

MobiLoc: Mobility Enhanced Localization

Prabal Dutta
Department of Electrical Engineering
The Ohio State University
2015 Neil Ave
Columbus OH 43210
dutta.4@osu.edu

Sarah Bergbreiter
EECS Department
University of California, Berkeley
497 Cory Hall
Berkeley CA 94720
sbergbre@eecs.berkeley.edu

General Terms

Algorithms, Performance, Design, Implementation

Keywords

Sensor Networks, Localization, Positioning, Ranging, AOA, TDOA, TOF

ABSTRACT

We study the relationship between mobility and localization in the context of wireless sensor networks. We observe that mobility can aid sensor nodes in determining the distance to neighboring nodes without requiring explicit coordination between the mobile object and the sensor network. Distance estimation between nodes is important for many existing localization algorithms and, in this paper, we describe a pair of algorithms with low time and message complexity for estimating these distances. This approach is distinct from prior approaches to estimating inter-node distances in that the nodes do not estimate this distance by directly measuring received signal strength or time of flight from another node. Similarly, we do not require that nodes can range, at three or more distinct times, a mobile beacon that knows and communicates its own position. Instead, our approach is based entirely on the nodes estimating the range to a mobile object in a coordinated fashion and then combining these range estimates to yield the distance estimate between all pairs of observing nodes. We also provide a mechanism for determining the quality of the estimated distances, allowing nodes to reject measurements that would likely result in poor distance estimates. The proposed algorithm is distributed and scalable since all computations and communications are purely local. We validate our approach through both simulated and empirical results.

1. INTRODUCTION

This section provides a brief background on the problem of localization in wireless sensor networks and describes the

problem's frequent dependence on distance estimation between neighboring nodes. The motivation underlying this approach to distance estimation is explored. Finally, the organization of the remainder of the paper is described.

1.1 Background

Wireless sensor networks hold great promise as an enabling technology for a variety of monitoring applications. Many such applications require that the sensor network report the time and location of an event. Time synchronization addresses the problem of establishing a coordinated notion of time. Localization is concerned with establishing a spatial coordinate system and determining the positions of the sensor nodes and targets within this coordinate system.

In some applications, the sensor nodes may number in the thousands and be deployed in a manner that results in random positions and orientations. Additionally, the nodes may move, be moved, or become inaccessible after deployment. In these cases, it may not be practical to individually configure each node with its position. A canonical application exemplifying these possibilities is the detection and tracking of targets traveling through an area in which sensor nodes have been deployed randomly[11]. In such applications, the past, present, or future locations of the target must be estimated with respect to a local or global coordinate system. In order to specify the target location with respect to the coordinate system, the sensor nodes must be aware of their own positions within this coordinate system, demonstrating the importance of localization in detection and tracking applications of sensor networks.

Localization is important in many other sensor network applications. There are cases in which node locations, however crude, are needed soon after deployment. For example, geographic routing requires nodes to route messages along gradients that are computed from node locations. For such cases, a simple localization scheme, perhaps based on hop-count [bib:savarese02robustpositioning](#), can be used to bootstrap the node positions. Despite its importance, however, localization remains a difficult problem to solve in the general case for a number of reasons including its sensitivity to range estimation errors, the uncertainty of real deployment environments, and the need for specialized hardware.

Many localization algorithms rely on the distances between nodes. This distance estimation is usually implemented with signal strength decay, time-of-flight (TOF), time-of-arrival

(TOA), or time-difference-of-arrival (TDOA) for inter-node range estimation. Signal strength is known to vary with hardware, the environment, and complex fading effects. As a result, received signal strength results in widely variable results and is often not used. Instead, ranging techniques based on the propagation speed of signals has been used in the recent literature to estimate distances. However, such TOF, TOA, and TDOA techniques require specialized hardware to support the localization system. This additional hardware adds costs and may not be useful beyond its limited localization function.

1.2 Motivation and Approach

Our motivation for this work comes from the observation that in many applications, the nodes may be equipped with sensors that can detect the distance to a moving object, or target, but may not require specialized hardware solely for localization if it were possible to perform this in some other way. A secondary motivation for this work is based on the observation that both RSSI and TOA are random functions not of time but of place [7]. TDOA is the difference between two TOA values for two signals with different propagation speeds. Since individual TOA values are Gaussian, their difference is Gaussian as well. Therefore, by spatially and temporally averaging these signals, better noise filtering is possible than with just temporal averaging as would be the case for static nodes ranging each other. We also observe that in some cases, a node’s location may not be needed at all *until* a target is actually present. In such cases, a lazy localization strategy, in which nodes only localize when the sensor network is being used to track an object, may suffice. This approach has the benefit of only consuming energy when the nodes have a reason to localize.

The goal of this work is to realize the following scenario. A set of sensor networks nodes are deployed into a region at unknown locations. A friendly (or unfriendly) target travels through the sensor network in a structured (or random) walk. The nodes determine the distance between themselves and their neighboring nodes using the techniques described in this paper. Once the node-to-node distances have been estimated, any one of the available range-based localization algorithms may be used to determine the position of the nodes. Once localized in this manner, the nodes can multilaterate the location of targets and report this information as needed.

1.3 Paper Organization

Section 2 reviews related work on localization in sensor networks. Section 3 identifies a variety of techniques for ranging, or estimating the distance between a target and a sensor node. Section 4 describes several algorithms to combine the node-to-target range estimations into node-to-node distance separations. Section 5 describes our experimental methodology, hardware, and software used to test our algorithms. Section 6 provides the results of our work. Section 7 identifies areas of improvement and discusses our future plans. Section 8 summarizes our results and provides our concluding thoughts.

2. RELATED WORK

In [6], the authors identify three requirements necessary for distributed algorithms that can be employed on large-scale

ad-hoc sensor networks consisting of more than 100 nodes: self-organizing (i.e. does not depend on global infrastructure), robust (i.e. tolerant to node failures and range errors), and energy efficient (i.e. require little computation and, especially, communication). The authors also identify three key context parameters: connectivity, anchor fraction, and range errors. We note that connectivity and anchor fraction are a common denominator in the sense that for any given deployment realization, the two parameters are either fixed or outside of our control. Consequently, we focus on range errors, the only remaining parameter.

In [10], the authors explore the use of a GPS-equipped mobile beacon to provide three distinct range estimates from three non-collinear points to each node. Bayesian inference is used to compute successive position estimates. Each new measurement constrains the probability distribution function representing the node’s position. This algorithm employs RSSI to determine the distance between the mobile beacon and the node. While this is the only work that we are familiar with that uses a mobile beacon in this manner, our work is distinct from it since our work neither requires that the mobile object, or target, know its own position, nor that it communicate this information to the nodes.

In [9], collaborative multilateration is presented. Position estimates are obtained by setting up a global non-linear optimization problem and solving it using iterative least squares. Both distributed and centralized computation models are presented and evaluated. Of particular relevance is the proposed method of establishing local coordinate systems by using laser rangefinders.

In [7], the authors investigate the effects of noise on range estimation. The paper concludes that range errors owing to RSSI are about twice the range errors owing to TOA. However, from a Cramer-Rao Bound analysis, the authors argue that at some density, a location system can perform as well using RSSI and TOA. A particularly relevant observation of this paper is that both RSSI *and* TOA are random functions not of time but of place. Obstructions between a pair of devices that cause shadowing and obstruction of the line-of-sight, or LOS, don’t change over time. However, two devices placed the same distance apart in a different area would have a different realization. The authors also note that RSSI and TOA can both be shown to demonstrate a very close fit to the Gaussian distribution. Based on these conclusions, we argue that the spatial averaging that is possible using a mobile target will result in smaller range estimation errors than for static nodes.

In [8], hop count to anchors is used as the basis for initiating localization and approximating range to a set of anchors. This phase is followed by a refinement phase in which more accurate positions are computed using the estimated ranges between nodes. The relevance of this work is that it provides a mechanism that is complimentary to our approach for cases in which the network requires at least a crude level of localization.

While our approach bears a resemblance to some of the ideas present in the related work, we are unaware of any other system that ranges a single mobile target as the basis for es-

timating the inter-node distances in support of localization. Once the approach outlined in this paper has been used to estimate inter-node ranges, any of the techniques for range-based localization described in this section can be used to transform the inter-node ranges into a coordinate system.

3. RANGE ESTIMATION

The approach outlined in this paper requires that sensor nodes be able to determine the distance to a target. Fortunately, a wide variety of techniques exist to allow a sensor to range a moving object. In [2], the design of an acoustic rangefinder is presented. Commercial ultrasonic rangefinders suitable for our purposes are available from Polaroid, Texas Instruments, and Devantech. These rangefinders use ultrasonic TOF and measure the round-trip-time (RTT) to estimate range. Inexpensive low-power rangefinders that use radar TOF to range objects can be licensed from McEwan Technologies. Laser rangefinders are available from a variety of sources including Nikon. It is possible to build inexpensive infrared triangulation rangefinders using commercially available light emitting diodes and position sensitive devices available from Hamamatsu Photonics. Pulse Doppler radars can be used in a ranging mode by counting zero crossings of the Doppler baseband signal. Such pulse Doppler radar sensors are available from Advantaca, and can be licensed from McEwan Technologies and Lawrence Livermore National Labs.

In addition to these direct ranging devices, sensors may be able to measure the energy dissipated from a target by viewing the target as an isotropically-decaying point source of energy. This approach is similar to measuring the RSSI and using that measurement to estimate distance. However, like RSSI, energy fields may be subject to many nuisance parameters and my experience a variety of environmental effects, making their use less desirable.

4. DISTANCE ESTIMATION

Given a range sensor that can determine the distance between a node and target, we now address the question of combining the range estimates from a pair of neighboring nodes into a single estimate, \hat{d} , of the distance separating the two nodes. We consider two general cases of target trajectories in computing \hat{d} : the target crosses the line segment connecting a pair of nodes or the target does not, as shown in Figures 1 and 2, respectively.

4.1 Case 1

As shown in Figure 1, the target crosses the line segment connecting the nodes. For the scenario in which the target take a generalized trajectory, Section 4.1.1 describes a method to compute \hat{d} . While the general solution always holds, its complexity can be improved for scenarios in which the target maintains a linear trajectory, as described in Section 4.1.2.

4.1.1 General Trajectory

Let us denote $r_1(t)$ and $r_2(t)$ as the distance at time t of the target from nodes N_1 and N_2 , respectively. Consider a function $r^+(t)$ such that

$$r^+(t) = r_1(t) + r_2(t) \quad (1)$$

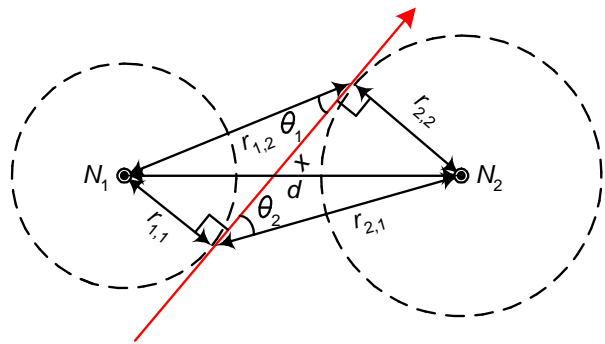


Figure 1: The trajectory of a target crosses the line segment connecting the nodes.

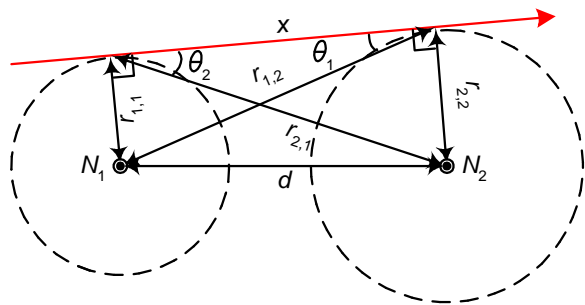


Figure 2: The trajectory of a target does not cross the line segment connecting the nodes.

We observe the minimum value of $r^+(t)$ occurs at the time, t_d , the target is crossing the line segment N_1N_2 and that

$$\hat{d} = r^+(t = t_d). \quad (2)$$

If $r^+(t)$ is monotonic, this computation requires $O(\lg n)$ operations using a binary search for t_d where n is the number of discrete samples of $r^+(t)$ over the interval of interest. If, however, $r^+(t)$ is *not* monotonic, then there may be many local minima, increasing the algorithm's complexity to $O(n)$. Note that for a linear trajectory, the time and message complexity remains $O(\lg n)$, assuming no optimizations. We observe that for a linear trajectory, the approach outlined in Section 4.1.2 is possible with constant complexity.

4.1.2 Linear Trajectory

The problem is to find d , the length of the line segment connecting N_1 and N_2 , given $r_{1,1}$, $r_{1,2}$, $r_{2,1}$, and $r_{2,2}$. Let θ_1 be the angle enclosed by x and $r_{1,2}$ and θ_2 be the angle enclosed by x and $r_{2,1}$. We can solve for d by applying the law of cosines

$$d = \sqrt{r_{1,1}^2 + r_{2,1}^2 - 2r_{1,1}r_{2,1} \cos(\pi/2 + \theta_2)} \quad (3)$$

and after substitution and simplification, we have

$$\hat{d} = \sqrt{r_{1,1}^2 + r_{2,1}^2 + 2r_{1,1}r_{2,2}} \quad (4)$$

and its dual

$$\hat{d} = \sqrt{r_{2,2}^2 + r_{1,2}^2 + 2r_{1,1}r_{2,2}} \quad (5)$$

In the special case where the target's trajectory, x , is perpendicular to and crosses the line segment N_1N_2 , we have $r_{1,1} = r_{1,2}$ and $r_{2,1} = r_{2,2}$. After substitution and simplification, we have

$$\hat{d} = r_{1,1} + r_{2,1} \quad (6)$$

and its dual

$$\hat{d} = r_{2,2} + r_{1,2} \quad (7)$$

which is the expected result.

The complexity of this computation is constant but the distance estimates that result from this computation may be poor if the target's motion is non-linear in the sensor's field of view.

4.2 Case 2

In Figure 2, the target does not cross the line segment connecting the nodes. For the scenario in which the target takes a general trajectory but crosses the *line* connecting points N_1N_2 within range of both sensors, Section 4.2.1 describes a method to compute \hat{d} . For the scenario in which the target maintains a linear trajectory, Section 4.2.2 describes an efficient method to compute \hat{d} .

4.2.1 General Trajectory

As before, let us denote $r_1(t)$ and $r_2(t)$ as the distance at time t of the target from nodes N_1 and N_2 , respectively. Consider a function $r^-(t)$ such that

$$r^-(t) = |r_1(t) - r_2(t)| \quad (8)$$

We observe the maximum value of $r^-(t)$ occurs at the time, t_d , the target is crossing the line that passes through the points N_1 and N_2 and that

$$\hat{d} = r^-(t = t_d) \quad (9)$$

As in the case of $r^+(t)$, if $r^-(t)$ is monotonic, this computation requires $O(\lg n)$ operations using a binary search to find t_d where n is the number of discrete samples of $r^-(t)$ over the interval of interest. Once again, we observe that for a linear trajectory, the approach outlined in Section 4.2.2 is possible with constant complexity.

4.2.2 Linear Trajectory

The problem, once again, is to find d , the length of the line segment connecting N_1 and N_2 , given $r_{1,1}$, $r_{1,2}$, $r_{2,1}$, and $r_{2,2}$. We call the value of \hat{d} computed using this configuration \hat{d}_L . As before, θ_1 is the angle enclosed by x and $r_{1,2}$ and θ_2 is the angle enclosed by x and $r_{2,1}$, giving

$$\hat{d} = \sqrt{r_{1,1}^2 + r_{2,1}^2 - 2r_{1,1}r_{2,2}} \quad (10)$$

and its dual

$$\hat{d} = \sqrt{r_{2,2}^2 + r_{1,2}^2 - 2r_{1,1}r_{2,2}} \quad (11)$$

In the special case where the target's trajectory, x , is perpendicular to and crosses the line passing through N_1N_2 but does not cross the line segment, N_1N_2 , we again have $r_{1,1} = r_{1,2}$ and $r_{2,1} = r_{2,2}$, giving

$$\hat{d} = |r_{1,1} - r_{2,1}| \quad (12)$$

and its dual

$$\hat{d} = |r_{2,2} - r_{1,2}| \quad (13)$$

which is the expected result.

Finally, in the special case where the target's trajectory, x , is parallel to the line passing through N_1N_2 , we have $r_{1,1} = r_{2,2}$ and $r_{1,2} = r_{2,1}$, giving

$$\hat{d} = \sqrt{r_{2,1}^2 - r_{1,1}^2} \quad (14)$$

and its dual

$$\hat{d} = \sqrt{r_{1,2}^2 - r_{2,2}^2} \quad (15)$$

which is usual the Pythagorean result.

4.3 Ambiguities and Assumptions

The two general cases of target trajectories outlined in Sections 4.1 and 4.2 (where the target either crosses the line segment connecting a pair of nodes or the target does not), are in general indistinguishable from each other since for any trajectory, a node N whose position is on one side of the trajectory line has an image, N' , situated at the reflection of N across the trajectory line. This ambiguity leads to two possible solutions for the distance separating a pair of nodes.

Despite the non-unique solution for any given trajectory, it is still possible to uniquely determine the distance separating two nodes by comparing the results of multiple distinct target trajectories. Each such trajectory will yield two possible solutions of which only one is the true distance. The other solution generally will vary for distinct trajectories. By comparing the distance estimates across multiple distinct trajectories, it is possible to identify the true distance as the mode, or most frequently occurring member, of this data set. However, the dataset is sparse initially (afterall, we do want to converge to a range estimate with a reasonably small number trajectories) and the data are noisy, precluding a search for single-valued mode in the data set. We will address this problem further in Section 4.4.

Prior to proceeding, we outline our assumptions as follow: (1) the sensing radius R_S is much greater than the average separation of the nodes; (2) only a single target is present in a network "neighborhood" at a given time; (3) a time synchronization service exists that allows the sensors to maintain a common timebase.

4.4 Pairwise Distance Estimation Algorithm

In order to estimate the distances in Sections 4.1 and 4.2, the nodes must exchange information with their neighbors about their own observations. In this section, we provide our algorithm for performing distance estimation between a pair of nodes.

4.4.1 Algorithm

Whenever a node initially detects the presence of a target, it notes the time, t_0 , and then begins keeping track of all subsequent range estimates to the target as long as the target remains present. When the target finally leaves the node's field of view, the node again notes the time, t_1 . We call the node *active* during this detection period spanning the time between t_0 and t_1 . While the node is active, it also keeps track of the shortest distance to the target it has measured, r_{CPA} , and the time of this measurement, t_{CPA} . The distance r_{CPA} denotes the closest point of approach or CPA.

The algorithm then tests the symmetry about t_{CPA} and the monotonicity over the segments t_0 to t_{CPA} and t_{CPA} to t_1 of the range samples. Even if the range samples are both symmetric and monotonic, the trajectory still may not be linear, but we assume that the trajectory *is* linear. At this point, the node broadcasts a message with the following fields: t_0 , t_1 , t_{CPA} , and r_{CPA} . The node also stores this information locally. Upon receiving a message from a neighbor, q , a node, p , checks to see if it is currently active. If so, then the node waits until it is no longer active before proceeding.

Linear Trajectory. If the trajectory is assumed to be linear, then at this point, the node processes the message if the predicate, P , in Equation 16 is true. Otherwise, the message is processed by the algorithm used in the general case.

$$P = (p.t_0 \leq q.t_{CPA} \leq p.t_1) \wedge (q.t_0 \leq p.t_{CPA} \leq q.t_1) \quad (16)$$

The predicate P tests whether the sending node, q , and receiving node, p , each observed the target at the time of the target's closest point of approach to the other. If the predicate is true, then node p sends a query message to node q requesting q 's range to the target, $q.p.r_{CPA}$, at the time of p 's CPA, $p.t_{CPA}$. Node p also includes its own range to the target, $p.q.r_{CPA}$ at time $q.t_{CPA}$ in this message. Node p determines $p.q.r_{CPA}$ by computing i according to Equation 17, and then retrieving the i -th range value collected after $p.t_0$.

$$i = (q.t_{CPA} - p.t_0)f_s \quad (17)$$

Upon receiving a response from node q , node p performs the following assignments

$$r_{1,1} = p.r_{CPA} \quad (18)$$

$$r_{1,2} = p.q.r_{CPA} \quad (19)$$

$$r_{2,1} = q.p.r_{CPA} \quad (20)$$

$$r_{2,2} = q.r_{CPA} \quad (21)$$

The node computes *four* different values of \hat{d} corresponding to the two cases described in Section 4.1 as \hat{d}_L and Section 4.2 as \hat{d}_H , respectively, and the dual ways of computing the value for each of these cases, \hat{d}_{H_1} , \hat{d}_{H_2} , \hat{d}_{L_1} , and \hat{d}_{L_2} . The node then averages the dual values such that

$$\hat{d}_H = (\hat{d}_{H_1} + \hat{d}_{H_2})/2 \quad (22)$$

$$\hat{d}_L = (\hat{d}_{L_1} + \hat{d}_{L_2})/2 \quad (23)$$

By averaging the duals, we reduce the error due to a single erroneous range estimate since each estimate is squared in only one of the two duals.

General Trajectory. If the range samples are either non-symmetric or non-monotonic, then the trajectory under consideration is not linear. Consequently, the node attempts to recover the distance estimate using ahd

In the case of a general trajectory, the message complexity will increase. The simplest and most effective method of finding the minimum sum of two sets of ranges is to communicate a node's entire dataset from t_0 to t_1 . When node p finishes its data collection, it broadcasts a request for range data in timespan $[t_0, t_1]$ to its neighbors. Upon receiving a range request, a neighbor node, q , will find any data that overlaps with p 's dataset and return this data along with the time endpoints of the overlapping data. Upon receiving neighbor q 's data, p will sum this with its own data and return the minimum sum (it's estimated range from node p to node q).

It is conceivable that nodes p and q might be able to find the minimum sum without exchanging their entire datasets. In the case of many trajectories, this minimum sum will occur between the closest point of approach to each node. In this case, it would only be necessary to exchange the data in $[t_{cpaA}, t_{cpaB}]$ between nodes. For nodes with large sensing radii, this could provide a savings in communication costs. However, there are trajectories in which the mobile object can pass close to both nodes, but not between them giving a non-optimal reading.

One approach to reducing message complexity is to use a binary search over the sample space. For such an approach, node p sends a sample of its data to q , after which the nodes use a gradient descent method to find the the minimum sum. This method, like nearly all gradient descent methods, is susceptible to local minima. As the data indicate in Figure 6, this concern is not merely academic. Regardless, for cases in which the dataset is large (e.g. large sampling radius or high frequency), a binary search method may be appropriate.

Computing the Most Likely Estimate. Each node keeps both the \hat{d}_H and \hat{d}_L values that are estimated in each round in a circular buffer. For any particular target trajectory, only one of these values will yield the actual separation between the nodes while the other value generally will vary for distinct trajectories. Consequently, about 50% of our estimates will tend to cluster around the true value of d while the other 50% will be incorrect. The distribution may be symmetric, right-tailed, or left-tailed, depending on the target trajectories.

Our problem, then, is to find the mode of this data set. Unfortunately, since the data are noisy, the mode is not single valued. Various methods [5] have been proposed to compute the mode of such a distribution. We adopt the median as an estimator of the mode, since we expect 50% of the data to

be clustered around the true distance. Therefore, with each new set of estimates, the node computes the median, \tilde{d} , and uses this value as the estimate $\hat{d} = \tilde{d}$.

We do note that a more robust estimator might be one that estimates the mode as

$$mode = 2 \times median - mean \quad (24)$$

```

integer i = 0, max;
integer ranges[2*Rs*fs/vmin];
integer times[2*Rs*fs/vmin];
integer t0, t1, tcpa, rcpa;
boolean active = false;
boolean symmetric = false;
boolean monotonic = false;

event detected(boolean detect) {
    active = detect;
    if (detect) {
        t0 = time();
        i = 0;
    }
    else {
        t1 = time();
        ranges = movingmedian3pt(ranges);
        ranges = movingaverage8pt(ranges);
        symmetric = checksymmetric();
        monotonic = checkmonotonic();
        send(t0, t1, tcpa, rcpa, symmetric, monotonic);
    }
}

event ranged(int range) {
    if (active) {
        ranges[i++] = range;
    }
}

event receivedRequestMsg(t0,t1) {
    j = find(t0);
    k = find(t1);
    send(ranges[j:k],times[j],times[k]);
}

event receivedDataMsg(ranges[j:k],t3,t4) {
    m = find(t3);
    n = find(t4);
    send(min(sum(ranges[j:k]) + ranges[m:n]));
}

boolean checksymmetric(t0, t1, tcpa) {
    boolean symmetric = true;
    for (j = 0; j < t1-t0; j++) {
        symmetric = |ranges[j] - ranges[t1-t0-j]|
            > threshold ? false : symmetric;
    }
    return symmetric;
}

boolean checkmonotonic(t0, t1, tcpa) {
    boolean symmetric = true;
    for (j = 0; j < t1-t0; j++) {
        symmetric = |ranges[j] - ranges[t1-t0-j]|
            > threshold ? false : symmetric;
    }

    rcpa = rcpa < range ? rcpa : range;
    tcpa = tcpa < range ? tcpa : time();

    return monotonic;
}

```

Listing 1: A pseudo-code representation of the algorithm.

5. IMPLEMENTATION

In order to characterize the performance of our algorithms, we ran several simulations in Matlab and validated our simulations with empirical test cases.

5.1 Network Nodes

The Mica2Dot mote, a member of the Mica family of motes developed at U.C. Berkeley [4], served as our network node. The Mica2Dot offers an Atmel processor clocked at 4MHz with 4KB of random access memory, 128KB of FLASH program memory, and 512KB of EEPROM memory. The motes run the TinyOS operating system [3], and are programmed using the NesC language [1]. Our nodes are shown in Figure 3.



Figure 3: Our experimental hardware consists of Mica2Dot motes (center), and clockwise from the top: rechargeable battery with top-mounted power adapter board, recharger contact board, HoneyDot magnetometer (unused in this experiment), ultrasonic transceiver board, and the complete sensor node including an inverted cone for reflecting the ultrasonic signal omnidirectionally.

5.2 Ultrasound Transceiver

We used an ultrasound transceiver board to simulate ranging data from the mobile mote to the network nodes, as shown in Figure 3. The mobile object carried an ultrasound transmitter that issued ultrasound “chirps” every 250ms, where each chirp consists of a radio message and ultrasound tone at 25kHz. While it is certainly unlikely that a target travelling through a sensor network will be equipped with an ultrasound transmitter, ultrasound provided a convenient platform with which to find ranging information to the mobile object without having specific information on the position of the mobile object. Each network node used an ultrasound receiver to measure the time difference of arrival (TDOA) between the radio message and ultrasound beep.

In order to obtain accurate ranging data, we ran calibration tests on each node, placing the transmitter at 10cm increments from the receiving node. The median of each data set is shown in Figure 8. A simple line fit provides the two parameters needed to calibrate future data. Using this same data, we subtracted out the median value from each data set

to measure the noise from the ultrasound transceiver. This noise was found to be surprisingly uniform over a ± 5 cm range as shown in Figure 9.

5.3 Experimental Setup

Our experimental testbed is shown in Figure 4. The setup consists of four Mica2Dot motes placed in a rectangle with sides of length 70cm and 90cm. There are four test trajectories: A, B, C, and D. A is diagonal red-colored straight line trajectory that separates the testbed into two halves with nodes 1 and 3 on one side and nodes 2 and 4 on the other side. B is straight line trajectory that separates the testbed into two halves with nodes 1 and 2 on one side and nodes 3 and 4 on the other side. C is a curved blue-dotted trajectory that starts near node 1 and weaves between nodes 1 and 2, then between nodes 1 and 3, then between nodes 3 and 4, then again between nodes 1 and 4, and finally between nodes 4 and 2. D is a curved yellow-dotted trajectory that starts near node 1 and follows the edge of the field toward node 3, then turns a corner and continues toward node 4. In this setup, the target was moved manually, although we hope to use autonomous or remote controlled robots in the future for this task.

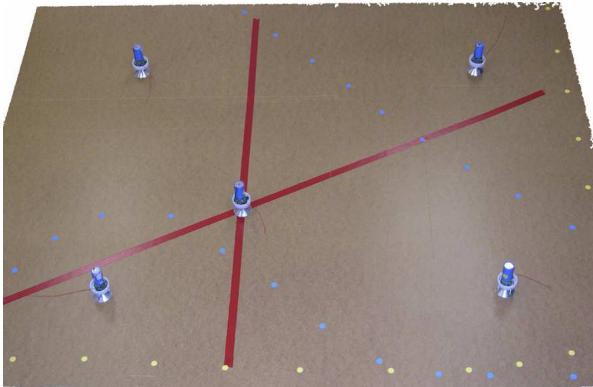


Figure 4: Our experimental setup consists of four Mica2Dot motes: bottom left(1), top left(2), bottom right(3), and top right(4).

6. RESULTS

We present the results of our experiments in this section. Figure 5 shows the measured range from each of the four nodes to the target over the observation window of interest for trajectory C.

In Figure 6, the sum, $r^+(t)$, is shown in red in the top half of each figure and difference, $r^-(t)$, is shown in blue in the bottom half of each figure. The black horizontal line shows the true distance. The minimum sum and the maximum difference are our estimators for the true distance between a pair of nodes. For the cases in which the target trajectory satisfies the requirement of crossing the line N_1N_2 , we see that the estimates are quite close to the actual distance. The Case 1 trajectory occurs for all node pairs except between 2 and 3. We see that in every case, the minimum value of the top line (sum) provides a good estimate. The Case 2 trajectory occurs for all node pairs except between 2 and 4.

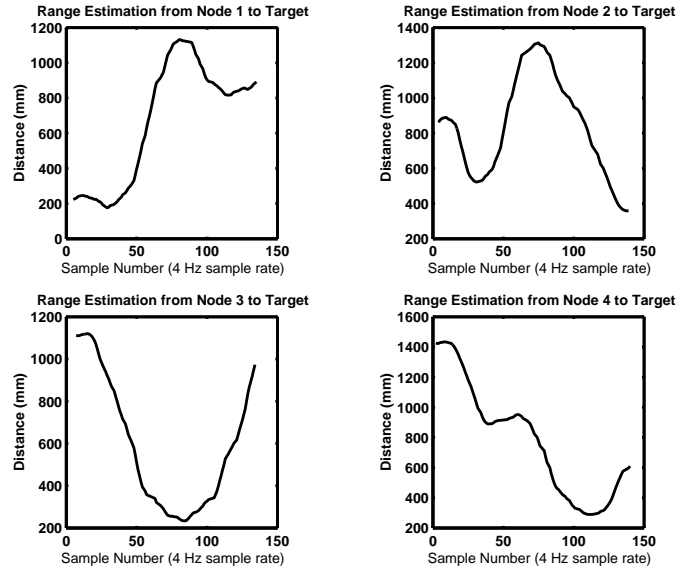


Figure 5: The range from each of the four nodes to the target over the observation window of interest for trajectory C.

Again, we see that in every case, the maximum value of the bottom line (difference) provides a good estimate of the true distance.

Table 1 compares the cumulative range errors of the CPA, sum, and difference algorithms as function of the number of passing targets. We see that both the CPA and sum algorithms provide only a few centimeters of error whereas the difference error is quite large.

	Run 1	Run 2	Run 3	Run 4
CPA Error	7cm	11cm	11cm	6cm
Sum Error	9cm	16cm	4cm	7cm
Difference Error	34cm	31cm	20cm	14cm

Table 1: A comparison of the cumulative range errors of the CPA, sum, and difference algorithms as function of the number of passing targets.

7. FUTURE WORK

Our approach performs poorly when the target neither maintains a constant heading nor crosses the line N_1N_2 . We have identified methods to determine when the target is not maintaining a constant heading, thus reducing the likelihood of poor distance estimates by rejecting these estimates or using the sum/difference algorithm. However, a more general approach would not place such a constraint on the target trajectory. Our future work will address this scenario. Consider, for example, a configuration that allows three independent sensor nodes to range the same target simultaneously at three distinct points in time, as shown in Figure 7.

The positions of the three nodes N_1, N_2 , and N_3 are unknown and we refer to these nodes as the *unknown* nodes.

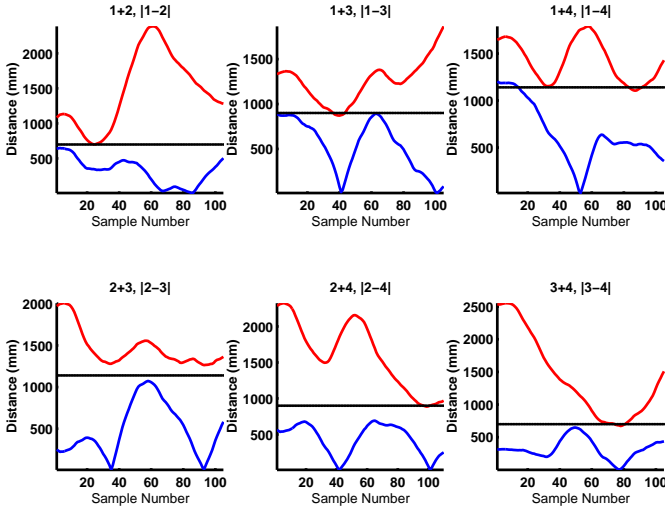


Figure 6: The sum, $r^+(t)$, is shown in red in the top half of each figure and difference, $r^-(t)$, is shown in blue in the bottom half of each figure. The black horizontal line shows the true distance.

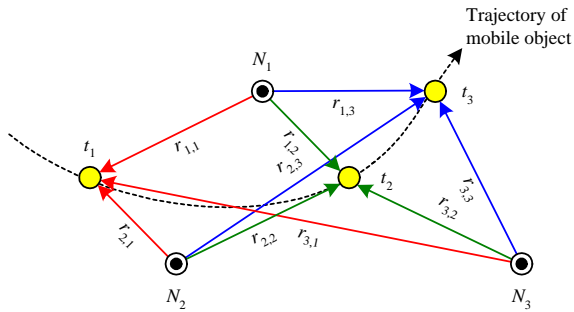


Figure 7: The trajectory of a target (yellow dot) and its distance $r_{n,t}$ from three sensor nodes at three points in time, where n is one of N_1, N_2 , and N_3 , and t is one of t_1, t_2 , and t_3 .

Each of these unknown nodes is able to determine the distance to the target at three distinct times t_1, t_2 , and t_3 . Let us define the *range graph* as the planar geometric structure whose six vertices are defined by the location of three unknown nodes N_1, N_2 , and N_3 , and of a target at three times t_1, t_2 , and t_3 , and whose edges are the nine ranges $r_{i,j}$. We have proven that the range graph is both rigid and unique if neither the three nodes nor the three points of the trajectory are collinear. Solving the resulting quadratically constrained optimization problem will yield the positions of the three nodes and position of the target at the three times.

8. CONCLUSIONS

We have presented a novel algorithm for determining the distance between a pair of neighboring nodes that are able to simultaneously range a target. Our approach works by exchanging a small number of messages between such nodes and is both distributed and scalable since all computations

and communications are local. In contrast to earlier work, our approach neither requires that the target know or communicate its own position nor that it be cooperative in its trajectory selection. We identified metrics that the nodes can use to determine the quality of the inter-node distance estimations based on the range estimates to the target. We also identified a variety of sensors capable of providing the kind of range information that is needed for our algorithm. We proposed a lazy localization strategy in which nodes determine the ranges to neighboring nodes, and consequently their own positions, only when targets actually pass by.

9. REFERENCES

- [1] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesc language: A holistic approach to networked embedded systems. In *Proceedings of Programming Language Design and Implementation (PLDI) 2003*, 2003.
- [2] L. Girod. Development and characterization of an acoustic rangefinder, 2000.
- [3] J. Hill. A software architecture supporting networked sensors. *Master's thesis, U.C. Berkeley Dept. of Electrical Engineering and Computer Sciences*, 2000.
- [4] J. Hill and D. Culler. Mica: A wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24, 2002.
- [5] M. Kendall and A. Stuart. *The Advanced Theory of Statistics*. Charles Griffin, London, U.K., 1979.
- [6] K. Langendoen and N. Reijers. Distributed localization in wireless sensor networks: A quantitative comparison. *Computer Networks (Elsevier)*, 43:499–518, Nov. 2003.
- [7] N. Patwari and A. O. Hero. Location estimation accuracy in wireless sensor networks.
- [8] C. Savarese, J. Rabaey, and K. Langendoen. Robust positioning algorithms for distributed ad-hoc wireless sensor networks. In *USENIX Technical Annual Conference*, pages 317–328. USENIX, 2002.
- [9] A. Savvides, H. Park, and M. B. Srivastava. The bits and flops of the n-hop multilateration primitive for node localization problems. *WSNA'02*, Sept. 2002.
- [10] M. L. Sichitiu and V. Ramadurai. Localization of wireless sensor networks with a mobile beacon. *NCSU Center for Advances Computing and Communications (CACC) Technical Report TR-03/06*, July 2003.
- [11] B. Sinopoli, C. Sharp, L. Schenato, S. Schaffert, and S. S. Sastry. Distributed control applications within sensor networks. *Proceedings of the IEEE, Special Issue on Sensor Networks and Applications*, 91(8):1235–1246, Aug. 2003.

APPENDIX

A. MISCELLANEOUS

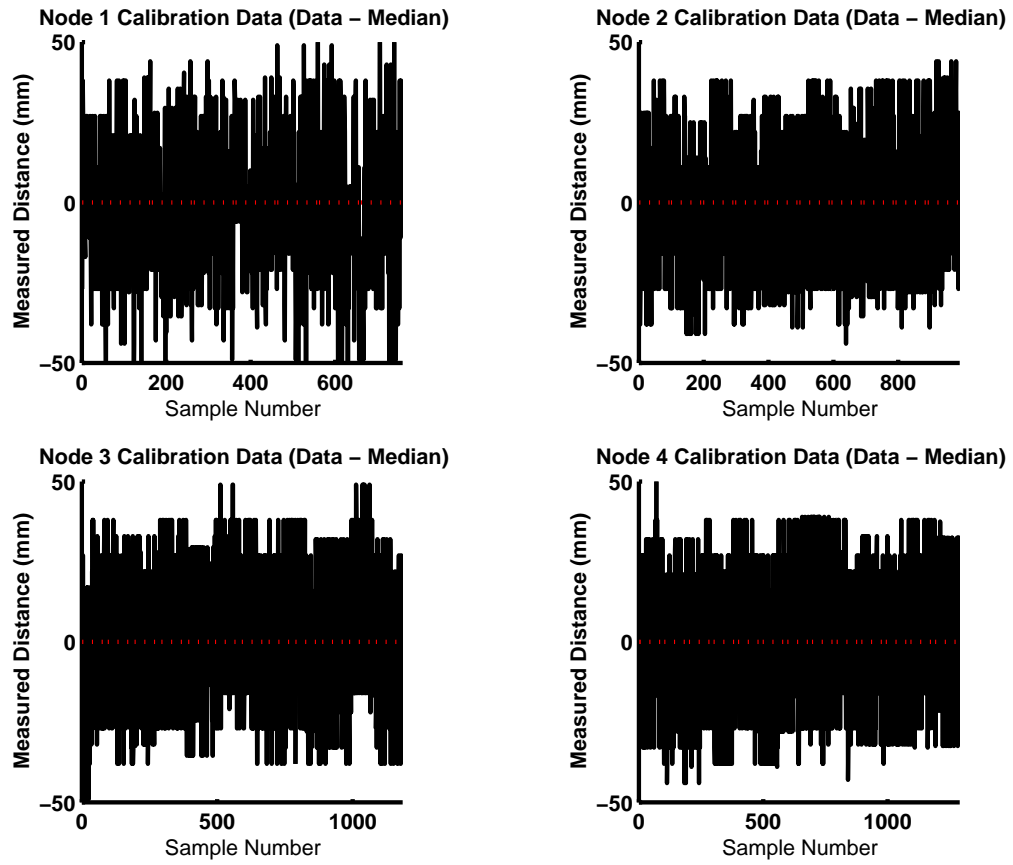


Figure 8: Temporal variation of the range error over time.

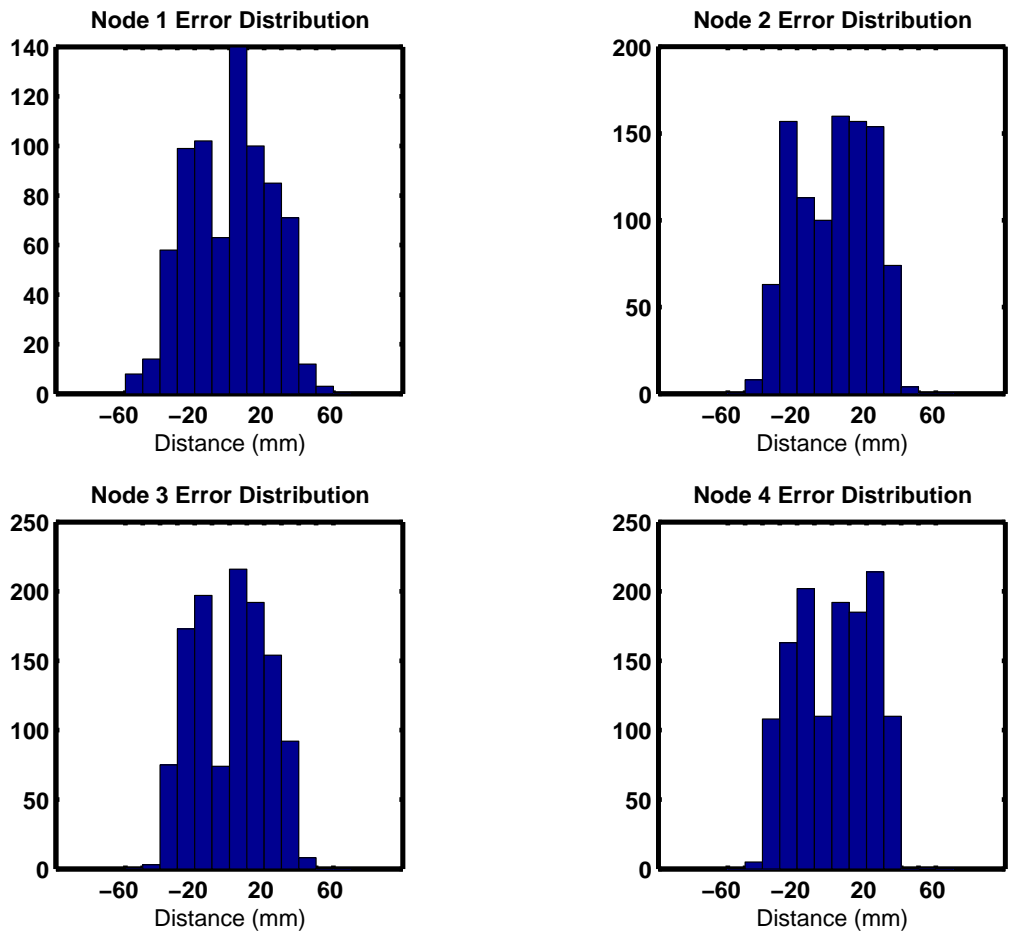


Figure 9: The range error distribution for our four nodes. More than 99% of the measurements fall within ± 6 cm of the true distance.

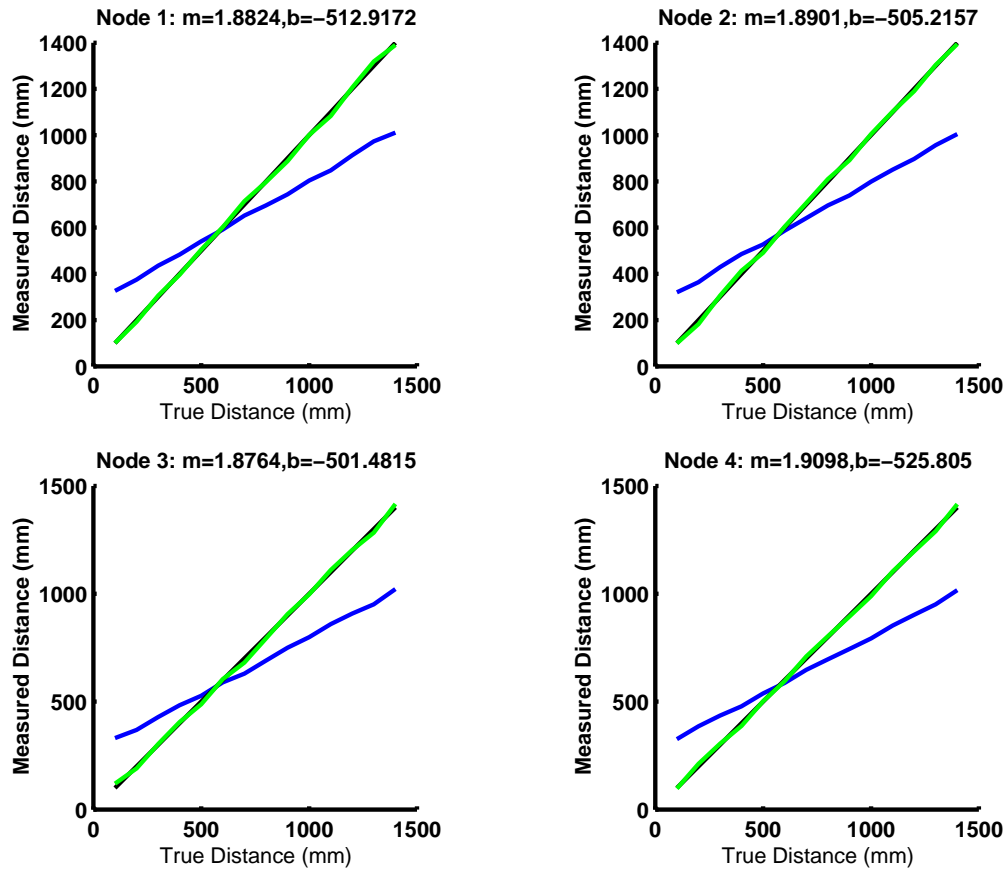


Figure 10: The uncalibrated and calibrated curves for our sensors.

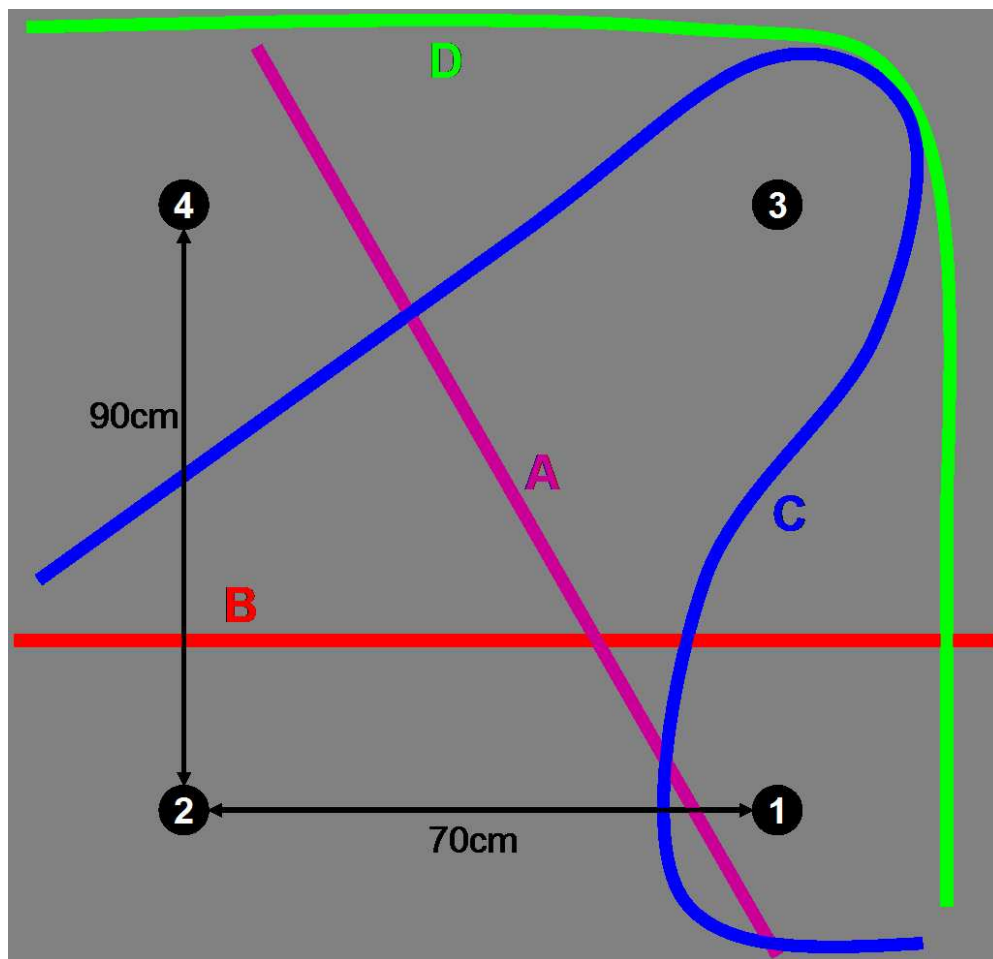


Figure 11: The target trajectories.