Design and Implementation of Reliable Localization Algorithms using Received Signal Strength

Jeffrey Vander Stoep

A thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science in Electrical Engineering

University of Washington

2009

Program Authorized to Offer Degree: Electrical Engineering

University of Washington
Graduate School

This is to certify that I have examined this copy of a master's thesis by

Jeffrey Vander Stoep

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

Committee Members:

_____
Radha Poovendran

_____
James Ritcey

Date: _____

In presenting this thesis in partial fulfillment of the requirements for a master's degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this thesis is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Any other reproduction for any purpose or by any means shall not be allowed without my written permission.

Signature_____

Date_____

University of Washington

Abstract

Design and Implementation of Reliable Localization Algorithms using Received Signal Strength

Jeffrey Vander Stoep

Chair of the Supervisory Committee:
Associate Professor Radha Poovendran
Electrical Engineering

Received Signal Strength (RSS) is a readily available and cost-effective method of location estimation, or *localization*, in wireless sensor networks (WSNs). However, RSS-derived distance estimates are known to be inaccurate, leading many researchers to conclude that RSS is an unreliable method for localization. In this work, we address the problem of developing reliable localization algorithms using unreliable RSS measurements. We evaluate several range-dependent RSS-based algorithms for accuracy and computational intensity. We then suggest improvements for existing algorithms using statistical information from RSS-derived distance estimates and demonstrate the improvements. Finally, we present a system based on the Intel Imote2 and the Linux OS that features three different localization algorithms along with variations to those algorithms to be run on a sensor network.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

This work is supported in part by the following grants: ONR YIP, N00014-04-1-0479; ARO PECASE, W911NF-05-1-0491; ARL CTA, DAAD19-01-2-001; and ARO MURI, W911NF-07-1-0287.

Chapter 1
INTRODUCTION

Technological advances in hardware and software for wireless communication have led to the ubiquity of wireless networks for personal, commercial, industrial, and military use. In many applications, device location plays a critical role in network operations and services. In a wireless sensor network (WSN), for example, alarm signals sent to a central monitoring system must include the location of the sensed phenomenon. Furthermore, the efficiency of network operations, such as routing, can be improved when node locations are available. The ability for devices to estimate their own location, referred to as *localization*, is thus a crucial service in wireless networks [10, 19].

One approach to wireless localization is for a central authority to learn a map of the network area through calibration measurements taken by network nodes. The central authority then estimates the location of each node, for example by using a maximum likelihood estimate [7, 22], and sends the computed location to the node. Such an approach is not applicable in WSN localization for two reasons. First, central processing of location information would require prohibitive communication overhead in the WSN. Second, changes to the WSN, either due to node addition or deletion or environmental changes, would require recalibration and retransmission of node locations.

An alternate approach to localization in WSNs involves the use of beacons. In beacon-based localization, nodes calculate their position using a distributed algorithm based on information obtained from nearby beacons, either neighboring sensor nodes or a set of specialized *anchor* nodes. The information available to each localizing node

consists of the location of each neighboring beacon and a physical measurement relating the node to the beacon. Techniques for beacon-based localization can be separated into two classes based on the measurements relating the node to the beacon: range-free and range-dependent methods. In a range-free localization algorithm [3, 8, 12, 16, 18], each node uses only the beacons' locations and the fact that the beacons are nearby to estimate its location. Range-free algorithms do not require an explicit measurement of the distance between the localizing node and the beacon. Contrastingly, in a range-dependent localization algorithm, each node estimates the distance to the nearby beacons and uses the distances and beacon locations to estimate its location. Range-dependent techniques rely on either time of arrival (TOA) [4], time difference of arrival (TDOA) [21], or received signal strength (RSS) to estimate the required distances. Of these three methods, all but RSS require specialized hardware that is costly in terms of resource expenditure and the monetary cost of hardware. TOA requires high-precision synchronization that cannot be obtained in a sensor with a 1 MHz clock. TDOA involves the use of a second medium and requires specialized hardware, such as the use of speakers, microphones, and amplifiers for TDOA using an acoustic channel.

Combined with the fact that RSS does not require specialized hardware and is available on most commercial off-the-shelf (COTS) hardware, including cellular telephones, WiFi-capable devices, and sensor nodes, the low-cost use of RSS as a distance estimation technique has lead to a number of RSS-based localization algorithms [4, 11, 13, 14, 17, 23]. In addition, the inclusion of RSS on radio chips is required by most wireless standards including IEEE 802.11, 802.15.4, and 802.16, which use RSS measurements in conjunction with other link parameters such as Link Quality Index (LQI) and Received Channel Power Indicator (RCPI). While RSS is included in most commercial hardware platforms, researchers have expressed doubts about the reliability of RSS measurements [3, 5, 8, 10], especially from beacons that are a considerable distance from the localizing node. An often suggested approach for achieving

reliable information from RSS measurements is to discard information from distant beacons that give unreliable measurements [9,15]. However, such an ability to discard measurements requires a relatively high density of beacons in the WSN.

Hence, in this work, we approach the problem of *developing localization algorithms which allow a node to compute a location estimate from unreliable RSS measurements.* We make the following contributions.

- We show that the RSS measurements from distant beacons contain useful information that can improve the quality of localization algorithms despite the high degree of uncertainty.

- We present a stochastic framework for RSS-based localization that incorporates the uncertainty in RSS measurements by replacing distance estimates with probability distributions. We show that stochastic framework for RSS-based localization can be incorporated into existing location estimation algorithms to improve the localization accuracy.

- We demonstrate the improvements in localization accuracy in simulation and implementation.

- We provide the software, documentation, and user's manuals for setup and implementation of a testbed of Imote wireless sensors.

The remainder of this paper is organized as follows. In Chapter 2, we review relevant background on RSS-based localization algorithms. In Chapter 3, we present our framework for stochastic characterization of distance measurements using RSS measurements. In Chapter 4, we present the modifications to existing algorithms. In Chapter 5, we demonstrate the improvement in localization accuracy using the proposed algorithms. In Chapter 6 we present a high-level view of the sensor network implementation. In Chapter 7 we supply step-by-step instructions on installation and

setup of the sensor network. We summarize our results and discuss future work in Chapter 8.

Chapter 2
BACKGROUND

In this chapter, we provide background on RSS-based range-dependent localization algorithms. Due to the large body of literature in localization, we review only the previous works which are discussed explicitly in this work.

Kwon et al. [10] proposed the use of a minimum mean-square error (MMSE) estimation for range-dependent localization in WSNs. In a distributed implementation, each sensor node estimates the distances to neighboring nodes and iteratively adjusts its current location using a *gradient descent* algorithm which converges to the MMSE estimate. The distributed implementation yields location estimates in a virtual coordinate system, which can be mapped into a global coordinate system using the reference coordinates of fixed-location beacon nodes.

Liu, Ning, and Du [14] proposed the use of a *majority voting* protocol in which the network area is partitioned into an evenly-spaced grid. Each node uses RSS-based techniques to estimate the distance to each neighboring beacon and constructs a virtual annulus, or ring, around the beacon's coordinates, where the inner and outer radius of the annulus represent a fixed error bound on the estimated distance. Each grid point receives a score equal to the number of annuli that contain it. The sensor then estimates its location as a point inside the region with maximum score, such as the center-of-mass of the region.

Yedavalli et al. [23] proposed a constraint-based approach which, similar to the grid-scoring protocol above, increments the score of each grid point which satisfies a distance constraint and decrements the score of each grid point which does not satisfy a distance constraint. The distance constraints are established using the relationship

that received signal strength decreases with distance, though explicit distance estimates are not computed. The location of the sensor is estimated in a method similar to the protocol in [14].

Savvides et al. [21] compared the performance of multilateration techniques using time difference of arrival and received signal strength measurements. The proposed techniques estimate the node locations using a maximum likelihood estimator with estimated distances obtained from either TDOA or RSS.

Chapter 3

STOCHASTIC FRAMEWORK FOR RSS-BASED LOCALIZATION

In this chapter, we propose a framework for RSS-based localization which incorporates the uncertainty in RSS-based distance estimation into the localization algorithm. We show that estimating distances from RSS measurements and then using these distance estimates for location estimation discards information which is useful to the localization algorithm. Before presenting our framework, we outline the process of mapping RSS measurements to distance estimates.

3.1 RSS-Based Distance Estimation

Various techniques have been proposed in the literature for mapping RSS measurements to distance estimates. The basic model used in localization literature [11, 13, 21, 23] is given by the formula

$$RSS = P_t - PL(d_0) - 10\alpha \log_{10} \frac{d}{d_0} + X_{\sigma_{RSS}}, \tag{3.1}$$

where RSS is the received signal strength in units of decibels with respect to milliwatts (dBm). In (3.1), d is the true distance from the sender to the receiver, α is the path-loss exponent, P_t is the transmit power of the sender in dBm, $PL(d_0)$ is the power loss in dBm at a reference distance d_0. The quantity $X_{\sigma_{RSS}}$ in dBm is a random variable representing the noise in the measured RSS and is often assumed to be a zero-mean Gaussian random variable with variance σ_{RSS}. The source of the noise $X_{\sigma_{RSS}}$ in measured RSS can come from both time varying and time-invariant sources. Time varying errors, such as interference, can be averaged out by taking multiple measurements corresponding to the same distance. Time-invariant errors,

such as shadowing due to heterogeneity in the medium resulting from objects such as walls or buildings, can cause the signal to degrade contrary to the path-loss model. These errors cannot be averaged out by taking multiple measurements, as the path-loss model cannot be specifically designed for each wireless channel in each deployed network. Several researchers have observed that the random effects of shadowing are appropriately modeled by assuming the error $X_{\sigma_{RSS}}$ is Gaussian.

In a WSN performing RSS-based localization according to existing techniques, each node measures an RSS value corresponding to each neighboring beacon node. This measured value RSS is mapped to a distance estimate $\hat{d}$. This mapping from RSS to $\hat{d}$ requires an expression for the distance $\hat{d}$ as a function of RSS and can be obtained by solving (3.1) for d, yielding

$$d = k10^{(P_t - RSS + X_{\sigma_{RSS}})/(10\alpha)} \tag{3.2}$$

where k is a constant incorporating both $PL(d_0)$ and $\alpha \log_{10}(d_0)$. To obtain a practical fit for the model parameters, we have taken RSS measurements at a set of reference distances d_{r_i}. For each distance d_{r_i}, the sample mean $\mu_{RSS}(d_{r_i})$ and sample standard deviation $\sigma_{RSS}(d_{r_i})$ are computed and plotted in Figure 3.1. The model parameters are computed from the set of sample means using a best-fit curve using (3.2) and averaged as $\mu_{RSS}(d_{r_i})$. A curve is fit to this sample mean using (3.2). An example of this model-fitting process is illustrated in Figure 3.1.

From Figure 3.1, we note that the resulting sample standard deviation $\sigma_{RSS}(d_{r_i})$ is approximately constant as a function of the reference distance d_{r_i}. To clarify this point, Figure 3.2 illustrates the sample standard deviation as a function of d_{r_i} and the corresponding linear approximation. The variance in the Gaussian random variable $X_{\sigma_{RSS}}$ is thus independent of the distance d between the transmitter and receiver [19].

Given the empirical values of k and α, the resulting model in (3.2) can be used to compute a distance estimate $\hat{d}$ from a measured RSS as

$$\hat{d} = k10^{(P_t - RSS)/(10\alpha)}. \tag{3.3}$$

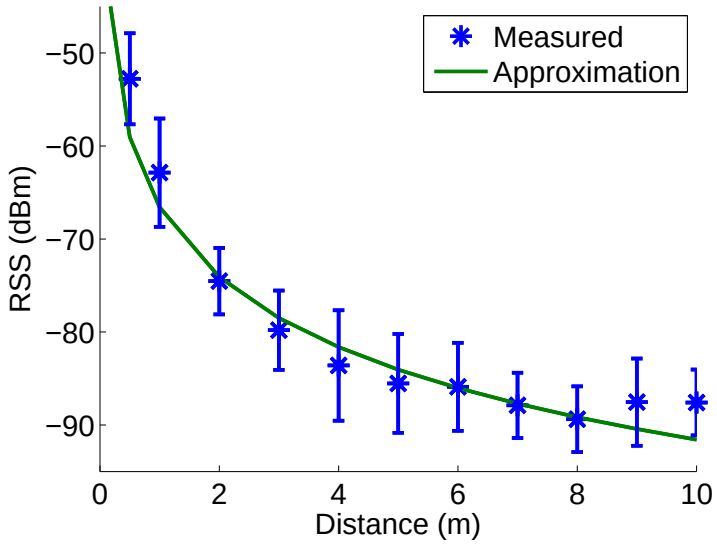

Figure 3.1: The sample mean and standard deviation are plotted for a set of reference distances d_{r_i}. The model parameters k and α are computed by fitting a curve to the reference model in (3.2) using the samples.

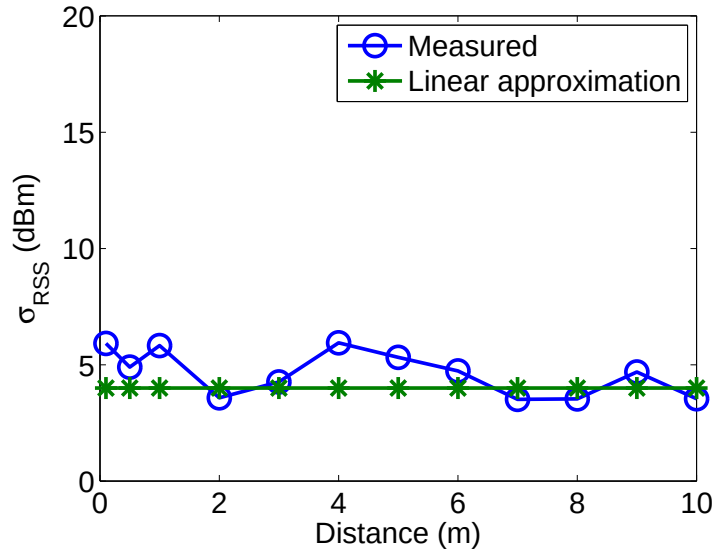

Figure 3.2: The sample standard deviation $\sigma_{RSS}(d_{r_i})$ over the set of reference distances is well-approximated by a constant function in d_{r_i}.

As shown in [19], this estimate is the maximum likelihood estimate (MLE) for the distance d given the measured RSS. Due to the non-linearity of the mapping from the measured RSS to the distance estimate $\hat{d}$ in (3.3), the constant standard deviation σ_{RSS} in the RSS measurement maps to a standard deviation σ_d in the distance estimate that increases linearly with distance. This result can be derived analytically using a delta approximation based on the Taylor series expansion [20] for dependent random variables X and Y such that $Y = f(X)$. The delta approximation states that the standard deviation σ_Y is approximated by

$$\sigma_Y \approx \sigma_X \cdot f'(\mu_X), \tag{3.4}$$

where μ_X and σ_X are respectively the mean and standard deviation of X and $f'(x)$ is the derivative of $f(x)$. Applying this result to the function $\hat{d}(RSS)$ mapping a measured RSS to a distance estimate $\hat{d}$ yields

$$\begin{aligned}\sigma_d &\approx \sigma_{RSS}\hat{d}'(\mu_{RSS}) \\ &= \sigma_{RSS} k \log(10^{1/(10\alpha)}) 10^{(P_t-\mu_{RSS})/(10\alpha)} \\ &= \sigma_{RSS} \log(10^{1/(10\alpha)}) d. \end{aligned} \tag{3.5}$$

Hence, the variance in the distance estimate $\hat{d}$ is directly proportional to the true distance d at which the measurement was taken. Researchers have suggested discarding measurements from beacons beyond a threshold distance due to the high degree of uncertainty in such measurements [3, 5, 8, 10]. However, the ability to discard measurements while obtaining a sufficient number of measurements from nearby beacons requires a relatively high beacon density.

In particular, when beacons with radio range r are randomly deployed with density ρ, the probability that a node is within range of at least k beacons, denoted by $p(k, r, \rho)$, is given by

$$p(k, r, \rho) = 1 - e^{-\rho\pi r^2} \sum_{i=0}^{k-1} \frac{(\rho\pi r^2)^i}{i!}, \tag{3.6}$$

which follows by use of spatial statistics for a Poisson point process of intensity ρ [6]. If RSS measurements beyond a threshold distance $r_T < r$ are to be discarded then the beacon density must increase to

$$\rho_T = \rho \left(\frac{r}{r_T} \right)^2 \tag{3.7}$$

in order to maintain the coverage probability $p(k, r_T, \rho_T)$ equal to $p(k, r, \rho)$ with the modified parameters. Furthermore, we note that when beacons are randomly deployed, the probability that each beacon measurement is within the threshold distance r_T decays rapidly as the threshold becomes more strict, as the coverage area of a beacon scales quadratically in r_T. This problem is depicted in Figure 3.3 for random deployment of beacons. Hence, we are interested in taking advantage of the estimation error in the distance estimates $\hat{d}_i$ from distant neighboring beacons instead of discarding this information.

3.2 Stochastic Model for Localization

In existing RSS-based localization algorithms, the typical operation is to estimate a distance $\hat{d}_i$ from the measured RSS_i corresponding to each neighboring beacon i. These distance estimates $\hat{d}_i$ are then passed to the localization algorithm along with the beacon coordinates (x_i, y_i) in the plane of the network. This two-step protocol operation is illustrated in the schematic in Figure 3.4. Since the standard deviation in each distance estimate $\hat{d}_i$ increases proportional to the true distance d_i to the beacon i, as given in (3.5), the accuracy of the resulting location estimate depends indirectly on the standard deviation σ_{d_i}. However, most RSS-based localization algorithms operate only with the estimates $\hat{d}_i$ and corresponding coordinates, without considering the inherent uncertainty in the estimates.

Given that the mapping from a measurement RSS to a distance estimate $\hat{d}$ has standard deviation proportional to the distance d between the transmitter and receiver, the ability to take advantage of the distance estimates from distant beacons

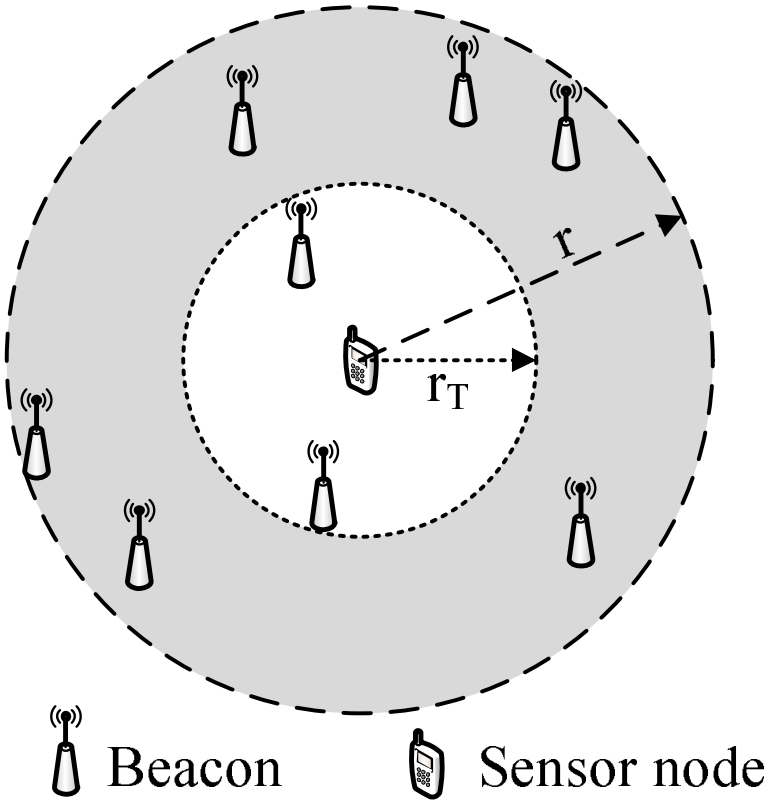

Figure 3.3: If measurements from beacons in the shaded region beyond a threshold distance $r_T < r$ are to be discarded, the beacon density ρ must increase by a factor $(r/r_T)^2$ to maintain the same average beacon coverage at the localizing node.

requires more information about the measurement RSS to be passed to the localization algorithm. We thus propose the following framework for RSS-based distance estimation.

As illustrated in Figure 3.5, the Gaussian distribution of noise in RSS measurements with constant variance σ^2_{RSS} can be mapped to a skewed distribution on the beacon distance d_i. Instead of constructing an estimate $\hat{d}_i$ for each distance d_i, corresponding to keeping only a single point in the RSS noise distribution, we construct the entire distribution $f_i(d_i|RSS_i)$ which characterizes the relationship between the RSS measurement RSS_i and the true beacon distance d_i. This distribution can be

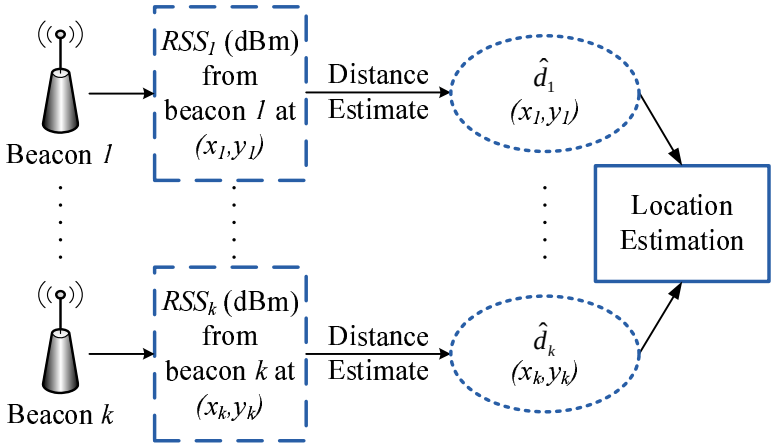

Figure 3.4: Existing RSS-based localization algorithms consist of two independent estimation problems. The first problem is to estimate the distance $\hat{d}_i$ to each beacon i from a measurement RSS_i, and the second problem is to estimate the location of the node using the distance estimates $\hat{d}_i$ and the beacon coordinates (x_i, y_i).

passed to the localization algorithm which can leverage the uncertainty in the set of distributions f_i to estimate the node location. Our new model of protocol operation is illustrated in Figure 3.6. We note that this new model effectively jointly estimates the beacon distances d_i in the form of optimizing the node location directly from the information available in the distributions f_i.

We note that the new model illustrated in Figure 3.6 is a generalization of the standard model in Figure 3.4 in that the localization algorithm can first compute an estimate $\hat{d}_i$ from each distribution $f_i(d_i|RSS_i)$ before estimating the node location.

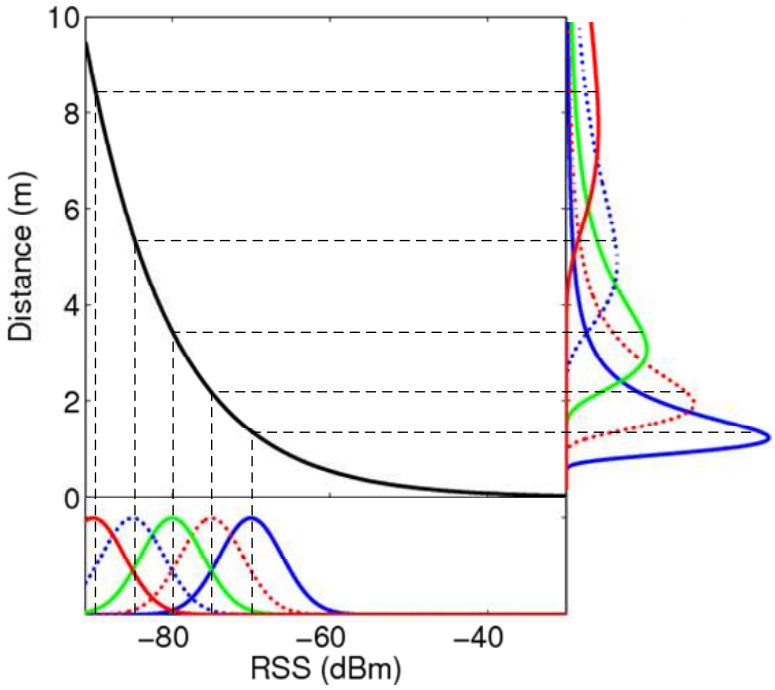

Figure 3.5: The mapping from RSS measurements to distance estimates in (3.2) can be used to map the Gaussian distribution of RSS measurement error to a distribution of the beacon distance d_i. Note that while measurements from distant beacons have high estimation variance, they still provide information to the localization algorithm.

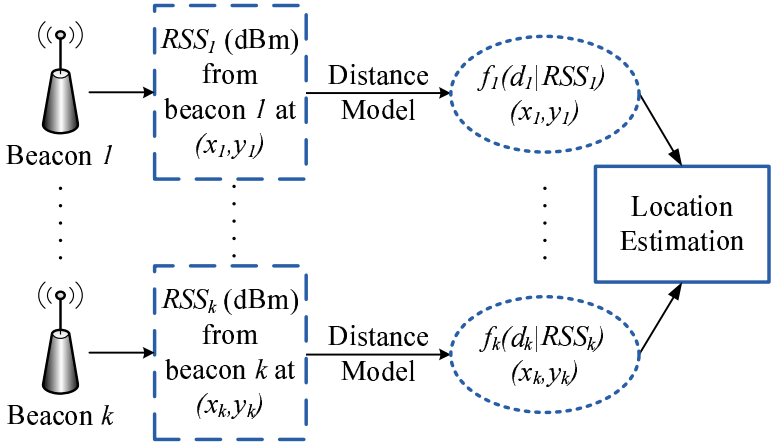

Figure 3.6: Our proposed model for RSS-based localization replaces the two-step estimation model in Figure 3.4 with a direct estimate of the node's location using a stochastic characterization of each distance d_i given the measurement RSS_i. This approach passes more information about the RSS measurement to the localization algorithm.

Chapter 4
STOCHASTIC LOCALIZATION ALGORITHMS

We next demonstrate how the stochastic characterization of distance measurements presented in Chapter 3 can be used with existing algorithms. We illustrate the application of our stochastic characterization by example using the gradient-descent algorithm of [10] discussed in Chapter 2. In this algorithm, the localizing node iteratively updates the location estimate (x, y) using estimated distances $\hat{d}_i$ to each beacon i by moving in the direction which reduces the error between $\hat{d}_i$ and the distance from the beacon coordinate (x_i, y_i) and the current estimate (x, y). The actual change to the estimate (x, y) is a summation of errors between distances over the set of neighboring beacons. In the algorithm in [10], the authors suggest that the linear combination of error vectors can be weighted by coefficients a_i, but they do not discuss the choice of coefficients.

In the stochastic framework, the quality of the information obtained from different beacon measurements depends on the distance d_i from the beacon, as shown in (3.5) in terms of the standard deviation σ_d of the distance estimate. We thus propose incorporating the standard deviation σ_{d_i} of each distance estimate in the coefficient a_i used in the gradient-descent algorithm by setting a_i inversely proportional to σ_d. In particular, since the coefficients a_i in the algorithm are relative, and scaling all coefficients by a constant has little effect on the accuracy of the algorithm, we propose to set each coefficient a_i to

$$a_i = \frac{1}{\hat{d}_i}, \tag{4.1}$$

where we take advantage of the linear relationship between the distance d_i and the standard deviation σ_d as in (3.5). Intuitively, this choice of coefficients allows the

algorithm to give priority to measurements of higher quality while still including information from distant beacons.

Chapter 5
SIMULATION

In this section, we present simulation results to compare the achievable improvements of the majority voting and gradient descent algorithms presented in Section 2. In our simulation study, we assume that a network of beacons and sensor nodes is randomly deployed over a region A. The beacon transmit power P_t is known to sensor nodes, and the path-loss model in (3.2) is assumed. A message is considered to be successfully received if the received power is higher than a minimum threshold that corresponds to an actual range of 10 m in a noise-free environment. When a sensor node receives a message from beacon i containing coordinates (x_i, y_i), an RSS measurement RSS_i is taken in order to generate the distribution $f_i(d_i|RSS_i)$.

We compute the results of several algorithms over a range of beacon densities ρ, scaled to the average number of beacons within range of each sensor node. We define the localization error of a single node as the distance from the estimated location to the true location of the node. Our simulation results represent an average over 1000 trials, with each trial corresponding to the deployment of 10 localizing nodes deployed in the region A. We show that the ReLoc algorithm offers improvement over the entire range of beacon densities when compared to existing algorithms.

We begin by showing the results of the majority voting algorithm introduced in [14]. Each sensor node discretizes the region A into an evenly-spaced grid. An annulus with inner radius $\hat{d}_i - \Delta$ and outer radius $\hat{d}_i + \Delta$ is constructed around each beacon i at estimated distance $\hat{d}_i$, and the score of each grid point (x, y) is computed as the number of annuli containing the point. The center of mass of all grid points with the highest score becomes the location estimate (x_0, y_0) for the node. We have chosen

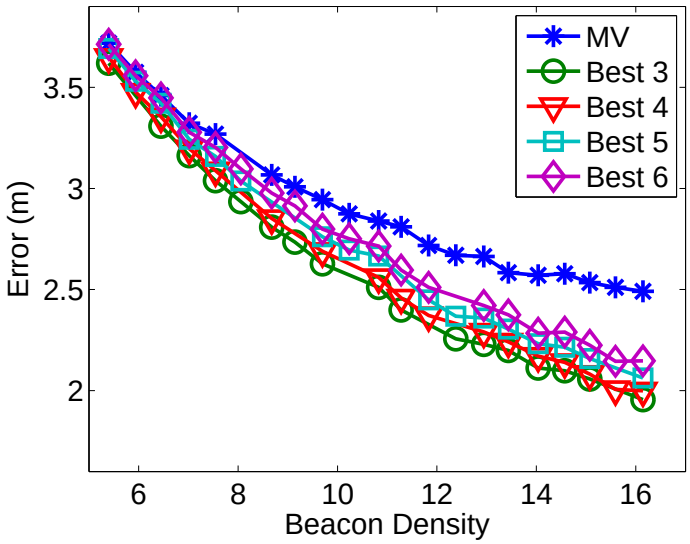

Figure 5.1: The majority voting (MV) approach introduced in [14] is simulated, incorporating the RSS measurements of all or only the best N measurements. The annulus width 2Δ is fixed with the error of $\Delta = 2.5$ m around the distance estimate $\hat{d}_i$.

a grid spacing of 0.1 m. We simulated several values of Δ and chose the best value of 2.5 m for comparison with our improvement of the algorithm. In addition, we simulate the results of this algorithm when each localizing node uses the measurements from only the nearest N beacons, corresponding to the suggested approach of discarding measurements with high standard deviation σ_d. Figure 5.1 illustrates the results for when the localizing node discards all but $N = 3$, 4, 5, and 6 beacon measurements with the highest RSS values, referred to as the *best* N approach.

As illustrated in Figure 5.1, the average localization error decreases as the beacon density increases for all simulated cases. At low beacon density, the result obtained by using only the best N beacons parallels the basic majority voting method. At higher beacon densities, however, the restriction to the RSS measurements from the nearest N beacons shows improvement. Restriction to the best $N = 3$ beacons gives the most improvement over the range of beacon densities because the probability that

$N = 3$ beacons are deployed a small distance away from the localizing node is higher than for larger values of N, leading to a lower error standard deviation in the location estimate.

For our improvment of the majority voting algorithm, we incorporate second order statistics into the algorithm. No longer is the annulus simply $\hat{d}\pm\Delta$, where Δ represents a constant number. Now Δ is related to the measured distance $\hat{d}$. Additionally, the vote provided by each node is now $\frac{1}{\hat{d}}$. Thus an anchor that is close will give a high vote and have a thin annulus. An alternate view of this method is to view a uniform approximation to the log-normal result displayed in Figure 3.5. Simulation results for this improvement are displayed in Figure 5.2, the average localization error using our improvement of the algorithm is illustrated over a similar range of beacon densities, demonstrating an improvement of 25% at low beacon density to 50% at high beacon density. We note that the inclusion of RSS measurements from all beacons using the imrovement gives greater accuracy than using only the best N measurements, even improving over the optimal $N = 3$ case as in Figure 5.1. This demonstrates that the distance distributions with high standard deviation σ_d contribute valuable information to the localization algorithm.

In addition to the majority voting algorithm, we present the result of applying the stochastic framework of Section 3 to the gradient descent algorithm in [10], as discussed in Section 4. In this simulation, we set each weight a_i as in (4.1), implying that the contribution of each beacon is inversely proportional to the standard deviation of the distance estimate. Figure 5.3 shows the results of the simulation and demonstrates improvements ranging from 5% to 33%.

We note that when the beacon density is low, there is a smaller probability that the distance estimates will differ significantly. Hence, the weights a_i incorporated into the algorithm will not differ significantly, and the improvement using our approach will be small. As the beacon density increases, however, the probability of differing weights increases, and weighting will show significant improvement over the standard

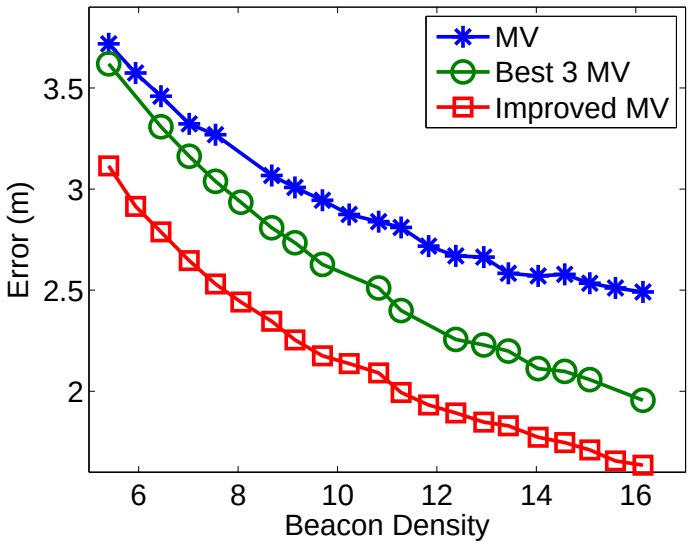

Figure 5.2: The improved majority voting algorithm is simulated along with basic majority voting and best 3 majority voting algorithms. The improvement obtained compared to the best 3 demonstrates that valuable information is contained in the measurements from distant beacons.

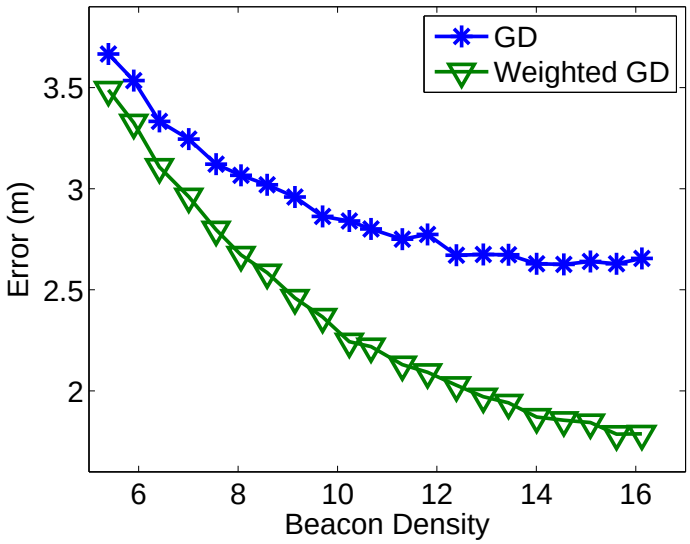

Figure 5.3: The gradient descent algorithm of [10] and the improvement suggested in Section 4 using the stochastic framework are simulated, illustrating the improvement in localization accuracy over a range of beacon densities.

algorithm.

Chapter 6
SENSOR NETWORK IMPLEMENTATION

The sensor network implementation has been achieved via the Crossbow Imote2 platform [1]. The Imote2 is designed around the low-power PXA271 XScale microcontroller with 802.15.4 radio and surface mount 2.4 GHz antenna. We have installed the Linux Kernel and the drivers necessary for radio communication. In this section we discuss the details of implementing localization onto a network of Imote2 wireless sensors.

6.1 Communicating with the network

In order to study the localizing sensor network, communication from the network to a destination outside of the network is required both to issue commands to the network and to receive location estimates from localizing sensor nodes. This section focuses on connecting to the sensor network, issuing commands, and receiving location information from the network.

A special node referred to as the *base* node will be the communication link between the sensor network and a personal computer. The base node will be connected to a PC using USB and communicating through USB via a socket. The base node's duties include connecting to the GUI, keeping track of routing information, and forwarding network commands from the GUI to the sensor network.

6.2 Routing

In large network deployments, nodes will likely be out of radio range of the base node. Thus message routing through other nodes is essential for control and data aquisition.

This particular network requires that each node be able to route a message to the base, and the base be able to route commands to each node. We have tried to accomplish two things with our routing protocols. First the protocol needs to be light weight and simple. Second the routing messages should never trigger other messages as this could cause a cascade of nodes trying to transmit simultaneously.

6.2.1 Routing information to the base node

Routing of information to the base node is accomplished through a distributed Bellman-Ford algorithm [2]. At set, unsynchronized, intervals each node broadcasts their fewest number of hops to the base. In this setup, we are assuming that all nodes have the same radio range. Thus if node a recieves a message from node b, then node b is within node a's range.

Table 6.1 illustrates the Bellman-Ford routing table. The *Last Heard* column is incremented at each timer interval. If the last heard column goes above a certain threshold, then the address is considered dead and removed from the routing table. Each time that a particular address is heard broadcasting its routing message, its last heard count is reset to zero.

Table 6.1: Bellman-Ford routing table.

Address	Hops to base	Next hop	Last Heard
1001	2	1003	2
1002	3	1001	4
1005	4	1002	1
.	.	.	.
.	.	.	.
.	.	.	.

6.2.2 Routing commands out from the base node

Routing messages from the base node out to members of the network makes use of the Bellman-Ford routing protocol described in Section 6.2.1. Once a node has a valid path to the base node, it sends a packet to the base node via that path. As the message traverses the path, each intermitent node appends its address to the packet. The resulting message can be seen in Table 6.2. Thus when the message reaches the base node, it contains the list of hops taken. This process is repeated periodically to ensure that the base node has an up to date path to each member of the network. Messages from the base node out to members of the network follow a similar patern, but with the hop sequence reversed. As the message makes its way through the network, each node removes its address from the end of the message. This message structure can be seen in Table 6.3.

Table 6.2: Message sent to base node to establish route

Source node addr	First hop addr	Second hop addr	...

Table 6.3: Routing commands from the base node to members of the network

Message data	Source node addr	Last hop addr	...	First Hop

Chapter 7

LINUX INSTALLATION AND SETUP OF THE IMOTE2 NETWORK.

Configuring the Imote2 network for localization is a complex process and requires five major steps. Loading the Linux kernel, configuring the radio, setting up the cross compiler, loading the localization program, and defining node IDs and the base node. The installation requires

1. Windows PC with a parallel port
2. Linux PC
3. Intel JTAG cable
4. Parallel port cable
5. IIB2400 debugger board
6. Imote2.

To facilitate a more manageable installation and setup process, the following directions have been divided into numbered steps. Figure 7.1 exhibits the Imote2 programming setup.

7.1 Installing the Linux Kernel

This section requires a Windows PC with parallel port.

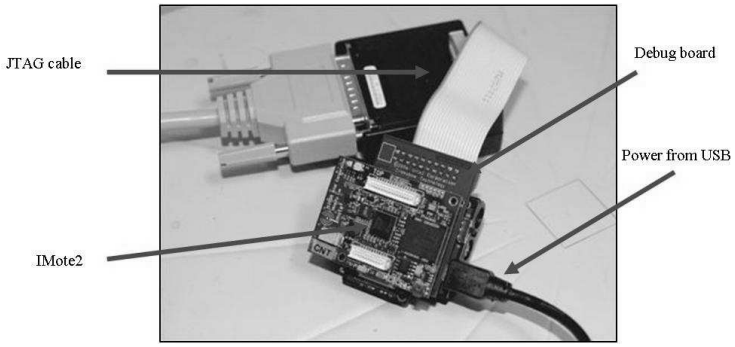

Figure 7.1: Imote2, JTAG cable, and IIB2400 debugger board connected to the USB cable

1. Download and install the XFlash C++ compiler from the Intel web page. A 30 day evaluation is available at http://software.intel.com/en-us/intel-sdp-home/.

2. In BIOS, change the parallel port settings to enable ECP and set the range to 378.

3. In Windows' device manager, open the LPT1 port and check *Enable legacy Plug and Play detection.*

4. Before plugging the Imote2 into the Windows PC, go to http://www.ftdichip.com/Drivers/D2XX.htm and download the appropriate driver. Unzipping this file should result in the folder *CDM 2.04.16 WHQL Certified.* Create a folder *C:\Program Files\FTDI Drivers* and copy the unzipped folder into this directory.

5. Connect the IIB2400 debugger board to the Windows PC via the parallel port cable and USB cable. Wait for the Windows PC to install the necessary software and then restart when indicated.

6. Download the bootloader *blob*, the Linux Kernel *zImage*, and system file *fs.jffs2*

from
http://ubi.cs.washington.edu/files/imote2/images/
backup_7_29_2008/Console_winGadget/images/.
Place them in the xflash directory C:\Program Files\Intel\SDT2.0\xflash\

7. Connect the IIB2400 to the parallel port of the Windows PC through the JTAG cable and directly to the Windows machine through the USB cable.

8. Power on the Imote2 by pressing the reset button.

9. In the file C:\Program Files\Intel\SDT2.0\xflash\boards.ini append the following:

```
[INTELMOTE2]
standardflash=GP_J3_K3_K18_8
monaddr=0x01000000
scanchain=PXA27X
algoaddr=0x5c000000
baseaddr=0x00000000
```

10. Create C:\Program Files\Intel\SDT2.0\xflash\intelmote2.xdb.

11. Open the Window's command prompt. Navigate to C:\Program Files\Intel\SDT2.0\xflash\ and execute the following commands:

```
xflash.exe -p intelmote2 -tt "Intel(R) JTAG Cable" blob -offset 0x00000000
xflash.exe -p intelmote2 -tt "Intel(R) JTAG Cable" zImage -offset 0x00040000
xflash.exe -p intelmote2 -tt "Intel(R) JTAG Cable" fs.jffs2 -offset 0x00240000
```

You should see something similar to

```
   Memory Successfully Burned.
File successfully burned.
```

12. Log into the Imote2 via Windows' Hyperterminal. The proper comm port can be decided by testing the available options. Set the following properties for the connection:

```
Bits per second: 115200
Data bits: 8
Parity: None
Stop bits: 1
Flow control: None
```

13. Login ID is *root*, password *rootme*.

14. Type *ifconfig* to obtain the correct IP address.

15. An alternative method for connecting is to use SSH. Remove the Imote2 from the debugger board, and connect it directly to the USB cable. Windows will automatically detect the *Linux USB Ethernet/RNDIS Gadget*. Choose to install from a specific location. Include C:\Program Files\FTDI Drivers\CDM 2.04.06 WHQL Certified\ in the search. Complete the installation.

16. In Windows' Network connections, right click on the *Linux USB Ethernet/RNDIS Gadget* connection and select properties. In the *General* tab, select *Internet Protocol (TCP/IP)* and select *properties*. Select *Use the following IP address:* and specify the IP as *192.168.99.100*.

17. Using Cygwin or SSH Secure Shell Client or another unix emulator, login using the command *ssh root@192.168.99.101* (or the appropriate address). The Password is *rootme*.

7.2 Configuring the radio

This section covers the steps needed to configure the Imote2's radio.

1. Go to https://sites.google.com/site/imote2linux/setting-up-radio and download the attachment tos_mac.ko. Copy to the imote2 in directory /lib/modules/2.6.14_r1.1/kernel/drivers/tosmac/ overwriting the previous version.

2. On the Imote2, add *tos_mac* to the /etc/modules file. Reboot the Imote2.

3. Create a character device by executing *mknod /dev/tosmac c 240 0* on the Imote2.

7.3 Seting up the cross compiler

This section requires a Linux PC.

1. On a Linux machine, download the ARM GCC compiler from http://ubi.cs.washington.edu/files/imote2/toolchain/ imote2_toolchain_3.4.3_binutils.tgz.

2. Uncompress to the root folder of the Linux PC.

3. Add export PATH=$PATH:/usr/local/arm/3.4.3/bin/ to ~/.bashrc.

7.4 Compiling and loading the localization program onto the Imote2

The section covers the steps required to download the localization code, compile it, and load it onto the imote2. Do the following steps on the Linux PC with where the cross-compiler was set up in Section 7.3.

1. First the code must be downloaded. It is currently available at http://code.google.com/p/imote2-localization/source/browse/trunk.

2. In the terminal, navigate to the directory with the localization code. Type *make*. This will result in the building of two files, *handler* and *handler_base*. The following directions are split up into directions for the base node and directions for anchors/localizing nodes

3. Base Node

   (a) Copy *local_base.sh* and *handler_base* to */root/* on the imote. This can be accomplished as follows. Transfer the file to a Windows PC that is connected to the imote. In a Unix like environment (Cygwin, SSH Shell, etc.) navigate to the directory on the pc with *local_base.sh* and *handler_base* and enter the following command:

   ```
   scp local_base.sh root@192.168.99.101
   scp handler_base root@192.168.99.101/
   ```

   (b) Allow each of these files to be executable. SSH onto the imote and in the */root/* directory, type:

   ```
   chmod +x local_base.sh
   chmod +x handler_base
   ```

   (c) OPTIONAL but recommended to allow the base node run its software upon bootup. Add the following symbolic link to */etc/init.d/*.

   ```
   cd /etc/init.d
   ln -s /root/local_base.sh
   ```

   This will allow the base node and routing programs to restart if it should crash.

4. Anchor/Localizing node

(a) Copy *local.sh* and *handler* to */root/* on the imote. This can be accomplished as follows. Transfer the file to a Windows PC that is connected to the imote. In a Unix like environment (Cygwin, SSH Shell, etc.) navigate to the directory on the pc with *local.sh* and *handler* and enter the following command:

```
scp local.sh root@192.168.99.101
scp handler root@192.168.99.101/
```

(b) Allow each of these files to be executable. SSH onto the imote and in the */root/* directory, type:

```
chmod +x local.sh
chmod +x handler
```

(c) Add the following symbolic link to */etc/init.d/*.

```
cd /etc/init.d
ln -s /root/local.sh
```

This will allow the node's localization and routing programs to restart if it should crash.

5. Transfer *handler* to the Windows PC and in a Unix like environment (Cygwin, SSH Shell, etc.) navigate to the directory with *handler* and enter the following command.

```
scp handler root@192.168.99.101/
```

6. SSH into the imote and make sure that the file is executable by entering.

```
chmod +x handler
```

7.5 Setting up the Network and defining node IDs

In this section we cover giving the node its 16 bit address and defining the base node. This will enable enable the systems routing tables to be set up.

1. Create of file in */root* called *router.conf*.

2. In *router.conf* the first line defines the nodes address and should be a four character hexidecimal number (e.g $111A$). Any value from 0000 to $FFFF$ may be selected. The second line is 0 if the node is the base node and 255 otherwise. There can only be one node defined as the base node for the network to function properly.

Chapter 8
CONCLUSION AND FUTURE WORK

In this paper, we investigated the problem of developing reliable localization algorithms using unreliable RSS measurements. We presented a stochastic framework for RSS-based localization that incorporates the RSS measurement uncertainty into the location estimation algorithm statistics of the beacon distances. We showed that stochastic framework for RSS-based localization can be incorporated into existing location estimation algorithms to improve the localization accuracy. We demonstrated the improvements in localization accuracy for two algorithms both in simulation and implementation in a WSN of imote2 devices.

Our future work will investigate of tradeoffs between computational overhead and localization accuracy using statistics of the beacon distances. In addition, we will investigate the translation of RSS measurement uncertainty into qualitative applications such as predicting link/channel quality and packet loss.

BIBLIOGRAPHY

[1] *Crossbow Technology: Imote2*, 2008. http://www.xbow.com/Products/iMote2.aspx.

[2] D. Bertsekas and R. Gallager. *Data networks. 1992.* Prentice Hall, 1992.

[3] N. Bulusu, J. Heidemann, and D. Estrin. GPS-less low-cost outdoor localization for very small devices. *IEEE Personal Communications*, 7(5):28–34, October 2000.

[4] S. Čapkun, M. Hamdi, and J.-P. Hubaux. GPS-free positioning in mobile ad-hoc networks. In *Proc. 34th Annual Hawaii International Conference on System Sciences (HICSS'01)*, pages 1–15, Maui, Hawaii, USA, January 2001.

[5] S. Čapkun and J.-P. Hubaux. Secure positioning of wireless devices with application to sensor networks. In *Proc. 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'05)*, Miami, FL, USA, March 2005.

[6] N. Cressie. *Statistics for Spatial Data.* John Wiley & Sons, Inc., 1993.

[7] B. Ferris, D. Hahnel, and D. Fox. Gaussian processes for signal strength-based location estimation. In *Proc. Robotics: Science and Systems*, Philidelphia, PA, USA, August 2006.

[8] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher. Range-free localization schemes for large scale sensor networks. In *Proc. 9th Annual International Conference on Mobile Computing and Networking, (MobiCom'03)*, pages 81–95, San Diego, CA, USA, 2003.

[9] M. Holland, R. Aures, and W. Heinzelman. Experimental investigation of radio performance in wireless sensor networks. In *Proc. 2nd IEEE Workshop on Wireless Mesh Networks (WiMesh'06)*, pages 140–150, Reston, VA, USA, September 2006.

[10] Y. M. Kwon, K. Mechitov, S. Sundresh, W. Kim, and G. Agha. Resilient localization for sensor networks in outdoor environments. In *Proc. 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, pages 643–652, Columbus, OH, USA, June 2005.

[11] A. LaMarca, J. Hightower, I. Smith, and S. Consolvo. Self-mapping in 802.11 location systems. In *Proc. 7th International Conference on Ubiquitous Computing (Ubicomp'05)*, pages 87–104, Tokyo, Japan, September 2005.

[12] L. Lazos and R. Poovendran. SeRLoc: Robust localization for wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 1(1):73–100, August 2005.

[13] Z. Li, W. Trappe, Y. Zhang, and B. Nath. Robust statistical methods for securing wireless localization in sensor networks. In *Proc. 4th International Symposium on Information Processing in Sensor Networks (IPSN'05)*, pages 91–98, Los Angeles, CA, USA, April 2005.

[14] D. Liu, P. Ning, and W. K. Du. Attack-resistant location estimation in sensor networks. In *Proc. 4th International Symposium on Information Processing in Sensor Networks (IPSN'05)*, pages 99–106, Los Angeles, CA, USA, April 2005.

[15] D. Lymberopoulos, Q. Lindsey, and A. Savvides. An empirical characterization of radio signal strength variability in 3-d IEEE 802.15.4 networks using monopole antennas. In *Proc. European Workshop on Wireless Sensor Networks (EWSN'06)*, pages 326–341, Zurich, Switzerland, February 2006.

[16] R. Nagpal, H. Shrobe, and J. Bachrach. Organizing a global coordinate system from local information on an ad hoc sensor network. In *Proc. 2nd International Workshop on Information Processing in Sensor Networks (IPSN'03)*, Palo Alto, CA, USA, April 2003.

[17] D. Niculescu and B. Nath. Ad hoc positioning system (APS). In *Proc. IEEE Global Telecommunications Conference (GLOBECOM'01)*, San Antonio, TX, USA, November 2001.

[18] D. Niculescu and B. Nath. DV based positioning in ad hoc networks. *Telecommunication Systems*, 22(1):267–280, January 2003.

[19] N. Patwari, A. O. Hero III, M. Perkins, N. S. Correal, and R. J. O'Dea. Relative location estimation in wireless sensor networks. *IEEE Transactions on Signal Processing*, 51(8):2137–2148, August 2003.

[20] J. A. Rice. *Mathematical Statistics and Data Analysis*. Duxbury Press, Belmont, CA, USA, 2nd edition, 1995.

[21] A. Savvides, C.-C. Han, and M. B. Srivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *Proc. 7th Annual International Conference on Mobile Computing and Networking (MobiCom'01)*, pages 166–179, Rome, Italy, July 2001.

[22] A. Schwaighofer, M. Grigoras, V. Tresp, and C. Hoffmann. GPPS: A gaussian process positioning system for cellular networks. *Advances in Neural Information Processing Systems*, 16, 2004.

[23] K. Yedavalli, B. Krishnamachari, S. Ravula, and B. Srinivasan. Ecolocation: A sequence based technique for RF localization in wireless sensor networks. In *Proc. 4th International Symposium on Information Processing in Sensor Networks (IPSN'05)*, pages 285–292, Los Angeles, CA, USA, April 2005.