

Robust Distributed Node Localization with Error Management

Juan Liu
Palo Alto Research Center
jjliu@parc.com

Ying Zhang
Palo Alto Research Center
yzhang@parc.com

Feng Zhao
Microsoft Research
zhao@microsoft.com

ABSTRACT

Location knowledge of nodes in a network is essential for many tasks such as routing, cooperative sensing, or service delivery in ad hoc, mobile, or sensor networks. This paper introduces a novel iterative method ILS for node localization starting with a relatively small number of anchor nodes in a large network. At each iteration, nodes are localized using a least-squares based algorithm. The computation is lightweight, fast, and any-time. To prevent error from propagating and accumulating during the iteration, the error control mechanism of the algorithm uses an error registry to select nodes that participate in the localization, based on their relative contribution to the localization accuracy. Simulation results have shown that the active selection strategy significantly mitigates the effect of error propagation. The algorithm has been tested on a network of Berkeley Mica2 motes with ultrasound TOA ranging devices. We have compared the algorithm with more global methods such as MDS-MAP and SDP-based algorithm both in simulation and on real hardware. The iterative localization achieves comparable location accuracy in both cases, compared to the more global methods, and has the advantage of being fully decentralized.

Categories and Subject Descriptors:

F.2.2 [Theory of Computation].

General Terms: Algorithms.

Keywords:

Target localization, Self localization, Ad hoc network, Information, Sensor network

1. INTRODUCTION

Sensor network tasks such as geo-routing, data-centric storage, spatio-temporal information dissemination, or collaborative signal processing rely on knowing geographic locations of sensor nodes or physical phenomena of interest. When GPS or other device assisted location technology is not available due to shadowing or cost, it is attractive to develop inexpensive localization approaches. In node local-

ization, most existing methods start with a small number of anchor nodes whose locations are known. Other nodes localize themselves with respect to these anchor nodes. In this paper, we introduce an iterative least-squares (ILS) approach to node localization, in which location information progressively propagates from anchor nodes to other nodes. The iterative approach has been studied by others, for example, as in multilateration [1, 2]. What is new about our approach is the introduction of an error control and a robust formulation of the localization problem so that the algorithm is less sensitive to noise and computes the location information incrementally.

(1) *Error control in iterative localization.* We have observed that many iterative methods such as multilateration [1] may suffer from adverse effects of error propagation and accumulation, as nodes being localized and becoming new anchors for other free nodes. This could lead to unbounded error in localization for large networks. The effect of error propagation is less prominent in global methods such as MDS [3] or SDP [4, 5], since global constraints tend to balance against each other. However, global methods are less amendable to distributed implementation in an ad hoc network. This paper introduces a local error control mechanism to determine and use reliable location data, and filter out outliers (“the bad seeds”) that may contaminate the entire network.

(2) *Robust least-squares localization.* At each iteration of localization, we compute a location estimate using a robust least-squares (RLS) formulation. Compared to the traditional least-squares (LS), RLS is more stable against measurement noise. We have developed an incremental algorithm for the LS/RLS method that incorporates a new sensor measurement into the location estimation without the need to recompute from scratch with all the previous measurements. This greatly improves the computational efficiency and allows an any-time implementation, which can terminate at any time and still provide usable information. The incremental, or any-time, aspect is particularly desirable in resource-constrained, decentralized networks where nodes are limited in on-board energy and computation. In contrast, most of existing techniques assume that all the relevant sensor data for estimation are available at the processing node, and perform localization in a batch mode. Furthermore, the incremental computation is efficient and lightweight, and can be implemented on mote-class sensor nodes such as the Berkeley motes.

Our method has been tested using both simulations and real experiments on a small network of Mica2 motes. Results

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MobiHoc'06, May 22–25, 2006, Florence, Italy.

Copyright 2006 ACM 1-59593-368-9/06/0005 ...\$5.00.

have shown that the error control mechanism is effective in mitigating the effect of error propagation. The error control significantly improves the localization quality and speeds up the convergence in iteration. The location accuracy is comparable to and in many cases better than those of more global methods such as MDS-MAP [3] or SDP [5].

Related work

Node localization is an active research area. A number of approaches [3, 4, 5] formulate the localization problem as joint estimation problems, where the parameters to estimate are the collection of unknown node locations $\{\mathbf{x}_j\}_{j=1,\dots,N}$. These global formulations consider sensor measurements as constraints on inter-node distances, and apply optimization techniques to find feasible solutions. For example, [3] localizes nodes based on connectivity constraints, and uses multidimensional scaling (MDS) to compute feasible locations. Similarly, [4, 5] model inter-node distance measurements as convex constraints, and use linear programming and semi-definite programming (SDP) methods to estimate the location of free nodes. There has also been development in localization algorithms using kernel-based learning approach [6], with RSS measurements. While these 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 sensor networks.

To reduce the need for non-local communication, various distributed node localization techniques have been proposed. The basic idea is to decompose a global joint estimation problem into smaller sub-problems and to iterate over the sub-problems [1, 7, 8, 9]. There are three groups of techniques related to our proposed method. The first group (e.g., [1, 7]) starts from anchors and uses local computation to iteratively localize free nodes. This approach greatly reduces computational complexity and communication overhead. However, it suffers from (1) low success rate (i.e., not being able to localize all the nodes) in networks with low node connectivity and a small number of well-separated anchors, and (2) being prone to error propagation and accumulation when the location information flows from anchors to free nodes. Despite its impact, the error propagation problem has not received adequate attention in the current literature. Localization using robust quadrilaterals has been proposed [10]. However, this method requires high connectivity, i.e., the existence of many robust quadrilaterals. In Sec. 4, we address error propagation in details and design a control mechanism to mitigate the problem. The second group (e.g., [11]) uses shortest path approximation to anchor nodes to approximate Euclidean distances. This approach introduces large position errors in anisotropic networks. Remedies to the shortest path approximation work only to certain extent. For example, the method described in [11] would fail when a small number of anchors, for instance three anchors, are located within one hop distance to each other. The third group uses local refinement [12] which requires an initial solution. Many local refinement methods exist, the most recent one is a distributed weighted MDS [13]. Local refinements are very effective and can be applied to the result of any algorithm. Due to space limitations, we will not discuss and compare refinement methods in this paper.

The computation structure for localization has been discussed in literature as well. A number of incremental methods, such as Kalman filter based approach [2] or particle filtering, have been proposed. Compared to these methods, our approach is computationally more efficient. The computational complexity of the Kalman filter based multilateration approach described in [2] can be a lot higher due to the need to maintain estimate prediction and Kalman gains. In this paper, we focus on designing light-weight algorithms that can be implemented on sensor nodes such as the Berkeley motes.

2. ITERATIVE LOCALIZATION: AN OVERVIEW

The basic idea of our iterative localization approach is to use nodes with known location (e.g., the anchor nodes) to localize others (i.e., the free nodes). Without loss of generality, we consider localization of a stationary network in a 2-D plane. We assume that the sensors are all range sensors producing distance measurements, typically derived from sensing of physical signals such as ultrasonic or RF transmitted from one node to another. Common examples of range sensors are:

- Time-of-arrival (TOA) sensor. It measures distance using time of flight between sender and receiver. The observation model is $z_i = \|\mathbf{x} - \mathbf{x}_i\|/c + \epsilon_i$, where z_i is the measurement of i -th sensor, \mathbf{x} and \mathbf{x}_i are target and sensor locations respectively, $\|\cdot\|$ is Euclidean norm, c is the traveling speed of the emitted signal, and ϵ_i is measurement noise.
- Received signal strength (RSS) sensor. It measures distance through signal attenuation. The observation model is $z_i = \frac{\eta}{\|\mathbf{x} - \mathbf{x}_i\|^2} + \epsilon_i$, η is the source amplitude or energy, assuming signals propagate isotropically and losslessly.

From measurement data z , one can infer about the distance:

$$\|\mathbf{x} - \mathbf{x}_i\| = f(z_i), \quad (1)$$

where the form of function $f(\cdot)$ may vary depending on the sensing model. For TOA, $f(\cdot) = cz_i$, and for RSS, $f(\cdot) \propto z_i^{-1/2}$.

We also define a distance constraint graph (DCG), in which vertices are sensor nodes, and edges represent distance constraints between pairs of nodes. Any pair of nodes that can reliably sense each other's signal (hence form range estimate), and can communicate with each other, either directly or via some intermediate nodes, are called mutually *immediate neighbors* in DCG.

In a network of range sensors, given the distance measurements or their approximations, each free node computes an estimate, $\hat{\mathbf{x}}_t$ using multilateration to be discussed in Sec. 3. Initially, only anchor nodes are aware of their locations. Free nodes are localized with respect to the anchors. The newly localized free nodes are then used as "pseudo-anchors" to localize other free nodes. Each iteration pushes location information from anchors to nearby free nodes. A pseudo-anchors may further refine its location estimate based on its newly localized neighbors.

The approach above works well for networks with reasonable dense anchors. However, if the anchor density is too

low, the approach may fail to localize all the nodes in the network and leave “empty spots”. The reason is that for the approach to make progress at each iteration, there must be at least one node with at least *three* neighbors with known locations. This assumption may not be satisfied in networks with only a few anchors, especially if the anchors are well separated. For these low anchor density networks, we introduce an optional initialization stage. During the initialization, anchor nodes broadcast their location information to the DCG. Each free node computes a shortest path to each of the nearby anchors, and use the path length in DCG 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. The initialization complexity can be controlled by limiting the shortest path computation to be within a certain hop count, or abandoning long hop-count paths if a node has already found three or more nearby anchors. Using this shortest path approximation, all nodes can be localized as long as there are *three* anchors in the entire network, which is the minimal requirement for all localization algorithms to obtain absolute coordinates.

To control the error propagation, we propose a mechanism to keep track of estimation error and determines which neighbors have reliable location information and which don’t. This mechanism filters out outliers (“the bad seeds”), preventing them from propagating further. To keep track of estimation error, we first consider the simplified problem of localizing a node t given \mathcal{N} , the set of neighbors in DCG with known locations. For the shortest path approximation, \mathcal{N} includes those 3~5 closest anchors to the node. The localization error comes from two sources:

- Vertex errors $\{e^v\}$ (for all node $\in \mathcal{N}$): this is the error in neighboring nodes, since their location information may contain error, especially for non-anchor neighbors. For anchor nodes, the vertex error is simply zero.
- Edge error $\{e^e\}$ (for all edges between t and node $\in \mathcal{N}$): this is the error in distance measurements. For regular distances, a Gaussian white noise is assumed. For shortest path approximations, error will be estimated as described in App. A.

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

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

In a sense, we not only compute a location estimate $\hat{\mathbf{x}}_t$, but also document the “quality” of the estimate using \hat{e}_t . In the iterative refinement 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. The process of computing estimate and characterizing error then repeats.

Table 1 outlines the iterative localization procedure. Details will be discussed in later sections. Sec. 3 introduces a robust least-squares formulation and explains the computational aspects. Sec. 4 analyzes error propagation and designs an error control mechanism, including the recursive update of vertex errors (2) and a registry update criterion. Sec. 5 presents simulation results and comparisons

ITERATIVE LOCALIZATION

Each node i holds the tuple $(\mathbf{x}, e^v)_i$, where

\mathbf{x} is the node location (or estimate);

e^v is the vertex error.

Each edge j corresponds to a tuple $(z, e^e)_j$, where

z is the distance measurement;

e^e is the edge error

Initialization step (optional)

computing shortest path to anchors

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.

Table 1: The iterative localization algorithm.

with other methods, and Sec. 6 describes the results using real TOA data from sensor hardware.

3. BASIC LOCALIZATION ALGORITHM: LEAST-SQUARES AND ROBUST LEAST-SQUARES

In our setting, a free node t collects measurements from all its neighbors with known or estimated location $\{\mathbf{x}_i\}$, where i indexes over the neighbors, and computes a location estimate. We first describe an LS formulation and a RLS formulation which is less sensitive to error. We then introduce an incremental algorithm for computing the LS/RLS solution.

3.1 Least-squares multilateration

Ignoring the measurement (edge) and neighbor location (vertex) errors for the moment, we square both sides of (1) and obtain:

$$\|\mathbf{x}\|^2 + \|\mathbf{x}_i\|^2 - 2\mathbf{x}_i^T \mathbf{x} = f^2(z_i), \quad i = 0, 1, \dots$$

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

$$-2(\mathbf{x}_i - \mathbf{x}_0)^T \mathbf{x} = (f^2(z_i) - f^2(z_0)) - (\|\mathbf{x}_i\|^2 - \|\mathbf{x}_0\|^2).$$

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

$$\mathbf{a}_i^T \mathbf{x} = b_i. \quad (3)$$

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:

$$\mathbf{A} \mathbf{x} = \mathbf{b}, \quad (4)$$

where $\mathbf{A} = (\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n)^T$ and $\mathbf{b} = (b_1, b_2, \dots, b_n)^T$. The least-squares solution to the linear system (4) is $\hat{\mathbf{x}}_t = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b}$.

Note that \mathbf{A} encodes the geometric information about the sensor configuration, and \mathbf{b} captures the information about sensor measurements. The accuracy of LS localization is thus influenced by these two factors. A pathological case is that nodes are colinear. In this case, \mathbf{A} is singular, and any slight perturbation in \mathbf{b} (i.e., measurement noise) can be amplified arbitrarily. For LS to work, we require that the nodes are well spaced, and \mathbf{A} is well-conditioned.

3.2 Robust least-squares formulation

The least-squares solution gives

$$\hat{\mathbf{x}}_t = \arg \min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2, \quad (5)$$

assuming noiseless conditions. However, in practice, we have error in both the neighbor location \mathbf{x}_i 's (i.e., vertex errors) and the measurement z_i 's (i.e., edge errors). We denote the perturbation in \mathbf{A} and \mathbf{b} as $\Delta\mathbf{A}$ and $\Delta\mathbf{b}$. The presence of the error motivates us to switch to a RLS formulation [14, 15]

$$\hat{\mathbf{x}}_t = \arg \min_{\mathbf{x}} E\|(\mathbf{A} + \Delta\mathbf{A})\mathbf{x} - (\mathbf{b} + \Delta\mathbf{b})\|^2. \quad (6)$$

The expectation is with respect to $\Delta\mathbf{A}$ and $\Delta\mathbf{b}$. The cost function seeks a location estimate which minimizes data discrepancy in the average sense. This optimization problem is convex with respect to \mathbf{x} . Without loss of generality, we assume that $\Delta\mathbf{A}$ and $\Delta\mathbf{b}$ are zero mean. The cost to minimize is

$$\begin{aligned} \mathcal{E} &= E\|(\mathbf{A}\mathbf{x} - \mathbf{b}) + (\Delta\mathbf{A}\mathbf{x} - \Delta\mathbf{b})\|^2 \\ &= \left(\mathbf{x}^T \mathbf{A}^T \mathbf{A} \mathbf{x} - 2\mathbf{x}^T \mathbf{A}^T \mathbf{b} + \mathbf{b}^T \mathbf{b} \right) + \\ &\quad \left(\mathbf{x}^T E[\Delta\mathbf{A}^T \Delta\mathbf{A}] \mathbf{x} - 2\mathbf{x}^T E[\Delta\mathbf{A}^T \Delta\mathbf{b}] + E[\Delta\mathbf{b}^T \Delta\mathbf{b}] \right) \\ &= \mathbf{x}^T (\mathbf{A}^T \mathbf{A} + E[\Delta\mathbf{A}^T \cdot \Delta\mathbf{A}]) \mathbf{x} \\ &\quad - 2\mathbf{x}^T [\mathbf{A}^T \mathbf{b} + E[\Delta\mathbf{A}^T \Delta\mathbf{b}]] \\ &\quad + (\mathbf{b}^T \mathbf{b} + E[\Delta\mathbf{b}^T \Delta\mathbf{b}]) \end{aligned}$$

Collecting the quadratic and linear terms, the solution to the above RLS problem is given by

$$\hat{\mathbf{x}}_t = \left(\mathbf{A}^T \mathbf{A} + \mathbf{C}_{\mathbf{A}} \right)^{-1} \cdot \left[\mathbf{A}^T \mathbf{b} + \mathbf{r}_{\mathbf{Ab}} \right] \quad (7)$$

where

- $\mathbf{C}_{\mathbf{A}} = E[\Delta\mathbf{A}^T \cdot \Delta\mathbf{A}]$ is the covariance matrix of perturbation of $\Delta\mathbf{A}$. The computation is explained in App. B.
- $\mathbf{r}_{\mathbf{Ab}} = E[\Delta\mathbf{A}^T \Delta\mathbf{b}]$ is the correlation. It is zero if $\Delta\mathbf{A}$ and $\Delta\mathbf{b}$ are uncorrelated. In App. C, we show that $\mathbf{r}_{\mathbf{Ab}}$ is often negligible compared to the data term $\mathbf{A}^T \mathbf{b}$, i.e., the correlation between $\Delta\mathbf{A}$ and $\Delta\mathbf{b}$ is very weak.

In this paper, we assume $\mathbf{r}_{\mathbf{Ab}}$ is negligible for the ease of derivation. The analysis can be extended without much technical difficulty even when $\mathbf{r}_{\mathbf{Ab}}$ is significant. Under this assumption, the RLS solution is

$$\hat{\mathbf{x}}_t = \left(\mathbf{A}^T \mathbf{A} + \mathbf{C}_{\mathbf{A}} \right)^{-1} \cdot \mathbf{A}^T \mathbf{b} \quad (8)$$

Compared to the LS solution $(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b}$, the RLS solution can be considered as using the error statistics $\mathbf{C}_{\mathbf{A}}$ as regularization. Stability is improved significantly, especially when \mathbf{A} is nearly singular or ill-conditioned. This is because

inverting \mathbf{A} in the LS formulation will magnify error by \mathbf{A} 's condition number, but adding $\mathbf{C}_{\mathbf{A}}$ will result in a smaller condition number. We further simplify the notation, defining $\mathbf{R}_{\mathbf{A}} = \mathbf{A}^T \mathbf{A} + \mathbf{C}_{\mathbf{A}}$, and have $\hat{\mathbf{x}}_t = \mathbf{R}_{\mathbf{A}}^{-1} \mathbf{A}^T \mathbf{b}$.

In practice, the error statistics $\Delta\mathbf{A}$ obtained from e^v often over-estimates the actual error. This is due to the conservative evaluation of $\mathbf{C}_{\mathbf{A}}$ as described in App. B. To compensate for that, we use $\mathbf{R}_{\mathbf{A}} = \mathbf{A}^T \mathbf{A} + \alpha \mathbf{C}_{\mathbf{A}}$. Here, α is a regularization parameter between 0 and 1.

3.3 Incremental computation

In the LS above, note that \mathbf{A} and \mathbf{b} are structured: each sensor measurement adds a row to the matrix \mathbf{A} and a corresponding element to the vector \mathbf{b} . This structure can be exploited to simplify computation. In this section, we use the notation \mathbf{A}_n and \mathbf{b}_n , with n being the number of measurements, to illustrate the computational aspect. Let $\mathbf{F}_n = \mathbf{A}_n^T \mathbf{A}_n$ and $\mathbf{g}_n = \mathbf{A}_n^T \mathbf{b}_n$, we can rewrite the LS solution as

$$\hat{\mathbf{x}}_t = \mathbf{F}_n^{-1} \mathbf{g}_n. \quad (9)$$

An additional measurement from a new sensor may be incorporated into the current LS computation by the following incremental update equations

$$\begin{aligned} \mathbf{F}_{n+1} &= \mathbf{F}_n + \mathbf{a}_{n+1} \mathbf{a}_{n+1}^T, \\ \mathbf{g}_{n+1} &= \mathbf{g}_n + \mathbf{a}_{n+1} b_{n+1}. \end{aligned} \quad (10)$$

We have to keep the intermediate states \mathbf{F}_n and \mathbf{g}_n for incremental update. They are constant in size: \mathbf{F}_n is 2×2 and \mathbf{g}_n is 2×1 in 2-D. Note that the sizes do not grow with the number of measurements. The incremental update (10) is clearly any time, i.e., it can terminate at any time after incorporating a reasonable number of measurements, and produce the location estimate $\hat{\mathbf{x}}$ via the matrix operation (9).

In the RLS formulation, the incremental computation can still apply, if we assume the vertex errors are uncorrelated. Instead of keeping the intermediate result for \mathbf{F}_n , we keep $\mathbf{R}_n = \sum_i (\mathbf{a}_i \mathbf{a}_i^T + E[\Delta\mathbf{a}_i \Delta\mathbf{a}_i^T])$, where \mathbf{a}_i is the i -th row of \mathbf{A}_n and $\Delta\mathbf{a}_i$ is the perturbation due to vertex error. (The computation of $E[\Delta\mathbf{a}_i \Delta\mathbf{a}_i^T]$ is presented in App. B.) A new neighbor i contributes to \mathbf{R}_n through an additive term. Hence, the computation can still be done incrementally.

4. ITERATIVE LOCALIZATION AND ERROR CONTROL

Iterative localization schemes often suffer from error propagation, in which nodes with large location error contaminate their neighbors' estimates. Essentially, error propagation is caused by the strategy of using the estimated node locations as pseudo-anchors 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. Moreover, the strategy may be slow to converge which means high communication and computation overhead, or get stuck at local optima which leads to excessive localization error. Fig. 1 shows a typical run where localization gets stuck at a local optimum. Even global optimization schemes are not completely immune to error propagation. For example, the relaxation method of [5] introduces the possibility of error propagation. The existence of error propagation is inherently a by-product of the optimization strategies.

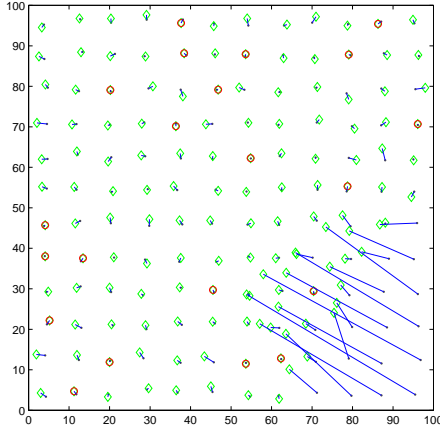


Figure 1: Localization gets stuck at a local optimum. Anchors are marked with circles. Estimated node locations are marked with diamonds, and true locations are plotted as dots.

Similar error propagation problems have been noted in iterative multilateration schemes. Various heuristics are proposed to mitigate the effect. For example, [2] 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 [10] selects spatially spread nodes to form quadrilaterals to minimize localization error. Here, we present an analysis on how localization error is generated, and how it propagates from node to node. This is essential to the design of our error control mechanism.

The error control mechanism comprises the following components:

- *Node registry.* Each node maintains a registry with information sufficient to localize other nodes. In our design the registry not only contains the node location (or location estimate), but also the corresponding estimation error. The error value reflects the level of uncertainty. It is useful in the next round for neighbor selection (see Table 1).
- *Neighbor selection.* This differentiates nodes based on the overall error in vertices and edges. Nodes with high overall errors are excluded from the neighborhood and not used to localize others. This prevents errors from propagating.
- *Update criterion.* At each iteration, if a new estimated location has error larger than the current error and a predefined threshold, the new estimate is discarded. This conditional update criterion prevents error from accumulating.

4.1 Error characterization

For any node t , its registry stores information $(\hat{\mathbf{x}}_t, \hat{e}_t)$, where $\hat{\mathbf{x}}_t$ is the location estimate (a 2-D vector), and \hat{e}_t is the corresponding error (a scalar). The registry is maintained recursively, as described in Table 1. At the initialization stage, each anchor node is initialized with a registry $(\text{anchor_loc}, \text{anchor_error})$. If the anchor location is exact,

the term *anchor_error* is zero. Otherwise, (e.g., GPS has small error), the *anchor_error* term is initialized to a suitable constant. The free node registries are initialized as $(\text{unknown_loc}, \infty)$.

The error in RLS estimation is characterized as follows. Given the selected neighborhood \mathcal{N} with corresponding error $\{e^v, e^e\}$, we need to compute the estimation error \hat{e}_t . Here, error is measured as the variance $E[\|\hat{\mathbf{x}}_t - \mu_{\mathbf{x}_t}\|^2]$ ($\mu_{\mathbf{x}_t}$ is the mean). This l_2 metric is easy to compute. Other metrics may also be considered, but the mathematical analysis may be different.

Following the notations in Sec. 3, 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, i.e., \mathbf{A} is certain, the error due to \mathbf{b} can be characterized as follows:

$$\begin{aligned} E\|e_{\Delta\mathbf{b}}\|^2 &= E\|\mathbf{R}_\mathbf{A}^{-1}\mathbf{A}^T\Delta\mathbf{b}\|^2 \\ &= E\left[\Delta\mathbf{b}^T\mathbf{A}\mathbf{R}_\mathbf{A}^{-T}\mathbf{R}_\mathbf{A}^{-1}\mathbf{A}^T\Delta\mathbf{b}\right] \\ &= \text{trace}[\mathbf{A}\mathbf{R}_\mathbf{A}^{-T}\mathbf{R}_\mathbf{A}^{-1}\mathbf{A}^T\text{Cov}(\Delta\mathbf{b})] \end{aligned}$$

Since we know \mathbf{A} , $\mathbf{R}_\mathbf{A}$, and $\text{Cov}(\Delta\mathbf{b})$ (see App. B), this error term can be obtained easily.

Now assume no edge errors and ignore the vertex errors in \mathbf{b} for simplicity, i.e., assume \mathbf{b} is deterministic and known, we can estimate the error due to uncertainties in \mathbf{A} (caused by vertex errors). Note that we can reorganize elements in the \mathbf{A} matrix into a long vector

$\mathbf{a} = (a_{11}, a_{21}, \dots, a_{n1}, a_{12}, a_{22}, \dots, a_{n2})^T$, where the element $a_{ij} = -2(x_{ij} - x_{0,j})$ for $i = 1, \dots, n$ and $j = 1, 2$ (n is the number of equations in (3)). If the error statistics of \mathbf{x}_i 's are known, we can estimate the statistics of Δa_{ij} as well. Let matrix

$$\mathbf{B} \triangleq \begin{pmatrix} b_1 & b_2 & \dots & b_n & 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & b_1 & b_2 & \dots & b_n \end{pmatrix}$$

It is easy to verify that $\mathbf{A}^T\mathbf{b} = \mathbf{B}\mathbf{a}$. In this notation, the estimate is simply $\mathbf{x}_t = \mathbf{R}_\mathbf{A}^{-1}\mathbf{B}\mathbf{a}$, and the error due to \mathbf{a} is

$$\begin{aligned} E\|e_{\Delta\mathbf{a}}\|^2 &= E\|\mathbf{R}_\mathbf{A}^{-1}\mathbf{B}\Delta\mathbf{a}\|^2 \\ &= \text{trace}[\mathbf{B}^T\mathbf{R}_\mathbf{A}^{-T}\mathbf{R}_\mathbf{A}^{-1}\mathbf{B}\text{Cov}(\Delta\mathbf{a})] \end{aligned}$$

The computation of $\text{Cov}(\Delta\mathbf{a})$ is presented in App. B. The total error is the summation of the two terms listed above, as they are assumed to be uncorrelated, i.e., $\hat{e}_t = E\|e_{\Delta\mathbf{b}}\|^2 + \beta E\|e_{\Delta\mathbf{a}}\|^2$, where β is a parameter to compensate for the over-estimation of the error due to \mathbf{a} . Small value of β works well in practice. The overall analysis provides a way of evaluating (2) from edge and vertex errors.

4.2 Neighbor selection

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 , i.e., we compute a $e_{\text{total}}(i) = e_i^v + e_{(i,t)}^e$. The nodes with lower sum are considered preferable. In our implementation, the node with the lowest sum is used as \mathbf{x}_0 . The selection is done by ranking the $e_{\text{total}}(i)$'s in an ascending order, picking the first three nodes, and set 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.

4.3 Update criterion

Even with “high-quality” neighbors with zero vertex error, the estimate could still be arbitrarily bad. For example, if the selected neighborhood consists of three collinear neighbors, the localization is inherently inaccurate. A slight measurement noise can be amplified significantly. Mathematically, the matrix \mathbf{A} is singular for the collinear case, and the error in \mathbf{b} can result in large error in the estimate location. To address this problem, we propose an update criterion. A node updates its registry with new estimate and new error information only when the new error is no larger than the existing one or bounded by a threshold. This heuristic mitigates the arbitrary noise amplification due to bad node geometry.

5. LOCALIZATION SIMULATION

The localization algorithm described above has been validated in simulations. We conducted two sets of simulations, one with large percentages of anchors (10% and 20%), and one with only the minimum number of anchors, three anchors, for networks in 2-D.

Two types of metrics are used to measure localization performance:

- *location error*: $\zeta_x = \sum_i |\hat{\mathbf{x}}_i - \mathbf{x}_i|/N$ measures deviation from ground truth, assuming the ground truth is known;
- *distance error*: $\zeta_d = \sum_j |\hat{d}_j - d_j|/M$ measures the average difference between measured distances (d_j) and the distances calculated from the estimated sensor positions (\hat{d}_j). This measures data fitting accuracy.

If the connectivity is high and the solution is unique, these two errors are closely related.

5.1 Networks with large numbers of anchors

The network is simulated in a $100\text{m} \times 100\text{m}$ field, with 160 nodes placed randomly according to a uniform distribution. Each node has a sensing range of 20m, which is $1/5$ of the total field width. Anchor nodes (marked as circles) are randomly chosen. The standard deviation of anchor nodes is 0.5m in horizontal and vertical directions and the white noise variance of distance measurements is 1.5m^2 .

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 registry described in Sec. 4. 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: 6 lost runs with 10% anchor nodes and 1 lost run with 20% anchors.

Fig. 2 reports the localization accuracy (with 10% anchor nodes) at the beginning of each iteration, with accuracy

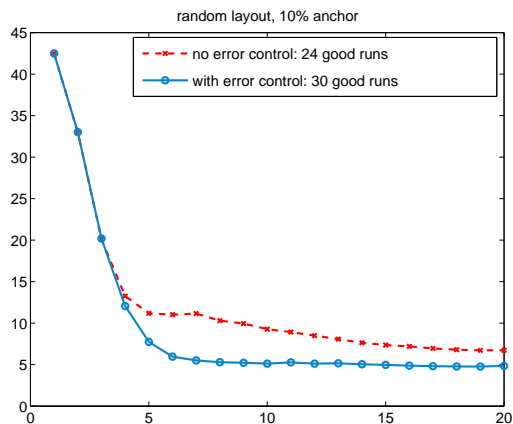


Figure 2: Localization accuracy for random network layout with 10% randomly placed anchor nodes. The horizontal axis is the number of iterations; the vertical axis is the average distance ζ_x between location estimates and ground truth, measured in meters.

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 are 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. Fig. 2 shows that seven iterations in the scheme without error control produces a localization accuracy of about 11m. With error control, the localization accuracy improves to about 6m. 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. Intuitively, when the percentage is low, the effect of error propagation is significant, and hence the benefit of error control.

Now we examine the computational complexity of the localization scheme. The LS/RLS localization without error control works very fast. Computation can be distributed to each free node to localize itself with respect to its known neighbors. As we have commented in Sec. 3.3, data representation is concise, with the \mathbf{F}_n matrices being 2×2 and the \mathbf{g}_n vectors being 2×1 . In the random grid layout described above, each node has on average $15 \sim 16$ neighbors. The overall computation per node at each iteration is on the order of $200 \sim 300$ real number operations (multiplication and additions).

With the error control mechanism described in Sec. 4, the estimation error term in node registry need to be updated at nearly every iteration. We have simulated local-

ization using MATLAB on a 1.8GHz Pentium II personal computer. Without error control, each node takes about 1.2ms per iteration. With error control, each iteration takes about 2.5ms. Hence the overall computational cost with error control roughly doubles. However, the improvement to location estimation due to error control is clearly worthwhile. Furthermore, as shown in Fig. 2, the error control method takes far fewer iterations to converge to the same accuracy level. From the storage point of view, the node registries are insignificant: compared to the case without error control, each node only incurs the additional storage cost of a scalar \hat{e}_t .

Similar results have been observed in localization of a network of RSS sensors.

5.2 Performance comparisons with other algorithms

There have been a number of other localization algorithms proposed in the literature. Here, we refer to our scheme as the ILS (incremental least-squares based) method. We compare the following methods:

- *ILS*: error controlled ILS with shortest path approximation in initialization.
- *ILSnspa*: error controlled ILS without shortest path approximation.
- *MDS-MAP*: localization based on multi-scaling using connectivity data [3]. It is very robust to noise and low connectivity.
- *SDP*: localization based on semi-definite programming [5], working well for anisotropic networks.
- *SPA*: localization using shortest-path length between node pairs [16]. 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. Two sets of outputs are produced to indicate the quality of localization algorithms: **error histogram**, which shows the overall node localization accuracy among all the random runs, and **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.

Figs 3 (a) and (b) show the localization accuracy (ζ_x) of these five methods 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 in a certain range, e.g., smaller than 1.5, between 1.5 and 2.5, and so on. We favor method with long bar for error < 1.5 . Table 2 shows the number of the best results over 100 instances for 10% and 20% anchors, respectively. The key observations from these results are:

(1). ILS with shortest path approximation works best overall and the simple shortest path approximation (without further iterations) is worst.

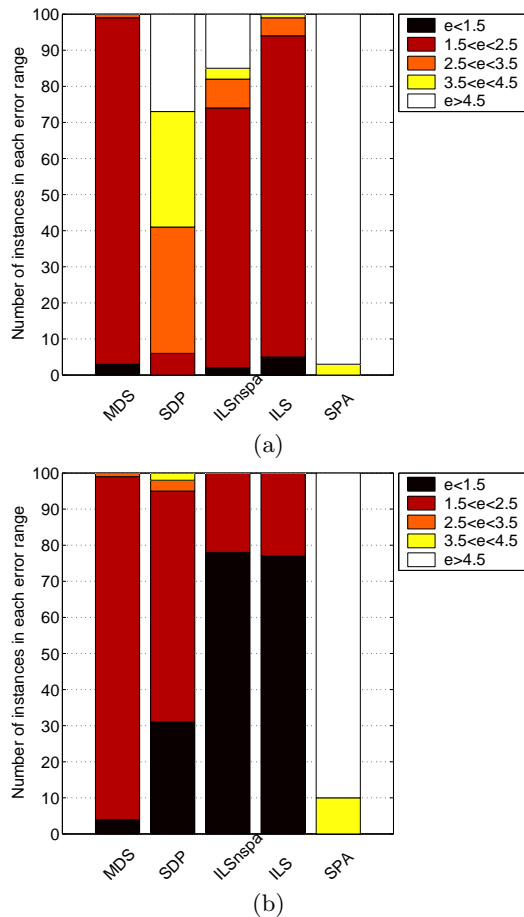


Figure 3: Error histograms: (a) 10% anchors and (b) 20% anchors

(2). More anchor nodes produce better results for most algorithms. When percentage of anchors is small (10%), ILS without shortest path approximation does not work well. MDS is relatively better in small number of anchor nodes and SDP is relatively better in large number of anchor nodes.

Anchor perc.	MDS	SDP	ILSnspa	ILS	SPA
10%	39	0	20	42	0
20%	0	6	35	64	0

Table 2: Instances of best localization results over 100 randomly generated test data for networks with large number of anchors. Bold entries highlight the best performance for each case.

5.3 Networks with three anchor nodes only

For the second test, we have four scenarios (Figure 4). These scenarios are generated as follows: Scenario (a) places nodes on a 7×7 grid within a $100\text{m} \times 100\text{m}$ area. Each node location is then jittered by a uniform noise with an amplitude of 20% of the grid size. Scenario (b) is similar to (a) except that the uniform noise has a larger amplitude (100% of the grid size). Scenario (c) or (d) is similar to (a) except some nodes are turned off to make a U or L shape. These

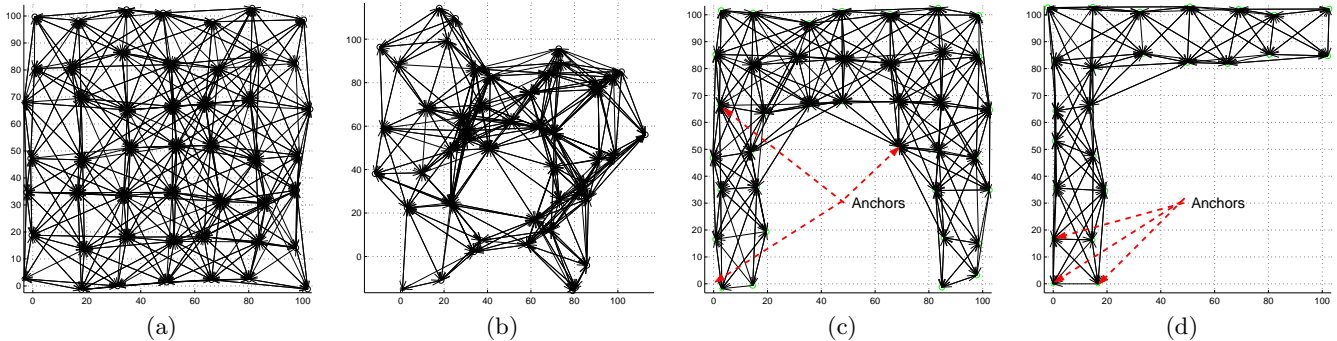


Figure 4: Four test scenarios: (a) small jitter; (b) large jitter (c) U shape and (d) L shape.

Scenarios	MDS	SDP	ILS _{nec}	ILS	SPA
(a)	19	0	0	86	0
(b)	24	0	6	70	0
(c)	4	3	10	84	0
(d)	0	1	56	43	0

Table 3: Instances of best localization results for localization simulations with 3 anchors.

shapes are designed to test the effect of shortest path approximation. Two nodes are connected, with a 0.9 probability, if the distance is no greater than 40m. For all scenarios, variance of measurement errors are set to be 1. For scenarios (a), (b) and (c), three anchors are randomly selected, and for scenario (d), three anchors are at the left-bottom corner.

To show the effectiveness of error control, ILS with or without error control (ILS_{nec}) are both tested. Figs 5 (a) – (d) show localization results in terms of distance error (ζ_d) distributions of these five methods for scenarios (a) – (d), respectively, and Table 3 shows the percentage of the best cases over 100 instances for (a) – (d), respectively. We have the following key observations from these results.

(1). For all algorithms, the result accuracies are ordered by (a), (b), (c) and (d), i.e., (a) has the best results and (d) is the worst. ILS with error control works best for all scenarios except (d), although it still has the highest percentage of small errors.

(2). MDS does not work well for scenario (d), and SDP is relatively better than MDS in (d). SPA still has the worst performance in all cases. ILS without error control performs quite well, especially in scenario (d), where it wins the number of best cases. This is due to the fact that the error estimation for shortest path approximation is way off due to anchors located at the end of the L-shape.

6. TESTBED EXPERIMENTS

We have tested ILS on our sensor hardware. In the experiment, sixteen nodes are placed in a 4×4 grid on the floor, with the grid spacing of 48 inches (Fig. 6a). Note that the algorithm works for arbitrary configurations. We use grid configuration as it is easy to measure ground truth and visualize the estimated node displacement due to localization error.

Each node is a Berkeley mote (Mica2), augmented with an ultrasonic ranging sensor that detects distance between

two nodes by measuring time of flight of ultrasonic pulses. Fig. 6b shows an individual node. This device can accurately measure distances up to 180 inches (15 feet), with error bounded by 3-4 inches, in an open environment. However, for large distances in indoor environments, the measurement is particularly sensitive to reflections off walls and ceilings.

In the experiment, ranging is performed in the Mica2 nodes, and the range data is collected at the base station, with a host connected to the base station performing the localization computation. Some noise reduction techniques have been applied to the raw data to remove white noise, asymmetry and bad connections [12]. It should be noted that this implementation is for the convenience of repeated data collection experiments, and the localization algorithm could have been implemented in a distributed manner.

6.1 Results and Comparisons

We have done six sets of experiments using the 4×4 setting (Fig. 6): the first four use four anchor nodes and the last two use three anchor nodes. A sample of the connectivity graph and range error histogram are in Fig. 7 (a) and (b), respectively.

For each data set, we have performed the following types of ILS localization:

1. **no EC**: ILS without error control.
2. **h. EC**: ILS with error control using a homogeneous noise model: $z = d + \epsilon$, where ϵ is Gaussian $N(0, \sigma)$. In our experiments, $\sigma = 1.7$ inches.
3. **nh. EC**: ILS with error control with a non-homogeneous noise model:

$$z = d + \epsilon_1, \text{ if } d < d_0; \quad d + \epsilon_2, \text{ otherwise.}$$

where d_0 is a threshold, ϵ_1 is $N(0, \sigma_1)$ and ϵ_2 is $N(0, \sigma_2)$. In our experiment, $d_0 = 120$ inches, $\sigma_1 = \sigma$ and $\sigma_2 = K\sigma$ where K is a large number (10^6). Intuitively, this noise model is obtained by observing that when distances are above certain threshold, the probability of getting bad data due to reflection increases dramatically. Such measurements will be discounted by the neighbor selection approach described in Sec. 4. With this non-homogeneous model, we are effectively avoiding using long range measurements unless there are no other sensors available.

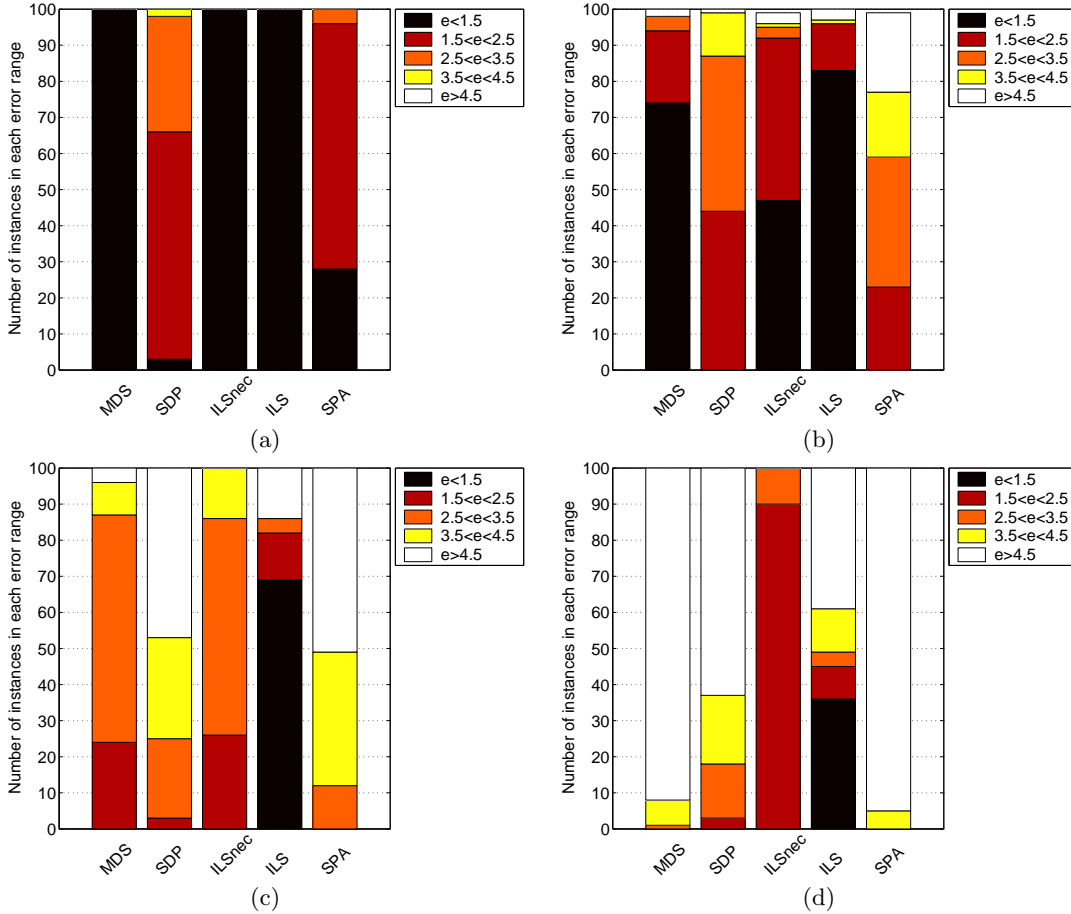


Figure 5: Error distributions: (a) small jitter; (b) large jitter (c) U shape and (d) L shape.

Data sets	no EC	h. EC	nh. EC	MDS	SDP	SPA
1	12.9	9.6	3.4	2.4	5.1	21.9
2	7.1	5.7	3.3	4.7	5.9	12.5
3	8.3	7.5	8.9	7.5	8.5	14.6
4	6.6	7.1	3.4	5.8	3.9	11.4
5	16.5	12.3	4.4	5.1	7.1	16.7
6	22.0	10.8	3.9	3.3	5.0	16.1

Table 4: Comparisons of localization methods. The reported data are in inches. Bold entries highlight the best performance for each data set

As with the simulation, we compare the performance of ILS with three other methods, MDS, SDP and SPA, on six data sets. Table 4 gives the location errors for these methods. MDS and SDP perform quite well overall. As with simulation, using error control with a suitable noise model, ILS can match their performance, and sometimes outperform them. The result of ILS on data set (1) is shown in Fig. 7 (c). On the other hand, choosing the right noise model is essential. This is not surprising, since error modeling and control is key to distributed localization, as described in Sec. 4.

6.2 Validating the Gaussian assumption

In the experiments above, we have assumed that the error in z_i is Gaussian. Here, we validate this assumption using real data. Fig. 7(b) shows the histogram of measurement error and its Gaussian fit. The measurement error is obtained by comparing to the ground truth in our testbed experiment. Visually the error has the bell-shape, which matches Gaussian pretty well. Furthermore, to assess the quality of Gaussian fit, we analyze high order moments.¹ If the data is truly Gaussian, it should match moments of all orders, hence we will have $EX^2 = \sigma^2$, $EX^4 = 3\sigma^4$, $EX^6 = 15\sigma^6$, and $EX^8 = 105\sigma^8$. Based on real data, we compute the empirical moments $m_K = \frac{1}{N} \sum_j e_j^K$ for $K = 2, 4, 6, 8$. Respectively they are:

- $m_2 = 4.51$, suggesting that $\sigma = 2.1$;
- $m_4 = 50.0$, suggesting that $\sigma = 2.0$;
- $m_6 = 800.5$, suggesting that $\sigma = 1.9$;
- $m_8 = 14698$, suggesting that $\sigma = 1.9$.

The fact that the high order moments are consistent is an additional piece of evidence that Gaussian fit is appropriate. One can perform more elaborate statistical test, such

¹We only need to consider the even-th order moments. For Gaussian with zero mean, all the odd-th order moments are zero.

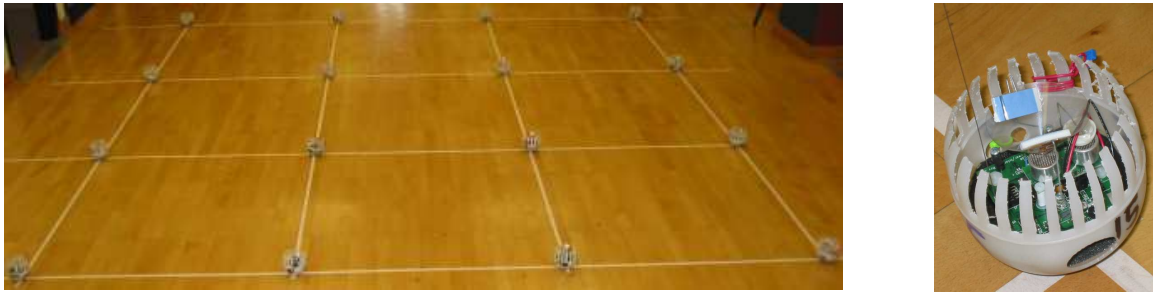


Figure 6: (a) Left panel: node deployment in a 4×4 grid; (b) Right panel: sensor node: Berkeley mote with ultrasonic ranging device.

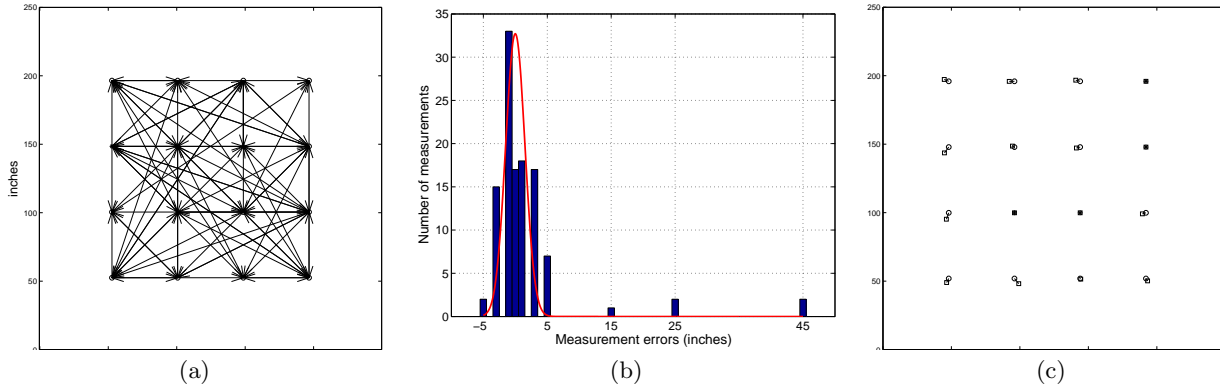


Figure 7: (a) Sample connectivity, (b) range measurement error histogram and its Gaussian fit, and (c) localization result of ILS with true locations as circles and estimates as squares.

nh. EC	$\sigma^2 = 1$	$\sigma^2 = 3$	$\sigma^2 = 6$	$\sigma^2 = 12$
Avg.Loc.Error	5.8	4.5	7.6	6.1

Table 5: Average localization error of ILS (nh EC) assuming varying choice of σ^2 .

as the chi-square test, to validate the Gaussian assumption, but that requires more data points than is available in this experiment.

A question comes up in practice: if we apply the ILS localization approach, how sensitive is the localization performance with respect to the choice of σ ? This concern is valid because the exact value of σ is not available, and one can only estimate with some accuracy. To answer this question, we vary our choice of σ and list the localization performance in Table 5. The algorithm assumes $\sigma^2 = 1, 3, 6, 12$, while the true variance is around 4. The localization error is obtained by averaging over all 6 data sets. From this table we can see that with the variance estimate off by a factor of 1/4 to 3, the localization performance varies less prominently (within a factor of 1.7). The overall performance is comparable to MDS and SDP.

7. DISCUSSION

Here we assess the ILS localization approach by its advantages and limitations. The main advantages include:

(1) **Locality in storage and computation.** ILS is a local method, and achieves comparable localization perfor-

mance as the more global methods such as MDS and SDP. The node registries are compact, each containing only a location vector and an estimation error term. The registries are stored distributedly on the individual nodes. Each node broadcasts its registry to its neighbors whenever the content of the registry changes. No central computation or storage is required. The initial network discovery phase of computing shortest path to nearby anchors can be done via local computation and storage as well.

(2) **Light-weight computation.** Computing location estimate using LS or RLS is fast. Computation can be distributed to nodes to localize themselves in parallel. As we have commented in Sec. 3, the data representation is concise, and the computation is incremental and efficient. Note that although we have used 2-D examples, the algorithm applies to 3-D problems as well.

It is also important to understand the limitation of ILS. An LS method is optimal when the perturbation to b_i 's are iid (independent identically distributed) Gaussian. RLS makes the additional assumption that the perturbation to \mathbf{A} is also independent to the perturbation to b_i 's. Furthermore, as explained in App. C, we assume that perturbation to \mathbf{A} and \mathbf{b} are uncorrelated, which is appropriate for the case when distance between sensors is much larger than the sensor node location error. However, it is possible that measurement errors are not independent, but rather affected by some systematic skew. In this case, LS and RLS loses optimality. If the sensors are low SNR sensors (such as under-water sensors or sensors in an environment with large amount of obstacles), the RLS uncorrelation assumption is not applica-

ble, and we are driven towards more conservative estimates. In comparison, the well-known Kalman filtering approaches (such as in [2]) assume linear Gaussian observation model and linear dynamics, but the perturbation does not have to be independent or identically distributed. Particle filter approaches (such as in [17]) use non-parametric models and even eliminate the linear Gaussian assumptions. The more sophisticated probabilistic models come at a higher computational cost. For example, Kalman filter explicitly maintains estimate prediction and Kalman gains at every step. The complexity of particle filter is often very high due to the need to maintain an adequate sample of particles.

Finally, we comment on the generality of error control. Error propagation is a generic problem in iterative estimation. We have proposed to augment an estimate with some measure of estimation quality, so that one can distinguish bad data from good ones in iterations. This idea is not restricted to the specific sensing modalities considered in this paper. As explained in Sec. 4, we have designed an error control scheme consisted of three major steps: (1) error characterization in node registry, (2) neighbor selection, and (3) update criteria. Although the detailed mathematical forms of error are derived for range sensors, the three-step approach can be readily extended to other sensing modalities. In this sense, the concept and mechanism of error control are general.

8. CONCLUSION

We have presented a light-weight algorithm for node localization. Location estimation using LS/RLS is any time, efficient, and suitable for implementation on resource-limited nodes. The novel error control mechanism mitigates the effect of error propagation in distributed iterative computation. The algorithm has been analyzed using simulation and tested with real data from ultrasound TOA on Berkeley Mica2 motes.

APPENDIX

A. ERROR CHARACTERIZATION FOR SHORTEST PATH APPROXIMATION

At the initialization stage, we use the shortest path length as an approximation to Euclidean distance. This introduces an “edge” error. Without loss of generality, we assume that the shortest path starts at $(0,0)$, and traverses through a set of edges with length $\{d_1, d_2, \dots, d_H\}$, where H is the number of hops in the shortest path. This can be modeled as a random walk, where each edge take an independent random direction θ_i , uniformly distributed over $[0, 2\pi]$, with length d_i . For simplicity, we first assume that all the d_i 's are noise free. After H segments, the random walk finishes at location (x, y) , with $x = \sum_{i=1, \dots, H} d_i \cos(\theta_i)$ and $y = \sum_{i=1, \dots, H} d_i \sin(\theta_i)$. We are interested in the Euclidean distance between the starting point (the origin) and the ending point, i.e., $r = \|(x, y)\|$. To simplify the analysis, we use the Rayleigh distribution to approximate r . To compute the error of shortest path approximation, we are interested in

$$\begin{aligned} \mathcal{V} &= E \left[\left(\sum_i d_i - r \right)^2 \right] \\ &= \left(\sum_i d_i \right)^2 - 2Er \cdot \sum_i d_i + Er^2 \end{aligned}$$

The statistics of r (Er and Er^2) can be derived from the uniform assumption of θ_i 's. For space reasons, we omit the detailed derivation, and only present the result here:

$$\mathcal{V} = \left(\sum_i d_i \right)^2 - \left(\sum_i d_i \right) \sqrt{\pi \sum_i d_i^2 + \sum_i d_i^2}. \quad (11)$$

In practice, \mathcal{V} is often an over-estimate. This is due to our conservative assumption that the shortest path is a complete random walk. In practice, we tune this down, weighting \mathcal{V} by a small factor. Estimating \mathcal{V} more realistically may require numerical simulation, which we leave as a topic of future investigation.

Now note that all the observed edge length d_i 's are actually noisy with variance σ^2 , hence we have to evaluate \mathcal{V} with respect to the distribution of d_i 's. From (11), we can see that the error due to distance measurement is bounded from above by $2H\sigma^2$. Therefore, the overall error is roughly $\mathcal{V} + 2H\sigma^2$. \square

B. CHARACTERIZING $\Delta \mathbf{A}$ AND $\Delta \mathbf{B}$

From the illustration in Sec. 3, we have $\mathbf{C}_A = \sum_i E[\Delta \mathbf{a}_i \cdot \Delta \mathbf{a}_i^T]$ and $\mathbf{a}_i = -2(\mathbf{x}_i - \mathbf{x}_0)$. Note that it is a 2×1 vector. The autocorrelation takes the form

$$\begin{aligned} E[\Delta \mathbf{a}_i \Delta \mathbf{a}_i^T] &= \begin{bmatrix} E[\Delta a_{i,1}^2] & E[\Delta a_{i,1} \Delta a_{i,2}] \\ E[\Delta a_{i,1} \Delta a_{i,2}] & E[\Delta a_{i,2}^2] \end{bmatrix} \\ &= 4 \cdot \begin{bmatrix} e_i^v & 0 \\ 0 & e_i^v \end{bmatrix} \end{aligned}$$

The second equation uses the approximation that the horizontal and vertical coordinates of \mathbf{x}_i are independent, each with covariance e_i^v . This is clearly a bit conservative.

For $b_i = (f^2(z_i) - f^2(z_0)) - (\|\mathbf{x}_i\|^2 - \|\mathbf{x}_0\|^2)$, as discussed earlier in Sec. 3, we ignore error in the norm term, and focus on error in z_i . For TOA, if z_i is contaminated with iid Gaussian noise $N(0, \sigma^2)$, it is easy to show that $E(\Delta b_i^2) = (2\sigma^4 + 4z_i^2\sigma^2)/c^2$. For RSS, we have $E(\Delta b_i^2) = \eta^2\sigma^2/z_i^4$.

C. CORRELATION \mathbf{R}_{AB}

Recall some facts from Sec. 3.2:

- $\mathbf{r}_{Ab} = E[\Delta \mathbf{A}^T \Delta \mathbf{b}]$;
- \mathbf{A} is a $n \times 2$ matrix whose i -th row is $\mathbf{a}_i^T = -2(\mathbf{x}_i - \mathbf{x}_0)^T$;
- \mathbf{b} is a $n \times 1$ column vector whose i -th element is $b_i = (f^2(z_i) - f^2(z_0)) - (\|\mathbf{x}_i\|^2 - \|\mathbf{x}_0\|^2)$.

It is easy to show that $\mathbf{r}_{Ab} = \sum_{i=1}^n E[\Delta \mathbf{a}_i^T \Delta b_i]$. Likewise, the data term is $\mathbf{A}^T \mathbf{b} = \sum_i \mathbf{a}_i^T b_i$. This appendix computes the value of \mathbf{r}_{Ab} and show that often it is negligible compared to the data term, i.e., $\mathbf{r}_{Ab} \ll \mathbf{A}^T \mathbf{b}$.

Without loss of generality, we assume the reference sensor is located at the origin, i.e., $\mathbf{x}_0 = 0$. Now we examine the contribution from the i -th sensor $E[\Delta \mathbf{a}_i^T \Delta b_i]$. Note that $\Delta \mathbf{a}_i$ is purely vertex error, while Δb_i is due to both edge error (through the z terms) and vertex error (through the $\|\mathbf{x}_i\|$ term). We assume that the error in z_i is independent of the error in \mathbf{x}_i . This is often a valid assumption due to sensing physics. Hence we only need to compute the

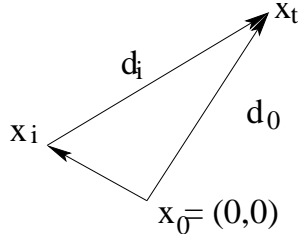


Figure 8: Node geometry.

θ	0	0.1π	0.2π	0.3π	0.4π	0.5π
$ b_i $	$4d_i^2$	$3.6d_i^2$	$2.6d_i^2$	$1.4d_i^2$	$0.4d_i^2$	0

Table 6: Varying angle θ and calculating b_i

correlation due to the \mathbf{x}_i term:

$$\begin{aligned}
 E[\Delta \mathbf{a}_i^T \Delta b_i] &= E[2\Delta \mathbf{x}_i \cdot (|\mathbf{x}_i + \Delta \mathbf{x}_i|^2 - |\mathbf{x}_i|^2)] \\
 &= 2E[\Delta \mathbf{x}_i \cdot (2\mathbf{x}_i^T \Delta \mathbf{x}_i + \Delta \mathbf{x}_i^T \Delta \mathbf{x}_i)] \\
 &= \begin{pmatrix} 4x_{i1}\sigma_{i1}^2 \\ 4x_{i2}\sigma_{i2}^2 \end{pmatrix}, \quad (12)
 \end{aligned}$$

where x_{i1} and x_{i2} are the x- and y- coordinate of sensor i , and σ_{i1}^2 and σ_{i2}^2 are the variance in x- and y- directions, respectively. When node error is small (for example, for anchor nodes), this correlation is small.

Now we compare the error correlation term (12) with the corresponding data term $\mathbf{a}_i^T b_i$ and show that the latter is often much larger. Note that

$$\mathbf{a}_i^T b_i = 2\mathbf{x}_i \cdot (d_i^2 - d_0^2 - |\mathbf{x}_i|^2), \quad (13)$$

where d_i is the shorthand notation for $f(z_i)$, which is the distance between sensor i and the target. The respective terms are labeled in Fig. 8. It is easy to see b_i is zero only when the vector \mathbf{x}_i is orthogonal to \mathbf{x}_t . But this orthogonal geometry is rather rare case. To get a gross feeling about how big b_i is, we do the back-of-the-envelope calculation, assuming \mathbf{x}_i and \mathbf{x}_t forms an angle θ and $d_i \approx d_0$. Table 6 lists the calculated b_i value for a number of θ values. The table shows that b_i is often a factor of 0.3 to 4 of the squared distance d_i^2 . Given that the absolute distance d_i is often much bigger than vertex error, b_i is much larger than the edge error σ_{i1}^2 and σ_{i2}^2 .

As a quick verification, we examine the above result in real experiments. In the testbed setting of Sec. 6, d_i is on the order of 40 to 150 inches, hence the measured b_i is on the order of $10^3 \sim 10^4$. The vertex error σ_i is much smaller (< 10 inches), hence the error correlation term $\mathbf{r}_{\mathbf{A}\mathbf{b}}$ is negligible compared to the data term $\mathbf{A}^T \mathbf{b}$.

Acknowledgment

This work is partly funded by the Defense Advanced Research Project Agency (DARPA) contract # F33615-01-C-1904.

D. REFERENCES

- [1] N. Bulusu, J. Heidemann, and D. Estrin, “Gps-less low cost outdoor localization for very small devices,”

- IEEE Personal Communications Magazine*, vol. 7, pp. 28–34, October 2000.
- [2] 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.
- [3] Y. Shang, W. Ruml, Y. Zhang, and M. P. Fromherz, “Localization from connectivity in sensor networks,” *IEEE Trans. on Parallel and Distributed Systems*, vol. 15, pp. 961 – 974, November 2004.
- [4] L. Doherty, K. S. J. Pister, and L. E. Ghaoui, “Convex position estimation in wireless sensor networks,” in *Proceedings of IEEE Infocom*, vol. 3, pp. 1655–1663, April 2001.
- [5] P. Biswas and Y. Ye, “Semidefinite programming for ad hoc wireless sensor network localization,” in *Proc. IPSN 2004*, (Berkeley, CA), April 2004.
- [6] X. Nguyen, M. I. Jordan, and B. Sinopoli, “A kernel-based learning approach to ad-hoc sensor network localization,” *ACM Transactions on Sensor Networks*, vol. 1, August 2005.
- [7] D. Niculescu and B. Nath, “Ad hoc positioning system (APS),” in *GLOBECOM (1)*, pp. 2926–2931, 2001.
- [8] C. Savarese, J. Rabaey, and K. Langendoen, “Robust positioning algorithms for distributed ad-hoc wireless sensor networks,” in *USENIX Technical Annual conference*, (Monterey, CA), June 2002.
- [9] L. Kleinrock and J. Silvester, “Optimum transmission radii for packet radio networks or why six is a magic number,” in *Proc. IEEE National Telecommunications Conference*, pp. 4.3.1–4.3.5, 1978.
- [10] D. Moore, J. Leonard, D. Rus, and S. Teller, “Robust distributed network localization with noisy range measurements,” in *Sensys*, November 2004.
- [11] H. Lim and J. Hou, “Localization for anisotropic sensor networks,” in *Proc. IEEE InfoComm05*, (Miami, FL), 2005.
- [12] Y. Zhang, M. Yim, L. Ackerson, D. Duff, and C. Eldershaw, “Stam: A system of tracking and mapping in real environments,” *IEEE, Wireless Communications Magazine*, December 2004.
- [13] J. A. Costa, N. Patwari, and A. Hero, “Distributed weighted-multidimensional scaling for node localization in sensor networks,” *ACM Transactions on Sensor Networks*, 2006. to appear.
- [14] L. E. Ghaoui and H. Lebret, “Robust solutions to least-squares problems with uncertain data,” *SIAM Journal on Matrix Analysis and Applications*, vol. 18, no. 4, pp. 1035–1064, 1997.
- [15] H. Hindi and S. Boyd, “Robust solutions to l_1 , l_2 , and l_∞ uncertain linear approximation problems using convex optimization,” in *Proceedings of the American Control Conference*, vol. 6, pp. 3487–3491, 1998.
- [16] K. Whitehouse, “Calamari: A localization system for sensor networks,” 2003. <http://www.cs.berkeley.edu/~kamin/calamari>.
- [17] A. Ihler, J. Fisher, R. Moses, and A. Willsky, “Nonparametric belief propagation for self-calibration in sensor networks,” in *Proceedings of Information Processing in Sensor Networks*, (Berkeley, CA), Apr. 2004.