Exploiting Ultrasound Sensitivity Patterns to Accurately Localize Mobile Motes in Real Time using Particle Filtering

Nathan Burkhart

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2007-67

http://www.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-67.html

May 18, 2007

Acknowledgement

I¿m eternally grateful to Menasheh Fogel, Jeremy Schiff, and Hongliang Ren for their assistance with this project, there¿s no way any of this would have happened without you. Obviously, my advisor Ken Goldberg was a big help, thanks for keeping me focused. Thanks to Philippe Cud¿re-Mauroux, Menasheh, Ken, and Mike Franklin for laying the groundwork for this project. And, of course, a big thanks to all of the students in the Automation Lab for their input and help with experimentation.

Exploiting Ultrasound Sensitivity Patterns to Accurately Localize Mobile Motes in Real Time using Particle Filtering

Abstract

Exploiting Ultrasound Sensitivity Patterns to Accurately Localize Mobile Motes in Real Time using Particle Filtering

by

Nathan Burkhart

Master of Science in Computer Science

University of California, Berkeley

Professor Ken Goldberg, Chair

I explore the use of a wireless sensor network to automatically track materials in a warehouse, in order to comply with spatial constraints. This system, which is required to operate in real-time, employs sensor motes that use a combination of ultrasound bursts (chirps) and radio frequency to compute pairwise distance estimates. However, these estimates are based on data that is inherently noisy and unreliable. To address this issue, I outline a probabilistic algorithm that makes use of an explicit model of the ultrasound sensitivity pattern of Cricket motes. This approach makes use of particle filtering, a non-parametric method that is able to estimate the positions of potentially mobile motes. Particle filtering employs an arbitrary probability distribution as a model of the observed physical evidence, and uses sampling to produce estimated object states. I develop a number of potential observation models and compare their effectiveness with simulations based on collected real-world sensor data and computed distance measurements. These experiments suggest that the use of this algorithm coupled with probabilistic sensor models can accurately localize up to eight mobile motes to within an average of 8.5 cm, while updating their estimated positions based on collected distance estimates as often as once every second.

Contents

Acknowledgements

I'm eternally grateful to Menasheh Fogel, Jeremy Schiff, and Hongliang Ren for their assistance with this project, there's no way any of this would have happened without you. Obviously, my advisor Ken Goldberg was a big help, thanks for keeping me focused. Thanks to Philippe Cudŕe-Mauroux, Menasheh, Ken, and Mike Franklin for laying the groundwork for this project. And, of course, a big thanks to all of the students in the Automation Lab for their input and help with experimentation.

Chapter 1

Introduction

Effectively managing the materials stored in warehouses is a multi-billion dollar global challenge. There are known database solutions used to monitor the entry and exit of materials, but they are not able to monitor the locations of materials. Often, warehouses are subject to regulatory requirements in terms of volumetric and spatial constraints. As a concrete example, regulations imposed by the California Fire Code (CFC) require that no more than 1000 gallons of a liquid corrosive (acid) material can be stored in a single warehouse, that an acid cannot be stored within twenty feet of a base, and that a flammable material cannot be stored within twenty feet of an oxidizer. Maintaining these constraints, which must be followed both during transit and storage, is not a trivial undertaking.

To tackle the problem of monitoring spatial constraints, I examine the use of a wireless sensor network to automatically track the pallets. This system would be required to operate in real-time, taking notice of the violations as they occur and alerting a human operator, while working with pallets that are potentially mobile. Wireless sensor motes used for localization employ some combination of ultrasound bursts (chirps), radio, infrared, or lasers; they compute the distance between pairs of motes by examining either the time of flight, angle, or strength of the received signal. Some examples of such technologies that are available commercially are the Cricket [1] [2], Mica2 [3], and Telos [4] motes. Though promising, the distance measurements computed by these motes are noisy due

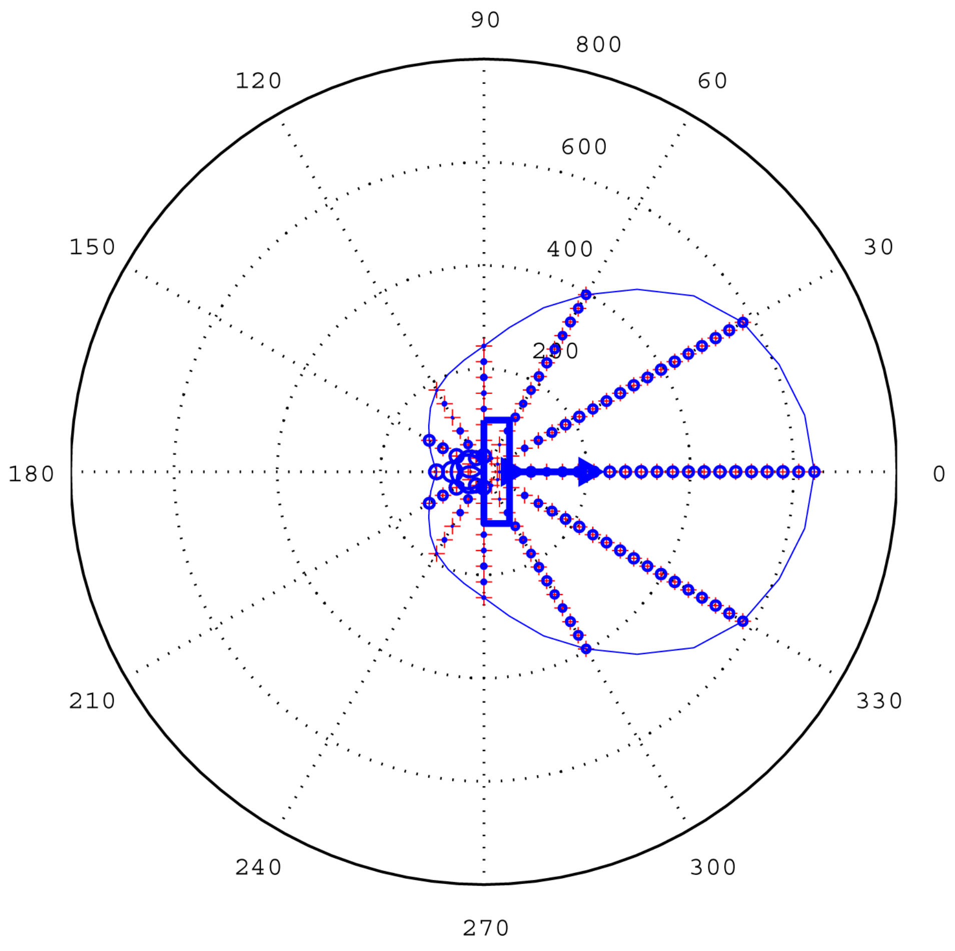


Figure 1.1: Chart of the ultrasound sensitivity pattern observed in [5]. The rectangle and arrow depict the location and orientation of the ultrasonic microphone. A polar function approximating the limit of the ultrasound signal range is drawn over the chart, highlighting the cardioid shape.

to the physical limitations of the sensor hardware, environmental disturbances, and the inherent imperfection of the analog world. As a result, the accuracy with which these measurements can be used to localize is degraded.

In [5], the authors carried out a detailed study of the capabilities of the Cricket motes, and they discovered that both the accuracy of the computed distance measurement and the maximum range at which the ultrasound signal could be detected was highly dependent on the relative angle between the transmitting and receiving mote. The experimental results of that study are visualized in Figure 1.1. As the figure shows, the acoustic pattern of the ultrasound signal exhibits a cardioid shape.

In this paper I propose that, by explicitly modeling the sensitivity pattern of the sensors used for localization, the accuracy of localization methods can be improved. I outline a probabilistic algorithm based on Particle Filtering [6] that takes advantage of an arbitrary probability distribution describing the sensitivity pattern of ultrasound sensors. I propose several possible methods for modeling this pattern, varying in their computation expense and accuracy. I then build a simulator that is able to execute the proposed algorithm using sensor data and computed distance measurements collected from real-world experimentation. Through simulations, I compare the accuracy of the various observation models, as well as explore the effectiveness of different mote configurations and algorithm parameters. These experiments suggest that the use of this algorithm coupled with probabilistic sensor models can accurately localize up to eight mobile motes to within an average of 8.5 cm, while updating their estimated positions based on collected distance estimates as often as once every second.

Chapter 2

Related Work

2.1 Spatial Constraints in Warehouse Management

Most of the research surrounding warehouse management is focused on optimal deployment and efficient retrieval [7] according to the principles of supply chain management. One of the more prominent technologies is Radio Frequency Identification (RFID), which has been put to use for inventory management and storage tracking [8]. An operational RFID system consists of one or more antennas that remotely retrieve data through radio waves from objects called tags. In [9] the authors discuss the use of RFID tags to improve the localization of mobile robots, outfitting the robot with two overhead antennas that compute distance readings relative to pre-deployed RFID tags and using these readings to estimate its location using a map of the physical environment. However, this system must be centrally operated using state-of-the-art technologies.

The authors in [10] also approach the problem of monitoring the relative proximity of materials in a warehouse using a wireless sensor network. In their constraint-based method, a spatial distance graph is constructed and the edges between nodes are labelled with distance constraints. Distance intervals between pairs of nodes are then derived using a modified Floyd-Warshall shortest path algorithm for all edges. This is a promising approach for proximity violation detection because it does not rely on any external infrastructure and can be executed in ad hoc sensor networks. However, the method assumes that the computed ranging results are free of noise, and each node in the network must have enough memory to store the entire graph and all distance constraints.

2.2 Localization in Wireless Sensor Networks

Beyond warehouse management, localization techniques for wireless sensor networks have been extensively studied in recent years; some excellent reviews of the field may be found in [11] [12]. The types of signals most commonly used in localization are acoustic, radio frequency, and laser. The approaches to compute pairwise distances using these signals include examination of the time of arrival [1], angle of arrival [13], and signal strength [14]. Many signal estimation techniques have been proposed to infer true object locations by aggregating multiple sensor measurements. For example, in [13] a sensor node is localized by collecting transmissions from three or more fixed beacon nodes and performing analysis based on the estimated angles of arrival of the signals.

For the particular application of dynamically monitoring spatial constraints, an inexpensive and scalable solution is required. Angle-of-arrival approaches typically need a special antenna configuration, and they are vulnerable to multi-path reflections. Measuring the received signal strength is relatively simple and inexpensive, but has not yet been shown to meet the dynamic resolution requirements of warehouse monitoring. Along this path, there have been some recent developments in [15] using the Mica2 mote where the phase offsets of radio frequencies are analyzed in order to infer distance. Estimation error was shown to be very small ($\sim$4cm on average), but the time necessary to accurately localize (about 80 minutes) would prohibit the tracking of dynamically moving nodes.

2.3 Cricket Motes

Due to both their purported capabilities and their availability, this paper examines the use of the MIT-developed Cricket motes [1] [2] for localization. The Cricket motes utilize a combination of radio frequency and ultrasound pulses, inferring pairwise distances based on the relative arrival times of the two signals. The system makes use of an on-board thermometer so that its computations can compensate for the effects of temperature on the speed of sound. They also implement interference avoidance and collision detection, in the case of the near-simultaneous arrival of radio and ultrasound signals from different transmitters. Furthermore, the Cricket location system in [16] implements communication scheduling to avoid radio collision and Kalman filtering to discard incorrect distance estimates.

In [16], a trilateration algorithm based on least-squared-error is used to derive the position of sensor nodes from multiple distance measurements. A robust quadrilateral is introduced as a clique to avoid flip ambiguities. Additionally, Kalman filters are used to derive the location of mobile nodes. Although the paper reported that the algorithm was indeed effective, I was unable to recreate their results in physical environments that were less than perfect. And although the effects on accuracy caused by the relative distance and angle of the motes were observed, along with the recognition of a cardioid sensitivity pattern, the authors did not explicitly compensate for these observations.

The most commonly recognized type of sensitivity beam pattern exhibited by ultrasonic sensors has one main lobe and several side lobes [17] [18], which is often abstracted to a cardioid shape without side lobes in the name of simplicity. In [19] an observation model that accounts for the sensitivity pattern of ultrasonic receivers is presented. The model incorporates both relative angle and distance measurements; however, the sensitivity pattern does not account for received signal behind the microphone, as observed in a cardioid sensitivity pattern.

2.4 Probabilistic Localization

Noisy sensor measurements are inevitable due to hardware limitations and environmental factors. In order to cope with this inherent uncertainty, probabilistic methods are often employed. Localization within a Bayesian probabilistic framework has found widespread applications in robotics [20], with demonstrated adequate performance. Moreover, in recent studies [21] [22] [23] [11] [24], similar methods based on a probabilistic framework have been introduced to the problem of localization within wireless sensor networks.

In nearly all approaches, probabilistic inferences are made by observing and aggregating multiple sets of noisy physical evidence. Furthermore, the analysis of sets of data over time can further improve localization, especially in the presence of mobile nodes. The authors in [12] demonstrated that, by using an algorithm based on a sequential Monte Carlo method, mobile seeds can be exploited to actually improve the precision of localization.

Particle filtering, the approach this paper chooses to explore, is an implementation of Bayesian filtering [25] based on a sequential Monte Carlo method. It makes use of two probabilistic models: a transition model and an observation model. Unlike the formal representation of the linear dynamic system in a Kalman filter, the key of particle filtering is the representation of the posterior distribution by a set of weighted samples, so that it can approximately represent any arbitrary probability distribution. In the context of localization [25], the position of each entity is first represented by a set of particles distributed according to some initial distribution. The transition model is then used to advance these sample states, and the new particles are assigned weights by the observation model according to the likelihood of the collected evidence observations (commonly the prior estimation). These weights are then used to estimate the entity's true state.

In [21], a particle filtering approach is used to track intruders in a room using a collection of passive infrared motion sensors with binary outputs. The authors equipped the room with an array of motion sensors, characterized the firing patterns of these sensors by conditional probability distributions, and used the constructed sensor model to derive the posterior probability by observing the sensor triggering events. It was shown that this method can accurately track an intruder in real time; it was able to take multiple photographs of the subject using a robotic camera informed by the algorithm's estimated intruder position.

Compared with Kalman filtering, the advantage of particle filtering is its capability to represent an arbitrary probability model and its flexibility when dealing with dynamic systems. Kalman filtering is known to be optimal only when both the transition model and observation model are linear Gaussian distributions. This makes Kalman filtering infeasible for pallet monitoring, because the altrasound sensors used for localization exhibit non-Gaussian behavior patterns as illustrated in Figure 1.1. Particle filtering explicitly approximates a probability distribution as a set of sampled points, so it will outperform extended Kalman filtering in situations where a linear Gaussian approximation cannot accurately describe the distribution.

However, particle filtering is not without its shortcomings. A set of samples must be maintained at all times, and the accuracy of the filtering is related to the size of this set. Therefore, the computational complexity of particle filtering is proportional to the number of samples, and the optimal number of samples is difficult to determine. For more detail, a computational complexity analysis of different types of particle filters can be found in [26]. It has been my experience, however, that this trade-off between computation time and accuracy proves to be more than acceptable for the proposed application of monitoring spatial constraints.

Chapter 3

Problem Statement

3.1 The Problem at Hand

The object of the proposed system is to monitor the proximity of different materials in a warehouse, so that spatial constraint violations can be detected. At the heart of this challenge is the problem of accurately localizing pallets of materials; if the location of each pallet is known, detecting spatial constraint violations is trivial. Thus, it is framed purely as a localization problem. The solution must be able to continuously monitor the location of both stationary and mobile pallets in a room of known dimensions, and it must operate in real time.

3.2 Proposed Solution

3.2.1 System Overview

Like in the global architecture proposed in [5], a number of Cricket motes are statically mounted in known locations on the ceiling, with the ultrasound sensor facing downward. These motes will be referred to as Beacons. Cricket motes are also affixed to the pallets on the floor, with the ultrasound sensor facing upward. These will be referred to as Motes. Refer to Figure 3.1 for a visual depiction of the described setup.

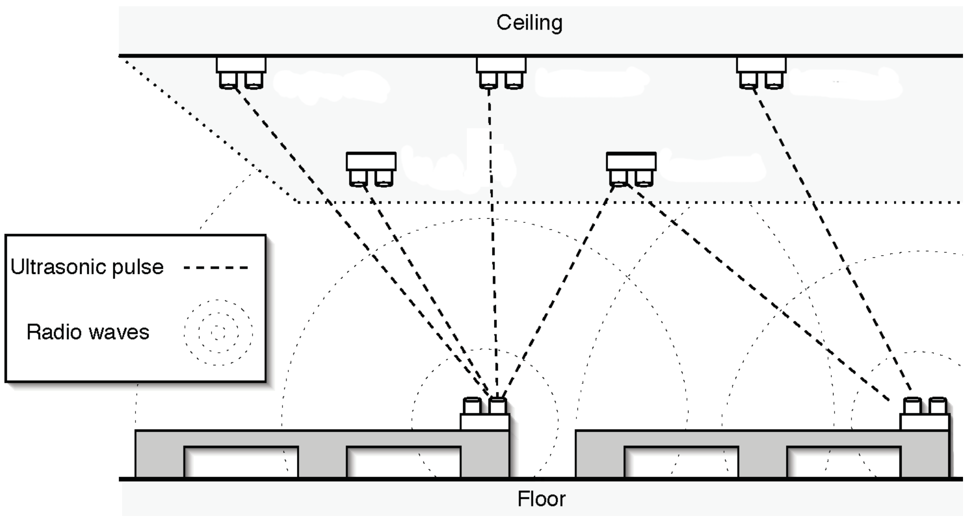


Figure 3.1: Visual depiction of the layout of Cricket motes for monitoring the location of hazardous materials in a warehouse, as seen in [5]. The motes on the ceiling are referred to as Beacons, while the motes affixed to the pallets on the floor are referred to as Motes.

At a steady, predetermined rate, each Mote simultaneously broadcasts both an ultrasound pulse and a radio message containing a unique Mote identifier and a monotonically increasing message number. For every pair of radio message and ultrasound pulse that is received, each Beacon computes a pairwise distance estimate between itself and the sending Mote based on the time difference in arrival of the two signals. This distance estimate is transmitted to a central processor, along with the Mote identifier and the message number taken from the received radio message. At set intervals, or iterations, the distance estimates collected during the previous iteration are analyzed and the Euclidean position of each Mote is estimated. This computation is achieved using a particle filtering algorithm, discussed in much more detail below in Section 4.2, that explicitly takes advantage of a defined model of the ultrasound sensitivity pattern.

3.2.2 Assumptions

It is assumed that the floor and ceiling of the warehouse are both flat and level, at least relative to each other. The surface of all pallets, when they are placed on the floor, are level and of a uniform height. The Motes are placed in the center of the pallets. Other than the Beacons, Motes, and pallets, there are no other objects in the room. There is a central computer that is able to receive transmissions from each ceiling-mounted Beacon; this is where the algorithm is executed. It is also assumed that the pairwise distance estimates computed by the Beacons are probabilistically independent of each other; the implementation of interference avoidance and collision detection lends credence to this assumption, and physical experimentation has also reinforced its validity.

3.2.3 System Inputs

The system is initialized with the Euclidean coordinates of each Beacon. The dimensions of the room are also specified by length, width, and height values. At each iteration, the algorithm collects pairwise distance estimates between some subset of all existing (Beacon, Mote) pairs (the pairs for which ultrasound and radio signals were successfully transmitted and received).

3.2.4 System Outputs

After each iteration, the system outputs the estimated Euclidean coordinates of every Mote.

Chapter 4

Particle Filtering

4.1 Overview of Particle Filtering

4.1.1 Introduction to Particle Filtering

Particle filters [6], sometimes referred to as sequential Monte Carlo methods, are sampling-based methods used to estimate Bayesian models. They tout a number of benefits relevant to the proposed application. When designed properly, they perform efficiently enough to be executed in real-time [21]. Particle filters are able to employ arbitrary probability distributions, as opposed Kalman filtering's assumption that the posterior density at every time step is Gaussian and can be parameterized by simply a mean and covariance. Furthermore, with a sufficient number of samples they approach the Bayesian optimal estimate, making them more accurate than other approximate methods such as extended Kalman filtering.

In particular, the employed algorithm makes use of the sampling importance resampling (SIR) filtering method proposed in [27]. This approach makes it such that the importance density can be easily sampled, making it more feasible for real-time execution than other variations on particle filtering. The following derivation and description of the generalized particle filtering algorithm in Sections 4.1.2 and 4.1.3 will make use of the notation given in Table 4.1.

Table 4.1: Notation for a general description of SIR particle filtering.

x_t	The true state of the object at time t.
$\hat{x}_t$	The estimated state of the object at time t.
$\hat{x}_{st}$	A sample state s of the object at time t.
w_{st}	The weight assigned to sample state s at time t.
z_t	The observed measurements collected at time t.

4.1.2 Derivation of Particle Filtering

The goal of particle filtering is to produce an estimate $\hat{x}_t$ of the true state of an object at time t, x_t, where x_t is defined by some possibly nonlinear function $x_t = f_t(x_{t-1})$. This estimate $\hat{x}_t$ is based on the aggregated set of all observed measurements $z_{1:t} = \{z_1, \ldots, z_t\}$. To compute $\hat{x}_t$, the algorithm must be able to calculate some confidence level regarding some potential state $\hat{x}_{st}$, given the available data $z_{1:t}$ leading up to time t. Thus, it is required to implicitly construct the probability density function (PDF) $P(\hat{x}_{st}|z_{1:t})$. It is assumed that the prior PDF, $P(\hat{x}_{s0}|z_0)$, where z_0 is the empty set of no measurements, is known. Because the function $x_t = f_t(x_{t-1})$ describes a Markov process of order one, meaning the probability distribution of future states of the object depends solely on the present state of the object, the following holds true:

$$P(\hat{x}_{st}|\hat{x}_{s(t-1)}, z_{1:t-1}) \quad = \quad P(\hat{x}_{st}|\hat{x}_{s(t-1)}) \tag{4.1}$$

Supposing that the PDF $P(\hat{x}_{s(t-1)}|z_{1:t-1})$ is known, the prior PDF for $\hat{x}_{st}$ can be obtained via the Chapman-Kolmogorov equation:

$$P(\hat{x}_{st}|z_{1:t-1}) \quad = \quad \int P(\hat{x}_{st}|\hat{x}_{s(t-1)})P(\hat{x}_{s(t-1)}|z_{1:t-1})d\hat{x}_{s(t-1)} \tag{4.2}$$

Using observed measurements z_t collected at time t, the prior distribution can then be updated by making use of Bayes' rule:

$$P(\hat{x}_{st}|z_{1:t}) \quad = \quad \frac{P(z_t|\hat{x}_{st})P(\hat{x}_{st}|z_{1:t-1})}{P(z_t|z_{1:t-1})} \tag{4.3}$$

$$P(z_t|z_{1:t-1}) \quad = \quad \int P(z_t|\hat{x}_{st})P(\hat{x}_{st}|z_{1:t-1})d\hat{x}_{st} \tag{4.4}$$

The relations defined in Equations 4.2, 4.3 give the basis for the optimal Bayesian solution. Though, in general, it cannot be solved analytically, SIR particle filtering is used to approximate the optimal solution. Because, as noted above, $P(\hat{x}_{s0}|z_0)$ is known, the only missing pieces in the formulation are $P(\hat{x}_{st}|\hat{x}_{s(t-1)})$ and $P(z_t|\hat{x}_{st})$. These are referred to as the transition model and observation model, respectively. Given a definition of these two models, SIR particle filtering produces the estimated state $\hat{x}_{st}$ by weighted sampling.

4.1.3 SIR Particle Filtering

At the outset of the algorithm, at $t = 0$, the set of sample states $\hat{x}_{st}$ are produced using the PDF $P(\hat{x}_{s0}|z_0)$, where z_0 is the empty set of no measurements. Once these initial samples are created, time t is advanced to $t = 1$ and the algorithm can proceed as follows, where t represents the current state of discretized time.

First, the set of observed measurements z_t is collected. From each sample $\hat{x}_{s(t-1)}$ a new sample state $\hat{x}_{st}$ is produced using the transition model, $P(\hat{x}_{st}|\hat{x}_{s(t-1)})$. It is important to note that this need not be a Gaussian model, or even a parametric model; any arbitrary PDF can be used. Each sample $\hat{x}_{st}$ is assigned a weight using the observation model: $\forall s \in S, w_{st} = P(z_t|\hat{x}_{st})$. These weights are then normalized, such that $\sum_{s \in S} w_{st} = 1$. The set of sample states is then rederived using these assigned weights, such that the number of times the state $\hat{x}_{st}$ appears in the new sample set is proportional to w_{st}. This resampling is done with replacement, so that the newly derived set of samples $\hat{x}_{st}$ replaces the previous one. Given this set of possible states $\hat{x}_{st}$, the estimated state of the object $\hat{x}_t$ is computed, such that the number of samples $\hat{x}_{st}$ that are equal to a particular state is proportional to the likelihood that $\hat{x}_t$ will be equal to that state.

4.2 Particle Filtering Applied to Localization

Table 4.2 contains important notation that will be used in the current section. In the proposed application of particle filtering to localization, a separate particle filter is executed for each individual Mote. Therefore, the state of the object x_{it} refers to the Euclidean

Table 4.2: Notation for the specific application of SIR particle filtering to localizing Motes based on distance estimates computed by ceiling-mounted Beacons.

$i \in M$	The set of Motes.
$j \in B$	The set of Beacons.
$s \in S$	The set of sample states.
$t \in T$	The set of discretized times.
x_{it}	The true state of Mote i at time t. State is defined by a pair of (x, y) Euclidean coordinates.
$\hat{x}_{it}$	The estimated state of Mote i at time t.
$\mathbf{z_{it}}$	The vector $[z_{i1t}, \ldots, z_{ijt}, \ldots, z_{imt}]$ of computed distance estimates relative to Mote i at time t. An individual element z_{ijt} is relative to an individual Beacon j, and m is the total number of Beacons, $\|B\|$.
$\hat{x}_{ist}$	The sample state s representing a possible state of Mote i at time t.
w_{ist}	The weight assigned to sample state $\hat{x}_{ist}$.
$\hat{d}_{ijst}$	The Euclidean distance between the sample state $\hat{x}_{ist}$ and Beacon j.
$\hat{\theta}_{ijst}$	The relative angle between the sample state $\hat{x}_{ist}$ and Beacon j.

coordinates of a single Mote i. During the calibration phase, which occurs once before the particle filtering algorithm can be carried out, the PDF $P(\mathbf{z_{it}}|\hat{x}_{ist})$ for the observation model is constructed using the experimental data collected in [5]. It is this model that lends novelty to the approach, incorporating empirical data in order to better predict the expected distance estimates. Multiple models were constructed for inclusion in the algorithm, detailed in Section 4.3. The actual execution of the particle filtering algorithm is described below.

At the start of the algorithm, the initial samples are distributed according to $P(\hat{x}_{is0}|\mathbf{z_{i0}})$, which places them uniformly at random within the defined space. Once these initial samples are created, t is advanced to $t = 1$, where t represents the current state of discretized time, and the algorithm can proceed as follows.

The algorithm execution during one timestep t is outlined in Algorithm 1. First, the set of computed distance estimates $\mathbf{z_{it}}$ is collected: for every Beacon j that received an ultrasound pulse and radio message from Mote i sometime since time $t-1$, z_{ijt} is computed by Beacon j and transmitted to the central processor and added to $\mathbf{z_{it}}$. The sample states

Algorithm 1 *FILTER*

```
Require: $(\forall s \in S(\hat{x}_{is(t-1)}))$
Ensure: $(\hat{x}_{it}, \forall s \in S(\hat{x}_{ist}))$
  $\mathbf{z_{it}} = \{\}$
  $FORALL\ j \in B$
      $IF$ Beacon $j$ received ultrasound and radio from Mote $i$
          add $z_{ijt}$ to $\mathbf{z_{it}}$
  $FORALL\ s \in S$
      draw $\hat{x}_{ist}$ from $P(\hat{x}_{ist}|\hat{x}_{is(t-1)})$
      $w_{ist} = P(\mathbf{z_{it}}|\hat{x}_{ist})$
  $total = \sum_{s \in S} w_{ist}$
  $FORALL\ s \in S$
      $w_{ist} = \frac{w_{ist}}{total}$
  $S = RESAMPLE(\forall s \in S(\hat{x}_{ist}, w_{ist}))$
  $\hat{x}_{it} = ESTIMATE(S)$
  return $(\hat{x}_{it}, S)$
```

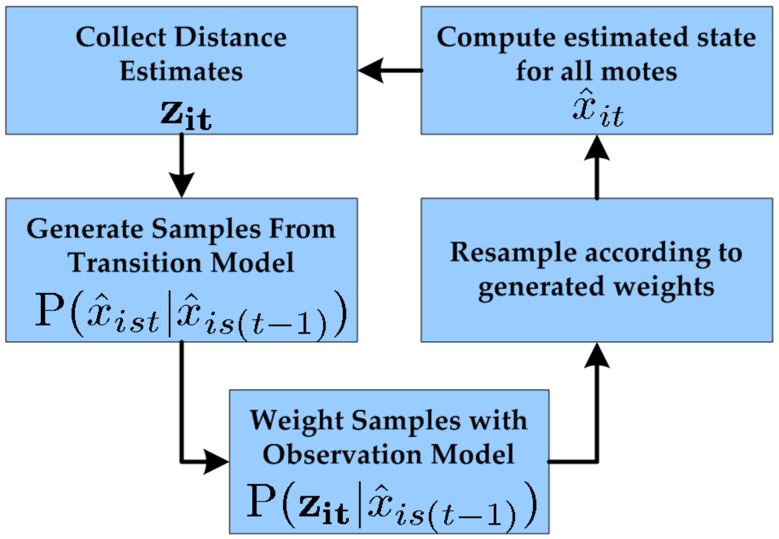
are then advanced according to the transition model, $P(\hat{x}_{ist}|\hat{x}_{is(t-1)})$, which adds Gaussian noise to the (x, y) coordinates of $\hat{x}_{is(t-1)}$. Each sample $\hat{x}_{ist}$ is then assigned a weight using the observation model: $w_{ist} = P(\mathbf{z_{it}}|\hat{x}_{ist})$. This observation model is discussed in greater detail in Section 4.3. These weights are then normalized such that $\sum_{s \in S} w_{ist} = 1$.

The set of samples is then resampled with replacement; [26] outlines a number of resampling algorithms. These new samples are sorted into discretized regions of Euclidean space, and the estimated state of Mote i, $\hat{x}_{it}$, is computed to be the discretized Euclidean location on which the highest number of samples fell. For a simple reference, Figure 4.1 gives a visual depiction of the execution of the algorithm during one iteration, at time t.

4.3 Constructing the Observation Model

4.3.1 Expanded Formulation

It was demonstrated in [5] that the pairwise distance estimates produced by Cricket motes vary depending on the relative angle of the two motes (Figure 4.2 gives a visual



Collect Distance Estimates
$\mathbf{z_{it}}$
Compute estimated state for all motes
$\hat{x}_{it}$
Generate Samples From Transition Model
$P(\hat{x}_{ist}|\hat{x}_{is(t-1)})$
Resample according to generated weights
Weight Samples with Observation Model
$P(\mathbf{z_{it}}|\hat{x}_{is(t-1)})$

Figure 4.1: An overview of one iteration t of the particle filtering algorithm. Pairwise distance estimates $\mathbf{z_{it}}$ are collected, samples are generated by the transition model $P(\hat{x}_{ist}|\hat{x}_{is(t-1)})$, the observation model then assigns each sample a weight $w_{ist} = P(\mathbf{z_{it}}|\hat{x}_{ist})$, and the particles are resampled with replacement using these weights. The output is the estimated state of the Mote $\hat{x}_{it}$.

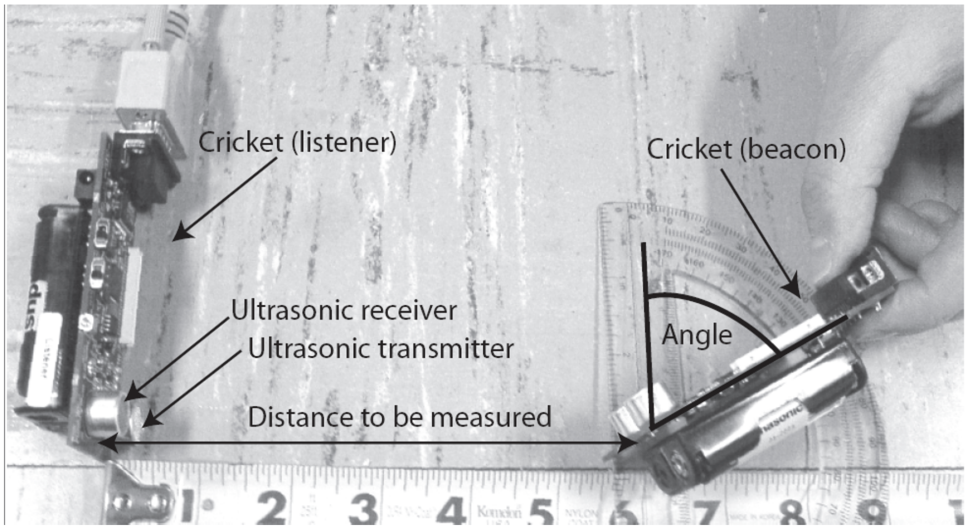


Figure 4.2: As displayed in [5], the authors recorded the distance estimate computed by two Cricket motes based on their relative distance and angle. The mote on the left (listener) calculated the estimated distance based on the time difference of arrival between the ultrasonic pulse and the radio signal emitted by the mote on the right (beacon).

depiction of their experimental setup). The authors compiled data that maps the relative distance and angle of two motes to a set of produced distance estimates. Through an expanded formulation of the observation model, $P(\mathbf{z_{it}}|\hat{x}_{ist})$, this empirical data can be exploited.

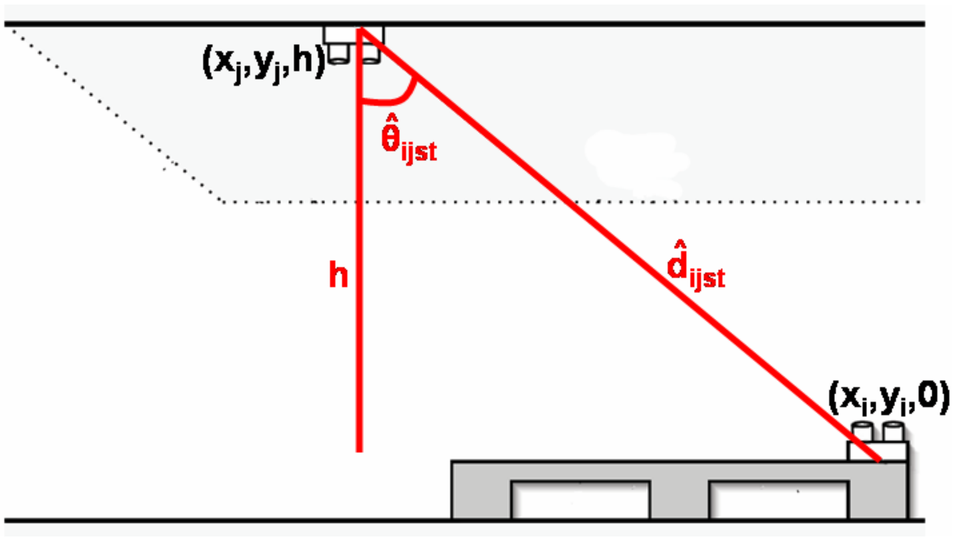


Figure 4.3: A visual depiction of the relative distance $\hat{d}_{ijst}$ and angle $\hat{\theta}_{ijst}$ between a Beacon j and a sample state $\hat{x}_{ist}$. In relation to the experiments carried out as in Figure 4.2, the Beacon j is considered the listener and the sample state $\hat{x}_{ist}$ is considered the beacon. Though the angle $\hat{\theta}_{ijst}$ is not the same angle that is labeled in Figure 4.2, they are geometrically equivalent.

First, it is assumed that the distance estimates produced by the Beacons are probabilistically independent of one another, as stated in Section 3.2.2. Based on this assumption, the observation model can be expressed as follows:

$$P(\mathbf{z_{it}}|\hat{x}_{ist}) \quad = \quad \prod_{j=1}^{m} P(z_{ijt}|\hat{x}_{ist}) \tag{4.5}$$

Because the Euclidean coordinates of all Beacons $j \in B$ are known, as specified in Section 3.2.3, given a sample state $\hat{x}_{ist}$ the relative distance and angle to each individual Beacon j can be computed. Let (x_i, y_i) represent the state $\hat{x}_{ist}$, let (x_j, y_j) represent the Euclidean coordinates of Beacon j, and let h represent the height of the ceiling to which j is mounted. For any given pair $(\hat{x}_{ist}, j)$, the relative distance $\hat{d}_{ijst}$ and angle $\hat{\theta}_{ijst}$ can be computed as follows:

$$\hat{d}_{ijst} \quad = \quad \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + h^2} \tag{4.6}$$

$$\hat{\theta}_{ijst} \quad = \quad \arccos\left(\frac{h}{\hat{d}_{ijst}}\right) \tag{4.7}$$

Figure 4.3 gives a visual depiction of these two values. Because these two aspects of the sample state $\hat{x}_{ist}$ can be computed relative to all Beacons j, the observation model can be

Table 4.3: Notation for the calibration phase, when the various observation models are constructed.

$p \in P$	The set of (d_p, θ_p) positions for which data was collected.
d_p	The relative Euclidean distance between listener and beacon in position p.
θ_p	The relative angle between listener and beacon in position p.
$\mathbf{z_p}$	The vector $[z_{p1}, \ldots, z_{pn}]$ of computed distance estimates between listener and beacon at position p.
μ_p	The mean of the computed distance estimates $\mathbf{z_p}$.
${\sigma_p}^2$	The variance of the computed distance estimates $\mathbf{z_p}$.
μ'_d	The mean of the computed distance estimates $\mathbf{z_p}$ over all positions p such that $d_p = d$.
${\sigma'_d}^2$	The variance of the computed distance estimates $\mathbf{z_p}$ over all positions p such that $d_p = d$.

further expanded from Equation 4.5 as follows:

$$P(\mathbf{z_{it}}|\hat{x}_{ist}) = \prod_{j=1}^{m} P(z_{ijt}|\hat{d}_{ijst}, \hat{\theta}_{ijst}) \quad (4.8)$$

Now in order to compute $P(\mathbf{z_{it}}|\hat{x}_{ist})$ for a given sample state $\hat{x}_{ist}$, the experimental data collected in [5] can be used to generate, for each individual Beacon j and corresponding distance estimate z_{ijt}, the probability that z_{ijt} would have been computed given the relative distance $\hat{d}_{ijst}$ and angle $\hat{\theta}_{ijst}$ of the two Euclidean positions.

Table 4.3 contains notation that will be used during the following discussion of the calibration phase. For each pair of relative distance and angle (d_p, θ_p) for which there was available data, I computed the mean μ_p and variance ${\sigma_p}^2$ of the collected distance estimates $\mathbf{z_p}$. Assuming that the distribution of computed distance estimates for any given pair (d_p, θ_p) is Gaussian, the probability $p(z|d_p, \theta_p)$ for any arbitrary distance estimate z can be computed using the Gaussian PDF:

$$p(z|d_p, \theta_p) = \frac{1}{\sqrt{2\pi{\sigma_p}^2}} \exp\{-\frac{(\mu_p - z)^2}{2{\sigma_p}^2}\} \quad (4.9)$$

However, this PDF can only be explicitly constructed for pairs (d_p, θ_p) for which there is an avilable mean μ_p and variance ${\sigma_p}^2$. In order to construct the observation model, $P(z_{ijt}|\hat{d}_{ijst}, \hat{\theta}_{ijst})$ must compute the probability for any arbitrary pair $(\hat{d}_{ijst}, \hat{\theta}_{ijst})$. A number of such observation models were built around this collected data, as described below in Sections 4.3.2 and 4.3.3.

4.3.2 Inverse Distance Weighting

In [28], Shepard outlines a method for defining a continuous function that provides interpolated values for unknown points given the values for known points. The formulation works in the face of irregularly-spaced data, presenting a continuous surface that fits the known points exactly. This approach, called inverse distance weighting (IDW), uses a weighting function to assign a greater influence to known points that are closer to the interpolated point. Furthermore, only some subset of known points that are the closest to the interpolated point, called the neighborhood, are considered when computing the interpolated value. In the context of the observation model, the value that must be interpolated is $P(z|d,\theta)$ given any arbitrary distance estimate z, distance d, and angle θ. The weighting factor must assign higher prevalence to values for known points $(z|d_p,\theta_p)$ that are closer to the interpolated point $(z|d,\theta)$. Thus, the weighting factor is defined as follows:

$$w(d,\theta,p) = \frac{1}{(\sqrt{(d-d_p)^2+(\theta-\theta_p)^2})^R} \tag{4.10}$$

The value R is a positive real number, called the power parameter, used to define the extent to which the proximity of the known point to the interpolated point dictates its influence. In the constructed observation model, I let $R = 2$, as it is the most commonly used value and, after a brief survey, seemed to provide the most favorable results. Let the value for known points (d_p,θ_p) be defined as the Gaussian PDF as in Equation 4.9. Given the weighting factor in Equation 4.10, the value $P(z|d,\theta)$ for any arbitrary unknown point (d,θ) can be computed as follows:

$$P(z|d,\theta) = \frac{\sum_{p\in N}\frac{p(z|d_p,\theta_p)}{w(d,\theta,p)}}{\sum_{p\in N}\frac{1}{w(d,\theta,p)}} \tag{4.11}$$

The set N refers to the set of closest neighboring points that are considered. In the constructed observation model, I let N be the six points in closest proximity to the interpolated point. In practice, lower values of N provided degraded results and higher values of N increased computation without any noticeable improvement to the results. I refer to this observation model as Asymmetric Weighted. In the context of the particle filtering algorithm, the observation model in Equation 4.8 can be built with the use of the following formula:

$$P(z_{ijt}|\hat{d}_{ijst},\hat{\theta}_{ijst}) = \begin{cases} p(z_{ijt}|\hat{d}_{ijst},\hat{\theta}_{ijst}) & \text{if } (\hat{d}_{ijst},\hat{\theta}_{ijst}) \in P \\ \frac{\sum_{p\in N} \frac{p(z_{ijt}|d_p,\theta_p)}{w(\hat{d}_{ijst},\hat{\theta}_{ijst},p)}}{\sum_{p\in N} \frac{1}{w(\hat{d}_{ijst},\hat{\theta}_{ijst},p)}} & \text{otherwise} \end{cases} \tag{4.12}$$

In order to verify that an asymmetric observation model, taking into account the relative angle of the Mote and Beacon, provides improved accuracy, I constructed a similar model that only considers the relative distance of the Mote and Beacon. Let μ'_d represent the mean of the distance estimates computed when the relative distance between listener and beacon was d, regardless of their relative angle, and let ${\sigma'_d}^2$ represent the variance of those estimates. For known values, the Gaussian PDF $p(z|d_p,\theta_p)$ from Equation 4.9 can be rewritten to only take into account d_p as follows:

$$p(z|d_p) = \frac{1}{\sqrt{2\pi{\sigma'_{d_p}}^2}} \exp\{-\frac{(\mu'_{d_p}-z)^2}{2{\sigma'_{d_p}}^2}\} \tag{4.13}$$

Furthermore, because relative angle is no longer considered, let the weighting function in Equation 4.10 be rewritten as follows:

$$w(d,p) = \frac{1}{(\sqrt{(d-d_p)^2})^R} \tag{4.14}$$

Let the value for known points (d_p,θ_p) be defined with the Gaussian PDF as rewritten in Equation 4.13. Given the weighting factor in Equation 4.14, the value $P(z|d,\theta)$ for any arbitrary unknown point (d,θ) can be computed using Equation 4.11 rewritten as follows:

$$P(z|d,\theta) = \frac{\sum_{p\in N} \frac{p(z|d_p)}{w(d,p)}}{\sum_{p\in N} \frac{1}{w(d,p)}} \tag{4.15}$$

I refer to this observation model as Symmetric Weighted. In the context of the particle filtering algorithm, the observation model in Equation 4.8 can be built with the use of the following formula:

$$P(z_{ijt}|\hat{d}_{ijst},\hat{\theta}_{ijst}) = \begin{cases} p(z_{ijt}|\hat{d}_{ijst}) & \text{if } (\hat{d}_{ijst},\theta) \in P \text{ for any arbitrary } \theta \\ \frac{\sum_{p\in N} \frac{p(z_{ijt}|d_p)}{w(\hat{d}_{ijst},p)}}{\sum_{p\in N} \frac{1}{w(\hat{d}_{ijst},p)}} & \text{otherwise} \end{cases} \quad (4.16)$$

A comparison of the relative accuracy of these two models when executed over real-world data is given in Section 5.3.2.

4.3.3 Directly Modeling Distance Estimates as a Function

Because this algorithm is intended to be run in real-time, any computation that can be done during the calibration phase in order to lighten the load during runtime is preferable. The use of inverse distance weighting requires that the probability $p(z_{ijt}|d_p,\theta_p)$ be computed for multiple pairs (d_p,θ_p) for each individual distance estimate z_{ijt}, because the values for unknown points are interpolated on the fly. In order to reduce the amount of computation done at runtime, the distribution $P(z|d,\theta)$ can be directly modeled as a function of d and θ. Namely, given functions to compute the mean $\mu(d,\theta)$ and variance $\sigma^2(d,\theta)$ for any arbitrary pair (d,θ), $P(z|d,\theta)$ can be computed using the Gaussian PDF:

$$p(z|d,\theta) = \frac{1}{\sqrt{2\pi\sigma^2(d,\theta)}} \exp\{-\frac{(\mu(d,\theta)-z)^2}{2\sigma^2(d,\theta)}\} \quad (4.17)$$

Because the ultrasound signal of the Cricket motes exhibits a cardioid pattern, as depicted in Figure 1.1, it seems logical that the mean of the distance estimates and, consequently, the variance can be modeled as polar functions of the relative distance and angle of the two motes:

$$\mu(d,\theta) = ad\cos\theta + bd\sin\theta + c \quad (4.18)$$

$$\sigma^2(d,\theta) = pd\cos\theta + qd\sin\theta + r \quad (4.19)$$

Using linear regression, the specific values for the coefficients a,b,c,p,q,r that provided the least squared error over all pairs (d_p,θ_p) were computed. That is, the coefficients a,b,c were computed to minimize the following equation:

$$\min \sum_{p\in P} (\mu_p - (ad_p\cos\theta_p + bd_p\sin\theta_p + c))^2 \quad (4.20)$$

Then, the coefficients p, q, r were computed to minimize the following equation:

$$\min \sum_{p \in P} ({\sigma_p}^2 - (pd_p \cos\theta_p + qd_p \sin\theta_p + r))^2 \tag{4.21}$$

Given these coefficients, we can compute $\mu(d,\theta)$ and $\sigma^2(d,\theta)$ for any arbitrary (d,θ). These functions can then be substituted into the Gaussian PDF so that $p(z|d,\theta)$ can be computed for any (d,θ). In the context of the particle filtering algorithm, by making use of Equations 4.18 and 4.19 the observation model in Equation 4.8 can be built with the use of the following formula:

$$P(z_{ijt}|\hat{d}_{ijst}, \hat{\theta}_{ijst}) = \frac{1}{\sqrt{2\pi\sigma^2(\hat{d}_{ijst}, \hat{\theta}_{ijst})}} \exp\{-\frac{(\mu(\hat{d}_{ijst}, \hat{\theta}_{ijst}) - z_{itj})^2}{2\sigma^2(\hat{d}_{ijst}, \hat{\theta}_{ijst})}\} \tag{4.22}$$

This observation model is referred to as Asymmetric Polar.

In order to evaluate the modeling of $\mu(d,\theta)$ and $\sigma^2(d,\theta)$ as polar functions, I constructed a similar observation model with the two formulas modeled as linear functions of (d,θ):

$$\mu(d,\theta) = ad + b\theta + c \tag{4.23}$$

$$\sigma^2(d,\theta) = pd + q\theta + r \tag{4.24}$$

Like in the construction of the Asymmetric Polar model, linear regression was used to compute the specific values for the coefficients a, b, c, p, q, r that provided the least squared error over all pairs (d_p, θ_p), such that the following equations were minimized:

$$\min \sum_{p \in P} (\mu_p - (ad_p + b\theta_p + c))^2 \tag{4.25}$$

$$\min \sum_{p \in P} ({\sigma_p}^2 - (pd_p + q\theta_p + r))^2 \tag{4.26}$$

These newly defined functions $\mu(d,\theta)$ and $\sigma^2(d,\theta)$ can now be plugged into Equation 4.22 in order to construct the observation model. This model is referred to as Asymmetric Linear.

Finally, in order to further validate the efficacy of an asymmetric observation model, taking into account the relative angle of the Mote and Beacon, another observation model was constructed that models both the mean and variance of the distance estimates as linear functions of only the relative distance d between the two motes, ignoring the angle θ:

$$\mu(d) = ad + c \tag{4.27}$$

$$\sigma^2(d) = pd + r \tag{4.28}$$

Again, linear regression was used to compute the specific values for the coefficients a, c, p, r that provided the least squared error over all values d_p, such that the following equations were minimized:

$$\min \sum_{p \in P} (\mu_p - (ad_p + c))^2 \tag{4.29}$$

$$\min \sum_{p \in P} ({\sigma_p}^2 - (pd_p + r))^2 \tag{4.30}$$

The observation model in Equation 4.8 can be built with the use of the following modified version of Equation 4.22, taking into account the different sets of parameters to the newly defined formulas $\mu(d)$ and $\sigma^2(d)$

$$P(z_{ijt}|\hat{d}_{ijst}, \hat{\theta}_{ijst}) = \frac{1}{\sqrt{2\pi\sigma^2(\hat{d}_{ijst})}} \exp\{-\frac{(\mu(\hat{d}_{ijst}) - z_{itj})^2}{2\sigma^2(\hat{d}_{ijst})}\} \tag{4.31}$$

This model is referred to as Symmetric Linear. A comparison of the accuracy of the five described observation models can be found in Section 5.3.2.

4.4 Computational Complexity

Let $|M|$ represent the number of Motes, $|B|$ represent the number of Beacons, and $|S|$ represent the number of samples that each filter generates. Let $F(B, S)$ represent the time needed to run one iteration of the particle filter, given in Algorithm 1, for an individual Mote, and let $T(M, B, S)$ represent the time needed to run one iteration of the entire system. Because there is a separate particle filter being run for each Mote, the following is true:

$$T(M, B, S) = M * F(B, S) \tag{4.32}$$

In order to determine $F(B, S)$, it will be helpful to separate the algorithm into distinct operations as follows. $F1(B, S)$ represents the time needed to compute the distance estimates $\mathbf{z_{it}}$. $F2(B, S)$ represents the time needed to advance the samples $\hat{x}_{ist}$ according to the transition model $P(\hat{x}_{ist}|\hat{x}_{is(t-1)})$. $F3(B,S)$ represents the time needed to assign weights to the samples according to the observation model $P(\mathbf{z_{it}}|\hat{x}_{ist})$. $F4(B,S)$ represents the time needed to normalize the weights w_{ist}. $F5(B,S)$ represents the time needed for resampling. $F6(B,S)$ represents the time needed to compute the estimated state of the Mote $\hat{x}_{it}$. Now, we can define $F(B,S)$ as the following:

$$F(B,S) = F1(B,S) + F2(B,S) + F3(B,S) + F4(B,S) + F5(B,S) + F6(B,S) \quad (4.33)$$

The computation of one distance estimate, z_{ijt}, is a constant-time operation. The computation of all distance estimates $\mathbf{z_{it}}$ relative to Mote i requires that an estimate be computed for all Beacons $j \in B$. Adopting the asymptotic notation defined in [29]:

$$F1(B,S) \in O(|B|) \quad (4.34)$$

The transition model $P(\hat{x}_{ist}|\hat{x}_{is(t-1)})$, which adds Gaussian noise to the Euclidean (x,y) coordinates of $\hat{x}_{is(t-1)}$, is a constant-time operation. Because it is executed once for all samples $s \in S$:

$$F2(B,S) \in O(|S|) \quad (4.35)$$

The assigning of a weight w_{ist} to an individual sample state $\hat{x}_{ist}$ by the observation model $P(\mathbf{z_{it}}|\hat{x}_{ist})$, according to Equation 4.8, requires that $P(z_{ijt}|\hat{d}_{ijst},\hat{\theta}_{ijst})$ is computed for all Beacons $j \in B$. In all of the observation models described above in Sections 4.3.2 and 4.3.3, that computation is a constant-time operation, so the assignment of an individual weight has order $|B|$ complexity. Beacause a weight w_{ist} is computed for all samples $s \in S$:

$$F3(B,S) \in O(|B| * |S|) \quad (4.36)$$

In order to normalize the weights, each individual weight w_{ist} must be divided by the sum of all weights, $\sum_{s \in S} w_{ist}$. These two operations, $sum = \sum_{s \in S} w_{ist}$ and $\forall s \in S, w_{ist} = \frac{w_{ist}}{sum}$, both have order $|S|$ complexity. Because they are each carried out once:

$$F4(B,S) \in O(|S|) \quad (4.37)$$

The process of resampling, as noted in [26], has $O(N)$ complexity, where N is the number of samples involved in the resampling procedure. So, in the context of this algorithm:

$$F5(B, S) \in O(|S|) \tag{4.38}$$

Finally, to compute the estimated state of the Mote, $\hat{x}_{it}$, the algorithm must examine each sample and assign it to a discretized region, then determine which discretized region contained the highest number of sample states. The assigning of an individual sample $\hat{x}_{ist}$ to a region is a constant-time operation, so the assignment for all samples $s \in S$ has order $|S|$ complexity. Determining the most highly populated region, which involves finding the max of no more than $|S|$ values, has order $|S|$ complexity. Because each of these operations is carried out once:

$$F6(B, S) \in O(|S|) \tag{4.39}$$

Now that the intermediate functions have been computed, Equation 4.33 can be rewritten as follows:

$$F(B, S) \in O(|B|) + O(|S|) + O(|B| * |S|) + O(|S|) + O(|S|) + O(|S|) \tag{4.40}$$

By the properties of asymptotic notation:

$$F(B, S) \in O(\max(|B|, |S|, (|B| * |S|)) \tag{4.41}$$

It is obvious that the maximum term must be $(|B| * |S|)$, so Equation 4.41 can be simplified to the following formula:

$$F(B, S) \in O(|B| * |S|) \tag{4.42}$$

Finally, the overall running time of the proposed system $T(M, B, S)$ can be expressed by rewriting Equation 4.32 as follows:

$$T(M, B, S) \in O(|M| * |B| * |S|) \tag{4.43}$$

Because the number of Motes $|M|$ is determined entirely independently of the system, the parameters that can be tuned to aid performance are the number of deployed Beacons

$|B|$ and the number of generated samples $|S|$. In Section 5.2.1, I explore these parameters and the resulting tradeoffs between running time and accuracy. The experimental results support this running time analysis.

Chapter 5

Experimental Results

5.1 Building the Simulator

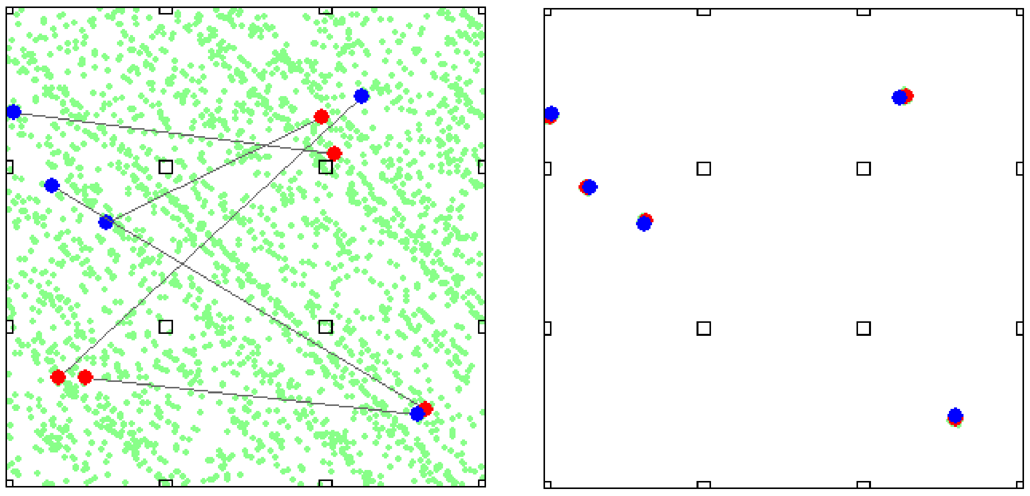
5.1.1 Simulator Environment

In order to efficiently evaluate this algorithm, I built a simulator in Java that is able to be run using either computer-generated or real-world data. A number of different parameters may be specified, as outlined in Table 5.1.

The simulator is made up of ∼2,400 lines of unoptimized Java code. I also crafted a visualization interface, the output of which is shown in Figure 5.1, using the Java Applet class. For all experiments described in Sections 5.2 and 5.3, the simulator was compiled using *javac* and run using JRE 1.5.0.01. The experiments were run on the UC Berkeley X Cluster, a set of 42 IBM xSeries 330 machines each equipped with two 1.0 GHz Intel Pentium III CPUs and 1.5GB ECC PC133 SDRAM, running Debian GNU/Linux 3.0. It is important to note that these are shared machines, so the total available resources were not allotted to the simulator during execution.

Table 5.1: Parameters that may be specified in the Java simulator.

roomDimenstions	The length, width, and height of the room.
moteConfiguration	The number, location, and movement pattern of all Motes $i \in M$. The movement patterns may either be formulaic, or specifically charted to model real-world Mote movements.
beaconConfiguration	The number and location of all Beacons $j \in B$.
distanceEstimates	The distance estimates $\forall i \in M \mathbf{z_{it}}$ that are computed by the Beacons at each iteration. These estimates can either be generated by some formula, or specified to match collected real-world distance estimates.
numSamples	The number of sample states $\|S\|$ generated by each individual particle filter ($\|S\|$ samples generated for each Mote $i \in M$).
transitionModel	The transition model $P(\hat{x}_{ist} \| \hat{x}_{is(t-1)})$.
observationModel	The observation model $P(\mathbf{z_{it}} \| \hat{x}_{ist})$.



(a) Initial Distribution

(b) After One Iteration

Figure 5.1: The Java Applet visualization tool for the simulator, showing a top-down view of the room. The true location of the Motes are shown in blue, and the estimated locations are shown in red, with a line connecting the sample state to the relevant Mote. The green dots represent all generated sample states, after the resampling step is completed. The Beacons are represented by squares. (a) shows the state of the system immediately after initialization, when the sample states are distributed uniformly at random. (b) shows the system after one iteration of Algorithm 1 has completed. Though difficult to see, the sample states are still present in the visualization in (b), they are just highly concentrated below the dots representing the true and estimated Mote states – because the blue and red dots are larger than the green dots, the samples are almost completely hidden from view.

5.1.2 Simulator Overview

The simulator runs independent of any notion of real time; instead, it is modeled around the abstract concept of iterations. Due to this, when it is executed using real-world data that is time-sensitive, such as mote movement patterns or generated distance estimates, that data is also abstracted into iterations. This approach made it so that carefully controlled experiments could be carried out and analyzed, as described in Section 5.3.

At initialization, the room dimensions are specified, along with the locations of all Beacons $j \in B$ and the starting locations of all Motes $i \in M$. The initial set of sample states, $\forall s \in S\{\hat{x}_{is0}\}$, is generated for all Motes $i \in M$. These sample states are distributed uniformly at random within the specified room dimensions.

The simulator execution flow is essentially the same as the overview of the particle filtering algorithm shown in Figure 4.1. At the start of each iteration t, the distance estimates $\mathbf{z_{it}}$ are generated for all Motes $i \in M$. To compute the distance estimates, a user-defined Java class must be supplied that implements the following function:

$$z_{ijt} \quad = \quad estimate(x_{it}, y_{it}, x_j, y_j) \tag{5.1}$$

Where (x_{it}, y_{it}) are the Euclidean coordinates x_{it} of Mote i at time t, and (x_j, y_j) are the Euclidean coordinates of Beacon j. This allows the class to compute a distance estimate based on the relative distance and angle between (x_{it}, y_{it}) and (x_j, y_j). If the supplied class is configured to use distance estimates collected from actual data, the function given in Equation 5.1 simply returns the recorded distance estimate computed by Beacon j, relative to Mote i, at time t. It is possible that, for a given pair of Mote i and Beacon j, no distance estimate is generated due to the ultrasound signal limit. In that case, it is simply not included in the computation of w_{ijst} for any samples $\hat{x}_{it}$, leaving the computation of w_{ijst} to be solely defined by those estimates that were actually generated. In the case that, for some Mote i, the set of distance estimates $\mathbf{z_{it}}$ is empty, its estimated location $\hat{x}_{it}$ will be equal to a randomly chosen sample state $\hat{x}_{ist}$ for some $s \in S$.

Next, the set of sample states is regenerated from the specified transition model

$P(\hat{x}_{ist}|\hat{x}_{is(t-1)})$. As with the distance estimates, the transition model is supplied by a user-defined class that implements the following function:

$$(x_{it}, y_{it}) \quad = \quad transition(x_{i(t-1)}, y_{i(t-1)}) \tag{5.2}$$

Where $(x_{i(t-1)}, y_{i(t-1)})$ are the Euclidean coordinates $x_{i(t-1)}$ of Mote i at time $t-1$. Next, the samples are weighted according to the observation model $P(\mathbf{z_{it}}|\hat{x}_{ist})$. Again, the observation model is supplied by a user-defined class that implements the following function:

$$w_{isjt} \quad = \quad weight(z_{ijt}, \hat{d}_{ijst}, \hat{\theta}_{ijst}) \tag{5.3}$$

Where w_{ijst} is the value $P(z_{ijt}|\hat{d}_{ijst}, \hat{\theta}_{ijst})$, as specified in Equation 4.8. The set of generated samples is then resampled with replacement, and the estimated state $\hat{x}_{it}$ is generated for all Motes $i \in M$.

At this point the output is generated, both by writing to a log file and, if running the visualization interface, to the GUI as depicted in Figure 5.1. The iteration number then increments from t to $t+1$, and the process begins again.

5.2 Analysis of Different Configuration Parameters

5.2.1 Running Time as a Function of Configuration

Because the simulator operates independent of a notion of real time, it executes each iteration as soon as the previous one completes. Therefore, the time spent executing the simulator indicates the maximum speed at which it can be run, giving a notion of the rate at which it would be able to accept incoming distance estimates if it were deployed in a real-time situation. By examining the average time taken per Mote i per iteration t, I am able to explore how modifying the system parameters affects its ability to run in real time.

The experiments described in this section were all run using the Asymmetric Linear observation model. Each test was run with 10 Motes, with random movement patterns: at each iteration t, the state x_{it} of Mote i was generated by adding Gaussian noise to the coordinates $(x_{i(t-1)}, y_{i(t-1)})$ of the previous state $x_{i(t-1)}$ (in cm):

$$x_{it} = (x_{i(t-1)} + random1, y_{i(t-1)} + random2) \quad (5.4)$$

Where $random1$ and $random2$ are numbers pulled from a Gaussian distribution with $\mu = 0$ and $\sigma = 10$. The room was of dimensions $1000cm$ by $1000cm$, with a ceiling height of $240cm$. The transition model $P(\hat{x}_{ist}|\hat{x}_{is(t-1)})$ used the same formula as for the Mote movement pattern generation, as given in Equation 5.4. The distance estimates $\mathbf{z_{it}}$ were generated using the same probability distribution as the observation model: given the relative distance d_{ijt} and angle θ_{ijt} between Mote i and Beacon j at time t, the distance estimate z_{ijt} is drawn from a Gaussian distribution with $\mu = \mu(d_{ijt}, \theta_{ijt})$ and $\sigma = \sqrt{\sigma^2(d_{ijt}, \theta_{ijt})}$, using the functions defined in Equations 4.23 and 4.24. Also, if the values for d_{ijt} and θ_{ijt} represent a point at which the ultrasound signal would be out of range due to the cardioid sensitivity pattern as depicted back in Figure 1.1, no distance estimate is generated.

In order to evaluate the effect of the number of generated samples $|S|$ on running time, I held the number of Beacons $|B|$ constant at 36, laid out in a evenly-spaced grid with a Beacon placed every 180 cm. The number of generated particles for each Mote $i \in M$ varied within the range $(0, 10000)$, in increments of 100. The time spent is broken down into number of seconds per Mote i per iteration t. This makes it possible to compute, for the given physical configuration, the maximum number of samples that can be generated while running under given real-time constraints. The results of these experiements are shown in Figure 5.2. The standard deviations for each value were ≈ 0, so they were not drawn into the chart.

As shown in Figure 5.2, the amount of time spent per Mote i per iteration t seems to scale linearly with the number of generated samples $|S|$ when the number of Motes $|M|$ and Beacons $|B|$ are held constant. This is consistent with the analysis in Section 4.4. Furthermore, the system is able to generate up to 6500 samples while processing the state of one Mote i at one-second granularity.

In order to evaluate the effect of the number of Beacons $|B|$ on running time, I held the number of generated samples $|S|$ constant at 1000 per Mote $i \in M$. The number of

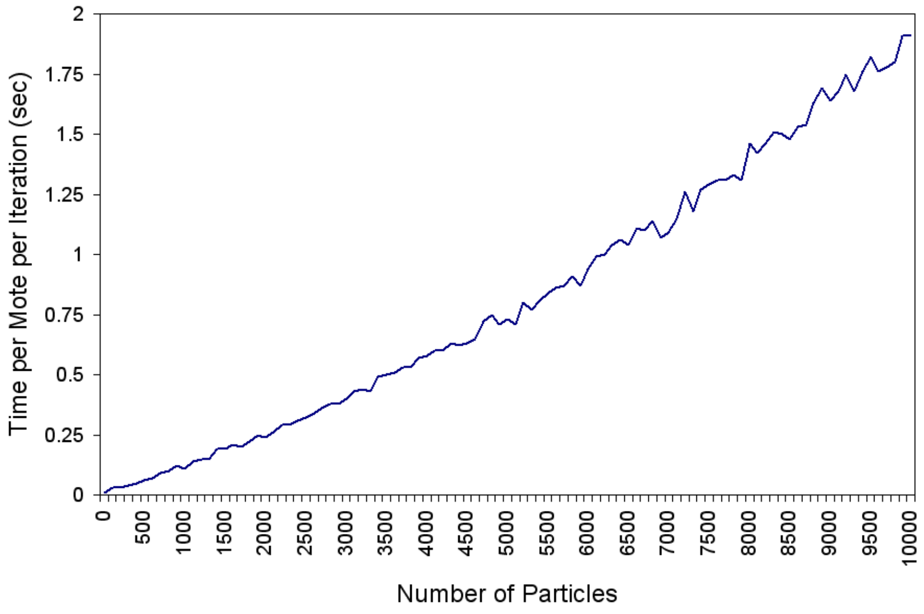


Figure 5.2: Running time as a function of the number of generated samples $|S|$. $|S|$ ranges from 0 to 10000, in increments of 100. The chart gives the time spent per Mote i per iteration t. The standard deviations for each value were ≈ 0, so they were not drawn into the chart. As depicted, for the chosen range of values the running time scales linearly with regard to $|S|$.

Beacons, randomly spaced within the room boundaries, varied within the range $(0, 500)$, in increments of 10. Again, the time spent is broken down into number of seconds per Mote i per iteration t. The results of these experiements are shown in Figure 5.3. Again, the standard deviations for each value were ≈ 0, so they were not drawn into the chart.

As shown in Figure 5.3, the amount of time spent per Mote i per iteration t also seems to scale linearly with the number of Beacons $|B|$, which is again consistent with the analysis in Section 4.4. While generating 1000 samples, the system is able to process readings from up to 360 deployed Beacons while processing the state of one Mote i at one-second granularity.

5.2.2 Error as a Function of Physical Configuration

In order to decide on a reasonable physical configuration, I ran a series of tests to evaluate how error is affected by the physical setup: the distribution of Beacons on the ceiling, and the height of the ceiling. As the output of the system, as defined in Section 3.2.4, is the estimated Euclidean coordinates of each Mote i at each time t, I measured error as

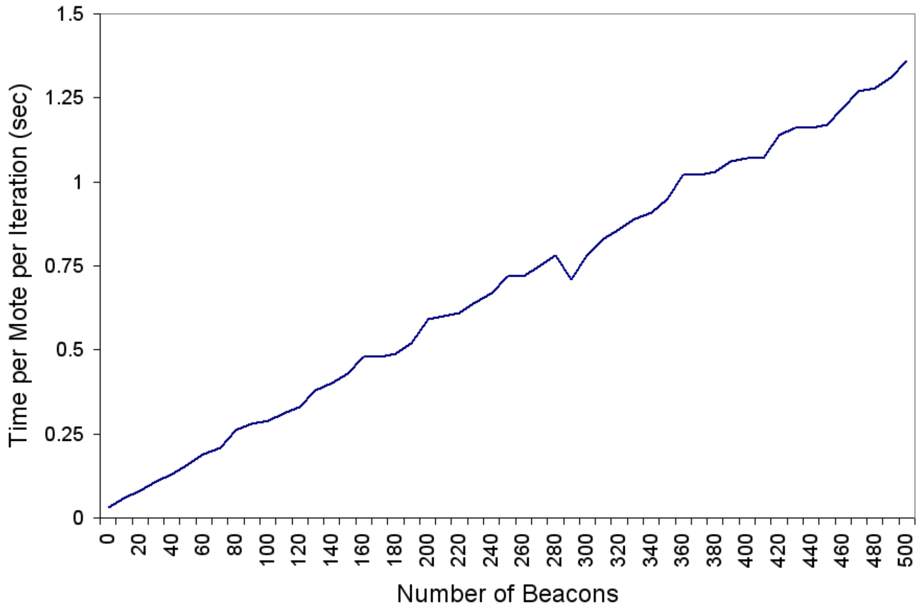


Figure 5.3: Running time as a function of the number of Beacons $|B|$. $|B|$ ranges from 0 to 500, in increments of 10. The chart gives the time spent per Mote i per iteration t. The standard deviations for each value were ≈ 0, so they were not drawn into the chart. As depicted, for the chosen range of values the running time scales linearly with regard to $|B|$.

the average Euclidean distance between the estimated state $\hat{x}_{it}$ and actual state x_{it} over each Mote $i \in M$ for all iterations $t \in T$:

$$error = \frac{\sum_{t\in T}\sum_{i\in M}\sqrt{(x'_{it} - \hat{x'}_{it})^2 + (y'_{it} - \hat{y'}_{it})^2}}{|T| * |M|} \tag{5.5}$$

Where (x'_{it}, y'_{it}), are the Euclidean coordinates of the actual state of Mote i, x_{it}, and $(\hat{x'}_{it}, \hat{y'}_{it})$, are the Euclidean coordinates of the estimated state of Mote i, $\hat{x}_{it}$.

The experiments described in this section were also run using the Asymmetric Linear observation model. The number and movement pattern of the Motes $i \in M$, the transition model $P(\hat{x}_{ist}|\hat{x}_{is(t-1)})$, and the room dimensions were the same as in the tests above in Section 5.2.1. The distance estimates $\mathbf{z_{it}}$ were generated in a similar way, except that the functions for $\mu = \mu(d_{ijt}, \theta_{ijt})$ and $\sigma = \sqrt{\sigma^2(d_{ijt}, \theta_{ijt})}$ were as defined in Equations 4.18 and 4.19, for the Asymmetric Polar observation model. This was done because it is slightly too optimistic to believe that distance estimates computed in real deployments would exactly mirror the expectations built into the observation model.

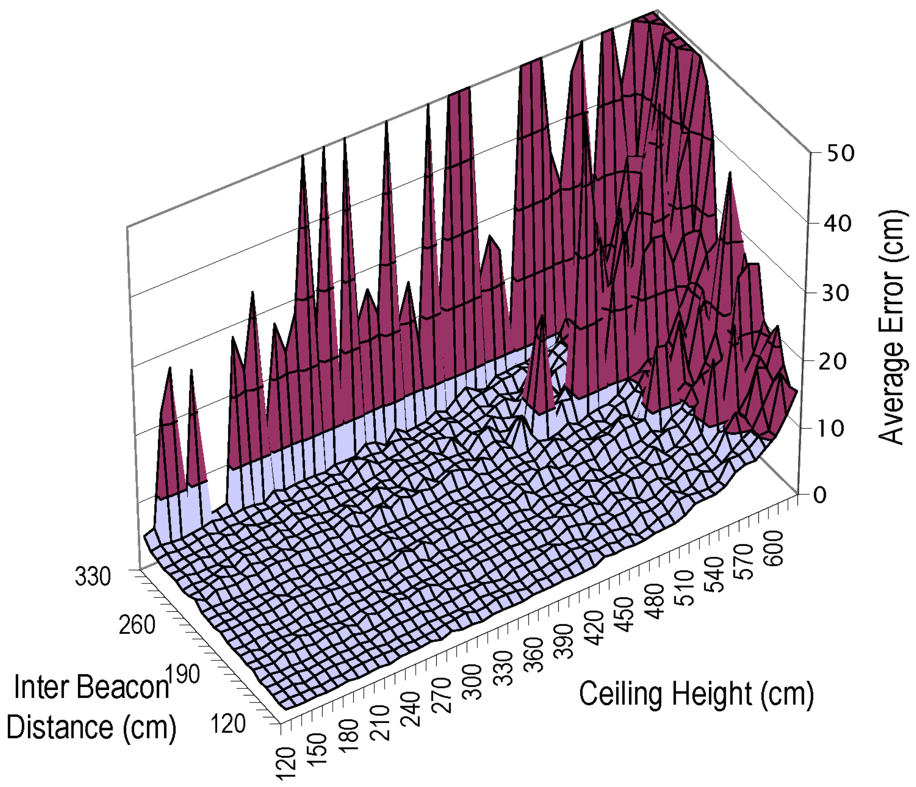


Figure 5.4: Error as a function of Beacon configuration (specified by inter-Beacon distance) and ceiling height. The light blue area shows a reasonable configuration zone, wherein the average error is less than 10 cm.

In the following tests, all parameters were held constant except for the ceiling height and Beacon configuration. The height of the ceiling (specified in *cm*) varied within the range $(120, 640)$, in increments of 10. The Beacons were laid out in an evenly-spaced grid, with the inter-Beacon distance (also specified in *cm*) varying within the range $(120, 350)$, in increments of 10. Figure 5.4 shows the average error, as defined in Equation 5.5, in terms of these two parameters. The light blue area represents those configurations for which the average error was less than 10 cm, an acceptable level of error given the application constraints. As the figure shows, there is a sharp error threshold relating to both inter-Beacon distance and ceiling height.

In order more clearly see these error thresholds, I charted the error while holding one of the two parameters constant. Figure 5.5 shows how error relates to inter-Beacon distance while the ceiling height is held constant at 240 cm. Figure 5.6 shows how error relates to ceiling height while the inter-Beacon distance is held constant at 180 cm.

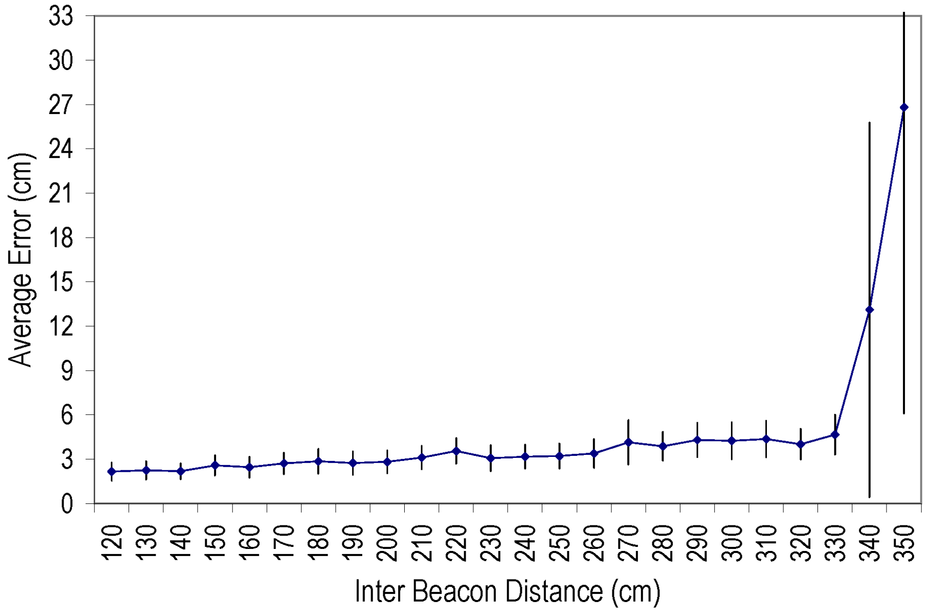


Figure 5.5: Error as a function of Beacon configuration (specified by inter-Beacon distance), with standard deviations shown. In these experiments, ceiling height was held constant at 240 cm. There is a sharp threshold of error when inter-Beacon distance exceeds 330 cm.

With ceiling height held at a constant 240 cm, the measured error increases very slightly with inter-Beacon distance, as depicted in Figure 5.5. However, when inter-Beacon distance increases beyond 330 cm, the error drastically increases. Along with a jump in average error, the standard deviation of that error significantly rises.

With inter-Beacon distance held at a constant 180 cm, the measured error increases slightly with ceiling height, as depicted in Figure 5.6. The error beings to increase more rapidly around the point when ceiling height is equal to 500 cm, but as with increasing inter-Beacon density there is a dramatic rise when ceiling height reaches a particular value. In this case, the threshold value is 590 cm, where the average error and standrad deviation of error both drastically jump. These two thresholds exist because they represent the points at which the ultrasound signal begins to reach its limit, leaving many Motes within range of fewer than three Beacons and unable to be accurately localized. For the points close to the threshold, some subset of the Motes are being accurately localized while others are not within range of enough Beacons, which is why there are large standard deviations in error but the average error is still within 36 cm. However, the points that are not depicted in

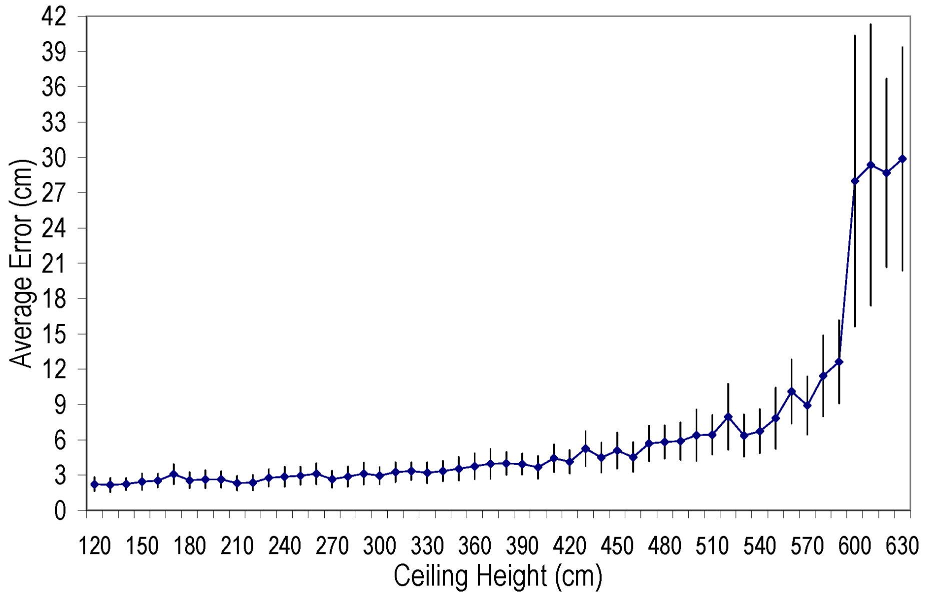


Figure 5.6: Error as a function of ceiling height, with standard deviations shown. In these experiments, inter-Beacon distance was held constant at 180 cm. There is a sharp threshold of error when ceiling height exceeds 590 cm.

the graph have average errors in the hundreds; in those cases, either the ceiling height or inter-Beacon distance have been increased to a point such that a majority of the Motes are out of range of the Beacons, leading their estimated locations to be no more than random guesses.

These results suggest that, so long as the Beacons are located in such a way that the Motes can communicate with a large enough subset to be localized (3 Beacons are sufficient to localize a Mote), the error remains stable.

5.3 Simulation Results Using Real-World Data

5.3.1 Collecting the Data

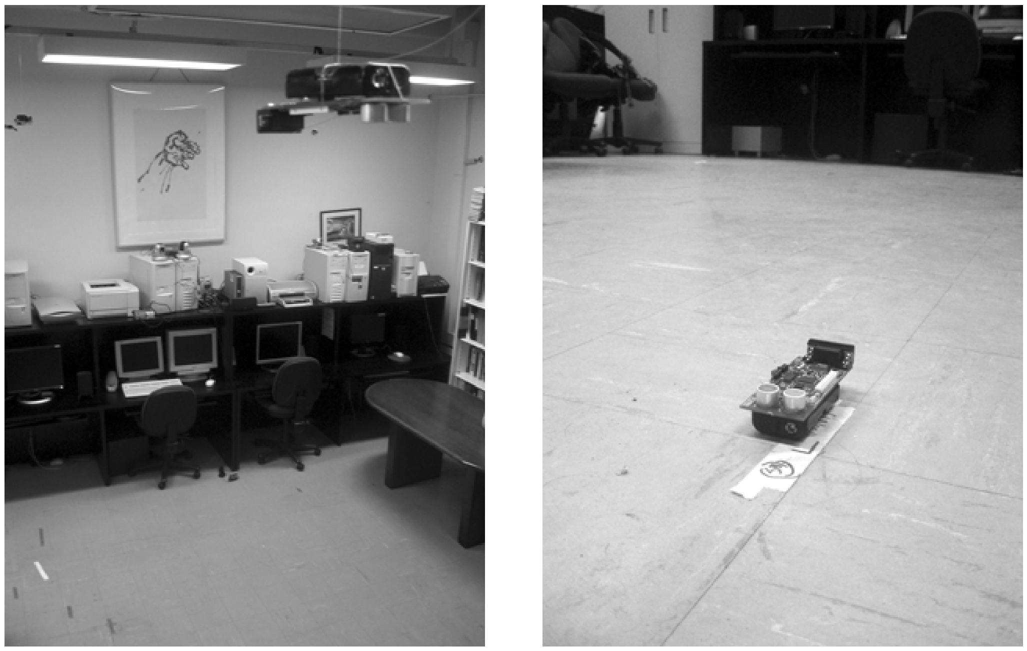
In order to verify the effectiveness of this solution, I collected distance estimates from actual Cricket motes and ran them through the simulator. The following experiments were run using MCS410CA motes (Cricket v2) running TinyOS v1.1.7. One mote was used as a

Mote on the floor, six were affixed to the ceiling and used as Beacons, and another was used as a bay station in order to collect and record data. The portions of the Cricket software (release 2.3.0) involved with the computation of distance estimates, namely the parts that emit the ultrasound and radio signals, and the parts that compute distance estimates based on those received signals, was directly copied so that the estimates would not be changed.

The Mote was programmed to emit an ultrasound pulse and radio message once per iteration, as defined by an internal counter. The radio message contained the iteration number, which was a monotonically increasing integer. Upon receipt of the ultrasound pulse and radio message, the Beacons were programmed to compute the pairwise distance estimate and then broadcast it, along with the iteration number and their own Beacon number. These messages contained information so that they could be differentiated from the messages sent out by the Mote on the floor. The final Cricket mote, connected to a computer through the serial port, recorded the iteration number, Beacon number, and distance estimate, and appended the information to a file. This file was used later to process the results using the simulator.

The Beacons were installed in a grid on the ceiling, $\sim$274 cm above the floor, with an inter-Beacon distance of $\sim$198 cm. They were situated parallel to the floor with the ultrasound sensors facing downward. Because they were suspended by wire, rotation parallel to the plane of the floor could not be controlled. Rotation in the other two dimensions, tilt and yaw, were avoided. Looking at the diagram in Figure 4.3, however, rotation parallel to the floor has no effect on the relative distance and angle, and is therefore negligible. A series of points were marked on the floor, one for each iteration. After each pulse, the Mote was moved to the next point and left stationary until after the next pulse was transmitted, to approximate the effect of continuous motion. The Mote was programmed to emit an ultrasound pulse and radio message every 10 seconds, in order to provide enough time for me to move it to the next location and get out of the way. Because the simulator has no notion of real time, the fact that 10 seconds elapsed between each pulse is irrelevant. Figure 5.7 gives photographic documentation of this physical setup.

A trajectory of points was marked along the floor, and the Mote was moved along the



(a) Beacons on Ceiling (b) Mote on Floor

Figure 5.7: Photographic documentation of the physical setup used for data collection. (a) shows a Beacon affixed to the ceiling parallel to the floor, with the ultrasound sensor facing downward. (b) shows the Mote on the floor, with the ultrasound sensor facing upward.

trajectory over the course of 30 iterations. At each iteration, the Mote emitted an ultrasound pulse and radio message, all Beacons within ultrasound range computed a distance estimate and sent it to the bay station, and those distance estimates were recorded. The Mote was moved along the exact same trajectory nine times, in order to provide enough data for reasonably reliable analysis. The results presented below in Section 5.3.2 are based on these nine sets of collected distance estimates. It is important to note that some of the reported error is due to experimental design, because the exact Euclidean coordinates of the ultrasound sensors were difficult to capture, especially when they were suspended nine feet above the ground.

Figure 5.8 shows the trajectory of the Mote being tracked, as well as the estimated trajectory by one of the simulator runs. The displayed trajectory is the one with overall error that most closely resembled the average error for that particular observation model, Asymmetric Polar.

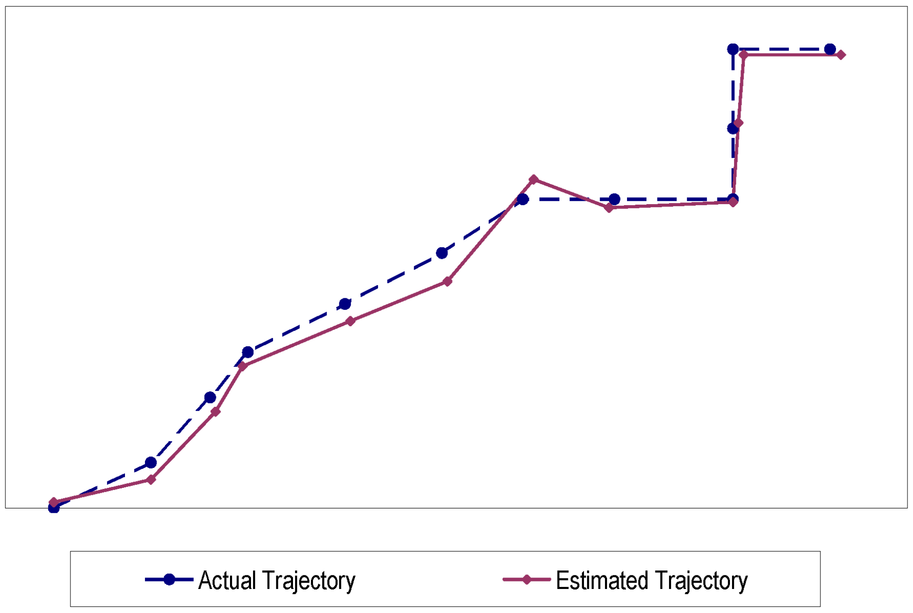


Figure 5.8: A top-down view of the Mote being tracked based on real-world data. The dimensions of the room are 450 cm x 250 cm. The observation model being used is the Asymmetric Polar model. The plot shows the trajectory of the one simulator run with overall error most closely resembling the average error as reported in Table 5.2.

5.3.2 Relative Accuracy of the Various Observation Models

Given the sets of collected distance estimates $\mathbf{z_{1t}}$ and the Euclidean coordinates of the Mote x_{1t} at each iteration $t \in \{1, \ldots, 30\}$, along with the Euclidean coordinates of the Beacons and the dimensions of the room, I executed a number of simulation runs using the five different observation models defined in Section 4.3. For the following experiments, the same transition model decribed by Equation 5.4 was used, and there were 1000 samples generated. The physical configuration, Mote movement patterns, and generated distance estimates were all dictated by the collected data. For each of the five observation models, the simulator was run over each of the nine sets of data 50 different times. All of the charts display the error as defined in Equation 5.5.

First, I compared the performance of the two models defined in Section 4.3.2, Asymmetric Weighted and Symmetric Weighted. Figure 5.9 charts the average error of each observation model over the nine physical runs. Over all sets of data, Asymmetric Weighted performs strictly better than Symmetric Weighted. This indicates that considering the relative angle

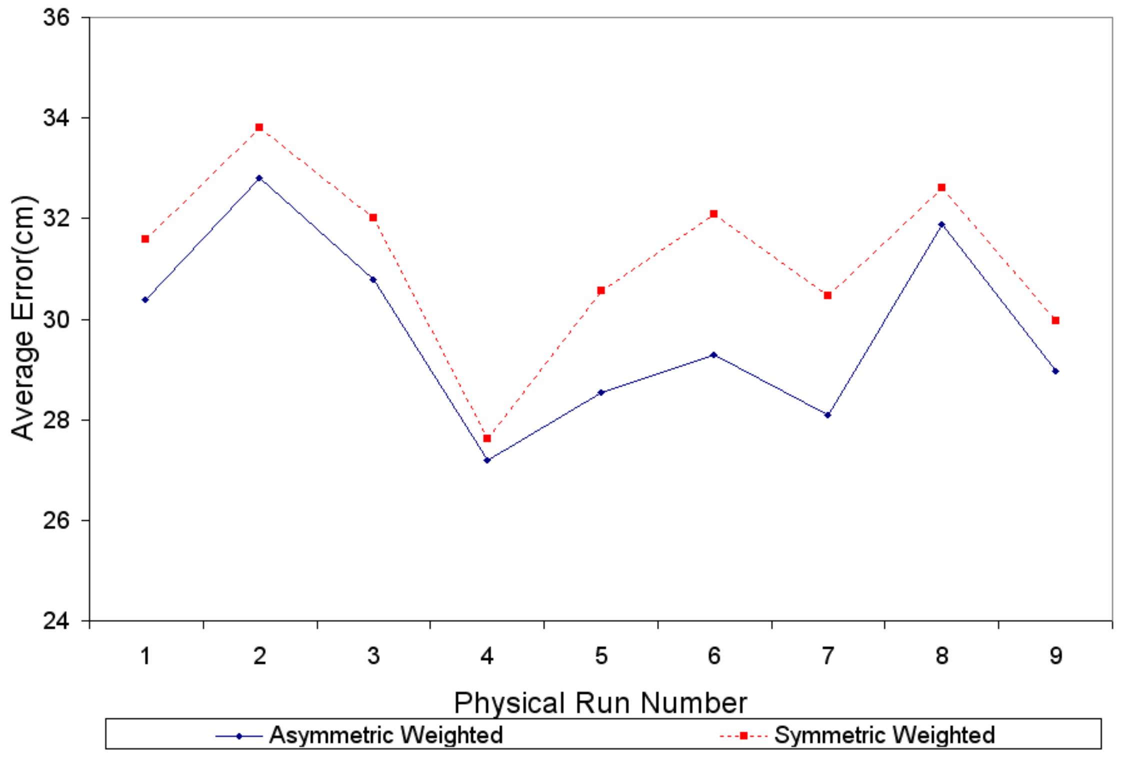


Figure 5.9: Average error of the different models using inverse distance weighting, Asymmetric Weighted and Symmetric Weighted. The model that takes relative angle into account, Asymmetric Weighted, peforms strictly better.

between the Mote and Beacon has a positive effect on the accuracy with which the Mote is localized.

Next, I compared the performance of the three models defined in Section 4.3.3, Asymmetric Polar, Asymmetric Linear, and Symmetric Linear. Figure 5.10 charts the average error of each observation model over the nine physical runs. Over all sets of data, Symmetric Linear performs strictly worse than the other two models. This provides further indication that considering the relative angle between the Mote and Beacon has a positive effect on the accuracy with which the Mote is localized. Though not strictly more accurate, Asymmetric Polar has an average error that is lower than Asymmetric Linear, leading me to believe that modeling the distance estimate as a polar function instead of as a linear function more closely resembles the ultrasound signal's cardiod pattern.

Finally, in order to compare the accuracy of the two approaches to building the obser-

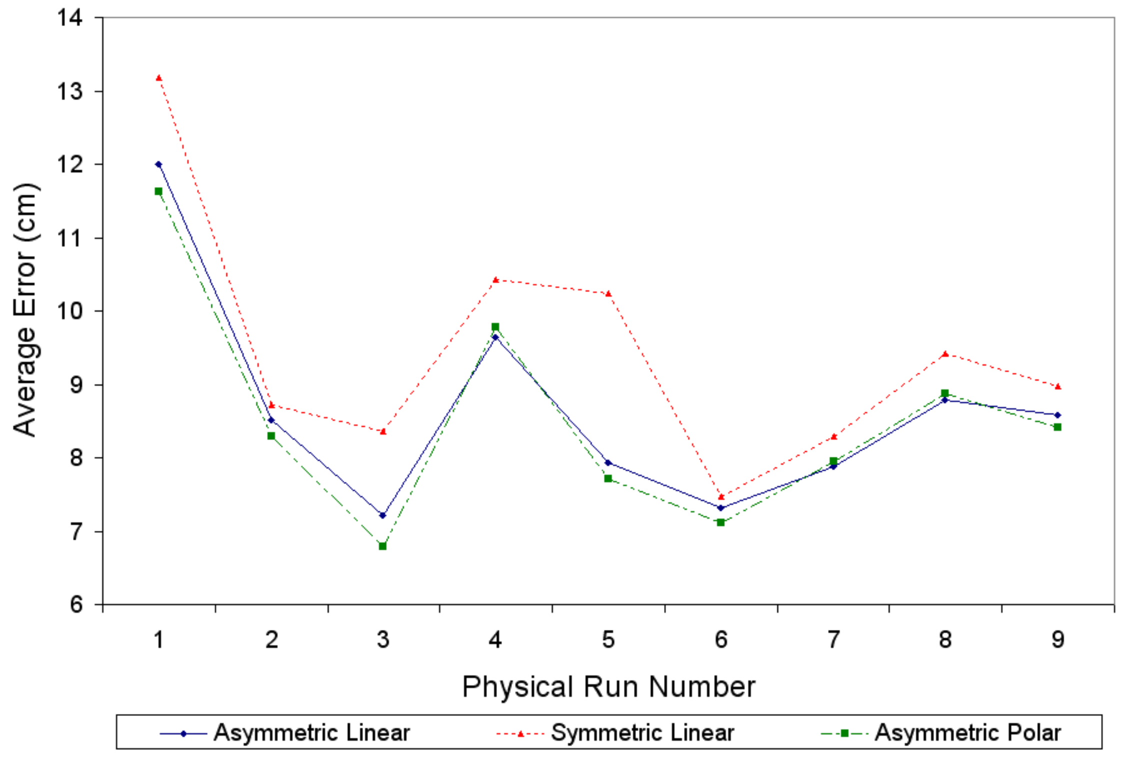


Figure 5.10: Average error of the different models that formulate the expected distance estimate as a function, Asymmetric Polar, Asymmetric Linear, and Symmetric Linear. The two models that takes relative angle into account, Asymmetric Polar and Asymmetric Linear, peform strictly better than Symmetric Linear. Though not strictly better, Asymmetric Polar outperforms Asymmetric Linear on average.

vation model as outlined in Sections 4.3.2 and 4.3.3, I compared the accuracy of the most effective model from each of the two categories. The results of this comparison are shown in Figure 5.11.

The approach outlined in Section 4.3.3, directly modeling the computed distance estimates as a function of relative distance and angle, far outperforms the use of inverse distance weighting. Furthermore, this approach requires less computation during runtime, as a large amount of analysis is done during the calibration phase. The average error and standard deviation of error over all nine physical runs is given in Table 5.2. Based on the given results, by directly formulating distance estimates as a function as opposed to interpolating through the use of inverse distance weighting, accuracy can be improved by an astonishing

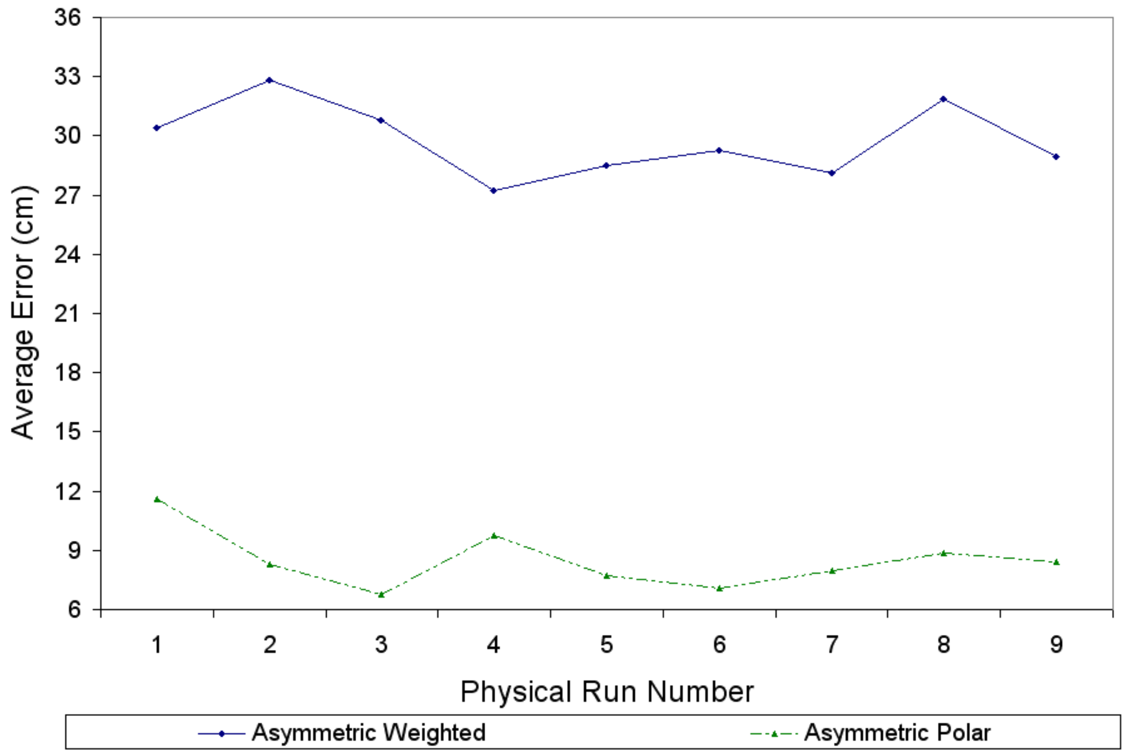


Figure 5.11: Average error of the Asymmetric Polar and Asymmetric Weighted models. Asymmetric Polar outperforms Asymmetric Weighted over all physical runs by at least 15 cm.

Table 5.2: Error of all five observation models over all sets of physical data.

Observation Model	Average Error	Standard Deviation of Error
Asymmetric Weighted	29.88180527	17.51069478
Symmetric Weighted	31.18660861	17.12334361
Asymmetric Polar	8.502053787	8.842719653
Asymmetric Linear	8.648593466	8.900444077
Symmetric Linear	9.116456309	9.634874106

71.5%, and by taking into account the relative angle between the Mote and Beacon accuracy can be improved further by up to 6.7%.

Using the Asymmetric Polar observation model, the moving Mote was able to be localized to within the acceptable range of 10 cm, as depicted in Figure 5.8. Furthermore, according to the data presented in Figure 5.2, because 1000 samples were used this tracking could have occurred while collecting distance estimates from the Beacons as much as once every $\sim$0.12 seconds. This is more than adequate for real-time tracking. Based on the collected data, the system would be able to accurately localize up to eight Motes, to within an average of 8.5 cm, with one-second time granularity. Moreover, these times are based on an implementation written in Java without taking efficiency into account, and executing Java bytecode through a Java virtual machine. It is very likely that the running times could be improved even further by fine-tuning the code and by compiling the application to machine language.

Chapter 6

Future Enhancements

6.1 Distributed Computation

Though the proposed system has proven to be able to localize Motes in real time, in a very large deployment it may become infeasible. The certralized architecture may not scale well as the warehouse becomes larger and contains more pallets. The distribution of the computation would make the system much more scalable. Furthermore, it would allow for the Motes and Beacons to be entirely self-contained, negating the need to relay all distance estimates to a central processor.

Along with distributing the computation, integrating communication between Motes would allow for Motes to be localized when they are not able to communicate with Beacons, whether due to the limitations of the ultrasound signal pattern or physical obstructions blocking the sound. As an additional benefit, implementing peer-to-peer communication between the Motes would allow the spatial constraints to be tracked while the pallets are in transit, when there are no pre-installed Beacons.

This presents a challenge in that each individual Mote would need to be localized with an incomplete set of information. Also, the processing power would be limited to that of the deployed Cricket motes, reducing the number of samples that may be generated and potentially degrading the running time of the algorithm.

6.2 Using Knowledge about the Physical Environment

Because the system is designed to be deployed in a warehouse, in a space of known dimensions with known physical characteristics, taking these characteristics into account could prove to be beneficial. For instance, incorporating the location of any obstructions in the room would allow the system to better handle situations when a Mote is unable to communicate with one or more Beacons. Also, taking into account which areas of the room are designated for storage, as opposed to aisles, would allow the system to make inferences as to how likely it is that a particular Mote is moving based on its location.

This additional information could be incorporated into both the transition model, used to generate sample states, and the observation model, used to determine the likelihood of produced distance estimates. Because particle filtering allows for arbitrary probability distributions for both models, very specialized models can be developed that take different environmental factors into account. These models could even incorporate real-world knowledge about the movement pattern of the pallets, such as specific times of the day when there are no human operators present and, therefore, the pallets will remain stationary.

6.3 Allowing Motion in Three Dimensions

The current system works under the assumption that all Motes are located on a single plane, parallel to the one on which the Beacons are located. In typical warehouses, materials are stored on shelves and, thus, their positions need to be tracked in three dimensions. This would require a much more sophisiticated observation model, but it would make the system much more versatile and more applicable to real-world scenarios. Combined with the distributed computation discussed in Section 6.1 and the knowledge of the physical environment discussed in Section 6.2, these refinements would create a much more robust system.

Chapter 7

Conclusion

In order to monitor the relative locations of hazardous materials in a warehouse, I proposed the use of wireless sensor motes that employ ultrasound pulses and radio signals to determine pairwise distance estimates. By affixing sensor motes to the pallets, as well as to the warehouse ceiling, the generated distance estimates can be used to localize the pallets. However, these generated distance estimates are imperfect due to the physical limitations of the hardware and environmental factors. As a result, using these measurements in order to localize the pallets can produce inaccurate results.

In order to mitigate this problem, I outlined a probabilistic algorithm using SIR particle filtering. This algorithm used a sampling-based method to estimate the location of potentially mobile motes in real time, based on the received distance estimates. Furthermore, the algorithm was augmented with probabilistic models of the sensors, taking into account both the relative distance and angle between the transmitting and receiving mote when computing distance estimates. I constructed several such models, either by using inverse distance weighting in order to interpolate expected distance measurements on the fly or modeling those distance estimates as a function of the relative sensor locations. Furthermore, I constructed models both with and without a notion of the relative angle between the sensor motes, in order to validate the claim that taking angle into account can improve the accuracy with which the models predicted sensor behavior.

In order to evaluate this approach, I built a simulator that is able to be run based on real-world sensor data and collected distance estimates. After examining simulator results to determine a reasonable physical configuration, I collected distance estimates generated by Cricket motes and evaluated the performance of the various observation models based on this real-world data. The results showed that considering the relative angle of the sensors can improve localization accuracy by 6.7%, and that the observation models based around formulating distance estimates as a function outperformed the models using inverse distance weighting by as much as 71.5%.

The simulations also showed that, given the current implementation in unoptimized Java, the system had no trouble tracking a mote in real time, estimating the location of up to eight mobile motes within 8.5 cm while processing received distance estimates once every second. This suggests that, with an implementation written optimized code that is compiled directly to native machine language instead of Java bytecode, much faster computation could be realized.

References

[1] D. Moore, J. Leonard, D. Rus, and S. Teller, "Robust distributed network localization with noisy range measurements," in *2nd International Conference on Embedded Networked Sensor Systems*, Baltimore, Maryland, USA, Nov. 2004, pp. 50–61.

[2] N. B. Priyantha, A. K. Miu, H. Balakrishnan, and S. Teller, "The cricket compass for context-aware mobile applications," in *7th Annual International Conference on Mobile Computing and Networking*, Rome, Italy, Jul. 2001, pp. 1–14.

[3] J. L. Hill and D. E. Culler, "Mica: a wireless platform for deeply embedded networks," *IEEE Micro*, vol. 22, no. 6, pp. 12–24, 2002.

[4] J. Polastre, R. Szewczyk, , and D. Culler, "Telos: enabling ultra-low power wireless research," in *4th International Symposium on Information Processing in Sensor Networks*, Los Angeles, California, USA, Apr. 2005, p. 48.

[5] P. Cudre-Mauroux, M. Fogel, K. Goldberg, and M. Franklin, "Sentry pallets for automated monitoring of spatial constraints: Models and initial experiments," in *IEEE International Conference on Robotics and Automation (ICRA'06)*, Orlando, Florida, USA, May 2006, pp. 4396–4398.

[6] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, Feb. 2002.

[7] J. P. V. den Berg, "A literature survey on planning and control of warehousing systems," *IIE Transactions*, vol. 31, no. 8, pp. 751–762, 1999.

[8] K. Michael and L. McCathie, "The pros and cons of RFID in supply chain management," in *International Conference on Mobile Business (ICMB '05)*, Sydney, Australia, Jul. 2005, pp. 623–629.

[9] D. Hahnel, W. Burgard, D. Fox, K. Fishkin, and M. Philipose, "Mapping and localization with RFID technology," in *IEEE International Conference on Robotics and Automation (ICRA '04)*, Boston, Massachusetts, USA, Apr. 2004, pp. 190–202.

[10] U. Bischoff, M. Strohbach, M. Hazas, and G. Kortuem, "Constraint-based distance estimation in ad-hoc wireless sensor networks," in *3rd European Workshop on Wireless Sensor Networks (EWSN '06)*, Zurich, Switzerland, Feb. 2006, pp. 54–68.

[11] K. Langendoen and N. Reijers, "Distributed localization in wireless sensor networks: a quantitative comparison," *Computer Networks*, vol. 43, no. 4, pp. 499–518, 2003.

[12] L. Hu and D. Evans, "Localization for mobile sensor networks," in *10th Annual International Conference on Mobile Computing and Networking*, Philadelphia, Pennsylvania, USA, Sep. 2004, pp. 45–57.

[13] A. Nasipuri and K. Li, “A directionality based location discovery scheme for wireless sensor networks,” in *1st ACM International Workshop on Wireless Sensor Networks and Applications*, Atlanta, Georgia, USA, Sep. 2002, pp. 105–111.

[14] N. Patwari and A. O. H. III, “Using proximity and quantized RSS for sensor localization in wireless networks,” in *2nd ACM International Conference on Wireless Sensor Networks and Applications*, San Diego, California, USA, Sep. 2003, pp. 20–29.

[15] M. Maroti, B. Kusy, G. Balogh, P. Volgyesi, K. Molnar, A. Nadas, S. Dora, and A. Ledeczi, “Radio interferometric positioning,” Institute for Software Integrated Systems, Vanderbilt University, TN, Tech. Rep. ISIS-05-602, Nov. 2005.

[16] N. B. Priyantha, “The cricket indoor location system,” Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, Jun. 2005.

[17] K. Michael and L. McCathie, “The arc-transversal median algorithm: a geometric approach to increasing ultrasonic sensor azimuth accuracy,” *IEEE Transactions on Robotics and Automation*, vol. 19, no. 3, pp. 513–521, 2003.

[18] D. P. Massa, “Choosing an ultrasonic sensor for proximity or distance measurement; part 2: Optimizing sensor selection,” *Sensors*, vol. 16, no. 3, pp. 28–43, 1999.

[19] B. Barshan and R. Kuc, “A bat-like sonar system for obstacle localization,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 4, pp. 636–646, Jul. 1992.

[20] S. Thrun, “Probabilistic robotics,” *Communications of the ACM*, vol. 45, no. 3, pp. 52–57, 2002.

[21] J. Schiff and K. Goldberg, “Automated intruder tracking using particle filtering and a network of binary motion sensors,” in *IEEE International Conference on Automation Science and Engineering (CASE’06)*, Shanghai, China, Oct. 2006, pp. 1–2.

[22] R. Stoleru, T. He, J. A. Stankovic, and D. Luebke, “A high-accuracy, low-cost localization system for wireless sensor networks,” in *3rd Iternational Conference on Embedded Networked Sensor Systems*, San Diego, California, USA, Nov. 2005, pp. 13–26.

[23] C. Taylor, A. Rahimi, J. Bachrach, H. Shrobe, and A. Grue, “Simultaneous localization, calibration, and tracking in an ad hoc sensor network,” in *5th International Conference on Information Processing in Sensor Networks*, Nashville, Tennessee, USA, Apr. 2006, pp. 27–33.

[24] H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping: Part i,” *IEEE Robotics and Automation Magazine*, vol. 13, pp. 99–110, Jun. 2006.

[25] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics.* Cambridge, Massachusetts, USA: MIT Press, 2005.

[26] M. Bolic, P. M. Djuric, and S. Hong, “Resampling algorithms for particle filters: A computational complexity perspective,” *EURASIP Journal on Applied Signal Processing*, vol. 2004, no. 15, pp. 2267–2277, 2004.

[27] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," *IEEE Proceedings F: Radar and Signal Processing*, vol. 140, no. 2, pp. 107–113, Apr. 1993.

[28] D. Shepard, "A two-dimensional interpolation function for irregularly-spaced data," in *Proceedings of the 1968 23rd ACM National Conference*, Aug. 1968, pp. 517–524.

[29] D. E. Knuth, "Big omicron and big omega and big theta," *ACM SIGACT News*, vol. 8, no. 2, pp. 18–24, Jun. 1976.