same limited perceptual abilities was capable of placing and recognizing one or more markers. Unlike the marker-less version, it was shown that by using the supplementary global information, one can resolve potentially incorrect correspondences and therefore unambiguously map a finite world (given adequate exploration).

In this chapter, we re-visit the marker-less problem. We present new exploration strategies that help reduce correspondence errors where possible and introduce an inference technique based on a beam-style search through consistent models in the exploration tree.

Our approach is similar in concept to work by Ranganathan and Dellaert [**101**] [**103**] [**102**]. The set of weighted partial world models we maintain in our inference technique has some similarity to the concept of a probabilistic topological map, as defined by these authors. In both our technique and theirs, a multi-hypothesis, topological space is maintained. The distinguishing difference is that, while we only apply a ranking heuristic function, they use odometry measurements to assign relative probabilities to each of the potential world models. In our work, we do not presuppose the availability of odometry data. Our work is also related to the research of Savelli and Kuipers [**111**] who address the loop closing problem by employing planarity constraints to select among potential topological maps.

It should be noted that practical applications of topological mapping must provide a method for the robot to reliably identify a topological node (or landmark), in the world being explored. In work by Choset and Nagatani [**19**], sonar data is used to identify and position the robot on the Voronoi graph, the vertices of which correspond to topological nodes. In work by Kuipers and Beeson [**61**] place recognition is achieved through a multi-process bootstrapping technique that includes sensory clustering and probabilistic inference. Other approaches consider the extraction of features from vision or other sensory data; *e.g.*

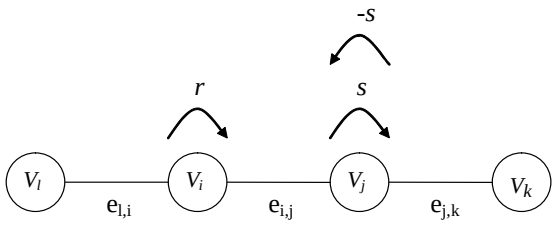


FIGURE 5.1. Diagram showing relationship of visited vertices in the context of the transition fuction δ.

work by Se *et al.* [**114**], work by *Sala et al.* [**110**], and work by Giguere *et al.* [**44**]. In this work, we leave for the moment this problem of identifying when the robot has reached a vertex, and focus on the topological mapping problem.

2. Problem Specification

We describe the problem of topological mapping in terms of the inference of an undirected, un-weighted graph in which the edges leading from a vertex can be assigned a local ordering and each vertex is given a non-unique signature. In particular, we consider the case where this signature is the degree of the node. The vertices of the graph correspond to distinguishable places in the world and the edges correspond to connecting bidirectional paths. As the graph is traversed by the robot, it is able to sense the label of its current vertex and apply a consistent local edge ordering. In other words, the robot is able to enumerate the edges of the place in a systematic way, (*e.g.* clockwise), relative to the edge by which it entered.

We refer to the edge by which the robot enters a place as a *reference edge*. The edge selected for the next move can be specified in relation to this reference edge. We define the transition (or motion) function δ as follows: $\delta(v_i, e_{i,j}, r) = v_j$ which means leave vertex v_i by the edge that is r edges (*e.g.* clockwise) after the reference edge $e_{i,j}$, and this takes us to vertex v_j. By recording its motions the robot is capable of retracing any previously taken trajectory since: if $\delta(v_i, e_{l,i}, r) = v_j$ and $\delta(v_j, e_{i,j}, s) = v_k$ then $\delta(v_j, e_{j,k}, -s) = v_i$ (Figure 5.1).

During each step of the exploration process, the robot records the label (degree) of its current topological node. As this exploration process continues an exploration tree is constructed, the full version of which contains a single world model for every consistent

correspondence among all previously visited topological nodes. Each *level* of this exploration tree will be based on the information obtained from the traversal of a potentially unexplored edge. At any step t, each of the maintained hypotheses in the tree are consistent with the observational data collected up to that point. As discussed in [**33**] by Dudek *et al.* the number of models consistent with the observations depends on the type of graph explored, but can experience explosive growth. This is especially true during the early part of the exploration in which not enough observations have been gathered to prove some models inconsistent. For the graphs we considered, we found that the size of the complete tree quickly becomes intractable for all but trivially small observation sequences. The goal of this work is to manage the growth of the exploration tree so that only those world models that appear of relatively high likelihood are retained.

3. Exploration Strategies

3.1. Breath-First Traversal (BFT). Here, for completeness, we briefly describe the original exploration strategy considered by Dudek *et al.* [**33**]. The strategy processes new edges in a first in first out manner, based on a breadth-first traversal of the world as observed by the robot. For example, when beginning in a vertex with two edges, the robot will traverse the first edge, return to the original vertex, traverse the second edge, and then return again to the original vertex. It has now explored its world up to a radius of one edge traversal. Let us call this a 'level one' exploration. In the next step of the BFT the robot will explore its world up to a radius of two edge traversals. Starting from the original vertex, it will traverse the first edge again, and then recursively do a 'level one' exploration starting with this new vertex. When complete, it will return to the original vertex, traverse edge two and do the same process again. Finally it will return to the original vertex having completed the second level of exploration.

At each level of exploration, the BFT strategy will reach each of the i^{th} neighbors of the vertex v_s where the robot starts the exploration. We define the i^{th} neighbors of a vertex v as all vertices terminating distinct paths of length i which originate from v. Note that a single vertex u may be present many times as an i^{th} neighbour [1] of v provided $(i > 1)$. If d is the diameter of a finite graph, then the BFT algorithm is guaranteed to visit all the

[1] In the case of $i = 1$, this is only possible if multiple edges are allowed between the same vertices; *i.e.* we are exploring a multigraph.

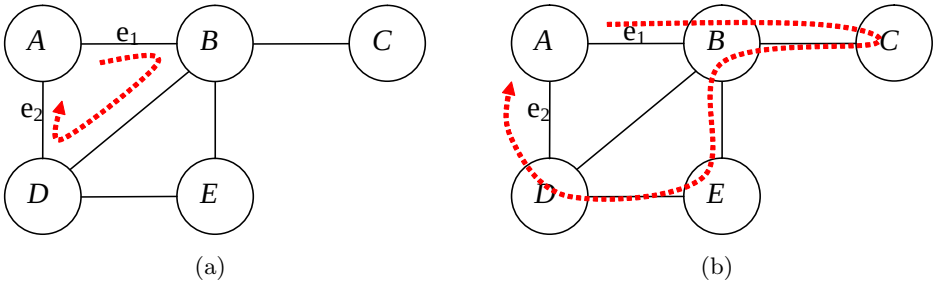


FIGURE 5.2. Example of a) counter-clockwise and b) clockwise ear starting from e_1 of vertex A.

vertices after concluding a level d exploration. A limiting factor when applying the BFT exploration strategy is the size of the exploration tree. In the next sections we will consider new exploration strategies which are designed to help slow the growth of the exploration tree.

3.2. Breadth-First Ears Traversal (BFET). For our purposes, a good exploration strategy will limit, as much as possible, the number of world hypotheses that need to be considered. Of the three types of errors originally identified by Dudek *et al.*, it may be possible to show inconsistent the second and third varieties: MIS-CORRESPONDENCE and NEW-LOOKS-OLD. Errors of the first type, OLD-LOOKS-NEW, in which the current location is assumed to be a new node, can only be diagnosed by considering the implausibility of the world model suggested. We can do no better than this since there is no method of detection for errors of type OLD-LOOKS-NEW. The strength of the original BFT exploration strategy is its guarantee of eventual coverage given a finite world, however, it appears that strategies employing more passes through the potentially previously explored areas can help reveal correspondence errors of the second and third type better than BFT.

We present a deterministic exploration strategy called breadth-first ears traversal (BFET) that, like BFT, is guaranteed of eventually visiting all vertices (and edges) of a finite world. In the next section we will describe a simple stochastic variant.

BFET incorporates within the original BFT algorithm a sub-exploration strategy that attempts to traverse each *ear* leading from the current vertex v. In graph theory, any undirected, 2-edge connected graph can be decomposed into a set of simple paths which are called ears [**69**]. In our work, however, we use the term 'ear' in a slightly different manner

which reflects the fact that the edges leading from any vertex in our graph can be assigned a relative ordering. We define an ear as the closed cycle one obtains by leaving a vertex on a specific edge and selecting for traversal from the following vertex the edge that is next to the reference edge in a *consistent orientation*, (clockwise or counter-clockwise), until one returns to the original vertex. In other words, we consistently select $r = 1$ or $r = -1$ in the transition function δ when tracing out an ear. For example, leaving an edge and making only 'right turns' until one returns to the original vertex will trace out a counter-clockwise ear. See Figure 5.2 for more examples of this concept. The same definition of ear has been used before by Rekleitis *et al.* [**107**].

The BFET sub-exploration strategy works as follows. For each edge leading from the vertex being currently explored in the BFT strategy, take the following steps:

(i) For an edge, e_1 leading from the vertex v, the robot explores the path p_1 beginning with e_1 in one direction (*e.g.* clockwise) for some number of steps (until, for example, a node with the same degree as v is encountered).

(ii) The robot then backtracks to vertex v and explores the path p_2 in the opposite direction (*e.g.* counter-clockwise) for the same number of steps beginning with the edge e_2 that is appropriately located with reference to e_1.

(iii) Steps 1 and 2 are repeated with larger and larger sets of steps taken in both directions until the degree trace for the path taken in two directions matches up; *i.e.* path p_1 visits its vertices in the reverse order of those in p_2.

This process is guaranteed to terminate given a finite graph since there is a bound on both the longest ear in the graph and also the number of ears that any nodes can belong to.

Upon completing the sub-exploration strategy, there is at least the potential that the robot actually visited the same set of same vertices twice in opposite order. Therefore, in the exploration tree, there must now exist a model of the world which reflects the fact that we have found a cycle leading from and back to the node we are currently investigating.

3.3. Loop-Based Exploration (LBE). We will now consider a non-deterministic exploration algorithm based on the BFET algorithm. Essentially BFET works by eliminating inconsistent models through the re-visiting of previously explored vertices in a cyclic manner. Our loop-based exploration strategy (LBE) attempts to capture the spirit of this approach.

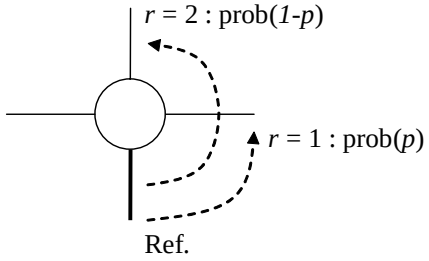


FIGURE 5.3. Diagram showing pictorial example of how the LBE algorithm selects the next edge to traverse with respect to the reference edge when entering a vertex.

LBE works as follows. If the robot is currently visiting a vertex of degree three or higher, then it selects with a probability p the first edge, $r = 1$, from the incoming reference edge for its next traversal (*e.g.* the first counter-clockwise). Otherwise, it takes with probability $(1 - p)$, the second edge, $r = 2$, from the incoming reference edge (*e.g.* the second counter-clockwise). See Figure 5.3 for an example of the edge selection process. If the current vertex is of degree two, then it selects the edge that is not the reference edge, and if the edge is of degree one, then it backtracks.

If a relatively large value of p is selected, this algorithm has the effect of visiting cycles in the graph one at a time, and having much the same effect on the exploration tree as the BFET algorithm for each cycle examined. The larger the value of p, the better, on average we explore a particular cycle, but this comes at the cost of the average coverage time for the graph. Although LBE can not guarantee coverage of a finite graph, we will show that given a good choice for p, in practice this strategy performs as well or better than the more complex BFET strategy. Determining bounds for the expected cover time as a function of p should be possible for a class of graphs, such as those that are planar. This would be related to the work of Jonasson and Schramn [**57**] on cover times for planar graphs and also the work of Koucky [**59**] on universal tranversal sequences.

4. Heuristic Weighted Search

In this section we describe a search algorithm which bounds the number of hypotheses maintained at each step of the exploration process based on heuristic evaluation function. We assume that the simplest models capable of explaining the observed data are the best

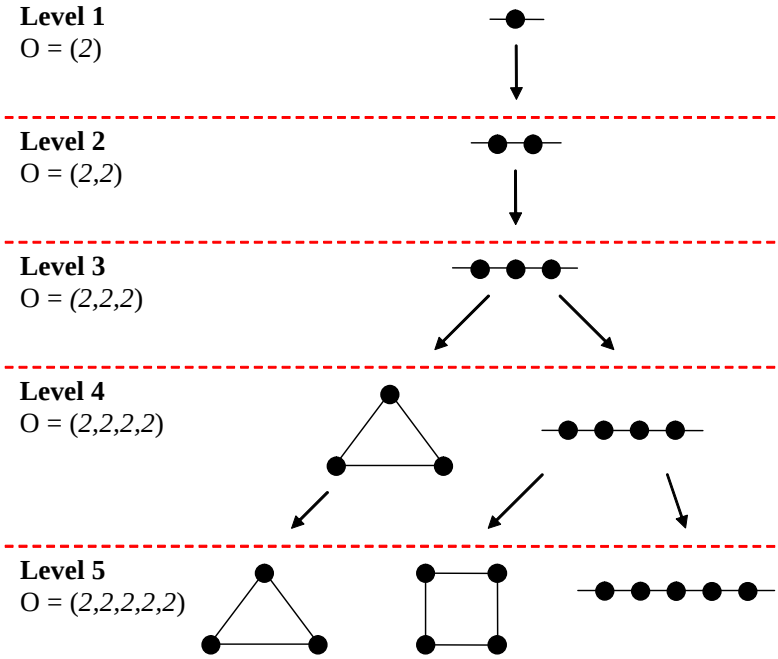
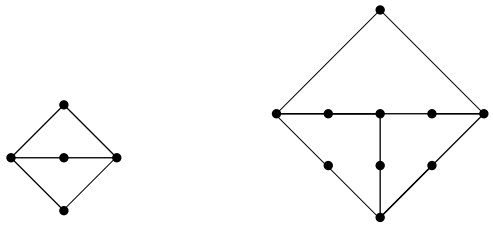


FIGURE 5.4. Consider a robot following an exploration strategy that requires it to take an edge other than the reference edge for each tranversal and which visits only nodes with the signature (degree) 2. This figure shows the full exploration tree with all models maintained for the first five observations. The models are ranked left to right for each level based on the heuristic discussed in Section 4. Up to Level 3 of the exploration tree, the model shown at each step is the only consistent world hypothesis which can explain the observations. During steps four and five of the exploration process, which correspond to Level 4 and Level 5 of the exploration tree, there are multiple models that are consistent with the data. During the fourth step, for example, we can either assume that the robot has revisited the first vertex it started from, or has discovered a new vertex. The first possibility corresponds to the higher ranking model since it only requires 3 vertices and suggests that we have fully explored the world. The second possibility corresponds to a model with a lower ranking since it requires 4 vertices and also contains edges leading to unexplored areas (dangling edges). (We assume that the world can not be a multi-graph in this example.)

ones and rank them accordingly. This principle, known as Occam's razor, is used throughout this thesis.

We define a simple hypothesis as one with as few vertices as possible and, for tie breaking purposes, one with as few singly-connected or *dangling* edges as possible; *i.e* minimal number of edges leading to areas that must still be unexplored according to the hypothesis. We define a hypothesis or graph with no edges leading to unexplored areas as



(a) (b)

FIGURE 5.5. Examples of closed graphs which could explain an endless sequence of observations recording the visiting of alternate vertices of degree 2 and 3.

one that is *closed*. We reward models that are approaching a closed state since we assume it is likely that ultimately the entire region will been explored. Figure 5.4 gives an example of a simple exploration tree and how the maintained world models would be ranked according to the heuristic we have specified above.

Consider another situation in which the robot has observed the node signatures (given by node degrees): $(2, 3, 2, 3, 2, 3, 2, 3, \dots)$ while following an arbitrary exploration strategy. We must surmise that the robot is in a cycle of some multiple of length two, or that our world contains a large component in which each adjacent topological node alternates between degree two and three. If we have done enough exploration to suggest that we should have covered the entire environment, then we might suspect a world that looks like one of the ones depicted in Figure 5.5. In most applications, there is probably some prior knowledge that can be exploited to give a rough idea of the size of the region being explored, and therefore, some guess of the probability of having achieved coverage of the area in question when using a given exploration strategy.

At each traversal of an edge during the exploration process, we first enumerate the new models that can be generated from each of the currently maintained world hypotheses, and we then rank them using our heuristic function. The top N of these models are then selected for maintenance and the rest are discarded. This approach allows online exploration, but risks throwing away the correct solution. Off-line variants could run the same algorithm repeatedly on the same observational sequence but employing an iteratively larger value for N until a suitable solution was obtained.

Our approach to hypothesis management is similar in spirit to the pruning heuristic based on the DST presented previously by Dudek *et al.* [**33**] (Equation 5.1). The authors

Edge to Node Ratio	Average Memory Usage	Trials Solved using DST	Trials Solved using Weighted Search
1.2	30.4	81	96
1.4	157.9	77	90
1.6	1564.5	82	87

TABLE 5.1. Result of pruning all models using the DST with $\gamma = 1.05$, and $C = 2$ as suggested by Dudek *et al.* in [**33**]. Results obtained from 100 trials on random 10 node graphs for three different edge to node densities using the BFT exploration strategy until edge coverage. Memory usage refers to maximum number of models maintained at any one level of the exploration tree. A graph was considered solved if the true solution was retained in the hypothesis space after the exploration was complete. For each trial, the pruning method was applied first and the memory usage measured. The weighted search method was then run with the maximum memory usage (N) set to the value used by the pruning method on the same trial. Results from graphs of densities exceeding 1.6 could not be practically obtained using the pruning algorithm because of the memory usage required.

suggest limiting the growth of the exploration tree by pruning all models with more nodes than a threshold that is set based on the number of nodes in the current smallest incomplete model. Both the original pruning approach and the method we present here attempt to maintain simple solutions and discard more complex ones. The main difference is that while the original approach slows the growth of the exploration tree, the new approach places a bound on how many world models are maintained.

5. Discussion of Results

We examined our approach to topological mapping in this problem domain through a number of experiments. Our simulation tool takes as input: an undirected graph representing the world to explore; the exploration strategy employed by the robot; the number of observations to gather; and the number of world hypotheses N to maintain. The simulator then determines if the robot, after its exploration, maintains in its world hypothesis space a graph that is isomorphically equivalent to the input graph (and its ranking in our hypothesis space). The graphs considered were randomly generated planar graphs produced by selecting a connected sub-graph of the Delaunay triangulation of a set of random points.

For medium sized, sparse graphs, our heuristic approach to managing the size of the exploration tree was generally successful at retaining the correct solution by the time coverage of the graph was achieved provided an adequate number of hypotheses was maintained. This was true regardless of the exploration strategy used. Figure 5.6 shows an example of

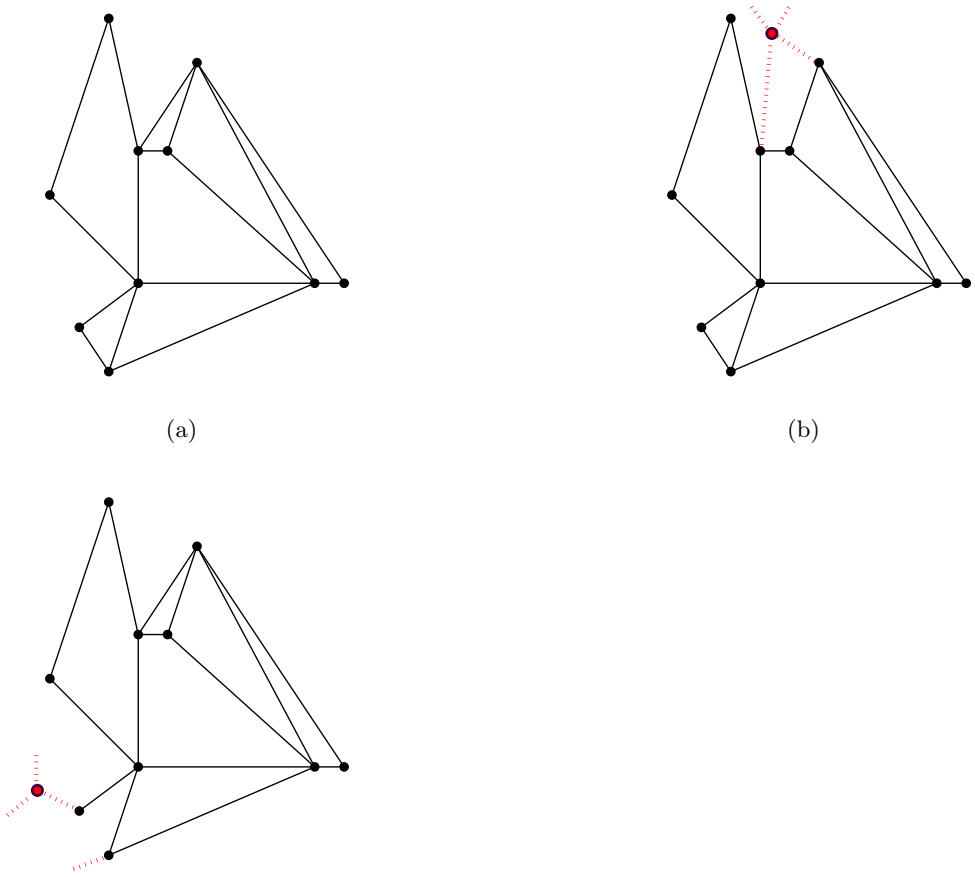


FIGURE 5.6. Example of the top three ranking world models, from a.) through c.), inferred by the algorithm with memory usage set to 20 models ($N = 20$) after running the BFET exploration strategy for 1000 steps on a 10 node graph with an edge to node ratio of 1.6. (Actual coverage was achieved at step 284.) The first ranked model is the correct one. Incorrect edges shown in dotted red.

a successful outcome on a ten-node graph. For each of the exploration strategies presented, the correct solution was found over 97 *per cent* of the time in 100 trials of ten-node graphs with node to edge ratios of 1.2 with 100 maintained hypothesis (Figure 5.7). For edge to node densities of 1.4, the correct solution was found over 92 *per cent* of the time in 100 trials of ten-node graphs and 100 maintained hypothesis (Figure 5.7).

Our weighted search algorithm performs well in comparison to the original pruning strategy presented in Dudek *et al.* [**33**]. Although effective at limiting the size of the exploration tree for simple graphs, the original approach occasionally prunes the correct solution

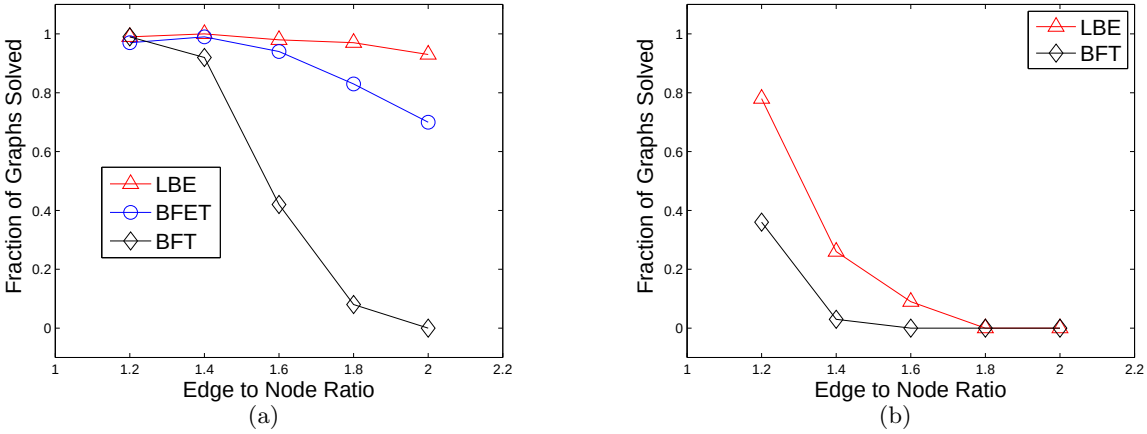


FIGURE 5.7. Fraction of graphs for which the true solution was retained in the hypothesis space after the exploration strategy under consideration reached edge coverage of the graph. Results were obtained from 100 trials at each edge density for graphs of size: a.) 10 nodes; and b.) 30 nodes. In this experiment 100 hypotheses were maintained by the mapping algorithm ($N = 100$). For LBE, the parameter p was assigned a value of 0.99. (BFET results were unobtainable for the larger graphs because of its poor cover time.)

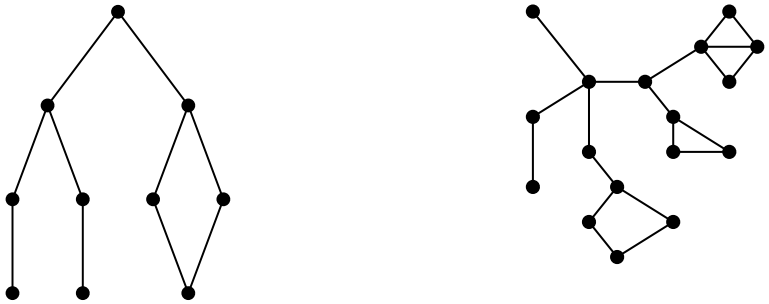


FIGURE 5.8. Examples of graphs solved previously in a.) [**34**] and b.) [**33**]. Each of these graphs were solved by our approach using LBE ($p = 0.99$) in less than half a second with $N = 1$; *i.e.* only one model was maintained throughout the exploration process (which was the correct one).

and potentially requires an unlimited amount of memory. We found that our current hypothesis selection algorithm was more accurate on average than the DST approach, when allowed the same memory usage (Table 5.1).

Figure 5.8 illustrates our approach on graphs considered in previous work. By using the LBE exploration strategy and the weighted search method, we were able to solve each

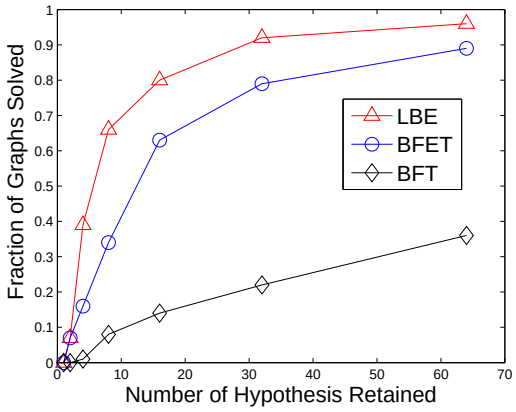


FIGURE 5.9. Fraction of graphs solved for different numbers of hypotheses maintained by the algorithm (value of N). Results obtained from 100 trials of 10 node graphs with an edge to node ratio of 1.6. For LBE, the parameter p was assigned a value of 0.99. The exploration strategy under consideration was run until edge coverage of the graph.

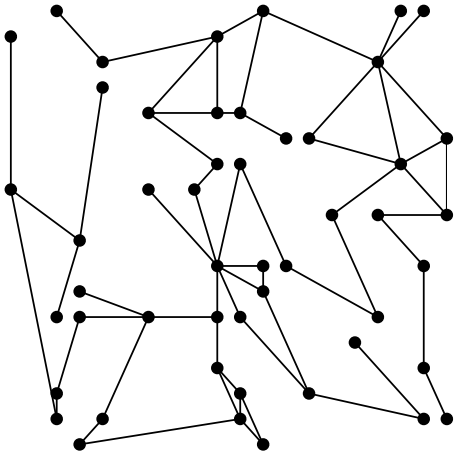


FIGURE 5.10. Example of a 50 node graph with an edge to node ratio of 1.2 that was solved by our approach in less than an hour. the correct graph was maintained by the algorithm (with $n = 1000$) as the first ranking model from the point of coverage onwards. LBE was used as the exploration strategy ($p = 0.99$) and achieved coverage at step 3918.

of these previously considered graphs while maintaining only one hypothesis for each step of the exploration process.

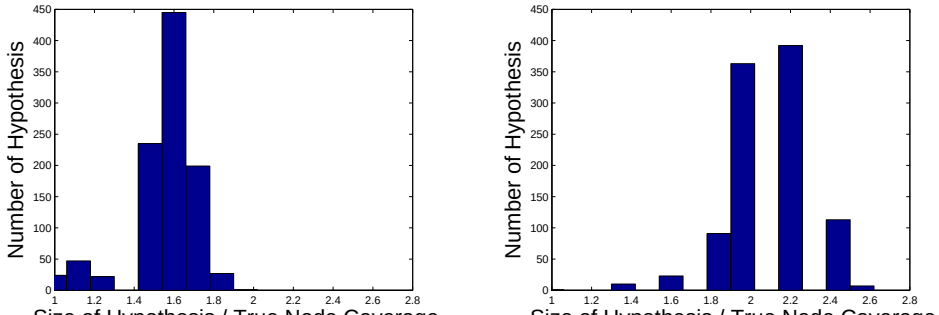


FIGURE 5.11. Distribution of the first 1000 hypotheses generated for a.) the BFT exploration strategy and b.) the BFET exploration strategy. The result was obtained from a typical run of the algorithm on a 10 node graph with an edge to node density of 1.6. BFT covered 7 of the 10 nodes in this time, while BFET covered only 5.

The difficulty of the topology inference problem increases with the density and size of the graph and the better performance of the new exploration strategies was apparent under the more difficult circumstances. Figure 5.7 shows a comparison of the different exploration strategies over ten-node and thirty-node graphs of various densities. Although the ranking results are not shown in these experiments, generally the correct graph was the first ranked model among those retained once coverage was achieved. Interestingly, the stochastic LBE exploration with a large enough value assigned to p, performed as good or better than the BFET strategy.

One parameter of interest when using the heuristic search algorithm is the number of models maintained. If not enough models are maintained throughout the exploration process, (the value assigned to N), then the chance of discarding the true solution is increased (Figure 5.9). However, for small graphs, good results can be obtained using LBE and BFET with just an arbitrarily small number of models. By increasing the number of models maintained, it is possible to correctly infer quite large graphs (Figure 5.10).

The *distribution* of the size of the hypotheses generated by the various exploration algorithms reveals that the newer strategies are better at discriminating among the smaller sized models, presumably by showing inconsistent errors of the MIS-CORRESPONDENCE and NEW-LOOKS-OLD types. For example, BFET quickly generates many hypotheses, a few of which are small and have stayed consistent through much exploration, and many

Strategy	**Mean Node Coverage**	**Normalized Model Size**
BFT	8.48 +/-(1.11)	1.22 +/-(0.19)
BFET	6.86 +/-(1.78)	1.67 +/-(0.30)
LBE ($p = 0.95$)	5.57 +/-(2.46)	2.15 +/-(0.57)
LBE ($p = 0.99$)	4.33 +/-(1.86)	2.56 +/-(0.79)

TABLE 5.2. Mean and standard deviation for coverage and model size normalized by coverage for the first 1000 hypotheses generated by the different exploration strategies. Results obtained from 100 trials on random 10 node graphs with an edge to node density of 1.6.

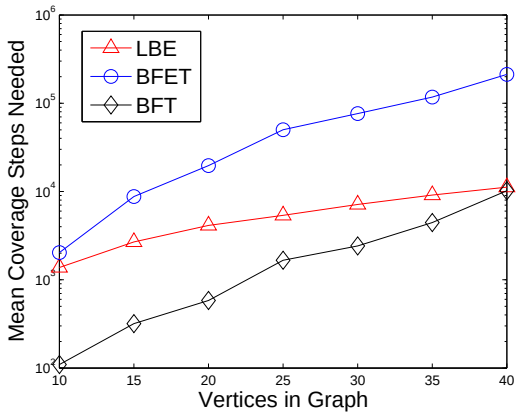


FIGURE 5.12. Average number of steps required for edge coverage of the graph for the different exploration strategies. Note the log scale for the vertical axis. Average was taken over 100 trials using an edge density of 1.6. For LBE, the parameter p was assigned a value of 0.99.

which are in relation quite large and therefore less believable (Figure 5.11). In one experiment, we allowed each exploration strategy to run until its corresponding hypothesis tree, (un-pruned), grew to 1000 models. Table 5.2 reveals the differences in the mean hypothesis size, (normalized by coverage), obtained over a number of trials for the different exploration strategies.

Although the BFET algorithm is guaranteed to cover a finite region, its cover time in practice was relatively poor (Figure 5.12). Unfortunately, this makes its use difficult for environments which are suspected to be large, since the probability of coverage would be low even after considerable exploration. In the environments we consider here, the LBE strategy does much better in practice, even with an aggressive value of p.

Edge to Node Ratio	Trials Solved		Mean Computational Effort	
	BFT	LBE	BFT	LBE
1.2	20	20	$1.08 * 10^3$	$1.11 * 10^3$
1.4	20	20	$5.98 * 10^3$	$6.86 * 10^3$
1.6	20	20	$3.02 * 10^4$	$2.2 * 10^4$
1.8	17	20	$2.76 * 10^5$	$5.95 * 10^4$
2.0	14	20	$4.31 * 10^5$	$8.93 * 10^4$

TABLE 5.3. Relative CPU time used by the BFT and LBE algorithms for different edge densities of 10 node graphs, averaged over 20 trials. Each exploration strategy was run until edge coverage of the graph under test was achieved. For both of the algorithms, iteratively larger values of N, starting with $N = 1$, were used on each graph until the graph was solved. After a failed attempt the value assigned to N was doubled. Attempts continued until success was obtained or the memory use exceeded 10000 hypotheses. Mean computational effort for each algorithm is reported as the average of the memory use N multiplied by cover time $|O|$ for each of the trials in which the graph was solved by both BFT and LBE.

For these experiments, a typical graph was usually solved (or not) in the order of a few minutes on a computer with a 2.2 GHz Intel Pentium 4 CPU and 1.00 GB of RAM using un-optimized Matlab code. The computational time required to run the topology inference algorithm was generally proportional to the number of hypotheses maintained (N) and the number of observations processed $|O|$. Since in these experiments the exploration strategy was followed until edge coverage of the graph was achieved, the relative processing required for each exploration strategy was proportional to its cover time (Figure 5.12) given equal memory requirements (the value assigned to N). The LBE strategy generally required less memory on average for a particular graph (Figure 5.9), however, and therefore was able to solve difficult problem instances faster than the other strategies. Table 5.3 compares the mean CPU time required to solve ten-node graphs of various densities for the LBE and BFT exploration strategies when a simple algorithm is used to adapt the N parameter to each graph. The LBE algorithm achieves better performance at the higher edge densities despite the larger number of steps required on average to cover the graph.

6. Discussion

In this chapter we have considered the topological mapping problem given a single mobile robot with extremely limited sensory capabilities. We have shown that even in the case of highly ambiguous, non-unique topological 'signatures' it possible for such a robot to infer a set of hypotheses for its environment that likely includes the true model. Our

approach combines an exploration strategy that attempts to eliminate inconsistent models with a beam style search that bounds the number of models maintained at each step based on the principle of Occam's razor.

Future work could consider the issue of handling more realistic sensory data. For example, incorporating additional, but still relatively poor, sensory data such as range only odometry into the inference process should greatly improve the accuracy of the approach. Additionally, it would be of interest to consider the effect of sensor errors; *i.e* missing or spurious observations. It is possible that these aims could by accomplished by shifting our heuristic based evaluation method to a probabilistic one. Such an approach could weigh the relative likelihood of the maintained models at any time based on previously calibrated measurement models and some prior over potential environments.

In the next two chapters, we will focus our attention on metric representations of the environment instead of topological representations. In the chapter that follows we will consider again a network of stationary sensing components, and afterwards, in the last portion of our investigations we will revisit mobile components, but in conjunction with stationary sensors.

CHAPTER 6

Probabilistic Self-Localization for Sensor Networks

In this chapter and the one that follows we consider the problem of recovering metric positional information from the environment. This is as opposed to topological positional information as has been presented up to now in this thesis. As discussed earlier, both topological and metric information play important roles in navigation and planning tasks. The inference of the relative metric positions of the components of a sensor network is commonly known as self-localization. In this chapter we consider the self-localization problem based on poor quality inter-sensor range data for a network of stationary components. In the next chapter we consider the localization problem with a network that has been augmented with a mobile robot, and utilizes its odometry measurements as a means of obtaining inter-sensor poses.

In these next two chapters we are interested in inferring a *probabilistic* representation of a sensor network pose in the form of a probability distribution function (PDF). We define this problem of obtaining and representing arbitrary distributions for the sensor locations as the *Probabilistic Sensor Localization Problem (PSLP)*. Our efforts in this area stand in contrast to previous self-localization work, most of which returns a single maximum likelihood estimate for the location of each sensor and fits a mathematically convenient distribution that can be expressed analytically to any uncertainty values.

By determining the PDF, the degree of certainty by which each sensor has been localized can be accurately indicated. This can be especially useful when dealing with the non-Gaussian and multi-modal distributions that can arise, for example, when only range data are available. The estimate of localization certainty can be used by a self-configuring

system to determine how additional resources should be used in order to improve localization accuracy; *i.e.* through the use of actuators, mobile components, or the additional deployment of sensors. Even if adaptation is not the goal, the certainty information can aid higher level applications relying on the metric localization data.

The self-localization method we present in this chapter uses an iterative Markov Chain Monte Carlo (MCMC) based algorithm to estimate a probability distribution function (PDF) for the network pose based on a measurement model for the collected range data. At each iteration, range data from those sensors best localized are incorporated into the algorithm. The final result of our algorithm is a particle representation of the PDF describing the position of each sensor.

In the research presented in this chapter, and to some extent in the thesis as a whole, we focus on computation and algorithmic complexity constraints and not on distributed computing. We assume that the sensor network has, or has assess to, the resources necessary for running computationally sophisticated algorithms. Note that the assumption of a hierarchical arrangement of network components based on computational power holds true for several real world sensor networks, especially in control and data collection systems [**123**] [**67**] [**129**]. For example, a typical network might contain a number of resource-limited sensors that pass messages using a wireless multi-hop protocol to a more powerful single-board 'gateway' computer which communicates to the outside world using a wireless Ethernet connection. In such an example, the gateway computer could periodically collect inter-sensor range data and update its network pose estimate using a version of the algorithm presented here.

In the remainder of this chapter we first provide a formal definition of the self-localization problem we are interested in and then we describe our MCMC-based approach to metric inference. Finally, we present an evaluation of our technique through simulations and give some discussion of the results.

1. Problem Description

The probabilistic sensor localization problem we are trying to solve in this chapter is to determine a PDF for the location of each sensor i in a network, given N sensors (including a number of beacons of known position), inter-sensor range data R, and a measurement

model L which characterizes the error in the distance estimates. The range data R consists of a (possibly incomplete) matrix $R = \{r_{ij}\}$ where each element r_{ij} represents a distance estimate between node i and j as measured by node i. In the case that each sensor has a range estimate to each other sensor, it is possible to obtain an estimate of the relative network pose using established multi-dimensional scaling (MDS) techniques [**8**], however, the problem is more difficult when some measurements between some sensors are unavailable.

Using the measurement model and the available inter-sensor range data, we can construct a model that returns the likelihood $p(d_{ij}|R, L)$ for any distance d_{ij} between sensors i and j as determined by sensor i. In the case of a range data estimate taken by sensor i, this is simply $p(d_{ij}|r_{ij}, L)$. In the case of missing measurements due to limited communication range, obstacles in the environment, or various other problems, the likelihood can be replaced with a distribution based on prior assumptions. For example, if communication signal strength is used for range estimates, the absence of a signal suggests a distance greater than some minimum value. In this case, a uniform distribution over a range of distance values might be appropriate.

The challenge is to provide a usable estimate of the PDF over all possible poses X. In traditional sensor-network self-localization, algorithms generally return an estimate of the maximum likelihood pose of each sensor. However, noise in the distance measurements dictates that any specific location estimate is actually a sample of a PDF, and the certainty of the measurement differs from node to node.

2. MCMC Sampling

To build a PDF for the pose of a network, our approach is to search over the space of possible network poses X, and sample configuration from areas of relatively high density using Markov Chain Monte Carlo (MCMC). These samples are then combined to form a particle representation of the PDF for the network pose. See Section 2.3 of Chapter 2 for a brief background on the technique of MCMC.

2.1. MCMC Sampling. We will first present a straight forward application of MCMC sampling to the problem presented in Section 1. In order to draw representative samples of the sensor locations, we construct the Markov chain using the Metropolis algorithm, an effective method of MCMC sampling [**124**]. Applying the Metropolis algorithm

to our problem requires that we can measure the relative densities of various network configurations and that we have a mechanism for proposing new configurations. The Metropolis algorithm constructs a Markov chain by accepting proposed transitions from the current state x_i to a new state x_j based on a probability determined according to:

$$\alpha = \min(1, \frac{\pi_j}{\pi_i})$$

where π_i and π_j are proportional to the density of the states x_i and x_j respectively. The proposal function should be crafted so that it has certain properties, *i.e.* it is possible to visit all states and the algorithm can not get caught in cycles (the resulting chain is ergodic). Given such a proposal mechanism, then after the initial configuration has been 'forgotten', (the chain has burned-in), the samples obtained from this technique should be representative of the underlying distribution. See Section 2.4 of Chapter 2 for additional background on the Metropolis algorithm.

For our problem instance, given the current state in the Markov chain X, specified by the combined location of each sensor in our network, we propose a symmetric transition to a new state X' by choosing a single sensor uniformly at random and altering its pose by adding a small amount of zero-mean, normally distributed noise. The new pose X' is a stochastic function based on the current pose X and is then accepted or rejected based on the acceptance probability:

$$\alpha = \min\left(1, \frac{p(X'|R, L)}{p(X|R, L)}\right) \tag{6.1}$$

where R is the observed range data and L is the measurement model. This proposal function can eventually allow the visiting of all potential states and will not get caught in cycles.

In order to apply Equation 6.1, we must be able to estimate the density of an arbitrary pose based on our measurement model L, and the measured range data R. The exact likelihood of a configuration X, where x_i gives the position of sensor i, can be evaluated according to Bayes Law:

$$p(X|R, L) = \frac{p(R, L|X)p(X)}{p(R, L)}$$

If we assume a uniform prior over network configurations X, our data R, and the measurement model L, then this equation simplifies to:

$$p(X|R, L) \quad \propto \quad p(R, L|X)$$

and if we further assume that the range data collected are independent of each other, then we arrive at the following equation for estimating the relative density of a specific network pose:

$$p(X|R,L) \quad \propto \quad \prod_{i=1}^{N}\prod_{j=1}^{N} p(d_{ij}|R,L) \tag{6.2}$$

where d_{ij} is the Euclidean distance between sensors i and j as determined by their locations x_i and x_j.

Theoretically, if we constuct a Markov chain as described in this section using Equations 6.2 and and run it long enough, then the samples drawn from the chain should be representative of the actual PDF for the pose of the network, however, in practice this is difficult. In was our finding that the chain as constructed above mixed relatively slowly and was ineffective at providing un-biased pose samples for all but the smallest of networks. In the case of exactly accurate range data, the problem of finding the maximum likelihood configuration for Equation 6.2 becomes that of embedding a weighted graph, which has been shown to be NP-hard [**113**]. There can be multiple (or infinite) number of realizations explaining a specific set of edge lengths. In the version of the problem with non-accurate edge data, these realizations correspond to different arrangements of the nodes that adequately explain the measurement data but result in substantially different pose estimates. Given perfect sensor fusion these cases should occur less often if there are large quantities of range data collected from a single region. In practice, however, alternate realizations of local groupings of sensors complicate the problem and lead to local minima.

2.2. Iterative MCMC. In this section we present an iterative, multi-chain sampling algorithm that attempts overcome some of the challenges involved in obtaining fair samples in this problem domain. We run parallel instances of a Markov chain, each instance, or *macro particle*, representing a single network pose estimate. We ultimately combine the samples obtained from each instance of the Markov chain in a description of a PDF for the pose of the network. We use an incremental approach of incorporating sensor information in order to avoid local minima as best as possible. The algorithm begins by sampling from a relatively simple underlying probability landscape and ends up drawing samples from the final distribution obtained by incorporating all the available range constraints.

The algorithm maintains a sub-group of nodes whose range data are used for localization. Sensors are incrementally added to the localizing sub-group based on the variance of their position estimates as maintained by each particle or instance of the Markov chain.

For ease of implementation, we assume the existence of a number of beacon nodes at known locations. However, it is possible to compute the relative sensor positions using our technique without the use of beacon nodes by specifying a preferred reference frame. For example, without loss of generality, one sensor can be forced to the origin, another to the horizontal axis, and a third to the positive vertical direction.

The full description of the algorithm is as follows:

(i) *Initialize Algorithm:* Initialize a localizing nodes sub-group $LocNodes$ to contain the beacon nodes of known position. Initialize a non-localizing nodes sub-group $NonLocNodes$ to include all the non-beacon nodes. Initialize M *particles* each maintaining a single estimate $X_m = \{x_{m1} \ldots x_{mN}\}$ for each sensor in the network.

(ii) *Update Particles Using Localized Sensors:* For each particle's estimate of the network pose X_m update the position estimate using MCMC as described in Section 2.1 with only the range data collected from those sensors in $LocNodes$. Each particle initializes the Markov chain with its previous belief of the network pose.

(iii) *Add to Localizing Sensors:* For each sensor i, compute the variance V_i of its position estimates $\{x_{1i} \ldots x_{Mi}\}$ as maintained by each particle. Add the k sensors with the lowest V values to $LocNodes$

(iv) *Iterate Until Done:* Iterate over steps 2 to 3 until all sensors have been inserted into $LocNodes$. The resulting M network position samples are now used to represent a PDF describing the positions of the sensors.

At each iteration the algorithm maintains a reasonable representation of the PDF given the sensor information incorporated up to that point. The algorithm depends on the manner of drawing representative pose samples, and on the manner of analyzing the variance of those samples in order to assess their accuracy.

In order to draw representative samples of the sensor locations, we maintain a separate Markov chain for each particle. The number of particles used during the algorithm affects

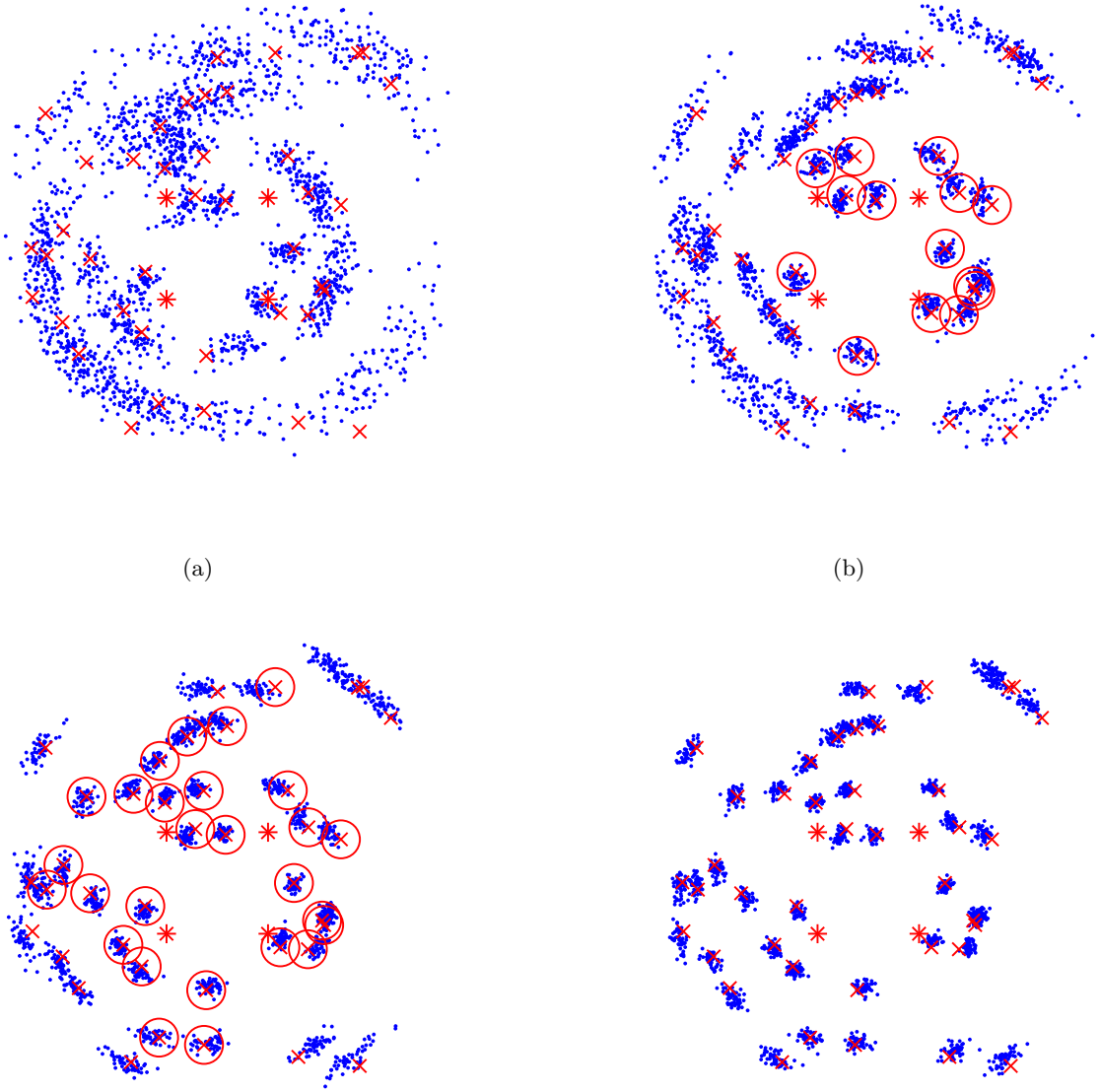
its ability to maintain a sufficient representation of the spatial probability distribution function for each sensor. Additional samples of the PDF can be drawn from the Markov chain maintained by any single particle, however, we suggest that these samples could share a common bias due to the slow mixing rate of the chain. Multiple particles, each randomly initialized, help to ensure a complete representation of the more complex distributions.

Although technically a new Markov chain is employed during each iteration of the algorithm, the burn-in time is minimal since each particle maintains its position estimate from the last iteration and uses it to initialize the chain. The additional range data added during each iteration re-shapes the target distribution; in most cases this extra information should help to further concentrate the PDF. The old localization estimate should be close enough to the new peak or peaks in the probability landscape that the MCMC should quickly approach the new steady-state distribution and afterwards provide meaningful localization samples. We run the Markov-Chain for a fixed number of proposals during each iteration of the algorithm which, for the moment, is hand-selected based on the scale of the problem. The number should be chosen such that the new configuration is given adequate mixing time in order to burn-in and reach a stable likelihood. We leave for future work the problem of adaptively optimizing the number of proposals per iteration.

A variance metric is used to quantify the certainty of a sensor's position based on the variance of the M position samples provided by each of the particles. The average Euclidean distance of each position estimate from the mean of the distance estimates is used as a metric. In the next section we will compare our iterative approach to the standard application of a MCMC sampler and further investigate the performance of the algorithm.

3. Results from Simulation

To evaluate the algorithm, a simulation was constructed based on a 2-D grid model of the environment. Sensors and beacons were distributed on a 100x100 grid. Four beacon nodes were placed near the centre of the grid, one in each quadrant, and the remaining sensors were distributed randomly within the area. Each sensor collected distance estimates to all neighboring sensors within some finite sensing range. Inter-sensor range data were



(a) (b)

(c) (d)

FIGURE 6.1. Example of localization results from a network of 40 sensors using 4 beacons and 50 particles. The stars indicate the beacon nodes and the crosses mark the true location of the sensors. a) The initial localization estimates using beacon data only. b) and c) Intermediate results incorporating data from the circled sensors. d) The final estimates incorporating the data from all the sensors.

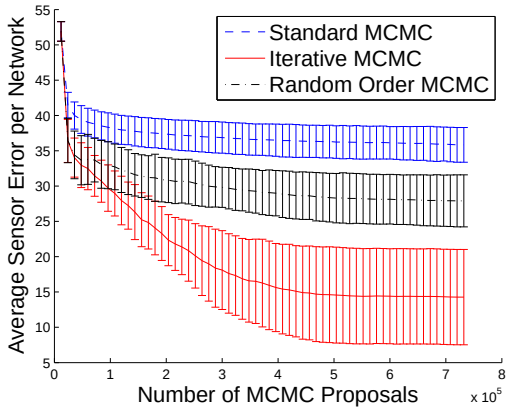


FIGURE 6.2. Plot of error as a function of Markov chain proposals averaged over 20 trials on networks of 4 beacons and 40 randomly distributed sensors. Error bars show one standard deviation. A sensing range of 30 units was used in this comparison. The Random Order MCMC algorithm is the same as the Iterative MCMC algorithm, but includes sensor data in random order.

drawn from a normal distribution with a mean equal to the true distance and a standard deviation equal to the square-root of the true distance.[1]

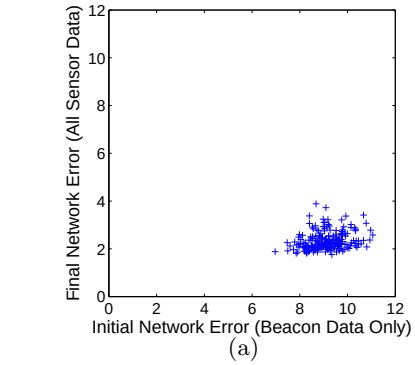
The algorithm was provided with the beacon locations, the beacon range data, the un-localized sensor range data, and an accurate measurement module. The resulting sensor localization estimates were assessed based on the mean Euclidean distance between the particle estimates and the true sensor location:

$$E_i = \frac{\sum_{m=1}^{M} d(x_{mi}, x_i')}{M} \tag{6.3}$$

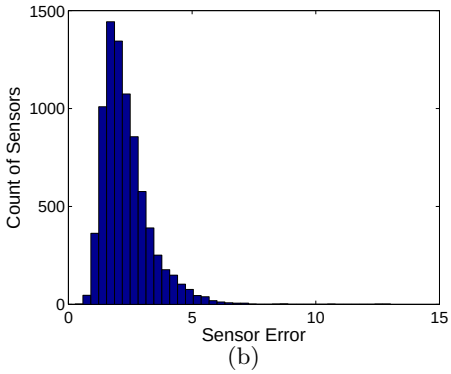
where x_i' is the true location of the sensor and $d(a, b)$ returns the Euclidean distance between a and b. Figure 6.1 shows an example of the localization algorithm on a 40 sensor network.

3.1. Assessment of Algorithm Performance. Our Iterative MCMC algorithm compares favorably to other sampling based approaches. In our experiments it both converges faster than our implementation of a standard MCMC sampler and returns a more accurate result. Figure 6.2 shows a comparison of the standard MCMC approach, our iterative approach, and a variant of the iterative approach on a difficult problem where the sensing range of each sensor is restricted to 30 units on a 100x100 unit grid. With this range, each sensor has a distance measurement to about ten other sensors on average. It

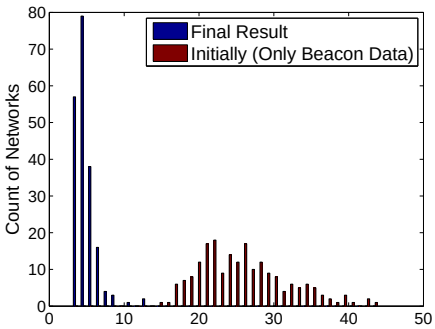
[1]Negative range estimates generated by this approach were truncated to zero.



(a)



(b)



(c)

FIGURE 6.3. Results from 200 trials of the algorithm on networks of 4 beacons and 40 randomly distributed sensors using an unlimited sensing range. a) Plot of final error output from the algorithm after including all sensor data as a function of the initial error calculated based on beacon data alone. b) Histogram of final sensor error across all networks (8000 sensors represented). c) Histogram comparing the error for the most poorly localized sensor in each network for the initial estimate using only beacon data and the final result. Vertical and horizontal axis for a) and the horizontal axis for b), and c) displays error in terms of distance units (simulation grid is of size 100x100).

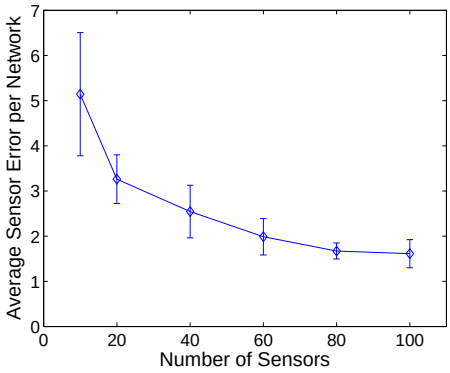


FIGURE 6.4. Average network error over 20 trials for different sized networks (confined to a 100x100 grid) with 4 beacon nodes. Error is displayed in terms of distance units and error bars show one standard deviation.

can be seen that the Iterative MCMC algorithm achieved a lower average per sensor error per network, as calculated by Equation 6.3, than the other approaches on twenty different trials of this problem type.

The effective performance of the algorithm depends on incorporating the sensor data in a manner that bootstraps each new Markov chain with likely pose values. For example, consider the extreme case of including the data of an isolated sensor for which there exists no connecting range estimate to any member of the localizing group. Clearly this information is not useful in refining the position of this node relative to the localizing group. Additionally, this type of sensor data can cause the algorithm to converge to a local minimum since isolated groups of network components can develop that have settled into a high likelihood arrangement of arbitrary orientation and offset. Ultimately, it can be difficult for the algorithm to reconcile such a group with the absolute positions of other sensors as dictated by the beacon nodes.

With a large sensing range, the algorithm was consistently able to use the noisy range data collected from the un-localized sensors to improve the final network pose estimate over that of the initial estimate using beacon data alone (Figure 6.3(a)). For example, in 200 trials of the algorithm on networks of 4 beacons and 40 randomly distributed beacons, the final average Euclidean sensor error, as calculated by Equation 6.3, was reduced to a 0.31 proportion of the initial result obtained by using beacon data only. The final average sensor error was 2.34 and the average particle variance was 2.83. Additionally, there was a high

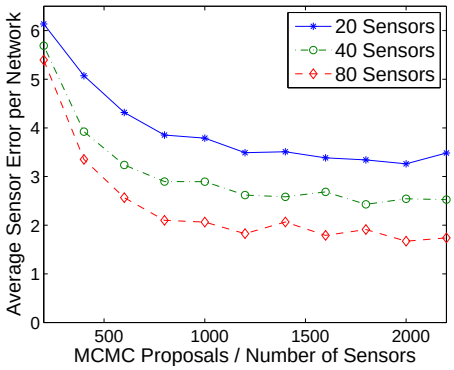


FIGURE 6.5. Final average sensor error averaged over 20 networks as a function of the total number of MCMC proposals divided by the number of sensors in the network.

level of consistency in the performance of the algorithm (Figure 6.3(b)). The final Euclidean error of the most poorly located sensor in each network was less than 5 units on a 100x100 grid for 98 *per cent* of the networks (Figure 6.3(c)). Furthermore, since a separate location estimate is provided for each particle used, poorly localized sensors should be identifiable based on the variance of their position estimates.

The performance of the algorithm improves as the density of sensors in the region increases (Figure 6.4). The higher sensor density results in multiple range estimates to any one sensor which are combined to reduce bias. Additionally, in our model, range estimates are of higher quality at shorter range.

Performance appeared to level off when the total number of proposals used in the MCMC increased beyond a certain threshold. This threshold appears to be proportional to the number of sensors in the network in the trials we have evaluated (Figure 6.5). In other words, each sensor appeared to require approximately the same number of proposals in order to converge to a final location distribution regardless of the size of the network the sensor belonged to. If this empirical observation holds under broader circumstances, then the rate of change in the variance between the macro-particles could be used as the terminating condition of the algorithm, and the computational time required to solve a network would be $T \in O(Nf(N))$ where N is the number of sensors and $f(N)$ is the computing power necessary to evaluate a single MCMC proposal as a function of the size of the network.

In the worst case, in which each sensor has a valid range estimate to every other sensor, $f(N) \in O(N)$ and therefore $T \in O(N^2)$. However, under realistic conditions, each

sensor will have some range beyond which distance estimates to other sensors are impossible or meaningless. Therefore, under ranged conditions a single MCMC proposal evaluation could be proportional to some range constant based on the ranging method employed, the environment, and the density of sensors.[2] Under these conditions $f(N) \in O(1)$ and therefore $T \in O(N)$.

4. Discussion

In this chapter we have presented and verified, through numerical simulations, an algorithm for the self-localization of a sensor network based on noisy inter-sensor range data. Unlike most previous related work, our method returns a representation of the PDF describing the position of each sensor in the network.

There are a number of open issues regarding improving the practicality of the algorithm. It would be interesting to explore run time optimization techniques, such as dividing larger networks into multiple regions, evaluating each region separately and then merging the final result. Also of interest is determining when a network has been localized to some degree based on an analysis of the final pose samples returned by the algorithm. This could allow the algorithm to terminate when a network has been adequately localized, or could be used to flag localization problems.

In the next chapter we consider the self-localization problem for a network that has been augmented with a mobile robot. Instead of utilizing inter-sensor range data, we consider using the odometry measurements of the robot to obtain relative pose estimates between the sensors.

[2]This would require discarding some additional 'negative' information which could be exploited regarding which sensors are not in range.

CHAPTER 7

Network Localization using a Mobile Robot

In this chapter we continue to look at the Probabilistic Sensor Localization Problem (PSLP) of inferring a *probabilistic* representation of a sensor network pose in the form of a probability distribution function (PDF). Unlike the last chapter, however, we assume that our network is augmented with a mobile sensor (or robot) which is capable of providing inter-sensor pose estimates through odometery measurements. The addition of a mobile robot gives the problem some similarity to the well studied SLAM problem in mobile robotics, and hence we are able to borrow and compare to techniques in that field.

1. Introduction

In the network localization scenario we consider in this chapter, the robot's motion through the network facilitates localization by explicitly transferring positional information between sensor locations. By maintaining an ongoing estimate of the robot's location, the position of any sensor it interacts with can be probabilistically estimated, (and updated), given the appropriate motion and measurement models.

Our approach is to employ a Markov Chain Monte Carlo (MCMC) based algorithm that allows us to sample from the probability distribution function (PDF) for the pose of a sensor network. We overcome the often-prohibitive computational effort required by MCMC approaches by employing the following techniques: 1) we employ a unique, odometry-specific proposal distribution that exploits the conditional dependencies present in our problem domain; 2) we apply Rao-Blackwellization to effectively reduce the dimensionality of our sample space; 3) we automatically tune the parameters of our proposal technique to achieve

desired acceptance rates; and 4) we employ convergence analysis based on the Gelman-Rubin statistic [**43**] as a stopping mechanism that informs us when the samples we have gathered closely represent the underlying pose distribution.

Our approach is capable of providing a full representation of the probability distribution function of the network pose. Information regarding the distribution enables later decisions to be made conditional on the confidence of the various pose estimates, and facilitates adaptive exploration processes and higher level task planning. We also consider a hybrid approach that uses our computationally expensive, but highly accurate, global inference technique as a corrective mechanism for a fast, online, established filtering method. In this manner we attempt to achieve a balance between the often conflicting goals of near real-time position estimation, and accurate, global inference.

The problem we consider in this work is similar to the simultaneous localization and mapping (SLAM) issue in traditional mobile robot research, but there are some key differences. Our sensors, which correspond to landmarks in the SLAM problem, are uniquely identifiable, so there is no correspondence ambiguity. Additionally, we can assume that the mobile robot will operate, for the most part, within the confines of a sensor-network deployed region and will ultimately visit the local area of each network component many times. Most importantly, in sensor network self-localization, the initial mapping effort is a one-time task, the results of which will likely be used for the lifetime of the network. Therefore the use of computationally sophisticated techniques is appropriate.

In the remainder of this chapter, we first provide some background on related work and then give a formal definition of the problem we are interested in. We then discuss the details of our approach to sensor network localization and assess its performance against those of established methods *via* simulations and real world experiments.

2. Problem Definition

The PSLP we consider in this chapter involves inferring the positions of each sensor node m_i, which is part of the map of the sensors $m^n = [m_1 m_2 ... m_n]$, based on measurements obtained by a robot (see Figure 7.1). These positions can only be measured relative to the position of the robot at a given time, s_t, which is the most recent component of the robot's path $s^t = [s_1 s_2 ... s_t]$ and so both quantities must be estimated simultaneously. The

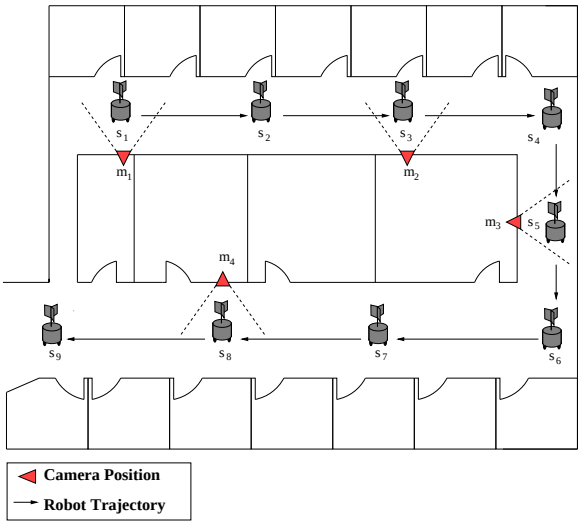


FIGURE 7.1. The mapping scenario described in this chapter. The robot moves through the environment gathering pose estimates to sensors and localizing its self as well as each of the encountered sensor in a common coordinate frame.

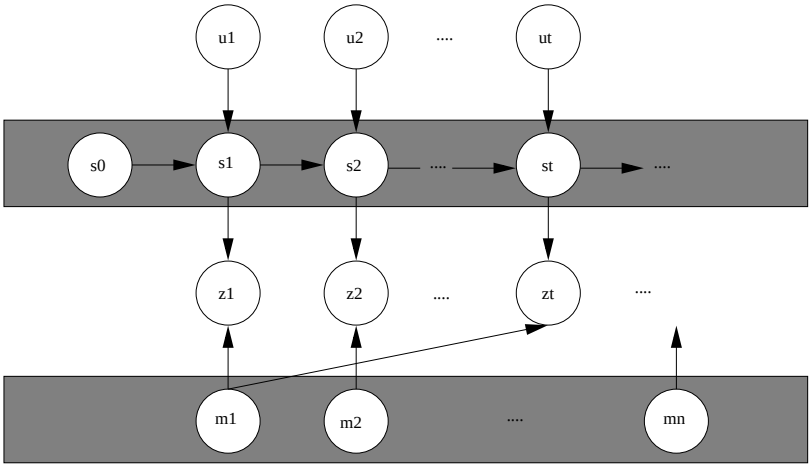


FIGURE 7.2. The quantities of interest in the Sensor Network localization problem can be modeled as a Bayesian Network in order to exploit conditional independencies. The robot poses and map (highlighted in grey) must be inferred, given observed data.

measurements available are the position of a sensor relative to the robot at time t, denoted z_t and the position of the robot at time t relative to its position at time $t-1$, denoted u_t.

This problem can be modeled as the probabilistic inference of the map and the robot poses conditioned on the observations, as represented by the underlying directed graphical model shown in Figure 7.2. The posterior distribution, $p(m^n, s^t|z^t, u^t)$, can be factored into

the product of many local conditional distributions, by exploiting the conditional independencies as is common practice for probabilistic graphical models.

For the sensor network localization problem involving a mobile agent, there are two classes of local conditional likelihoods: $p(s_t|u_t, s_{t-1})$ which is known as the motion, or odometry model of the robot; and $p(z_t|s_t, m_i)$, the measurement model which relates the poses of the sensor nodes to that of the robot. These two distributions can be determined empirically or by physical modeling for a particular instantiation. In the remainder of the chapter we will present a MCMC global inference technique that attempts to directly sample from the posterior distribution, $p(m^n, s^t|z^t, u^t)$, based on models of these two distributions.

3. Probabilistic Sensor Network Self-Localization using MCMC

Our global inference approach uses MCMC to generate a number of samples of the pose of the network according to our probability model. These are then used as a particle-based representation of the underlying probability distribution function. We form a graph $< V, E >$, where V is the set of vertices and E the set of connecting edges. The vertices of this graph are the robot positions over time s^t and the sensor locations m^n. The edges, or constraints, are the odometry measurements u^t connecting consecutive robot positions and the measured relative positions z^t between the robot poses and network components. Using a model characterizing the error in the measurements, we can calculate the density of any particular configuration $x = (m^n, s^t)$ through the application of Bayes law:

$$\begin{aligned} p(x|z^t, u^t) &= \frac{p(z^t, u^t|x)p(m^n, s^t)}{p(z^t, u^t)} \\ p(x|z^t, u^t) &\propto p(z^t, u^t|x)p(m^n, s^t) \end{aligned}$$

We assume that the prior, $p(m^n, s^t)$, is constant, so the relative likelihood of a particular configuration can be evaluated by factoring $p(z^t, u^t|x)$, into the product of the likelihoods of all constraints (edges) given our motion and measurement models. In this manner we

can evaluate the relative density of our target distribution given a configuration:

$$\begin{aligned} \pi(x) &= p(z^t, u^t|x) \\ &= \prod_k p(z_k|x) \prod_k p(u_k|x) \\ &= \prod_k p(z_k|s_k, \theta_k) \prod_k p(u_k|s_k, s_{k-1}) \end{aligned} \tag{7.1}$$

where θ_k indicates the sensor node $m_i, i \in \{1:n\}$ observed by (or observing) the robot at time step k.

Given this ability to calculate the relative density of our target distribution at a specific point, we can employ the Metropolis-Hastings (MH) algorithm [**124**] to generate representative samples. MH constructs a Markov chain by accepting proposed transitions from the current state x_i to a new state x_j based on their relative likelihoods and that of the proposal function. In theory this approach can be used to characterize any distribution given only the ability to calculate the target density and a reasonable proposal scheme. See Section 2.4 of Chapter 2 for additional background on the MH algorithm.

In our application of the MH algorithm to sensor network localization, we use a proposal function $Q(x'|x)$, which we will define below, that generates a new state x' given the current state x. The proposal x' is then either accepted or rejected with probability α, where α is calculated as:

$$\alpha = \min\left(1, \frac{\pi(x')Q(x|x')}{\pi(x)Q(x'|x)}\right) \tag{7.2}$$

3.1. Odometry-Specific Proposal Scheme. In order to improve mixing, we employ a proposal scheme which exploits domain knowledge regarding the sequential nature of our odometry measurements. We use the information that a change in position early in the odometry path of the robot effects its position from that point forward in time. To model this behaviour, the current state $x = (s^t, m^n)$ in the chain is altered to produce a new proposed state x' through the following procedure:

(i) A pose s_j is selected (as described shortly).

(ii) The initial $j-1$ robot poses, $[s_1, \ldots, s_{j-1}]$, are kept the same as in x.

(iii) The position of robot pose s_j is altered by the addition of zero-mean, normally distributed noise with a covariance Σ_j.

(iv) The effect of the change in s_j is propagated forward to change the locations of all following robot poses, $[s_{j+1}, \ldots, s_t]$. That is, the successive odometry constraints are kept rigid.

The above steps are repeated as new samples are required. In order to obtain a balanced sampling, iterated rounds of random selection without replacement are performed to select poses. Step 4 in this procedure temporarily blocks correlated components of the state space together during proposals in order to facilitate more rapid mixing. Blocking is not an uncommon technique when correlation is present among features in the target distribution and has been shown effective in the past; *e.g.* see [**48**]. If step 4 was skipped, the resulting method could be considered a variant of single-component Metropolis-Hastings, a common technique for constructing a Markov chain in high dimensional state spaces [**45**].

While this proposal method applies to the odometry portion of our state space, there remains the component made up of the sensor positions. In the next section, we will describe the Rao-Blackwellization (RB) process that approximately marginalizes out this factor of the joint distribution allowing for a further improvement in mixing. Instead of using RB, however, one may alternate the proposal scheme described above for the robot poses s^t with one that proposes alterations individually to each of the sensor positions $m_1, \ldots, m_n$ in turn. Here the ordering in which the proposals to the sensors are made will not effect the outcome since the processes are independent. In the next paragraph, we will briefly describe such a proposal scheme for the sensors m^n as it will aid the description of the RB step.

For sensor node m_i, we generate a proposal distribution $Q_i(x'|x)$ that is based on constraints existing between sensor m_i and any of the robot poses. Specifically, let θ_k indicate the sensor node $m_i, i \in \{1 : n\}$ observed by (or observing) the robot at time step k , as defined earlier. Now let S_i represent the set of those poses such that if $s_k \in S_i$ then $\theta_k = m_i$. Now let Z_i represent the corresponding set of constraints (or measurements) providing pose estimates between sensor m_i and each of the robot poses $s \in S_i$. A linear approximation is then applied to the measurement model, yielding a separate Gaussian distribution for m_i given each pose $s \in S_i$ and its corresponding measurement $z \in Z_i$. The

product of these separate distributions is calculated and used as the proposal distribution Q_i for the location of m_i. A sample is then drawn: $(x', y', \theta') \sim Q_i$ as a potential new location for m_i and is accepted or not based on the equation (from Equation 7.2):

$$\begin{aligned} \alpha &= \min\left(1, \frac{\pi(x')Q(x|x')}{\pi(x)Q(x'|x)}\right) \\ S &= \min\left(1, \frac{p(m_i'|S_i, Z_i)Q_i(m_i|S_i, Z_i)}{p(m_i|S_i, Z_i)Q_i(m_i'|S_i, Z_i)}\right) \end{aligned} \tag{7.3}$$

where m_i' represents the newly proposed location for m_i. This proposal scheme takes advantage of our current belief of where the robot poses are situated in order to explore high probability regions in which the sensors could be located. Note that in the case that no approximation is necessary in this step, for example if the measurement model is already Gaussian, then the acceptance ratio is always one and this mixing approach is equivalent to a Gibbs-style proposal.

3.2. Rao-Blackwellization. Through a relatively benign approximation that has been used by techniques in related domains we can greatly accelerate the mixing rate of our chain. Although our method is only guaranteed to produce exactly representative samples for certain classes of models, a good approximation to the real distribution is obtained in most circumstances, and the results considerably exceed the efforts achieved by standard filtering techniques under the same circumstances.

Instead of sampling from $p(s^t, m^n|z^t, u^t)$ directly, we sample from the factor $p(s^t|z^t, u^t)$. This is accomplished by approximating $p(m^n|s^t, z^t, u^t)$ with a closed form and marginalizing this factor out. We use a product of Gaussians similar to the proposal function Q described in the previous section which is obtained by linearizing the measurement model, yielding separate distributions for m_i given each pose $s \in S_i$ and its corresponding measurement $z \in Z_i$. We then take the product of these separate distributions, $q_i(m_i|s^t, z^t, u^t)$, as an

approximation to $p(m_i|s^t, z^t, u^t)$. We can now approximate $p(s^t|z^t, u^t)$ as follows:

$$\begin{aligned} p(s^t|u^t, z^t) &= \int p(s^t|u^t)p(m^n|s^t, u^t, z^t)dm^n \\ &= p(s^t|u^t)\prod_{i=1}^{n}\int p(m_i|s^t, u^t, z^t)dm_i \\ &= p(s^t|u^t)\prod_{i=1}^{n}\int p(m_i|S_i, Z_i)dm_i \\ &\approx p(s^t|u^t)\prod_{i=1}^{n}\int q_i(m_i|S_i, Z_i)dm_i \\ &\approx p(s^t|u^t)\prod_{i=1}^{n} E[q_i(m_i|S_i, Z_i)] \end{aligned} \tag{7.4}$$

where S_i and Z_i are as defined in the previous section. Given samples drawn from the approximation of $p(s^t, |z^t, u^t)$, we can characterize the target distribution $p(s^t, m^n|z^t, u^t)$ using either the approximation to $p(m^n|s^t, z^t, u^t)$, or the real distribution; the second requiring the use of a technique such as importance sampling. The process we describe in this section, in which the accuracy of an estimator is improved by marginalizing out variables, is a technique referred to as Rao-Blackwellization [**17**].

3.3. Automatic Tuning. To ensure adequate mixing, the chain is run for an initial tuning period during which the proposal parameter values for each component of the state space, (specifically, each robot pose s_t), are automatically adjusted to approach a desired mixing ratio, L. This tuning period is divided into a number of smaller time windows $[t_1, t_2 \ldots]$ in which the chain is run for some fixed number of proposals. The proportion of proposals accepted for each component, j, is then calculated for the current time window t, and this value is compared to the target mixing ratio L and adjusted accordingly using an exponential averaging scheme for use in the next time window. After time window t, The Σ value for each component is adjusted as follows:

$$\Sigma_j^{t+1} = \begin{cases} \Sigma_j^t(1-\alpha) - CI\alpha, & M_j^t < L - \delta \\ \Sigma_j^t(1-\alpha) + CI\alpha, & M_j^t > L + \delta \\ \Sigma_j^t, & L - \delta <= M_j^t <= L + \delta \end{cases}$$

where M_j^t is the mixing rate of component j during time window t, I is the identity matrix, and α, δ and C are selected for appropriate learning. After the initial tuning period the Σ values are fixed.

3.4. Stopping Mechanism. One of the key issues when employing MCMC is determining how long it takes for a set of samples drawn from the chain to approach the target distribution. In practice, the initial portion of the chain is discarded to reduce the correlation of subsequent samples with the starting point and to allow the chain to move into high likelihood, representative configurations; *i.e.* the burn-in period. Typically, after the burn-in period, samples are drawn periodically from the chain, with some fixed number of proposals in-between each sample. Given adequate mixing, these samples are then taken as representative of the target distribution.

As an indicator of convergence our approach employs the Gelman-Rubin statistic [**43**], which is based on a variance analysis of instances of the chain restarted from different initial positions, (which should be over-dispersed with respect to the target distribution). The resulting value, calculated for a single feature, is referred to as a *potential scale reduction factor* (PSRF) and suggests how the estimated variance for the feature under consideration could be improved by additional simulations. The key idea is that if the system has converged, then the samples should exhibit a large degree of agreement regarding the statistics of the problem. Essentially, a PSRF value near one suggests suggests that each of the restarts obtained samples that share similar characteristics and therefore are presumably close to the target distribution. On the other hand, a PSRF value far from one suggests that a full tour of the target distribution has not yet been obtained. A $PSRF < 1.2$ is sometimes used as a guideline for "approximate convergence" [**12**].

When attempting to obtain a particle representation for the PDF of the network pose, our technique employs a number of instances of the algorithm described above running in parallel. Each separate instance starts from a different initial configuration and runs independently. After each instance has run for some set number of proposals, we calculate the Gelman-Rubin statistic for a number of indicator features, namely the X and Y coordinates of the sensor positions. We employ the maximum $PSRF$ value obtained from these indicator features as a metric to assess convergence. Under normal operation of our algorithm, if the calculated value of this metric falls below a threshold value, (*e.g.* 1.2), then

simulations are halted, and the samples from each of the parallel chains are combined as the output. Otherwise we resume the simulation, and continue to obtain samples until our metric suggests that we are near convergence. For the sake of convenience and brevity, in the remainder of this paper we will refer to the parallel instances of the algorithm, described above, as *restarts* and our calculated convergence metric as the *PSRF* obtained.

4. MCMC corrected Filter-based Localization

One possible application for the MCMC localization algorithm described above is to use it as a complement to faster, filter-based localization techniques such as an extended Kalman filter (EKF) or a Rao-Blackwellized particle filter (RBPF). By incorporating all the information gathered by the robot and the stationary sensors, the MCMC algorithm can generate an accurate estimate of the PDF for the network pose. Although we employ both Rao-Blackwellization and a proposal scheme which exploits the sequential nature of odometry measurements in order to improve the efficiency of the approach, the global MCMC algorithm is relatively computationally sophisticated in comparison to filter-based localization approaches.

In a hybrid sensor network localization approach, we assume available a fast filter-based localization technique that can be used when there has not been the opportunity to run the MCMC localization technique. This can be useful for the initial exploration, and also when quick adjustments are required such as when a new network component is added, or an existing component fails. Additionally, the filter can be employed for everyday navigation tasks once the relative position of the network components has been determined to satisfaction. The MCMC algorithm could be run, periodically, after some amount of exploration, or perhaps continuously as a background process. The results could then be used to improve the localization estimates obtained from the filter-based approach.

To investigate this concept we consider a hybrid approach composed of an EKF algorithm and our MCMC algorithm. In this localization scenario, the MCMC is run periodically, and used to correct the mean of the EKF. When employed, the MCMC algorithm is bootstrapped with the EKF mean and only run until shortly after the probability of the samples obtained levels off. The final sample obtained is then used to replace the current EKF mean (see Figure 7.3). In this situation, the MCMC algorithm is employed as an

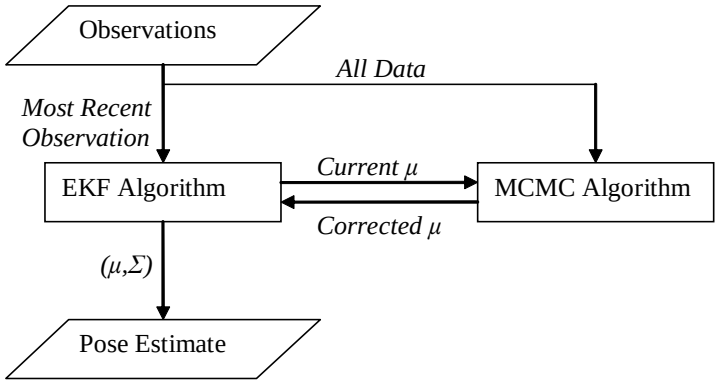


FIGURE 7.3. Flow chart depicting an example hybrid approach to network localization in which the global MCMC algorithm is run periodically to correct the mean of the EKF.

optimizer to improve the probability of an existing configuration, as opposed to a sampler. Starting with the EKF mean increases the effectiveness of the MCMC process by allowing the chain to begin with a relatively high likelihood sample. Generally, this reduces the time required by the chain to find a configuration near a local maximum (which is hopefully the global maximum). Correction from the MCMC approach should increase the effectiveness of the EKF by reducing the build-up of error which occurs after repeated linearization. Ideally, this allows the hybrid estimation method to remain accurate over longer periods and in the face of larger measurement noise.

5. Results from Simulations

We investigated the performance of our localization algorithms on data obtained from a realistic simulator. Using this simulator we found evidence demonstrating the superior ability of our MCMC-based localization algorithm to accurately represent network pose distributions in comparison to two filtering techniques commonly used in similar domains: the Rao-Blackwellized particle filter (RBPF), and the extended Kalman filter (EKF).

5.1. Simulation Details. Our simulator places N sensor nodes uniformly at random on a two-dimensional plane. These nodes are connected *via* potential pathways by selecting a sub-graph of the Delaunay triangulation. (A random graph synthesis technique employed throughout this thesis.) The motion of a robot is then simulated through this environment as T distinct steps made from the region of one sensor to another (see Figure

Noise Level	Measurement Model		Motion Model	
	Positional Noise (cm)	Angular Noise (rad)	Positional Noise (%)	Angular Noise (%)
Low	5	0.0125	2.5	2.5
Moderate	10	0.025	5	5
High	20	0.05	10	10

TABLE 7.1. Table of different noise levels used in simulations. Values given in standard deviations. In these experiments the motion model noise is dependent on the amplitude of the motion, while the noise added to a sensor measurement is independent of the actual distance from which it was taken.

7.7). To choose a destination sensor, a quasi-random walk strategy is employed. That is, the robot first selects uniformly at random from the *un-visited* neighbours of its current sensor. If there are no un-visited neighbours, a neighbor is re-visited at random.

At each step t, the robot first executes a rotation, changing only its orientation, and then performs a translation in the new direction. These two motions are captured as the odometry measurement u_t. At the end of the translation, a new measurement z_t is obtained from the sensor located nearest the region occupied by the robot. For these experiments, zero-mean, normally distributed noise is added to the odometry measurements for each of the rotation and translation motions, and also for the measurement. We assume that the motion and measurement model utilized by our localization algorithms are accurate; that is, we know the mean μ and covariance Σ for each of the noise signals added to our measurements. Table 7.1 shows noise parameters used for different levels of noise in our experiments.

In order to provide benchmarks with which to compare the performance of our localization algorithm, we implemented for comparison purposes, two popular Bayesian filtering approaches: a Rao-Blackwellized particle filter (RBPF), see [**31**] [**90**], and also an Extended Kalman Filter (EKF), see [**81**]. The EKF approach was based on the methodology described in [**106**].

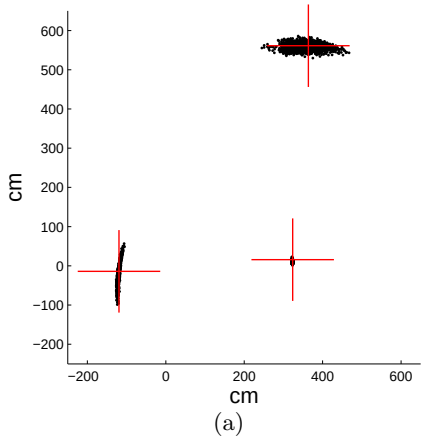
In the case of the RBPF, we considered both a 'basic' variant that uses the true motion model as the proposal distribution and also a variant that applies a linearization to the two-step motion (rotation and translation) and incorporates the most recent evidence (also through linearization) into a closed form proposal. For both versions, the sensor node distributions were maintained internal to each particle as Gaussians. Although a full discussion

of the comparative performance of these two variants would detract from the focus of this chapter, briefly we can report that when the noise parameters used in the motion and measurement models of our simulation were sufficiently increased, the 'basic' variant generally performed as well or better as the second variant given the same computational effort. We suggest that this was because the proposal distribution of the second variant became less accurate under noisy conditions due to its reliance on linearization assumptions. Where RBPF results are reported, we present data from the variant with the best performance.

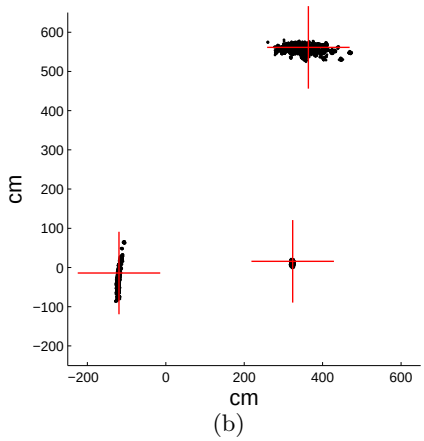
5.2. Performance Analysis. The simulation results presented in this chapter suggest that our technique out-performs both the EKF and the RBPF at the task of inferring a network pose distribution. Figures 7.4, 7.5, and 7.6 show the results obtained from the different inference algorithms on the same simulation data for a small scale version of the localization problem under different noise conditions; see Table 7.1 for a description of the noise parameters. For the MCMC approach, each restart is initialized with values obtained by running the RBPF with only enough particles to maintain a non-zero probability configuration. As described in Section 3, in this experiment, all of the MCMC restarts are run until the set of samples produced by each instance has similar statistical characteristics. Although it provides no guarantees, this analysis gives strong evidence for the convergence of the employed chain in the problems considered in these experiments and suggests that the results obtained should closely reflect the actual distribution suggested by the data and models used by our simulator.

Figures 7.4, 7.5, and 7.6 illustrate the output of the different algorithms for a small scale localization scenario. It can be observed that the RBPF when used with $K = 5000$ particles produces a similar distribution to the MCMC algorithm, although the samples are not as homogeneously distributed.[1] Linearization approximations made by the EKF along with its limited expressiveness reduce its accuracy and hence its output is the most different from the MCMC result. This difference is most apparent under high noise conditions.

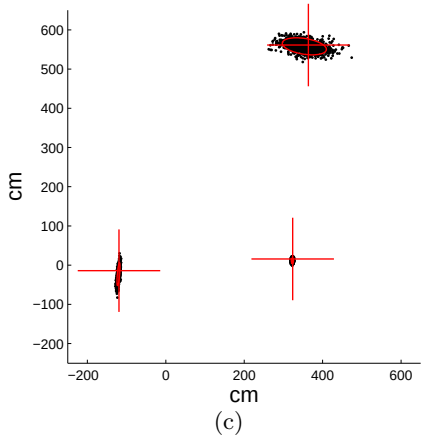
[1]As a final step, in both our RBPF implementation and the MCMC algorithm, the sensor locations are sampled from the closed form approximation to their distribution given the robot poses. This is done for each sample/particle obtained.



(a)

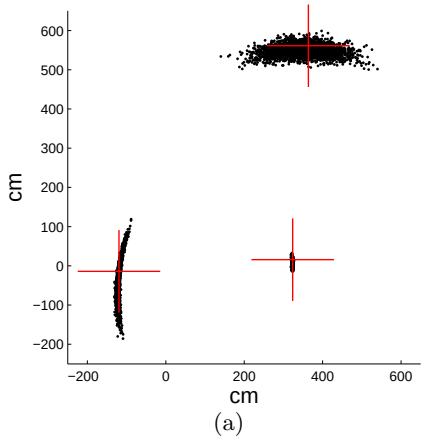


(b)

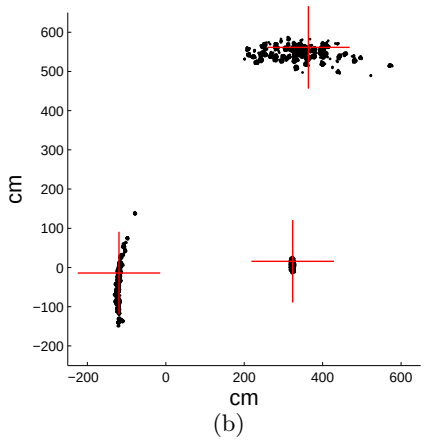


(c)

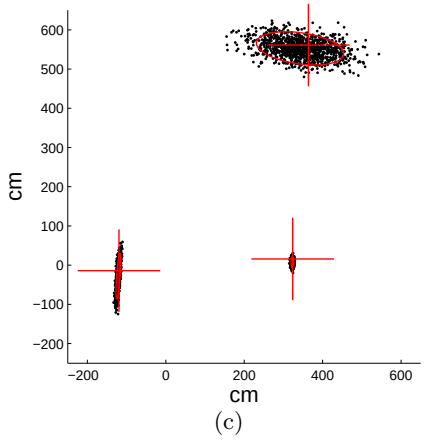
FIGURE 7.4. Results obtained on data obtained from the simulator with *low noise* for a robot path of 4 steps through a 3 sensor network for the algorithms: a.) MCMC b.) RBPF ($K = 5000$), and c.) EKF. The crosses indicate the ground truth sensor positions. For the EKF, the samples are drawn from the mean and covariance obtained for the position of the sensors; a three standard deviation uncertainty ellipse is overlaid.



(a)

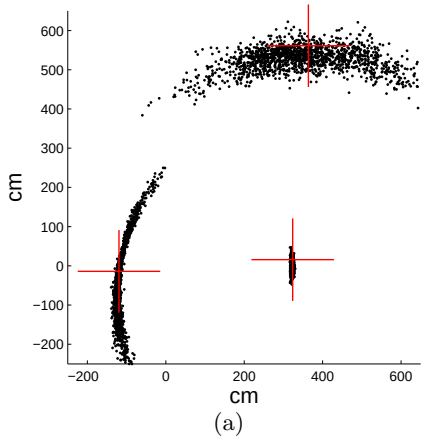


(b)

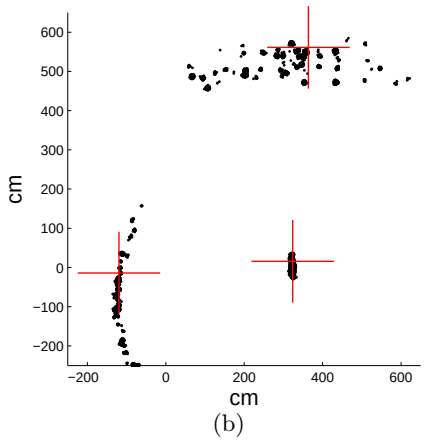


(c)

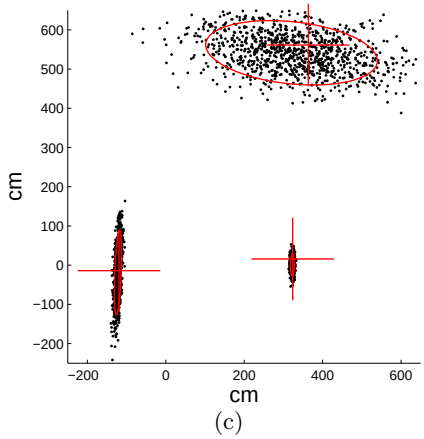
FIGURE 7.5. Results obtained on data obtained from the simulator with *moderate noise* for a robot path of 4 steps through a 3 sensor network for the algorithms: a.) MCMC b.) RBPF ($K = 5000$), and c.) EKF. The crosses indicate the ground truth sensor positions. For the EKF, the samples are drawn from the mean and covariance obtained for the position of the sensors; a three standard deviation uncertainty ellipse is overlaid.



(a)



(b)



(c)

FIGURE 7.6. Results obtained on data obtained from the simulator with *high noise* for a robot path of 4 steps through a 3 sensor network for the algorithms: a.) MCMC b.) RBPF ($K = 5000$), and c.) EKF. The crosses indicate the ground truth sensor positions. For the EKF, the samples are drawn from the mean and covariance obtained for the position of the sensors; a three standard deviation uncertainty ellipse is overlaid.

MCMC Within-Cloud $M = 5$, $N = 480$ (10 Comparisons)	$\mu = 15.21$ $\sigma = 0.68$	
Algorithm	$\boldsymbol{D_h}$	$\boldsymbol{\lvert D_h - \mu \rvert / \sigma}$
RBPF ($K = 20000$)	15.50	1.85
RBPF ($K = 10000$)	18.69	6.55
RBPF ($K = 5000$)	22.92	12.74
RBPF ($K = 1000$)	43.75	43.22
EKF	73.05	86.07

TABLE 7.2. Comparison of the results obtained from the different algorithms using the Hausdorff distance metric on data obtained from the simulator for a robot path of 4 steps through a 3 sensor network with moderate noise.

In order to quantitatively compare the distributions obtained from the different approaches, we employed the generalized Hausdorff distance:

$$D_h = k\text{th} \sup_{a \in A} \inf_{b \in B} (||a - b||)$$

where $||a - b||$ is calculated using a L2-norm and k is set to to the $95th$ quantile. In order to correctly interpret a value obtained from this metric we divide the control particle cloud into M disjoint sets of samples, each of size N. We then calculate the mean and standard deviation of the D_h value found between each pair of sample sets. When interpreting a distance value found between the control set and a comparison set, each of size N, we can measure the number of standard deviations between the new distance and the previously computed control value. One would expect similar particle clouds to yield values within three standard deviations or so, while clouds with significant differences should obtain larger values.

Table 7.2 shows the distance metric values obtained under moderate noise conditions when the particle clouds from the different algorithms are compared to the MCMC result; the same experiment presented in Figure 7.5. It can be seen that while the PDF suggested by the EKF is significantly different from the MCMC result, the performance of the RBPF improves as a function of the number of particles used. For this size of a problem, the data obtained from the RBPF with 20,000 particles is not significantly different from that of the MCMC technique. We observed a similar result for the RBPF across the three different noise levels we considered in this problem. The EKF, however, performed better under lower noise conditions.

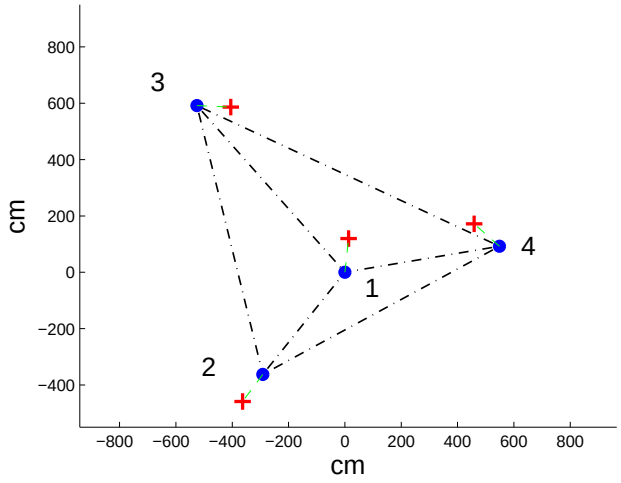
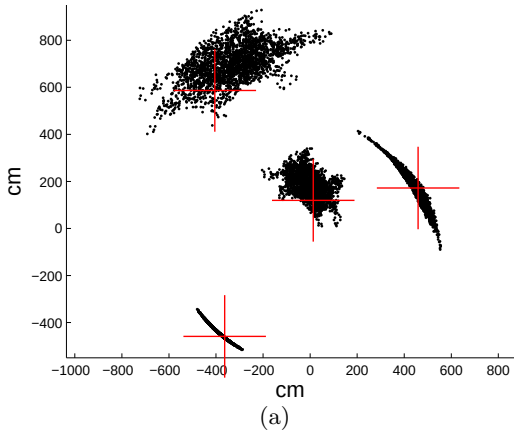


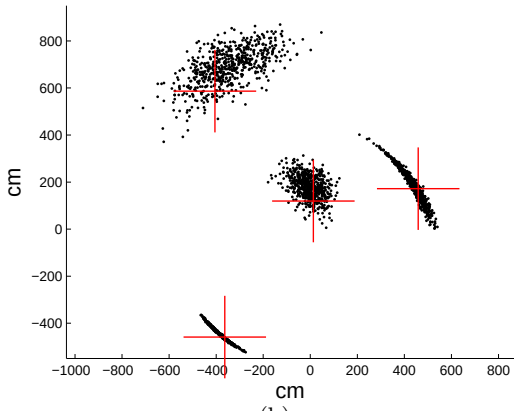
FIGURE 7.7. Example simulated sensor network environment. The red crossed indicate sensor positions, the blue circles indicate regions near each of the sensors which may be visited by the mobile robot, and the dotted lines indicate potential pathways

As the scale of the problem increases, however, it becomes increasingly difficult for the filtering techniques to accurately characterize the distribution. For example, figures 7.7, 7.8, 7.9 and 7.10 show the results obtained from the different algorithms on the same network with moderate noise as the path length of the robot increases. From the MCMC result, it can be seen that the uncertainty of the distribution suggested by the MCMC result decreases as more information is incorporated into the estimate in the form of additional sensor measurements. When the path length is short, the filtering approaches depict approximately the same PDF as the MCMC result. As path length taken by the robot increases, however, the support of the distributions suggested by the filtering approaches rapidly decreases compared to the MCMC result. For example, the PDF suggested by the RBPF collapses to a single small region after only 9 steps by the robot, even with 20000 particles. Although not as extreme, the distribution suggested by the EKF also shrinks considerably at this point. At the end of this simulation the distributions suggested by both the RBPF and the EKF do not contain the ground truth position of any of the sensors.

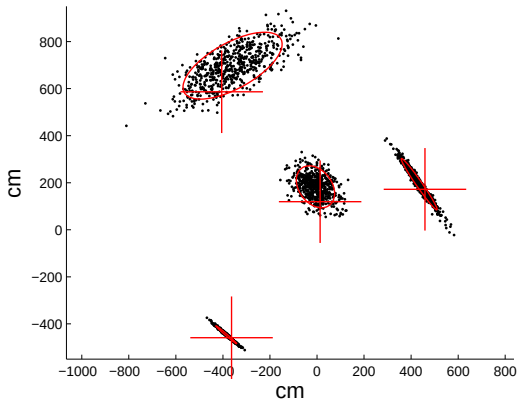
Figure 7.11 shows an example of results obtained from the different inference algorithms on a moderately sized problem in which the robot visits each of the sensors a number of times. In this problem instance, while both the EKF and the RBPF provide good estimates



(a)

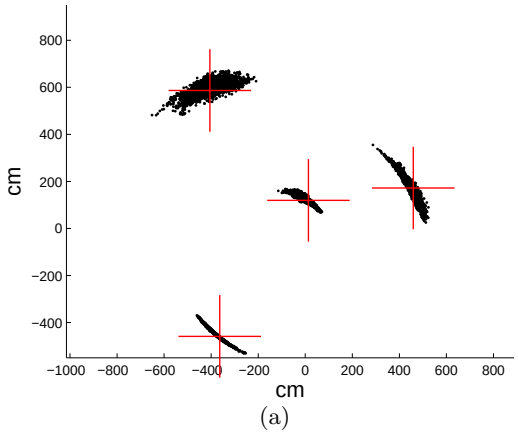


(b)

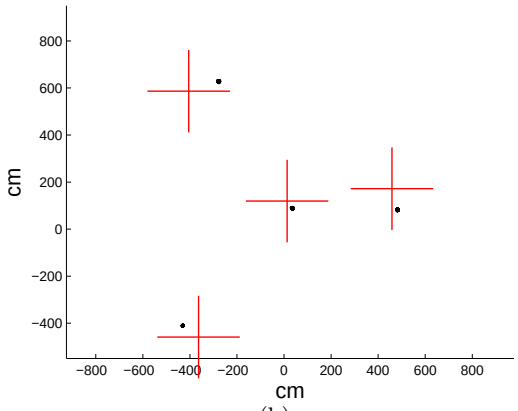


(c)

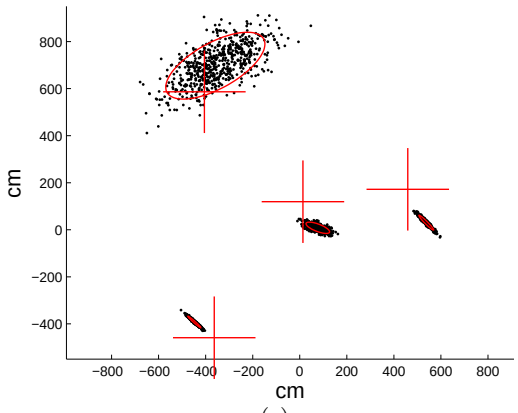
FIGURE 7.8. Results obtained on data obtained from the simulated environment shown in figure 7.7 with moderate noise after the robot visited sensor regions: (1, 2, 4, 1, 3), for the algorithms: a.) MCMC b.) RBPF ($k = 20000$) (not all particles shown), and c.) EKF. the crosses indicate the ground truth sensor positions. for the EKF, the samples are drawn from the mean and covariance obtained for the position of the sensors; a three standard deviation uncertainty ellipse is overlaid.



(a)

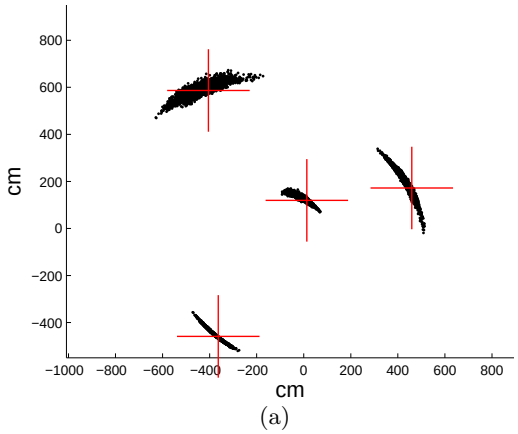


(b)

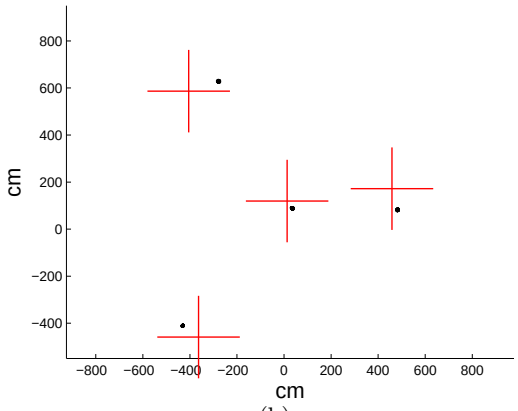


(c)

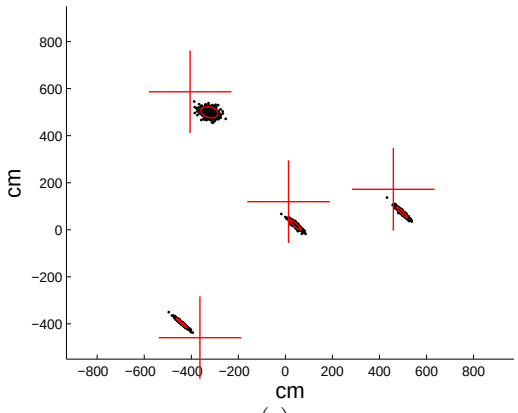
FIGURE 7.9. Results obtained on data obtained from the simulated environment shown in Figure 7.7 with moderate noise after the robot visited sensor regions: (1, 2, 4, 1, 3, 2, 4, 2, 1), for the algorithms: a.) MCMC b.) RBPF ($K = 20000$) (not all particles shown), and c.) EKF. The crosses indicate the ground truth sensor positions. For the EKF, the samples are drawn from the mean and covariance obtained for the position of the sensors; a three standard deviation uncertainty ellipse is overlaid.



(a)



(b)



(c)

FIGURE 7.10. Results obtained on data obtained from the simulated environment shown in Figure 7.7 with moderate noise after the robot visited sensor regions: (1, 2, 4, 1, 3, 2, 4, 2, 1, 4, 1, 3, 1), for the algorithms: a.) MCMC b.) RBPF ($K = 20000$) (not all particles shown), and c.) EKF. The crosses indicate the ground truth sensor positions. For the EKF, the samples are drawn from the mean and covariance obtained for the position of the sensors; a three standard deviation uncertainty ellipse is overlaid.

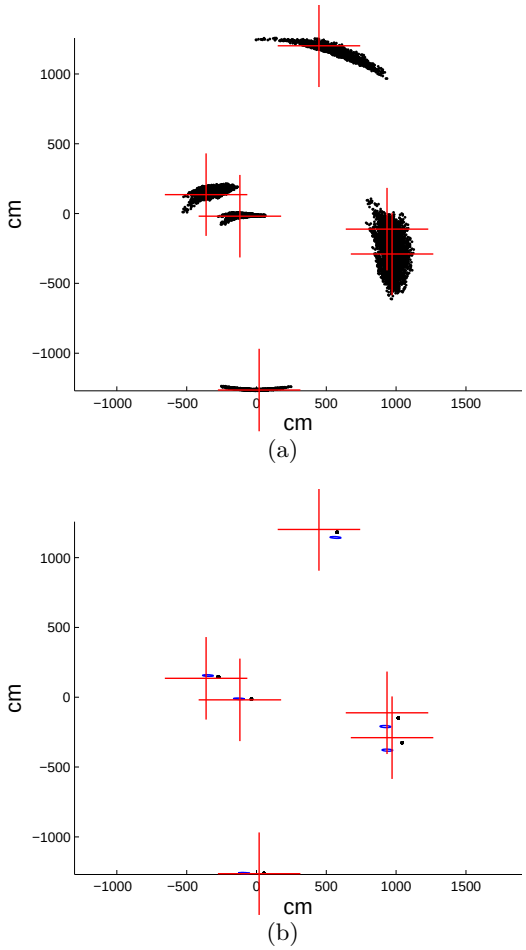


FIGURE 7.11. Results for data obtained from the simulator with moderate noise for a robot path of 50 steps through a 6 sensor network for the algorithms: a.) MCMC b.) RBPF ($K = 20000$), (black particles) and EKF (blue uncertainty ellipses). The red crosses indicate the actual sensor positions.

of the maximum likelihood location for the sensors, their uncertainty estimates are extremely poor in comparison to the MCMC approach which, we argue, is portraying the underlying distribution with reasonable accuracy. The EKF is over-confident and the RBPF suffers severely from the particle-depletion problem and shows a lack of diversity. In general, in our simulations, we observed that both the EKF and the RBPF suggest distributions that diverge from that suggested by the MCMC algorithm as the path length of the robot increases and this divergence is usually towards over-confidence. Further insight can be gained by considering the likelihood of the final configurations obtained in this example. Given an adequate burn-in time, the log likelihoods of the final configurations obtained by

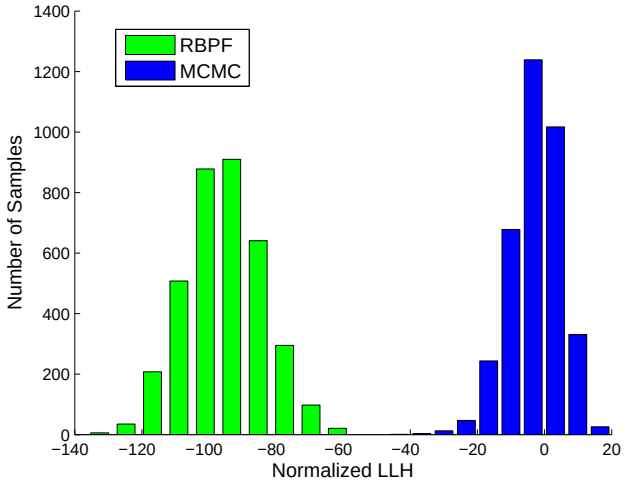


FIGURE 7.12. Histogram comparing the relative log likelihoods of the final configuration samples obtained from the MCMC and RBPF ($k = 20000$) techniques for the simulation result shown in Figure 7.11. The likelihoods were normalized such that ground truth had a log likelihood of zero.

MCMC approach the same order of magnitude as ground truth, and typically are much higher in likelihood than results obtained from the RBPF, even with a large number of particles; *e.g.* see Figure 7.12.

To use the MCMC technique to provide a maximum likelihood estimate (MLE) for the sensor positions, one can consider the sample with maximum likelihood (ML) or the mean of the samples obtained. In our simulations, we observed that the mean of the cloud consistently gave good results, although an estimate obtained from the RBPF on the same problem instance generally had similar accuracy; *e.g.* see Figure 7.13. The performance of the maximum likelihood MCMC sample had a much higher variance, and while it was occasionally extremely accurate as an estimator, it was overall less consistent.

5.3. Convergence Issues. Figure 7.14 demonstrates the improved convergence properties of the odometry-based proposal scheme used in conjunction with RB over single-component Metropolis-Hastings. Presumably the application of RB removes some of the correlation between individual components of the state space and allows much larger jumps than would otherwise be possible. Supporting this idea is the observation that the automatically tuned sigma values for individual components, (*i.e.* the robot poses s^t), are in general larger when RB is employed than when it is not for the same measurement data.

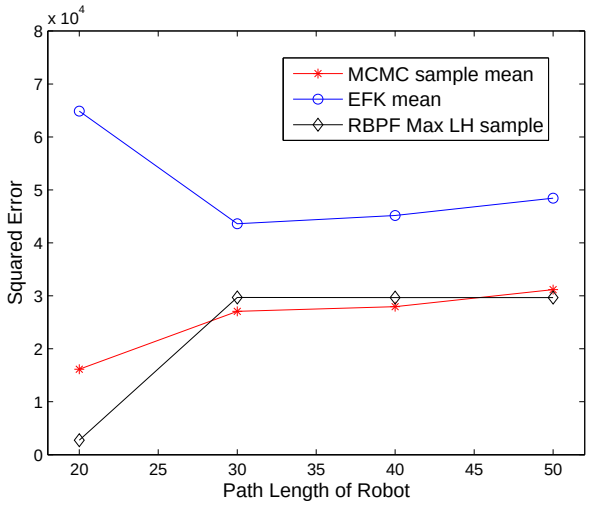


FIGURE 7.13. Squared error of MLE of sensor positions as a function of robot path length through a 6 senor network; (the same simulation presented in Figure 7.11). The result obtained from the mean of the RBPF samples was similar, but poorer, than the RBPF maximum likelihood sample in this experiment and not presented for improved clarity.

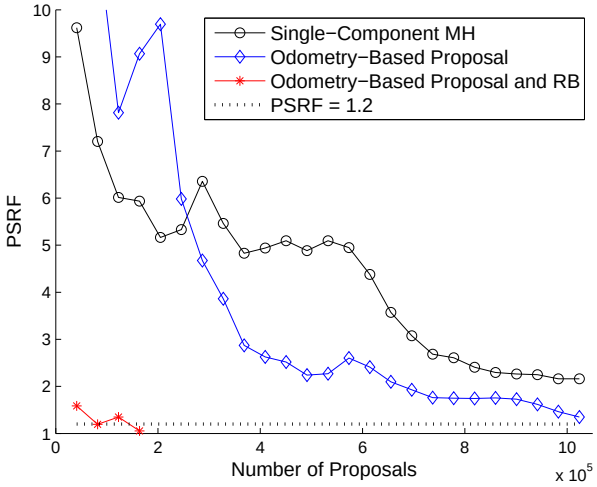


FIGURE 7.14. Example of PSRF as a function of computational effort for different variants of the MCMC global inference algorithm. Data are presented based on simulation data gathered from a 4 sensor, 12 path length scenario. The PSRF is calculated given 4 restarts of each algorithm.

5.4. Analysis of EKF-mean Correction. In addition to using the MCMC approach for stand-alone probabilistic self-localization, we also assessed its use as a complement to the faster EKF approach as described in Section 4. We found that under moderate and high noise conditions, on average, the application of EKF-mean correction *via* the MCMC algorithm improved the squared error of the EKF mean. For example, Figure 7.15 shows the results of the hybrid MCMC/EFK approach on small networks under different noise conditions. For these small path lengths, the application of EKF-mean correction did not make a large difference under low noise conditions. Considerable reductions in error were observed, however, for moderate and high noise conditions.

Figure 7.16 demonstrates the ability of the hybrid EKF-MCMC approach to reduce the error inherent in the linear filtering approach in larger scenarios. For the long path lengths, and the moderate noise conditions considered in this experiment, the EKF estimate eventually accumulates large errors due to linearization. However, the EKF is able to maintain a more accurate estimate of the sensor positions by applying the MCMC algorithm at regular intervals (every 10 steps) in the robot's path. As shown in the figure, the estimation error grows much more slowly when this type of correction is applied.

In general, we observed that if the EKF is performing well, then the configuration it suggests is often already in a high likelihood location. In this case the MCMC algorithm does not necessarily move it towards the ground truth position, but just alters it slightly in order to locally improve its probability. Where the correction provided the most help is when the EKF was performing poorly; *i.e.* the correction prevents the EKF from diverging.

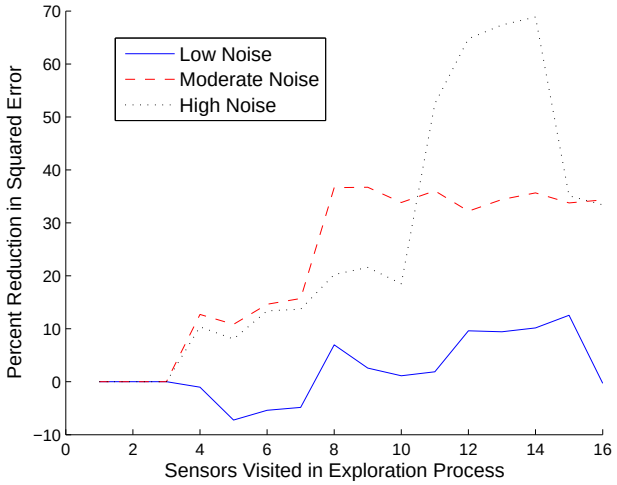


FIGURE 7.15. Percentage improvement in the squared error of the EKF mean using MCMC correction for different noise levels. The results obtained from 100 trials on 4 node sensor networks.

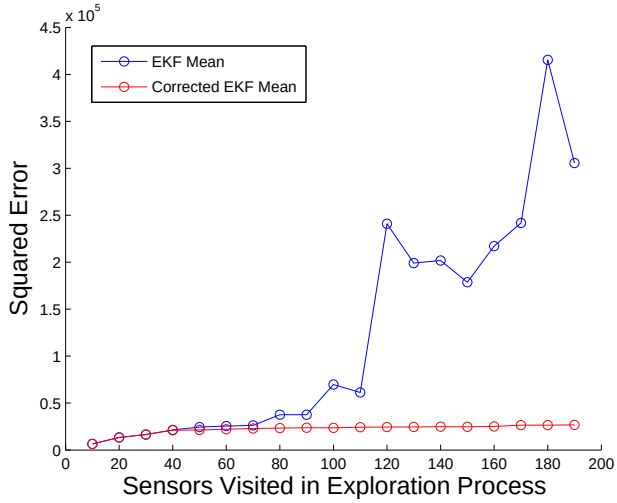


FIGURE 7.16. Mean error in estimation of final sensor positions for EKF estimation alone and the hybrid EKF-MCMC approach. The results obtained from 100 trials on 10 node sensor networks with moderate levels of noise.

6. Experimental Data

We applied our MCMC approach to localization on mapping data gathered from a deployed camera sensor network and a single mobile robot (see Figure 7.17). The target sensor network is located in an office environment, and consists of seven networked cameras. The robot traveled through a pair of loops connected by a long straight hallway with length

(a) (b)

FIGURE 7.17. Pictures of the components of the camera sensor network used in the experimental results of this chapter.

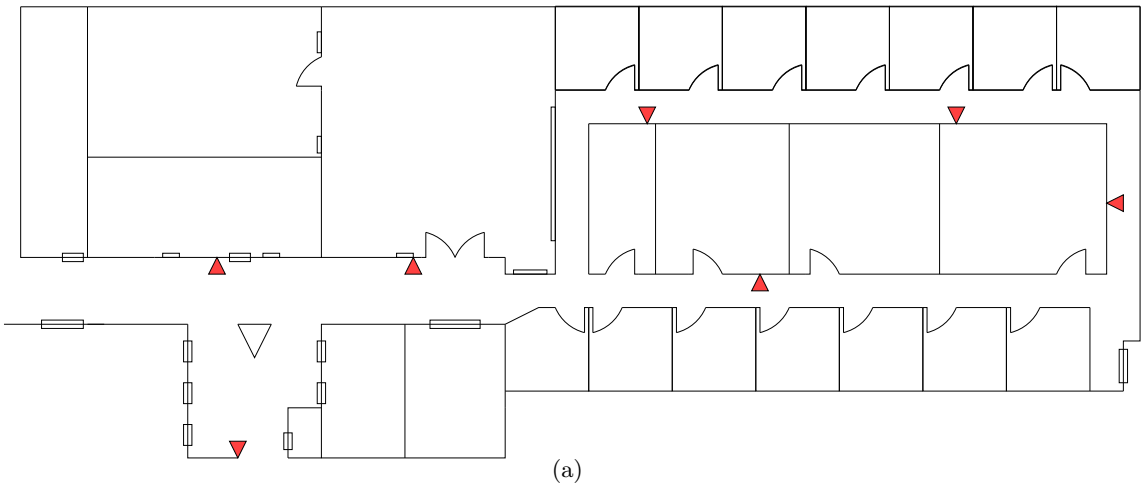
approximately 50 m as shown in Figure 7.18(a). A Nomadics Scout robot mounted with a target with six recognizable patterns was used to perform a calibration procedure and obtain position measurements using a method described in Rekleitis *et al.* [**105**].

The system for conducting these experiments was implemented by Dave Meger while conducting research towards his Master's Thesis at McGill University under the combined supervision of Dr. Gregory Dudek and Dr. Ioannis Rekleitis (see [**82**] for futher details). Some of the infrastructure for the network was based on the experimental setup we describe in this text in Section 5 of Chapter 3. The data was gathered by Meger and Rekleitis and has been used in previously published results [**83**] [**106**]. Meger and Rekleitis have collaborated with us in applying the techniques presented in this chapter to this same data set.

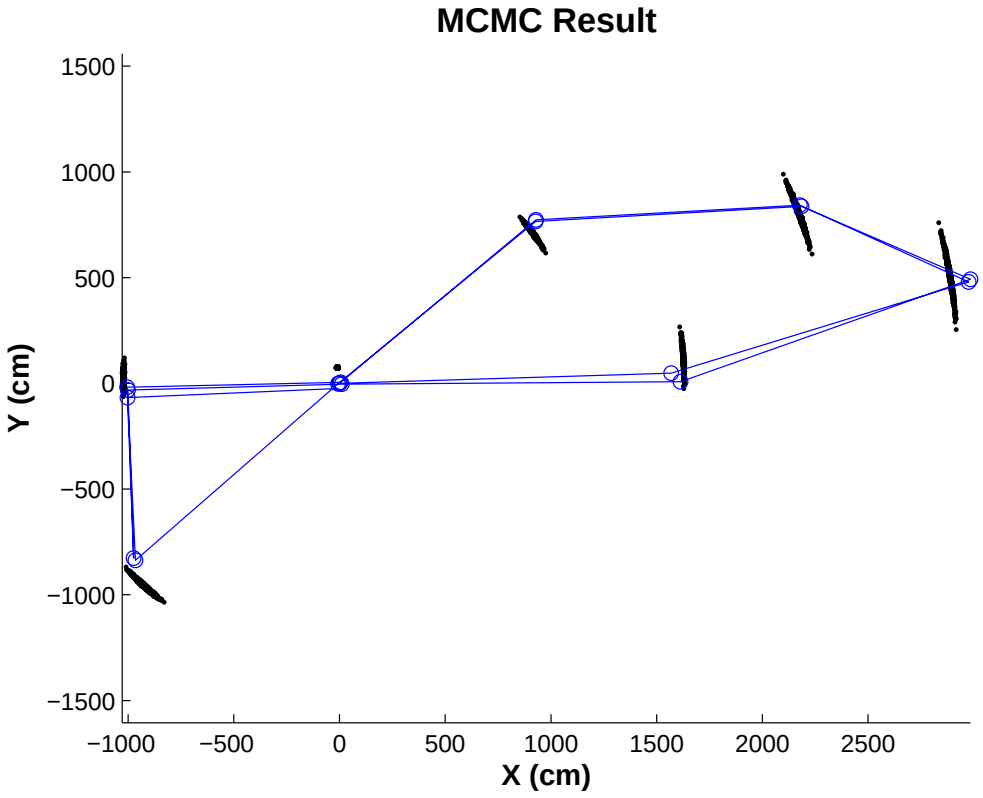
Due to the size of the environment, and lack of line-of-sight between sensor positions, ground truth data could not be collected for this experiment, but there are several measures which can be used for qualitative assessment of estimation accuracy on this data. First, care was taken to return the robot to within a few centimetres of its initial position at the end of the run, which implies the first and last robot positions should agree very closely in any accurate estimate. Also, sensor location accuracy can be estimated visually, by comparing to the sensor locations recorded on Figure 7.18(a).

When applied to the data gathered during these experiments our algorithm converged in under two hours[2] on a P4, 3.2 GHz machine with 1 GB of RAM. Figure 7.18 shows the

[2]This duration was calculated *per* restart of the algorithm. In this problem instance four restarts were employed for the purpose of applying the convergence analysis.



(a)



(b)

FIGURE 7.18. (a) Approximate floor plan showing sensor locations during the experiment. (b) The estimated robot path (based on a MLE estimate) and distributions of the sensor positions resulting from our approach.

results obtained. This figure includes an approximation to the robot path as a sequence of linear motions based on the ML configuration obtained. Although this MCMC approach is relatively computationally expensive, sensor network calibration can be considered a one-time expense and accurate location and uncertainty results can be utilized for higher level planning and reasoning purposes throughout the lifetime of the system. The final robot positions can be observed to lie within a meter from the initial position, which is a strong indicator of map accuracy, as the path length is over 200 m in total.

7. Discussion

This chapter presents an approach to the Probabilistic Sensor Localization Problem that exploits a combination of emplaced sensing nodes and a moving robot. Our approach is capable of providing a representation of the underlying PDF with much greater efficiency and accuracy than other currently available options, and furthermore, provides a principled stopping mechanism for determining when enough computational effort has been expended. This work also demonstrates the limitations of current filtering-based techniques at accurately representing uncertainty in sensor network localization.

Although aimed at the sensor network domain, this technique can also be employed in any SLAM scenario involving a robot and landmarks with known correspondences. Even for larger scale problems in which the computation required becomes an issue for 'on-platform' implementation, this approach can be run as an off-line batch process or as a corrective mechanism for faster techniques. One value might be to provide a measuring-stick for tuning the performance of faster algorithms, particularly where their uncertainty measurements are concerned. Additionally, the nature of the MCMC algorithm makes it suitable for extensions to a multi-robot scenario where multiple mobile robots explore the same sensor network collecting information from their interaction with the sensor nodes and with each other.

The work also entails some open problems. One open issue is how the quantity of data collected affects the accuracy with which the PDF can be represented. It appears that one possible optimization step could be to omit some of the constraints in order to quickly arrive at a distribution. Data could then be incrementally included to improve accuracy. This

could yield an 'anytime'-style algorithm that could quickly produce usable results which become more refined with additional computation.

Other issues relate to the use of the MCMC algorithm as a corrective technique for faster approaches. In this chapter, we have only considered correcting the mean of the EKF. In principle, however, the uncertainty represented by the EKF covariance could also be corrected. Running the MCMC technique to convergence at some point in the robot's path would make correcting the EKF covariance at that time-step trivial, but would both make the EKF redundant and defeat the speed advantage of incorporating the technique. A more interesting approach would be to correct the EKF covariance based only on the final high probability samples obtained by the MCMC as in burns in. For example, a simple approach might be to increase, if necessary, the appropriate diagonal components of the EKF covariance until the likelihood of the final MCMC obtained sample is within some threshold. A related direction would be to alter a standard RBPF such as the one implemented in this work to include a global correction step utilizing our MCMC approach. Incorporating an MCMC step in a RPBF has been considered before [**31**] and should improve particle diversity and ultimately bring the distribution suggested closer to the target distribution. It would be interesting to see how much computation would be needed to obtain results with reasonable uncertainty estimates.

CHAPTER 8

Conclusion and Future Work

1. Summary

In this thesis we have considered the problem of inferring a representation of the environment given limited sensory data. We presented two main sub-problems: one interested in inferring the topology of the environment; and one interested in inferring metric relationships in the environment. In this research, we have considered these two issues as separate problems, however, topological and metric information are highly complementary and can allow an intelligent system to build up a functional representation of its surroundings which can be used as input to higher level processing tasks. In our research into these two topics, we have considered different aspects of the respective problems and investigated them through simulations and experiments. For the most part we consider statistical methods that employ stochastic sampling techniques to provide approximate solutions to problems for which computing the optimal or exact solution is intractable.

The first sub-problem is made up of Chapters 3, 4 and 5. In Chapter 3 we presented the bulk of our work on topology inference and assessed the approach with experiments, while in Chapters 4 and 5 we presented some interesting related investigations but relied solely on simulations for verification purposes. The second sub-problem is made up of Chapters 6 and 7. Here the bulk of the work on metric inference along with experimental validation is presented in Chapter 7. In the preceding Chapter 6 we introduce the concept of self-localization and present some closely related work based on numerical simulations. A number of different statistical and inference techniques and technologies have been explored

in the process of our investigations in both sub-problems. The focus has been mainly on issues related to sensor networks, but mobile robots have been incorporated into our investigations in Chapters 5 and 7.

We have shown that with various degrees of success, depending on the quality of the sensing information available, an intelligent system can infer a great deal of information regarding its placement in the environment and potentially its uncertainty regarding this placement. Our research could potentially allow a system to recover complementary metric and topological information regarding the immediate environment. In Chapters 3 and 4 we have shown that non-discriminating observations of activity in the monitored region can be sufficient to infer network connectivity information while in Chapter 5 we show that a topological map of a region can be inferred by a mobile robot with only rudimentary sensing abilities. Chapters 6 and 7 demonstrate the ability of a sensor network to infer the relative positions of its components through noisy inter-sensor pose estimates. In all these cases we have worked with observational data of the quality that could be obtained from typical low-cost sensors.

The work addressed in this thesis demonstrates the potential for the self-calibration of sensor networks and other intelligent systems. We have presented an investigation into several research areas related to various aspects of this self-calibration problem. Simulations and experimental results suggest that our techniques compare favorably to related approaches and show real world potential.

2. Future Work

The potential for future work on various portions of this thesis bas been considered throughout the text, and the reader is directed towards the individual chapters for these discussions. In this section, we collect together broader future directions suggested by the research.

Our current work in sensor network self-calibration attempts to address several aspects of inferring environmental parameters given imperfect observational data, however, in order to make these approaches practical we need to address *performance* and *scalability*. Some of the algorithms discussed in this paper employ heavy probabilistic techniques which limit their application to networks of less than a hundred nodes; a number far less

than that envisioned in future ad-hoc deployments. Therefore, improvements in terms of performance or scalability to either topological or metric inference are obvious possibilities for future research. For example, the computationally sophisticated MCEM approach to topology inference presented in Chapter 3 could initialized with one of the faster heuristic techniques for topology inference presented in Chapters 4 or 5. Both of these heuristic based techniques are less robust to realistic levels of error, but could potentially initialize the probabilistic approach with a relatively high likelihood configuration which could lead to faster convergence. In the self-localization work presented in Chapter 6 and in Chapter 7, obtaining an accurate PDF of the pose of the network requires running the MCMC chain to convergence which becomes impractical with larger networks. It is possible that one could achieve improvement in run time by partitioning the underlying constraint graph into multiple regions of high edge density, solve each region separately and then combine the results in a principled manner.

The thesis has focused on two classes of techniques which can be used to recover a representation of the surrounding environment. The information each inference approach can learn regarding the environment is complementary to the other. Each approach, however, requires a different set of inputs and functions independently of the other approach. An intriguing possibility is incorporating the two types of inference together in some manner that is capable of exploiting the relationship between a topological and metric viewpoint. Such a hybrid approach could potentially function more efficiently in some situations than running two separate algorithms independently. It seems clear that knowledge regarding the metric distances between sensors could be exploited to improve the topology inference algorithm and *visa versa*. For example, assumptions about maximum agent velocities could help restrict or make less likely certain trajectory samples. These clues should speed up convergence of the topology algorithm and improve its accuracy by avoiding inconsistent network configurations. Likewise, the connectivity information could help suggest sensors that are closely or distantly located based on delay times. It would be interesting to simultaneously solve for the topology and relative metric locations of a sensor network based on both observational data and inter-sensor distance estimates.

A potentially interesting and closely related research direction would be to slightly alter our problem formulation and allow our sensors to move. Instead of exploiting existing

motion in the environment the network of sensors could exploit observations based on their own motion in order to gain an understanding of the world. For example, one problem formulation could allow the network of sensors, which can now be considered a fleet of mobile robots, to observe each other only when they pass by in close proximity. Additionally, they could observe some relatively uninformative information regarding landmarks in the environment; *e.g.* a non-unique signature. If we assume the odometry error of a single robot is limited to range information only and its motion through the environment is essentially random, then many of the approaches applied in the two self-configuration techniques previously described can be leveraged to some degree.

3. Final Thoughts

In this thesis we have presented a number of methods for inferring a representation of the environment given limited sensory data. A key application of this type of research is in the area of sensor network self-calibration. There is a need for some level of automation in the calibration process of systems which are made up of large numbers of sensors. The realization of practical and economic deployments of sensor networks for purposes such as climate and emission monitoring will be hampered by complex installation processes that involve many laborious manual steps. Our research is ultimately aimed at alleviating some of the configuration complexities involved in deploying large, multi-component systems. This type of problem will almost certainly grow in importance both for its theoretical and practical value as distributed sensing becomes more prevalent.

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. A. Cayirci, *A survey on sensor networks*, IEEE Communications Magazine **40** (2002), no. 8, 102–114.

[2] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan, *An introduction to MCMC for machine learning*, Machine Learning **50** (2003), 5–43.

[3] T. Arici and Y. Altunbasak, *Adaptive sensing for environment monitoring using wireless sensor networks*, in Proc. IEEE Wireless Communications and Networking Conference (WCNC) (Atlanta, GA), March 2004.

[4] Y. Bar-Shalom (ed.), *Multitarget multisensor tracking: advanced applications*, vol. II, Artech House, 1992.

[5] C. R. Barnes, J. R. Delaney, B. M. Howe, and N. Penrose, *Neptune: A regional cabled observatory in the northeast pacific*, White paper for Ocean Research Interactive Observatory Networks (ORION) meeting, January 2004.

[6] M.A. Batalin, G.S. Sukhatme, and M. Hattig, *Mobile robot navigation using a sensor network*, IEEE International Conference on Robotics and Automation (New Orleans, Louisiana), vol. 1, April 26-May 1 2004, pp. 636 – 641.

[7] L. E. Baum and T. Petrie, *Statistical inference for probabilistic functions of finite state markov chains*, Annals of Mathematical Statistics **37** (1966), 1554–1563.

[8] I. Borg and P. Groenen, *Modern multidimensional scaling: theory and applications*, Springer, New York, 1997.

[9] A. Boukerche, H.A.B.F. Oliveira, E. F. Nakamura, and A.A.F. Loureiro, *Secure localization algorithms for wireless sensor networks*, **46** (2008), no. 4, 96–101.

[10] J. E. Boyd and J. Meloche, *Evaluation of statistical and multiple-hypothesis tracking for video traffic surveillance*, Machine Vision and Applications **13** (2003), 344–351.

[11] M. Bramberger, A. Doblander, A. Maier, B. Rinner, and H. Schwabach, *Distributed embedded smart cameras for surveillance applications*, Computer **39** (2006), no. 2, 68–75.

[12] S. P. Brooks and A. Gelman, *General methods for monitoring convergence of iterative simulations*, Journal of Computational and Graphical Statistics **7** (1998), 434–455.

[13] N. Bulusu, D. Estrin, L. Girod, and J. Heidemann, *Scalable coordination for wireless sensor networks: self-configuring localization systems*, Sixth International Symposium on Communication Theory and Applications (ISCTA-01) (Ambleside, Lake District, UK), July 2001, pp. 1–6.

[14] N. Bulusu, J. Heidemann, and D. Estrin, *Gps-less low cost outdoor localization for very small devices*, IEEE Personal Communications Magazine **7** (2000), no. 5, 28–34.

[15] W. Burgard, D. Fox, H. Jans, C. Matenar, and S. Thrun, *Sonar-based mapping with mobile robots using EM*, Proc. 16th International Conf. on Machine Learning, Morgan Kaufmann, San Francisco, CA, 1999, pp. 67–76.

[16] S. Capkun, M. Hamdi, and J. Hubaux, *GPS-free positioning in mobile ad-hoc networks*, Proc. 34th Annual Hawaii International Conference on System Sciences, Jan. 2001, pp. 10–.

[17] G. Casella and C. P. Robert, *Rao-blackwellisation of sampling schemes*, Biometrika **83** (1996), no. 1, 81–94.

[18] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao, *Habitat monitoring: Application driver for wireless communications technology*, 2001 ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean, April 2001.

[19] H. Choset and K. Nagatani, *Topological simultaneous localization and mapping (SLAM): toward exact localization without explicit localization*, IEEE Transactions on Robotics and Automation **17** (2001), no. 2, 125 – 137.

[20] M. J. Coates, *Distributed particle filtering for sensor networks*, Int. Symp. Information Processing in Sensor Networks (Berkeley, California), April 2004.

[21] N. Correal and N. Patwari, *Wireless sensor networks: Challenges and opportunities*, MPRG/Virgina Tech Wireless Symposium, 2001.

[22] M. Couture, M. Barbeau, P. Bose, and E. Kranakis, *Incremental construction of k-dominating sets in wireless sensor networks*, (2008).

[23] K. Dantu and G. S. Sukhatme, *Rethinking data-fusion based services in tiered sensor networks*, The Third IEEE Workshop on Embedded Networked Sensors, 2006.

[24] F. Dellaert and M. Kaess, *Square Root SAM: Simultaneous location and mapping via square root information smoothing*, International Journal of Robotics Research **25** (2006), no. 12, 1181–1203.

[25] F. Dellaert, S. Seitz, C. Thorpe, and S. Thrun, *EM, MCMC, and chain flipping for structure from motion with unknown correspondence*, Machine Learning, special issue on Markov chain Monte Carlo method **5** (2003), 45–71.

[26] B. Delyon, M. Lavielle, and E. Moulines, *Convergence of a stochastic approximation version of the em algorithm*, Annals of Statistics (1999), no. 27, 94–128.

[27] A. Dempster, N. Laird, and D. Rubin, *Maximum likelihood from incomplete data via the EM algorithm*, Journal of the Royal Statistical Society **39** (1977), 1–38.

[28] X. Deng, E. Milios, and A. Mirzaian, *Robot map verification of a graph world*, Journal of Combinatorial Optimization **5** (2001), no. 4, 383–395.

[29] X. Deng and A. Mirzaian, *Competitive robot mapping with homogeneous markers*, IEEE transactions on Robotics and Automation **12** (1996), no. 4, 532–542.

[30] J. Djugash, S. Singh, G. Kantor, and W. Zhang, *Range-only SLAM for robots operating cooperatively with sensor networks*, Proc. of the International Conference on Robotics and Automation, May 2006, pp. 2078–2084.

[31] A. Doucet, N. de Freitas, K. Murphy, and S. Russell, *Rao-blackwellised particle filtering for dynamic bayesian networks*, In Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (Stanford, California), Morgan Kaufmann, 2000.

[32] G. Dudek, *Environmental representation using multiple abstraction levels*, Proceedings of the IEEE **84** (1996), no. 11, 1684–1704.

[33] G. Dudek, P. Freedman, and S. Hadjres, *Mapping in unknown graph-like worlds*, Journal of Robotic Systems **13** (1996), no. 8, 539–559.

[34] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, *Using local information in a non-local way for mapping graph-like worlds*, International Joint Conference on Artificial Intelligence (Chambery, France), August 1993.

[35] ———, *Map validation and robot self-location in a graph-like world*, Robotics and Autonomous Systems **22** (1997), no. 2, 159–178.

[36] G. Dudek and D. Marinakis, *Topological mapping with weak sensory data*, AAAI National Conference on Artificial Intelligence (Vancouver, Canada), July 2007, pp. 1083–1088.

[37] T.J. Ellis, D. Makris, and J. Black, *Learning a multicamera topology*, Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (Nice, France), October 2003, pp. 165–171.

[38] D. Estrin, R. Govindan, J. S. Heidemann, and S. Kumar, *Next century challenges: Scalable coordination in sensor networks*, Mobile Computing and Networking, 1999, pp. 263–270.

[39] R. B. Fisher, *Self-organization of randomly placed sensors*, Eur. Conf. on Computer Vision (Copenhagen), May 2002, pp. 146–160.

[40] G. S. Fishman, *Monte carlo concepts, algorithms, and applications*, Springer-Verlag, New York, 1996.

[41] D. Gay, P. Levis, and D. Culler, *Software design patterns for tinyos*, Proc. ACM SIGPLAN/SIGBED 2005 Conference on Languages, Complilers, and Tools for Embedded Systems (LCTES'05) (Chicago), June 2005, pp. 40–49.

[42] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler, *The nesc language: A holistic approach to networked embedded systems*, ACM SIGPLAN Conference on Programming Language Design and Implementation (San Diego, CA.), June 2003, pp. 1–11.

[43] A. Gelman and D. B. Rubin, *Inference from iterative simulation using multiple sequences (with discussion)*, Statistical Science **7** (1992), 457–511.

[44] P. Giguere, G. Dudek, C. Prahacs, and S. Saunderson, *Environment identification for a running robot using inerial and actuator cues*, Proceedings of Robotics: Science and Systems (Philadelphia, USA), August 2006.

[45] W. Gilks, S. Richardson, and D.J. Spiegelhalter, *Markov chain monte carlo in practice*, Chapman and Hall, 1996.

[46] J.-S. Gutmann and K. Konolige, *Incremental mapping of large cyclic environments*, International Symposium on Computational Intelligence in Robotics and Automation (CIRA'99), (Monterey, CA), November 1999.

[47] K. Z. Haigh, W. Foslien, and V. Guralnik, *Visual query language: Finding patterns in and relationships among time series data*, Seventh Workshop on Mining Scientific and Engineering Datasets (Lake Buena Vista, FL), April 2004.

[48] F. Hamze and N. de Freitas, *From fields to trees: On blocked and collapsed mcmc algorithms for undirected probabilistic graphical models*, Proc. Uncertainty in Artificial Intelligence, 2004.

[49] W. Hastings, *Monte carlo sampling methods using markov chains and their applications*, Biometrika **57** (1970), 97–109.

[50] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister, *System architecture directions for networked sensors*, Architectural Support for Programming Languages and Operating Systems, 2000, pp. 93–104.

[51] T. Huang and S. J. Russell, *Object identification in a bayesian context*, IJCAI, 1997, pp. 1276–1283.

[52] ———, *Object identification: A bayesian analysis with application to traffic surveillance*, Artificial Intelligence **103** (1998), no. 1-2, 77–93.

[53] A. T. Ihler, J. W. Fisher III, R. L. Moses, and A. S. Willsky, *Nonparametric belief propagation for self-calibration in sensor networks*, IEEE Journal of Selected Areas in Communication **23** (2005), no. 4, 809–819.

[54] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, *Directed diffusion for wireless sensor networking*, IEEE/ACM Transactions on Networking, vol. 11, 2003, pp. 2–16.

[55] O. Javed, Z. Rasheed, K. Shafique, and M. Shan, *Tracking across multiple cameras with disjoint views*, The Ninth IEEE International Conference on Computer Vision (Nice, France), vol. 2, Oct. 2003, pp. 952–957.

[56] M. Jerrum and A. Sinclair, *Markov chain monte carlo method: an approach to approximate counting and integration*, Approximation Algorithms for NP-hard Problems (D.S.Hochbaum, ed.), PWS Publishing, Boston, 1996.

[57] J. Jonasson and O. Schramn, *On the cover time of planar graphs*, Electronic Communications in Probability, 2000, pp. 85–90.

[58] R. M. Karp, *Reducibility among combinatorial problems*, Complexity of Computer Computations (R. E. Miller and J. W. Thatcher, eds.), New York:Plenum, 1972, pp. 85–103.

[59] M. Koucky, *Universal traversal sequences with backtracking*, In Proc. of 16th Annual IEEE Conference on Computational Complexity, 2001, pp. 21–27.

[60] F. Kuhn, T. Moscibroda, and R. Wattenhofer, *Initializing newly deployed ad hoc and sensor networks*, in Proceedings of 10th Annual International Conference on Mobile Computing and Networking (MOBICOM), 2004, pp. 260–274.

[61] B. Kuipers and P. Beeson, *Bootstrap learning for place recognition*, AAAI National Conference on Artificial Intelligence (Edmonton, Canada), 2002.

[62] 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** (1991), 47–63.

[63] P. Levis, S. Madden, D. Gay, J. Polastre, R. Szewczyk, A. Woo, E. Brewer, and D. Culler, *The emergence of networking abstractions and techniques in tinyos*, the

First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI 2004), 2004, pp. 1–14.

[64] P. Levis, N. Patel, D. Culler, and S. Shenker, *Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks*, USENIX/ACM Symposium on Network Systems Design and Implementation(NSDI 2004), 2004.

[65] J.J. Liu, J. Liu, J. Reich, P. Cheung, and F. Zhao, *Distributed group management for track initiation and maintenance in target localization applications*, Proceedings of 2nd International Workshop on Information Processing in Sensor Networks (IPSN) (Palo Alto, CA.), April 2003, pp. 113–128.

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

[67] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, *Wireless sensor networks for habitat monitoring*, ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02) (Atlanta, GA), September 2002.

[68] D. Makris, T.J. Ellis, and J. Black, *Bridging the gaps between cameras*, IEEE Conference on Computer Vision and Pattern Recognition CVPR 2004 (Washington DC), vol. 2, June 2004, pp. 205–210.

[69] Y. Maon, B. Schieber, and U. Vishkin, *Parallel ear decomposition search (EDS) and st-numbering in graphs*, (1986), 277–298.

[70] D. Marinakis and G. Dudek, *Sensor network topology inference*, Neural Information Processing Systems Workshop: Intelligence Beyond the Desktop (Whistler, Canada), Dec 2005.

[71] ———, *Topology inference for a vision-based sensor network*, In Proc. of Canadian Conference on Computer and Robot Vision (Victoria, Canada), May 2005, pp. 121–128.

[72] ———, *A practical algorithm for network topology inference*, IEEE Intl. Conf. on Robotics and Automation (Orlando, Florida), May 2006, pp. 3108–3115.

[73] ———, *Probabilistic self-localization for sensor networks*, AAAI National Conference on Artificial Intelligence (Boston, Massachusetts), July 2006, pp. 976–981.

[74] ———, *Self-calibration of a vision-based sensor network*, Image and Vision Computing (In Press) (2006).

[75] ———, *Topological mapping through distributed, passive sensors*, International Joint Conference on Artificial Intelligence (Hyderabad, India), Jan. 2007, pp. 2147–2152.

[76] ———, *Occam's razor applied to network topology inference*, IEEE Transactions on Robotics **24** (2008), no. 2, 293 – 306.

[77] D. Marinakis, G. Dudek, and D. Fleet, *Learning sensor network topology through monte carlo expectation maximization*, IEEE Intl. Conf. on Robotics and Automation (Barcelona, Spain), April 2005, pp. 4581–4587.

[78] D. Marinakis, P. Giguere, and G. Dudek, *Learning network topology from simple sensor data*, In Proceedings of the Canadian Conference on Artificial Intelligence (Montreal, Canada), May 2007, pp. 417–428.

[79] D. Marinakis, D. Meger, I. Rekleitis, and G. Dudek, *Hybrid inference for sensor network localization using a mobile robot*, AAAI National Conference on Artificial Intelligence (Vancouver, Canada), July 2007, pp. 1089–1094.

[80] A. Martinelli, N. Tomatis, and R. Siegwart, *Some results on SLAM and the closing the loop problem*, IEEE/RSJ Intenational Conference on Intelligent Robots and Systems (Edmonton, Canada), 2005.

[81] P. Maybeck, *Stochastic models, estimation ond control*, vol. 1, Academic, New York, 1979.

[82] D. Meger, *Planning, localization, and mapping for a mobile robot in a camera network*, Master's thesis, McGill University, 2007.

[83] D. Meger, I. Rekleitis, and G. Dudek, *Autonomous mobile robot mapping of a camera sensor network*, The 8th International Symposium on Distributed Autonomous Robotic Systems (DARS) (Minneapolis, Minnesota), July 2006, pp. 155–164.

[84] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller, *Equation of state calculation by fast computing machines*, Journal of Chemical Physics **21** (1953), 1087–1092.

[85] T. M. Michell, *Machine learning*, McGraw-Hill, Boston, 1997.

[86] M. Montemerlo and S. Thrun, *Simultaneous localization and mapping with unknown data association using fastslam*, IEEE International Conference on Robotics and Automation (Taipei, Taiwan), vol. 2, 14-19 Sept. 2003, pp. 1985 – 1991.

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

[88] ———, *FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges*, Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI) (Acapulco, Mexico), IJCAI, 2003.

[89] D. Moore, J. Leonard, D. Rus, and S. Teller, *Robust distributed network localization with noisy range measurements*, Proc. of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys '04) (Baltimore), November 2004.

[90] K. Murphy, *Bayesian map learning in dynamic environments*, In Proceedings of Advances in Neural Information Processing Systems (Denver, Colorado), MIT Press, 1999, pp. 1015–1021.

[91] Paul Newman and Kin Ho, *SLAM-loop closing with visually salient features*, IEEE Intl. Conf. on Robotics and Automation (Barcelona, Spain), April 2005.

[92] D. Niculescu and B. Nath, *Ad hoc positioning system (APS) using AoA*, Proc. of Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (San Francisco, CA.), vol. 3, 2003, pp. 1734–1743.

[93] S. Oh, S. Russell, and S. Sastry, *Markov chain monte carlo data association for general multiple-target tracking problems*, Proc. of the 43rd IEEE International Conference on Decision and Control (CDC) (Paradise Island,Bahamas), vol. 1, Dec 2004, pp. 735–742.

[94] M. A. Paskin, C. E. Guestrin, and J. McFadden, *A robust architecture for inference in sensor networks*, In Proc. of the Fourth International Symposium on Information Processing in Sensor Networks 2005 (IPSN-05), April 2005, pp. 55–62.

[95] H. Pasula, S. Russell, M. Ostland, and Y. Ritov, *Tracking many objects with many sensors*, IJCAI-99 (Stockholm), 1999, pp. 1160–1171.

[96] N. Patwari, A.O. Hero, M. Perkins, N.S. Correal, and R.J. O'Dea, *Relative location estimation in wireless sensor networks*, IEEE Transactions on Signal Processing, 8, vol. 51, Aug 2003, pp. 2137–2148.

[97] R. Peng and M. L. Sichitiu, *Probabilistic localization for outdoor wireless sensor networks*, **11** (2007), no. 1, 53–64.

[98] F.P. Preparata and M.I. Shamos, *Computational geometry: An introduction*, Springer-Verlag, New York, NY, 1985.

[99] M. G. Rabbat, M.A.T. Figueiredo, and R.D. Nowak, *Network inference from co-occurrences*, IEEE Transactions on Information Theory **54** (2008), no. 9, 4053–4068.

[100] A. Rahimi, B. Dunagan, and T. Darrell, *Simultaneous calibration and tracking with a network of non-overlapping sensors*, CVPR 2004, vol. 1, June 2004, pp. 187–194.

[101] A. Ranganathan and F. Dellaert, *Inference in the space of topological maps: an MCMC-based approach*, Intenational Conference on Intelligent Robots and Systems (Sendai, Japan), 2004, pp. 1518–1523.

[102] ———, *Data driven MCMC for appearance-based topological mapping*, Proceedings of Robotics: Science and Systems (Cambridge, USA), June 2005.

[103] ———, *A rao-blackwellized particle filter for topological mapping*, Proc. of the International Conference on Robotics and Automation (Orlando, USA), May 2006, pp. 810–817.

[104] C. Rasmussen and G. Hager, *Probabilistic data association methods for tracking multiple and compound visual objects*, IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 23, 2001, pp. 560–576.

[105] I. Rekleitis, D. Meger, and G. Dudek, *Simultaneous planning localization, and mapping in a camera sensor network*, Robotics and Autonomous Systems (RAS) Journal, special issue on Planning and Uncertainty in Robotics **54** (2006), no. 11, 921–932.

[106] ———, *Simultaneous planning, localization, and mapping in a camera sensor network*, Robotics and Autonomous Systems (2006), (in print).

[107] I. M. Rekleitis, V. Dujmović, and G. Dudek, *Efficient topological exploration*, Proceedings of International Conference in Robotics and Automation (Detroit, USA), May 1999, pp. 676–681.

[108] ———, *Efficient topological exploration*, Proceedings of International Conference in Robotics and Automation (Detroit, USA), May 1999, pp. 676–681.

[109] M. Rosencrantz, G. Gordon, and S. Thrun, *Locating moving entities in indoor environments with teams of mobile robots*, Second international joint conference on Autonomous agents and multiagent systems (Melbourne, Australia), 2003, pp. 233–240.

[110] P. L. Sala, R. Sim, A. Shokoufadeh, and S. J. Dickinson, *Landmark selection for vision-based navigation*, IEEE Transactions on Robotics and Automation (2005).

[111] F. Savelli and B. Kuipers, *Loop-closing and planarity in topological map-building*, In Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (Sendai, Japan), vol. 2, Sept. 2004, pp. 1511 – 1517.

[112] A. Savvides, C.C. Han, and M.B. Strivastava, *Dynamic fine-grained localization in ad-hoc networks of sensors*, 7th annual international conference on Mobile computing and networking (Rome, Italy), 2001, pp. 166–179.

[113] J. B. Saxe, *Embeddability of weighted graphs in k-space is strongly np-hard*, In Proc. 17th Alleron Conf. Commun. Control Comput., 1979, pp. 480–489.

[114] S. Se, D. Lowe, and J. Little, *Vision-based mobile robot localization and mapping using scale-invariant features*, IEEE Intl. Conf. on Robotics and Automation (Seoul,Korea), May 2001.

[115] H. Shatkay and L. P. Kaelbling, *Learning topological maps with weak local odometric information*, International Joint Conference on Artificial Intelligence (San Mateo, CA), 1997, pp. 920–929.

[116] J. Shin, L. J. Guibas, and F. Zhao, *A distributed algorithm for managing multi-target identities in wireless ad-hoc sensor networks*, Information Processing in Sensor Networks Second International Workshop, IPSN 2003 (Palo Alto, CA), April 2003, pp. 223–238.

[117] S. Simhon and G. Dudek, *A global topological map formed by local metric maps*, IEEE Intenational Conference on Intelligent Robots and Systems (Victoria, Canada), vol. 3, October 1998, pp. 1708–1714.

[118] R. Smith, M. Self, and P. Cheeseman, *Estimating uncertain spatial relationships in robotics*, Autonomous Robot Vehicles (I.J. Cox and G. T. Wilfong, eds.), Springer-Verlag, 1990, pp. 167–193.

[119] R. C. Smith and P. Cheeseman, *On the representation and estimation of spatial uncertainty*, The International Journal of Robotics Research **5** (1986), no. 4, 56–68.

[120] S. Solomon, D. Qin, M. Manning, Z. Chen, M. Marquis, K.B. Averyt, M.Tignor, and H.L. Miller (eds.), *IPCC, 2007: Summary for policymakers. in: Climate change 2007: The physical science basis. contribution of working group I to the fourth assessment report of the intergovernmental panel on climate change*, Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA, 2007.

[121] C. Stauffer and K. Tieu, *Automated multi-camera planar tracking correspondence modeling*, Proceedings of the IEEE Computer Vision and Pattern Recognition, vol. 1, July 2003, pp. 259–266.

[122] G. P. Stein, *Tracking from multiple view points: Self-calibration of space and time*, Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on., vol. 1, June 1999, pp. 521–527.

[123] E. N. Swanston, D. Gregory, D. B. Richardson, and H. Campbell, *An overview of the metropolitan toronto traffic control computer system*, In Proc. of the 1983 International Electrical and Electronics Conference (Toronto, Ontario), March 1983, pp. 202–208.

[124] M. Tanner, *Tools for statistical inference*, 3 ed., Springer Verlag, New York, 1996.

[125] S. Teller, J. Chen, and H. Balakrishnan, *Pervasive pose-aware applications and infrastructure*, IEEE Computer Graphics and Applications, July/August 2003, pp. 14–18.

[126] S. Thrun, D. Fox, and W. Burgard, *A probabilistic approach to concurrent mapping and localization for mobile robots*, Machine Learning and Autonomous Robots (joint issue) (1998).

[127] D.A. Tregouet, S. Escolano, L. Tiret, A. Mallet, and J. L. Golmard, *A new algorithm for haplotype-based association analysis:the stochastic-EM algorithm*, Annals of Human Genetics **68** (2004), no. 165.

[128] Y. Vardi, *Network tomography: estimating source-destination traffic intensities from link data*, Journal of the American Statistical Association **91** (1996), 365–377.

[129] H. Wang, J. Elson, L. Girod, D. Estrin, and K. Yao, *Target classification and localization in a habitat monitoring application*, In Proc. of the IEEE ICASSP, 2003.

[130] G.C.G. Wei and M.A. Tanner, *A monte-carlo implementation of the EM algorithm and the poor man's data augmentation algorithms*, Journal of the American Statistical Association **85(411)** (1990), 699–704.

[131] D. F. Wolf and G. S. Sukhatme, *Mobile robot simultaneous localization and mapping in dynamic environments*, Autonomous Robots **19** (2005), no. 1, 53–65.

[132] Y. Xu, J. Heidemann, and D. Estrin, *Geography-informed energy conservation for ad hoc routing*, ACM/IEEE International Conference on Mobile Computing and Networking (Rome, Italy), USC/Information Sciences Institute, July 2001, pp. 70–84.

Document Log:

Manuscript Version 0.9

Typeset by $\mathcal{A}_{\mathcal{M}}\mathcal{S}$-LaTeX — 12 June 2009

Dimitrios Paul Marinakis

McGill University, 3480 University St., Montréal (Québec) H3A 2A7, Canada, *Tel.* : (514) 398-7071

E-mail address: dmarinak@@cim.mcgill.ca

Typeset by $\mathcal{A}_{\mathcal{M}}\mathcal{S}$-LaTeX