

Localizing Tags Using Mobile Infrastructure

Ying Zhang, Kurt Partridge, and Jim Reich

Palo Alto Research Center
3333 Coyote Hill Road, Palo Alto, CA, 94304, USA
{yzhang, kurt, jreich}@parc.com

Abstract. This paper presents algorithms, simulations, and empirical results of a system that finds relative tag positions in 3D space using a new approach called “mobile infrastructure.” Mobile infrastructure consists of one or more sensors in a fixed configuration on a mobile platform, and a set of tags affixed to objects or locations in the environment which the users want to localize. It is especially useful in cases where infrastructure is needed only temporarily, such as during installation, calibration, or maintenance. Mobile infrastructure can cover a much larger area than installed infrastructure with the same number of sensors, and is especially useful in cases where localization hardware costs are asymmetric, with expensive sensors and inexpensive tags. The data collected at various positions are combined by a simple “leapfrog” procedure, with constrained optimization to obtain better accuracy. Our system achieves about one foot (0.3 meter) accuracy with 90% confidence in indoor environments.

1 Introduction and Related Work

Location-awareness is a key component for achieving context-awareness. In most cases, a ubiquitous computing system must know the locations of its sensor and actuator devices. However, few systems bootstrap the locations of devices in a way that is acceptable to end-users. Often, users are expected to manually enter room names or manually survey geometric coordinates. This greatly increases the barrier to entry of ubiquitous computing systems and location based services, due to the expense of skilled installer labor, high probability of data-entry error and difficulty of troubleshooting configuration errors.

There are two general approaches to automatically bootstrap device positions. The first approach is to use an already-available positioning system. However, GPS only works reliably outdoors and outside of city centers, GSM [15] or TV-signal based [20] localization are precise only to tens of meters, and WiFi either has similar precision problems [9–11] and/or requires extensive preparatory manual surveying and calibration [5, 8]. Powerline positioning [16] has also been proposed, but it also requires a manual survey.

The second approach is for devices to self-localize by collectively determining their positions relative to each other. Time-Of-Arrival (TOA) and Time Difference of Arrival (TDOA) ultrasonic-based systems, while accurate to a few centimeters [14, 23], are limited to a roughly five meter range and are subject

to reflections, obstructions, and directionality restrictions. In many cases, they only work in 2D spaces [19, 25]. Radio Signal Strength (RSS) systems [7] are also subject to environmentally-determined attenuation, and are precise only to several meters. Ultrawideband (UWB) systems have better range (up to 100m) and accuracy (down to 15cm), but these systems are either not commercially available [1], or are still too expensive for many applications [2, 3], since they require the developer to embed an expensive wireless chip in each device.

In this paper, we introduce a third approach to bootstrapping device locations. Our approach is based on taking a series of measurements from ultrawideband Angle-of-Arrival (AOA) sensors mounted on a mobile cart. UWB provides the high-accuracy localization in the presence of clutter and environmental variation, while the cart’s mobility allows our platform to act like a set of “virtual nodes,” covering many positions.

Our experiments test this approach using Ubisense’s localization hardware [4]. Unlike other UWB positioning systems, Ubisense has two types of nodes: UWB transmitters (“Ubitags”), which are small (6cm x 9cm x 1cm), light (50g), and cheap), and UWB receivers (“Ubisensors”), which are larger (21cm x 13cm x 7cm), heavier (690g), and more expensive. Ubisensors determine the position of the Ubitags. The Ubisense system is designed to be a general-purpose location infrastructure. In its intended use, the Ubisensors are permanently installed in the upper (usually four) corners of a room, and their relative positions are carefully measured and manually input by a skilled technician.

In our system, we reverse the roles of the Ubisense hardware: the Ubisensors are attached to the a moveable cart (see Fig. 7), and the Ubitags are assumed to be embedded or attached to the non-moving devices that are to be localized. As the cart is wheeled around, it computes the positions of the tags it sees. Provided that different perspectives detect enough overlapping tag sets, it is possible to determine the positions of all the tags in a single, common reference frame, which may span a space far larger than could be covered by a comparable set of Ubisensors. Furthermore, the cart can be later reused for a completely different installation. We call this approach “mobile infrastructure.”

Our work contributes an algorithm for static tag localization based on AOA measurements, evaluates the effectiveness of this algorithm both in simulation and in practice, and evaluates improvements to the simple leapfrog algorithm with constrained optimization technique. We also study AOA data obtained from Ubisensors, to preprocess and pre-filter the raw data to get better and more robust localization results.

There are similarities between our approach and mobile-assisted localization (MAL) [17, 18, 21, 22, 24]. Generally, MAL algorithms estimate relative distances rather than angles and assume that all nodes, mobile and static, can both transmit and receive; generally, they have assumed homogeneous installations in which the mobile nodes are identical to the static nodes. Our work also differs from SLAM [13] in the robotics community, in which one or more robots simultaneously produce a map of the walls and obstacles in the environment (often through laser scanners or vision) while navigating. However, SLAM is not fo-

cused on determining the relative locations of tagged objects, and is specifically intended for autonomous mobile platforms rather than human-controlled ones.

The rest of the paper is organized as follows. Section 2 provides a system overview for tag localization using mobile infrastructure. Section 3 presents the algorithms, a leapfrog procedure and refinements to it using constrained optimization. Section 4 presents simulation results that evaluate the algorithms given different sensor configuration and noise parameters. Section 5 demonstrates the system using a real hardware deployment, with an emphasis on preprocessing and pre-filtering AOA data from the Ubisense System. Section 6 concludes the paper and suggests future directions.

2 System Overview

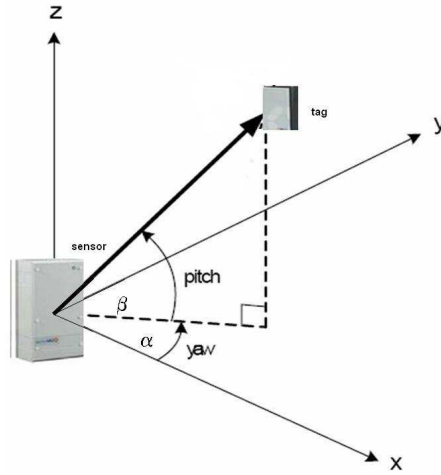


Fig. 1. Local reference frame for a sensor and AOA of a tag.

Each AOA sensor has six degrees of freedom in space: $x_s, y_s, z_s, a_s, b_s, r_s$ where x_s, y_s, z_s are 3D coordinates and a_s, b_s, r_s are yaw, pitch and roll angles, respectively. In the rest of this paper, we assume $r_s = 0$. Tags have x_t, y_t, z_t locations, but no orientation. Each AOA sensor's parameters (x_s, y_s, z_s, a_s, b_s) define its own reference frame. The position of a tag in a sensor's reference frame uniquely determines the AOA: yaw α and pitch β (Fig. 1). Each tag/sensor pair introduces two equations. Let x_t^s, y_t^s, z_t^s be the location coordinates of a tag in the sensor's reference frame, and α and β be the yaw and pitch angles, respectively. From these, the following two equations may be deduced:

$$x_t^s \sin(\alpha) - y_t^s \cos(\alpha) = 0 \quad (1)$$

$$x_t^s \sin(\beta) - z_t^s \cos(\beta) \cos(\alpha) = 0 \quad (2)$$

where x_t^s, y_t^s, z_t^s can be obtained given the sensor's position x_s, y_s, z_s, a_s, b_s and the tag's position x_t, y_t, z_t in a global reference frame according to a standard coordinate transformation [6]:

$$\begin{pmatrix} x_t^s \\ y_t^s \\ z_t^s \\ 1 \end{pmatrix} = \begin{bmatrix} T' & -T' \begin{pmatrix} x_s \\ y_s \\ z_s \end{pmatrix} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x_t \\ y_t \\ z_t \\ 1 \end{pmatrix} \quad (3)$$

where T is the rotational transformation matrix for the sensor frame:

$$\begin{bmatrix} \cos(a_s) \cos(b_s) - \sin(a_s) \cos(a_s) \sin(b_s) \\ \sin(a_s) \cos(b_s) \cos(a_s) \sin(a_s) \sin(b_s) \\ -\sin(b_s) & 0 & \cos(b_s) \end{bmatrix} \quad (4)$$

Using the set of equations (Eqs. 1 and 2 for each tag/sensor pair), one can compute a tag's location x_t, y_t, z_t given AOA data from two sensors with known locations (four equations, three unknowns). Or one can compute a sensor's orientation (a_s and b_s) given AOA data from a fixed tag at a known position relative to the sensor (two equations, two unknowns).

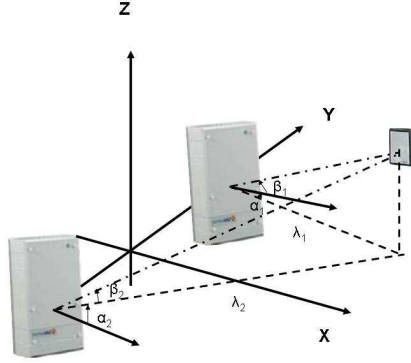


Fig. 2. Two sensor platform with the local frame in the center between the sensors.

Our problem is to localize a set of static tags in 3D space. The sensor pair on the cart is depicted in (Fig. 2), and for simplicity, we set $b_s = 0$. The distance between the two sensors is fixed. The cart can move in four degrees of freedom, x_c, y_c, z_c and yaw a_c . A simple, fixed transformation exists between the cart coordinates and sensor coordinates. At each cart position, only a subset of tags can be seen by the sensors, however, we assume each tag can be seen by the sensors at least once at some cart position.

We roughly estimate how many cart positions are needed for tag localization as follows. Let the first cart position be the global reference frame. Each new cart

position adds four unknowns. If at least k tags can be seen at a cart position, at least $4k$ equations are added. Let n be the total number of tags and m be the number of extra cart positions. There are $3n + 4m$ unknowns and at least $4k(m + 1)$ equations. One may solve the set of equation for the unknowns if $4k(m + 1) \geq 3n + 4m$, i.e., the number of equations is greater than the number of unknowns.

Note that the number of cart positions is only one factor in determining whether the equations can be solved. Another important factor is the connectivity of the overall tag/sensor pairs. A tag is *connected* to a sensor at a position if it can be seen by the sensor at that position. There are $2(m + 1)$ sensor nodes and n tag nodes for $m + 1$ positions and n tags, which constitute a bipartite graph. Such a graph has to be at least connected to have all coordinates in the same reference frame. For example, if three tags are seen by the sensors in the first cart position, and a totally different three sensors are seen by the sensors in the second cart position, one cannot obtain the six tag locations in a common reference frame, although $4 \times 3 \times 2 \geq 3 \times 6 + 4$. Let $c \geq 1$ be the minimum number of tags in common for a new location. Given n tags, if at most K tags can be seen at a time, i.e., at most $K - c$ new tags can be seen at each extra location, we need $K + (K - c)m \geq n$ to see all the tags. In the previous example, we have $3 + (3 - 1) < 6$.

The inputs to our algorithm are AOA data at each cart position, i.e., α and β , where $\alpha(i, j, l)$ and $\beta(i, j, l)$ are the yaw and pitch angles, respectively, from sensor j to tag i at l 'th position. AOA sensing scope is limited by both distance and angles. If tag i is out of range of sensor j at position l , $\alpha(i, j, l)$ and $\beta(i, j, l)$ are set to infinity. Given the input data α and β , we first filter out bad cart positions. A cart position is bad if it does not have at least two sensor/tag connections. We remove such cart positions since each position adds four variables and one pair of sensor/tag connection adds two equations; a good cart position must add more equations than variables.

3 Localization Algorithm using the Mobile Sensing Pair

We can plug the data α and β into the equations (1) - (2) in the previous section, and use a nonlinear solver to find the n tag locations and m cart positions. However, since the constraints are nonlinear and the size of the problem is large ($4m + 3n$ variables for m positions and n tags), the solution in practice is intractable. Fortunately, due to the special configuration of the sensor platform, we can get a complete or partial solution using a set of linear equations.

3.1 Closed-form Solutions

The procedure consists of two components: from sensors to tags, and from tags to sensors.

From Sensors to Tags Given a position of the two-sensor frame (Fig. 3), x, y, z, a , the two sensor locations are x_1, y_1, z, a and x_2, y_2, z, a , respectively, with $x_1 = x - d \sin(a)/2$, $y_1 = y + d \cos(a)/2$ and $x_2 = x + d \sin(a)/2$ and $y_2 = y - d \cos(a)/2$ where d is the distance between the two sensors.

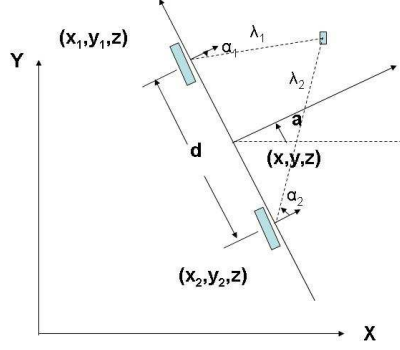


Fig. 3. 2D projection of the two sensor platform.

Let λ_1 and λ_2 be the distances from a tag to the two sensors in the XY plane, respectively. We have

$$x_t = x_1 + \lambda_1 \cos(\alpha_1 + a) \quad (5)$$

$$y_t = y_1 + \lambda_1 \sin(\alpha_1 + a) \quad (6)$$

$$x_t = x_2 + \lambda_2 \cos(\alpha_2 + a) \quad (7)$$

$$y_t = y_2 + \lambda_2 \sin(\alpha_2 + a) \quad (8)$$

and from

$$\lambda_1 \cos(\alpha_1 + a) - \lambda_2 \cos(\alpha_2 + a) = x_2 - x_1 = d \sin(a)$$

$$\lambda_1 \sin(\alpha_1 + a) - \lambda_2 \sin(\alpha_2 + a) = y_2 - y_1 = -d \cos(a)$$

one can solve for λ_1 and λ_2 . Setting these values in Eqs. (5) and (7), we have

$$x_t = \frac{1}{2}[(x_1 + \lambda_1 \cos(\alpha_1 + a)) + (x_2 + \lambda_2 \cos(\alpha_2 + a))] \quad (9)$$

and similarly,

$$y_t = \frac{1}{2}[(y_1 + \lambda_1 \sin(\alpha_1 + a)) + (y_2 + \lambda_2 \sin(\alpha_2 + a))] \quad (10)$$

Using equations

$$z_t = z + \lambda_1 \tan(\beta_1) \quad (11)$$

$$z_t = z + \lambda_2 \tan(\beta_2) \quad (12)$$

we have

$$z_t = z + \frac{1}{2}(\lambda_1 \tan(\beta_1) + \lambda_2 \tan(\beta_2)) \quad (13)$$

From Tags to Sensors If the pair of sensors can see multiple tags with known positions, the sensor frame position x, y, z, a can be obtained. Let α_{ik} and β_{ik} be yaw and pitch angles from sensor k to tag i , respectively, and let λ_{ik} be the projected distance between tag i and sensor k on the XY plane. If a tag i can be seen by both sensors, we have

$$-\lambda_{i1} \sin(\alpha_{i1}) + \lambda_{i2} \sin(\alpha_{i2}) = d \quad (14)$$

$$\lambda_{i1} \cos(\alpha_{i1}) - \lambda_{i2} \cos(\alpha_{i2}) = 0 \quad (15)$$

and we can compute λ_{i1} and λ_{i2} . Given a pair of tags i and j , and a sensor at x_k, y_k, z_k, a , where $k = 1$ or 2 , we have

$$x_k = x_i - \lambda_{ik} \cos(\alpha_{ik} + a) \quad (16)$$

$$y_k = y_i - \lambda_{ik} \sin(\alpha_{ik} + a) \quad (17)$$

$$x_k = x_j - \lambda_{jk} \cos(\alpha_{jk} + a) \quad (18)$$

$$y_k = y_j - \lambda_{jk} \sin(\alpha_{jk} + a) \quad (19)$$

and

$$x_i - x_j - \lambda_{ik} \cos(\alpha_{ik} + a) + \lambda_{jk} \cos(\alpha_{jk} + a) = 0 \quad (20)$$

$$y_i - y_j - \lambda_{ik} \sin(\alpha_{ik} + a) + \lambda_{jk} \sin(\alpha_{jk} + a) = 0 \quad (21)$$

$$(22)$$

Let (20) $\cos(a)$ + (21) $\sin(a)$ we have

$$(x_i - x_j) \cos(a) + (y_i - y_j) \sin(a) = \lambda_{ik} \cos(\alpha_{ik}) - \lambda_{jk} \cos(\alpha_{jk}) \quad (23)$$

and let (20) $\sin(a)$ - (21) $\cos(a)$ we have

$$(x_i - x_j) \sin(a) - (y_i - y_j) \cos(a) = -\lambda_{ik} \sin(\alpha_{ik}) + \lambda_{jk} \sin(\alpha_{jk}) \quad (24)$$

If a sensor sees n tags, there are $2(n - 1)$ linear equations with two variables $\cos(a)$ and $\sin(a)$. When $n \geq 2$, one can solve the set of linear equations and obtain $\cos(a)$ and $\sin(a)$. Therefore $a = \arctan\left(\frac{\sin(a)}{\cos(a)}\right)$. Using Eq. (16) and (17) for each tag i , we have

$$x_1^i = x_i - \lambda_{i1} \cos(\alpha_{i1} + a)$$

$$y_1^i = y_i - \lambda_{i1} \sin(\alpha_{i1} + a)$$

$$x_2^i = x_i - \lambda_{i2} \cos(\alpha_{i2} + a)$$

$$y_2^i = y_i - \lambda_{i2} \sin(\alpha_{i2} + a)$$

and also

$$\begin{aligned} z_1^i &= z_i - \lambda_{i1} \tan(\beta_{i1}) \\ z_2^i &= z_i - \lambda_{i2} \tan(\beta_{i2}) \end{aligned}$$

Therefore, the estimated locations for sensor 1 and 2 seeing n tags are

$$\left(\frac{1}{n} \sum_{i=1}^n x_1^i, \frac{1}{n} \sum_{i=1}^n y_1^i, \frac{1}{n} \sum_{i=1}^n z_1^i \right)$$

and

$$\left(\frac{1}{n} \sum_{i=1}^n x_2^i, \frac{1}{n} \sum_{i=1}^n y_2^i, \frac{1}{n} \sum_{i=1}^n z_2^i \right),$$

respectively. The center of the sensor frame is at: $\left(\frac{x_1+x_2}{2}, \frac{y_1+y_2}{2}, \frac{z_1+z_2}{2} \right)$.

3.2 Leapfrog Algorithm

The leapfrog algorithm is the approach we use to calculate all tag positions in a common reference frame. Starting from the position from which we can see the maximum number of tags (this will be used as the reference frame), the leapfrog algorithm alternates between computation of tag locations and cart locations, using the closed-form solutions from the previous section, until all the locations are obtained. In particular, after locating the tags using the in the current sensor frame, it then selects as the next frame the sensor frame with the maximum number of *known tags* (i.e. those which have been observed by any of the sensor frames we have already processed). It computes the location of this new frame, and proceeds iteratively. The pseudocode is shown in Table 1.

The leapfrog algorithm is simple, but it may only give a partial solution if there are not enough connections. For example, in line 1 of the algorithm, there may be no tags connected to both sensors, i.e. $cTs = \emptyset$, although there may be tags connected to one sensor. Also, in line 5, there may be no next unknown sensor frame that can see at least two known tags. Another potential weakness of this algorithm is that errors in localization of the previous tags will propagate into the localization of the next frame, accumulating over time. How to select the best next sensor frame to reduce error propagation is an open question for future research. Some techniques in error control for self-localization similar to those in [12] may be employed. This error accumulation might also be limited by adding a small number of anchor points with known locations, e.g., using a blueprint of the building.

3.3 Optimization-based Algorithms

The leapfrog algorithm does not work well when connectivity is low and when inputs are noisy. The optimization-based algorithms below are more robust and work by minimizing the least-square errors for all equations.

Let $e_k = 0$ be an equation from one tag/sensor pair. One can minimize $\frac{1}{2} \sum_k e_k^2$ for all tag/sensor pairs. We found the constrained optimization (`fmincon`

<p>Inputs: $\alpha_{ijl}, \beta_{ijl}$: angles from tag i to sensor j at cart position l</p> <p>Outputs: x_i, y_i, z_i: locations for all tags; x_l, y_l, z_l, a_l: positions of the mobile sensor frame;</p> <p>Notations: l: current sensor frame kTs: the set of tags with known locations cTs: the set of tags connected to the current sensor frame</p> <p>Initialization: $l \leftarrow 1$: the first sensor frame is the reference frame, $kTs \leftarrow \emptyset$: all tags are unknown position</p> <p>0. while there is a sensor frame l:</p> <p>1. Let cTs be the set of tags connected to frame l;</p> <p>2. Let $cnTs \leftarrow cTs \setminus kTs$ be the set of connected tags with unknown locations</p> <p>3. Compute tag locations for each tag in $cnTs$ using the closed-form solution in Section 3.1</p> <p>4. $kTs \leftarrow kTs \cup cnTs$</p> <p>5. Let l be the next unknown sensor frame with the maximum connections to known tags</p> <p>6. Compute the position of the sensor frame using the closed-form solution in Section 3.1</p> <p>7. end while</p>
--

Table 1. Leapfrog algorithm for tag localization using a mobile infrastructure.

in Matlab) works particularly well for this problem, where constraints are ranges for locations ($[-b, b]$ where $2b$ is the size of the x , y or z dimension) and angles $[-\pi, \pi]$.

We have tried two variations of this approach. The first variation (LSLeapfrog) is to apply the optimization procedure at each leapfrog step, i.e., at each step, from sensors to tags or from tags to sensors, using the closed-form solution first, and then apply the least-square minimization with the closed-form solution as the initial value. The second variation (LeapfrogLS) uses the original leapfrog solution as the initial value for least-square minimization of all equations.

In the next section, we compare the performance of these three algorithms (Leapfrog, LSLeapfrog, LeapfrogLS) with two different tag configurations and analyze their robustness in the presence of noise.

4 Simulation Results

In this section, we analyze the localization performance in two scenarios: “Wall,” in which tags are put on four walls of a room, and “Hallway,” in which tags are distributed along two walls of a narrow hallway. A total of 12 tags are used, in an area bounded by $[-150, 150]$ inches¹. Figure 4 shows the wall and hallway scenarios.

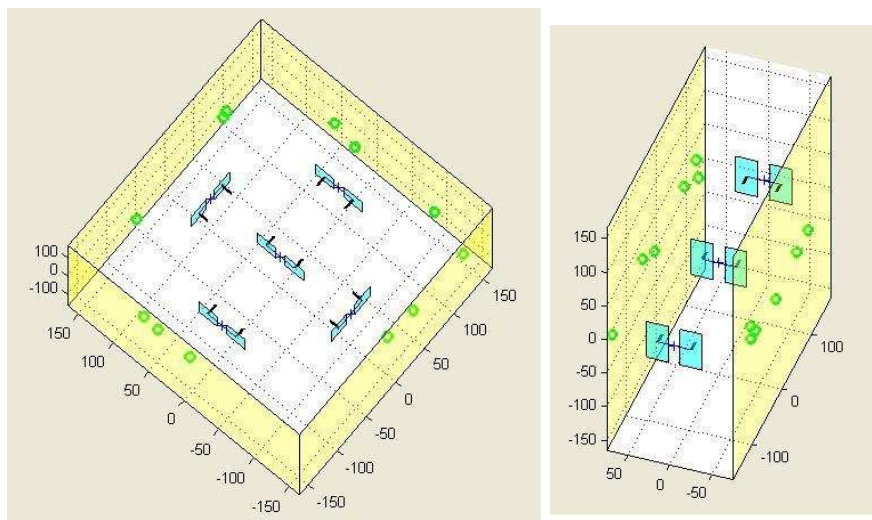


Fig. 4. The “Wall” scenario (left) and “Hallway” scenario (right).

¹ In this paper, distances are measured in inches (1 in = 2.54 cm) and angles are measured in radians (1 radian = $180/\pi$ degrees)

In our simulations, we use sensors separated by either 25 inches or 40 inches, with additive sensor noise modeled by a uniform distribution² in $[-m, m]$ where m is either 0.01 or 0.02 radians, a value based on the measurements of Section 5.1. The algorithm itself does not require an explicit noise model; this is for performance analysis only. The AOA sensing range of the sensor is $[-1.2, 1.2]$ radians for yaw angles and $[-1.0, 1.0]$ radians for pitch angles, the limits of the actual Ubisense hardware.

We then compare the three algorithms of the previous section are compared using *average estimation accuracy*. The estimation accuracy for one tag is the distance between the true and estimated location; the *average estimation accuracy* across all tags is used as the overall performance metric of localization.

Figure 5 shows samples of the 2D projections onto the XY plane for these two cases (where the noise is 0.02 radians and the distance between two sensors is 40 inches) resulting from LeapfrogLS; the lines from the tags and sensors indicate the displacements between the actual and estimated locations.

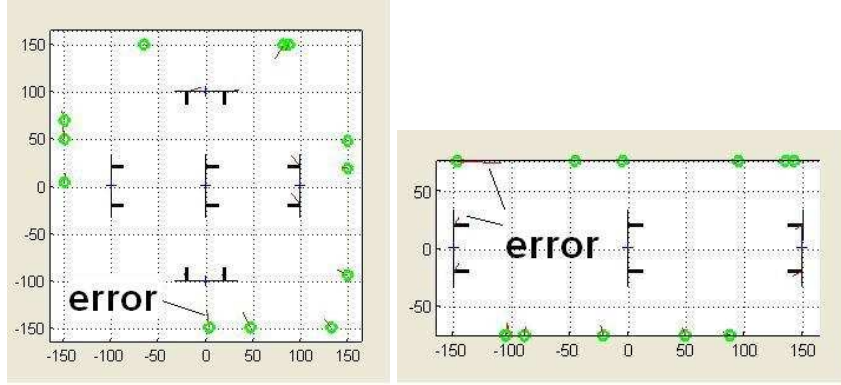


Fig. 5. localization results, left: wall tags, right: hallway tags.

For the wall case, each of the four walls has three tags placed in random positions, and for the hallway case, each side has six tags in random positions. In both cases, the heights are uniformly randomly distributed in $[-24, 24]$ inches. For the wall case, there are five cart positions, with the first position in the middle of the room. In the hallway case, there are three cart positions, with the first one in the middle of the hallway. Note that we use fixed, rather than random, cart positions to reduce run-to-run variations since cart positions can affect the localization results significantly. We generate 30 random cases for each selection of sensor separation and noise range, and run the three algorithms on each of the 30 cases. Location accuracies of 90% confidence level for the wall case is shown in Table 2, where $d; m$ indicates the distance between the sensors d in

² In other simulations, normal distributions gave similar results.

inches and angle noise range m in radians. Each entry x represents at least 27 out of 30 runs (i.e. 90%) have error less than x inches. We can see that in all cases, LeapfrogLS is better than LSLeapfrog, which is in turn better than Leapfrog. The distance between the sensor pair also matters: the larger the distance, the smaller the error.

algorithms	25 in; 0.01	40 in; 0.01	25 in; 0.02	40 in; 0.02
Leapfrog	107 in	68 in	143 in	95 in
LSLeapfrog	54 in	31 in	67 in	58 in
LeapfrogLS	15 in	11 in	39 in	15 in

Table 2. 90% confidence level for all algorithms on wall case.

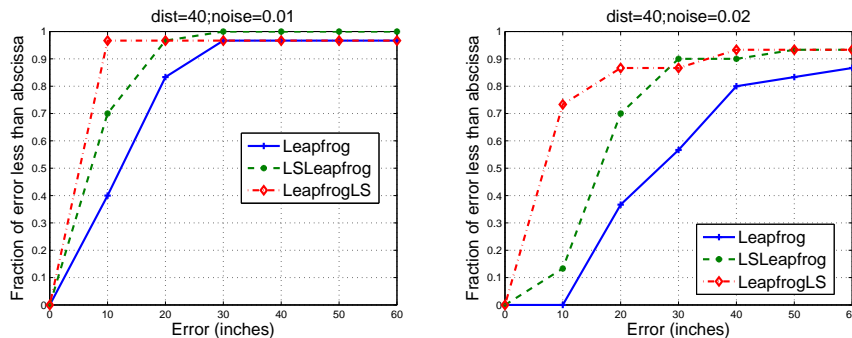


Fig. 6. Error distribution for hallway case with distance 40in, left: noise 0.01, right: noise 0.02.

Figure 6 shows the cumulative error distribution function for the hallway case, with distance between the sensor pair 40 inches, and noise of angles 0.01 and 0.02 radians, respectively. Again, we see that LeapfrogLS is better than LSLeapfrog, which is in turn better than Leapfrog, and the larger the angle noise, the bigger the error.

5 Real Experiments

We tested our algorithms using the Ubisense Location System. Figure 7 left shows a prototype of the system: two Ubisensors are mounted vertically on the poles with distance 40 inches. Our initial test had eight tags on two walls, and the space between neighboring tags was about 24 inches (Fig. 7 right).

The simulated and real experiments differed in their source of input data. For simulated cases, data were generated given the tag sensor positions and the



Fig. 7. Left: the mobile cart with two Ubisensors, right: 8 Tags on two walls.

noise model. For real experiments, data were generated from continuous AOA sensor readings during operation. In order to get a set of good data inputs corresponding to a set of cart positions, we moved the cart to multiple locations and stopped at each location for 5 to 10 seconds to get stable angle readings. The next section describes how to get a clean set of angle data for each location of the cart.

5.1 Data Preprocessing

In order to understand the noise characteristics of Ubisense data, we performed a series of experiments. The results from these experiments guided us to develop a robust data extraction algorithm from a continuous source of input data.

We first studied horizontal and vertical angle variations given a static pair of a sensor and a tag. We put a sensor and a tag in a fixed position, and measured AOA data in a given time period. Although details depended on the relative positions of the sensor and the tag, we found that error distributions with respect to the mean were very similar. Figure 8 shows the histogram from one data set. The standard deviations are between 0.01 and 0.03 radians for both yaw and pitch angles in our experiments. The distribution seemed not to be affected by distance or angle, although angles approaching the boundary of the valid range (about ± 1.2 radians in yaw and ± 1.0 radians in pitch) sometimes did result in readings with large variations. We filtered out data at large angles for robustness. To reduce variations in angle readings, we also averaged multiple data points for a stable position.

The raw AOA input to our algorithm is a continuous series of data points, with each entry: $timeslot, sensor, tagID, \alpha, \beta$, where $timeslot$ is the time slot in which the data is taken (one slot is about 1/40 second), $sensor$ and $tagID$ indicate from which pair of Ubisensor and Ubitag the data were obtained, and α and β are the pitch and yaw angles, respectively. Although in a stable position the angle variations are small, when the sensors are moving, readings are generally

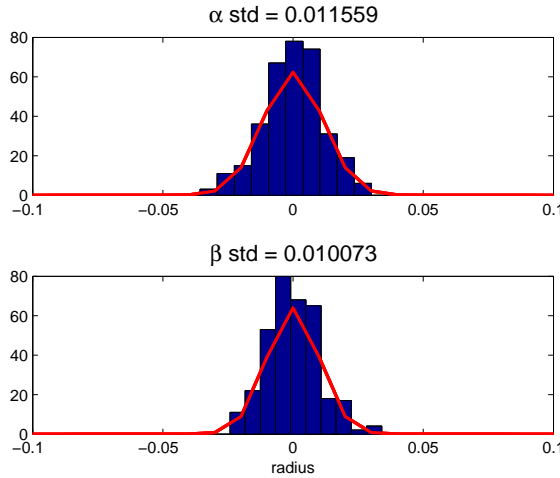


Fig. 8. Histogram of error distribution with respect to mean for a fixed sensor/tag. Errors are measured in radians.

not reliable. In our system, we attempt to use only data from *stationary* points, where the data are relatively stable.

First, we segment the input data so that consecutive data points with both α and β variations less than ϵ are grouped. Only groups with the number of data points more than n are kept. In our experiments, we set $\epsilon = 0.05$ and $n = 5$. Each group has a starting time slot s and ending time slot e , and an average α and β in the period of $[s, e]$. Figure 9 shows the original and grouped data points for a sensor/tag pair. Note that, as seen in the figure, this process also removes some isolated bad data. After grouping data points for each tag/sensor pair, the next step is to find a common set of data points which are consistent for all tag/sensor pairs. Such a set of points corresponds to the set of stationary positions in the trajectory of the mobile sensor platform. Figure 10 illustrates this process for two data sources. Let the start and end time of a period be represented as a left and right parenthesis, respectively. We first order all the parentheses along the time line and then search for periods with balanced left and right parenthesis. Note that it is possible that not all sources have data in a given time period.

5.2 Experiment Results

Figure 11 shows 2D and 3D views of the estimated locations of the 8 tags in Fig. 7 (right) using LeapfrogLS. To get a sense of the error in these experiments, we use the mean square error (MSE) of the model fitting error. Note that this is not the error from the ground truth, but merely the residual error remaining in equations 1 and 2 once the algorithm terminates.

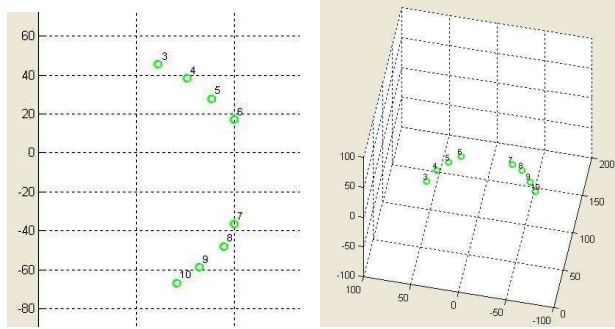


Fig. 11. Results of 8-tag localization (units in inches).

Let $e_k = 0$ be an equation from one tag/sensor pair. We use $\sqrt{\frac{1}{2n} \sum_{k=1}^n e_k^2}$, where n is the number of equations, for error estimates for real experiments. For the above example, the MSE of LeapfrogLS is 2.7 inches, and the MSE for LSLeapfrog and Leapfrog are 4.7 inches and 11 inches, respectively. In general, we find that the smaller the MSE, the smaller the position error. In cases where more than one solution exists, one may get a small MSE with large position error. However, we can usually avoid such cases by moving to a sufficient number of positions and taking enough data to guarantee a unique solution.

6 Conclusions and Future Work

We have presented an algorithm, and simulation and empirical studies of our approach to localizing static tags using mobile infrastructure to cover an area far larger than the sensor range. We show how a closed-form solution can be used to yield approximate results, and to what extent further optimizations improve these results when applied iteratively (LSLeapfrog) and when applied globally (LeapfrogLS) (e.g., location accuracy of 1 ft. (0.3 m) vs. 3 ft. (1 m) or 5 ft. (1.5 m)).

Note that although errors accumulate in the leapfrog procedure, the global optimization for the solution refinement minimizes the overall errors. In describing our experiments, we have used the MSE of the data fitting error rather than the MSE of the deviation from ground truth. Future experiments will measure both the accuracy of our localization and its growth as we use the mobile infrastructure over increasingly large areas.

One advantage of our approach is that no computer control of cart movement is needed; the natural movement of the cart during an installation process should be sufficient, although a few extra stops might be needed. Our algorithm automatically filters out bad cart positions in which too few tags are in view. In general, the more good cart positions, the more data, and the better the results. Compared to the Ubisense Location System with static sensors, our system achieves similar results as a fully “untethered” (i.e. without TDOA

measurements) configuration, which has an accuracy of about 0.4 m with 90% confidence.

One way to extend this work would be to incorporate other kinds of motion data, using inertial sensors or odometry. A cart with an inertial or wheel sensor could provide an alternative way to know when the cart was stationary. A wheel-mounted odometric distance measurement could also be combined with the tag readings to improve the overall system accuracy. Our system's performance would also improve if the extra TDOA data from the Ubisense hardware were used to augment the AOA data, or if greater numbers of sensors could be used to achieve better connectivity.

Although the approach presented in this paper used two sensors on the cart, it is theoretically possible to localize static tags with only a single AOA sensor, if distance data of the mobile sensor are available. Such a system would be smaller and perhaps could be integrated into a portable device such as a cellphone with an embedded accelerometer and would enable a greater set of applications. One example would be for home assistant robots. A robot with an AOA sensor could localize a set of tags, and at the same time, localize itself using the static tags as landmarks. On the other hand, rather than using only a single sensor, a system with more than two sensors could have a larger detection field, could collect data in fewer steps, and could allow for greater tag connectivity.

A greater set of applications would also be enabled by allowing some tags to be mobile, provided that enough tags remain stationary when the mobile infrastructure is moved. Furthermore, it should be possible to reuse tags as the process proceeds. Alternatively, localized tags might be left behind as a persistent localization infrastructure, allowing a single mobile sensor to be localized anywhere in the space. These classes of system can provide generalized precise positioning when it matters, without the cost of a permanently installed infrastructure.

Acknowledgement

We would like to thank Dr. Patrick Cheung for helping to build the mobile sensor infrastructure. We also like to thank the regional sales and technical support personnel from Ubisense for providing valuable information and assistance.

References

1. MERL project on UWB ranging and locating. <http://www.merl.com/projects/uwbranging/>.
2. Multispectral solutions, inc. <http://www.multispectral.com/>.
3. Time domain. <http://www.timedomain.com/>.
4. Ubisense precise real-time location. <http://www.ubisense.net>.
5. P. Bahl and V.N. Padmanabhan. An in-building RF-based location and tracking system. In *IEEE InfoComm*, 2000.
6. J.J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison Wesley, 1989.

7. A. Genco. Three step BlueTooth positioning. In *Location- and Context-Awareness, 2005*, Lecture Notes in Computer Science 3479, 2005.
8. K. Kaemarungsi and P. Krishnamurthy. Properties of indoor received signal strength for WLAN location fingerprinting. In *IEEE MobiQuitous*, 2004.
9. Anthony LaMarca, Yatin Chawathe, Sunny Consolvo, Jeffrey Hightower, Ian Smith, James Scott, Timothy Sohn, James Howard, Jeff Hughes, Fred Potter, Jason Tabert, Pauline Powledge, Gaetano Borriello, and Bill Schilit. Place lab: Device positioning using radio beacons in the wild. In *Proceedings of Pervasive, 2005*.
10. Julie Letchner, Dieter Fox, and Anthony LaMarca. Large-scale localization from wireless signal strength. In *Proceedings of the National Conference on Artificial Intelligence (AAAI 2005)*, 2005.
11. H. Lim, L.C. Kung, J. C. Hou, and H. Luo. Zero-configuration: Robust indoor localization: Theory and experimentation. In *IEEE InfoComm*, 2006.
12. J. Liu, Y. Zhang, and F. Zhao. Robust distributed node localization with error management. In *ACM MobiHoc 2006*, 2006.
13. M. Montemerlo and S. Thrun. *The FastSLAM Algorithm for Simultaneous Localization and Mapping*. Springer Tracts in Advanced Robotics, forthcoming, 2007.
14. H. L. Muller, M. McCarthy, and C. Randell. Particle filters for position sensing with asynchronous ultrasonic beacons. In *Location- and Context-Awareness, 2006*, Lecture Notes in Computer Science 3987, 2006.
15. V. Otsason, A. Varshavsky, A. LaMarca, and E. de Lara. Accurate GSM indoor localization. In *UbiComp 2005*, 2005.
16. S. N. Parel, K. N. Truong, and G. D. Abowd. Powerline positioning: A practical sub-room-level indoor location system for domestic use. In *UbiComp 2006, LNCS 4206*, 2006.
17. P. N. Pathirana, N. Bulusu, A. Savkin, and S. Jha. Node localization using mobile robots in delay-tolerant sensor networks. *IEEE Transactions on Mobile Computing*, 4(4), July/August 2005.
18. N. B. Priyantha, H. Balakrishnan, E. D. Demaine, and S. Teller. Mobile-assisted localization in wireless sensor networks. In *IEEE Conference on Computer Communications (InfoCom05)*, 2005.
19. N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *Proc. of the Sixth Annual ACM International Conference on Mobile Computing and Networking (MOBICOM)*, 2000.
20. M. Rabinowitz and James Spilker Jr. A new positioning system using television synchronization signals. <http://www.rosun.com/>.
21. C. Taylor, A. Rahimi, and J. Bachrach. Simultaneous localization, calibration and tracking in an ad hoc sensor network. In *5th International Conference on Information Processing in Sensor Networks (IPSN06)*, 2006.
22. C. Wang, Y. Ding, and L. Xiao. Virtual ruler: Mobile beacon based distance measurements for indoor sensor localization. In *The Third International Conference on Mobile Ad-hoc and Sensor Systems (MASS06)*, 2006.
23. Andy Ward, Alan Jones, and Andy Hopper. A new location technique for the active office. *IEEE Personal Communications*, 4(5):42 – 47, October 1997.
24. C. Wu, W. Sheng, and Y. Zhang. Mobile sensor networks self localization based on multi-dimensional scaling. In *IEEE ICRA07*, 2007.
25. Y. Zhang, M. Yim, L. Ackerson, D. Duff, and C. Eldershaw. Stam: A system of tracking and mapping in real environments. *IEEE Wireless Communications*, Decemeber 2004. <http://www.parc.com/era/nest/STAM>.