

Controlling Error Propagation in Mobile-Infrastructure Based Localization

Ying Zhang and Juan Liu

Palo Alto Research Center Inc.
3333 Coyote Hill Rd
Palo Alto, CA 94304, USA
Emails: {yzhang,jjliu}@parc.com

Abstract. Many iterative localization schemes suffer from the negative effect of error propagation, where sensor noise results in estimation errors which then get accumulated and amplified over localization iterations. This paper extends our earlier work in mobile-infrastructure based localization and proposes a computationally efficient error control mechanism to mitigate the error propagation effect. In particular, we show how the error can be characterized and controlled. Simulation results have shown that our error control mechanism achieves comparable location accuracy as global optimization-based localization methods and has the advantage of being much more computationally efficient and easily distributable.

Keywords: Detection and Tracking, Localization, Error Control

1 Introduction and Motivation

It is important to have the capability to localize objects when a Global Positioning System (GPS) is unavailable (e.g., in an indoor environment) or inaccurate (e.g., in a city center surrounded by high-rise buildings). Much research has been done in localization [1–13], focusing on developing algorithms to estimate location information based on sensor measurements. Many algorithms are iterative bootstrapping in nature: given nodes with known locations, estimate the location for a set of unknown (free) nodes; the estimated nodes are then used to localize others. While this approach is computationally efficient and easily distributable, it often suffers from the negative effect of error propagation. Noise in sensor measurements can get amplified due to sensing geometry, and the estimation error can propagate and accumulate during the iterations. This result in poor localization accuracy.

The problem of error propagation and accumulation has been noted in localization literature [14] [15] [16]. Unlike most existing approaches which use certain geometry-related heuristics, our prior work [17] introduces a novel systematic approach, which formalizes the heuristics and performs a quantitative analysis of error characteristics. The basic idea is to document each location estimate with a quality measure and be discretionary on which data measurement

to use in localization. In this paper, we extend the error control mechanism in [17] to mobile-infrastructure based localization [18]. We will investigate where localization error comes from, and how it can propagate from nodes to nodes. We will explain in details how the error control mechanism is devised to manage information with various degrees of reliability. Simulation results show that the error control significantly improves the localization quality, reducing average localization error by a factor of 3 or more, yet it is still computationally efficient, requiring much less computation than optimization-based refinements developed in [18].

The other related work here is mobile-assisted localization (MAL). The basic idea is to allow the nodes to move to cover space so that a signal node can act like a set of “virtual nodes” to provide more constraints to localization. Most existing MAL-like work [19–23] assumes homogeneous installation, where the mobile nodes are identical to static nodes, and all nodes can both transmit and receive. The mobile-infrastructure-based localization in [18] is different in the sense that the mobile nodes are receivers, powerful in computation capability, while the static nodes are transmitters and do not require any computation.

This paper is organized as follows. Section 2 presents the overview of mobile-infrastructure-based localization developed in [18]. Section 3 analyzes errors from the mobile-infrastructure based localization and describes the error control method which selects neighbors based on location uncertainty. Section 4 evaluates the algorithms given different configuration and noise parameters in simulation. Section 5 concludes the paper and suggests future directions.

2 Localization with Mobile Infrastructure

In our prior work [18], we have proposed the use of “mobile infrastructure”, which uses angle-of-arrival sensors from Ubisense [24]. Two types of nodes are used: ultra-wide band (UWB) transmitters (known as “Ubitags”) and UWB receivers (known as “Ubisensors”). Ubisensors determine the angle-of-arrival (AOA) of UWB signals emitted from the Ubitags. Inherently the Ubisensors are more complex than the Ubitags and far more expensive (approximately 40 times in price). The Ubisense system is designed to be a general-purpose location infrastructure. In its intended use, the Ubisensors are permanently installed in a room, and their relative positions are carefully measured and manually input by a skilled technician. Ubitags can be static or moving. The problem with this intended use is the fact that Ubisensor’s high price prohibits large scale deployment. To solve this problem, we have proposed the framework of “mobile infrastructure” to make the system more cost effective by taking advantage of the cost asymmetry between Ubisensors and Ubitags. The mobile infrastructure consists of a pair of Ubisensors mounted on a mobile cart. In this case, the Ubisensors are considered as infrastructure nodes, moving from place to place, hence the name “mobile infrastructure”. In contrast, the Ubitags are deployed in larger quantity, and are considered as ad hoc nodes. Note that the roles of Ubisensors and Ubitags are reversed from the original intended use. This role reversal is cost effective if the

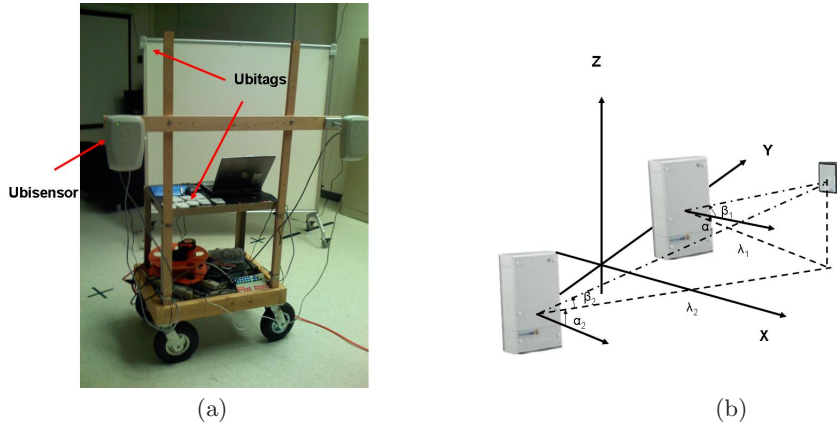


Fig. 1. (a) Mobile platform and its (b) two sensor frame.

unknown locations are widely spread out, spanning multiple sensing ranges, or if the number of mobile objects is much smaller than the number of static objects with unknown locations.

The mobile infrastructure constitutes a sensor frame with the two AOA sensors in a fixed distance (Fig. 1(b)). Although AOA sensors has six degrees of freedom in general, two of them are fixed in this frame (the pitch and yaw angles are fixed to 0). The sensor frame has four degrees of freedom in space: (x_s, y_s, z_s, a_s) , where x_s, y_s, z_s are 3D coordinates and a_s is the yaw angle. The location of a *sensor frame* is defined to be the center between two sensors. Without loss of generality, we assume initially the sensor frame is at $(0,0,0)$ and facing $a_s = 0$. The localization problem in this case is to estimate tag locations and the locations (center and yaw) of all subsequent sensor frames based on the AOA measurements. In particular, the measurements are the yaw and pitch angles: $\{\alpha(i, j, l), \beta(i, j, l)\}$, where the index set (i, j, l) denotes the measurement from sensor j to tag i at the l -th sensor frame location. In the absence of noise, the measurement satisfies the following geometry:

$$\begin{cases} (x_i - x_j) \sin(a_j + \alpha_{ij}) = (y_i - y_j) \cos(a_j + \alpha_{ij}) \\ (x_i - x_j) \sin \beta_{ij} = (z_i - z_j) \cos \beta_{ij} \cos(a_j + \alpha_{ij}) \end{cases} \quad (1)$$

where (x_i, y_i, z_i) is tag location, (x_j, y_j, z_j, a_j) is sensor location and α_{ij}, β_{ij} are AOA readings.

For this platform, we have developed a *leapfrog* procedure [18]. It is iterative, starting from the initial sensor frame at $(0,0,0)$, it alternates between computation of tag locations and sensor frame locations: from sensors with known location, it estimates tags with unknown locations; and from known tags, it estimates sensor frames with unknown locations. This alternation continues until all the tag locations and frame locations are obtained. In particular, after locating the tags using the current sensor frame, it then selects as the next frame the

sensor frame connected to the maximum number of known tags. It computes the location of this new frame, and proceeds iteratively. In the rest of this paper, we call a frame or a tag *known*, if its location is obtained already, and *free* if its location is to be computed. Here we briefly describe the localization computation without much explanation. Interested readers may refer to [18] for details.

From Sensors to Tags Given the location of a sensor frame, x, y, z, a , we would like to estimate the tag locations (x_t, y_t, z_t) from the angle measurements $\{\alpha_k, \beta_k\}_{k=1,2}$.

For a tag seen by a sensor frame, let λ_1 and λ_2 be the distances from the tag to the two sensors in the XY plane. From Fig. 1(b), we have the following geometry:

$$\begin{bmatrix} -\sin(\alpha_1) & \sin(\alpha_2) \\ \cos(\alpha_1) & -\cos(\alpha_2) \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} d \\ 0 \end{bmatrix}. \quad (2)$$

Here d is the distance between the two sensors in the sensor frame, which is fixed and known a priori. We solve for λ_1 and λ_2 . The tag locations are estimated as:

$$\begin{bmatrix} x_t \\ y_t \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \cos(a + \alpha_1) & \cos(a + \alpha_2) \\ \sin(a + \alpha_1) & \sin(a + \alpha_2) \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix}. \quad (3)$$

$$z_t = z + \frac{1}{2}(\lambda_1 \tan(\beta_1) + \lambda_2 \tan(\beta_2)). \quad (4)$$

From Tags to Sensors If a sensor frame sees multiple known tags, the location of the sensor frame x, y, z, a can be obtained. Let α_{ik} and β_{ik} be yaw and pitch angles from sensor k ($k = 1$ or 2) to tag i , respectively, and let λ_{ik} be the projected distance between tag i and sensor k on the XY plane, which can be computed by (2). It is easy to verify

$$\begin{aligned} (x_i - x_j) \cos(a) + (y_i - y_j) \sin(a) &= \lambda_{ik} \cos(\alpha_{ik}) - \lambda_{jk} \cos(\alpha_{jk}) \\ (x_i - x_j) \sin(a) - (y_i - y_j) \cos(a) &= -\lambda_{ik} \sin(\alpha_{ik}) + \lambda_{jk} \sin(\alpha_{jk}) \end{aligned}$$

All the λ 's are known from Eq.(2). We use an approximate method to solve for a by treating the above as two linear equations of the variables $\cos(a)$ and $\sin(a)$, solving for $\cos(a)$ and $\sin(a)$. Therefore $a = \arctan(\frac{\sin(a)}{\cos(a)})$.

The sensor frame location, i.e., the midpoint between two sensors, computed using tag i is estimated as:

$$\begin{bmatrix} x_s^i \\ y_s^i \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} - \frac{1}{2} \begin{bmatrix} \cos(\alpha_{i1} + a) & \cos(\alpha_{i2} + a) \\ \sin(\alpha_{i1} + a) & \sin(\alpha_{i2} + a) \end{bmatrix} \begin{bmatrix} \lambda_{i1} \\ \lambda_{i2} \end{bmatrix}. \quad (5)$$

$$z_s^i = z_i - \frac{1}{2}(\lambda_{i1} \tan(\beta_{i1}) + \lambda_{i2} \tan(\beta_{i2})). \quad (6)$$

For n known tags seen by the frame, the result is the mean of all estimates.

One serious problem of this algorithm is that errors in localization of the previous tags (or frames) will propagate into the localization of the future frames (or

tags), accumulating over time. The problem of error propagation and accumulation can be controlled to some degree by optimization-based refinements during or after the leapfrog procedure. It has been shown in [18] that the refinements improve the location accuracy significantly. The problem with optimization-based methods, however, is the computational complexity. It loses the effectiveness for large scale problems, or when the number of variables becomes large. In the rest of this paper, we will focus on how to efficiently mitigate the error propagation and improve localization accuracy.

3 Error Analysis and Error Control

Due to perturbation in sensor observations, localization error may occur. Measurement noise can have an effect on the localization accuracy, and the severity is decided by geometry. For example, if a free node is localized using AOA measurements from known nodes that are far away, then error in angle measurements results in large offset in space. Another pathological case is that the unknown node is localized with a set of nodes that are all collinear with itself. In this case, the location estimate has unbounded error. If this localized node is then used as a known node to localize other nodes, the location error will further affect the accuracy of other nodes. The error could accumulate over localization iterations, and the localization performance can be bad especially in large scale networks.

Our prior work [17] advocates the discrimination of nodes based on a quantitative characterization of their uncertainty. The basic idea is follows: when a node is localized with respect to its connected known nodes, not all these nodes are equal; certain node may have more reliable location information than others. It is hence preferable to use only reliable known nodes to avoid error propagation. This can be done by keeping track of uncertainty in every step of localization. Any time a location estimate is obtained, the companion estimation for the uncertainty of the location is performed, which formally analyzes how much error is generated in this localization step. This is known as error characterization.

In the rest of this section, we extend this framework to the **Leapfrog** localization procedure and show how uncertainties are characterized and used to mitigate error propagation.

3.1 Leapfrog with Error Control

We defer the problem of error estimation to the next section (Sec. 3.2) and start with a brief overview of the general structure of our error control mechanism. The main idea is to rank nodes (sensor frames and tags) based on the uncertainty of their location information, denoted as e_i for each node i . Only nodes with low uncertainty are used to localize others. This prevents error from propagating. Furthermore, the location update is conditional: a newly estimated location is used to update an old estimate only when the former is believed to have lower uncertainty. This conditional update criterion prevents error from accumulating.

Leapfrog with error control **LeapErrCon** consists of the following components:

(i) *First frame selection*: The first frame serves as reference frame; its location and direction defines the coordinate system. It is also the starting point of the **Leapfrog** procedure. In this first frame, we hope to initialize a substantial number of tags so that later iterations can be “jump-started”. Hence we favor choosing a sensor frame with a large amount of connections to tags.

(ii) *Node selection for localization*: For **Leapfrog**, only one frame (the current one) is used for tag localization and all connected tags are used for frame localization. **LeapErrCon** selects a subset of connected known nodes for localization in both cases. The selection is based on predicted errors $\{e_i\}$. **LeapErrCon** ranks nodes with increasing uncertainty e_i and selects the first k , where $3 * \text{mean}(\{e_i | i \leq k\}) < e_{k+1}$. This is an empirical criterion which seems to work well.

(iii) *Node update*: In **LeapErrCon**, a tag is localized every time a connected frame is localized. For an unknown tag, both the location and its uncertainty are registered. For a known tag, replace the old registration if the new estimation error is smaller. We further use a forgetting factor update criterion, i.e, $\mathbf{x}_t = (x, y)$, $\mathbf{x}_t = \gamma \mathbf{x}_t^{new} + (1 - \gamma) \mathbf{x}_t^{old}$. This partial update ensures that a noisy measurement will not cause big drift in the location estimate.

(iv) *Next frame selection*: In general, we prefer to first localize tags and sensors with moderate estimation error, and defer noisier estimation problem to later iterations. **LeapErrCon** estimates the total location error for all candidates of the next frame, and selects the one with minimum error.

The pseudo-code is shown in Table 1. Unlike **Leapfrog**, more than one sensor frames may be used to localize a tag and a tag may be localized many times, as long as its location uncertainty decreases.

3.2 Error Estimation for the Leapfrog Procedure

The localization task is to obtain an estimate $(\hat{x}, \hat{y}, \hat{z})$ for any free node (tag or sensor) as a function of other known node locations $\{(x_i, y_i, z_i)\}$ and measurements $\{\alpha_i, \beta_i\}$. Localization error hence comes from two sources: the uncertainty in neighbor positions $\{(x_i, y_i, z_i)\}$ and the uncertainty in each measurement $\{\alpha_i, \beta_i\}$. Assuming we know all the statistics of $\{(x_i, y_i, z_i)\}$ and $\{\alpha_i, \beta_i\}$, can we derive a statistical characterization of the estimate $(\hat{x}, \hat{y}, \hat{z})$? Error characterization attempts to answer this question. In the **Leapfrog** process, error characterization is recursive: the free node derives error characteristics based on neighboring locations and measurement noise. In the next round, this node is used to localize others, hence its error becomes one of the uncertainty sources for the other nodes. Despite the simple formulation, exact error characterization is difficult. We use the same grossly simplifying assumptions as in [17], assuming all noises are Gaussian and the computations are approximately linear. Under these assumptions, error characterization becomes feasible for practical use.

In this section, we analyze error generated from the **Leapfrog** procedure: (a) localizing a tag using known sensor frames (Eq.(3)), and (b) localizing a sensor frame using known tag locations (Eq.(5)). The mobile-infrastructure cart moves in a horizontal plane. The z -dimension is estimated after (\hat{x}, \hat{y}) are obtained.

<p>Inputs: $\alpha_{ijl}, \beta_{ijl}$: angles from tag i to sensor j at frame l</p> <p>Outputs: x_i, y_i, z_i: locations for all tags; x_l, y_l, z_l, a_l: locations of sensor frames;</p> <p>Notations: kTs: the set of tags with known locations cTs: the set of tags connected to the current known sensor frame</p> <p>Initialization: $l \leftarrow$ <i>the first sensor frame selected using step (i)</i>, $kTs \leftarrow \emptyset$: no known tags</p> <p>0. while there is a new known sensor frame l: 1. Let cTs be the set of tags connected to frame l; 2. For each tag in cTs: 2.a <i>select connected known frames using step (ii)</i> 2.b compute the location of the tag 2.c compute error estimation of the tag using Eq. (7) 2.d <i>update both tag location and uncertainty using step (iii)</i> 3. $kTs \leftarrow kTs \cup cTs$ 4. <i>Let l be the next free sensor frame selected by step (iv)</i> 5.a <i>select connected known tags using step (ii)</i> 5.b compute the location of the frame 5.c compute error estimation of the frame using Eq. (7) 5.d <i>update both frame location and uncertainty using step (iii)</i> 6. end while</p>
--

Table 1. Leapfrog with error control LeapErrCon; error control parts are in italic.

Hence from the error-propagation perspective, the error in \hat{z} will not affect error in (\hat{x}, \hat{y}) . For simplicity, we only characterize error in 2D rather than in 3D. Likewise, the pitch angle β is only used to estimate \hat{z} . Hence we ignore this error and only model the error in the yaw angles $\{\alpha\}$.

First, the measurement angle uncertainty in α results in uncertainties in (λ_1, λ_2) (Eq.(2)). Note that λ_1 and λ_2 are the distance from a tag to a pair of sensors, and the solution to Eq.(2) is

$$\begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = \frac{1}{\sin(\alpha_2 - \alpha_1)} \begin{bmatrix} d \cos(\alpha_2) \\ d \cos(\alpha_1) \end{bmatrix}.$$

The uncertainty due to α_1 and α_2 can be characterized using Taylor expansion, e.g., $d \cos(\alpha_1 + \Delta\alpha_1) = d \cos(\alpha_1) - d \sin(\alpha_1) \Delta\alpha_1$. Assuming that α_1 and α_2 has roughly the same amount of noise, and ignoring error in $\alpha_1 - \alpha_2$, we have the following:

$$\text{cov}(\lambda_1, \lambda_2) = \frac{d^2 \sigma^2}{\sin^2(\alpha_2 - \alpha_1)} \begin{bmatrix} \sin^2(\alpha_2) & 0 \\ 0 & \sin^2(\alpha_1) \end{bmatrix}$$

where σ^2 is measurement variance of α_1 and α_2 . It gives the quantitative classification of the error propagation due to the geometry of the sensor frame and

the tag. It is clear that if $\alpha_1 - \alpha_2$ is close to 0, uncertainty from measurements of α to λ is amplified. Intuitively the location estimate error will be big when the two sensors and the tag are collinear, or if the tag is very far away from the sensor frame.

Secondly, the distance pair (λ_1, λ_2) is then used to estimate the tag or sensor position (Eq.(3) or Eq. (5)) via a linear transform B . To characterize the contribution of this term to the overall localization uncertainty, we use the covariance:

$$\Omega^\lambda = B \cdot cov(\lambda_1, \lambda_2) \cdot B'$$

where

$$B = \frac{1}{2} \begin{bmatrix} \cos(\alpha_1 + a) & \cos(\alpha_2 + a) \\ \sin(\alpha_1 + a) & \sin(\alpha_2 + a) \end{bmatrix}.$$

This is assuming that the linear transform B is error-free, hence only the contribution from the distance pair needs to be counted. However, the uncertainty in α_1 and α_2 also directly contributes to B . To take that into account, we now assume that the distance pair (λ_1, λ_2) is error-free and only count the contribution of α_1 and α_2 . Note that from Eq.(3) and Eq.(5), we have the location estimate being the sum of three terms: 1. $(x, y)'$, 2. α_1 's contribution $\frac{\lambda_1}{2}(\cos(\alpha_1+a), \sin(\alpha_1+a))'$, and 3. α_2 's contribution $\frac{\lambda_2}{2}(\cos(\alpha_2+a), \sin(\alpha_2+a))'$. To compute the contribution of these individual terms, we again use Taylor expansion and compute the respective covariance:

$$\begin{aligned} \Omega^{B_1} &= \frac{\lambda_1^2 \sigma^2}{4} \begin{bmatrix} \sin^2(\alpha_1 + a) & 0 \\ 0 & \cos^2(\alpha_1 + a) \end{bmatrix}, \\ \Omega^{B_2} &= \frac{\lambda_2^2 \sigma^2}{4} \begin{bmatrix} \sin^2(\alpha_2 + a) & 0 \\ 0 & \cos^2(\alpha_2 + a) \end{bmatrix}. \end{aligned}$$

where Ω^{B_1} and Ω^{B_2} are the contribution from α_1 and α_2 respectively. Let Ω^v be the location uncertainty of the known node v used in location computation, the *location uncertainty* due to both location and measurement errors from the known node is estimated by

$$\Omega = \Omega^v + \Omega^\lambda + \Omega^{B_1} + \Omega^{B_2}. \quad (7)$$

For a set of known nodes used for this computation, the uncertainty is set to be the average of the uncertainty computed from each participated node.

Every time a node location is estimated or refined, this error characterization step is performed. The covariance matrix Ω is computed and stored in the node registry of the newly localized node. It is compact, only of size 2×2 , and is updated via light-weight computation. When examining nodes in the neighborhood, we use the trace $\omega = trace(\Omega)$ as the quality measure. Larger ω means weaker confidence in the node position, and hence the node should be used more discretely.

4 Simulation Experiments

In this section, we study the effectiveness of the error control method with respect to optimization-based refinements, using two simulated scenarios.

4.1 Simulation Scenarios

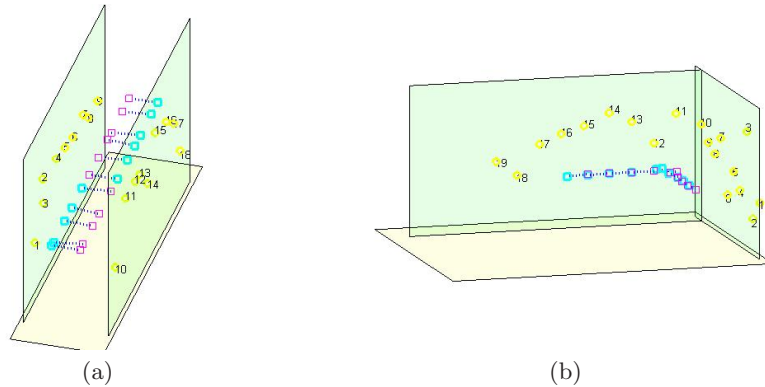


Fig. 2. Two test scenarios: (1) Hallway and (2) Walls; tags are circles and sensors are squares. Each sensor frame is shown by two squares connected by a dotted line, with the left sensor bold.

We study two scenarios that are most common in indoor localization:

1. *Long Hall* (Fig. 2(a)): tags are distributed on two sides of a long hallway. The mobile platform is moving from one end to the other end, always facing center of the hallway.
2. *Long Walls* (Fig. 2(b)): tags are distributed along two neighboring walls of a room. The mobile platform is moving from one wall to the other, always facing the wall.

The only input scale of the system is the distance d between the two sensors, which is set to 1 without loss of generality ¹.

Other constants for simulation are defined as follows. The width of the hallway is 3. The horizontal distances between the neighboring tags or frames are random with standard deviation 0.2. The orientation of the frame has a standard deviation of 0.1 radians. In all cases, heights of tags are random with zero mean

¹ Note that we don't have any unit for distances here since it is all relative to what unit we use for the distance between the two sensors. For example, if the distance between the two sensors are d units, all the results will be scaled by d units.

and standard deviation 1, and the height of the frame is fixed to be zero². AOA data are modeled according to the Ubisense system [24]: angle errors are zero mean, with ranges of yaw (α) and pitch (β) angles in $[-1.2, 1.2]$ and $[-1.0, 1.0]$ radians, respectively. Since Ubisensors can see up to 100 meters in line of sight, we assume no distance constraints for sensing.

4.2 Performance Metrics

One of the performance metrics for localization is location accuracy. *Location accuracy* for tag localization is the displacement or error between the actual and estimated tag locations. The error from one tag is the distance between the actual and estimated tag $\rho \triangleq |\mathbf{x} - \hat{\mathbf{x}}|$. For n tags with errors $\{\rho_i\}_{i=1, \dots, n}$, there are various ways to aggregate the errors, for example, using mean ($\frac{1}{n} \sum_i \rho_i$), mean square ($\frac{1}{n} \sqrt{\sum_i \rho_i^2}$), or maximum ($\max_i \rho_i$) values. All of these metrics ignore error distributions which are important for many applications. In this paper, we advocate accuracy with 90% confidence, i.e., sorting $\{\rho_i\}_{i=1, \dots, n}$ in ascending order, and use the $k = \lceil 0.9n \rceil$ entry value as the accuracy metric.

For each scenario, we generate a total of N random cases, and again use the accuracy of 90% confidence for representing the accuracy of localization in that scenario. In the rest of this paper, location accuracy or error means location accuracy with 90% confidence.

Another important performance metric is the computation time of obtaining the tag locations. Localization with error control takes more time, but it is a lot more efficient than optimization-based methods.

4.3 Algorithms for Comparisons

There are four variations of the **Leapfrog** procedure, defined by two flags: (1) error control flag (EC), and (2) refinement flag (RF). When EC is off, **Leapfrog** with no error control is used, and when EC is on, **LeapErrCon** (Table 1) is used. When RF is on, optimization-based refinements [18] are applied:

- **Leapfrog**: The **Leapfrog** procedure [18] without refinements.
- **LeapErrCon**: The **Leapfrog** procedure applying the error control mechanism (Table 1).
- **Leapfrog-RF**: **Leapfrog** with refinement, but no error control.
- **LeapErrCon-RF**: **LeapErrCon** with refinement.

4.4 Simulation Results

In this section, we show some simulation results using the four variations of the leapfrog algorithm on the two indoor scenarios. In particular, we are interested in

² Note that although the algorithm works for varying frame heights, we only simulate frames with fixed height since the mobile platform we have (Fig. 1(a)) can only move on a plane.

comparing error control mechanisms with optimization-based refinements, from small to large number of nodes, and at different density of tag/cart distributions. In all cases, we generate 30 random data sets and compare location accuracy and localization time using the four variations of the leapfrog procedure. All simulations were run on a PC laptop (Dell Latitude D400, Intel(R) Pentium(R), 1.7GHz, and 1GB RAM) with code written in Matlab.

Hallway We tested the hallway case using six sets of data with the hallway length scales from 10, 20 to 30, and standard deviations of the AOA noise (σ) of 0.01 or 0.02 radians. Tags were placed liberally on the wall, with an average spacing of 1 (i.e. the same distance as the spacing between the two sensors). The mobile platform (equivalently the sensor frame) was moved along the hallway, stopped randomly with a mean distance of 1 as well, to make measurements.

For 30 random data sets of each case, we also study the error distributions among different variations of the leapfrog algorithm. Fig. 3 shows the cumulative

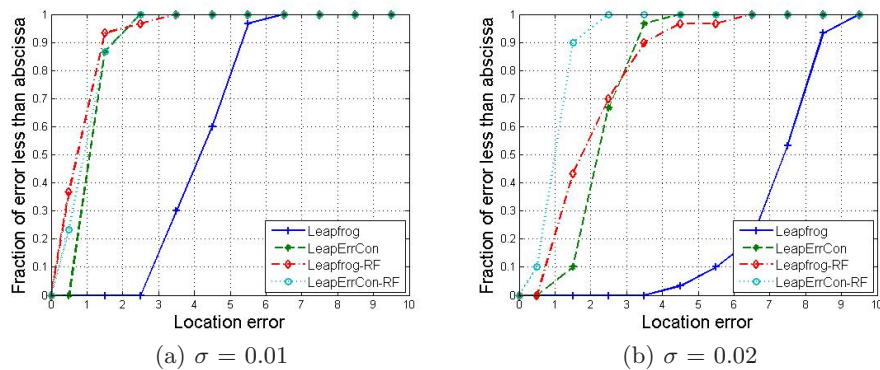


Fig. 3. Cumulative error distribution of the 30 random cases, with hallway length 20, noise 0.01 and 0.02 radians, respectively.

error distribution of the hallway case with hallway length 20 for measurement noise σ of (a) 0.01 and (b) 0.02 radians. We see that in both cases, **Leapfrog** does not work well. When the measurement noise is small, all variations except the basic **Leapfrog** work well, when the noise increases, **LeapErrCon-RF** works the best.

Fig. 4 shows both the localization time and accuracy for the hallway scenario for measurement noise 0.02 radians. From the result, we can make conclusions that (1) the basic **Leapfrog** procedure does not work well in this case, (2) **LeapErrCon** improves the accuracy significantly, with much less computation time than refinements, (3) **LeapErrCon** works better than refinements for large

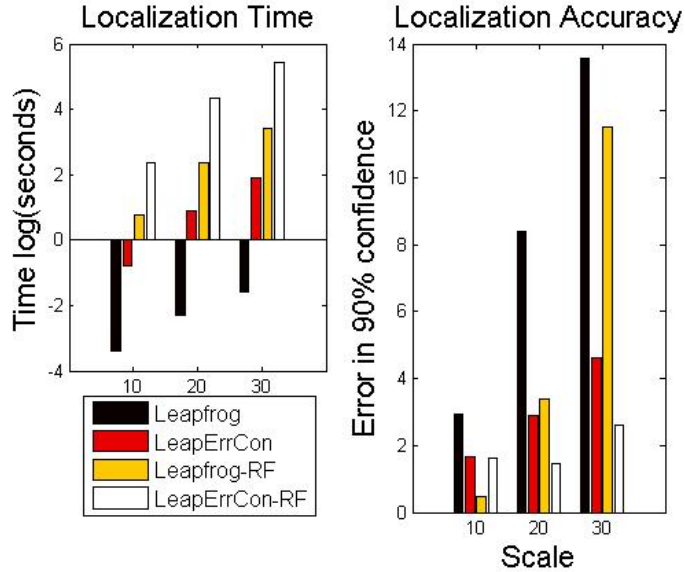


Fig. 4. Performance comparisons: localization time and accuracy. Note that time is in log scale.

scales, and (4) LeapErrCon-RF has the best accuracy as expected, however, it takes much longer to obtain a result, and the improvement is only marginal.

The results demonstrate the effectiveness of LeapErrCon. In addition, note that LeapErrCon’s computation only involves a neighborhood of nodes. It is computationally efficient and easily distributable.

Walls For this case, we study the effect of the density of tag distributions. Like in the hallway case, the tags and sensor frames were spaced with an average distance of 1, 2 or 3. The sensor frames (i.e., the mobile platform) are slightly far away from the wall, with a distance to the wall set to 1.5 times of the distance of the mean spacing between tags, so that the frame at any point can see enough overlapping tags. The location error due to measurement noise would be larger as the tag distances increase. In simulation, we set measurement noise σ to 0.02, and wall length being 10, 20, and 30, for tag spacing 1, 2, and 3, respectively. Fig. 4.4 shows one test case for the wall scenario, using two variations of the algorithms: LeapErrCon and Leapfrog-RF. We see that LeapErrCon works better.

5 Conclusion and Future Work

We have presented an error control mechanism for the leapfrog procedure to localize tags using mobile infrastructure. Simulation showed that the error control

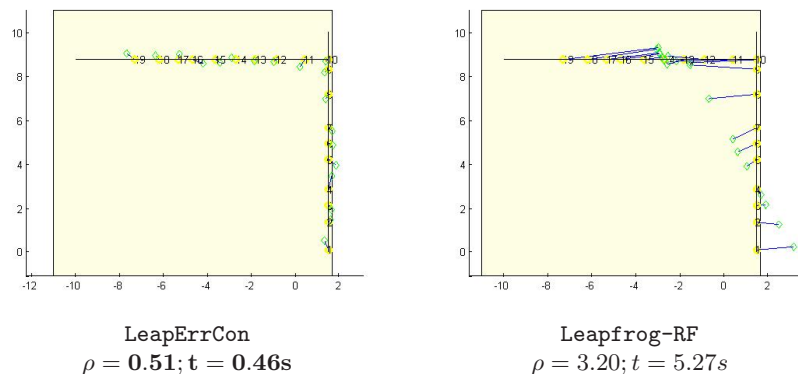


Fig. 5. Walls tag localization shown in 2D plane. Wall length 10 and AOA variance 0.02. Location accuracy and localization time are shown in ρ and t , respectively for each case. Circles are ground truth and diamonds are estimated locations. Best localization accuracy is shown in bold.

scheme significantly improves the localization accuracy and achieves comparable performance as global optimization-based method at only a fraction of the computation cost.

One way to extend this work is to incorporate other kinds of motion data, using inertial sensors or odometry. A wheel-mounted odometric distance measurement could also be combined with the Ubisensor readings to improve the overall system accuracy. Such a system could be smaller with only one sensor. A more interesting case is to have a Ubisensor integrated with a portable device such as a cellphone with an embedded accelerometer, such devices would enable a greater set of applications. Error control would be even more important in these cases so as to use the data resulting small error propagation.

References

1. Shang, Y., Ruml, W., Zhang, Y., Fromherz, M.P.: Localization from connectivity in sensor networks. *IEEE Trans. on Parallel and Distributed Systems* **15**(11) (November 2004) 961 – 974
2. Doherty, L., Pister, K.S.J., Ghaoui, L.E.: Convex position estimation in wireless sensor networks. In: *Proceedings of IEEE Infocom*. Volume 3. (April 2001) 1655–1663
3. Biswas, P., Ye, Y.: Semidefinite programming for ad hoc wireless sensor network localization. submitted to IPSN 2004
4. Bulusu, N., Heidemann, J., Estrin, D.: Gps-less low cost outdoor localization for very small devices. *IEEE Personal Communications Magazine* **7**(5) (October 2000) 28–34
5. Savarese, C., Rabaey, J., Langendoen, K.: Robust positioning algorithms for distributed ad-hoc wireless sensor networks. In: *USENIX Technical Annual conference*, Monterey, CA (June 2002)

6. Kleinrock, L., Silvester, J.: Optimum transmission radii for packet radio networks or why six is a magic number. In: Proc. IEEE National Telecommunications Conference. (1978) 4.3.1–4.3.5
7. Costa, J., Patwari, N., Hero, A.: Distributed weighted-multidimensional scaling for node localization in sensor networks. *ACM Transactions on Sensor Networks* **2**(1) (February 2006)
8. Patwari, N., Ash, J.N., Hero, A.O., Moses, R., Correal, N.S.: Locating the nodes. *IEEE Signal Processing Magazine* **54** (July 2005)
9. LaMarca, A., Chawathe, Y., Consolvo, S., Hightower, J., Smith, I., Scott, J., Sohn, T., Howard, J., Hughes, J., Potter, F., Tabert, J., Powledge, P., Borriello, G., Schilit, B.: Place lab: Device positioning using radio beacons in the wild. In: *Proceedings of Pervasive*. (2005)
10. Letchner, J., Fox, D., LaMarca, A.: Large-scale localization from wireless signal strength. In: *Proceedings of the National Conference on Artificial Intelligence (AAAI 2005)*. (2005)
11. Lim, H., Kung, L., Hou, J.C., Luo, H.: Zero-configuration: Robust indoor localization: Theory and experimentation. In: *IEEE InfoComm*. (2006)
12. Bahl, P., Padmanabhan, V.: An in-building RF-based location and tracking system. In: *IEEE InfoComm*. (2000)
13. Kaemarungsi, K., Krishnamurthy, P.: Properties of indoor received signal strength for WLAN location fingerprinting. In: *IEEE MobiQuitous*. (2004)
14. Savvides, A., Park, H., Srivastava, M.B.: The n-hop multilateration primitive for node localization problems. *Mobile Networks and Applications* **8**(4) (2003) 443–451
15. Niculescu, D., Nath, B.: Ad hoc positioning system (APS). In: *GLOBECOM* (1). (2001) 2926–2931
16. Moore, D., Leonard, J., Rus, D., Teller, S.: Robust distributed network localization with noisy range measurements. In: *Sensys*. (November 2004)
17. Liu, J., Zhang, Y., Zhao, F.: Robust distributed node localization with error control. In: *ACM MobiHoc06*. (2006)
18. Zhang, Y., Partridge, K., Reich, J.: Localizing tags using mobile infrastructure. In: *Location- and Context- Awareness*. (2007) *Lecture Notes in Computer Science*, 4718.
19. Pathirana, P.N., Bulusu, N., Savkin, A., Jha, S.: Node localization using mobile robots in delay-tolerant sensor networks. *IEEE Transactions on Mobile Computing* **4**(4) (July/August 2005)
20. Priyantha, N.B., Balakrishnan, H., Demaine, E.D., Teller, S.: Mobile-assisted localization in wireless sensor networks. In: *IEEE Conference on Computer Communications (InfoCom05)*. (2005)
21. Taylor, C., Rahimi, A., Bachrach, J.: Simultaneous localization, calibration and tracking in an ad hoc sensor network. In: *5th International Conference on Information Processing in Sensor Networks (IPSN06)*. (2006)
22. Wang, C., Ding, Y., Xiao, L.: 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. Wu, C., Sheng, W., Zhang, Y.: Mobile sensor networks self localization based on multi-dimensional scaling. In: *IEEE ICRA07*. (2007)
24. Ubisense: Ubisense precise real-time location <http://www.ubisense.net>.