Simultaneous Localization and Tracking in Wireless Ad-hoc Sensor Networks

by

Christopher J. Taylor

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degrees of

Bachelor of Science in Computer Science and Engineering

and Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

May 6, 2005

Author .....
Department of Electrical Engineering and Computer Science
May 6, 2005

Certified by .....
Jonathan Bachrach
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Simultaneous Localization and Tracking in Wireless Ad-hoc Sensor Networks

by

Christopher J. Taylor

Submitted to the
Department of Electrical Engineering and Computer Science

May 6, 2005

In Partial Fulfillment of the Requirements for the Degrees of
Bachelor of Science in Computer Science and Engineering
and Master of Engineering in Electrical Engineering and Computer Science

Abstract

In this thesis we present LaSLAT, a sensor network algorithm that uses range measurements between sensors and a moving target to simultaneously localize the sensors, calibrate sensing hardware, and recover the target's trajectory.

LaSLAT is based on a Bayesian filter that updates a probability distribution over the parameters of interest as measurements arrive. The algorithm is distributable and requires a fixed amount of storage space with respect to the number of measurements it has incorporated. LaSLAT is easy to adapt to new types of hardware and new physical environments due to its use of intuitive probability distributions: one adaptation demonstrated in this thesis uses a mixture measurement model to detect and compensate for bad acoustic range measurements due to echoes.

We present results from a centralized implementation of LaSLAT using a network of Cricket sensors. In both 2D and 3D networks, LaSLAT is able to localize sensors to within several centimeters of their ground truth positions while recovering a range measurement bias for each sensor and the complete trajectory of the mobile.

Thesis Supervisor: Jonathan Bachrach
Title: Research Scientist

Acknowledgments

First and foremost I would like to thank Ali Rahimi, who has been an inexhaustible source of instruction and concrete good ideas throughout the year. It is fair to say that I could not have done this without his help.

I also want to thank my advisor, Jonathan Bachrach, who gave me the chance to tackle a difficult problem and helped me stay focused and on target. He also made sure I had resources when I needed them, and did the legwork to acquire Cricket hardware for me, often on short notice.

And of course, thanks to Jonathan and Vijay Raghavan, our DARPA program manager, for formulating the simultaneous localization and tracking problem. Without their inspiration I would never have stumbled upon this topic.

I also appreciate the efforts of two UROPs in my group, Tony Grue and Tom Hsu, who, along with Jonathan, came in on nights and weekends to help me set up and conduct experiments when the lab was empty. Measuring sensor positions with a tape measure is a labor-intensive and unpleasant process – our struggles provide all the motivation this thesis requires.

I am grateful to DARPA for funding this research under contract number F33615-01-C-1896.

Finally, thanks to my parents and fiancee Jennifer for their support, understanding, and encouragement throughout the process. They suffered my complaints when nothing was working and suffered my delusions of grandeur when everything was.

Contents

List of Figures

Chapter 1

Introduction

This thesis presents LaSLAT, an algorithm for localizing and calibrating a sensor network while simultaneously tracking a moving target. LaSLAT performs these tasks using range measurements between the sensors and the target. LaSLAT does not require sensor nodes with known positions, ranging information between sensors, or constraints on the path of the target.

1.1 Localization

Many sensor network applications require that the locations of the individual sensors be known, since sensor readings are in general of little use without geographic context. However, the same attributes that make sensor networks attractive make obtaining this information difficult. By placing a large number of relatively cheap sensors, it is possible to obtain many accurate measurements from sensors close to phenomena of interest; however, the sheer number of sensors and the need to minimize costs precludes manually recording the sensors' locations. It also precludes brute force solutions such as equipping each sensor with a GPS unit.

Consequently, we would like the sensors to determine their own positions after placement. This is known as *localization*, and is typically achieved by having each sensor compute range measurements to its neighboring sensors, then algorithmically embedding the graph formed by these ranges into a coordinate system (see chapter

2). This coordinate system is then used to perform location-dependent tasks such as geographic packet routing or target tracking.

Localization is often complicated by the difficulty of obtaining enough accurate pair-wise range measurements between sensors. Inter-sensor ranges can be corrupted by noise or lost entirely due to occluded line-of-sight. Thus, consistently accurate localization requires robustness in the face of missing or low quality measurements.

Nevertheless, localization is rarely if ever the purpose of a network. Sensor networks are typically deployed to observe active phenomena in the environment, and require accurate localization as a means to that end. As a result, there is pressure in localization research to achieve accuracy and robustness using as little hardware as possible.

1.2 Tracking

Target tracking is one of the motivating localization-dependent applications of sensor networks. In tracking applications, sensors jointly observe phenomena, which may be people or objects passing through the network or physical effects such as bullet shock-waves or anomalous sounds. Once a phenomenon is detected, the sensors collaborate to determine its spatial location. This estimate is reported to a computer or person monitoring the network. Target tracking networks can provide indoor navigation services to hand-held users or mobile robots, track friendlies and hostiles on a battlefield, or monitor movement of inventory in a store or warehouse.

Tracking is a well-understood problem (see chapter 2). Given the locations of the sensors and accurate range information to the target, it is straightforward to determine the target's position. Consequently, traditional tracking applications tend to be split into two separate phases. First, the network is localized using a specialized algorithm. After localization completes, the network enters a target tracking phase in which target positions are estimated based on the discovered sensor positions.

1.3 Calibration

In an ideal world, sensors would arrive from the factory fully calibrated to begin taking accurate measurements of their surroundings. However, this ideal situation is rarely achieved. For instance, deployment conditions such as temperature affect the accuracy of ranging algorithms based on acoustic time-of-flight by altering the speed of sound. Furthermore, as shown in [35], differences between sensors can also result in mis-calibrations that are difficult to correct before deployment. Calibration in the field can therefore offer meaningful improvement in both localization and target tracking accuracy. As with localization, there is considerable economic incentive to develop auto-calibration algorithms that allow sensors to self-calibrate in the field without external intervention.

1.4 Simultaneous Localization and Tracking

For sensor networks deployed to track moving targets, some authors have suggested using a moving target (or *mobile*) to assist in localizing the network [8, 12, 27]. This approach is attractive for several reasons:

- It requires no additional localization-specific hardware on the individual themselves, potentially reducing both their size and cost.

- Unlike sensors, a mobile can move freely in the network. This provides a larger number and a greater diversity of measurements for use in localization, which helps reduce the effect of noisy measurements and static environmental obstacles.

- Use of a mobile means that the sensors are no longer constrained to have line of sight to each other. This enables a broader range of network deployments: in particular, it allows sensors to be placed to optimize tracking coverage rather than to facilitate localization. This is especially significant because many types of ranging hardware are directional. Traditional localization algorithms there-

fore introduce a deployment conflict: sensors must be oriented so that they can range to several other sensors in addition to observing areas where targets are likely to appear. This wastes sensing resources and limits deployment options. SLAT eliminates this problem, since it works best when sensors use their full field of view to sense targets.

- Localization using a mobile potentially allows localization estimates to continuously improve, even after the network begins tracking targets. In particular, sensors that are closest to high traffic areas can be localized very precisely. In turn, this facilitates high precision target tracking in these areas.

Existing methods fail to realize the full potential of mobile-based localization. In particular, most require that the position of the mobile be known at all times. Consequently, the mobile must be constrained to a well-known trajectory or equipped with a GPS-like system. Furthermore, existing methods ignore the sensor calibration problem entirely.

We consider a more general scenario, in which the mobile is allowed to move arbitrarily on an unknown path. As it moves, the mobile periodically emits a signal that allows sensors in the network to compute their range to the mobile. These ranges are expected to be corrupted by noise. Using solely these noisy range measurements, the objective is to localize the sensors in the network, calibrate the sensors' ranging hardware if necessary, and track the position of the mobile. Accordingly, we label this problem Simultaneous Localization and Tracking (SLAT).

1.5 LaSLAT

Our solution to the SLAT problem takes the form of a Bayesian filter. The filter uses range measurements taken by the network to update a joint probability distribution over the positions of the sensors, the trajectory of the mobile, and the calibration parameters of the network. To avoid some of the representational and computational complexity of general Bayesian filtering, we use Laplace's method to approximate our

state with a Gaussian after incorporating each batch of measurements. As a result, we call our algorithm LaSLAT.

LaSLAT inherits many desirable properties from the Bayesian framework. The probabilistic model used in LaSLAT insures that measurement noise is averaged out as more measurements become available, which improves localization and tracking accuracy in high-traffic areas. The filtering framework incorporates measurements in small batches, providing on-line estimates of all locations, calibration parameters, and their uncertainties. This allows the mobile to be tracked in near real-time. It also speeds up the convergence of the algorithm and reduces impact on the network.

Since LaSLAT maintains estimate uncertainties, mobiles can be dispatched on-line if desired to improve localization estimates in regions where localization uncertainty is high. As we required in the problem statement, mobiles may move arbitrarily through the environment, with no constraint on their trajectory or velocity. Furthermore, multiple mobiles may be used in conjunction to expedite initial localization.

If available, ancillary localization information such as position estimates from GPS, anchor nodes, or radio-based ranging can be easily incorporated into our framework. Our algorithm avoids expensive matrix operations by operating on sparse inverse covariance matrices rather than operating directly on dense covariance matrices. This helps LaSLAT run quickly and facilitates performing the LaSLAT computations distributedly in the network.

The user deploying the network may specify a coordinate system for LaSLAT to honor when localizing sensors. If no coordinate system is specified, LaSLAT recovers locations in a coordinate system that is correct up to a translation, rotation, and possible reflection.

We demonstrate these features by accurately localizing a dense network of 27 sensors to within one or two centimeters. The sensor nodes are wireless Crickets [1] capable of measuring their distance to a moving beacon using a combination of ultrasound and radio pulses. In a larger and sparser network, we localize sensors to within about eight centimeters. In both cases, a measurement bias parameter is accurately calibrated for all nodes. Finally, we present results from two experiments

in three dimensions, in which nodes were localized to within seven centimeters.

Chapter 2

Related Work

2.1 Localization

Localization is a well established problem in sensor networks, and many approaches have been developed in the literature. The vast majority of these techniques treat localization as a stand-alone procedure that takes place in advance of or concurrently with other operations in the network. In the traditional localization problem, sensors are capable of determining the distance between themselves and other nearby sensors. This may be done using relatively accurate methods such as acoustic time-of-flight (TOF), or using imprecise but cheap methods such as radio signal strength indicators (RSSI). In dense networks, radio hop count (DV) may be used as a surrogate for distance.

Several authors [10, 17, 31] have used multidimensional scaling (MDS) as a localization technique. MDS is very effective at recovering sensor topologies when accurate distances are available between all pairs of sensors. Its performance degrades substantially when some distances are unavailable.

Many algorithms [4, 5, 20, 23, 25, 30, 32] also depend on the presence of *anchor* nodes, which are sensors that know their true locations a priori through out-of-band means such as pre-programming or GPS. These algorithms localize non-anchor nodes by construction using distances from each non-anchor node to several anchor nodes. However, anchor nodes are expensive, and not always perfectly accurate (i.e. GPS).

Furthermore, many of these algorithms suffer from propagation of errors: sensors several hops from a beacon often accumulate considerable position error.

Priyantha et al. [29] suggest a two-phase approach that is similar in some ways to the approach proposed in this thesis. An initialization phase uses an imprecise technique based on radio connectivity between nodes to embed the sensors in a plane. Once initialization is complete, a second phase uses precise time of flight ranges between adjacent sensors to make local position adjustments.

Ihler et al. [16] treat localization as an inference problem on a graphical model. This allows them to use nonparametric belief propagation (NBP) to produce an estimate of sensor positions. This offers several advantages, many of which are also offered by our LaSLAT technique. NBP allows the use of non-Gaussian measurement models. It also produces an uncertainty measure that provides a context for use of its sensor position estimates.

A number of authors [7,21,22,24,31] have proposed algorithms based on coordinate system stitching. These algorithms follow a three step divide-and-conquer process. First, the network is split into small overlapping subregions. Next, each subregion computes a *local map*, which is simply an embedding of the subregion in a relative coordinate system. Finally, adjacent subregions are *registered* into a common coordinate system using the overlapping nodes between local maps. By performing registration recursively, all the subregions are incorporated into a single global coordinate system.

Moore et al. [22] suggest a particularly sophisticated approach to subregion formation, in which the subregions are chosen based on their likelihood of forming accurate local maps. This technique has the advantage of producing a more homogeneous accuracy level across the network. However, it achieves this by refusing to localize sensors that cannot be positioned using pair-wise ranges to nearby nodes.

This illustrates a fundamental problem with localization based on inter-sensor ranges: the limited availability of inter-sensor range data means that some sensors may prove impossible to accurately localize. The advantage of simultaneous localization and tracking (SLAT) is that these nodes may become localizable in the future after additional sightings of the mobile. Furthermore, SLAT offers highest accuracy

where mobiles move most frequently, whereas inter-sensor range-based algorithms typically offer highest accuracy where sensor density is greatest or in areas where ranging between sensors is easiest. In an ideal world, these regions would coincide, but in an ad-hoc deployment, this may not be the case.

2.2 Tracking

Like localization, target tracking is a well established problem in sensor networks. However, most literature on the subject assumes that localization information is available before tracking begins. As a result, most of the literature on tracking considers cluster formation [9], power economy [36], or related problems such as target identification and classification [3].

In this thesis, we assume that targets are easy for sensors to identify, allowing us to focus purely on tracking. This is reasonable in any circumstance where the targets are tracked voluntarily – for instance when the network is guiding a mobile robot or providing in-building location information to a hand-held computer user. We also ignore concerns such as power conservation, and focus instead on what information can be extracted from sensor measurements once they are obtained.

2.3 Calibration

Several authors [6, 35] have researched automatic calibration in sensor networks. Bychkovskiy et al. [6] calibrate photo-sensors in a dense network by making use of the fact that adjacent sensors are likely to observe the same level of stimulus.

The most directly relevant work to this thesis is the Calamari Ad-hoc Localization System by Cameron Whitehouse [35]. Calamari uses acoustic time difference of arrival (TDoA) ranging between sensors to perform localization. Whitehouse showed that post-deployment calibration of the acoustic sensors dramatically improved localization accuracy. He also developed a macro-calibration technique in which individual sensor parameters are chosen to optimize the performance of the network as a whole.

2.4 SLAT

Various authors have used mobiles to localize sensor networks [8, 12, 26–28], but these methods assume the location of the mobile is known. One exception is [12], which builds a constraint structure as measurements become available. Compared to [12], we employ a very extensible statistical framework that allows more realistic measurement models. Priyantha et al. [28] show how to guide a mobile to form rigidly localizable structures. Our problem differs from the one discussed in [28] in that we make no assumptions about the trajectory of the mobile.

Our method is most similar to [26], which used an Extended Kalman Filter (EKF) to track an underwater vehicle while localizing sonar beacons capable of measuring their range to the vehicle. We replace the EKF's approximate measurement model with one based on Laplace's method. This provides faster convergence and greater estimation accuracy. We also demonstrate that the Bayesian filtering framework can calibrate the sensor nodes, and that the computation is capable of distributing over the sensor nodes in a straightforward way.

2.5 SLAM

The SLAT problem is similar to SLAM and the 3D Structure from Motion (SFM) problem in computer vision. In these problems, two sets of unknown variables are coupled in such a way that jointly estimating the two sets is relatively difficult, while estimating one set with the other set given is relatively easy. In sensor network localization, knowing the position of the nodes significantly simplifies tracking, and knowing the position of the mobile significantly simplifies localization. In SLAM and SFM, this relationship holds between the pose of the robot or camera and that of features in the scene.

Our solution to SLAT adopts various important refinements to the original Extended Kalman Filter (EKF) formulation of SLAM [33]. LaSLAT processes measurements in small batches and discards variables that are no longer needed, as demon-

strated by McLauchlan [19]. Following [34], LaSLAT operates on inverse covariances of Gaussians rather than on covariances directly to accelerate updates and facilitate distributed computation.

Chapter 3

LaSLAT

LaSLAT uses a Bayesian filtering framework. Under this framework, as each batch of measurements becomes available, it is used to update a prior distribution over sensor locations, the mobile trajectory, and various sensing parameters. The resulting posterior distribution is then propagated forward in time using a dynamics model to make it a suitable prior for use with the next batch of measurements.

In LaSLAT, after incorporating each batch of measurements the posterior distribution is approximated with a Gaussian using Laplace's method [13]. Consequently, the amount of state saved between batches is constant with respect to the number of measurements taken in the past. The Gaussian approximation also simplifies propagation using the dynamics model and incorporation of the next batch of measurements.

3.1 Approximate Bayesian Filtering for SLAT

As the mobile moves through the network, it periodically emits *events* which allow some of the sensors to measure their distances from the mobile.

Let $\mathbf{e}_j$ denote the location of the mobile when it generated the jth event. The tth batch $\mathbf{e}^t$ is a collection of consecutive events $\mathbf{e}^t = \{\mathbf{e}_m \dots \mathbf{e}_{m+n}\}$, with $\mathbf{e}^t_j$ denoting the jth event in the tth batch. Each LaSLAT iteration incorporates the measurements from a single batch of events.

Let $\mathbf{s}_i = \begin{bmatrix} \mathbf{s}_i^x & \mathbf{s}_i^\theta \end{bmatrix}$ represent the unknown parameters of sensor i, with $\mathbf{s}_i^x$ denoting

the sensor's position and $\mathbf{s}_i^\theta$ its calibration parameters. Then $\mathbf{s} = \{\mathbf{s}_i\}$ is the set of all sensor parameters. The scalar y_{ij}^t denotes the range measurement between sensor i and the jth event in batch t, so $\mathbf{y}^t = \{y_{ij}^t\}$ is the collection of all range measurements in batch t.

For each batch t, $\mathbf{e}^t$ and $\mathbf{s}$ are the unknown values that must be estimated. We aggregate these unknowns into a single variable $\mathbf{x}^t = \begin{bmatrix}\mathbf{s} & \mathbf{e}^t\end{bmatrix}$ for notational simplicity. Note that as defined, $\mathbf{e}$ and $\mathbf{s}^x$ may be vectors of either 2D or 3D points, allowing LaSLAT to be run easily in either two or three dimensions.

The Bayesian filtering framework is a non-linear, non-Gaussian generalization of the Kalman Filter. For each batch t, it computes the posterior distribution over sensor parameters, $\mathbf{s}$, and events locations, $\mathbf{e}^t$, taking into account all range measurements taken so far:

$$p(\mathbf{x}^t|\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^t).$$

In LaSLAT, we wish to update this distribution as range measurements become available, and discard measurements as soon as they have been incorporated. To do this, one can rewrite the distribution in terms of a measurement model and a prior distribution derived from the results of the previous iteration. Rewriting $p(\mathbf{x}^t|\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^t)$ as $p(\mathbf{x}^t|\mathbf{y}^{old}, \mathbf{y}^t)$, we get by Bayes' rule:

$$\begin{aligned} p(\mathbf{x}^t|\mathbf{y}^{old}, \mathbf{y}^t) &\propto p(\mathbf{y}^t, \mathbf{x}^t|\mathbf{y}^{old}) \\ &\propto p(\mathbf{y}^t|\mathbf{x}^t, \mathbf{y}^{old}) p(\mathbf{x}^t|\mathbf{y}^{old}) \\ &\propto p(\mathbf{y}^t|\mathbf{x}^t) p(\mathbf{x}^t|\mathbf{y}^{old}), \end{aligned} \tag{3.1}$$

where proportionality is with respect to $\mathbf{x}^t$. The final equality follows because when the sensor and mobile locations are known, the past measurements do not provide any additional useful information about the new batch of measurements. The distribution $p(\mathbf{y}^t|\mathbf{x}^t)$ is the measurement model: it reflects the probability of a set of observations given a particular configuration of sensors and event locations (Section 3.2).

The distribution $p(\mathbf{x}^t|\mathbf{y}^{old})$ summarizes all information collected prior to the cur-

rent batch of measurements, in the form of a prediction of $\mathbf{x}^t$ and an uncertainty measure. It can be computed from the previous estimate, $p(\mathbf{x}^{t-1}|\mathbf{y}^{old})$, by applying a dynamic model:

$$p(\mathbf{x}^t|\mathbf{y}^{old}) = \int_{\mathbf{x}^{t-1}} p(\mathbf{x}^{t-1}|\mathbf{y}^{old})p(\mathbf{x}^t|\mathbf{x}^{t-1})\ d\mathbf{x}^{t-1}. \tag{3.2}$$

The distribution $p(\mathbf{x}^t|\mathbf{x}^{t-1})$ models the dynamics of the configuration from one batch to another by discarding old event locations and predicting the locations of new events (Section 3.4).

When the measurement model $p(\mathbf{y}^t|\mathbf{x}^t)$ is not Gaussian, the updates (3.1) and (3.2) become difficult to compute. We handle the non-Gaussianity of the measurement model by approximating the posterior $p(\mathbf{x}^t|\mathbf{y}^{old})$ with a Gaussian distribution $q(\mathbf{x}^t|\mathbf{y}^{old})$ using Laplace's method (Section 3.3). This Gaussian becomes the basis for the prior distribution for the next batch. q is much simpler to save between batches than the full posterior – in particular, it allows all the old measurements to be discarded. Table 3.1 summarizes the steps of LaSLAT.

Other approximate Bayesian filters such as the Extended Kalman Filter (EKF) or particle filters could also be used in place of our Laplacian method. The EKF differs from our algorithm because it does not perform a full optimization when incorporating each event. In many cases this is a helpful optimization; however, as we show in chapter 6, on the SLAT problem it sacrifices accuracy and speed of convergence. Particle filters allow a closer approximation of the posterior distribution, especially when the distribution is multi-modal. However, our algorithm seems to perform well in practice while requiring significantly less computation.

3.2 Measurement Model

New measurements influence localization and calibration estimates via the measurement model. A measurement model is a probability distribution $p(\mathbf{y}^t|\mathbf{x}^t)$ over a batch of range measurements, given a particular choice of the calibration parameters and

1. Observe a new batch of measurements $\mathbf{y}^t$.
2. Represent the posterior $p(\mathbf{x}^t|\mathbf{y}^t, \mathbf{y}^{old})$ in terms of the prior $p(\mathbf{x}^t|\mathbf{y}^{old})$ and the measurement model $p(\mathbf{y}^t|\mathbf{x}^t)$ using Equation (3.1).
3. Using Newton-Raphson [14], compute curvature at the mode of $p(\mathbf{x}^t|\mathbf{y}^t, \mathbf{y}^{old})$ and use it to construct the approximate posterior $q(\mathbf{x}^t|\mathbf{y}^t, \mathbf{y}^{old})$. This posterior is the estimate for the batch t (Section 3.3).
4. Compute the prediction $p(\mathbf{x}^{t+1}|\mathbf{y}^t, \mathbf{y}^{old})$ using $q(\mathbf{x}^t|\mathbf{y}^t, \mathbf{y}^{old})$ (Section 3.4).
5. Using the prediction as the new prior, return to step 1 to process batch $t + 1$.

Table 3.1: One iteration of LaSLAT. Incorporates batch t and prepares to incorporate batch $t + 1$.

positions for the sensors and the mobile. One advantage of the LaSLAT framework is that this measurement model can be tailored to specific measurement hardware and deployment parameters. The measurement model encapsulates details such as the kind of data being observed (angle of arrival, distance, radio signal strength, etc), as well as the types of noise that are possible in the environment.

In this thesis, we develop a measurement model that reflects the characteristics of the popular acoustic time difference of arrival (TDoA) ranging technique. In most TDoA implementations, the transmitter (in this case the mobile) emits a tagged radio message, then a short time later produces an acoustic signal. Upon hearing the radio message, the sensors in the network activate their microphones and listen for the arrival of the acoustic signal. Once the acoustic signal arrives, the sensors use the difference in arrival time between the acoustic signal and the radio message to estimate the distance between the sensor and the transmitter.

This technique can be quite accurate, especially when line-of-sight exists between the transmitter and the receiver. The Cricket ranging system [1], which uses a single inaudible ultrasound pulse, can measure distances up to ten meters with error as small as 1-2 centimeters.

However, TDoA measurements are susceptible to several types of error. First, various random delays in the ranging process introduce small (sub-centimeter) errors, which take the form of a variance from measurement to measurement. Second,

mis-calibration can result in a measurement *bias*. For instance, temperature or humidity can cause the actual speed of sound to be different from the pre-calibrated constant used by the sensor. The measurement bias can easily vary from deployment to deployment, making accurate pre-calibration at the factory nearly impossible. In the Cricket ranging system, we have observed biases of between 3 and 10 centimeters. The biases vary only slightly from sensor to sensor, but are typically consistent throughout a deployment area. Finally, TDoA measurements can be vulnerable to both echoes and ambient noise. In both cases errors occur because the time of flight observed by the sensor does not correspond to the straight line distance from the sensor to the event; instead, it corresponds to a longer path (in the case of an echo), or no path at all (in the case of an ambient noise). In one test environment, a security system produced 40 kHz ultrasound that occasionally interfered with the Crickets' ranging system.

In this section, we ignore echo effects, and assume that each measurement is a corrupted version of the true distance between the event and the sensor that took the measurement:

$$y_{ij}^t = \|\mathbf{s}_i^x - \mathbf{e}_j^t\| + s_i^\theta + \omega_{ij}^t, \tag{3.3}$$

where $\|\cdot\|$ indicates the vector 2-norm, giving the Euclidean distance between $\mathbf{s}_i^x$ and $\mathbf{e}_j^t$. ω_{ij}^t is a zero-mean Gaussian random variable with variance σ^2, and $\mathbf{s}_i^\theta$ is a bias parameter that models an unknown shift due to sensor mis-calibration. In section 4.1 we present a richer model that performs better in the presence of echoes.

As defined in equation (3.3), $p(y_{ij}^t|\mathbf{s}_i, \mathbf{e}_j^t)$ is a univariate Gaussian with mean $\|\mathbf{s}_i - \mathbf{e}_j^t\| + s_i^\theta$ and variance σ^2. Since each measurement y_{ij}^t depends only on the sensor $\mathbf{s}_i$ that took the measurement and the location $\mathbf{e}_j^t$ of the mobile when it generated the event, the measurement model factorizes according to

$$p(\mathbf{y}^t|\mathbf{x}^t) = \prod_{i,j} p(y_{ij}^t|\mathbf{s}_i, \mathbf{e}_j^t), \tag{3.4}$$

where the product is over the sensors and the events that they perceived in batch t. Equation (3.4) is the complete measurement model for a batch of measurements.

Though LaSLAT is designed to operate in their absence, LaSLAT can make use of inter-sensor range measurements if they are available. These can be encoded using a generative model similar to equation (3.3):

$$y_{kl}^t = \|\mathbf{s}_k^x - \mathbf{s}_l^x\| + s_k^\theta + \omega_{kl}^t, \tag{3.5}$$

where $\mathbf{s}_k$ and $\mathbf{s}_l$ are the sensor pair that generates the inter-sensor range measurement y_{kl}^t, and ω_{kl}^t is additive zero-mean Gaussian noise. The resulting probability distribution $p(y_{kl}^t|\mathbf{s}_k, \mathbf{s}_l)$ becomes an additional factor in the product (3.4). This demonstrates the ease with which LaSLAT can rigorously incorporate additional sources of localization information.

3.3 Incorporating Measurements

The measurement model derived in the last section can be combined with a prior distribution $p(\mathbf{x}^t|\mathbf{y}^{old})$ using equation (3.1) to find the posterior distribution $p(\mathbf{x}^t|\mathbf{y}^{old}, \mathbf{y}^t)$.

This posterior has no compact representation – in particular, it takes up space proportional to the total number of measurements observed by the network since the first batch. To curb this complexity, we save only a Gaussian approximation of the posterior. This Gaussian representation requires constant space with respect to the number of measurements observed, which allows LaSLAT to run indefinitely without requiring additional memory. It also has the advantage of being easy to distribute across the sensor network (chapter 5).

This approximate Gaussian posterior $q(\mathbf{x}^t|\mathbf{y}^{old}, \mathbf{y}^t)$ can be obtained from the prior distribution $p(\mathbf{x}^t|\mathbf{y}^{old})$ and the measurement model $p(\mathbf{y}^t|\mathbf{x}^t)$ using Laplace's method [13].

To fit an approximate Gaussian distribution $q(x)$ to a distribution $p(x)$, Laplace's method first finds the mode x^* of $p(x)$, then computes the curvature of the negative log posterior at x^*.

$$\Lambda^{-1} = -\frac{\partial^2}{\partial x^2} \log p(x)\Big|_{x=x^*}.$$

The mean and covariance of $q(x)$ are then set to x^* and Λ respectively. Notice that when p is Gaussian, the resulting approximation q is exactly p. For other distributions, the Gaussian q locally matches the behavior of p about its mode.

The mode finding problem can be expressed as:

$$\begin{aligned}
\mathbf{x}^{t*} &= \arg\max_{\mathbf{x}^t} p(\mathbf{x}^t|\mathbf{y}^t, \mathbf{y}^{old}) \\
&= \arg\min_{\mathbf{x}^t} -\log\left[p(\mathbf{y}^t|\mathbf{x}^t)p(\mathbf{x}^t|\mathbf{y}^{old})\right] \\
&= \arg\min_{\mathbf{x}^t} (\mathbf{x}^t - \mu)^T \Omega_{\mathbf{x}} (\mathbf{x}^t - \mu) + \frac{1}{\sigma^2} \sum_{i,j} (\|\mathbf{s}_i^x - \mathbf{e}_j^t\| + s_i^\theta - y_{ij}^t)^2, \qquad (3.6)
\end{aligned}$$

where $\mu = E\left[\mathbf{x}^t|\mathbf{y}^t, \mathbf{y}^{old}\right]$, and $\Omega_{\mathbf{x}} = \mathrm{Cov}^{-1}\left[\mathbf{x}^t|\mathbf{y}^{old}\right]$.

We use the Newton-Raphson iterative optimization algorithm [14] to find the mode $\mathbf{x}^{t*}$ and the curvature $\mathbf{H}$ (Appendix B). Following Laplace's method, the mean $E\left[\mathbf{x}^t|\mathbf{y}^{old}, \mathbf{y}^t\right]$ of q is set to $\mathbf{x}^{t*}$ and its inverse covariance $\mathrm{Cov}^{-1}\left[\mathbf{x}^t|\mathbf{y}^{old}, \mathbf{y}^t\right]$ is set to $\mathbf{H}$. Representing q using its inverse covariance allows us to avoid computing the matrix inverse $\mathbf{H}^{-1}$ after adding each measurement, which significantly improves performance and facilitates a distributed implementation of our algorithm (Chapter 5).

The Gaussian approximation described in this section works well when the posterior distribution has a single strong mode. However, when the posterior contains several equally probable modes, the Gaussian approximation effectively discards all but one. Consequently, this algorithm is vulnerable to substantial errors when the measurements and prior do not favor a unique estimate. This can occur if the mobile does not move very much during a batch and the prior distribution is mostly uninformative. In that case, the Gaussian approximation risks committing too quickly to a particular parameter estimate. In our experience, this problem can be avoided by using larger batch sizes when the prior's covariance is large. It can also be solved by using a Gaussian mixture model or a particle filter to approximate the posterior distribution between batches. These alternatives are able to fit a multi-modal posterior; however, they are more complex and consequently require greater computational power to manipulate.

3.4 Dynamics Model

In this thesis, we assume mobiles can move arbitrarily and that sensors are stationary. When propagating the posterior $q(\mathbf{x}^t|\mathbf{y}^{old}, \mathbf{y}^t)$ forward in time, we need only retain the components that are useful for incorporating the next batch of measurements. Thus, we may remove the estimate of the mobile's trajectory from batch t, but we must incorporate a guess for the mobile's path during batch $t+1$. Therefore, the prediction step of Equation (3.2) can be written:

$$p(\mathbf{x}^{t+1}|\mathbf{y}^{old}, \mathbf{y}^t) = p(\mathbf{s}, \mathbf{e}^{t+1}|\mathbf{y}^{old}, \mathbf{y}^t) = p(\mathbf{e}^{t+1})q(\mathbf{s}|\mathbf{y}^{old}, \mathbf{y}^t) \tag{3.7}$$
$$q(\mathbf{s}|\mathbf{y}^{old}, \mathbf{y}^t) = \int_{\mathbf{e}^t} q(\mathbf{x}^t|\mathbf{y}^{old}, \mathbf{y}^t)\, d\mathbf{e}^t.$$

The Gaussian $q(\mathbf{x}^t|\mathbf{y}^{old}, \mathbf{y}^t)$ captures the posterior distribution over sensor locations given all measurements taken so far, and has already been computed by the method of section 3.3. We obtain $q(s|\mathbf{y}^{old}, \mathbf{y}^t)$, by marginalizing out the mobile's trajectory during batch t.

The prior $p(\mathbf{e}^{t+1})$ is Gaussian with very broad covariance, indicating that the future trajectory of the mobile is unknown. In some applications, it may be possible to use past trajectories to make better guesses for $\mathbf{e}^{t+1}$. For example, $p(\mathbf{e}^{t+1})$ could be used to require events to form a smooth path. This can help accurately position events with few quality measurements. For maximum generality, we will not attempt to do so in this section, meaning that the mobile is allowed to move arbitrarily between events. In section 4.2, we present a method of requiring that events follow a smooth trajectory.

The operations of Equation (3.7) can be carried out numerically by operating on the mean and inverse covariance of $q(\mathbf{x}^t|\mathbf{y}^{old})$. First, partition according to $\mathbf{s}$ and $\mathbf{e}^t$:

$$E\left[\mathbf{x}^t|\mathbf{y}^{old}\right] = \begin{bmatrix} E[\mathbf{s}|\mathbf{y}^{old}] \\ E[\mathbf{e}^t|\mathbf{y}^{old}] \end{bmatrix}$$

$$\mathrm{Cov}^{-1}\left[\mathbf{x}^t|\mathbf{y}^{old}\right] = \begin{bmatrix} \Omega_{\mathbf{s}} & \Omega_{\mathbf{se}^t} \\ \Omega_{\mathbf{e}^t\mathbf{s}} & \Omega_{\mathbf{e}^t} \end{bmatrix}.$$

Marginalizing out $\mathbf{e}^t$ produces a distribution $q(\mathbf{s}|\mathbf{y}^{old})$ whose mean is the $\mathbf{s}$ component of the mean of $q(\mathbf{x}^t|\mathbf{y}^{old})$ and whose inverse covariance is:

$$\mathrm{Cov}^{-1}\left[\mathbf{s}|\mathbf{y}^{old}\right] = \Omega_{\mathbf{s}} - \Omega_{\mathbf{se}^t}\Omega_{\mathbf{e}^t}^{-1}\Omega_{\mathbf{e}^t\mathbf{s}}. \tag{3.8}$$

The parameters of $p(\mathbf{x}^{t+1}|\mathbf{y}^{old})$ are those of $q(\mathbf{s}|\mathbf{y}^{old})$, augmented by zeros to account for an uninformative prior on $\mathbf{e}^{t+1}$:

$$\begin{aligned} E\left[\mathbf{x}^{t+1}|\mathbf{y}^{old}\right] &= \begin{bmatrix} E\left[\mathbf{s}|\mathbf{y}^{old}\right] \\ \mathbf{0} \end{bmatrix} \\ \mathrm{Cov}^{-1}\left[\mathbf{x}^{t+1}|\mathbf{y}^{old}\right] &= \begin{bmatrix} \mathrm{Cov}^{-1}\left[\mathbf{s}|\mathbf{y}^{old}\right] & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}. \end{aligned} \tag{3.9}$$

The components of the inverse covariance of $p(\mathbf{x}^{t+1}|\mathbf{y}^{old})$ corresponding to $\mathbf{e}^{t+1}$ are set to $\mathbf{0}$, corresponding to infinite variance, which in turn captures our lack of *a priori* knowledge about the location of the mobile in the new batch. The mean is arbitrarily set to $\mathbf{0}$. If some information is known *a priori* about $\mathbf{e}^{t+1}$, then the $\mathbf{0}$ components of $E\left[\mathbf{x}^{t+1}|\mathbf{y}^{old}\right]$ and the bottom right $\mathbf{0}$ components of $\mathrm{Cov}^{-1}\left[\mathbf{x}^{t+1}|\mathbf{y}^{old}\right]$ can be used to capture that knowledge.

3.5 Prior Information and Initialization

Prior information about the sensor parameters is easy to incorporate into LaSLAT. Such information might be available because the sensors were placed in roughly known positions, or because another less accurate source of localization is available. Prior information may also be used to define LaSLAT's global coordinate system, by fixing the relative positions of several sensors. In addition, calibration in the factory might supply prior information.

If such prior information is available it can be supplied as the prior when incorporating the first batch of measurements. We set the covariance of this prior to $\sigma_0 \mathbf{I}$, with σ_0 a large scalar, which makes the prior diffuse. The large covariance allows measurements to override the positions prescribed by the prior, but provides a sensible default when few measurements are available. The mode of this prior (or for subsequent iterations, the mode of $p(\mathbf{s}|\mathbf{y}^{old})$) is also used as the initial iterate for the Newton-Raphson iterations. To obtain the initial iterate for an event, we use the average estimated location of the three sensors with the smallest range measurements to the event.

In our experiments, we utilize the radio connectivity of the sensors to obtain prior localization information. The initialization step described by Priyantha et al. [29] (see Appendix A) provides rough position estimates to serve as a prior before any measurements are introduced. This prior takes the form $p(\mathbf{s}^x) \propto \exp\left[-\frac{1}{2\sigma_0^2}\sum_i \|\mathbf{s}_i^x - x_i^0\|^2\right]$, where x_i^0 is the position of the ith sensor as predicted by the initialization step and σ_0 is a large variance.

We have observed empirically on Cricket hardware [1] that though measurement biases vary between deployments, they tend to be fairly consistent from sensor to sensor. This information can be used as a prior over the sensor measurement biases s_i^θ. We encode this information as a distribution with the form $p(\mathbf{s}^\theta) \propto \exp\left[-\frac{1}{2\sigma_b^2}\sum_{i\sim j}(s_i^\theta - s_j^\theta)^2\right]$, where the summation is over sensors that are in close proximity to each other, and σ_b is a constant used to tune the impact of this prior on the measurement biases. This illustrates a important attribute of LaSLAT: it is easy to add platform-specific optimizations to improve performance. In many localization algorithms this is impossible: a change in hardware or deployment environment often necessitates a completely new approach.

Chapter 4

LaSLAT Extensions

In the last chapter, we described the core LaSLAT algorithm. In this chapter, we consider several useful enhancements. First, we describe a measurement model that provides robust detection and elimination of erroneous range measurements. Then, we show how to require that LaSLAT's event position estimates follow a smooth path.

4.1 Robust Measurement Outlier Rejection

When using time difference of arrival (TDoA) acoustic ranging, many external factors can cause non-Gaussian measurement error. For instance, in some environments, ambient sounds can cause completely spurious measurements. In addition, environmental echoes can cause substantial delays in the sound pulse's time of flight, resulting in highly errant measurements. This effect is illustrated in figure 4-1. In these types of environments, it can be helpful to define a more robust measurement model. We propose a mixture model for this purpose. Let h_{ij}^t be a Bernoulli random variable that is 1 with probability p. Then a generative model of measurements might be:

$$y_{ij}^t = \begin{cases} \|s_i^x - \mathbf{e}_j^t\| + s_i^\theta + \omega_{ij}^t, & h_{ij}^t = 1 \\ u_{ij}^t, & h_{ij}^t = 0 \end{cases}, \tag{4.1}$$

where u_{ij}^t is a uniform random variable over all possible range measurements. Equation (4.1) produces an accurate measurement with probability p, and an uninformative

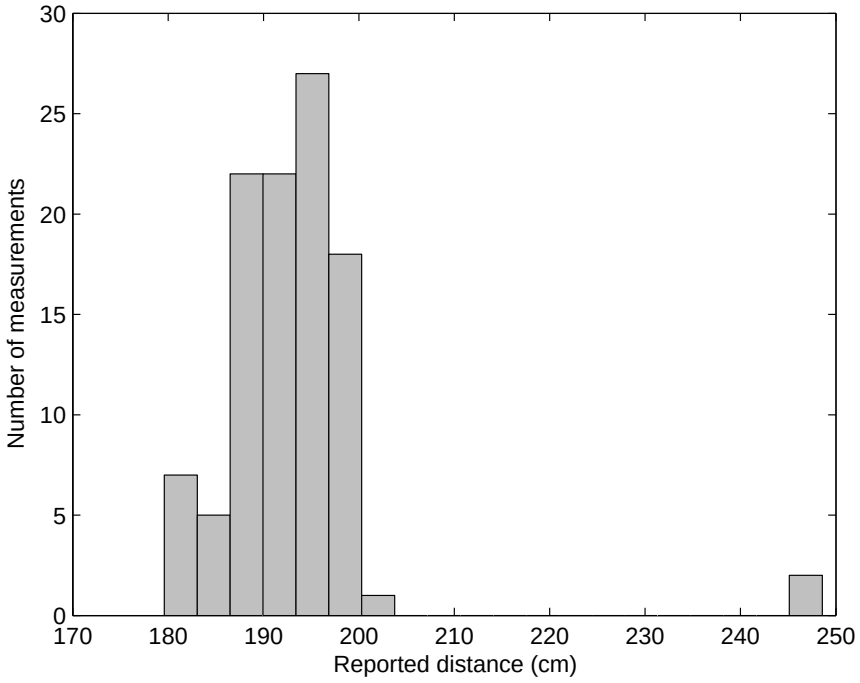

Figure 4-1: An example of outlying range measurements. This data was generated by rotating a beacon cricket at a fixed distance from a sensor. This histogram plots the range measurements taken by the sensor. Notice that the majority of the measurements are reasonably close to the true distance (much of the variance is caused by the movement of the ultrasound transmitter on the beacon with respect to the sensor); however, a few measurements are nearly half a meter too long. These outlying measurements can dramatically decrease localization and tracking performance, so it is desirable to detect and discard them.

and therefore useless result with probability $1 - p$. Physically, when a measurement occurs, there are three possible results:

- The radio message is received, and the sensor's microphone is activated. However, before the acoustic pulse arrives at the sensor, an unexpected environmental sound reaches the sensor, causing a short measurement.

- The radio message is received, and the sensor's microphone is activated. However, the acoustic pulse is deflected by an obstacle. Before the microphone deactivates due to a timeout, an echo of the acoustic pulse arrives, causing a long measurement.

- The radio message is received, and the acoustic pulse arrives at the microphone without incident. The measurement is accurate.

Equation (4.1) models this as a Bernoulli trial: the third case occurs with probability p, and one of the error cases occurs with probability $1-p$. h_{ij}^t is the random variable representing the outcome of this trial for measurement y_{ij}^t. Equation (4.1) leads directly to the probability distribution:

$$p(y_{ij}^t|s_i,\mathbf{e}_j^t) = p(h_{ij}^t=1)\frac{1}{Z_1}\exp\left[-\frac{(y_{ij}^t - \|s_i^x - \mathbf{e}_j^t\| - s_i^\theta)^2}{2\sigma^2}\right] + p(h_{ij}^t=0)\frac{1}{Z_2}, \quad (4.2)$$

where Z_1 and Z_2 are normalization constants. This changes the structure of equation (3.6), since each measurement y_{ij}^t now has an unknown explanatory variable h_{ij}^t. This new structure is straightforward to optimize using an Expectation Maximization (EM) algorithm [13]. Let $\theta = [\mathbf{x}^t]$. EM finds a maximizing θ repeatedly applying the following update step:

$$\begin{aligned}
\theta^{k+1} &= \arg\max_\theta \; E_{p(\mathbf{h}^t|\mathbf{y}^t,\theta^k)}\left[\log p(\mathbf{y}^t,\mathbf{h}^t|\theta,\mathbf{y}^{old}) + \log p(\theta|\mathbf{y}^{old})\right] \\
&= \arg\max_\theta \; \sum_{\mathbf{h}^t} p(\mathbf{h}^t|\mathbf{y}^t,\theta^k)\left[\log p(\mathbf{y}^t,\mathbf{h}^t|\theta,\mathbf{y}^{old}) + \log p(\theta|\mathbf{y}^{old})\right] \\
&= \arg\max_\theta \; \log p(\theta|\mathbf{y}^{old}) + \sum_{i,j}\sum_{h_{ij}^t} p(h_{ij}^t|y_{ij}^t,\theta^k)\log p(y_{ij}^t,h_{ij}^t|\theta) \qquad (4.3)
\end{aligned}$$

4.1.1 E-step

In the E-step, we must compute $p(h_{ij}^t|y_{ij}^t,\theta^k)$ for all i and j. Since $\log p(y_{ij}^t, h_{ij}^t = 0|\theta)$ is constant with respect to θ, the corresponding terms have no effect on the maximization (4.3). Consequently, we need only calculate:

$$\begin{aligned}
p(h_{ij}^t=1|y_{ij}^t,\theta^k) &= \frac{p(y_{ij}^t,h_{ij}^t=1|\theta^k)}{\sum_h p(y_{ij}^t,h|\theta^k)} \\
&= \frac{p(h_{ij}^t=1)p(y_{ij}^t|h_{ij}^t=1,\theta^k)}{\sum_h p(h)p(y_{ij}^t|h,\theta^k)}
\end{aligned}$$

This expression is easy to compute since it is in terms of the measurement model conditioned on h and the Bernoulli distribution $p(h)$:

$$p(h_{ij}^t = 1 | y_{ij}^t, \theta^k) = \frac{pN}{pN + (1-p)/Z_2}, \tag{4.4}$$

where

$$N = \frac{1}{Z_1} \exp\left[-\frac{(y_{ij}^t - \|s_i^x - \mathbf{e}_j^t\| - s_i^\theta)^2}{2\sigma^2} \right].$$

4.1.2 M-step

In the M-step, we optimize equation (4.3) to find a new estimate θ^{k+1}. Let $w_{ij}^{t,k} = p(h_{ij}^t = 1 | y_{ij}^t, \theta^k)$ as computed in the E-step by equation (4.4). Equation (4.3) reduces to:

$$\begin{aligned}
\theta^{k+1} &= \arg\max_\theta \log p(\theta | \mathbf{y}^{old}) + \sum_{i,j} w_{ij}^{t,k} \log p(y_{ij}^t, h_{ij}^t = 1 | \theta) \\
&= \arg\min_{\mathbf{x}^t} (\mathbf{x}^t - \mu)^T \Omega_{\mathbf{x}} (\mathbf{x}^t - \mu) + \frac{1}{\sigma^2} \sum_{i,j} w_{ij}^{t,k} (\|\mathbf{s}_i^x - \mathbf{e}_j^t\| + s_i^\theta - y_{ij}^t)^2.
\end{aligned} \tag{4.5}$$

The M-step (4.5) is therefore a re-weighting of the original LaSLAT optimization problem (equation (3.6)). As a result, it can be performed analogously using Newton-Raphson (see Appendix B). Note that the weight w_{ij}^t of a measurement y_{ij}^t after the final EM iteration corresponds to the probability that the measurement is accurate given the current parameter estimates. Thus, accurate measurements are assigned weights close to one, and highly inaccurate measurements are assigned weights close to zero. This accomplishes the goal of rejecting outlying measurements.

4.1.3 Outlier rejection summary

When the update step (4.3) is performed repeatedly, the θ^k's converge to $\mathbf{x}^{t*}$, an optimal estimate of sensor parameters and event locations that detects and ignores outlying measurements. This improves LaSLAT's performance in the presence of

environmental ambient sounds or echoes due to physical obstacles.

4.2 Specifying Mobile Dynamics

In section 3.4, we assumed that the mobile moved arbitrarily. However, in some cases (for instance when tracking targets that move continuously and trigger events frequently), one may improve performance by explicitly requiring that successive events occur close to each other. In this section, we present a technique for adding such a constraint to LaSLAT.

As shown in section 3.4, the prior distribution for LaSLAT $p(\mathbf{x}^{t+1}|\mathbf{y}^t, \mathbf{y}^{old})$ is computed according to:

$$p(\mathbf{x}^{t+1}|\mathbf{y}^{old}, \mathbf{y}^t) = p(\mathbf{e}^{t+1})q(\mathbf{s}|\mathbf{y}^{old}, \mathbf{y}^t). \tag{4.6}$$

Until now, we have defined $p(\mathbf{e}^{t+1})$ to be a Gaussian with high covariance. Let $\mathbf{e}_j^{t+1} = [\mathbf{e}_j^x \ \mathbf{e}_j^v \ \mathbf{e}_j^a]$, where $\mathbf{e}_j^v$ and $\mathbf{e}_j^a$ represent the velocity and acceleration of the mobile at the time of event j, and define $\mathbf{e}_0^{t+1}$ to be the estimated parameters of the event immediately preceding batch $t+1$. Then we can express $p(\mathbf{e}^{t+1})$ as follows:

$$p(\mathbf{e}^{t+1}) = \prod_{j=1}^{n} p(\mathbf{e}_j^{t+1}|\mathbf{e}_{j-1}^{t+1}). \tag{4.7}$$

We enforce smoothness by setting $p(\mathbf{e}_j^{t+1}|\mathbf{e}_{j-1}^{t+1})$ to a Gaussian with mean Ae_{j-1}^{t+1} and covariance $\sigma_d^2 I$. A is a matrix that expresses the expected physical motion of the mobile. We assume that events occur at a constant rate, and ignore quadratic terms, allowing us to use the matrix:

$$A = \begin{bmatrix} I & I & 0 \\ 0 & I & I \\ 0 & 0 & I \end{bmatrix}$$

as the dynamics matrix. This matrix has the advantage of being computationally straightforward while having the desired effect of smoothing the event positions. σ_d is a tunable constant that allows us to vary the effect of the smoothness prior versus the effect of measurement data on the events' position estimates.

Thus, we can rewrite equation (4.7) as:

$$p(\mathbf{e}^{t+1}) \propto \exp\left[-\frac{1}{2\sigma_d^2}\|\mathbf{e}_j^t - A\mathbf{e}_{j-1}^t\|^2\right].$$

This term can be rewritten in the form:

$$p(\mathbf{e}^{t+1}) \propto \exp\left[-\frac{1}{2}(\mathbf{e}^t - \mu_d)^T B(\mathbf{e}^t - \mu_d)\right],$$

where the block tridiagonal matrix B and vector μ_d are constants expressible in terms of A and $\mathbf{e}_0^{t+1}$:

$$B = \frac{1}{\sigma_d^2}\begin{bmatrix} I + A^TA & -A^T & & \\ -A & \ddots & \ddots & \\ & \ddots & I + A^TA & -A^T \\ & & -A & I \end{bmatrix} \qquad \mu_d = B^{-1}\begin{bmatrix} -A \\ \mathbf{0} \\ \vdots \end{bmatrix}\mathbf{e}_0^{t+1}.$$

B and vector μ_d can then incorporated into equation (3.9) as follows:

$$E\left[\mathbf{x}^{t+1}|\mathbf{y}^{old}\right] = \begin{bmatrix} E\left[\mathbf{s}|\mathbf{y}^{old}\right] \\ \mu_d \end{bmatrix}$$
$$\mathrm{Cov}^{-1}\left[\mathbf{x}^{t+1}|\mathbf{y}^{old}\right] = \begin{bmatrix} \mathrm{Cov}^{-1}\left[\mathbf{s}|\mathbf{y}^{old}\right] & \mathbf{0} \\ \mathbf{0} & B \end{bmatrix}.$$

As we demonstrate in chapter 6, this smoothness constraint can noticeably improve performance on occasional poorly measured events.

Unfortunately, the smoothness constraint complicates the marginalization step shown in equation (3.8), since the inverted matrix $\Omega_{\mathbf{e}^t}^{-1}$ is no longer diagonal. Conse-

quently, the smoothness prior is recommended for use only when LaSLAT computations are being performed centrally.

Chapter 5

Implementation

Our current implementation sends measurement batches to a central computer; however, we show here that LaSLAT can be feasibly distributed if desired. We also explore running time bounds for both a distributed implementation and a centralized implementation. Finally, we discuss the trade-off between centralized and distributed processing in LaSLAT.

5.1 Definitions

In order to quantify LaSLAT's performance, several definitions are required. Let n_{local} be the expected number of sensors within a one-hop *neighborhood* of a sensor s. The one-hop neighborhood can be thought of as a circle around s whose radius is two times the sensor's maximum sensing range. Thus, the one-hop neighborhood contains all sensors that can sense an event in common with s. We assume that s can communicate directly with all of the sensors in this one-hop neighborhood.

Furthermore, we define n_{events} to be the number of events observed by s in a single batch of measurements. n_{events} can be held constant by varying the amount of time between batches.

Note that n_{local} and n_{events} are constants fixed at deployment by the network designer. This means that asymptotic bounds in n_{local} and n_{events} are in some sense constant bounds, since they enable sensors to be provisioned with an amount of

memory and processing power that is guaranteed sufficient no matter how many events the network observes.

5.2 Graph Locality

The Gaussian prior $p(\mathbf{x}^t|\mathbf{y}^{old})$ is completely summarized by a vector of means $\mathbf{x}^{t*}$ and an inverse covariance Ω.

The symmetric inverse covariance matrix $\Omega = \left[\begin{smallmatrix} \Omega_s & \Omega_{se} \\ \Omega_{es} & \Omega_e \end{smallmatrix}\right]$ defines an undirected graph between sensors and events. Two vertices in this graph are connected if their corresponding block in Ω is non-zero. We say Ω has local connectivity if the corresponding graph only connects sensors that are within one hop of each other and connects events only to the sensors that measured the event. As we will show, in LaSLAT Ω always has local connectivity.

5.3 Distributability of the Prior

The mean vector is straightforward to distribute: each sensor simply stores its own mean, those of its one hop neighbors, and those of all nearby events. Thus, storage for means requires at most $O(n_{local} + n_{events})$ per sensor.

Ω requires more careful consideration, but is also distributable. If Ω has local connectivity, each row of Ω corresponding to a sensor has about $n_{local}+n_{events}$ non-zero entries. Each row corresponding to an event has less than n_{local} non-zero elements, since only sensors within the neighborhood of the event obtain measurements to it. Locally connected matrices are therefore easy to distribute. Each sensor stores its own rows in Ω. Event rows are delegated randomly to a sensor for storage, leaving sensor storing about n_{events}/n_{local} event rows. The amount of data stored by each sensor is consequently $O(n_{local} + n_{events})$. If the computation is performed centrally, then the central computer must store this same amount of data per sensor in the network.

5.4 Performing Computations

LaSLAT consists of two significant computational steps: incorporating measurements and applying the dynamics model. As formulated in this paper, these steps can be performed using only local communication between sensors that have witnessed a common event. Furthermore, these operations retain local connectivity in the prior inverse covariance matrix Ω.

5.4.1 Measurement incorporation

The principal operation involved in incorporating new measurements is a Newton-Raphson iteration, shown in equation (B.3). Each Newton-Raphson iteration has two parts. First a matrix and vector must be computed based on equation (B.3). Then, a least squares optimization must be performed.

The matrix and vector can be computed locally, since they require only the parts of Ω and $\mathbf{x}^{t*}$ that are found locally and the measurements to any local events. The communication and time costs for each are proportional to $n_{events} * n_{local}$, which is the minimum time required for each sensor to broadcast new measurements to neighboring sensors and receive their measurements in return. It is similarly the minimum time for all sensors to transmit their measurements to a central computer.

Once the matrix and vector are computed, the least squares optimization can be performed using Gauss-Seidel iterations [2]. Gauss-Seidel is guaranteed to converge when solving symmetric positive definite systems of equations like those found in LaSLAT. Each iteration of Gauss-Seidel requires $O(n_{local})$ computation and $O(1)$ radio messages per sensor and event when distributed, or $O(n * n_{local})$ computation when centralized, where n is the total number of sensors and events. In practice, we find that Gauss-Seidel converges in a few tens of iterations for our systems, since LaSLAT does not require high precision convergence.

Gauss-Seidel requires that sensors perform their processing in a consistent order, which diminishes the potential parallelization of the least squares computation. However, with a constant bound on the number of Gauss-Seidel iterations, the total time

required for each distributed Newton-Raphson iteration is $O(n * n_{local})$. Note that this is not a tight upper bound: as the sensor parameters begin to converge, many parameters will not need to be updated every batch. This increases the amount of parallelism that can be exploited, allowing the total running time to approach $O(n_{local})$, the amount of time required to simply locate the newest events. See [2] for more details and an in-depth description of Gauss-Seidel iterations.

5.4.2 Event marginalization

Marginalization is performed using equation (3.8):

$$\mathrm{Cov}^{-1}\left[\mathbf{s}|\mathbf{y}^{old}\right] = \Omega_{\mathbf{s}} - \Omega_{\mathbf{s}\mathbf{e}^t}\Omega_{\mathbf{e}^t}^{-1}\Omega_{\mathbf{e}^t\mathbf{s}}.$$

It is distributable because each sensor row is updated only on behalf of local events. Since Ω_{se^t} and Ω_{e^ts} are sparse and Ω_{e^t} is block diagonal, the total time required is only $O(n_{local} * n_{events})$ per sensor. All the computations can occur in parallel.

Unfortunately, the smooth dynamics extension developed in section 4.2 produces an Ω_{e^t} that is block tridiagonal and has a dense matrix inverse in general. As a result, the smooth dynamics extension ruins local connectivity. Thus, smooth dynamics may only be employed when LaSLAT computations are performed at a central computer.

5.4.3 Preservation of local connectivity

It remains to be shown that the LaSLAT computations retain local connectivity in Ω when the smooth dynamics extension is not employed. This is easily confirmed by induction on Ω_s. The initial prior has $\Omega_s = \sigma_0 I$, which is diagonal and therefore locally connected. As we show in the appendix, the measurement incorporation and Gaussian approximation steps do not change the connectivity of Ω. The connectivity of Ω_s only changes when events are marginalized out of the Gaussian prediction by equation (3.8). It can be verified, however, that the $-\Omega_{se}\Omega_e^{-1}\Omega_{es}$ term added to Ω_s only affects elements of Ω_s whose corresponding sensors observed an event in common during the most recent batch. As a result, Ω_s retains local connectivity

during LaSLAT operations.

5.4.4 LaSLAT convergence

Each LaSLAT batch requires at least one Newton-Raphson optimization, which consists of several iterations. However, these iterations need not continue until the solution is fully converged. In fact, if each LaSLAT batch performs only one Newton-Raphson iteration, then the resulting algorithm is almost precisely the Extended Kalman Filter (EKF) form of SLAT. As we show in chapter 6, additional Newton-Raphson iterations substantially improve performance. However, little performance is lost if the number of iterations is bounded at a small constant. In our experiments, Newton-Raphson often converged in less than ten iterations. In two dimensions, frequently as few as three or four were required for convergence. Thus, the number of iterations may be treated as a constant factor.

When using measurement outlier rejection (section 4.1), even less Newton-Raphson iterations are required per optimization, since the EM optimization runs Newton-Raphson repeatedly. The EM optimization remains distributable, since the only additional processing step is the computation of measurement weights.

5.5 Centralized vs. Distributed Implementation

As we have demonstrated, LaSLAT is amenable to both centralized and distributed implementation. In a centralized implementation, the network must transport all range observations to a central computer, which performs LaSLAT computations and optionally returns position estimates to the network. In a distributed implementation, the LaSLAT computations are performed in-network.

Consider a small network in a controlled environment such as a building. The experiments in chapter 6 are all representative of such a scenario. In this case, the central computer can be positioned within a single radio hop of all or nearly all sensors. In this case, the best performance is obtained by transferring all the measurements to the central computer. Centralization in this case reduces the radio bandwidth,

memory, and processing requirements on the sensors, decreasing the hardware cost of the network. Currently, sensors remain somewhat expensive and radio bandwidth remains a scarce commodity, so centralization is very desirable.

As the network grows larger distributed computation begins to look compelling. When multiple hops are required to transmit data to the central computer, the energy and bandwidth cost of centralization increases, particularly for nodes near the central computer. In this case, distributing LaSLAT may save power by keeping computation and communication local. Furthermore, in some scenarios (such as battlefield applications), it may be impossible to provision the network with a central computer, in which case distributed computation is necessary.

In many cases, however, centralized computation is straightforward and desirable even in large networks. For instance, in large indoor networks, it is possible to use pre-existing high-speed wireless or wired networks to transmit data rapidly to a central computer, which need not be co-located with the network. In these environments, the network forms a hierarchy in which sensors transmit directly to a base station, which in turn transmits to the central computer over a high-speed link. This architecture has the advantage that the individual sensors need relatively few capabilities. The tracking sensor required for LaSLAT, for example, could be little more than a radio, an ultrasound receiver, and a tone detector. Such a limited sensor is likely to be cheaper and require less power than a sensor with enough processing power, memory, and radio bandwidth to perform distributed computations. The savings may be used to provision a smaller number of computation nodes or base stations with a high speed link or fast processor and a more substantial power supply. Furthermore, the hierarchical network is better suited to delivering tracking data from the network to end users, since end users are more likely to be connected to a high speed network than to the sensor network's radio system.

LaSLAT has been designed to perform well with either a centralized or a distributed implementation. The results in this thesis were all computed using the centralized variant. The distributed variant performs exactly the same computations, and can therefore be expected to achieve the same results.

Chapter 6

Results

Our experiments use the Cricket ranging system [1] (see figure 6-1). Sensor Crickets are placed in an area, and one Cricket is attached to a mobile. The mobile Cricket periodically emits an event (a radio and ultra-sound pulse) at a rate between one and three per second. At each sensor, the difference in arrival time of these two signals is proportional to the distance between the sensor and mobile. The crickets can therefore estimate ranges from these arrival times. No range measurements between the sensor Crickets are collected. The measurements are transmitted to a desktop machine, which processes them in batches using LaSLAT, which we implemented in Java. The ultra-sound sensor on a Cricket occupies a 1 cm by 2 cm area on the circuit

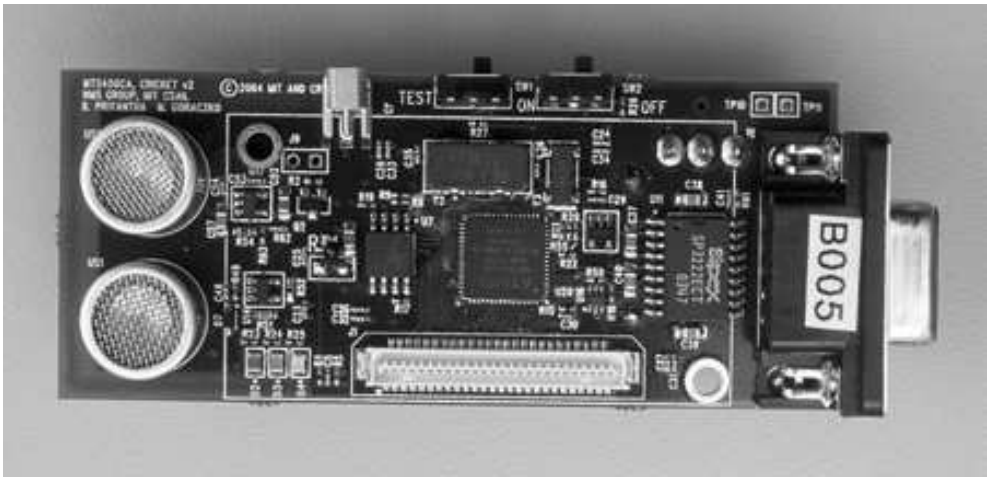

Figure 6-1: The Cricket sensor node used in our experiments.

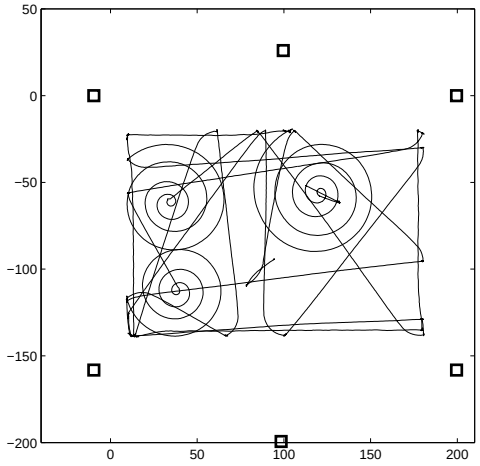

Figure 6-2: Small network setup. Six sensors (squares) are arranged around a rectangular enclosure. A camera captured the ground truth trajectory of the ROOMBA. The ROOMBA followed the trajectory depicted.

board, so it difficult to estimate the ground truth location of a Cricket beyond that accuracy.

In all of the experiments, LaSLAT recovered sensor locations in a relative coordinate system that can be aligned to a global coordinate system using a single rigid transformation (a rotation and translation). In order to compare LaSLAT's results to measured ground truth, we computed the necessary rigid transformation using the method described in [15].

6.1 ROOMBA Experiment

Our first experiment used the same setup as [22]. Six sensor crickets were placed around a rectangular enclosure 2.1 meters by 1.6 meters. A ROOMBA robotic vacuum cleaner with the mobile Cricket attached was allowed to move freely within the enclosure, generating about 250 events. See Figure 6-2.

Most events were measured by all 6 sensors. An initial localization guess was obtained from radio connectivity information using the initialization routine of [29]. The resulting average localization error in this initial guess was 66 cm. This initial guess

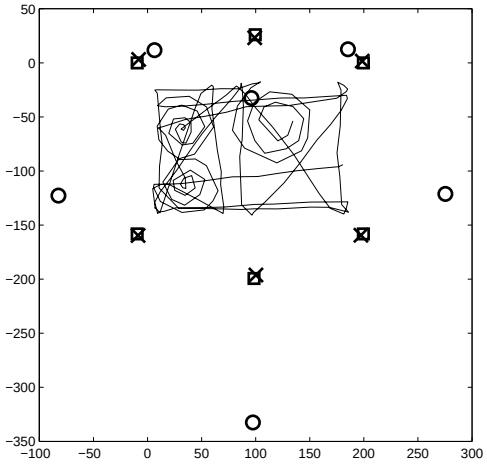

Figure 6-3: Recovered trajectory and sensor positions. Circles are guesses of initial sensor locations obtained from radio connectivity (appendix A). LaSLAT processed measurements in batches of 30 events, and recovered sensor locations depicted by crosses. The trajectory is also correctly recovered. LaSLAT improves considerably on the initial localization guess obtained from connectivity. After a global rotation and translation, the average localization error for the sensors was 1.8 cm, which is within the error tolerance of the ground truth.

was used as a prior and an initial iterate for LaSLAT. LaSLAT incorporated range measurements in batches of 30. Each mode finding operation required an average of only 2.8 Newton-Raphson steps. Figure 6-3 shows the estimated sensor localizations and trajectory. Since the output had an arbitrary rotation and translation, it was rigidly aligned to fit the rotation and origin of the ground truth using the algorithm described in [15]. The final localization error was 1.8 cm, averaged over the sensor nodes. This is within the error tolerance of the ground truth.

6.2 Two Dimensional Experiments

In the remainder of our two dimensional experiments, sensors were placed facing upwards on the floor of the coverage area. The mobile cricket was manually moved through the network. It was suspended at a constant height of about 190 centimeters, and oriented facing the floor. Due to the conical propagation of ultrasound from the mobile, this approximated a radial spread of sound in two dimensions. Range

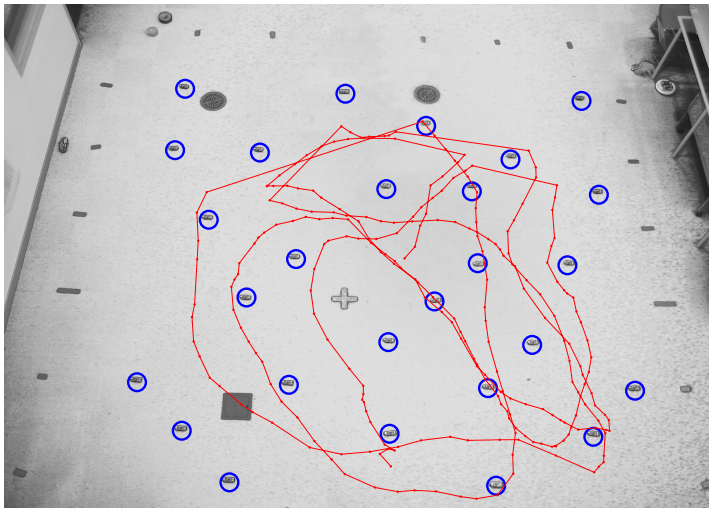

Figure 6-4: Sensor locations and mobile trajectory for a medium size network. Circles outline each of the 27 sensor nodes. Markers on the trajectory depict the location of events. 250 of the 1500 events are shown, with consecutive events connected by a line. The mobile was offset from the ground plane and could pass over nodes. To generate this figure, a homography that accounts for the camera transformation was used to project real-world coordinates to image coordinates.

measurements gathered in these experiments were adjusted in a pre-processing phase to remove the effect of relative height.

6.2.1 27 node experiment

Our second experiment involved a larger network with 27 Cricket sensor nodes deployed in a 7 m by 7 m room. Whereas in the previous experiment the nodes were on the perimeter of the ROOMBA mobile's trajectory, in this experiment, we manually pushed a mobile through the network, generating about 1500 events. Figure 6-4 shows the location of the sensors and part of the trajectory of the mobile projected on a top view picture of the setup.

Each event was heard by about 10 sensors. Figures 6-5(a)-(d) show localization and tracking output as event batches are processed, along with the ground truth and estimated mobile trajectories for that batch. Error ellipses show unit standard

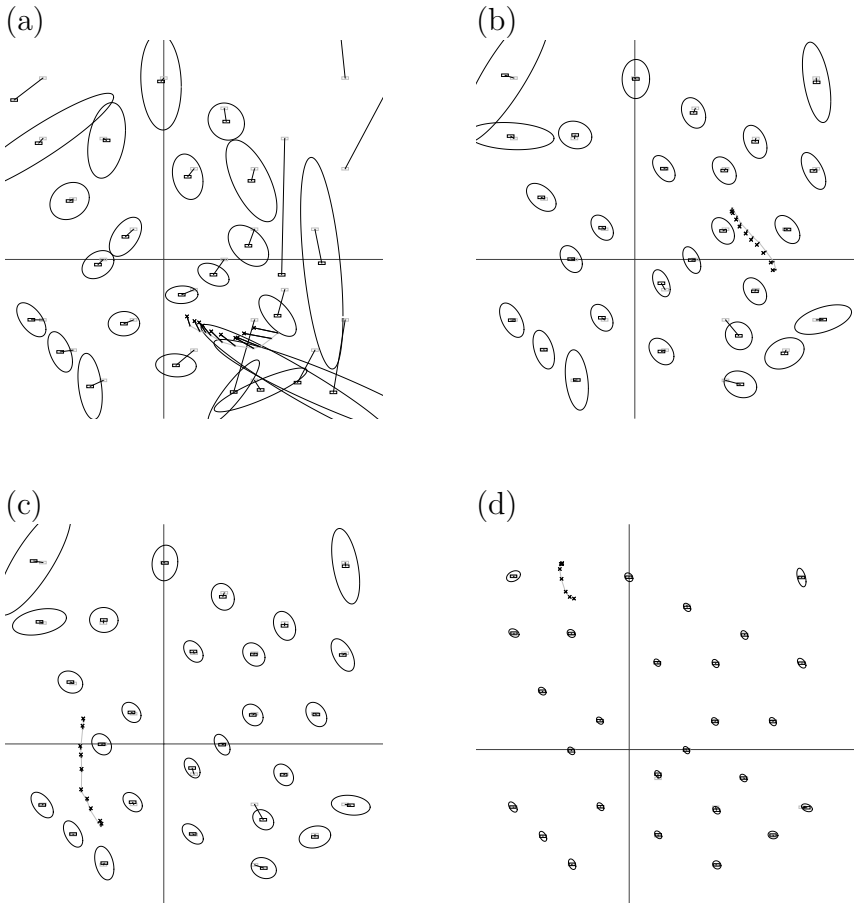

Figure 6-5: The output of LaSLAT after incorporating (a) 50, (b) 120, (c) 160, and (d) 1510 events. The batch size was 10. Recovered mobile trajectory (crosses) and ground truth mobile trajectory (solid line) for the latest batch are connected by a line to show correspondences. Estimated mobile locations (dark rectangles) and the ground truth mobile locations (light rectangles) are also connected with a line to show correspondence. Error ellipses shrink as more data becomes available. Between events 120 and 160 (sub-figures (b) and (c)), the mobile swept around the bottom of the network, and the error ellipses and localization error diminished for those sensors. Tracking improved as sensors became better localized.

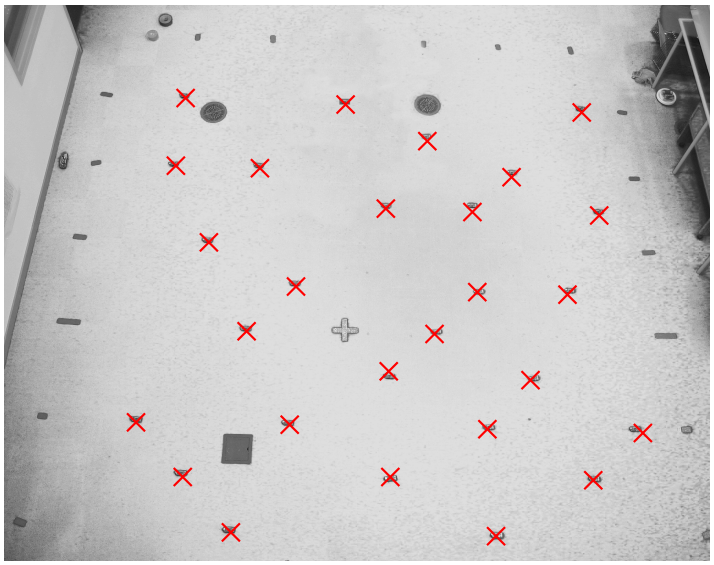

Figure 6-6: Final LaSLAT localization result, with batch size of 10. Crosses show estimated sensor locations. These are correctly estimated to fall on the corresponding sensor. Average localization is 1.9 cm.

deviation contours for each sensor node. Nodes have high uncertainty at early stages, but when the mobile passes near a node, its error ellipse shrinks appropriately. In this experiment, a measurement bias of about 23 cm was computed for each sensor node. Figure 6-6 shows the final localization of the nodes, reprojected on the picture of the setup. This experiment used batches of 10 measurements and produced a final localization error of 1.9 cm. Since the ground truth is only accurate to a few centimeters, localization performance is best examined visually via Figure 6-6.

We compared LaSLAT using varying batch sizes to the Extended Kalman Filter (EKF), which is identical to LaSLAT limited to one Newton-Raphson iteration. Figure 6-7 shows average localization errors as events were processed. The EKF performs best with no batching (batch size = 1). LaSLAT converges faster and also exhibits lower steady state localization error. As batch sizes are increased, so does the rate of convergence of LaSLAT. Batching also improves the final localization error. LaSLAT, with batch sizes of 1, 10 and 40, produced final localization errors of 3 cm, 1.9 cm, and 1.6 cm respectively. On average, LaSLAT took 3 Newton-Raphson iterations

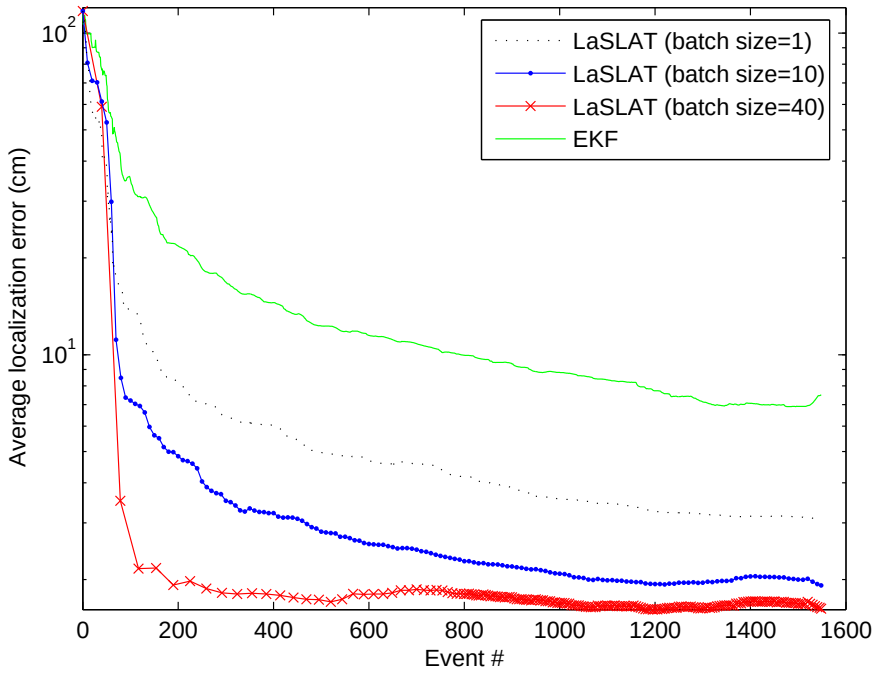

Figure 6-7: Localization error as a function of the number of events observed for EKF and various batch sizes for LaSLAT. LaSLAT converges more quickly and attains a lower steady state error than the EKF. Furthermore, larger batch sizes improve the convergence rate and the steady state error of LaSLAT.

Figure 6-8: A sparser 2D sensor network with 49 nodes in a 10m by 17m environment.

to incorporate each batch. The EKF's final localization error was 7.5 cm, which is outside the error tolerance for the ground truth.

6.2.2 49 node experiment

Figure 6-9 shows localization results on a larger network (49 sensors) deployed over a larger area (10 m by 17 m). For comparison, the experiment setup is shown in figure 6-8. With about 0.3 sensors per square meter, this network is about half as dense as the one shown in Figure 6-4, which had about 0.5 sensors per square meter. As a result, on average only 5 sensors heard each event, and the localization error was about 7.5 cm. The algorithm also determined a measurement biases of about 20 cm for all nodes. For all batch sizes, the EKF produced an average localization error of about 80 cm, showing that the improvement due to Laplace's method can be very

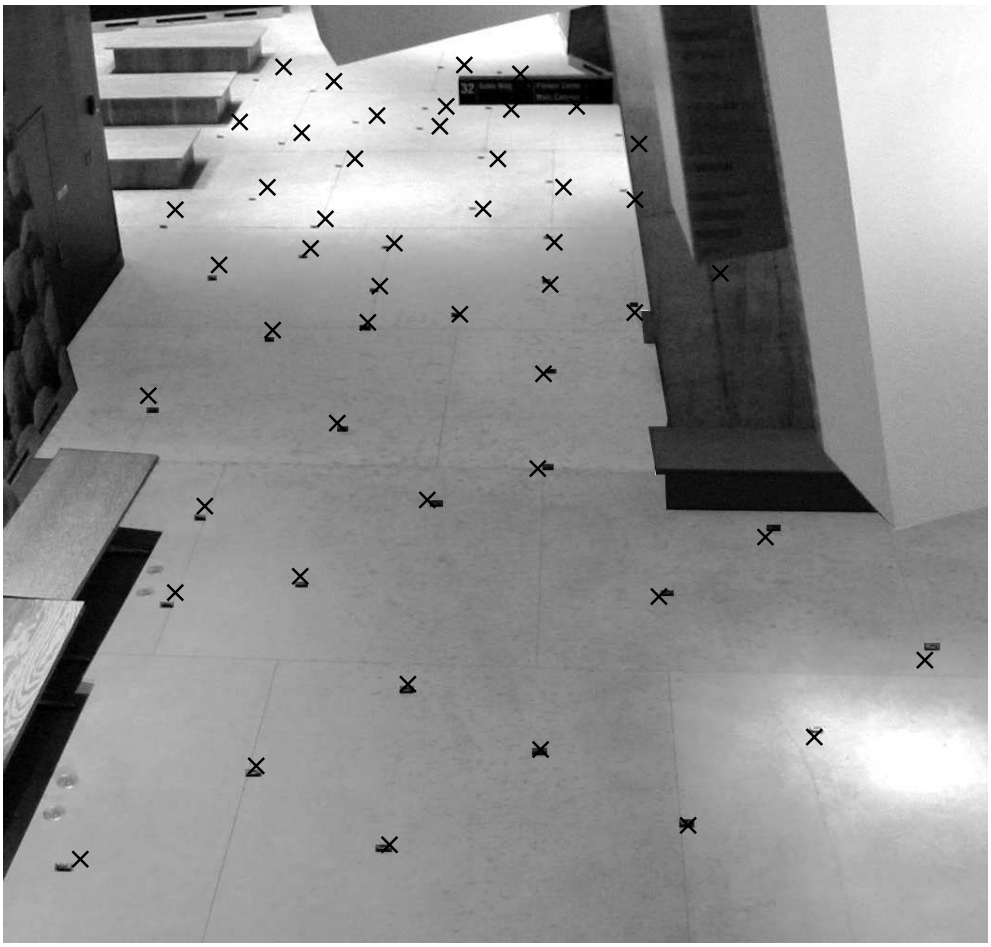

Figure 6-9: LaSLAT localization result on a sparser sensor network with 49 nodes in a 10m by 17m environment. Crosses indicate the recovered sensor locations, projected onto the image. The average localization error was 7.5 cm.

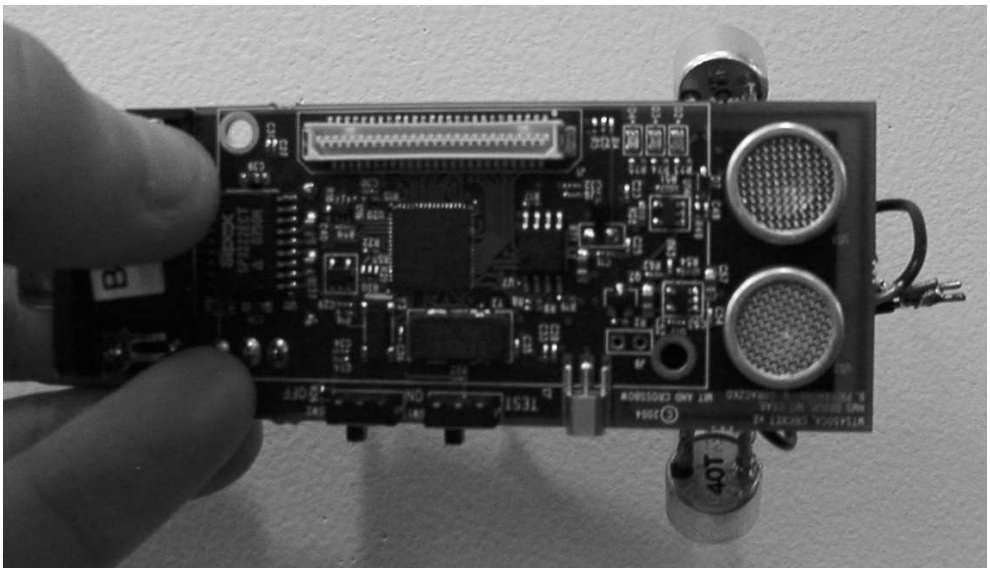

Figure 6-10: The omnidirectional cricket used in three dimensional experiments.

significant.

6.3 Three Dimensional Experiments

It is possible to use LaSLAT to localize and track sensors and mobiles in a three dimensional environment. We performed two experiments designed to test LaSLAT in reasonable three dimensional deployments.

For these deployments, crickets were placed on the floor as in the two dimensional experiments. In addition, crickets were attached to walls using Velcro. A few sensors were also placed atop furniture in the area. We also modified the mobile cricket by attaching two additional ultrasound transducers. These transducers more closely simulate an omnidirectional acoustic pulse than the conic emanation of the standard cricket transducer. See figure 6-10 for a picture of this modified cricket.

The three dimensional environments present two obstacles that are not present in the two dimensional experiments. First, the furniture in the area drastically increases the effect of echoes on range measurement quality. These echoes are suppressed using the measurement outlier rejection scheme presented in section 4.1. Second, the initialization algorithm we used in two dimensions [29] does not provide for localization

Figure 6-11: Smaller environment used for 3D localization experiments.

in 3D. However, we developed a simple extension (see appendix A) to allow us to initialize LaSLAT satisfactorily.

6.3.1 40 node experiment

In our first three dimensional experiment, we placed 40 crickets on the floor and walls of a 4 x 6 meter room, which contained all of its normal furniture: tables, chairs, printers, and a refrigerator. This mobile was carried by hand through the room and moved completely arbitrarily, including changes in speed, loops, and twists. Each event was observed by on average 17 sensors. The network is shown in figure 6-11. Localization results are plotted in figure 6-12. LaSLAT localized sensors to within 7 cm while successfully tracking the path of the mobile in 3D. Much of this error is accounted for by the difficulty of measuring ground truth in this environment.

The best 3D results were obtained using a relatively large batch size of 250 events. Smaller batch sizes lose too much information in the Gaussian approximation used to preserve state between batches, causing LaSLAT to converge slowly. In 3D, the optimization (3.6) has a more complicated structure due to the additional unknown parameters, which larger batches help clarify. Thus, a large initial batch helps LaSLAT

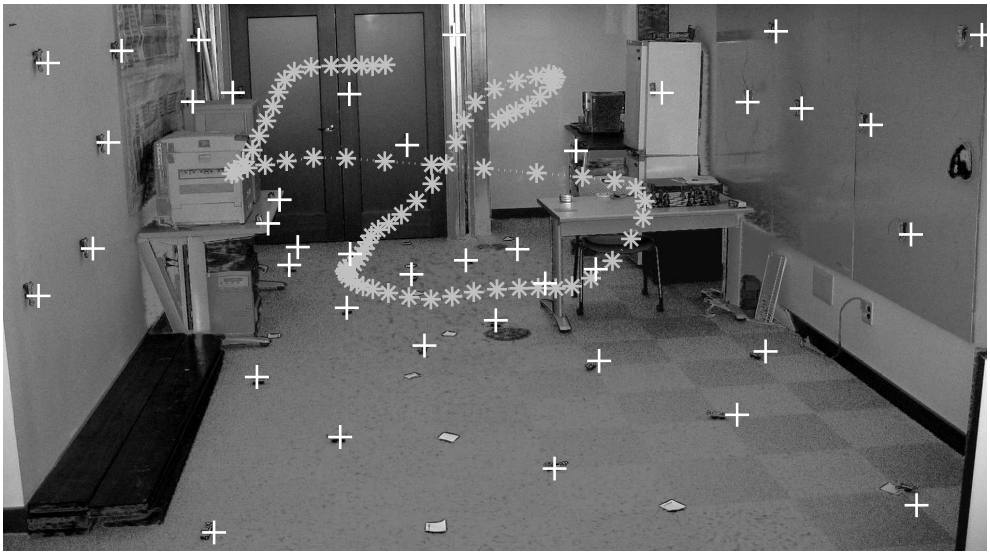

Figure 6-12: LaSLAT results plotted on a picture of the network. Plus signs indicate the estimated 3D positions of the sensors. A small portion of the mobile trajectory (about 80 events) is plotted as asterisks connected by a dotted line. LaSLAT localized sensors to within 7 cm.

make rapid progress towards an accurate estimate. After the first few batches, the batch size may be decreased to speed computation, since the later batches need only refine the already fairly high quality estimate.

The 3D environment shown in figure 6-11 is considerably less sanguine to ultrasound time-of-flight ranging than the two dimensional environments. Most events caused several very erroneous measurements. Figure 6-13 shows a partial screen capture from our Java-based LaSLAT implementation. The dark line indicates a portion of the trajectory recovered by LaSLAT. The light line is recovered using multilateration on the ground truth sensor positions and the raw measurement data, and therefore serves as a baseline tracking algorithm. LaSLAT's improved tracking performance is due to the combination of sensor calibration (section 3.2), measurement outlier rejection (section 4.1), and smooth dynamics (section 4.2). Some of these techniques can be and have been adapted to pure tracking applications. However, LaSLAT is able to perform the same high fidelity tracking while simultaneously localizing and calibrating the network.

We ran LaSLAT both with and without smooth dynamics and outlier rejection to

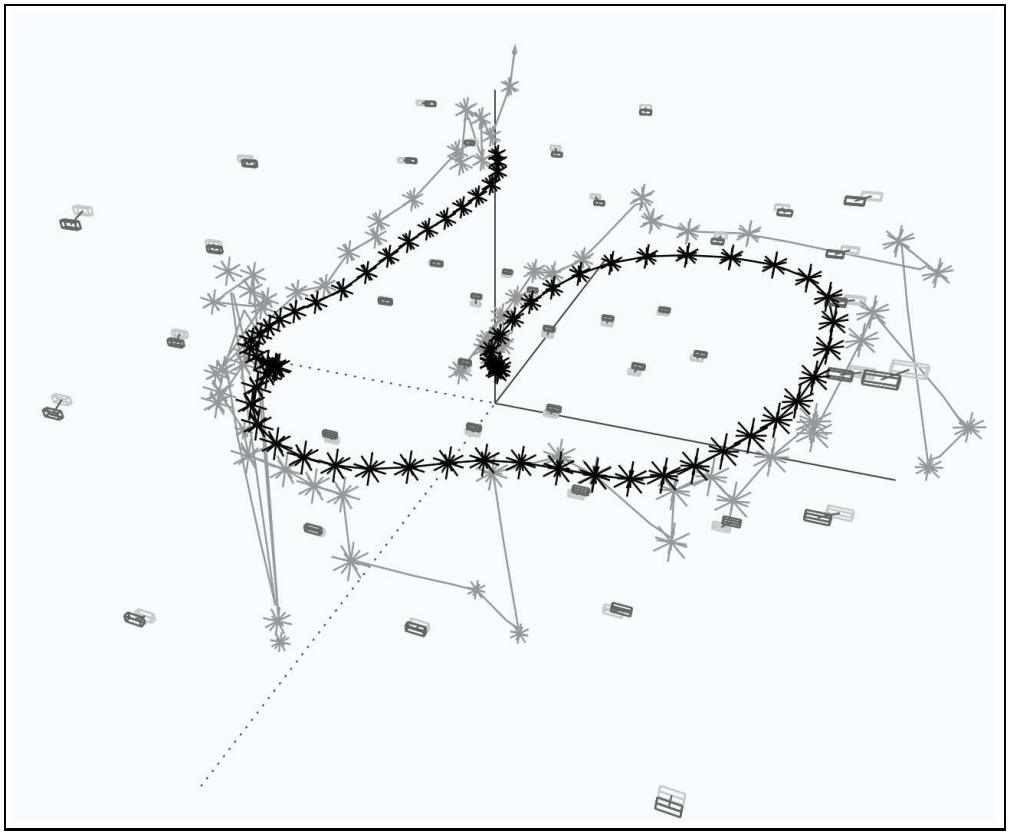

Figure 6-13: Sample LaSLAT 3D tracking results. The black line shows the mobile trajectory recovered by LaSLAT on the 40 node 3D topology. The gray line shows the mobile trajectory recovered using multilateration on raw measurements and ground truth sensor positions. Stars represent the position of the mobile at the time of an event. Approximately three events occurred every second. Note that LaSLAT very noticeably outperforms the naive tracking algorithm, even though it does not know the sensor positions *a priori*.

observe the techniques' impacts on localization accuracy. The results for the 40-node 3D experiment are graphed in figure 6-14. Note that LaSLAT with smooth dynamics and outlier rejection performs best, with a final accuracy of less than 7 centimeters.

Smooth dynamics makes relatively little difference to sensor localization, as its final average position error is 8 centimeters. This is accounted for by the fact that smooth dynamics primarily affects events that deviate from their neighbors. Since most events tend to be accurately placed without requiring smooth mobile dynamics, correcting the few deviants events has only a small effect on the sensor position estimates.

However, measurement outlier rejection makes a very substantial difference on this data set. Without outlier rejection, the final localization estimate has a mean error of 31 centimeters. This suggests that the outlier rejection technique presented in section 4.1 is effective at detecting and de-emphasizing errant measurements.

For reference, we also plot the performance of the Extended Kalman Filter (EKF) on this 40-node experiment. The final EKF estimate has on average over half a meter of localization error.

6.3.2 55 node experiment

In our second 3D experiment, we used 55 sensors to cover the floor and walls of a 7 x 10 meter room. Since the sensors were packed less densely, on average only 11 sensors witnessed each event. A portion of the network is shown in figure 6-15, and the results are plotted in figure 6-16. LaSLAT localized the sensors to within 7 cm, consistent with the previous experiment.

This environment was better suited to the Cricket ultrasound ranging system since it contained few obstacles. Consequently, sensors reported fewer bad measurements. Nevertheless, enough poor measurements were taken that outlier rejection improved performance noticeably.

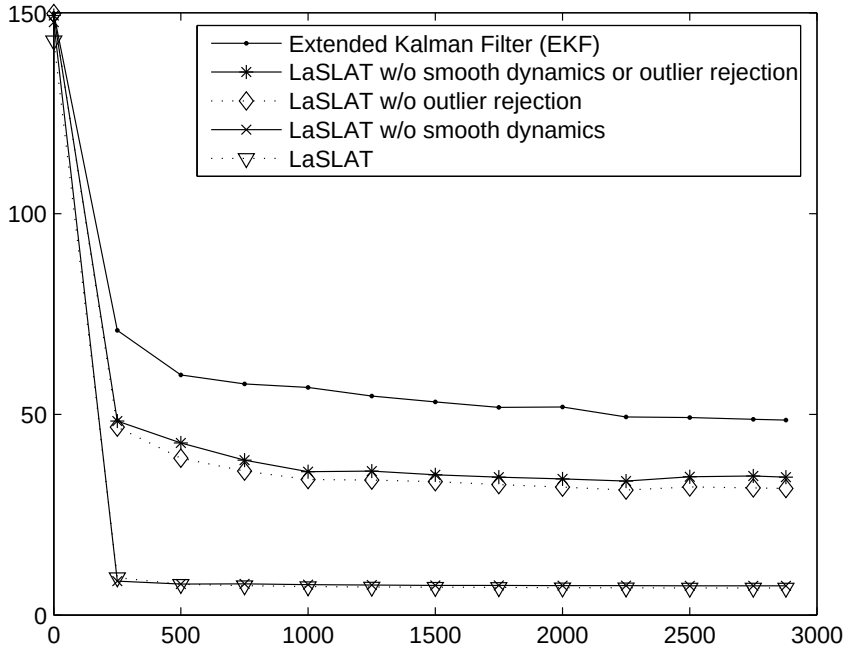

Figure 6-14: Performance impact of dynamics and outlier rejection on a 3D dataset. Note that all four variants of LaSLAT outperform the Extended Kalman Filter (EKF) implementation.

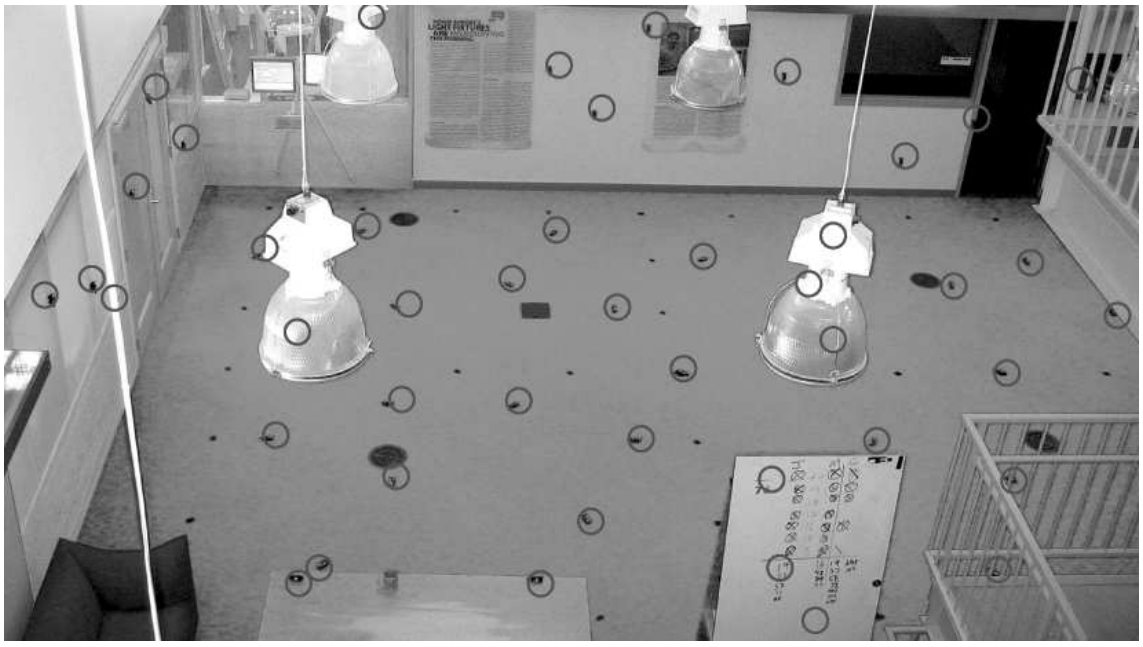

Figure 6-15: Part of the 55 node network for 3D localization experiments. Sensors are circled in blue.

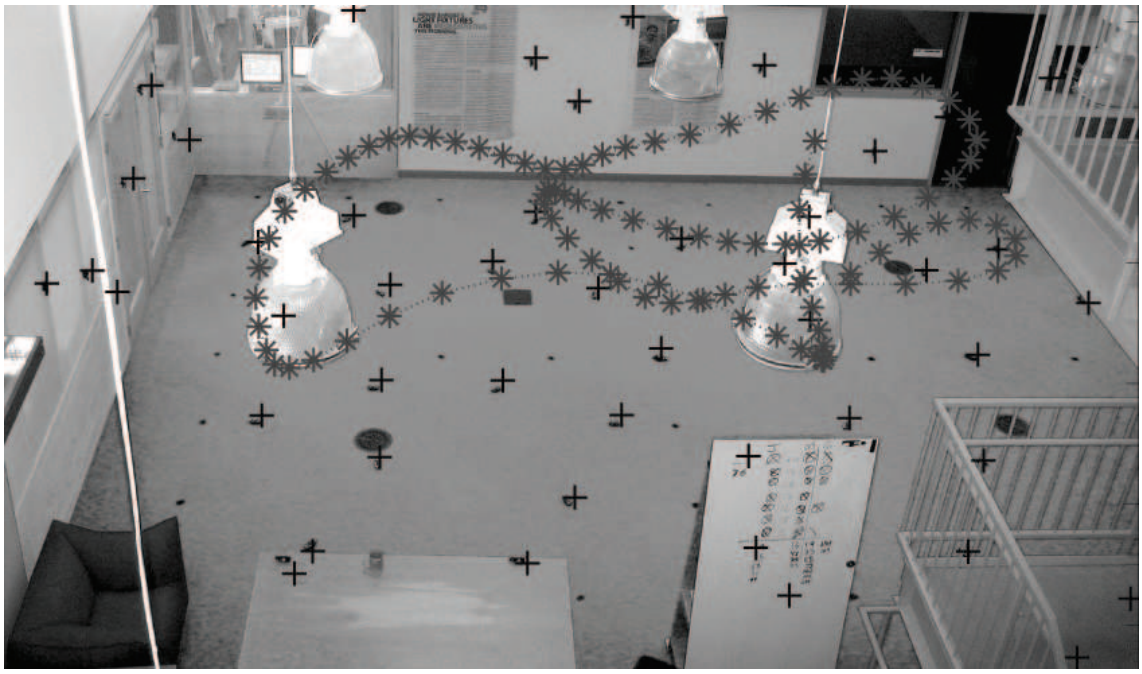

Figure 6-16: Results for the 55 node 3D experiment. Estimated positions are plotted with crosses, and a portion of the recovered mobile trajectory is plotted using asterisks and a dotted line. LaSLAT localized this network with a mean error of 7 centimeters.

Chapter 7

Future Work

In addition to the features developed in this thesis, LaSLAT raises several interesting possibilities for future work.

7.1 Hallway Alignment

As we demonstrated in chapter 6, LaSLAT performs very well in networks that have a reasonably convex shape. However, its performance diminishes when the network is divided into convex areas separated by narrow connections: for instance, two rooms separated by a narrow hallway. In these topologies, LaSLAT accurately localizes sensors in the rooms with respect to other sensors in the same room. However, it cannot always orient the rooms as a whole with respect to each other. It appears that the measurements gathered from events in and around the hallway area typically fail to unambiguously resolve the alignment. Augmenting LaSLAT using a coordinate system stitching approach similar to [7,21,22,24,31] could help resolve this ambiguity.

7.2 Stationary Targets

In our experiments, the mobile moved continuously for the majority of the experiment. However, real mobiles can be expected to remain stationary, potentially for lengthy periods of time. After LaSLAT has been running for some time, this poses no problem.

However, during the first few batches of LaSLAT difficulties can arise. A stationary mobile causes a large number of highly similar events. This results in a batch of events that resolves few ambiguities in the sensor positions. This leads in turn to a multi-modal posterior whose complexity is lost in LaSLAT's Gaussian approximation. As a result, LaSLAT's estimate may converge artificially to a very poor estimate. The corresponding prediction has low covariance since it is the product of many events, but is nevertheless quite inaccurate.

There are a number of techniques available to resolve this problem. First, it may be possible to use a heuristic over the collected measurements or the LaSLAT results to detect uninformative batches of events. Bad batches can then be ignored. In effect, LaSLAT would wait to begin localization until good quality data becomes available. Alternatively, we could use a multi-modal posterior approximation instead of the Gaussian approach detailed in this thesis.

7.3 Multi-modal Posterior Approximation

The Gaussian approximation described in section 3.3 has an important disadvantage: it cannot accurately represent a multi-modal posterior distribution. Consequently, LaSLAT requires that the initial measurement batches identify a single high probability mode. This necessity can be diminished by using a different approximation such as a mixture of Gaussians or a particle filter. These approximations can model multi-modal distributions; however, they increase the processing load for a LaSLAT batch.

We were not able to experiment with these types of representation for this thesis. However, they may be necessary for accurate localization of some networks, especially when areas have few measurements or the mobile is somewhat stationary. They may also permit reduced batch sizes when performing three dimensional SLAT.

7.4 Low Quality Hardware

In this thesis, we used the Cricket platform [1] for all of our experiments. The Cricket ranging algorithm is highly accurate in the absence of echoes, achieving 1-2 centimeter accuracy on ranges of up to ten meters. However, we suspect that LaSLAT will remain effective when less accurate hardware is used.

One significant cause of error in some acoustic time of flight ranging systems can be manufacturing differences between sensors. In our experiments with the Cricket system, we encountered no need to model such inter-sensor differences explicitly. In fact, as we indicated in section 3.5, we found it helpful to encourage sensor measurement biases to be similar. However, [35] shows that some types of acoustic ranging can benefit from careful modeling of individual sensor characteristics. LaSLAT seems well-suited to perform this type of analysis.

We have shown in this thesis that LaSLAT performs very well on Cricket networks that are deployed with reasonable density in indoor environments; however, we are very interested in determining the minimal hardware profile required for effective use of LaSLAT. This means performing additional experiments on networks with reduced sensor density, event frequency, and measurement accuracy. The low density experiments presented in chapter 6 represent initial research in this area. Nevertheless, much remains to be done, especially with regard to the use of cheaper and lower quality sensor hardware.

7.5 Sensor Dynamics

LaSLAT is well-suited to localizing moving sensors. In section 3.4, we stated the assumption that sensor positions are stationary. However, LaSLAT could easily be adapted to use the more relaxed constraint that sensors may move over time. This sensor mobility constraint might resemble the smooth dynamics extension discussed in section 4.2. This would allow LaSLAT based tracking networks even greater flexibility: a user could move sensors after deployment to optimize the network. If sensors were

able to move autonomously, sensors could automatically migrate to achieve the desired quality of sensor coverage.

7.6 Acoustic-only LaSLAT

This thesis has been written under the assumption that sensors compute ranges using Time Difference of Arrival (TDoA) ranging; thus, the mobile emits a tagged radio pulse followed by an acoustic pulse. Sensors use the difference in time of arrival between the radio message and the acoustic pulse to estimate distance.

However, the measurement model (section 3.2) can be adapted to use an acoustic pulse alone. In this case, the sensors must be time synchronized, which can be done using an off-the-shelf time synchronization system such as [18]. The sensors measure the arrival times of acoustic pulses, which they report to LaSLAT. These arrival times can be seen as a function of the true distance between the event and the sensor and the true time of the event. The event time becomes an additional parameter to be estimated by the Bayesian filter. Such a measurement model would allow SLAT to localize and track using only acoustic time of arrival. More generally, it may be possible to localize using ambient environmental noise, as long as recognizable noises are witnessed by enough sensors.

7.7 Distributed Implementation

The results provided in chapter 6 were generated using a centralized implementation. Measurements were taken in a real network, then exfiltrated to a laptop for processing. However, as we show in chapter 5, LaSLAT can be feasibly distributed to perform the necessary computations in-network. As we argue in that section, it is not always desirable to distribute LaSLAT processing; however, a distributed implementation will allow LaSLAT to be of assistance in networks where centralized computation is infeasible.

Chapter 8

Conclusion

In this thesis we presented LaSLAT, a sensor network algorithm that simultaneously localizes sensors, calibrates sensing hardware, and tracks unconstrained moving targets using only range measurements between the sensors and the mobile. On both two- and three-dimensional networks using the Cricket ranging system, LaSLAT was able to localize sensors to within several centimeters of their ground truth positions while recovering a range measurement bias for each sensor and the complete trajectory of the target.

LaSLAT is based on a Bayesian filter, which updates a probability distribution over the quantities of interest as measurements arrive. The algorithm is distributable, and requires only a constant amount of space with respect to the number of measurements incorporated. LaSLAT is easy to adapt to new types of hardware and new physical environments due to its use of intuitive probability distributions: one adaptation demonstrated in this thesis uses a mixture measurement model to detect and compensate for bad acoustic range measurements due to echoes.

Localization and calibration using an unconstrained mobile offers several advantages over localization and calibration using inter-sensor ranges. Since the mobile may move freely, the measurements taken by the sensors have greater diversity and therefore diminished likelihood of systematic errors. In particular, the mobile can help overcome interference from fixed obstacles such as furniture.

Use of a mobile also means that the sensors are no longer constrained to have

line of sight to each other. This enables a broader range of network deployments: in particular, it allows sensors to be placed to optimize tracking coverage rather than to accommodate the peculiarities of a localization algorithm.

LaSLAT in its current form could be used to facilitate deployment of indoor navigation systems, either for human use or as a guidance system for mobile robots. Using LaSLAT, the sensors for such a system could be deployed in an ad-hoc fashion on walls, floors, or ceilings. They could then use the movements of persons to be tracked (i.e. targets) to localize and calibrate in-place, forming a high-precision location service with little human effort.

LaSLAT also suggests an interesting automated sensor deployment system using a mobile robot. Such a robot would survey its environment, periodically dropping sensors. The sensors would provide high quality landmarks for the robot's mapping system, while the robot provided a moving target for LaSLAT-based localization of the sensors. The sensors could even provide a long term record of the mapping operation, which they could propagate to other robots not tasked with surveying.

While many impediments remain, LaSLAT is an important step toward cheap, easily deployable, and accurate sensor networks for target tracking.

Appendix A

Initialization

Initialization plays an important role in LaSLAT. As several authors [22, 29] have observed, inferring locations using distance measurements can suffer from *local minima*. The Newton-Raphson optimization described in appendix B does not always find the globally minimal positioning. Rather, it finds a minimum near its initial value, which may or may not be the global minimum.

This problem is typically solved in one of two ways. The first method involves running the optimization several times using different random choices of initial value. The best result from these optimizations is taken as the global minimum.

In the second method, which we use in LaSLAT, the initial iterate is chosen intelligently to be close to the true solution. As a result, the Newton-Raphson optimization converges to the global minimum on the first try. After the first LaSLAT iteration, the estimate from the previous iteration is generally a good initial iterate for Newton-Raphson. However, we depend on a separate initialization algorithm to provide an initial iterate for the first LaSLAT iteration. In particular, we use a distributed initialization algorithm suggested by Priyantha et al. [29]. We have also experimented with techniques such as gradient multilateration [23] and multidimensional scaling [17]. In our experience, the algorithm from [29] offered the best performance. This algorithm provides a coarse position estimation for the sensors in the network by defining a polar coordinate system based on hop counts between specially chosen sensors. It is described in detail below.

A.1 Initialization Using Radio Connectivity

Let $h_{i,j}$ be the minimum number of radio hops required to forward a message from sensor i to sensor j. Perform the following steps:

1. Choose node n_0 to be the sensor with minimal ID. Propagate a gradient from n_0 to compute $h_{0,k}$ for all k.

2. Choose node n_1 to be the sensor k with maximal $h_{0,k}$, breaking ties by ID. n_1 is one "corner" of the network. Propagate a gradient from n_1.

3. Choose node n_2 to be the sensor k with maximal $h_{1,k}$, breaking ties by ID. n_2 is the corner opposite n_1. Propagate a gradient from n_2.

4. Choose node n_3 to be the sensor k that minimizes $|h_{2,k} - h_{1,k}|$, breaking ties by the sum $h_{2,k} + h_{1,k}$ (any further ties are resolved by ID). n_3 is the corner between n_1 and n_2. Propagate a gradient from n_3.

5. Choose node n_4 to be the sensor k that minimizes $|h_{2,k} - h_{1,k}|$, breaking ties by maximizing $h_{3,k}$ (any further ties are resolved by ID). n_4 is the final corner. Propagate a gradient from n_4.

6. Choose node n_5 to be the sensor k that minimizes $|h_{2,k} - h_{1,k}| + |h_{4,k} - h_{3,k}|$, breaking ties by ID. This sensor is closest to the center of the network. Propagate a gradient from n_5.

7. For each sensor k, define $\rho_k = h_{5,k} * R$, and $\tan(\theta_k) = \frac{h_{1,k} - h_{2,k}}{h_{3,k} - h_{4,k}}$, where R is the maximum range of the sensor's radio. Compute the position $\mathbf{s}_k^x$ by converting the polar coordinate (ρ_k, θ_k) to rectangular coordinates.

This initialization algorithm tends to be highly inaccurate (see figure A-1). However, the chosen positions tend to help the LaSLAT optimization avoid local minima that decrease accuracy. Our specific choice of initialization is somewhat irrelevant – any reasonably accurate localization algorithm could be substituted.

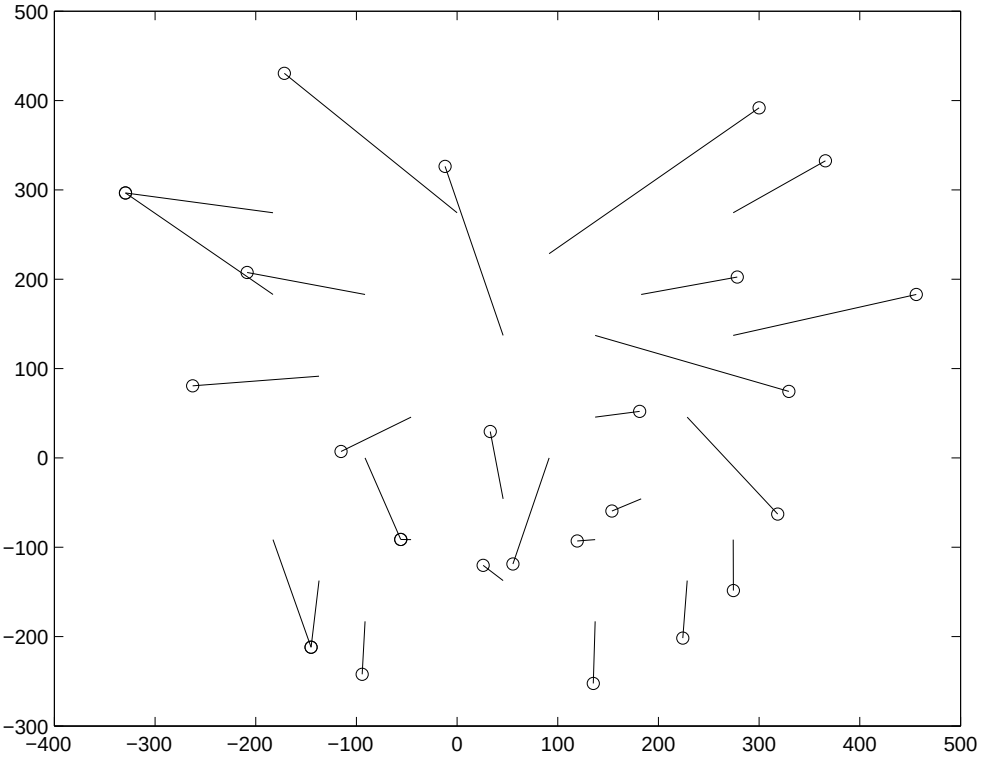

Figure A-1: Initialization results for the 27 node 2D experimental topology. Initial position estimates are marked with circles. Lines are drawn from these estimates to the corresponding ground truth sensor locations. The average position estimate error is 70 cm.

A.2 Initialization in 3D

The algorithm described in the previous section is not designed to localize sensor configurations that have height differences. In general, most two dimensional localization algorithms extend poorly to practical three dimensional topologies. Most two dimensional topologies resemble a "cloud" of sensors – sensors are evenly scattered throughout a generally convex area. In three dimensions, this is typically infeasible, especially indoors. Instead, practical deployments look more like figure 6-11, where sensors are placed mostly on walls and either the floor or the ceiling. As a result, naive three dimensional extensions to 2D localization algorithms such as [11, 17, 22, 29, 31] tend to perform badly on real networks.

We have developed a simple method of extending these algorithms to three dimensions, using a single additional bit of information that indicates whether a sensor is placed on a floor or a wall. This information could be obtained by a relatively cheap and imprecise orientation sensor on each node or by manual configuration. Also, it is probable that in an indoor environment the form factor of the sensors will vary between wall sensors and floor sensors, making manual configuration straightforward.

If the radio range is sufficiently small (this may be accomplished by temporarily reducing the power of the sensors' radios), a 3D topology like figure 6-11 will appear to be two dimensional from the perspective of radio connectivity, where wall sensors appear at the outer extremities of the topology. In this case, a two dimensional localization algorithm such as the one described in the previous section may be applied. As a post-processing step, the regions of the two dimensional plane containing wall sensors may be "folded" upwards into the height dimension to approximate the structure of the 3D environment.

In our experience so far, this trivial modification allows algorithms such as [29] to be successfully used in 3D LaSLAT.

Appendix B

Newton-Raphson

In this appendix, we show how to optimize equation (3.6) using the Newton-Raphson iterative optimization algorithm. We also derive the curvature of the negative log posterior required for Laplace's method (section 3.3), and show that mode finding preserves local connectivity.

B.1 Finding a Mode

Equation (3.6) can be expressed in the form of non-linear least squares. Let $f_{ij}(\mathbf{x}) = \|\mathbf{s}_i^x - \mathbf{e}_j^t\| + \mathbf{s}_i^\theta$, and define $\mathbf{f}(\mathbf{x})$ as a column vector consisting of all $f_{ij}(\mathbf{x})$. Using $\Omega_\mathbf{x} = \mathrm{Cov}^{-1}\left[\mathbf{x}^t|\mathbf{y}^{old}\right]$, and $\mu_\mathbf{x} = E\left[\mathbf{x}^t|\mathbf{y}^{old}\right]$, we can write equation (3.6) as:

$$\arg\min_\mathbf{x} \frac{1}{\sigma^2}\|\mathbf{f}(\mathbf{x}) - \mathbf{y}^t\|^2 + (\mathbf{x} - \mu_\mathbf{x})^T\Omega_\mathbf{x}(\mathbf{x} - \mu_\mathbf{x}) \tag{B.1}$$

Each iteration of Newton-Raphson maps an iterate $\mathbf{x}^{(t)}$ to the next iterate $\mathbf{x}^{(t+1)}$ by approximating (B.1) with a linearization about $\mathbf{x}^{(t)}$, then optimizing over $\mathbf{x}$:

$$\mathbf{x}^{(t+1)} = \arg\min_\mathbf{x} \frac{1}{\sigma^2}\left\|\nabla\mathbf{f}^{(t)}\mathbf{x} - \mathbf{b}\right\|^2 + (\mathbf{x} - \mu_\mathbf{x})^T\Omega_\mathbf{x}(\mathbf{x} - \mu_\mathbf{x}), \tag{B.2}$$

where the matrix $\nabla\mathbf{f}^{(t)}$ is the derivative of $\mathbf{f}$ with respect to $\mathbf{x}$ at $\mathbf{x}^{(t)}$, and the column vector $\mathbf{b} = \nabla\mathbf{f}^{(t)}\mathbf{x}^{(t)} - \mathbf{f}\left(\mathbf{x}^{(t)}\right) - \mathbf{y}^t$.

Equation (B.2) is a linear least squares problem in terms of $\mathbf{x}$. Its solution can be found by setting the derivative with respect to $\mathbf{x}$ to zero. This leaves a linear problem that can be solved by matrix inversion:

$$\left[\Omega_{\mathbf{x}} + \frac{1}{\sigma^2}\nabla\mathbf{f}^{(t)\top}\nabla\mathbf{f}^{(t)}\right]\mathbf{x} = \Omega_{\mathbf{x}}\mu + \frac{1}{\sigma^2}\nabla\mathbf{f}^{(t)\top}\mathbf{b}. \tag{B.3}$$

Furthermore, differentiating (B.2) one more time results in $\mathbf{H} = \Omega_{\mathbf{x}} + \frac{1}{\sigma^2}\nabla\mathbf{f}^{(t)\top}\nabla\mathbf{f}^{(t)}$, which is required for Laplace's method (section 3.3). Since (B.2) is an approximation to the negative log posterior (3.6), $\mathbf{H}$ serves as an approximation to its Hessian at $\mathbf{x}^{(t)}$.

B.2 Locality of Mode Finding

Because the true distance f_{ij} depends only on sensor i and event location j, each row of $\nabla\mathbf{f}^{(t)}$ is made up of zeros, except at locations corresponding to the ith sensor and the jth event. Thus $\nabla\mathbf{f}^{(t)\top}\nabla\mathbf{f}^{(t)}$ has local connectivity. If $\Omega_{\mathbf{x}} = \mathrm{Cov}^{-1}\left[\mathbf{x}^t|\mathbf{y}^{old}\right]$ has local connectivity, then the updated covariance matrix $\mathrm{Cov}^{-1}\left[\mathbf{x}^t|\mathbf{y}^{old}\right] = \Omega_{\mathbf{x}} + \nabla\mathbf{f}^{(t)\top}\nabla\mathbf{f}^{(t)}/\sigma^2$ also has local connectivity. Therefore as claimed in chapter 5, incorporating a batch of measurements preserves local connectivity.

Bibliography

[1] H. Balakrishnan, R. Baliga, D. Curtis, M. Goraczko, A. Miu, N. B. Priyantha, A. Smith, K. Steele, S. Teller, and K. Wang. Lessons from developing and deploying the cricket indoor location system. Technical report, MIT Computer Science and AI Lab, `http://nms.lcs.mit.edu/projects/cricket/#papers`, 2003.

[2] D. P. Bertsekas and J. T. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, 1989.

[3] R. Brooks, P. Ramanathan, and A. Sayeed. Distributed target classification and tracking in sensor networks. In *Proceedings of the IEEE*, volume 91. IEEE, August 2003.

[4] N. Bulusu, V. Bychkovskiy, D. Estrin, and J. Heidemann. Scalable, ad hoc deployable rf-based localization. In *Grace Hopper Celebration of Women in Computing Conference 2002, Vancouver, British Columbia, Canada.*, October 2002.

[5] W. Butera. *Programming a paintable computer*. PhD thesis, mit, 2002.

[6] V. Bychkovskiy, S. Megerian, D. Estrin, and M. Potkonjak. A collaborative approach to in-place sensor calibration. UCLA Tech Report.

[7] S. Capkun, M. Hamdi, and J. Hubaux. GPS-free positioning in mobile ad-hoc networks. In *HICSS*, 2001.

[8] V. Cevher and J.H. McClellan. Sensor array calibration via tracking with the extended kalman filter. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 5, pages 2817–2820, 2001.

[9] W. Chen, C. Hou, and L. Sha. Dynamic clustering for acoustic target tracking in wireless sensor networks. In *IEEE International Conference on Network Protocols*, 2003.

[10] B. Dalton and M. Bove. Audio-based self-localization for ubiquitous sensor networks. In *118th Audio Engineering Society Convention*, 2005.

[11] L. Doherty, L. El Ghaoui, and K. S. J. Pister. Convex position estimation in wireless sensor networks. In *Proceedings of Infocom 2001*, April 2001.

[12] A. Galstyan, B. Krishnamachari, K. Lerman, and S. Pattem. Distributed online localization in sensor networks using a moving target. In *Information Processing In Sensor Networks (IPSN)*, 2004.

[13] A. Gelman, J. Carlin, H. Stern, and D. Rubin. *Bayesian Data Analysis*. Chapman & Hall/CRC, 1995.

[14] G. Golub and C.F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1989.

[15] B. K. P. Horn. Relative orientation. *International Journal of Computer Vision*, 4:59–78, January 1989.

[16] A. Ihler, J. Fisher, R. Moses, and A. Willsky. Nonparametric belief propagation for self-calibration in sensor networks. In *Information Processing in Sensor Networks (IPSN)*, 2004.

[17] X. Ji and H. Zha. Sensor positioning in wireless ad hoc networks using multidimensional scaling. In *Infocom*, 2004.

[18] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. Robust multi-hop time synchronization in sensor networks, 2004.

[19] P. F. McLauchlan. A batch/recursive algorithm for 3D scene reconstruction. *Conf. Computer Vision and Pattern Recognition*, 2:738–743, 2000.

[20] J. McLurkin. Algorithms for distributed sensor networks. Master's thesis, UCB, December 1999.

[21] L. Meertens and S. Fitzpatrick. The distributed construction of a global coordinate system in a network of static computational nodes from inter-node distances, 2004.

[22] D. Moore, J. Leonard, D. Rus, and S. Teller. Robust distributed network localization with noisy range measurements. In *Proceedings of ACM Sensys-04*, Nov 2004.

[23] R. Nagpal, H. Shrobe, and J. Bachrach. Organizing a global coordinate system from local information on an ad hoc sensor network. In *2nd International Workshop on Information Processing in Sensor Networks (IPSN '03), Palo Alto, published as Lecture Notes in Computer Science LNCS 2634*, April 2003.

[24] D. Niculescu and B. Nath. Ad hoc positioning system (aps, 2001.

[25] D. Niculescu and B. Nath. Localized positioning in ad hoc networks, 2003.

[26] E. Olson, J. J. Leonard, and S. Teller. Robust range-only beacon localization. In *Proceedings of Autonomous Underwater Vehicles*, 2004.

[27] P. Pathirana, N. Bulusu, S. Jha, and A. Savkin. Node localization using mobile robots in delay-tolerant sensor networks. *IEEE Transactions on Mobile Computing*, 4(4), Jul/Aug 2005.

[28] B. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller. Mobile-assisted localization in wireless sensor networks. In *IEEE INFOCOM*, Miami, FL, March 2005.

[29] N. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller. Anchor-free distributed localization in sensor networks, 2003.

[30] A. Savvides, C. Han, and M. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *Mobile Computing and Networking*, pages 166–179, 2001.

[31] Y. Shang, W. Ruml, Y. Zhang, and M. Fromherz. Localization from mere connectivity. In *MobiHoc*, 2003.

[32] S. Simic and S. Sastry. Distributed localization in wireless ad hoc networks, 2002.

[33] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In *Uncertainity in Artificial Intelligence*, 1988.

[34] S. Thrun, Y. Liu, D. Koller, A.Y. Ng, Z. Ghahramani, and H. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. Submitted for journal publication, April 2003.

[35] C. Whitehouse. The design of calamari: an ad-hoc localization system for sensor networks. Master's thesis, University of California at Berkeley, 2002.

[36] X. Yu, K. Niyogi, S. Mehrotra, and N. Venkatasubramanian. Adaptive target tracking in sensor networks. In *Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS'04)*, January 2004.