Localization on the Pushpin Computing Sensor Network Using Spectral Graph Drawing and Mesh Relaxation

Michael Broxton *mbroxton@media.mit.edu*
Joshua Lifton *lifton@media.mit.edu*
Joseph A. Paradiso *joep@media.mit.edu*
Responsive Environments Group, MIT Media Lab, Cambridge, MA 02139, USA

This work approaches the problem of localizing the nodes of a distributed sensor network by leveraging distance constraints such as inter-node separations or ranges between nodes and a globally observed event. Previous work has shown this problem to suffer from false minima, mesh folding, slow convergence, and sensitivity to initial position estimates. Here, we present a localization system that combines a technique known as spectral graph drawing (SGD) for initializing node position estimates and a standard mesh relaxation (MR) algorithm for converging to finer accuracy. We describe our combined localization system in detail and build on previous work by testing these techniques with real 40-kHz ultrasound time-of-flight range data collected from 58 nodes in the Pushpin Computing network, a dense hardware testbed spread over an area of one square meter. In this paper, we discuss convergence characteristics, accuracy, distributability, and the robustness of this localization system.

I. INTRODUCTION

Location services are a fundamental tool in any wireless sensor network. In most applications, the time and location of sensor data are relevant, and many classes of applications require each node to know its location in a globally shared coordinate system. Recent research has leveraged location-rich information such as network adjacency[16], radio proximity[17], relative angles[13], or distance to a global sensory stimulus[3, 18] to achieve localization across many sensor nodes. These methods all place distance constraints on the locations of node coordinates, forming a (usually) non-linear optimization problem.

Localization can be boiled down to the general problem of drawing a graph given a set of vertices V and a set of edges E that connect certain vertices. A vertex may represent the location of a sensor node or it may represent the location of a distinct global event detected by a subset of nodes in the network (e.g. the distance to a lightning strike or beacon). Edges represent the distances between vertices in the graph. These distances may be relatively precise (e.g., a sonar time of flight measurement) or coarse (e.g. a binary value denoting whether two nodes are adjacent in the network). The goal of a localization algorithm is to determine the spatial coordinates of the vertices V given a set of edges E that provide constraints.

Under certain simplifying assumptions, linear solutions to this localization problem exist. For example, if certain "anchor" nodes have *a priori* knowledge of their position in the network, then another node can localize itself with a linear algorithm that takes its distance to the anchor nodes as input. Such anchor-based schemes include lateration[3] and bounding box methods such as APIT[8] and min-max[21].

In contrast, anchor-free localization schemes are generally non-linear but do not rely on such strong assumptions. Common non-linear techniques for sensor node localization include semi-definite programming[1], gradient descent[19], mesh relaxation[9], and metric multi-dimensional scaling (MDS)[22]. However, while these techniques can handle a more general set of constraints, they do not guarantee unique or even correct solutions. For example, a non-linear optimization algorithm may converge to a solution that does not reflect the true physical topology of the sensor nodes, but corresponds instead to a local minimum in the optimization landscape. See [20] and [14] for a more complete description of the degeneracy problem. In such cases, additional constraints, such as network distance between nodes[6, 20], can be used to disambiguate the solution. One way to speed up convergence and avoid bad solutions due to false minima is to pick a good initial guess of the nodes' actual coordinates and use this as a starting point for the optimization algorithm.

In this paper, we incorporate a method called *spectral graph drawing* that produces a set of initial coordinate estimates given only a set of distance constraints between nodes[10]. The resulting initial estimates closely approximate the actual spatial coordi-

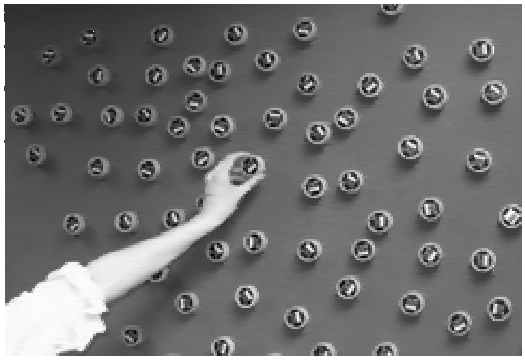

Figure 1: Pushpins are placed by hand on a 1.2-m by 1.2-m planar substrate that provides connections for power and ground. Placing a Pushpin node is as simple as pressing a thumbtack into a cork-board.

nates. These initial coordinate estimates subsequently serve as the starting point for a combination mesh relaxation/lateration algorithm, which in turn arrives at a final estimate of the position of each node in the network. In addition, our localization scheme includes pre- and post-processing steps for rejecting spurious, outlying measurements in the sensor data.

In principle, the distance constraints used in both the spectral graph drawing and mesh relaxation algorithms can originate from any ranging technique (e.g. nearest neighbor received radio signal strength). The work presented here uses time-of-flight measurements between each node and a globally observable ultrasonic "ping" event as distance constraints. Ping events are generated by a special handheld hardware device (the "Pinger") that simultaneously generates a flash of light and pulse of 40-kHz ultrasound at the press of a button. Distance is measured as the time difference of arrival between these two signals divided by the speed of sound.

Although the Pinger is an artificial signal source, it is meant to emulate a global sensor stimulus that might appear in a natural setting. For example, lightning and thunder provide a similar pair of stimuli that could be used to localize a sensor network spread over the area of several square kilometers This would be similar in principal to localizing lightning strikes, which has been demonstrated using a several-kilometer baseline array of sensors at NASA's Kennedy Spaceflight Center[24]. In principal, any pair of signals with coincident time and space origins and differing propagation speeds could be used to generate time-of-flight distance constraints. Munitions, fireworks, and gunshots are examples of phenomena that generate flashes of light and audible wavefronts. Signals also propagate at different speeds through different materials. For example, it might be possible to measure the distance to an explosion by detecting it both in the air and through the ground. Additionally, since the first pulse primarily serves to synchronize the sensor nodes, it is possible to localize by measuring the time of arrival of a single signal if the clocks of the sensor nodes are synchronized by some other means.

The remainder of this paper describes the experimental setup, the constituent individual algorithms used in our localization scheme, the workings of the localization scheme as a whole, a quantitative characterization of the results of our localization scheme, and directions for future work, including improvements, possible modifications, and open questions.

II. EXPERIMENTAL SETUP

Developing an algorithm on a sensor network is much like designing an electronic integrated circuit – much of the design process is spent either in simulation or on a physical prototype. Simulations can rapidly determine optimal system parameters and facilitate exhaustive testing over a wide range of inputs without invoking an expensive fabrication process. Such tests are impractical or too time-consuming to implement in a physical system. On the other hand, prototyping is essential for verifying that the design is robust to noise, non-isotropic signal propagation, part variability, processing and power limitations, and other non-ideal characteristics of the real world. We have capitalized on the strengths of both of these approaches in our development of the Pushpin localization system. In this section, we give an overview of our platforms for hardware prototyping and simulation.

II.A. Hardware Testbed

The Pushpin Computing testbed[12] is a dense wireless sensing platform with facilities for fast prototyping of hardware and algorithms. See Figure 1. The experiments and results presented in this paper made use of 58 Pushpin sensor nodes distributed randomly over an 1-m^2 planar area. A single Pushpin node is shown in Figure 2. The Pushpin name originates from the two metal pins protruding from the underside of each node from which the node derives its power and ground connections. These pins are pressed into a 1.2-m by 1.2-m board, where they make contact with two parallel sheets of metal foil (which carry power and ground) that are sandwiched between layers of insulating foam. A single Pushpin node measures 3-cm in diameter by 3-cm in height and consists of a modular stack of four circuit boards, one for each basic

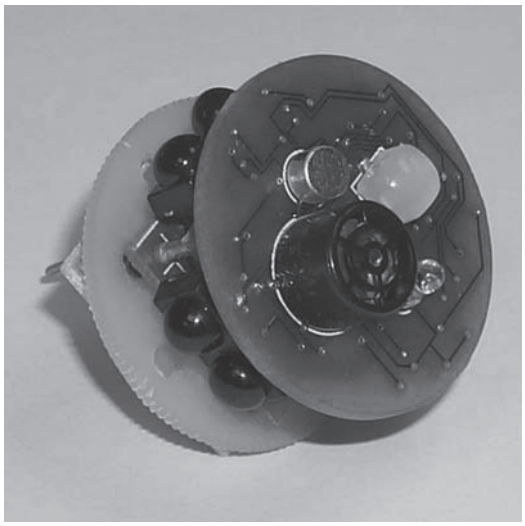

Figure 2: A single Pushpin node, shown fully assembled with the ultrasound time-of-flight expansion module. Each node is approximately 3-cm in diameter.

Figure 3: The "Pinger" delivers a simultaneous flash of light and burst of 40-kHz ultrasound. Each Pushpin can measure the difference in time of arrival of these two signals and thus calculate the distance between itself and the Pinger.

function of a wireless sensor node: power, communications, processing (each node is driven by a 22-mips, 8-bit 8051-core micro-controller), and sensing.

The ultrasound time-of-flight (TOF) expansion module is a sensing layer designed specifically for Pushpin localization experiments. It contains three sensors (a phototransistor, a sonar transducer, and an electret microphone) and one actuator (an RGB LED). Of these, the phototransistor and ultrasound transducer are used for localization, while the microphone and LED serve as additional I/O. The module is shown attached to a Pushpin in Figure 2.

The Pushpin platform is an unusually dense realization of a wireless sensor network; a feature made possible by its use of infrared (IR) communication, which is easier to constrain to short distance than radio frequency (RF) signals. Despite their high density, the Pushpins have only 10 network neighbors on average (approximately 17% of the network), hence the Pushpins have a similar network topology to a much sparser sensor network such as might be deployed "in the wild." However, whereas access to individual nodes in a large sensor network may be limited by the very large area over which it is distributed, the entire Pushpin network sits within arms reach of the sensor network developer, making it ideal for rapid prototyping and testing of distributed, ad hoc algorithms and applications. For example, new code can be simultaneously uploaded to every node in the network in less than a minute via an IR spotlight connected to a PC.

II.B. The Pinger

Using the localization system developed in this paper, a sensor network deployed in the field might use global phenomena detected in common across several nodes, such as lightning strikes or exploding munitions, to aid it in ad hoc localization. In order to test these algorithms on the Pushpin network, we developed a device that generates similar stimuli on a much smaller scale. This aptly-termed "Pinger" generates a simultaneous flash of light and burst of 40-kHz sonar, which are detected by the phototransistor and sonar receiver on the TOF Expansion module on each Pushpin. A Pushpin estimates its distance to the Pinger as the time difference of arrival of these two signals divided by the speed of sound in air (343.6 m/s at 20^oC). To generate a global stimulus for localization, the Pinger is held somewhere (anywhere) above the Pushpin network and triggered with the press a button. The Pinger is shown in Figure 3.

II.C. The Pushpin Simulator

The Pushpin Simulator is custom software that emulates a Pushpin network of between 10 and 100 nodes. Each virtual Pushpin is given its own memory and a thread of execution on the host machine. Pushpin threads interact by passing data packets to their nearest neighbors on the virtual network. This architecture closely resembles the distributed nature of real Pushpins, therefore code written for simulated Push-

pins is very similar to code for Pushpins in the hardware testbed. In essence, simulated Pushpins collaborate, share state, and respond to sensor data by passing and processing network messages. In this respect, the Pushpin Simulator is a high-level simulator, as opposed to low-level simulators such as Avrora[23], which provides a cycle-accurate, processor instruction level simulation.

The simulator allows easy control over the input to the localization algorithm; namely the ultrasound time-of-flight measurements. The simulator can either use real measurements recorded from the hardware testbed or simulated measurements generated according to some statistical model. Similarly, the placement of the simulated Pushpins in the virtual environment can be random, or set to the actual "ground truth" positions of the real Pushpins in the hardware testbed.

III. LOCALIZATION ALGORITHMS

There are three principal algorithms in the Pushpin localization system: spectral graph drawing, mesh relaxation, and lateration. We focus here primarily on spectral graph drawing and mesh relaxation, since they have received relatively little attention in the current localization literature. Lateration, which is a form of triangulation, is a comparatively well-known technique for sensor network localization[11].

III.A. Spectral Graph Drawing

Spectral Graph Drawing (SGD) is a technique for producing a set of vertex coordinates in k dimensions given only a set of edge lengths between vertices. The resulting coordinates closely adhere to the constraints imposed by the edge lengths. Like multi-dimensional scaling, force-directed graph drawing, and principal component analysis, SGD was conceived as a technique to help visualize high dimensional data in a low dimensional space. The technique itself is quite old, dating back to the work of Hall[7] in 1970. However, it has seen little use since that time and has only recently been proposed as a technique for sensor node localization by Yehuda Koren[10] and Craig Gotsman[6]. These techniques are summarized below.

A sensor network can be described abstractly as a set of vertices $V = \{V_1, ..., V_n\}$ and edges $E = \{\langle i, j \rangle\}$ that make up a graph $G(V, E)$. Edges have associated weights w_{ij} proportional to the *adjacency* of two vertices V_i and V_j. The weight is larger if the vertices are more closely connected (e.g. if they are physical closer to each other). If two vertices are not connected, $w_{ij} = 0$. In practice, Koren recommends that a measured distance between two nodes be converted to an adjacency weight via $w_{ij} = exp(-d_{ij})$ or $w_{ij} = \frac{1}{1+d_{ij}}$ (we use the former mapping in our implementation)[10].

To clarify, sensor nodes and Pinger locations are *both* represented as vertices in the graph. The lengths d_{ij} between sensor nodes and pings (derived from TOF measurements) are used to compute these weights. Since the distances between pairs of sensor nodes or pairs of pings is not directly measured, these weights are set to zero[1].

The connectedness of the graph can be summarized by placing these weights in an *adjacency matrix* A, where

$$A_{ij} = \begin{cases} 0 & i = j \\ w_{ij} & i \neq j \end{cases}$$

Next, we define the *degree* of a node to be the sum the weights between itself and other nodes:

$$deg(i) = \sum_j w_{ij}$$

The values of $deg(i)$ are placed into a diagonal matrix D such that $D_{ii} = deg(i)$.

These definitions allow us to formulate a 1-dimensional graph drawing problem. Specifically, we would like to find a n-dimensional vector called $\mathbf{x}$ that contains the 1-dimensional coordinates of n vertices by solving the following constrained optimization problem:

$$\mathbf{x} = \arg\min_{\mathbf{x}'} S(\mathbf{x}') \tag{1}$$

$$\text{given: } \mathbf{x}^T D\mathbf{x} = 1, \mathbf{x}^T D 1_n = 0$$

$$\text{where: } S(\mathbf{x}) = \sum_{<i,j>\in E} w_{ij}(x_i - x_j)^2$$

Here, 1_n is a vector of length n that contains only 1's. There are two forces at play in this minimization problem. Minimizing $S(\mathbf{x})$ tends to shorten the lengths between the vertices proportionally according to the weights w_{ij}, thereby pulling the graph into the correct shape. At the same time, the constraint that $\mathbf{x}^T D\mathbf{x} = 1$ provides a repulsive force, preventing this minimization process from collapsing into a degenerate solution in which all edge lengths are all equal to

[1]Logical distance over the network can be used to approximate the distance between pairs of sensor nodes. These distance estimates can be used to compute approximate weights, and ultimately, an approximate SGD solution. Section V.B.1 contains SGD results where node-to-node distances are estimated to be proportional to logical distance over the network.

zero. The second constraint, $\mathbf{x}^T D 1_n = 0$, is equivalent to saying that $\mathbf{x}$ should have a mean of zero. That is, it removes translational ambiguity from the solution vector by forcing $\mathbf{x}$ to be zero-centered.

The presence of D in the constraints achieves what is called *degree normalization* of the solution vector $\mathbf{x}$. Without it, a vertex that has a much lower degree than the rest (e.g. if it had missing measurements or had fewer constraints to begin with) has significantly less attractive force acting on it in the minimization of $S(\mathbf{x})$. As a result, it will be overly separated from its neighbors when its final coordinates are computed, and the remainder of the nodes will be closely clustered in comparison. A degree normalized solution corrects for this, adding an extra repulsive force to nodes with higher degrees, thus preventing them from bunching up at the origin while one vertex with a substantially lower degree is left as an outlier. Degree normalization, one of Koren's primary contributions in [10], is essential for creating a well-distributed layout of vertices that is useful as an initial guess for mesh relaxation.

The power of spectral graph drawing lies in the fact that this minimization problem has a very convenient solution: *The vector of coordinates* $\mathbf{x}$ *that minimizes equation (1) according to the given constraints is the eigenvector* v_2 *associated with the* **second** *largest eigenvalue of the matrix*

$$Z = \frac{1}{2}(I + D^{-1}A) \tag{2}$$

A lengthy, though not complicated, proof of this can be found in the appendix of [10]. The method for finding coordinates in a second dimension is identical except that $\mathbf{x}$ is forced to be orthogonal to both v_1 and v_2. In this case, the solution is the eigenvector v_3 that is associated with the third largest eigenvalue of (2). Coordinates in a third dimension can be found if $\mathbf{x}$ is forced to be orthogonal to v_1, v_2, and v_3, and so on.

In essence, the problem of determining a geometrical layout for a set of vertices has been reduced to an ordinary eigenvector computation. Many well-understood algorithms exist for finding eigenvectors. In this case, power iteration is an appropriate choice, since it is convenient to apply the constraints at each step in the iteration ($\mathbf{x}^T D\mathbf{x} = 1$ via normalization, $\mathbf{x}^T D 1_n = 0$ via Gram-Schmidt Orthogonalization)[5].

An attractive property of spectral graph drawing is that it can be formulated as a fully distributed algorithm that requires only neighbor-to-neighbor communication transactions. Gotsman and Koren demonstrate in [6] that the power iteration technique is mathematically equivalent to directing a node to repeatedly move its estimated coordinates to the centroid of the estimated coordinates of its neighbors. This technique, sometimes referred to as the *diffusion* technique for localization, has also been developed by Bulusu et. al.[4]. However, these researchers do not explicitly draw a connection to spectral graph drawing in their implementations, and therefore do not benefit from the deeper theoretical insight of the more mathematical formulation given by Gotsman and Koren.

We have opted to implement a centralized version of this algorithm on a single node in our work because in our case, there is global state (the locations of the pinger events) that cannot be shared using only neighbor to neighbor interactions. However, regardless of whether it is centralized or distributed, the result of running the algorithm is the same.

III.B. Mesh Relaxation

A mesh relaxation algorithm aims to simulate a physical system of masses connected by springs. It is a useful approach in a distributed system, since information passing need only occur between nodes connected by a distance constraint. This can greatly reduce communication overhead, especially when distance constraints only exist between neighbors on the network.

Consider a graph, or mesh, with n vertices. Each vertex V_i maintains an estimate $\mathbf{X}_i[t]$ of its own 3-dimensional spatial coordinates. The goal of mesh relaxation is to incrementally improve the estimate during each discrete time step t. Distances between some, but not necessarily all, vertices have been measured and recorded such that each vertex V_i retains a set of measurements between itself and other nodes in the mesh, $D_i = \{d_{ij}\}$. Estimated coordinates can initially be chosen randomly, selected using the results of the spectral graph drawing technique described in Section III.A, or chosen based on additional information that may be available, such as inertial measurements integrated over time[9] or an approximate coordinate system built using logical distance over the network[20].

During each discrete time step in the relaxation process, every node computes the *force* acting on it due to the constraints imposed by the positions and relative distances of its neighbors. The force due to each neighbor is proportional to the difference between the estimated distance and the measured distance to the neighbor. This is a vector with a magnitude and direction equal to

$$\begin{aligned} |\mathbf{f}_{ij}[t]|| &= k(||\mathbf{X}_i[t] - \mathbf{X}_j[t]|| - d_{ij}) \\ \angle\mathbf{f}_{ij}[t] &= \angle(\mathbf{X}_i[t] - \mathbf{X}_j[t]) \end{aligned}$$

In the equation above, $||\cdot||$ is the Euclidean norm and k is the spring constant that controls the speed of convergence of the algorithm. We found the value of k must be tuned depending on the number of vertices and the lengths of the distance constraints. If k is too large, the system will be unstable and oscillations will result. If it is too small, convergence will be slow. We found that $k = 0.15$ was appropriate in our simulation with 15 vertices (10 anchor Pushpin nodes and 5 ping events) and average distance constraints of 1.0-m (distance between Pushpin nodes and ping events). The sensitivity of k to the number of vertices and edges in this type of mesh relaxation should be carefully considered in a system without a fixed number of vertices and edge constraints, as we have.

The total force acting on a vertex is the sum of these individual forces; $\mathbf{F}_i[t] = \sum_j \mathbf{f}_{ij}[t]$. A node's new coordinate estimate is the sum of the old estimate plus this force; $\mathbf{X}_i[t+1] = \mathbf{X}_i[t] + \mathbf{F}_i[t]$.

IV. LOCALIZATION SYSTEM

The Pushpin localization system combines non-linear and linear techniques, capitalizing on both the versatility of the "anchor-free" spectral graph drawing and mesh relaxation approaches as well as the efficiency and ease of use of the "anchor-based" lateration algorithm. In addition, the system includes pre and post-processing steps for sensor calibration and outlier rejection. The complete, end-to-end system is described below.

IV.A. Calibration

Error in ultrasound time-of-flight measurements can originate from dispersion, speckle, changing air currents, interference among ultrasound transmitters, part variability, discretization, and analog conditioning circuitry. Our tests indicate that the largest source of systematic error is due to a limitation of the TOF module sonar detection circuitry, which consists of a fixed-threshold, uni-polar rising-edge discriminator. Ideally this circuit should detect the sonar signal at some point within the first full cyle of when it arrives at the Pushpin. This places the fundamental limit on the precision of detecting a sonar signal at roughly one full cycle of the sonar wave, or 1-cm. In practice, we have found that the receiving transducer takes several cycles to "ring up" to the discriminator threshold. This induces a delay that is proportional to the distance between the Pinger and the TOF module. We believe this phenomenon to be the result of increasing dispersion of the ultrasound ping as the distance between the Pushpins and the Pinger is increased.

Based on a characterization of the error in measurements made over a range of distances between the Pinger and a single Pushpin, we have devised a simple linear calibration scheme that removes distance dependent systematic error in time-of-flight measurements. This calibration model is fairly simplistic – it accounts for neither the additional sonar signal attenuation (and corresponding time delay) that may result at different Pinger angles, nor the variability of ring-up characteristics across different sonar receiver parts. Nonetheless, the statistical distribution of the remaining TOF measurement error after calibration has been applied is closely compatible with a zero-mean gaussian with a standard deviation of 0.57-cm, though the plot did show some evidence for broad tails and small asymmetry that suggest other un-modeled sources of error[2].

IV.B. Pre-processing

Spurious outliers are caused by interference and multi-path (reflections) of the sonar signal. Bad measurements can foul the localization results if they are not detected and rejected. In a sensor network, this must occur in a distributed manner. We have formulated a distributed method for rejecting outliers based on exchanging distance measurements between nodes in a one-hop communication neighborhood. A measurement is rejected if it differs by more than a certain number of standard deviations from the median of measurements made at neighboring nodes. This technique capitalizes on the implicit notion that communication range roughly correlates to physical distance in a wireless sensor network. Using this technique, we have found that roughly 10 nodes out of 58 will reject at least one of five measurements during a localization run. If a node rejects too many measurements, it may no longer have enough constraints for localization. If this is the case, the node will withdraw from the localization process. On an average trial, 2 nodes out of 58 will withdraw for this reason.

IV.C. Primary localization

During primary localization, a small subset of nodes on the network called *anchor* nodes cooperate to estimate their own coordinates as well as the coordinates of the global events generated by the Pinger. A distributed, ad hoc election process selects ten anchor nodes and one *origin* node that have five good

(not outlying) range measurements[2]. The origin node, which is so named because it is arbitrarily assigned the coordinates $(0, 0, 0)$, plays the unique role of collecting and storing the range measurements of all the anchor nodes in into an adjacency matrix[3]. This is then used by the spectral graph drawing and mesh relaxation algorithms, which run in centralized form on the origin node. This computation is fairly efficient for an adjacency matrix built from 10 anchors and 5 global events. An 8-bit microcontroller with 128-kB of memory (for the adjacency matrix and temporary variables) and a 22-MIPS processor would be capable of running both algorithms in under a minute. Hence, although the origin node does the majority of the processing at this stage, it need not be endowed with more processing power than any other sensor node in the network.

IV.D. Secondary localization

During primary localization, most of the nodes are passive; i.e. they collect time-of-flight measurements and reject outliers, but they do not participate in the primary localization process. However, the origin broadcasts the coordinates of the global Pinger events to all passive nodes at the end of the primary phase of localization. This information, along with the distances a passive node had previously measured to the global events, is sufficient to compute a position via lateration[11].

IV.E. Post-Processing

After both primary and secondary localizations have completed, the nodes must perform a final check to ensure that their estimated coordinates are reasonable. Once again, the implied physical proximity of nodes that are neighbors on the network is used to detect nodes that have computed coordinates in gross disagreement with the coordinates of their neighbors. If a node's coordinates are more than a certain threshold from the median of it neighbors' coordinates, it withdrawals from localization. As with the pre-processing technique, outlier rejection requires only neighbor-to-neighbor communication transactions.

[2]Ten nodes ensures that there is sufficient information to find a unique localization solution. See [2] for more details.

[3]Alternatively, the adjacency matrix could be filled with node-to-node constraints generated by logical distance over the network (hop count) in addition to or in lieu of TOF measurements.

V. RESULTS

We now present a characterization of the Pushpin localization system described in Section IV. Ultrasound events were generated by the Pinger device (Section II.B), which was manually triggered at random locations within a hemisphere above the plane of the Pushpin network. All pings occurred within 2-m of some Pushpin in the network. We collected 10 data sets, each consisting of five time-of-flight measurements per Pushpin. Data sets were downloaded through the network to a PC, imported into the Pushpin Simulator, and played back for a set of 58 simulated Pushpins. In each of the test configurations described in the sections below, a total of 10 localization trials were run per data set, yielding 100 sets of estimated coordinates.

V.A. Measuring Localization Error

In our case, "ground truth" coordinates are obtained by photographing the array of Pushpins using a digital camera with a telephoto lens (to reduce the effects of image warping), and digitally extracting the coordinates with custom image manipulation software. A 0.1-m by 0.1-m square is included in the image for scale. In order to assess the accuracy of our localization system, we considered two methods for aligning the the coordinates produced by the localization algorithms (*estimated* coordinates) with the "ground truth" coordinates (*ground* coordinates): (1) A transform comprised of only a translation, rotation, and possible reflection (RTR); and (2) a more general homogeneous projective transformation that can also include scaling and shearing. When the fit is computed as in (1), we refer to it as an *RTR fit*. The fit computed as in (2) is referred to as an *LLSE fit* because the projective transformation can be found most easily by computing a linear least squares estimate. More details of the fitting process can be found in [2].

After the estimated coordinates have been fit to the ground coordinates using one of the above methods, the localization error can be assessed. The objective is to determine the average error between the transformed estimated coordinates $(\hat{x}', \hat{y}', \hat{z}')$ and a set of ground coordinates (x, y, z) obtained manually by photographing the array as described above. The following *mean absolute error* metric can be used to compute the localization error for n sensor nodes:

$$\bar{e}_{mae} = \frac{\sum_{i=1}^{n} \sqrt{(x_i - \hat{x}_i')^2 + (y_i - \hat{y}_i')^2 + (z_i - \hat{z}_i')^2}}{n} \quad (3)$$

	Network SGD	TOF SGD	Mesh Rlx.
Mean	5.53	3.07	2.12
Median	5.26	2.76	1.83
Std Dev	2.84	1.87	1.42
Min	1.73	0.75	0.49
Max	10.5	6.47	4.82

Table 1: Summary statistics of the error between estimated and ground truth coordinates at the end of primary localization *for the 10 anchor nodes only*. Statistics are accumulated over 100 localization trials that span 10 different Pinger configurations. All data is fit using the LLSE. Measurements are in centimeters.

Other statistics including the variance, median, minimum, and maximum values are computed in a similar manner using other standard statistical formulae.

V.B. Localization Accuracy

We begin our characterization with a look at the accuracy of the spectral graph drawing and mesh relaxation algorithms when localizing the 10 anchor nodes. Next, we characterize the accuracy of the combined spectral graph drawing, mesh relaxation, and lateration algorithm when localizing all 58 nodes. Recall that the localization of all 58 nodes builds upon the localization of the 10 anchor nodes.

V.B.1. Accuracy over Anchor Nodes

This test assesses the accuracy of anchor placement using the spectral graph drawing and mesh relaxation algorithms during the primary phase of localization (Section IV.C). Two variants of spectral graph drawing have been considered. One uses only the hop count between anchor nodes on the network as a distance metric in the adjacency matrix (we refer to this as *network SGD*), while the second variant uses more precise time-of-flight distance measurements (this is *TOF SGD*). In a third scenario, TOF SGD results were refined using mesh relaxation. One hundred localization trials (ten for each data set from the hardware test bed) were run for each of the three variants. Localization statistics were computed after each localization trial. The average statistics over all trials appear in Table 1.

One notable result in the table is that TOF SGD on its own is only one centimeter less accurate than TOF SGD refined by mesh relaxation. This suggests that spectral graph drawing is useful as a stand alone method for localization. Omitting mesh relaxation entirely would lead to a simpler, though slightly less accurate solution. Furthermore, network SGD achieves

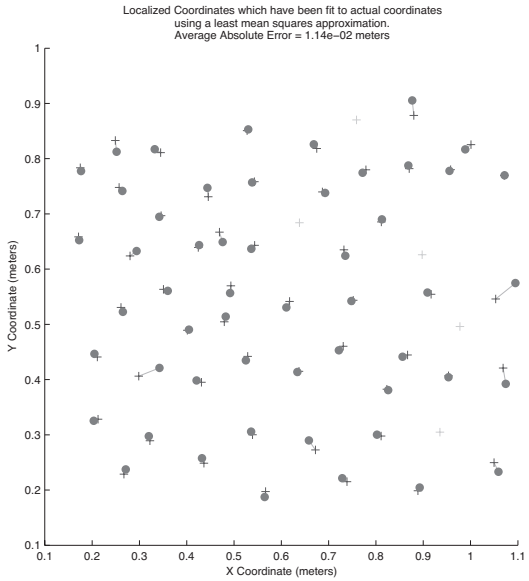

Figure 4: Estimated Pushpin coordinates generated using the Pushpin localization system that have been aligned with ground coordinates via a LLSE fit. The crosses denote ground truth coordinates and the circles denote coordinates estimated by the localization system. The lines indicates the correspondence between estimated and ground coordinates. Cyan colored nodes have been rejected as outliers.

an accuracy that is roughly half the average inter-node spacing of the Pushpins – a level sufficient for applications requiring only coarse localization.

V.B.2. Accuracy over all Nodes

The subsequent tests assess the accuracy of the *overall* localization system described in Section IV. Figure 4 shows the results of a typical localization trial on the Pushpin localization system as fit to the ground coordinates using the LLSE. The plot shows that the majority of nodes achieve very high localization accuracy, though five nodes were not able to localize at all. Taken on its own, however, this plot does not give much insight into how consistent localization results are from trial to trial, an issue examined in the following tests.

Next, we considered how our localization system performed given simulated time-of-flight data generated from three different error models.

- *No Error*: Used to determine the baseline accuracy of the localization system in the ideal case.
- *Gaussian Error*: A zero mean, 0.57-cm std. dev. gaussian based on the characterization of the time-of-flight sonar measurements that was briefly discussed in Section IV.

	Real Pings	Simulated Pings		
	Pinger	No Error	Gaussian Error	GM Error
Mean	2.30	0.06	1.26	3.65
Median	1.69	0.04	1.08	2.66
Std Dev	2.36	0.07	0.82	2.99
Min	0.20	0.01	0.11	0.30
Max	13.5	0.47	4.39	13.1
% Unlocalized	10%	0%	5%	3%

Table 2: Summary statistics for the error between estimated and ground truth coordinates *for all 58 nodes* after the complete localization system has run to completion. Statistics are accumulated over 100 localization trials that span 10 different Pinger configurations. All data is fit using the LLSE. Measurements are in centimeters.

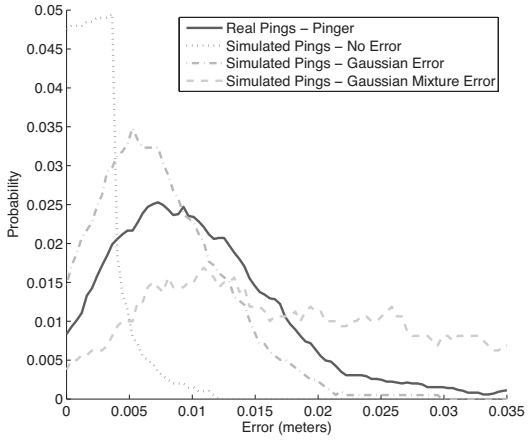

Figure 5: Statistical distribution of localization error over 100 localization trials. Various sources of time-of-flight measurements are represented.

- *Gaussian Mixture Error*: The distribution of time-of-flight measurement error described in Section IV showed evidence of heavy tails and a slight asymmetry that is was better fit using a mixture of gaussians.

Figure 5 shows the error distribution for various measurement sources over 100 localization trials. Outlier rejection was active throughout these tests. Table 2 shows various statistics averaged over the trials. Note that the distribution of localization error is asymmetric: the localization error for most nodes is very low, but the average is inflated by a few nodes that have unusually high error. In general, these are minor outliers that are missed by the outlier detection mechanisms.

As expected, simulated pings with no measurement error lead to extremely low localization error (0.06-cm)[4]. Our expectation that the localization error using real pings is generally greater than it is under the simple gaussian error model is also confirmed in the Figure. Furthermore, the difference in the mean values of these distributions is less that one standard deviation. Finally, the error distribution observed using real data falls in between the distributions for simulated pings under the Gaussian and Gaussian Mixture error models. This suggests that we have achieved a reasonable understanding of the error sources in our system.

Finally, we direct the reader to compare these results to those in [3], where we characterize a similar localization system on the Pushpin system that uses a simple Lateration-based algorithm rather than SGD and MR. With Lateration alone, we achieved a mean localization error of 4.93-cm and an average std. dev. of 3.04-cm, which is clearly higher than our results here. For a more detailed comparison of these two systems, see [2].

V.C. Outlier Rejection

In this section we quantify the effectiveness of the outlier rejection techniques introduced in Section IV. Outlier rejection allows the Pushpins to maintain high localization accuracy in the face of spurious errors in sensor measurements. We separately consider two types of outlier rejection used in the Pushpin localization system: (1) Rejection of outlying time-of-flight measurements prior to localization, and (2) rejection of outlying coordinate estimates after localization. However, we emphasize that both of these schemes should be used in concert for best results.

V.C.1. Outlying TOF measurements

Nodes reject measurements that grossly disagree with the measurements made by their neighbors on the network. The threshold for rejection of a ping event is given in terms of a number of standard deviations

[4]We believe that the unusual shape of the error distribution for the "no error" error model may be due to instability in the mesh relaxation process that occurs when the solution is very near convergence.

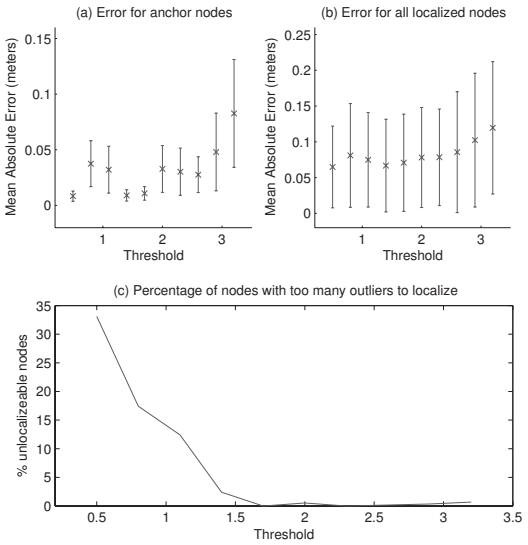

Figure 6: The effects of a preprocessing step that rejects outlying time-of-flight measurements (Section IV.B). The threshold for rejection is the number of standard deviations from the norm of a node's neighobors' time-of-flight measurements. (a) shows the average localization error of *ten anchor nodes only* for various threshold values. (b) shows the average localization error of *all 58 nodes* (in (a) and (b), the x marks the mean for the ten trials, and the error bars show one standard deviation). (c) shows the number of nodes that were unable to localize because too many of their distance constraints were rejected as outliers. We found a threshold of 1.8 to be a reasonable trade-off between overall accuracy and the number of rejected nodes. The post-processing step of coordinate outlier rejection was disabled for these tests.

from the median of neighboring nodes' measurements for the same event. In our test, we tried 10 different threshold values that varied between 0.5 and 3.2. Ten localization trials were run for each threshold value. For these tests, only outlying measurements, not outlying coordinate estimates, were rejected. The results for various threshold values are plotted in Figure 6.

Frame (a) in the figure shows the localization error for the anchors nodes only. The plot indicates that mean localization error can be drastically reduced if the rejection threshold is sufficiently low. The standard deviation of localization error is also dramatically reduced when outlier rejection is active. Frame (b) shows a similar but less pronounced trend when error is considered over all 58 nodes. However, frame (c) of Figure 6 shows that there is a trade-off to be made between the desired localization accuracy and the number of nodes that cannot localize because they have rejected too many measurements as outliers.

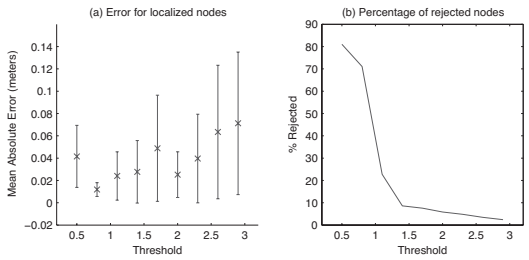

Figure 7: The effects of a post-processing step that rejects nodes with outlying coordinates (Section IV.E). The threshold for rejection is the number of standard deviations from the median of a node's neighbors' coordinates. (a) shows average localization error for various threshold values (x denotes the mean and the error bars show one standard deviation) (b) plots the number of nodes that were unable to localize because they were rejected as outliers. We found a threshold of 1.7 to be a reasonable trade-off between overall accuracy and the number of rejected nodes.

V.C.2. Outlying coordinate estimates

The goal of coordinate outlier rejection is to remove nodes that have been assigned coordinates that grossly disagree with the coordinates of their neighbors. This test assesses how localization accuracy and the number of un-localized sensor nodes vary as a function of the coordinate outlier rejection threshold. The threshold was varied from 0.5 to 2.9 in increments of 0.3 standard deviations. Ten localization trials were run for each threshold value. Figure 7 shows the results. An increase in localization error and decrease in the percentage of unlocalized nodes can once again be seen as the threshold is increased.

V.D. Shearing and Scaling of the Coordinate System

As discussed in Section V.A, the error of the Pushpin localization system was measured in two ways – 1) fitting the estimated coordinates to the ground coordinates using a rotation-translation-reflection (RTR) fit and then calculating the average error, and 2) fitting the estimated coordinates to the ground coordinates using a linear least squares estimate (LLSE) fit and then calculating the average error. The results thus far given concern only the LLSE fit. We found that the RTR fit results in a consistently larger average error than the LLSE fit, implying there is a uniform scaling and shearing of the estimated coordinates arrived at by the Pushpin localization system. See Figure 8. The LLSE fit compensates for such scaling and shearing, whereas the RTR fit cannot, giving an average error for the trial depicted in Figure 8 of 0.09-cm and 32.38-cm, respectively.

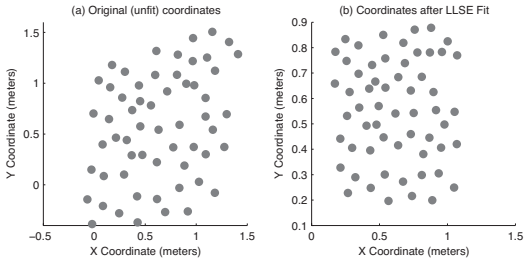

Figure 8: The Pushpin localization system produces estimated coordinates with uniform shearing and scaling that cannot be corrected by a rotation-translation-reflection fit (left), but is compensated for by a linear least squares estimate fit to the ground coordinates (right). This implies that the system is underconstrained – i.e. there are numerous solutions that satisfy the edge constraints generated from five Pinger events. The original data were simulated pings without added error.

Although the average error of a RTR fit of the estimated coordinates is rather large, the distance contraints imposed by the time-of-flight measurements to Pinger events are satisfied nonetheless. For example, although the average error of a RTR fit of estimated coordinates for the trial depicted in Figure 8 is 32.38-cm, the largest disagreement between a time-of-flight measurement of a Pinger event and the estimated distance to the same event is only 0.01-cm.

This implies the system is underconstrained; there are numerous solutions that satisfy all the constraints. Obtaining a stricter set of constraints so as to eliminate this degeneracy is the domain of graph rigidity theory [20]. Both [20] and [14] also explicitly address the problem of degenerate solutions in sensor network localization by selecting anchor nodes using heuristic and geometric constraints. Nonetheless, the estimated coordinates arrived at by the Pushpin localization system are still valid so long as a LLSE fit can be made, for example, by knowing a priori the global coordinates of a small number of ground control points that have been identified and localized in the frame of reference of the sensor network.

VI. FUTURE WORK

There are obvious avenues for improving our localization results. First, the ability to make use of more than five Pinger events could improve overall localization accuracy. This would be a simple extension, since the algorithms we have used readily incorporate additional pings by increasing the size of the adjacency matrix or by averaging results for different combinations of pings. Utilizing extra pings, it should be possible to reduce average localization error to roughly 1-cm, which is the theoretical limit on the accuracy for detecting an ultrasound signal using a unipolar rising-edge discriminator with a fixed threshold. While mesh relaxation has worked well in our tests, it converges relatively slowly compared to other methods. In our tests, it took an average of 5,000 iterations of relaxation before the mesh converged. Several more efficient approaches are well known in the field of optimization, and some of these have recently been applied to the problem of localizing sensor nodes. Some work in this area includes semi-definite programming[1], non-linear least squares[15], and distributed Kalman filters[21].

A logical extension of our current approach of basing localization constraints on the distances to global sensor phenomenon might be called “ambient localization.” The key idea here is that the localization system should rely as little as possible on the mechanism for generating global stimuli and instead treat such stimuli as parts of the environment rather than additional infrastructure. This work at least shows progress toward this goal by obviating the need for prior knowledge of the absolute position of the source of a pair of global stimuli. However, we would like to generalize our approach further by instead measuring the *time of arrival* of a global signal. In this scenario, the absolute time origin of the signal becomes another parameter to be estimated. The solution to this higher dimensional search problem requires additional constraints and more computation, but these are readily available either from additional participating nodes, or from additional global events. The spectral graph drawing, mesh relaxation, and lateration algorithms all readily generalize to higher dimensional search spaces.

VII. CONCLUSION

We have demonstrated a unified localization system comprising several algorithms that produces consistent results with low localization error, even when using noisy real-world sensor data. Our approach employs two non-linear localization techniques, namely spectral graph drawing and mesh relaxation, as well as a linear lateration algorithm. Overall, the Pushpin localization system achieves a mean absolute error of 2.3-cm and an error standard deviation of 2.36-cm when using ultrasound time-of-flight measurements with a simple rising-edge detector. Supplemental techniques for outlier rejection have been shown to be very effective at decreasing localization error when constraints are generated using data from real sensors.

VIII. Acknowledgments

We wish to thank the Things That Think Consortium and other sponsors of the MIT Media Lab for their generous support. Portions of this work are supported by NSF grant #ECS-0225492.

References

[1] P. Biswas and Y. Ye. Semidefinite programming for ad hoc wireless sensor network localization. In *Proc. of the 2nd ACM int'l conference on wireless sensor networks and applications*, pages 46 – 54, 2004.

[2] M. Broxton. Localization and sensing applications in the pushpin computing network. Master's thesis, EECS Department, Massachusetts Institute of Technology, 2005.

[3] M. Broxton, J. Lifton, and J. Paradiso. Localizing a Sensor Network via Collaborative Processing of Global Stimuli. In *Proc. of the European Conference on Wireless Sensor Networks*, 2005.

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

[5] J. W. Demmel. *Applied Numerical Linear Algebra*. SIAM, 1997.

[6] C. Gotsman and Y. Koren. Distributed graph layout for sensor networks. In *Proc. of the International Symposium on Graph Drawing*, 2004.

[7] K. M. Hall. An r-dimensional quadratic placement algorithm. *Management Science*, 17:219 – 229, 1970.

[8] T. He, C. Huang, B. Blum, J. Stankovic, and T. Abdelzaher. Range-Free Localization Schemes in Large Scale Sensor Networks. In *Proc. of the 9th annual int'l conference on Mobile computing and networking*, pages 81–95, 2003.

[9] A. Howard, M. J. Mataric, and G. Sukhatme. Relaxation on a Mesh: a Formalism for Generalized Localization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2001.

[10] Y. Koren. On spectral graph drawing. In *Proc. of the 9th International Computing and Combinatorics Conference (COCOON)*, 2003.

[11] K. Langendoen and N. Reijers. Distributed localization in wireless sensor networks: a quantitative comparison. *The International Journal of Computer and Telecommunication Networking*, 43(4):499 – 518, November 2003.

[12] J. Lifton, M. Broxton, and J. Paradiso. Experiences and direction in pushpin computing. In *Information Processing in Sensor Networks, Special track on Platform Tools and Design Methods for Network Embedded Sensors (SPOTS)*, 2005.

[13] S. Lindebner, H. H. Fruehauf, J. Heubeck, R. Wansch, and M. Schuehler. Evaluation of direction of arrival location with a 2.45 ghz smart antenna system. to appear.

[14] D. Moore, J. Leonard, D. Rus, and S. Teller. Robust distributed network localization with noisy range measurements. In *Proc. of the ACM SenSys*, 2004.

[15] R. L. Moses, D. Krishnamurthy, and R. M. Patterson. A Self-Localization Method for Wireless Sensor Networks. *EURASIP Journal on Applied Signal Processing*, pages 348 – 358, 2003.

[16] R. Nagpal, H. Shrobe, and J. Bachrach. Organizing a Global Coordinate System from Local Information on an Ad Hoc Sensor Network. In *2nd International Workshop on Information Processing in Sensor Networks (IPSN)*, April 2003.

[17] N. Patwari and A. O. Hero. Using Proximity and Quantized RSS for Sensor Localization in Wireless Networks. In *Proc. of the 2nd ACM int'l conference on Wireless sensor networks and applications*, pages 20 – 29, 2003.

[18] N. Patwari and A. O. Hero. Manifold learning algorithms for localization in wireless sensor networks. In *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, May 2004.

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

[20] N. B. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller. Anchor-Free Distributed Localization in Sensor Networks. Tech Report 892, MIT Laboratory for Computer Science, 2003.

[21] A. Savvides, H. Park, and M. Srivastava. The Bits and Flops of the N-Hop Multilateration Primitive for Node Localization Problems. In *Proc. of the 1st ACM int'l workshop on Wireless sensor networks and applications*, pages 112–121, 2002.

[22] Y. Shang, W. Ruml, Y. Zhang, and M. P. J. Fromherz. Localization from Mere Connectivity. In *Proc. of the 4th ACM int'l symposium on mobile ad hoc networking and computing*, pages 201 – 212, 2003.

[23] B. L. Titzer, D. K. Lee, and J. Palsberg. Avrora: Scalable sensor network simulation with precise timing. In *Proc. of the Fourth International Conference on Information Processing in Sensor Networks (IPSN)*, 2005.

[24] Unknown Author. Efficient Processing of Data for Locating Lightning Strikes. Technical Brief KSC-12064/71, NASA Kennedy Space Flight Center.