Localizing a Sensor Network via Collaborative Processing of Global Stimuli

Michael Broxton
MIT Media Laboratory
Cambridge, MA USA
Email: mbroxton@media.mit.edu

Josh Lifton
MIT Media Laboratory
Cambridge, MA USA
Email: lifton@media.mit.edu

Joe Paradiso
MIT Media Laboratory
Cambridge, MA USA
Email: joep@media.mit.edu

***Abstract*— In order for nodes in a sensor network to meaningfully correlate their sensor readings, they must first determine their position in a globally shared coordinate system. Though there are many approaches which are suitable for achieving localization in the general case, sensor nodes are uniquely suited to use their sensing capabilities to aid them in this task. Global events which are detected in the environment surrounding the sensor network can serve as points of correspondence which, through collaborative processing on the network, provide nodes with sufficient information to compute their position.**

We have implemented an algorithm based on this approach in the Pushpin Computing sensor network: a dense, 55 node network which is spread over an area of 0.5 square meters. By queuing off of the minimum number of ultrasound pulses and light flashes needed to determine 2D coordinates using a simple lateration approach, we show that nodes in the Pushpin network can compute their position with an average error of 5-cm and a error standard deviation of 3-cm. In this paper we present this localization system and characterize its accuracy in our hardware testbed.

I. Introduction

In contrast to typical communication networks, sensor networks are envisioned to derive their utility largely from an awareness of their physical environment at the most basic level. Such an awareness might simply involve each sensor node monitoring its own locality and routing the raw data back to a base station. However, the real potential for sensor networks lies in the ability to correlate sensed data among nodes over both space and time. Thus, determining and maintaining a shared global coordinate system and time base among all nodes in a sensor network (termed 'localization' and 'synchronization,' respectively) are fundamental tools for processing and acting upon sensor data. The work presented in this paper focuses on localization. We refer interested readers to [17] for an excellent introduction to all aspects of synchronization and to [1] for a tutorial essay on localization.

There are many classes of localization schemes, a reflection of the flexibility afforded by the ability of each node in a sensor network to sense (and in some cases act) locally as well as communicate with nearby nodes. For example, manually coding each sensor node with its location is clearly one solution, albeit an undesirable one due to scalability considerations. Robot-assisted sensor networks (see [9]) might be a viable alternative. Similarly, providing a location system via a fixed infrastructure, such as beacons that broadcast their position, has been explored in [19], among others. GPS is an extreme version of this method. Deploying nodes rich in computational, range-finding, and/or energy resources to help lighter weight nodes localize themselves has been extensively explored, as in [6]. Alternatively, each node can be individually equipped with range-finding hardware to detect relative positions of other nodes, as demonstrated in [27]. Most wireless communication systems can serve this purpose by considering received signal strength, although this is known to be quite inaccurate in many cases. Taking this idea to its logical conclusion, [3], [13] consider network hops as the base metric of physical distance, essentially equating physical topology with network topology. Section II covers some of these methods in more detail.

In this work, we present a sensor network localization algorithm and implementation that leverages the fact that an individual node's measurements of global stimuli (i.e. stimuli which can be detected by every node in the sensor network) are spatially and temporally correlated with measurements taken at other nodes. One example of this might be a lightning strike or a thunder-clap. These signals could be detected by nodes over an area of several square kilometers. For example, NASA Kennedy Space Flight Center has implemented a system to localize lightning strikes using a sensor array that is deployed across an area of 10-km [25]. Fireworks, munitions, and other explosive devices produce similar correlated global stimuli of different modalities (i.e. light and sound). Any pair of signals with coincident time origin and different propagation speeds can, in principle, provide useful localization constraints. For example, the distance to an impact could be estimated by the difference in arrival time of sound traveling through the ground and sound traveling through the air.

In our experimental setup, a handheld device called the "Pinger" is used to generate an external, global stimulus consisting of visible light and ultrasound to aid the sensor network in its localization task. Although artificial, the pinger device is meant to simulate a natural global event that might be detected by a sensor network deployed "in the field." No knowledge of the pinger position is known ahead of time, nor is the position of pinger constrained in any way except as to be somewhere (anywhere) above the sensor network. In particular, the pinger does not serve as a base station with a known position. Our algorithm uses in-network comparison of ultrasound time-of-flight measurements of signals emitted by this device to allow for lateration, a technique for finding a node's position given the distance of the node from several known points. In essence, the technique presented here is similar to a situation in which many hikers who can speak to each other over radio are lost in the wilderness and all witness several lightning strikes and their resulting thunder claps. By keeping track of the delays between seeing a flash of lightning and hearing the corresponding thunder clap and then comparing these times, the hikers can determine their relative positions.

In addition to the algorithm itself, we also present initial results from an implementation on the Pushpin Computing sensor network platform, a dense one-hundred-node network spread over an area of approximately two square meters. (At the time of this writing, only 55 of the 100 nodes are being used, but the remaining 45 will be available after minor maintenance.) Specifically, we cover details of the implementation, which uses the difference in time of arrival between a flash of light a ping of ultrasound as the global stimuli; a characterization and dicussion of localization accuracy (about 5-cm average error with a standard deviation of about 3-cm); possible applications and future work.

II. Related Work

Due to its fundamental role in correlating sensor readings, localization has become a subject of significant interest in sensor network research. The localization systems that have been proposed vary widely depending on the design constraints of the specific problem under consideration, but they can generally be classified according to the type of range measurement and the type localization algorithm they use.

A. Ranging and Localization Methods

Both acoustic ranging [6], [19], [27] and received signal strength [7] are distance metrics commonly used in sensor network localization. Savvides et al. [23] discuss the trade-offs between using acoustic ranging and received signal strength. They conclude acoustic ranging is generally more reliable than received signal strength because acoustic ranging depends only on the time difference between received signals, while received signal strength relies on signal amplitude, which is very sensitive to environmental factors. Also of note is the potential for ultra-wideband (UWB) radio for localization based on RF time of flight [14], although this field is still evolving. Similarly, there has been some recent research into using directional radio antennas for localization [11].

It has been proposed in [3], [7], [13] that localization can be accomplished without the aid of specialized sensing hardware. These authors propose that large, dense networks of a thousand or more nodes can reliably equate physical distance to logical distance over the network (hop count). Simulations show large sensor networks successfully performing localization with a network hop distance metric. As seen in Figure 3, given the scale lengths of the Pushpin network (mean distance between nodes, mean communication range, and square root of the area spanned by the entire network), network topology does not map well to physical topology in our case. However, as sensor networks grow in node number and density, this approach will become more accurate.

The distributed localization algorithm can generally be chosen independently of the method for finding inter-node distance. Most localization algorithms require *anchor nodes* in the network – nodes that have been preprogrammed with their location [6] or localized using specialized hardware such as GPS [23]. An unlocalized node measures its distance to the anchor nodes and uses this information in conjunction with the anchor coordinates to calculate its position in any of a variety of ways, such as determining whether it is bounded by a polygon defined by the anchor points [7], [10], or employing lateration [13], [23], a technique described later in this paper. A good comparison of several methods can be found in [10].

Some recent research in anchor-free localization involves distributed mesh relaxation [8], [18]. Unfortunately, this techniques is prone to error from false minima occurring during relaxation. Possible solutions to this problem are discussed in [18].

B. Hardware Testbeds

There are a handful of examples where localization algorithms based on acoustic ranging have been implemented on hardware testbeds. In the AHLoS system [23], heavy-weight nodes equipped with GPS help light-weight nodes to localize using lateration. In this case, both received signal strength and ultrasound time of flight were considered as potential ranging techniques.

Another technique very similar to the Pushpin localization scheme was demonstrated in the Calimari system [27], where sensor nodes equiped with speakers and microphones determined inter-node distance by measuring acoustic time of flight of audible chirps sent out by individual nodes. Because the Calimari system relies solely on signals which are generated by its sensor nodes, it does not require any external infrastructure or stimulus in order to perform localization. On the other hand, each node must be equipped with a full suite (receiver and transmitter) of range-finding hardware. Calimari's range measurements are also inherently limited by the low frequency (long wavelength) of the audio signal used to chirp between neighboring nodes. The reported error of the Calimari system is 30-cm.

Another novel localization approach involving acoustic time of flight is described in [6]. In this system, a Berkeley mote sends an acoustic, pseudo-random noise signal to several iPaq base station nodes which have been pre-programmed with knowledge of their own position. These heavy weight nodes collaboratively estimate the position of the mote by correlating the received pseudo-random noise with a known version of the same signal. Once obtained, the position estimate is transmitted back to the mote over the network. Localization error of 10-cm has been achieved using this technique.

In the Cricket system [19], [20], which is related to early work at AT&T [26] and Intersense [5], mobile devices called listeners monitor RF packets and ultrasound pulses from an infrastructure of beacons which are distributed throughout an indoor environment. Each beacon is pre-programmed with its location and transmits this information over RF while simultaneously sending out an ultrasound pulse. Aided by signals from several beacons, a listener can determine its location and orientation in three dimensions. The Cricket localization system achieves an average error of 1-cm.

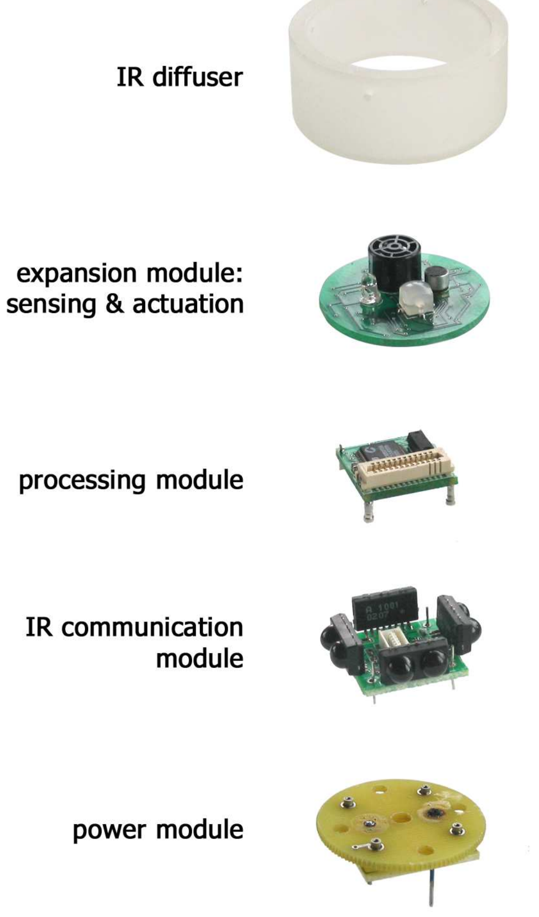

Fig. 1. An expanded view of the four modules and diffuser comprising a single Pushpin node.

III. Experimental testbed

Pushpin Computing began as an initiative to explore very dense distributed sensor networks. The Pushpin research is driven by the vision that sensor networks will some day be small and dense enough to coat surfaces in an "electronic skin" with sensing and processing capabilities comparable to our own [15]. That said, the Pushpin network may also be considered a "model" of larger sensor networks; it is a testbed on which sensor network algorithms can be rapidly prototyped on hardware with realistic communication and sensing characteristics.

A. Hardware

The Pushpin network consists of 100 nodes mounted on a powered substrate. For this paper, however, only

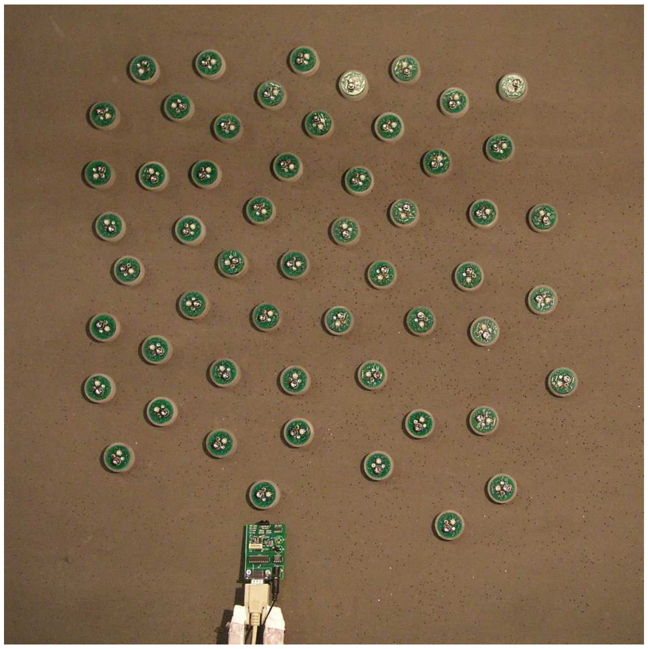

Fig. 2. The Pushpin sensor network with 55 nodes as configured when all the data presented here was taken. The visible portion of the surface they are mounted on is approximately 1-m on a side. Forty-five more nodes lie in wait. The large circuit board protruding from the bottom of the image is a gateway Pushpin that communicates between Pushpins outfitted with IR communication modules and a serial port on the desktop computer. This is the primary channel of collecting data off the network.

55 nodes were used. Although repositioning a node is a trivial matter no more difficult than repositioning a thumbtack on a soft corkboard, for the sake of consistency all data presented herein was taken with the Pushpins positioned as shown in Figure 2.

A Pushpin node has a modular, stacked architecture comprised of four circuit boards, one each devoted to power, communication, processing, and sensing, as illustrated in Figure 1. Each node receives power and ground through a pair of tensile pins located on the bottom of its power layer. These pins make contact with two parallel metal sheets embedded in a polyurethane foam substrate measuring 1.2-m by 1.2-m by 0.02-m.

The Pushpin processing board contains an 8051-core 8-bit, 22-MIPS microcontroller made by Silicon Labs (formerly Cygnal) [24]. This processor has 2.25-Kbytes of RAM and 32-Kbytes of non-volatile flash memory, as well as a host of on-board digital and analog peripherals used for sensing and actuation.

The Pushpin processor transmits and receives data at 96-kbps using infrared communication hardware on the communication module, where an IR transceiver points in each of the four cardinal directions, thereby enabling communication with neighboring nodes. In order to more evenly disperse the transmitted IR in all directions, a frosted polycarbonate ring is placed around each Pushpin. Figure 3 shows the IR communication radius of several nodes. The short range of the IR communications enables the Pushpins to become an exceptionally compact platform that realizes a wireless sensor network. RF communication would be difficult to constrain at this short range; any node broadcasting RF would likely be received by the entire network.

The sensing module used here includes a phototransistor, 40-kHz ultrasound transducer, and electret microphone. The phototransistor and the ultrasound transducer were specifically included for use in the localization process, whereas the microphone is meant for general-purpose audio sensing in future applications. In addition to amplified versions of all three sensor channels, enveloped versions of the ultrasound and audio microphones are available, as is the raw signal from the phototransistor. For actuation, the sensing module includes an RGB LED that is used to indicate the status of the node or as a display element when a node participates as a "smart pixel" in a distributed display task.

B. Software

Programming an entire network of sensor nodes can be an onerous task. We have addressed this bottleneck by creating a bootloader that always resides in the flash memory of every Pushpin. The bootloader performs version checking, error detection, and error correction on software updates received through the Pushpin IR communication hardware, and then writes the updates to bootable non-volatile program memory. We use a large 108-LED "IR spotlight" to send updates to every Pushpin simultaneously from a desktop computer where new code is written and compiled. Using this interface, we can reprogram all nodes with a new operating system and program software in less than a minute, thus significantly reducing the time required for a debug cycle.

Once a software update is complete, the bootloader hands control over to Bertha, the Pushpin operating system compiled with application code. Bertha manages the hardware on the Pushpin, making basic services available to application code through a set of simple APIs. Some of the services provided by Bertha include a real time clock, random number generation, sensor/actuator access, and access to interrupt routines. The localization software, for example, uses the interrupt routines and the real time clock to measure the difference in arrival times between a light flash and ultrasound ping.

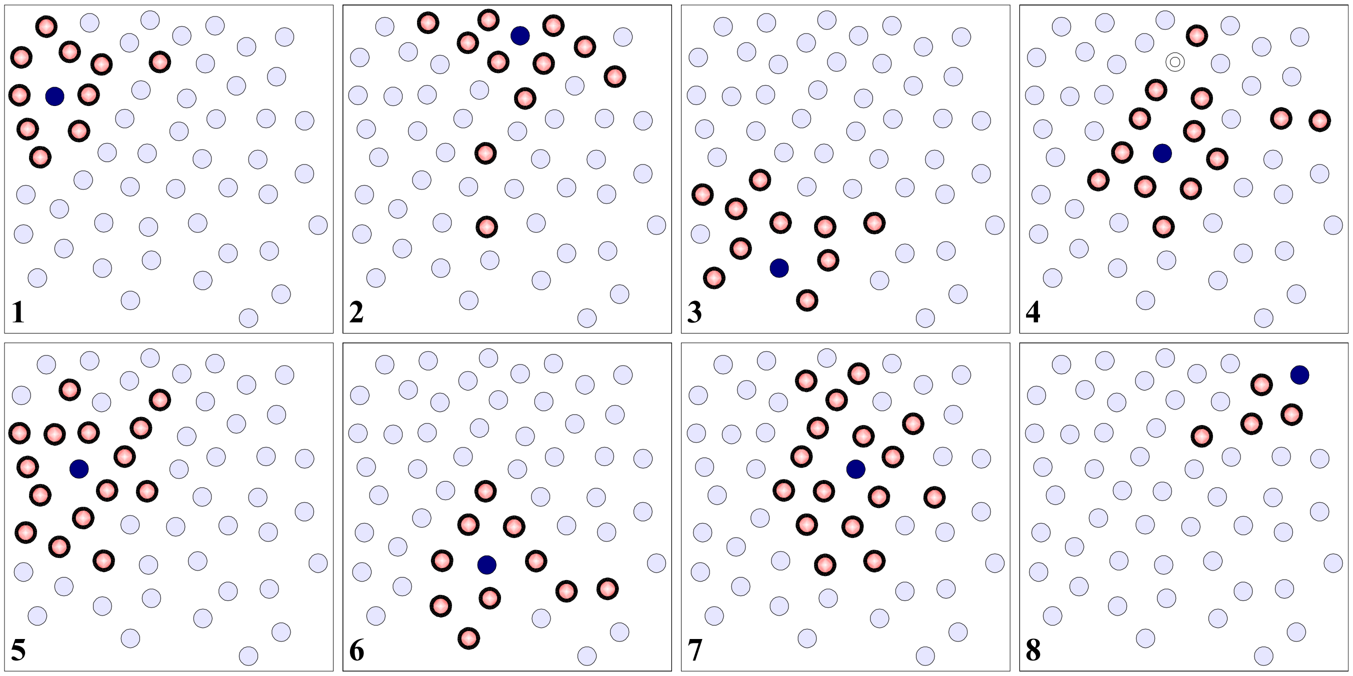

Fig. 3. The neighborhood of nodes with which a Pushpin can communicate varies considerably from Pushpin to Pushpin. The dark, solid circle in each of the above frames indicates a Pushpin that is constantly transmitting packets via IR. The circle with bold outlines indicate which of the surrounding nodes reliably receives the packets directly from the originating node. The single outlined node in frame 4 indicates a node that sporadically enjoyed good reception. Aside from this one exception, all other nodes in the neighborhood received virtually 100% of the transmitted packets. Note that there was no other network traffic and the neighborhoods shown only depict one-way communication. In particular, note the transmitter in frame six belongs to the neighborhood of the transmitter in frame 2, but not the other way around; clearly, the neighborhoods shrink if two-way communication is required for membership. The data presented above were taken from the Pushpin testbed as configured in Figure 2.

Perhaps the most important service provided by Bertha is a comprehensive communication library which includes an implementation of directed diffusion [4] for basic network routing. For the sake of simplicity, each node in this experiment was pre-programmed with a globally unique network address to identify it for the purpose of communication and data analysis.

C. Pinger

We have created a handheld device, as shown in Figure 4, called the "pinger" composed of a camera flash and an array of 40-kHz ultrasound transmitters. A button mounted on the top of the pinger triggers a simultaneous flash of light and burst of sonar from the array. Given knowledge of the propagation speeds of these two signals, a sensor node can determine its distance to the point from which they were emitted (a 'point of correspondence') by measuring the difference of their arrival times. Assuming all nodes in the network measure the difference of arrival between the flash and ultrasound pulse, a simple comparison of these measurements indicate which nodes could possibly be nearby and which nodes are further off. Such correspondence points serve to constrain the localization problem.

It should be noted that the approximately 1-centimeter wavelength of the 40-kHz sultrasound places a theoretical upper limit on the accuracy of the time of flight measurement made by this system using a simple rising edge discriminator. Furthermore, the array of nine ultrasound transmitters is roughly of the same scale as the inter-node distance between Pushpins, so it certainly isn't an ideal point source and contributes to overall error.

IV. Localization System

This section gives an overview of the localization system developed and implemented on the Pushpins. As previously mentioned, this algorithm combines aspects of ultrasound time of flight and lateration into an approach that does not require any prior knowledge of the location of any of the sensing nodes or the pinger emitting the pulse of ultrasound and flash of light. To the best of our knowledge, we believe our method is unique in this sense.

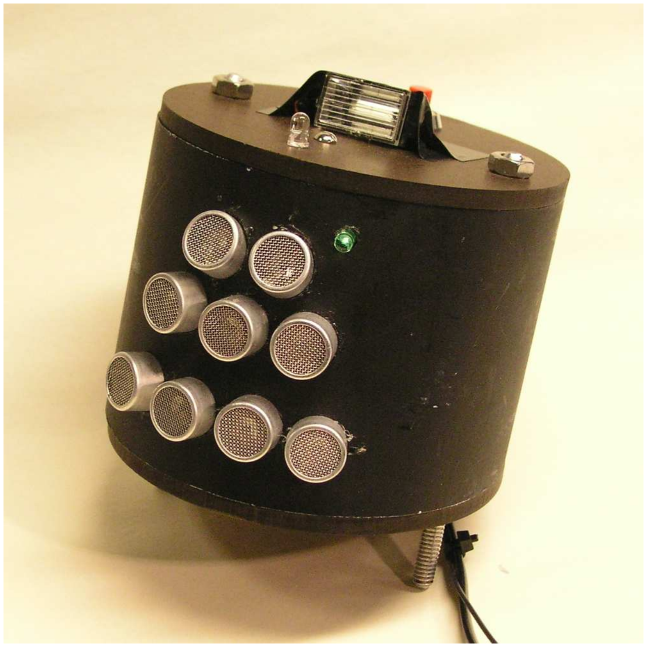

Fig. 4. The “pinger” delivers a simultaneous flash of light and burst of 40-kHz ultrasound. Each Pushpin can measure the difference in time of arrival of these two signals and use this to calculate the distance between itself and the Pinger.

A. *Establishing Anchor Nodes*

The first flash of light signals the begining of the localization process. The flash of light is detected roughly simultaneously by all nodes, which each start a hardware timer that runs either until they detect an ultrasound pulse or the timer overflows. If the timer overflows, the node assumes it has missed the ultrasound pulse and it stops participating in the localization algorithm altogether. If the sonar pulse is received, the node computes the distance from itself to the pinger, which is simply proportional to the time of flight of the ultrasound pulse and is easily calculated given the speed of sound in air.

After a short delay ensuring every node has had a chance to hear the sonar ping, a leader election algorithm begins to determine which node was closest to the pinger. The leader election algorithm consists primarily of pairwise comparisons of each node’s measured time of flight for the ultrasound pulse in question. These pairwise comparisons continue until the network agrees which node is closest to the pinger event. This node is elected as an anchor point. A new anchor point is elected in this manner each time the pinger is triggered. In parallel with this process, newly elected anchor points compute their position. Figure 5 illustrates the process of computing anchor node coordintates.

The anchor point elected first is arbitrarily chosen to be the origin of the new coordinate system, and is given the coordinates $(0, 0)$ (frame **A**). The second anchor point is assigned coordinates on the new Y-axis. To determine this anchor’s y-coordinate, anchor points 1 and 2 share information over the network about their respective distances to the first and and second pinger location. Using these distances, which show up as the boldface and hairline dotted lines in frame **B**, the baseline distance - which is the y-coordinate of the second anchor - can be computed using basic trigonometry. The coordinates of the final anchor point are placed in the $+X$-halfplane. These coordinates are computed using the triangle relations illustrated in frame **C** of the figure.

An important assumption is made about the geometry of this trigonometric calculation: the pinger assumed to always be triggered directly above some node in the network. The node immediately below the pinger is nearest to the pinger, thus it will become an anchor node. With this assumption, any node can compute its baseline distance to an anchor point by solving a simple right triangle relation. Without this assumption, additional constraints would be necessary to determine the baseline distance to an anchor. The right angle constraint was chosen because it is easily justified for a dense sensor network: If the pinger is held at a random location above the Pushpin network, the inter-node spacing is small enough that the pinger is not likely to be more than a few centimeters away from being directly above some node. Also, considering that the width of the ultrasound transmitter array on the pinger is roughly as wide as the average distance between nodes, this is not a very severe constraint. We discuss possible departures from this assumption in Section VII.

B. *Lateration*

Once three anchor points have been established, there is sufficient information to find a complete localization solution by solving a set of linear equations. This technique, called lateration, is described in [10], among others, and is summarized below. In general, lateration can be used to find the coordinates of a node given any number of anchor points. However, for the sake of creating a simple implementation that runs efficiently on the 8-bit Pushpin microcontroller, we have restricted ourselves to the case where only three anchor points (the minimal number required for 2D coordinates) are considered. We discuss the possibility of using more than three anchor points in section VII.

To find the coordinates (x, y) of node **d** using basic lateration, we proceed as follows. Given the coordinates

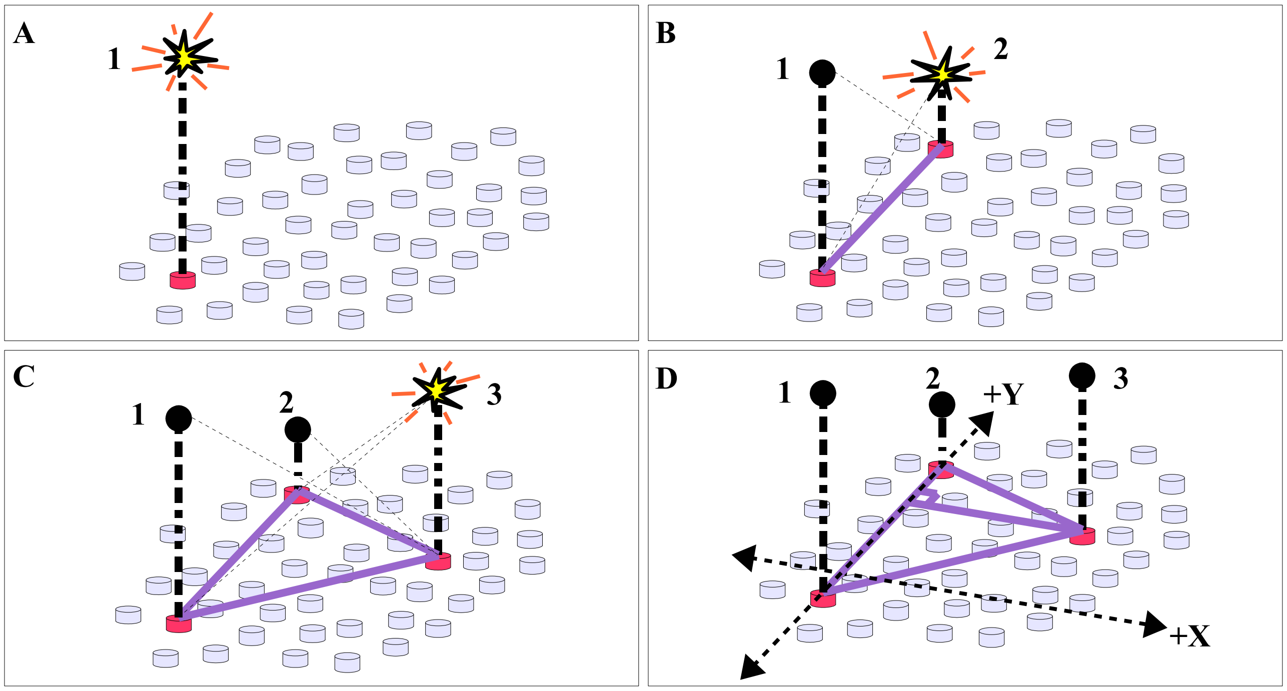

Fig. 5. A) Each firing of the ultrasound/light pinger allows the network to select a single anchor point, namely the node closest to the location of the pinger. B) Once two anchor points are selected, they can determine the distance separating them by simple geometry. C) Three non-colinear anchor points guarantee that a planar coordinate system can be built up. D) The orientation and handedness of the coordinate system is arbitrarily, but consistently, chosen.

of three anchor points {**a**,**b**,**c**} and the distances from each anchor point to node **d**, we can assert that the coordinates of **d** must satisfy the following Euclidean distance metrics.

$$\begin{aligned}(x_a - x)^2 + (y_a - y)^2 &= d_a^2\\(x_b - x)^2 + (y_b - y)^2 &= d_b^2\\(x_c - x)^2 + (y_c - y)^2 &= d_c^2\end{aligned}$$

where (x_i, y_i) are the coordinates of anchor i and d_i is the distance from node **d** to anchor **i**, for all $i \in \{a, b, c\}$. All quantities except x and y are known. We can linearize this system by subtracting the third equation from the first two and collecting terms:

$$\begin{aligned}2(x_a - x_c)x &+ 2(y_a - y_c)y\\&= x_a^2 - x_c^2 + y_a^2 - y_c^2 + d_c^2 - d_a^2\\2(x_b - x_c)x &+ 2(y_b - y_c)y\\&= x_b^2 - x_c^2 + y_b^2 - y_c^2 + d_c^2 - d_b^2\end{aligned}$$

We now have a linear system of the form $A\begin{pmatrix}x\\y\end{pmatrix} = b$. If A is non-singular, the solution to this equation can be trivially computed by inverting a two-by-two matrix.

V. Results

Unless otherwise noted, the results presented in this section were derived with the Pushpin sensor network configured as described earlier. Specifically, the configuration included 55 Pushpins arranged as depicted in Figure 2 and five localization trials each consisting of three ultrasound/light pings that were randomly located (but non-colinear) within the 1 cubic meter space above the Pushpin network.

A. Ultrasound Time of Flight

In order to provide a useful baseline in our error analysis of the Pushpin localization scheme, we have characterized the accuracy of the time of flight measurements taken at a single node. The accuracy of the lateration algorithm depends heavily on accurate range measurements. Hence, any improvements to the range-finding measurement due to calibrating the ultrasound sensor on the Pushpins would increase localization accuracy as well.

The error plot in Figure 6 shows the results of an experiment in which the pinger was held directly in front of a stationary Pushpin at distances between 0-m and 3.5-m with 0.1-m increments. The plot shows what appears

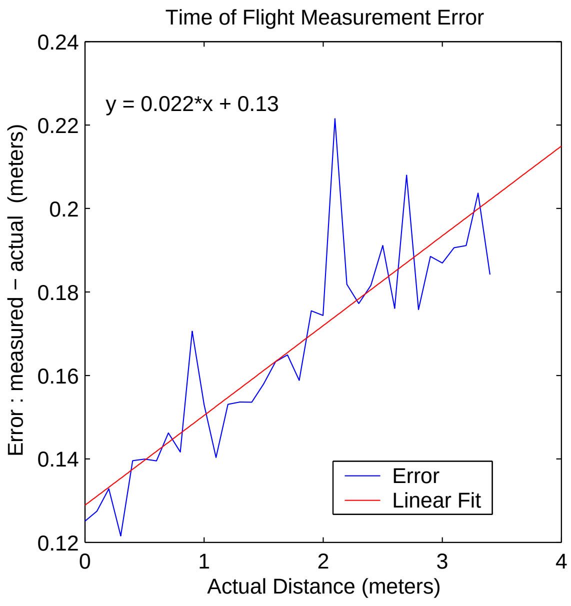

Fig. 6. Characterization of the error in the time of flight measurement taken at a single Pushpin. Four measurements were taken at each of the 10-cm intervals.

to be a baseline error of 0.13-m that increases linearly with increasing actual distance. We postulate that the error in the ultrasound time of flight measurement is due to some combination of dispersion, speckle, changing air currents, interference among ultrasound transmitters, ring-up time for the ultrasound receiver, time walk from decreasing signals with a constant threshold, and part variability. This error was compensated for using a linear calibration based on these measurements. Several Pushpins were characterized in this manner and their calibration coefficients averaged to arrive at the calibration coefficients ultimately applied to every Pushpin node. Once the ultrasound receiver was calibrated, the average error of the range measurements was reduced to 0.03-m.

B. *Localization Error Analysis*

In this section, we assess the performance of the Pushpin localization system. The objective is to determine the average error between *estimated* coordinates $(\hat{x}, \hat{y})$ produced by the algorithm in section IV and a set of coordinates (x, y) measured manually with the aid of simple image manipulation software, which we will refer to as *measured* coordinates. The following metric was used to compute the average absolute coordinate error for N sensor nodes:

$$e_{abs} = \frac{\sum_{i=1}^{N} \sqrt{(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2}}{N} \quad (1)$$

In order to find the measured coordinates, the array of Pushpins was photographed using a digital camera. A 0.1-m by 0.1-m square was included in the image to provide a reference for image scale and to verify that no measurement error would be introduced by image warping caused by the camera lense. The coordinates were extracted manually on a computer from this digital image.

A Pushpin that was wired to the serial port of a PC was used to query the network for the estimated coordinates of each node. The entire data collection process took less than 30 seconds for 55 Pushpins. Once the data was collected, outliers were manually rejected. An outlier was considered to be any coordinate that was more than 0.5-m away from the average of its neighbors' localized coordinates. This criteria for rejecting outliers was chosen because it could be easily implemented as an ad hoc post-processing step of the localization algorithm by using simple neighber-to-neighbor network transactions. We found that there were typically 3-4 outliers per localization trial.

As given, the algorithm produces a set of coordinates in an arbitrary reference frame that depends on the position of the three anchor points which in turn, are determined by the location of the pinger at each light/ultrasound event. To account for this, the measured and estimated coordinate systems must somehow be aligned. We considered two different methods of aligning the coordinate systems: (1) A tranform comprised of only a translation, rotation, and reflection; and (2) a more general homogeneous affine tranform that could also include scaling and shearing. When equation 1 is applied to estimated coordinates tranformed as in (1), we refer to this as *unscaled error*, since the distance metric in these coordinates has not been scaled by the tranformation, and still reflects absolute distances in the physical world. The error in coordinates tranformed as in (2) is refered to as *scaled error*.

Homogeneous affine transformations are a superset of translations and rotations, so we expect the scaled error to be smaller than the unscaled error since the scaled error includes extra degrees of freedom that allow for a better fit to the measured coordinates. The degree to which these two approaches produce similar results tells us how immune the estimated coordinate system is to shearing and scaling. Immunity to scaling and shearing is desirable in many applications where the distances measured in the estimated coordinate system must correspond to real, absolute distance in the physical world. For applications where only relative distance

TABLE I

LOCALIZATION ERROR (SCALED)

Trial#	mean error (m)	error standard deviation (m)
1	0.0451	0.0291
2	0.0583	0.0402
3	0.0478	0.0303
4	0.0450	0.0268
5	0.0501	0.0260

measurements are necessary, minor scaling and shearing effects can be tolerated.

1) Scaled Error: Finding the scaled error is a straightforward matter of finding the linear least squares approximation. More explicitly, an affine transformation A must be found that, given homogeneous estimated coordinates $(\hat{x}, \hat{y}, 1)$ in the estimated coordinate system, yields a pair of coordinates $(\hat{x}', \hat{y}', 1)$ in the actual coordinate system. The goal is to minimize the error between the transformed, estimated coordinates $(\hat{x}', \hat{y}', 1)$ and the measured coordinates $(x, y, 1)$ for every node in the network. This can be accomplished by solving the following linear system:

$$A \begin{pmatrix} \hat{x}_1 & \hat{x}_2 & & \hat{x}_N \\ \hat{y}_1 & \hat{y}_2 & \dots & \hat{y}_N \\ 1 & 1 & & 1 \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & & x_N \\ y_1 & y_2 & \dots & y_N \\ 1 & 1 & & 1 \end{pmatrix}$$

$$AX = B$$

where X is an array of untransformed, estimated coordinates and B is an array measured coordinates. This over-determined linear system has a linear least squares solution of the form:

$$A = BX^T(XX^T)^{-1} \tag{2}$$

Figure 7 depicts the estimated coordinates after being transformed as in equation (2). Once transform (2) has been applied, the error metric (1) can determine the average absolute error between the measured and estimated coordinates.

We applied this technique in five seperate localization trials. For each localization trial, the mean absolute error and standard deviation were found. These calculations are tabulated in Table I.

2) Unscaled Error: For the sake of simplicity in finding the unscaled error, we first aligned the origins of the estimated coordinate system with the origin of the actual coordinate system (to find the proper translation), and then reflected and rotated the localized coordinate system about the origin so as to minimize error described in equation (1). Figure 8 depicts the data after being rotated and translated as just described. The resulting error and error standard deviation using this approach is tabulated in Table II.

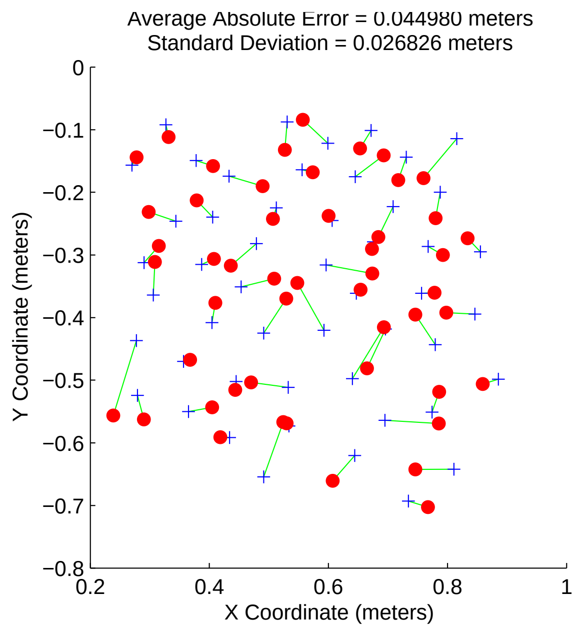

Fig. 7. Scaled Error: Estimated Pushpin coordinates after being fit to measured coordinates using a least mean squares approximation. The crosses denote measured coordinates and the circles denote coordinates estimated using lateration. The line connecting each cross to a circle indicates which estimated coordinate corresponds to which measured coordinate.

TABLE II

LOCALIZATION ERROR (UNSCALED).

Trial#	mean absolute error (m)	absolute error standard deviation (m)
1	0.0609	0.0277
2	0.0558	0.0387
3	0.0661	0.0493
4	0.0627	0.0393
5	0.0593	0.0399

C. Diagonal Wipe Demonstration

As a first test of our localization algorithm, we devised a simple visual demonstration meant to show that each node knows its coordinates. Once all nodes localize themselves, by referencing their common timebase as synchronized by the flashes, they individually simulate a visual "wipe" effect by animating a line that starts at x=-1 meter and moves toward x=+1 meters parallel to the Y-axis. Once the line passes the position a node has localized to, the node changes color from green to red.

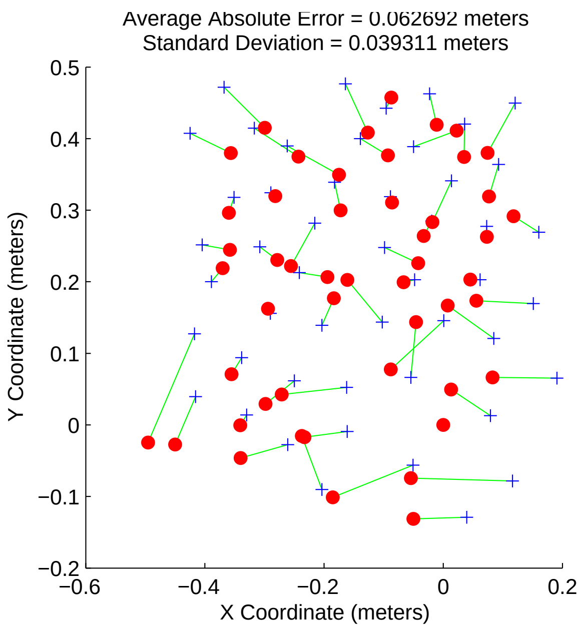

Fig. 8. Unscaled Error: Estimated Pushpin coordinates after being fit to measured coordinates using only a translation, rotation, and reflection. The crosses denote measured coordinates and the circles denote coordinates estimated using lateration. The line connecting each cross to a circle indicates which estimated coordinate corresponds to which measured coordinate.

The results of this demonstration visually verify that the nodes have been spatially localized. In essence, the localized sensor network is being used as a simple display. Figure 9 shows a time lapse of the wipe demonstration.

VI. Discussion

The results in Section V have immediate implications. The diagonal wipe demonstration in Section V-C indicates that the basic localization algorithm is functioning well. This is verified by the relatively low localization errors and standard deviations found in Section V-B. Furthermore, the error results using the unscaled method of calculating error are comparable to the error results using the scaled method, indicating that the localized coordinate system is nearly free of distortion due to scaling and shearing. As expected, the the error resulting from the scaled method is slightly lower than the error resulting from the unscaled method, with the exception of trial 2 which is close enough to suggest that the discrepancy may not be statistically significant (we are looking into this). Also encouraging is the consistency between trials of mean error and error standard deviation.

Although these results are promising, there is plenty of room for improvement and additional verification. In particular, some recent papers [16], [21] compute the Cramer-Rao bound (CRB), a statistical technique that provides a lower bound on the variance of any unbiased estimator, for the lateration problem. Their results suggest the error variance of the location estimate can be very high near the edges of the convex hull created by the anchor nodes. These papers also find that the CRB is extremely sensitive to anchor number, location, and geometry. Generally speaking, a higher number of anchors drastically reduces the CRB, hence incorporating additional pinger events may considerably improve our results.

Furthermore, it would be useful to establish the CRB for the specific estimation method described in this paper. In particular, the approximation that the pinger is held over an anchor node should be carefully checked via the CRB to ascertain whether anchor location uncertainty introduces significant error in the localization estimate. In principal, it should be possible to reduce location errors and variances to approximately 1-cm, which is the theoretical bound on the accuracy of ultrasonic measurement given a unipolar threshold detection on a fixed-frequency point source.

In addition, this algorithm should be able to adapt to measurement outliers and faulty hardware. A set of simple heuristics for detecting failures and correcting them would adequate for eliminating the most common errors. The approach described in section V-B for removing outliers by direct neighbor-to-neighbor comparison of coordinates is an example of one such a heuristic.

VII. Future Work

The approach and results in this paper have demonstrated that global stimuli can be used to generate anchor points and localize nodes via lateration in a hardware testbed. Improved position accuracy, robustness to node failure and measurement error, and a thorough error characterization using the CRB are natural avenues of improvement for the current approach. However, we still seek to create a more general localization framework that frees us of some of the restrictions and assumptions made in our work thus far.

First, the algorithm 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 *single* global signal. In this

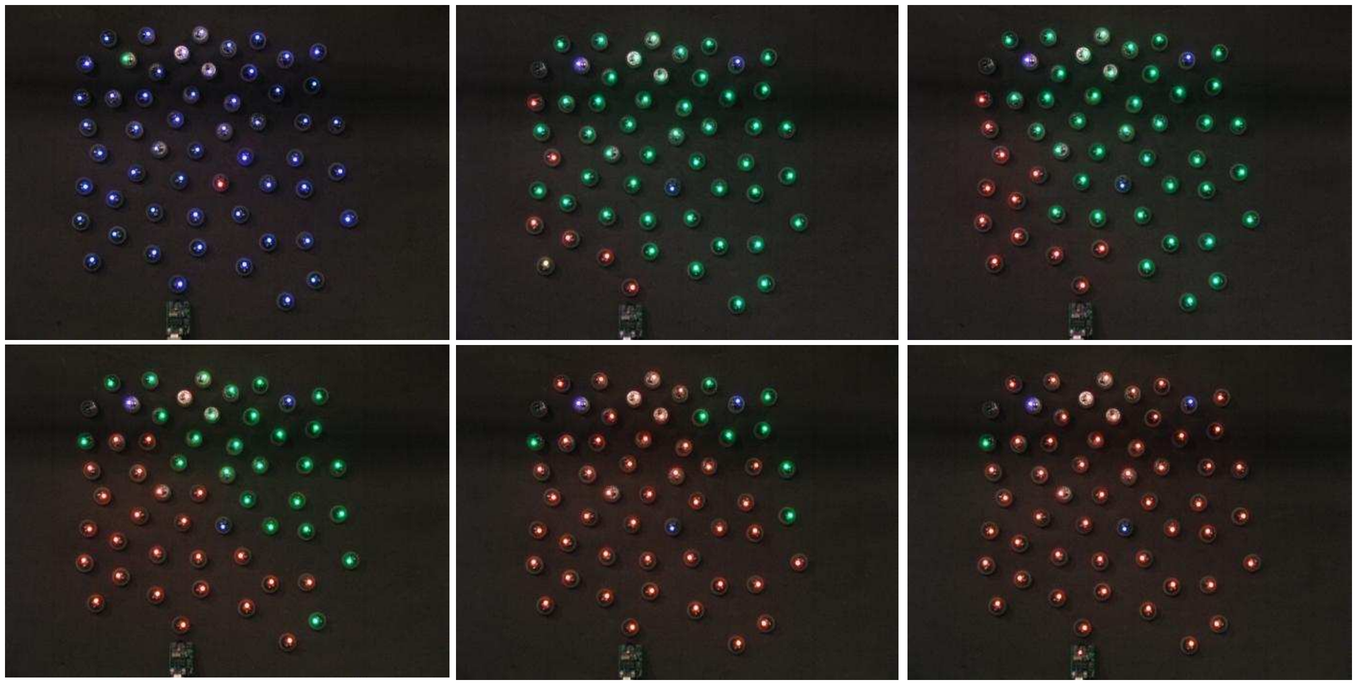

Fig. 9. Starting in the upper left, going from left to right: 1) Two pings have been fired and their respective anchors elected. The first anchor (origin) is the red node in the middle, the second anchor is the green node in the upper left. The line connecting them defines the Y-axis in the localized coordinate system. 2) A third ping was fired and the third anchor elected in the upper right. The third anchor defines the handedness of the coordinate system. In this case, the +X direction is where it normally is for a right-handed coordinate system. All three anchors are now blue. 3) A simulated diagonal wipe can be seen entering from the -X direction propagating in the +X direction. The line that delineates the boundary of the wipe is parallel to the Y-axis. 4) The diagonal wipe effect is *not* a result of communication among nodes, but rather the result of all nodes sharing a global time base and coordinate system and using these to individually simulate the same moving wipe boundary. 5) The wipe maintains its structure as it passes over the network. A green outlier can be seen in the upper left. 6) The wipe has passed, but will wrap around toroidally.

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.

We would also like to move away from the restriction that the pinger must be triggered above some node in the network. This would allow nodes to localize off of a broader range of events, and more importantly, would eliminate any error introduced from uncertainty in the pinger location. Unfortunately, by removing this assumption, the localization problem is no longer linear, hence it must be solved using an iterative, non-linear optimization approach. Several such 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 mesh relaxation [8], semi-definite programming [2], non-linear least squares [12], and distributed Kalman filters [22]. We have preliminary simulation results suggesting a mesh relaxation approach with an appropriately chosen cost function can reliably solve the generalized Pushpin localization problem, but this algorithm has yet to be implemented on our hardware testbed.

Finally, as mentioned earlier, localization is a fundamental tool for building applications with sensor networks. Now that we are well on our way to having the tool, the challenge now is to exploit it. An obvious choice as to where to begin in this regard is to create a more compelling and visually complex display application than the diagonal wipe demonstration presented in Section V-C. A first step in this direction might include using the pushpin array to do both simple shape recognition and display. Further on is the possibility of creating the sensor network version of PostScript. If we were also to pursue synchronization as one of our goals, it may be possible to employ the microphone-equipped nodes as an acoustic phased array. Such a system would demonstrate the ability to first localize itself, and then to further localize events in its environment, in this case audio sources.

VIII. Conclusion

We have developed and implemented a sensor network localization algorithm that combines aspects of ultrasound time-of-flight range-finding with in-network lateration to arrive at a localization method free of the need for prior knowledge of absolute position. This method relies on correlated light flashes and ultrasound pulses to elect anchor points used in a standard linear lateration algorithm. We explored two methods of calculating localization error, both of which resulted in a mean absolute error of approximately 5-cm and standard deviation of approximately 3-cm. In addition to determining localization error, a simple visual demonstration making use of the localization data was implemented.

IX. 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 National Science Foundation grant #ECS-0225492.

References

[1] J. Bachrach, "Position determination in sensor networks," to appear in Stojmenovic I (Ed), Mobile ad hoc networking, John Wiley and Sons, 2004.

[2] P. Biswas and Y. Ye, "Semidefinite programming for ad hoc wireless sensor network localization," in *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, 2004, pp. 46 – 54.

[3] W. Butera, "Programming a Paintable Computer," Ph.D. dissertation, Massachusetts Institute of Technology, 2002.

[4] R. G. Chalermek Intanagonwiwat and D. Estrin, "Making Sensor Networks Practical with Robots," in *Sixth Annual International Conference on Mobile Computing and Networking*, August 2000.

[5] E. Foxlin, M. Harrington, and G. Pfeifer, "Constellation: a wide-range wireless motion-tracking system for augmented reality and virtual set applications," in *Proceedings of the 25th annual conference on computer graphics and interactive techniques*, July 1998, pp. 371–378.

[6] L. Girod, V. Bychkobskiy, J. Elson, and D. Estrin, "Locating tiny sensors in time and space: A case study," in *Proceedings of the International Conference of Computer Design (ICCD) 2002*, 2002.

[7] T. He, C. Huang, B. Blum, J. Stankovic, and T. Abdelzaher, "Range-Free Localization Schemes in Large Scale Sensor Networks," in *Proceedings of the 9th annual international conference on Mobile computing and networking*, 2003, pp. 81–95.

[8] 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.

[9] A. LaMarca, W. Brunnette, D. Koizumi, M. Lease, S. B. Sigurdsson, K. Sikorski, D. Fox, and G. Borriello, "Making Sensor Networks Practical with Robots," in *Pervasive Computing First International Conference*, August 2002, pp. 152–166.

[10] K. Langendoen and N. Reijers, "Distributed localization in wireless sensor networks: a quantitative comparison," *The International Journal of Computer and Telecommunication Networking*, vol. 43, no. 4, pp. 499 – 518, November 2003.

[11] N. Malhotra, M. D. Krasniewski, S. Bagchi, and W. Chappell, "Location Estimation in Ad-Hoc Networks with Directional Antenna," 2004, submitted to ACM Sensys 2004.

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

[13] R. Nagpal, H. Shrobe, and J. Bachrach, "Organizing a Global Coordinate System from Local Information on an Ad Hoc Sensor Network," April 2002, 2nd International Workshop on Information Processing in Sensor Networks (IPSN '03).

[14] K. Pahlavan, L. Xinrong, and J. Makela, "Indoor geolocation science and technology," *IEEE Communications Magazine*, vol. 40, no. 2, pp. 112–118, February 2002.

[15] J. Paradiso, J. Lifton, and M. Broxton, "Sensate Media – multimodal electronic skins as dense sensor networks," *BT Technology Journal*, vol. 22, no. 4, October 2004.

[16] N. Patwari and A. O. H. III, "Using Proximity and Quantized RSS for Sensor Localization in Wireless Networks," in *Proceedings of the 2nd ACM international conference on Wireless sensor networks and applications*, 2003, pp. 20 – 29.

[17] A. Pikovsky, M. Rosenblum, and J. Kurths, *Synchronization: A universal concept in nonlinear sciences*. Cambridge University Press, 2001.

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

[19] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," in *Mobile Computing and Networking*, 2000, pp. 32–43.

[20] N. B. Priyantha, A. K. L. Miu, H. Balakrishnan, and S. Teller, "The Cricket Compass for Context-Aware Mobile Applications," in *Proceedings of the 6th ACM MOBICOM Conference*, July 2001.

[21] A. Savvides, W. Garber, S. Adlakha, R. Moses, and M. B. Srivastava, "On the Error Characteristics of Multihop Node Localization in Ad-Hoc Sensor Networks," in *Proceedings of the 2nd International Workshop on Information Processing in Sensor Networks (IPSN'03)*, 2003.

[22] A. Savvides, H. Park, and M. Srivastava, "The Bits and Flops of the N-Hop Multilateration Primitive for Node Localization Problems," in *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, 2002, pp. 112–121.

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

[24] Silicon Laboratories, " Silicon Laboratories Homepage ," http://www.silabs.com, 2003.

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

[26] A. Ward, A. Jones, and A. Hopper, "A New Location Technique for the Active Office," *IEEE Personal Communications*, vol. 4, no. 5, pp. 42–47, October 1997.

[27] K. Whitehouse, "The Design of Calamari: an Ad-hoc Localization System for Sensor Networks," Master's thesis, University of California at Berkeley, 2002.