Research Article

Error Control in Distributed Node Self-Localization

Juan Liu and Ying Zhang

Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA 94304, USA

Correspondence should be addressed to Juan Liu, jjliu@parc.com

Location information of nodes in an ad hoc sensor network is essential to many tasks such as routing, cooperative sensing, and service delivery. Distributed node self-localization is lightweight and requires little communication overhead, but often suffers from the adverse effects of error propagation. Unlike other localization papers which focus on designing elaborate localization algorithms, this paper takes a different perspective, focusing on the error propagation problem, addressing questions such as where localization error comes from and how it propagates from node to node. To prevent error from propagating and accumulating, we develop an error-control mechanism based on characterization of node uncertainties and discrimination between neighboring nodes. The error-control mechanism uses only local knowledge and is fully decentralized. Simulation results have shown that the active selection strategy significantly mitigates the effect of error propagation for both range and directional sensors. It greatly improves localization accuracy and robustness.

1. INTRODUCTION

In ad hoc networks, location information is critical to many tasks such as georouting, data centric storage, spatio-temporal information dissemination, and collaborative signal processing. When global positioning system (GPS) is not available (e.g., for indoor applications) or not accurate and reliable enough, it is important to develop local positioning system (LPS). Recent years have seen intense research on this topic [1]. One approach for LPS is to use fingerprinting that requires extensive preparatory manual surveying and calibration [2–6] and is not reliable in terms of dynamic changes of the environment. The other approach is for devices to *self-localize* by collectively determining their positions relative to each other using distance [7] or directional [8, 9] sensing information. Our research is focused on self-localization.

Node self-localization techniques can be classified into two categories: centralized algorithms based on global optimization, and distributed algorithms using local information with minimal communication. While the first category methods are powerful and produce good results, they require substantial communication and computation, and hence may not be amendable to in-network computation of location information on resource-constrained networks such as sensor networks. In this paper, we focus on the second cat-

egory, *distributed* node self-localization. Various distributed node localization techniques have been proposed in the sensor network literature [10, 11]. The basic idea is to decompose a global joint estimation problem into smaller subproblems, which only involve local information and computation. Then localization iterates over the subproblems [12–16]. This approach greatly reduces computational complexity and communication overhead.

However, one problem with distributed localization is that it often suffers from the adverse effects of error propagation and accumulation. As a node being localized and becoming new anchor for other free nodes, the estimation error in the first node's location can propagate to other nodes and potentially get amplified. The error could accumulate over localization iterations, and this may lead to unbounded error in localization for large networks. The effect of error propagation may also occur in global methods such as MDS [17] or SDP [18, 19], but is less prominent, due to the fact that global constraints tend to balance against each other.

Although the error characteristics of localization have been studied in literature [20, 21], the problem of error control has not received adequate attention. Our early work [22] is the first paper presenting the idea of using a node registry to formally characterize error in iterations and choose neighbors selectively in localization to filter out outlier estimates

(bad seeds) that may otherwise contaminate the entire network. In this paper, we extend the early work and present a more general error-control mechanism, applicable to a variety of sensing modalities, such as range sensors (time-of-arrival (TOA) or received-signal-strength (RSS)) and directional sensors (camera, microphone array, etc). The error control consists of three components: (1) *error characterization* to document node location with uncertainty; (2) a *neighbor selection* step to screen out unreliable neighbors— it is preferable to only use nodes with low uncertainty to localize others; this prevents error propagating to other nodes and contaminating the entire network; (3) an *update criterion* that rejects a location estimate if its uncertainty is too high; this cuts the link of error accumulation. This error-control mechanism is lightweight, and only uses local knowledge.

Although we will be presenting localization algorithms in later sections, for example, the iterative least-squares (ILS) algorithm for range-based localization and the geometric ray intersection and mirror reflection algorithms for direction-based localization, we would like to point out that the focus of this paper is not on any particular localization algorithm, but rather on controlling error in order to mitigate the effect of error propagation. It is a simple fact that all localization schemes are imperfect and result in some error. Most work in the localization literature focuses on elaborate design of localization algorithms and fine-tuning to produce small localization error. In this paper, we take a different perspective by addressing questions such as where localization error comes from, how it can propagate from node to node, and how to control it. We explain in details how the error-control mechanism is devised to manage information with various degree of uncertainty. Our method has been tested in simulations. Results have shown that the error-control mechanism is powerful in mitigating the effect of error propagation. It significantly improves localization performance and speeds up convergence in iterations. For range-based localization, despite the fact that the underlying localization (ILS) is very simple, it achieves performance comparable to and in many cases better than that of more global methods such as MDS-MAP [17] or SDP [19]. Similar improvements have been observed in our early experiments on a small network of Mica2 motes with ultrasound time-of-arrival ranging [22]. For directional-based localization, we show that the error-control method outperforms the basic localization mechanism [8], reducing localization error by a factor of 3-4. Experiments on a real platform using the Ubisense real-time location system [23] will be conducted in the near future.

This paper is organized as follows. Section 2 presents the overview of the distributed node self-localization. Section 3 describes the error-control mechanism. Sections 4 and 5 apply this mechanism to range-based and angle-based node localization, respectively. Section 6 concludes the paper.

2. DISTRIBUTED LOCALIZATION: AN OVERVIEW

Most localization approaches assume that a small number of anchor nodes know their location a priori and then progressively localize other nodes with respect to the anchors. Anchorless localization is also feasible for some algorithms, such as building relative maps using MDS-MAP [17] or forming rigid structures between nodes as described in [24]. It is possible to develop error control for anchorless localization, but this requires a more elaborate mechanism which remains our future research. In this paper, we assume the existence of a small set of anchor nodes.

In general, localization is to derive unknown node locations $\{\mathbf{x}_t\}_{t=1,\dots,N}$ based on a set of sensor measurements $\{z_{t,i}\}$ and anchor node locations. Each measurement provides a constraint on the relative position between a pair of sensors. In this paper, we consider the two most commonly used sensor types: range sensors and directional sensors. Both types have a large variety of commercially available products. A range sensor provides distance information between nodes, typically derived from sensing of physical signals such as acoustic, ultrasonic, or RF transmitted from one node to another. Distance can be derived from time-of-arrival (TOA) measuring time of flight between the sender and the receiver, or via received signal strength (RSS) following a model of signal attenuation. A directional sensor measures the relative direction from one node to another, that is, $(\mathbf{x}_i - \mathbf{x}_t)/\|\mathbf{x}_i - \mathbf{x}_t\|$. There are ample examples of directional sensors: cameras [25], microphone arrays with beamforming capability, UWB positioning hardware such as the Ubisense product [http://www.ubisense.net]. Without loss of generality, we consider localizations in a 2D plane. Most of the technical points illustrated in 2D can readily be extended to 3D.

Distributed node localization is iterative in nature. We use multilateration-type localization [12] as a vehicle for illustration. Initially, only anchor nodes are aware of their locations. A free node is localized by incorporating sensor measurements from anchors in its local neighborhood $\mathcal{N}$. In the case of range sensors, a free node can be localized if it can sense at least 3 nodes with known locations. The newly localized free nodes are then used as "pseudoanchors" to localize other neighboring free nodes. Here neighbors are not topological neighbors in a communication network, but rather in a sensing network graph (SNG) defined as follows: vertexes are sensor nodes, and edges represent distance or angle constraints between pairs of nodes. Any pair of nodes that can reliably sense each other's signal (hence form sensor measurement $z_{t,i}$) are called mutual *immediate neighbors* in SNG. We assume that neighbors in SNG can communicate with each other, either directly or via some intermediate node; in most cases, communication ranges are larger than sensing ranges. Each iteration progressively pushes location information over edges of SNG, for example, from anchors to nearby free nodes, and from pseudoanchors to their neighbors. The iteration may terminate if node locations no longer change or if a computation allowance has been exhausted. Algorithm 1 shows the iterative procedure. Technically, one iteration means one complete sweep in the **do-while** loop. We assume that the nodes are updating their locations in a globally synchronous fashion, that is, each free node updates their location based on the information of its neighbors from the previous iteration. The updates can be done simultaneously across nodes. There are various research work on packet scheduling to avoid collision, for instance, using pre-assigned time-slot in TDMA. In this paper, we do not discuss

```
ITERATIVE LOCALIZATION
Each node i holds x, where
        x_i is the node location (or estimate);
        x_i = null if location is unknown
Free node to be localized is denoted by t
Each edge corresponds to a measurement z_{t,i};
do {
    for each free node t
        examine local neighborhood N;
        find all neighbors in N with known location
        compute location estimate x̂_t;
    } while termination condition is not met.
```

ALGORITHM 1: Iterations in distributed multilateration.

the packet scheduling problem. Note that the procedure in Algorithm 1 does not rely on a collision-free communication assumption. If a free node does not receive enough information from its neighbors due to collision, it just skip the state of computing a location estimate and remains unknown.

3. ERROR CONTROL IN ITERATIVE LOCALIZATION

Distributed localization such as the procedure illustrated in Algorithm 1 often suffers from error propagation. Estimated node locations are not perfect. Their uncertainty may further influence neighboring nodes. Over iterations, the error may propagate to the entire network. Essentially, error propagation is caused by the strategy of using the estimated node locations as pseudoanchors to localize other nodes. While this strategy greatly reduces the amount of communication and computation required and is more scalable, it also introduces potential degradation in localization quality. The optimization strategy is analogous to a coordinate descent algorithm, which, at any step, fixes all but one coordinate (node location in this case), finds the best solution along the flexible axis, and iterates over all axes. Just as a coordinate descent algorithm may have slow convergence and get stuck at ridges or local optima, this node localization strategy suffers from similar problems. Figure 1 shows a typical run where localization gets stuck. Moreover, the strategy may be slow to converge which means high communication and computation overhead. Even global optimization schemes are not completely immune to error propagation. For example, the relaxation method of [19] introduces the possibility of error propagation. The existence of error propagation is inherently a by-product of the optimization strategies.

Various heuristics are proposed to mitigate the effect of error propagation. For example, [26] weights multilateration results with estimated relative confidence, and [7] discounts the effect of measurements from distant sensors based on the intuition that they are less reliable and may amplify noise. Recent work on cluster-based localization [24] selects spatially spread nodes to form quadrilaterals to minimize localization error. In this paper, rather than using heuristics, we seek to provide a formal analysis of localization error.
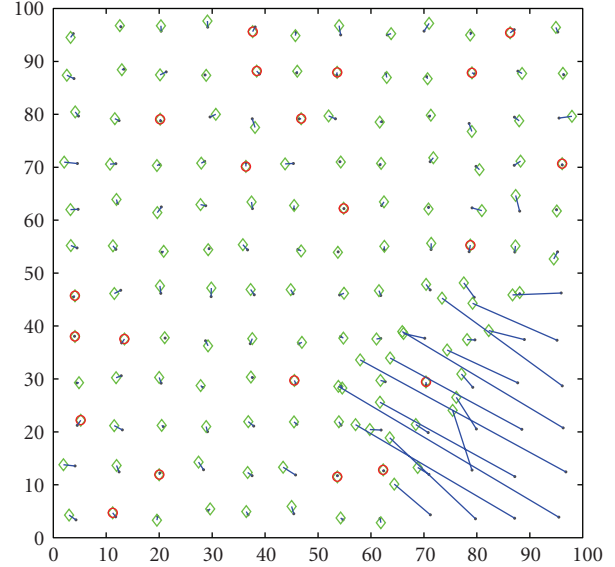

FIGURE 1: Localization gets stuck at a local optimum. Estimated node locations are marked with diamonds, true locations are plotted as dots, and solid lines show the displacement between the estimated locations and the ground truth (i.e., each line is the estimation error of a node). Anchors are marked with circles.

The basic idea of error control is simple: when a node is localized with respect to its neighbors, not all neighbors are equal. Certain neighbors may have more reliable location information than others. It is hence preferable to use only reliable neighbors to avoid error propagation. Based on this intuition, we propose an error-control method consisting of three components as follows.

(1) *Error characterization.* Each time we compute a location estimate, we perform the companion step of characterizing the uncertainty in the estimate. Each node maintains a registry that contains the tuple (location estimate, location error). It is useful in the next round for neighbor selection (see Algorithm 2).

(2) *Neighbor selection.* This step differentiates neighbors based on uncertainty in their respective node registries. Nodes with high uncertainty are excluded from the neighborhood and not used to localize others. This prevents errors from propagating.

(3) *Update criterion.* At each iteration, if a new estimated location has error larger than the current error or a predefined threshold, the new estimate is discarded. This conditional update criterion prevents error from generating.

Algorithm 2 shows the same iterative localization procedure as in Algorithm 1 but with error control. The error-control steps are marked in italic fonts. Note that for this error-control mechanism to work, the free node would have to know not only the location of its neighbors, but also their respective uncertainty e^v. From a practical implementation point of view, the uncertainty information can be piggy-backed on the same packet when the location information

ITERATIVE LOCALIZATION WITH ERROR CONTROL
Each node i holds the tuple $(\mathbf{x}, e^v)_i$, where
        $\mathbf{x}_i$ is the node location (or estimate) of neighbor i;
        e_i^v (vertex error) is the uncertainty in $\mathbf{x}_i$.
The free node to be localized is denoted as t
Each edge j corresponds to a tuple $(z, e^e)_{t,i}$, where
        $z_{t,i}$ is the sensor measurement regarding node t and neighbor i;
        $e_{t,i}^e$ (edge error) is the uncertainty in $z_{t,i}$
do {
    for each free node t
        examine local neighborhood $\mathcal{N}$;
        select neighbors based on vertex and edge errors $\{e^v\}$ and $\{e^e\}$
        compute location estimate $\hat{\mathbf{x}}_t$;
        estimate error $\hat{e}_t$;
        decide whether to update t's registry with the new tuple $(\hat{\mathbf{x}}_t, \hat{e}_t)$.
    } **while** termination condition is not met.

ALGORITHM 2: Distributed localization with error control. The error-control steps are shown in italic fonts.

is sent. The edge error is known from sensing characteristics and does not require additional communication.

In this section, we address the design principles of error control, and defer the detailed implementation to Sections 4 and 5.

3.1. *Error characterization*

The basic problem of localization is, for any free node t, given its neighbor locations $\{\mathbf{x}_i\}_{i \in \mathcal{N}}$ and the corresponding sensor measurements $\{z_{t,i}\}_{i \in \mathcal{N}}$, how to obtain an estimate

$$\hat{\mathbf{x}}_t = f(\{\mathbf{x}_i\}, \{z_{t,i}\}). \tag{1}$$

Localization error of a nonanchor node t comes from two sources.

(1) Uncertainty in each neighbor location $\mathbf{x}_i$. A neighbor may have imperfect information regarding its location, especially nonanchor nodes. We call this error "vertex error" (because a neighbor is a vertex in the SNG) and use the shorthand notation e^v.

(2) Uncertainty in each sensor measurement $z_{t,i}$. We call this "edge error" and use the notation e^e.

The error $\hat{e}_t$ in the location estimate $\hat{\mathbf{x}}_t$ is a function of both vertex and edge errors:

$$\hat{e}_t = g\left(\{e_i^v\}_{i \in \mathcal{N}}, \{e_{(t,i)}^e\}_{i \in \mathcal{N}}\right). \tag{2}$$

In this paper, we seek to find the proper form of $g(\cdot, \cdot)$ to characterize error. In the iterative localization process, the error characterization is recursive: the node derives error characteristics based on vertex and edge errors from its local region. In the next round, this node is used to localize others, hence its error $\hat{e}_t$ becomes the vertex error e^v for the neighboring nodes.

Despite the simple formulation, error characterization is difficult. Ideally, one would like to express uncertainty as

probability distribution, for example, $e_i^v = p(\mathbf{x}_i)$, and derive the exact form of $\hat{e}_t$. But anyone with even superficial knowledge on statistics will recognize the difficulty: it is extremely hard to derive a distribution of $f(a, b, c)$ from the distribution of its individual variables a, b, and c. The function f could be complicated, and the variable may be dependent, in which case we need the joint distribution $p(a, b, c)$. As localization progresses, the error characterization problem quickly becomes intractable. To overcome the difficulty, we make several grossly simplifying assumptions. First, we assume all variables are Gaussian, in which case the uncertainty can be characterized by a variance (for scalar) or a covariance matrix (for vectors). This reduces the form of e^v and e^e down from a probability distribution to only a few numbers. Secondly, we assume that the function f can be linearized with only mild degradation. This assumption is necessary because only when f is linear will the result $\hat{\mathbf{x}}_t$ (1) remain Gaussian. Thirdly, we assume that variables in (1) are independent, hence the covariance e_t will be the sum of the contribution from each variable $\mathbf{x}_i$'s and $z_{t,i}$'s. These simplifying assumptions enable us to carry forward error characterization with the progression of localization, and to differentiate node qualitatively. We recognize that these assumptions are sometimes inaccurate. In contrast, the exact quantitative differentiation is out of reach. Furthermore it may not be necessary since our goal is to rank neighbors and select a subset of good ones, hence any qualitative measure producing roughly the same order and the same subset should suffice.

In our scheme, each node t has a registry containing the tuple $(\mathbf{x}_t, e_t^v)$. We will illustrate in details how $\mathbf{x}_t$ and e_t^v are computed in localization using range sensors and directional sensors in later sections (Sections 4 and 5, resp.). We note that any location estimation step is followed by the companion step of computing the uncertainty of the location estimate. This effectively doubles the computation complexity in each iteration. Is it worthwhile? We will be addressing this question using simulation experiments. We would also

like to point out that although error characterization is designed to discriminate neighbors in localization iterations, it can also be used in follow-up tasks after localization. For example, tasks such as in-network signal processing need to know node locations, but the performance may be further enhanced if it also knows the rough accuracy of node locations. It can optimize with respect to this additional information, even when such information is qualitative.

3.2. Neighbor selection

Neighbor selection has been proposed in several papers such as [20, 21] to differentiate neighbors based on heuristics about the noise-amplifying effect of node geometry, or based on estimation bounds such as Cramer-Rao lower bound (CRLB). In this paper, we use formal error characterization (2) to prepare the ground for neighbor selection. As we will see in the later sections, geometry-like heuristics are often special cases that can be easily derived from our error characterization step. We do not use CRLB because it is often too loose. In our method, we select neighbors based on their vertex error and edge error. Vertex error is recorded in the node registry. Neighboring nodes can be sorted based on their respective registries. Edge error is the uncertainty in pairwise sensor measurements. This can be derived from sensing physics.

3.3. Update criterion

Even with "high-quality" neighbors and good sensor measurements with mild noise perturbation, the estimate could still be arbitrarily bad if the neighbors happen to be in some pathological configuration. For example, as we will see in Section 4, collinearity in neighbor positions greatly amplifies measurement noise and result in bad estimate. To address this problem, we propose an update criterion to reject bad estimates based on their quality. A few metrics can be used here: (1) the uncertainty $\hat{e}_t^v$: reject if it is too big; and (2) data fitting error: reject if the estimate does not agree with sensor observation data.

4. ERROR CONTROL IN LOCALIZATION USING RANGE SENSORS

A range sensor measures the distance from itself to another node, that is,

$$z_i = \|\mathbf{x} - \mathbf{x}_i\|. \tag{3}$$

Most localization schemes using range sensors are based on multilateration, using distance constraints to form rigid structures. If anchor density is low, we use an optional initialization stage. During the initialization, anchor nodes broadcast their location information. Each free node computes a shortest path in SNG to each of the nearby anchors, and use the path length as an approximation to the Euclidean distance. The shortest path can be computed locally and efficiently using Dijkstra's algorithm. Note that it is sufficient for a free node with shortest paths to 3~5 anchors to obtain an

initial location estimate. In this section, we first describe a simple least-squares (LS) algorithm for location estimation, then proceed to discuss the corresponding error characterization and error-control method.

4.1. Least-squares multilateration

Ignoring the measurement (edge) and neighbor location (vertex) errors for the moment, we square both sides of (3) and obtain

$$\|\mathbf{x}\|^2 + \|\mathbf{x}_i\|^2 - 2\mathbf{x}_i^T\mathbf{x} = z_i^2, \quad i = 0, 1, \ldots. \tag{4}$$

From $|\mathcal{N}|$ such quadratic constraints, we can derive $n = (|\mathcal{N}| - 1)$ linear constraints by subtracting the $i = 0$ constraint from the rest as follows:

$$-2(\mathbf{x}_i - \mathbf{x}_0)^T\mathbf{x} = (z_i^2 - z_0^2) - \left(\|\mathbf{x}_i\|^2 - \|\mathbf{x}_0\|^2 \right). \tag{5}$$

The $(i = 0)$th sensor is used as the "reference." Letting $\mathbf{a}_i = -2(\mathbf{x}_i - \mathbf{x}_0)$ and $b_i = (z_i^2 - z_0^2) - (\|\mathbf{x}_i\|^2 - \|\mathbf{x}_0\|^2)$, we simplify the above as

$$\mathbf{a}_i^T\mathbf{x} = b_i. \tag{6}$$

Here, $\mathbf{a}_i$ is a 2×1 vector, and b_i is a single scalar.

Thus, we have obtained n linear constraints, expressed in matrix form

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{7}$$

where $\mathbf{A} = (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n)^T$ and $\mathbf{b} = (b_1, b_2, \ldots, b_n)^T$. The least-squares solution to the linear system (7) is $\hat{\mathbf{x}}_t = \mathbf{A}^\dagger\mathbf{b}$, where $\mathbf{A}^\dagger$ is the pseudoinverse of $\mathbf{A}$, that is, $\mathbf{A}^\dagger = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$. In later text, we use the shorthand notation $I_A = (\mathbf{A}^T\mathbf{A})^{-1}$ when necessary. This linearization formulation is commonly used in localization; see [22, 26], for examples. The computation is lightweight, since $\mathbf{A}$ is only of size $n \times 2$, with n typically being small, and $\mathbf{b}$ is of size $n \times 1$.

4.2. Error characterization and control

In the formulation in (7), $\mathbf{b}$ captures the information about sensor measurements, and $\mathbf{A}$ encodes the geometric information about the sensor configuration. The accuracy of localization is thus influenced by these two factors. First, the error in the measurement z's (i.e., edge errors) will result in some uncertainty in $\mathbf{b}$. In particular, assume no vertex errors, that is, $\mathbf{A}$ is certain, the error due to $\mathbf{b}$ can be characterized as follows:

$$\text{Cov}(e_{\Delta\mathbf{b}}) = \text{Cov}(\mathbf{A}^\dagger\Delta\mathbf{b}) = \mathbf{A}^\dagger \cdot \text{Cov}(\Delta\mathbf{b}) \cdot (\mathbf{A}^\dagger)^T. \tag{8}$$

A pathological case is that nodes are collinear. In this case, $\mathbf{A}$ is singular, so is the pseudoinverse $\mathbf{A}^\dagger$. With a large condition number, $\mathbf{A}^\dagger$ greatly amplifies any slight perturbation in $\mathbf{b}$ (i.e., measurement noise). This is the case shown in Figure 2(a). The estimate, marked as a star, has a covariance which is a long ellipsoid. In contrast, Figure 2(b) shows a location estimation example where the neighbors are well
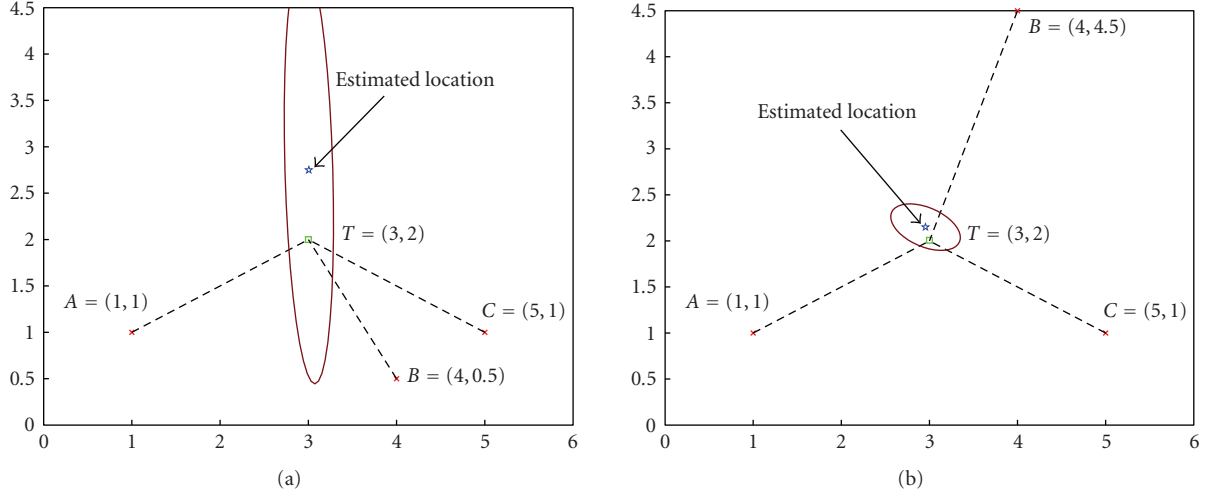
FIGURE 2: Estimation error for different neighbor geometry: (a) three almost collinear neighbors and (b) three neighbors forming a well-spaced triangle. The estimate covariance is plotted as an ellipsoid.

spaced. In this case, **A** is well-conditioned, and the result estimation error is small.

Secondly, we consider the noise in neighbor locations (vertex error). Note that we can reorganize elements in the **A** matrix into a long vector $\mathbf{a} = (a_{11}, a_{21}, \ldots, a_{n1}, a_{12}, a_{22}, \ldots, a_{n2})^T$, where the element $a_{ij} = -2(x_{ij} - x_{0,j})$ for $i = 1, \ldots, n$ and $j = 1, 2$ (n is the number of equations in (6)). If the error statistics of $\mathbf{x}_i$'s are known, we can estimate the statistics of Δa_{ij} as well. Let matrix

$$\mathbf{B} \triangleq \begin{pmatrix} b_1 & b_2 & \cdots & b_n & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & b_1 & b_2 & \cdots & b_n \end{pmatrix}. \qquad (9)$$

It is easy to verify that $\mathbf{A}^T\mathbf{b} = \mathbf{Ba}$. Using this notation, the original estimate $\mathbf{x}_t = \mathbf{I_A}\mathbf{A}^T\mathbf{b}$ can be written as $\mathbf{x}_t = \mathbf{I_A}\mathbf{Ba}$. The error due to $\mathbf{a}$ is

$$\text{Cov}(e_{\Delta\mathbf{a}}) = \text{Cov}(\mathbf{I_A}\mathbf{B}\Delta\mathbf{a}) = \mathbf{I_A}\mathbf{B}\text{Cov}(\Delta\mathbf{a})\mathbf{B}^T\mathbf{I_A}^T. \qquad (10)$$

The total error is the summation of the two terms listed above. The overall analysis provides a way of evaluating (2) from edge and vertex errors. Note that the computation of error only involves multiplication of small matrices: **A** is of size $n \times 2$, **B** is of size $2 \times 2n$, $\mathbf{I_A}$ is of size 2×2, and the covariances $\text{Cov}(\Delta\mathbf{a})$ and $\text{Cov}(\Delta\mathbf{b})$ are of size $2n \times 2n$ and $n \times n$, respectively. No matrix inversion is involved.

With closed-form easy-to-evaluate error characterization, error control becomes simple. The neighbor selection step determines among the neighbors with known locations whose measurement to use and whose to discard. We use a simple heuristic. For any node $i \in \mathcal{N}(t)$, we sum up the vertex error e^v and the edge error e^e for the edge between t and i, that is, we compute a total score

$$e_{\text{total}}(i) = e_i^v + e_{(i,t)}^e. \qquad (11)$$

The nodes with lower sum are considered preferable. The summation form of total error (11) is merely heuristic, but makes intuitive sense: for any given node i, if it is used to localize others, its location error e_i^v will add uncertainty to the localized result $\hat{\mathbf{x}}_t$; furthermore, the measurement error $e_{(i,t)}^e$ will cause $\hat{\mathbf{x}}_t$ to drift around by roughly the same amount. This is not exact though, because the final localization result depends not only on node i, but on the geometry of all selected neighbors. The exact error should be evaluated with all neighbor combinations ($2^{|\mathcal{N}|}$ combinations altogether). Note that the goal of neighbor selection is not to find the optimal combination of nodes, but rather to filter out outlier nodes with bad quality. Hence, we retreat to this simple heuristics which has linear complexity $o(|\mathcal{N}|)$.

In our implementation, the node with the lowest sum is used as $\mathbf{x}_0$. The neighbor selection is done by ranking the $e_{\text{total}}(i)$'s in an ascending order, picking the first three nodes, and setting a threshold that is 3σ above the third lowest e_{total} value, where σ is the standard deviation of edge errors. Nodes with the error sum below the threshold are selected. The nodes that are 3σ above are considered outliers, and excluded from the neighborhood. The 3σ threshold value is empirical, which seems to work well in practice. The update criterion examines the new estimate $(\hat{\mathbf{x}}_t, \hat{e}_t)$ tuple, and rejects it if the error $\hat{e}_t$ is larger than a predefined threshold.

4.3. Simulation experiment

The localization algorithm described above has been validated in simulations. A network is simulated in a 100 m × 100 m field, with 160 nodes placed randomly according to a uniform distribution. Each node has a sensing range of 20 m, which is 1/5 of the total field width. Anchor nodes are randomly chosen. The standard deviation of anchor nodes is 0.5 m in horizontal and vertical directions. Distance measurements are simulated with Gaussian noise with zero mean
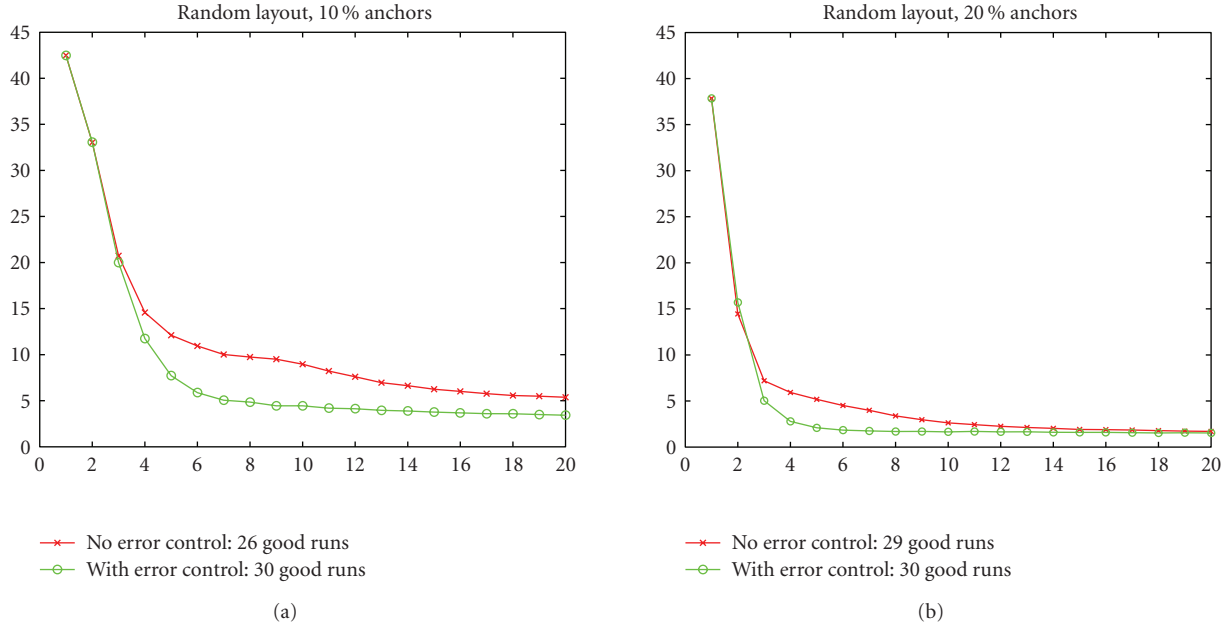
FIGURE 3: Localization accuracy for random network layout with (a) top panel: 10%, and (b) bottom panel: 20% randomly placed anchor nodes. The horizontal axis is the number of iterations; the vertical axis is the average distance between location estimates and ground truth, measured in meters.

and a variance of $1.5\,\mathrm{m}^2$. Since it is simulation, and the ground truth is known, we use the location error $\zeta_{\mathbf{x}} = \sum_t |\hat{\mathbf{x}}_t - \mathbf{x}_t|/N$ measuring the average deviation from ground truth as the performance metric.

To study the effect of error control, we compare localization performance with and without error control. The shortest path initialization is not used in this experiment because the anchor percentage is relatively high. The scheme with error control actively selects from its neighborhood which measurements to use and which to reject, using the error estimation described in Section 4.2. For each scheme, 30 independent runs are simulated in a network of TOA sensors with 10% and 20% anchor nodes, respectively. We consider a run "lost" if the localization scheme produces a larger error than that of randomly selecting a point in the network layout as the estimate; otherwise, we consider it a good run. With error control, all 30 runs are good. In contrast, the scheme without error control loses a few runs: 4 lost runs with 10% anchor nodes and 1 lost run with 20% anchors.

Figure 3(a) reports the localization accuracy (with 10% anchor nodes) at the beginning of each iteration, with accuracy measured as location errors. The accuracy results are plotted as circles and crosses, for localization with and without error control, respectively. The first few iterations produce large localization error; this is because only a fraction of the nodes is localized. After 4-5 iterations, almost all nodes are localized, and after that the nodes iterate to refine their location estimate. The figure clearly indicates that the error control strategy improves localization significantly. In particular, error control speeds up localization. Figure 3(a) shows that seven iterations in the scheme without error control produces a localization accuracy of about 11 m. With

error control, the localization accuracy improves to about 6 m. Furthermore, to achieve a given localization accuracy, the scheme with error control needs far fewer iterations. For example, in the same setting, error control takes about 6–8 iterations to stabilize. To achieve the same accuracy, more than 20 iterations are needed in the scheme without error control. From the communication perspective, although error control requires the communication of error registry, it pays to do so, since overall, much fewer rounds of communication are needed.

The advantage of error control is most prominent when the percentage of anchor nodes is low. As the percentage increases, the improvement diminishes (Figure 3(b)). Intuitively, when the percentage is low, the effect of error propagation is significant, and hence the benefit of error control. With error control, each iteration takes more computation, since the error registry need to be updated. We have simulated our localization algorithm using MATLAB on a 1.8 GHz Pentium II personal computer. In the baseline scheme, each node takes about 1.2 milliseconds per iteration, and the error control scheme takes about 2.5 milliseconds. This rough comparison shows that the amount of computation doubles in each iteration. However, as we have shown in Figure 3, the error control method takes less iterations to converge to a given accuracy level and reduces lost track possibilities. So if the accuracy requirement is high, the error control method is recommended.

4.3.1. Comparison with global localization algorithms

There have been a number of other localization algorithms proposed in the literature. Here, we refer to our scheme as the

TABLE 1: Instances of best localization results over 100 randomly generated test data for networks with a large number of anchors. **Bold** entries highlight the best performance for each case.

Anchor percentage	MDS	SDP	ILSnspa	ILS	SPA
10%	39	0	20	**42**	0
20%	0	6	35	**64**	0

incremental least-squares- (ILS-) based method. We compare ILS with the following methods:

(1) *ILS*: error controlled ILS with shortest path approximation in initialization;

(2) *ILSnspa*: error controlled ILS without shortest path approximation;

(3) *MDS-MAP*: localization based on multiscaling using connectivity data [17]. It is very robust to noise and low connectivity;

(4) *SDP*: localization based on semidefinite programming [19], working well for anisotropic networks;

(5) *SPA*: localization using shortest-path length between node pairs [27]. This is equivalent to the initialization step of ILS without further iterations.

Among the methods, MDS-MAP and SDP are global in nature, although heuristics have been used to distribute computation. SPA is very simple and easy to implement in distributed networks and used as a baseline comparison.

To compare performance, we generate 100 random instances of sensor field layout and run all the algorithms for each instance. The first performance metric is *error histogram*, shown in Figures 4(a) and 4(b) for each method for 10% and 20% anchor nodes, respectively. Here, the histogram is drawn in the form of vertically stacked bars, each bar indicates how many instances produce an average node localization error $\zeta_\mathbf{x}$ in a certain range, for example, smaller than 1.5, between 1.5 and 2.5, and so on. We favor method with long bar for error < 1.5 (estimates being accurate) and with short bar or no bar for large errors (estimates being robust). From this figure, we see that ILS performs well, comparable to MDP, better than SDP and SPA.

The second performance metric is *best case performance*, which indicates the number of instances that an algorithm produces the best results. If two algorithms produce the same best result (within 0.01 accuracy), both will be counted. Table 1 shows the number of the best results over 100 instances for 10% and 20% anchors, respectively. Again, we see that ILS produces most instances of best result, outperforming MDS and SDP global methods. This is amazing but not entirely surprising, given that we have carefully avoided error propagation and accumulation.

Our earlier work [22] has experimented with localization using an extremely low anchor density, where the whole network consists of only three anchors. This is the minimal requirement for range-based localization. Similar performance has been observed. In this setting, ILS performs much better than MDS or SDP. Interested readers may refer to [22] for more details.

5. ERROR CONTROL IN LOCALIZATION USING DIRECTIONAL SENSORS

A directional sensor in a 2D plane has 3 degrees of freedom: x-location, y-location, and a reference angle θ. Localization attempts to estimate these parameters if they are unknown. In a 3D space, the parameter set becomes (x-, y-, z-locations, yaw, pitch, roll). In this paper, we focus on the 2D case for simplicity. In this section, we first describe the basic localization algorithm and then present the error control method.

5.1. Basic localization algorithm

Directional sensors are inherently more complicated than range sensors and often more expensive in practice. To localize a set of directional sensors, we use the assistance of objects, which can be sensed by multiple sensors simultaneously. For example, a directional sensor can be a camera, and objects can be points in the field of view, especially points with easy-to-detect structural features such as corners. From sensor observations (e.g., images), one can extract constraints regarding the relative position between sensors and objects, and estimate unknown parameters in the world coordinate. Another example of directional sensor is radar, which uses beamforming technique to estimate the direction of signal with respect to its reference angle. Objects in this case can be airplanes.

The localization problem is formulated as follows. The network consists of sensors $\mathcal{S} = \{(\mathbf{x}_i, \theta_i)\}$ and a number of objects $\mathcal{O} = \{\mathbf{x}_o\}$. To start with, we assume that only a few sensors (anchors) know their parameters and no object parameters are known a priori. The goal of estimation is to estimate all $\mathcal{S}$ and $\mathcal{O}$. For localization, we use an iterative approach that is similar to multilateration: from anchor sensors, we estimate the location of neighboring objects; these objects are then used to estimate other unknown sensors; and so on. The algorithm alternates between using sensors to localize objects and using objects to estimate sensors, until the localization converges or some termination criterion is met.

5.1.1. Localizing an object using several known sensors

This step is easy. Let $\mathbf{x}_i$ and θ_i be the location and orientation of the sensor, respectively, and let α_i be the angle of the object from the sensor reference. Each measurement defines a ray originated from the sensor:

$$[-\sin(\theta_i + \alpha_i) \cos(\theta_i + \alpha_i)] \cdot (\mathbf{x} - \mathbf{x}_i) = 0. \qquad (12)$$

Given that the object can be seen by a set of sensors, the object location can be obtained by ray intersection, that is, by solving a linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$, where $\mathbf{A} = (\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n)^T$ and $\mathbf{b} = (b_1, b_2, \dots, b_n)^T$, in which $\mathbf{a}_i^T = (-\sin(\theta_i + \alpha_i), \cos(\theta_i + \alpha_i))$ and $b_i = \mathbf{a}_i^T \mathbf{x}_i$.

5.1.2. Estimating a sensor using several objects with known location

For this task, localization based on circle intersection has been proposed, for instance, in [8, 9]. The basic idea is as
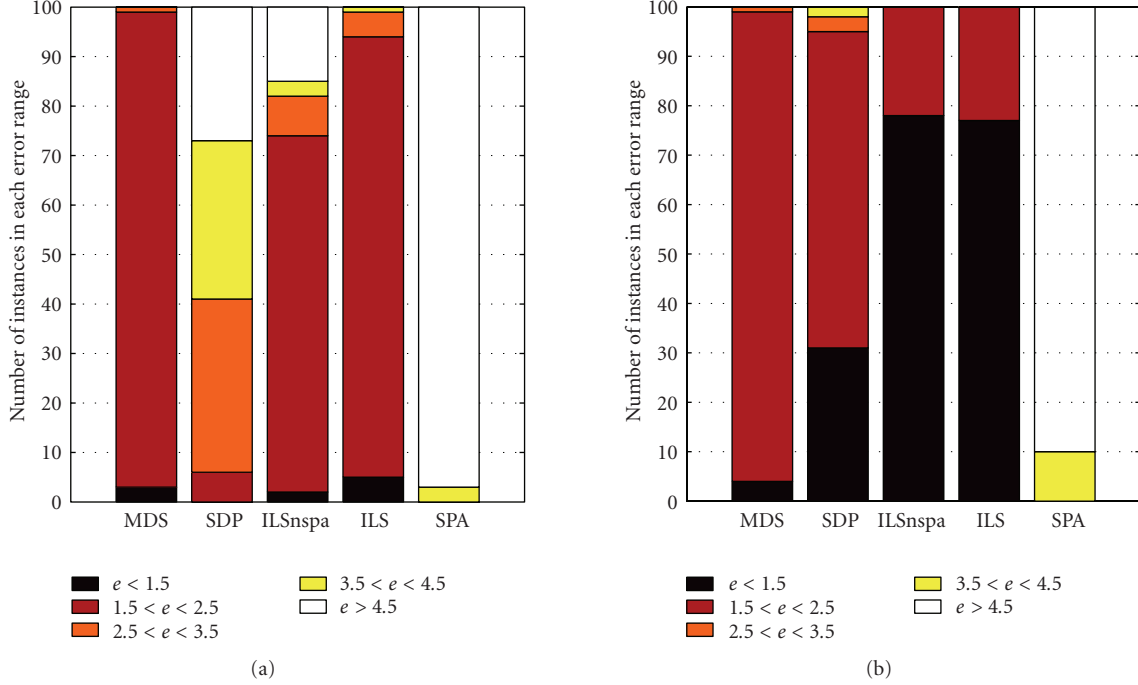
FIGURE 4: Error histograms: (a) 10% anchors and (b) 20% anchors.

follows: eliminating θ by taking the angle difference $\beta_{i,j} = \alpha_i - \alpha_j$, this produces the angle between the two rays from the sensor to objects i and j. All possible locations of the sensor form an arc, uniquely defined by the chord between object i and object j and the inscribed angle $\beta_{i,j}$. For example, in Figure 5, it is easy to verify that the central angle $\angle AO_{AB}B$ is $2\pi - 2\beta_{AB}$, where β_{AB} is the inscribed angle $\angle ASB$. We use the notation $\vec{\mathbf{n}}_{AB}$ to denote the unit vector orthogonal to $\mathbf{x}_B - \mathbf{x}_A$, and derive the center position as

$$\mathbf{x}_{O_{AB}} = \frac{\mathbf{x}_A + \mathbf{x}_B}{2} + \frac{\|\mathbf{x}_A - \mathbf{x}_B\|}{2\tan\beta_{AB}} \cdot \vec{\mathbf{n}}_{AB}. \qquad (13)$$

The first term is the midpoint between A and B. The second term travels along the radial direction $\vec{\mathbf{n}}_{AB}$ by length $\|\mathbf{x}_A - \mathbf{x}_B\|/(2\tan\beta_{AB})$ to get to the center. Likewise, one can also obtain the radius as $\|\mathbf{x}_A - \mathbf{x}_B\|/(2|\sin\beta_{AB}|)$.

The location of the sensor can be estimated by intersecting multiple arcs. Once the sensor location is known, the reference angle can be estimated trivially. In this paper, we use an equivalent method but with a slightly different form, shown in Figure 5. Rather than intersecting two circles, which is a nonlinear operation, we find sensor location S by mirroring A with respect to the line linking the two centers O_{AB} and O_{AC}. Mathematically,

$$\mathbf{x}_S = \mathbf{x}_A - 2\left[(\mathbf{x}_A - \mathbf{x}_{O_{AB}})^T \cdot \vec{\mathbf{n}}\right]\vec{\mathbf{n}}, \qquad (14)$$

where $\vec{\mathbf{n}}$ is the unit vector orthogonal to the line from O_{AB} to O_{AC}. The second term in (14) is twice the projection of the
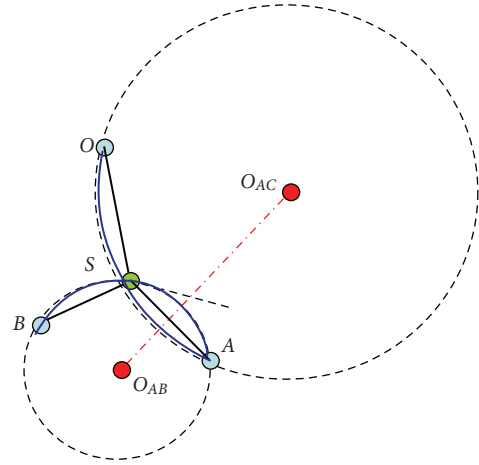

FIGURE 5: Estimating an unknown sensor using three objects.

displacement $\mathbf{x}_A - \mathbf{x}_{O_{AB}}$ onto the orthogonal direction $\vec{\mathbf{n}}$. All steps (13)-(14) are easy to compute, lending themselves well to the error characterization step described below.

5.2. Error control

In this section, we describe our error control scheme separately for the two alternating localization steps of estimating objects and sensors. For each step, we illustrate the three components of error control: error characterization, neighbor selection, and update criterion.

5.2.1. Localizing objects using known sensors

The basic estimation algorithm is ray intersection (12). Intuitively, the location estimate error will be big when the sensors and the object are collinear, or if the object is very far away from all sensors around the same direction. In both cases, all rays are almost parallel, causing the intersection to be sensitive to noise. Similar observation has been made in [8], pointing out that small angles are more susceptible to noise than large angles. These observations can be formalized using our error characterization. Estimation is due to two sources: sensor location error (vertex error), which causes the corresponding ray to shift, and angle measurement error (edge error), which causes the ray to rotate.

(1) The estimation error due to vertex error can be derived in closed form, similar to the derivation for range sensors in Section 4. The covariance is $\mathbf{A}^{\dagger} \cdot \text{Cov}(\Delta \mathbf{b}) \cdot (\mathbf{A}^{\dagger})^{T}$, where $\mathbf{A}$ and $\mathbf{b}$ are defined earlier in Section 5.1. Collinearity leads to poorly conditioned $\mathbf{A}$ amplifying noise; this is also similar to range sensors.

(2) The error due to the angle measurement α affects the sin and cos terms in (12), and can be approximated using first-order Taylor expansion, that is, $\cos(\theta + \alpha + \Delta\alpha) = \cos(\theta + \alpha) - \sin(\theta + \alpha) \cdot \Delta\alpha$ and $\sin(\theta + \alpha + \Delta\alpha) = \sin(\theta + \alpha) + \cos(\theta + \alpha) \cdot \Delta\alpha$. Through this linearization, we can compute the contribution to location error via linear transformations.

To localize an object, at least two sensors need to be known. The neighbor selection step ranks all known sensors based on its location error (trace of covariance in the registry) in an ascending order, and set a threshold that is twice the value of the second lowest value. Any sensor with error above the threshold is considered too noisy and discarded from the neighborhood. The update criterion is also simple: any new estimate is examined by two metrics: its level of uncertainty, measured as the trace of the covariance, and the data fitting error, measured as the deviation between the actually observed angle measurement to the would-be angle measurements if the sensor were located at the estimated position. If the uncertainty and data fitting error are both low compared to their respective thresholds, the new estimate is accepted and the node registry is updated. Otherwise, the estimate is discarded.

5.2.2. Estimating sensor based on objects with known locations

As described in Section 5.1, we first derive arc specifications such as the radius and the center. Uncertainty in the object locations and angle measurement will translate into uncertainty in center location. Although the computation of center (13) is straightforward, characterizing its uncertainty requires some approximation. The vertex error (uncertainty in $\mathbf{x}_A$ and $\mathbf{x}_B$) contributes to the first term, the midpoint between A and B, that is, $(\mathbf{x}_A + \mathbf{x}_B)/2$, giving a covariance of $(\text{Cov}(\mathbf{x}_A) + \text{Cov}(\mathbf{x}_B))/4$. It also contributes to the second term via the length $\|\mathbf{x}_A - \mathbf{x}_B\|$ and the direction $\vec{\mathbf{n}}_{AB}$. We ignore this contribution assuming that although A and B may vary their
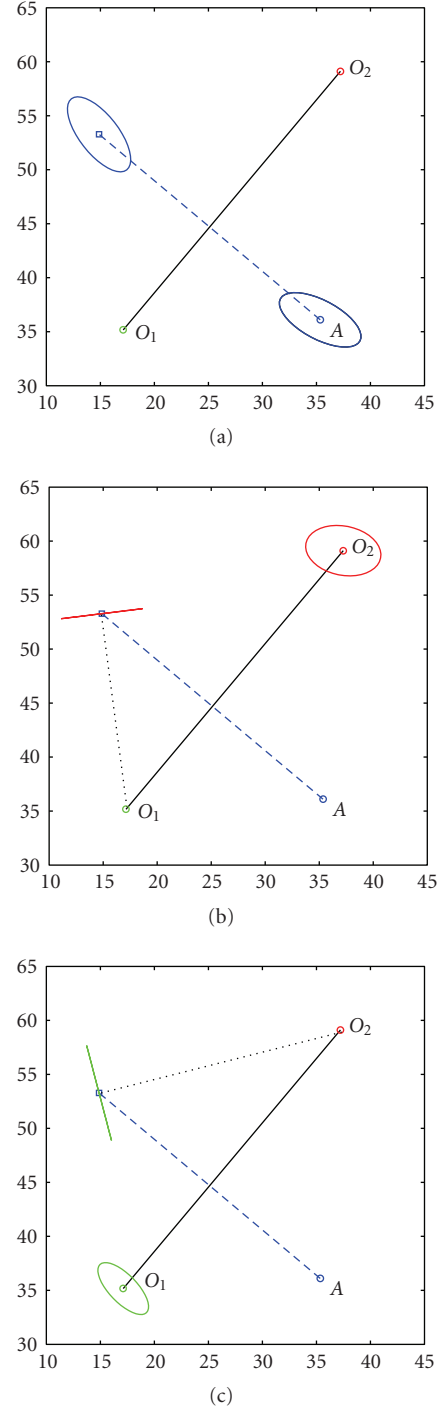

FIGURE 6: Error in calculating reflection of A with respect to the line linking two points (O_1 and O_2): (a) contribution from A's uncertainty, (b) contribution from O_2's uncertainty, and (c) contribution from O_1's uncertainty.

locations, the overall distance between them and the direction from one to another do not change much. This assumption is reasonable if A and B are well separated, and each has a covariance that is sufficiently small compared to the distance between the two. The edge error (uncertainty in β_{AB}) affects only the second term in (13) via $1/\tan\beta_{AB}$. Fixing all other
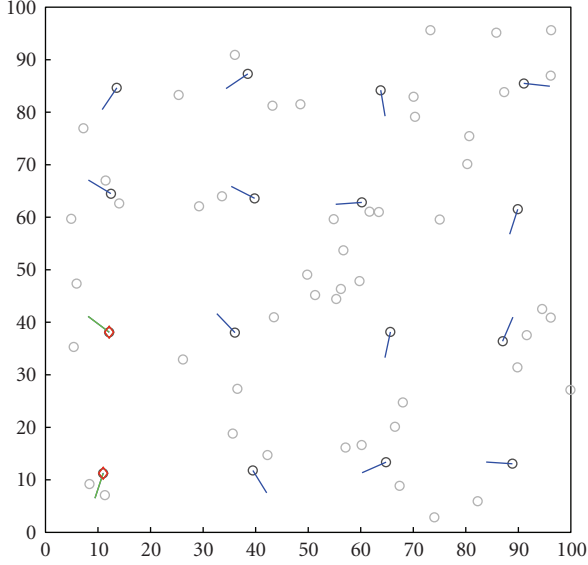
FIGURE 7: A network consisting of directional sensors (location marked with black circles, reference angles shown in short thick lines), and objects (gray circles). Anchor sensors are marked as red diamonds.

variables, the contribution to uncertainty can be calculated via Taylor expansion $1/\tan(\beta + \Delta\beta) = 1/\tan\beta - 1/\sin^2\beta \cdot \Delta\beta$.

The second step is to find the mirror reflection of A with respect to the line linking the centers O_1 and O_2 (see Figure 6). Note that both centers have some location error, shown as the ellipsoids in Figures 6(b) and 6(c), respectively. The error in the estimated location S can be calculated in the following way.

(1) Fixing the two ends of the line and varying the location of A, the uncertainty is just the symmetry of the uncertainty in A (Figure 6(a)).

(2) Fixing A and one end of the line (O_1) to calculate the contribution from the uncertainty of the other end (the ellipsoid around O_2 in Figure 6(b)), this causes the line to dangle around the fixed end and produces uncertainty in A's mirror reflection with respect to the line. The error should be a short piece of arc dangling around A's mirror reflection, marked with a small square. However, an arc is not a Gaussian density, making further error characterization difficult. In Figure 6(b), it is shown as a short line segment to approximate.

(3) Fixing A and O_2, and varying O_1 location. This is symmetric with case (2). The error is also a short arc, shown in Figure 6(c).

The total error is the sum of the three terms. This is based on the assumption that error in A and the two ends are independent.

The neighbor selection is the same as in the localizing object using known sensors case, except that the threshold value is slightly larger. The update criterion is also similar, rejecting estimates based on the level of uncertainty and data fitting error.
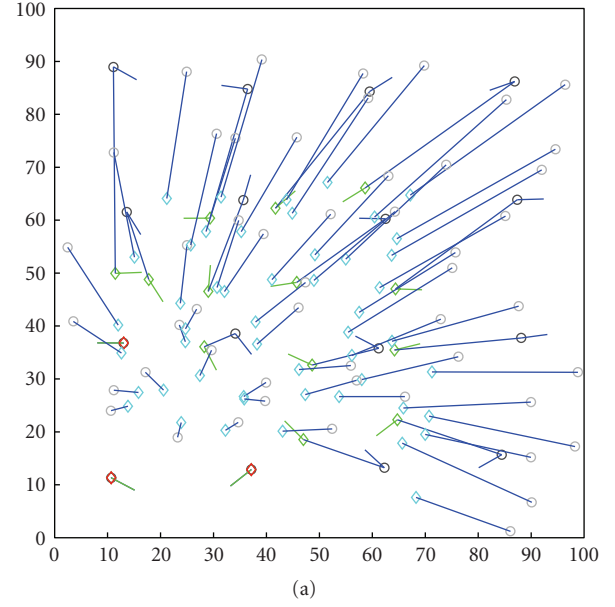
(a)
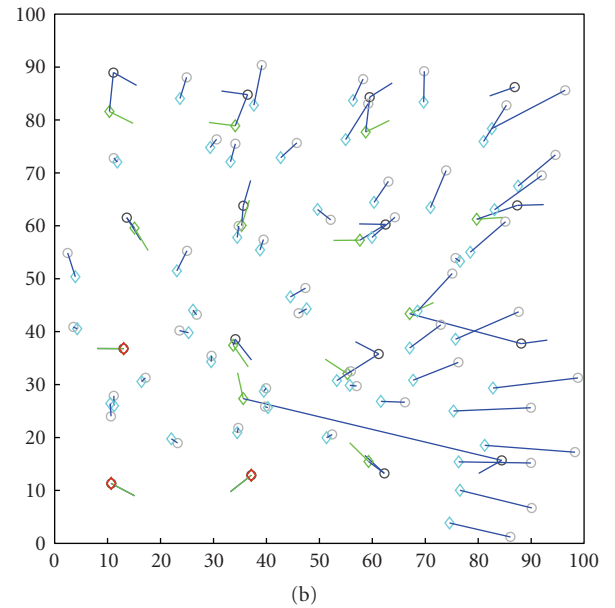
(b)

FIGURE 8: Localization results: (a) without error control and (b) with error control.

5.3. Simulation experiment

The network is simulated in a 100×100 m field, with 100 objects randomly spaced according to a uniform distribution, and 16 directional sensors forming a rough 4×4 grid. Figure 7 shows an example network, in which the objects are marked with gray circles and directional sensors marked with black circles. Each sensor has a reference angle shown with short rays. Anchor sensors are marked in dark. The bottom two sensors in the left-most column are always anchors. Other nodes have a certain probability, for example, 10% chance of being an anchor. Each directional sensor can sense the angle of an object within a radius of 50 m to itself. In our
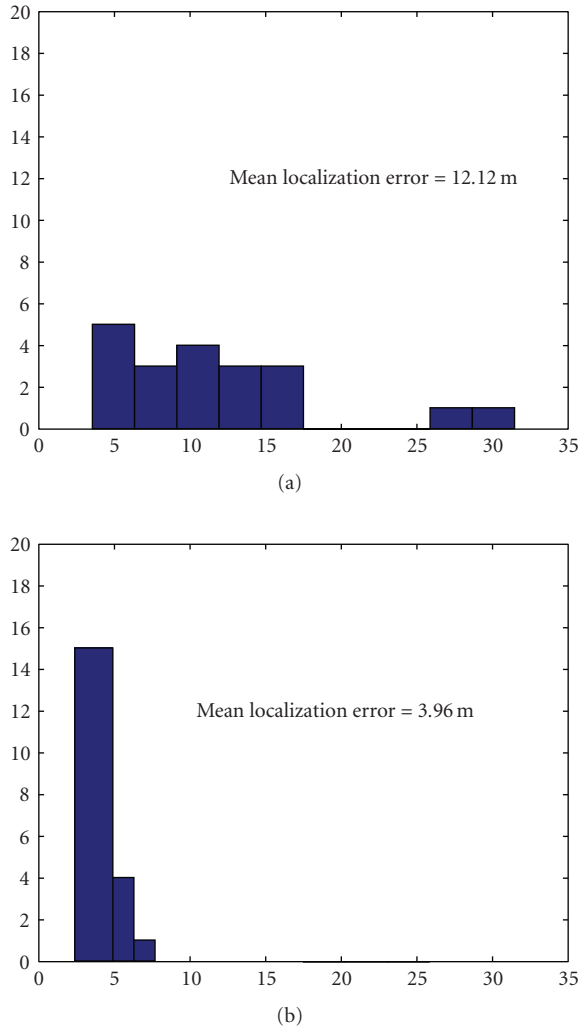
(a)

(b)

FIGURE 9: Error histogram obtained over 20 simulations runs: (a) without error control and (b) with error control.

simulation, angle measurements are simulated with independent Gaussian noise of zero-mean and standard deviation $5°$.

Figure 8(a) shows the localization result without error control. The ground truth locations are marked with circles, and estimated locations are marked with diamonds. The lines show the displacement between the two. Notice that the lines are often long, indicating that many sensors and objects are estimated to be in the wrong position. This is most prominent in nodes far away from the bottom-left corner. This is expected, since more hops to anchors mean a longer error accumulation chain. Figure 8(b) shows the localization result with exactly the same setup, but now with error control. It is apparent that the estimation performance is much better. The lines are much shorter except for only a few.

Figure 9 shows the histogram of average localization error ζ_x over 20 simulation runs. The left panel (Figure 9(a)) is the histogram without error control. We see that most runs produce an average localization error in the 5–20 m range, and two runs produce very high error (~ 30 m). These runs are basically "lost runs" where the error propagation is so bad

that localization iterations get stuck and cannot improve. Averaging over all 20 runs, the mean ζ_x is 12.12 m. In contrast, Figure 9(b) shows the error histogram with error control. Notice the differences as follows. First, the dynamic range of ζ_x is much narrower, from 2.5 m to 7 m, showing that error control mechanism is less sensitive to randomization in simulation. The largest ζ_x is around 7 m, and no lost run has occurred. This shows that localization with error control is more robust. Secondly, the error control clearly produces much more accurate localization results. The mean ζ_x across all simulation runs is 3.96 m. This is a factor of 3—4 improvement over the case without error control (12.12 m). Finally, notice that the error control scheme can achieve low ζ_x values that are never achieved without error control. This shows that if high localization accuracy is required, error control becomes a necessity rather than an improvement measure.

We are also interested in comparing with global optimization methods to localize directional sensors. So far there are not many publications of global methods on this topic. We will leave this comparison to our future investigation.

6. CONCLUSION

In this paper, we have examined the problem of error propagation and accumulation in distributed node self-localization. To mitigate the effect, we have presented an error-control mechanism. The basic idea is to keep track of estimation error and document each location estimate with its level of uncertainty. This enables neighbor selection to discard neighbors with unreliable location information from being used to localize other free nodes. Effectively, it cuts the link of error propagation. Furthermore, an update criterion is proposed to reject bad estimates. This prevents large error from generating. We have applied the error control mechanism to localize range sensors and directional sensors. Simulations have shown that the error-control algorithm significantly improves localization performance in terms of accuracy and robustness. Although the localization methods in each iteration are very simple, with the help of error control, they achieve very good localization performance, comparable to or better than more global methods in range-based localization simulations.

REFERENCES

[1] K. W. Kolodziej and J. Hjelm, *Local Positioning Systems: LBS Applications and Services*, CRC Press, Boca Raton, Fla, USA, 2006.

[2] A. LaMarca, Y. Chawathe, S. Consolvo, et al., "Place lab: device positioning using radio beacons in the wild," in *Proceedings of the 3rd International Conference on Pervasive Computing (PERVASIVE '05)*, vol. 3468, pp. 116–133, Munich, Germany, May 2005.

[3] J. Letchner, D. Fox, and A. LaMarca, "Large-scale localization from wireless signal strength," in *Proceedings of the 20th National Conference on Artificial Intelligence and the 17th Innovative Applications of Artificial Intelligence Conference (AAAI '05)*, pp. 15–20, Pittsburgh, Pa, USA, July 2005.

[4] H. Lim, L.-C. Kung, J. C. Hou, and H. Luo, "Zero-configuration, robust indoor localization: theory and experimentation," in *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM '06)*, pp. 1–12, Barcelona, Spain, April 2006.

[5] P. Bahl and V. N. Padmanabhan, "RADAR: an in-building RF-based user location and tracking system," in *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '00)*, vol. 2, pp. 775–784, Tel Aviv, Israel, March 2000.

[6] K. Kaemarungsi and P. Krishnamurthy, "Properties of indoor received signal strength for WLAN location fingerprinting," in *Proceedings of the 1st Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MOBIQUITOUS '04)*, pp. 14–23, Boston, Mass, USA, August 2004.

[7] D. Niculescu and B. Nath, "Ad hoc positioning system (APS)," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '01)*, vol. 5, pp. 2926–2931, San Antonio, Tex, USA, November 2001.

[8] D. Niculescu and B. Nath, "Ad hoc positioning system (APS) using AOA," in *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '03)*, vol. 3, pp. 1734–1743, San Francisco, Calif, USA, March-April 2003.

[9] R. Peng and M. L. Sichitiu, "Angle of arrival localization for wireless sensor networks," in *Proceedings of the 3rd Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON '06)*, vol. 1, pp. 374–382, Reston, Va, USA, September 2006.

[10] R. L. Moses, D. Krishnamurthy, and R. M. Patterson, "A self-localization method for wireless sensor networks," *EURASIP Journal on Applied Signal Processing*, vol. 2003, no. 4, pp. 348–358, 2003.

[11] D. Niculescu and B. Nath, "Position and orientation in ad hoc networks," *Ad Hoc Networks*, vol. 2, no. 2, pp. 133–151, 2004.

[12] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low-cost outdoor localization for very small devices," *IEEE Personal Communications*, vol. 7, no. 5, pp. 28–34, 2000.

[13] C. Savarese, J. M. Rabaey, and K. Langendoen, "Robust positioning algorithms for distributed Ad-Hoc wireless sensor networks," in *Proceedings of USENIX Annual Technical Conference*, pp. 317–327, Monterey, Calif, USA, June 2002.

[14] L. Kleinrock and J. Silvester, "Optimum transmission radii for packet radio networks or why six is a magic number," in *Proceedings of the IEEE National Telecommunications Conference (NTC '78)*, pp. 4.3.1–4.3.5, Birmingham, Ala, USA, December 1978.

[15] J. A. Costa, N. Patwari, and A. O. Hero III, "Distributed weighted-multidimensional scaling for node localization in sensor networks," *ACM Transactions on Sensor Networks*, vol. 2, no. 1, pp. 39–64, 2006.

[16] N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero III, R. L. Moses, and N. S. Correal, "Locating the nodes: cooperative localization in wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 22, no. 4, pp. 54–69, 2005.

[17] Y. Shang, W. Ruml, Y. Zhang, and M. Fromherz, "Localization from connectivity in sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 11, pp. 961–974, 2004.

[18] L. Doherty, K. S. J. Pister, and L. El Ghaoui, "Convex position estimation in wireless sensor networks," in *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '01)*, vol. 3, pp. 1655–1663, Anchorage, Alaska, USA, April 2001.

[19] P. Biswas and Y. Ye, "Semidefinite programming for ad hoc wireless sensor network localization," in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN '04)*, pp. 46–54, Berkeley, Calif, USA, April 2004.

[20] A. Savvides, W. L. Garber, R. L. Moses, and M. B. Srivastava, "An analysis of error inducing parameters in multihop sensor node localization," *IEEE Transactions on Mobile Computing*, vol. 4, no. 6, pp. 567–577, 2005.

[21] D. Niculescu and B. Nath, "Error characteristics of ad hoc positioning systems (APS)," in *Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '04)*, pp. 20–30, Roppongi Hills, Tokyo, Japan, May 2004.

[22] J. Liu, Y. Zhang, and F. Zhao, "Robust distributed node localization with error management," in *Proceedings of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '06)*, pp. 250–261, Florence, Italy, May 2006.

[23] Y. Zhang, K. Partridge, and J. Reich, "Localizing tags using mobile infrastructure," in *Proceedings of the 3rd International Symposium on Location- and Context-Awareness (LoCA '07)*, vol. 4718 of *Lecture Notes in Computer Science*, pp. 279–296, Oberpfaffenhofen, Germany, September 2007.

[24] D. Moore, J. Leonard, D. Rus, and S. Teller, "Robust distributed network localization with noisy range measurements," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)*, pp. 50–61, Baltimore, Md, USA, November 2004.

[25] E. Krotkov, *Exploratory visual sensing with an agile camera*, Ph.D. thesis, University of Pennsylvania, Philadelphia, Pa, USA, 1987.

[26] A. Savvides, H. Park, and M. B. Srivastava, "The *n*-hop multilateration primitive for node localization problems," *Mobile Networks and Applications*, vol. 8, no. 4, pp. 443–451, 2003.

[27] K. Whitehouse, "Calamari: a localization system for sensor networks," http://www.cs.virginia.edu/~whitehouse/research/localization/, 2003.