Moving-Baseline Localization

Jun-geun Park Erik D. Demaine Seth Teller
MIT Computer Science and Artificial Intelligence Laboratory
{jgpark, edemaine, teller}@csail.mit.edu

Abstract

The moving-baseline localization (MBL) problem arises when a group of nodes moves through an environment in which no external coordinate reference is available. When group members cannot see or hear one another directly, each node must employ local sensing and inter-device communication to infer the spatial relationship and motion of all other nodes with respect to itself.

We consider a setting in which nodes move with piecewise-linear velocities in the plane, and any node can exchange noisy range estimates with certain sufficiently nearby nodes. We develop a distributed solution to the MBL problem in the plane, in which each node performs robust hyperbola fitting, trilateration with velocity constraints, and subgraph alignment to arrive at a globally consistent view of the network expressed in its own "rest frame." Changes in any node's motion cause deviations between observed and predicted ranges at nearby nodes, triggering revision of the trajectory estimates computed by all nodes.

We implement and analyze our algorithm in a simulation informed by the characteristics of a commercially available ultra-wideband (UWB) radio, and show that recovering node trajectories, rather than just locations, requires substantially less computation at each node. Finally, we quantify the minimum ranging rate and local network density required for the method's successful operation.

1. Introduction

Location determination is a fundamental problem, attracting human attention since antiquity. Today, GPS (Global Positioning System [8]) infrastructure enables inexpensive hand-held receivers to determine earth-relative position to within a few meters, in outdoor environments with sufficient sky visibility. However, location determination remains an incompletely solved problem in "GPS-denied" environments, where GPS service is unavailable or of low quality: indoors; underground (e.g. in tunnel, bunker, or cave networks); underwater; and in sky-obstructed outdoor environments (e.g. valleys, forests, and urban canyons). Effective location and motion estimation in such environments is the focus of the present paper.

A central goal of localization research and development is to realize a user-borne device capable of reporting the user's location and orientation accurately during excursions of arbitrary length and duration within GPS-denied environments. One strategy is to use inertial sensing to perform dead-reckoning. However even devices incorporating heavy, expensive inertial sensors can incur unbounded position errors of 0.1 percent of the total distance traveled; more typical errors are between one and ten percent [23]. *Relative* position errors between many nodes, each performing dead-reckoning, would diverge even faster. Some GPS-denied localization methods depend upon previously or concurrently deployed infrastructure, such as passive or active fiducial markers or beacons, imposing a deployment burden that is unacceptable or impractical in many application domains.

This paper addresses the problem of determining positions and velocities for a group of devices (or *nodes*) moving within a GPS-denied environment. Like others, we take inspiration from real devices that can measure their *range* to, and communicate with, some subset of other nodes, and we propose a distributed algorithm that reconstructs a globally consistent view of the network derived solely from local observations. However, we depart from previous work in this area by supposing that *all* nodes are in motion, while also assuming no external coordinate reference and no previously deployed infrastructure. This scenario arises from real-world settings in which, for example, a group of people or robots moves cooperatively through a GPS-denied environment to perform some task (e.g., emergency response). We refer to localization methods operating in the absence of a fixed reference frame as *moving-baseline localization* (MBL) methods.

1.1. Algorithmic Setting

We consider an instance of MBL in which each moving node can repeatedly generate a time-stamped measurement of the *range*, or separation distance, between it and certain other sufficiently nearby nodes, and can discover the unique identifier (i.e., integer ID) of, and exchange infor-

978-0-7695-3157-1/08 $25.00 © 2008 IEEE
DOI 10.1109/IPSN.2008.65

IEEE computer society

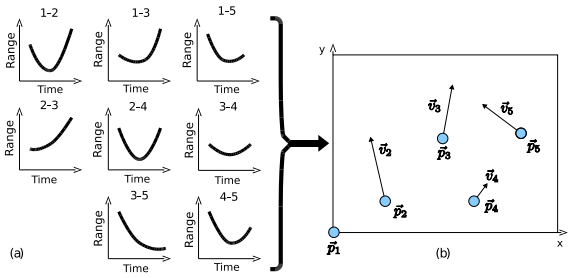


Figure 1. MBL as a local-to-global estimation problem. Available data (a) consists of a time series of range measurements at each node. Problem solution (b) consists of an estimate of all nodes' motions in each node's rest frame (solution for node 1 shown).

mation with, any node to which it can range. This choice of setting is motivated by existing devices with these capabilities, such as Crickets [16] and UWB (ultra-wideband) radios [12]. The problem we face is then to combine a collection of local measurements (time-series range data, with node identifiers) into a single, global estimate of all node motions (Fig. 1). Our method estimates a trajectory for each node that is consistent with recent range measurements involving that node.

We develop a distributed algorithm for MBL in the plane. We start by analyzing the mathematical abstraction in which each node moves with a fixed velocity, then generalize to piecewise-linear trajectories. We assume that range data is inherently noisy, and model ranging noise as a distribution determined by experiments with real UWB devices. We show that MBL can be solved in this setting through robust hyperbola fitting, trilateration, and subgraph alignment. We implement and analyze the algorithm in simulation, and discuss its extension to less restricted settings.

2. Related Work

For static nodes and noise-free ranging, a theoretical foundation for network localization has been elucidated in terms of graph rigidity theory [5], but has not, to our knowledge, been extended to settings in which all nodes are moving. When ranging is noisy, researchers have formulated Kalman filters that incorporate single noisy range or bearing measurements arriving asynchronously, performing "single-constraint-at-a-time" tracking [22]. To combat geometric ambiguity in trilateration, researchers have formulated a uniqueness criterion for network localization, suppressing localization of nodes with ambiguous position solutions [7]. One distributed localization method uses "robust quadrilaterals" (well-shaped 4-cliques in the network ranging graph) to combat noisy ranging, achieving localization accuracy that is a small multiple of ranging noise when all nodes, or all but one node, are fixed [14]. A "mobile-assisted" localization method uses ranges from one moving node to align fixed, localized but otherwise partitioned subclusters [15]. An "anchor-free" localization method treats settings where inter-node hop counts are well-correlated with inter-node metric distance [17]. None of these methods handles all-node motion other than by re-localizing the entire network.

Ranging noise is not, in general, the only source of localization error; solution methods themselves introduce error due to their differing algorithms for computing local (subgraph) embeddings and propagating information about local solutions through the network in order to relate nodes separated by many hops. This phenomenon has been analyzed to produce lower bounds for network localization error [19], and used as a basis for comparison of a variety of localization methods [24].

Researchers have also studied localization for mobile sensor networks. Sequential Monte-Carlo localization, developed for mobile robot localization, was adapted for mobile networks in range-based [4] and range-free [10] settings. Another algorithm maintains the intersections of convex polygons to estimate node locations [3]. However, these approaches assume the existence of anchor nodes or external coordinate references. There is a method that does not rely on anchor nodes [26], but it requires an accelerometer for each node, and does not recover velocity information.

Several MBL methods have been proposed to support autonomous underwater vehicle (AUV) operations. One method integrates acoustic communication and ranging to achieve localization, but requires deployment of three fixed, surveyed beacons to serve as position references [6]. Other researchers equip a subset of AUVs with relatively expensive, high-quality proprioceptive sensors (e.g., inertial measurement units), which transmit their dead-reckoning navigation estimates to cheaper, less-capable vehicles [21]. All vehicles are thus subject to position errors that grow without bound [23]. This approach has also been pursued in the sensor network community, with distributed filtering for fusing position estimates [20].

One MBL method developed to support collaborative autonomous robotics proceeds in rounds, in each round designating some nodes as nonmoving "portable landmarks" while allowing other nodes to move [11]. This framework does not support independent or spontaneous motion, for example by human individuals operating as a team, and introduces communications latency as nodes coordinate their movements. Another method enables all nodes to move simultaneously, but requires that each node be able to observe range and bearing to all other nodes [18]; no known long-range sensor can provide such measurements in the presence of complex occlusion.

The thrust of the present paper differs from the work described above in three significant ways.

First, our primary goal is not to recover a motion estimate for all nodes in an absolute frame, but rather, for each node, motion estimates for all *other* nodes expressed in the frame in which that node is at rest. This goal arises from our desire to provide situational awareness for a person moving, with others, within a GPS-denied space.

Second, we make no use of external coordinates or preferred anchor nodes, nor do we require any infrastructure deployment or configuration prior to localization.

Third, we treat the case in which all nodes are moving, rather than treating each time instant as a separate static localization problem to be solved in isolation, or designating some nodes as fixed and some as moving. We model all nodes as moving along piecewise-linear trajectories, and recover descriptive parameters for those trajectories.

3. Moving-Baseline Localization

We assume that each node: has a unique integer identifier; can discover, range to, and communicate with nearby nodes; and maintains the time t, either locally or through a network synchronization method (e.g. [13]). These pairwise interactions induce a *dynamic network* in which two nodes i and j share an edge ij when and only when they can exchange information. Finally we assume that when ij exists, a discrete sequence of range measurements $r_{ij}(t)$ is available at node i, describing the measured range from node i to node j at time t, as observed at node i (Fig. 2).

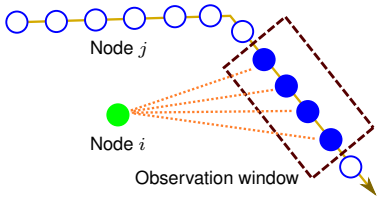


Figure 2. Time-series range data $r_{ij}(t)$.

We start with the simplest instance of MBL: planar motion, with each node moving along a straight-line path at constant speed. We can then cast the problem of recovering node trajectories as a low-dimensional optimization (Fig. 3). Specifically, we must recover four DOFs (degrees of freedom) per node: the quantities $\vec{p}_i$ and $\vec{v}_i$ in the expression

$$\vec{L}_i(t) = \vec{p}_i + t \cdot \vec{v}_i \tag{1}$$

where $\vec{p}_i$ and $\vec{v}_i$ represent the (2-DOF) origin and (2-DOF) velocity vector of the ith node's motion, and $\vec{L}_i(t)$ is the location of that node at time t. The observed ranges $r_{ji}(t)$ between two nodes $\vec{L}_i(t)$ and $\vec{L}_j(t)$ then lie on a hyperbola defined by:

$$r_{ji}(t)^2 = m_{ji}^2 + (t - t_{ji}^c)^2 s_{ji}^2 \tag{2}$$

where t_{ji}^c denotes the time at which nodes i and j make their closest approach, m_{ji} denotes the node separation

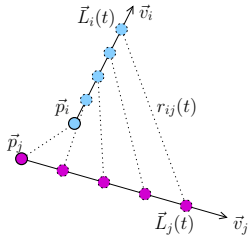


Figure 3. MBL recovers four DOFs per node.

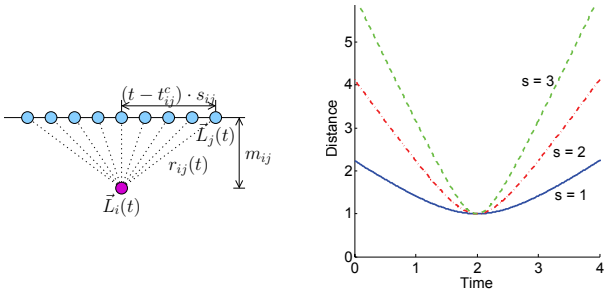


Figure 4. The distance between two points $\vec{L}_i(t)$ and $\vec{L}_j(t)$ moving with constant velocities traces a hyperbola with respect to time t.

distance at this time, and s_{ji} denotes the relative speed $\|\vec{v}_j - \vec{v}_i\|$ (Fig. 4). We define $H_{ij} = (s_{ij}, t_{ij}^c, m_{ij})$ as the *motion hyperbola parameters* for nodes i and j. (Note that $H_{ij} = H_{ji}$, and r_{ji} at any specific time can be computed from Eqn. 2.) Our goal is to construct, from all available motion hyperbola parameters, an optimal global motion solution relative to a global isometry (i.e., an arbitrary rigid translation, rotation, reflection and inertial or constant-velocity coordinate transformation).

We chose to recover node trajectories, rather than estimating all node locations independently (i.e., solving the static problem in isolation) at each time-step, for three reasons. First, our approach requires recovery of fewer parameters ($4N$ versus $2M$ for N nodes, M range measurements, and $M \gg N$). Second, we can use the motion model for both interpolation and prediction, using fewer computational resources and compensating for communication and computation latency at each receiver (and at each user display). Third, we can use the recovered velocities for high-level reasoning, rejecting physically nonsensical motions.

3.1. Overview

Each node estimates motion path geometry in its own inertial coordinate system. We define a *cluster* to be any connected set of nodes, and a *local cluster* as a cluster containing a node and its neighbors. We defined above the motion hyperbola parameters for node j as observed from node i given three or more range samples $r_{ji}(t)$. Once these parameters have been estimated (§ 3.2), each node estimates the relative motion of each of its neighbors (§ 3.3), then constructs a local cluster by aligning computed positions and

velocities (§ 3.4). Each node broadcasts its local cluster solution, enabling every other node to construct its own global view of the network (§ 3.5).

Ranging noise corrupts low-level motion estimation, causing error in the computed localization solution. Each node monitors error by comparing predicted and observed ranges, and reinitiates localization (thus revising its estimates of all other nodes' trajectories) whenever the observed error exceeds a threshold (§ 3.6).

As noted above, in the absence of an external coordinate reference, the most we can hope to recover is some set of motion paths that are consistent with all range measurements, but ambiguous up to an isometry. The isometry's translation, rotation and reflection components can be resolved only with additional information, such as GPS or anchor coordinates at three or more nodes. The inertial ambiguity is not an issue in our setting, because each node's MBL solution is expressed within its own inertial frame.

3.2. Hyperbola Estimation

In this section, we address the problem of estimating the motion hyperbola parameters $H_{ij} = (s_{ij}, t^c_{ij}, m_{ij})$ from time-stamped range measurements. Since the motion model (Eqn. 2) is quadratic, given a sequence of n discrete range observations between two nodes, $(r_z, t_z), z = 1, ..., n$, we consider the following quadratic model

$$r_z^2 = y_z = \gamma t_z^2 + \beta t_z + \alpha + \epsilon_z. \quad (3)$$

Parametric regression methods such as ordinary least-squares estimation are often used to estimate $\hat{\gamma}$, $\hat{\beta}$, and $\hat{\alpha}$. However, ordinary least-squares estimation is not robust when the data contain significant noise and outliers (modeled as ϵ_i), as in our setting. We therefore apply nonparametric robust quadratic fitting [2], which performs well even when the error term is not normally distributed. For $n \geq 3$, the motion hyperbola parameters representing the relative motion of two nodes can be calculated from Eqns. 2 and 3 as:

$$\hat{s} = \sqrt{\hat{\gamma}}; \quad \hat{t}^c = \hat{\beta}/(-2\hat{\gamma}); \quad \hat{m} = \sqrt{\hat{\alpha} - \hat{\beta}^2/(4\hat{\gamma})}.$$

The recovered parameters s, t^c, and m define the slope of the hyperbola's asymptote, the x coordinate, and the y coordinate of the hyperbola's vertex (Fig. 4) respectively. Estimation accuracy increases in general for more samples and for samples closer to the hyperbola vertex (i.e., the time of the nodes' closest approach). After calculating the motion hyperbola parameters, each node communicates them to its 1-hop neighbors so that they can make use of them in estimating their own local clusters.

3.3. Path Estimation Geometry

The parameters estimated in the previous section capture the relative position and motion of a pair of nodes. With

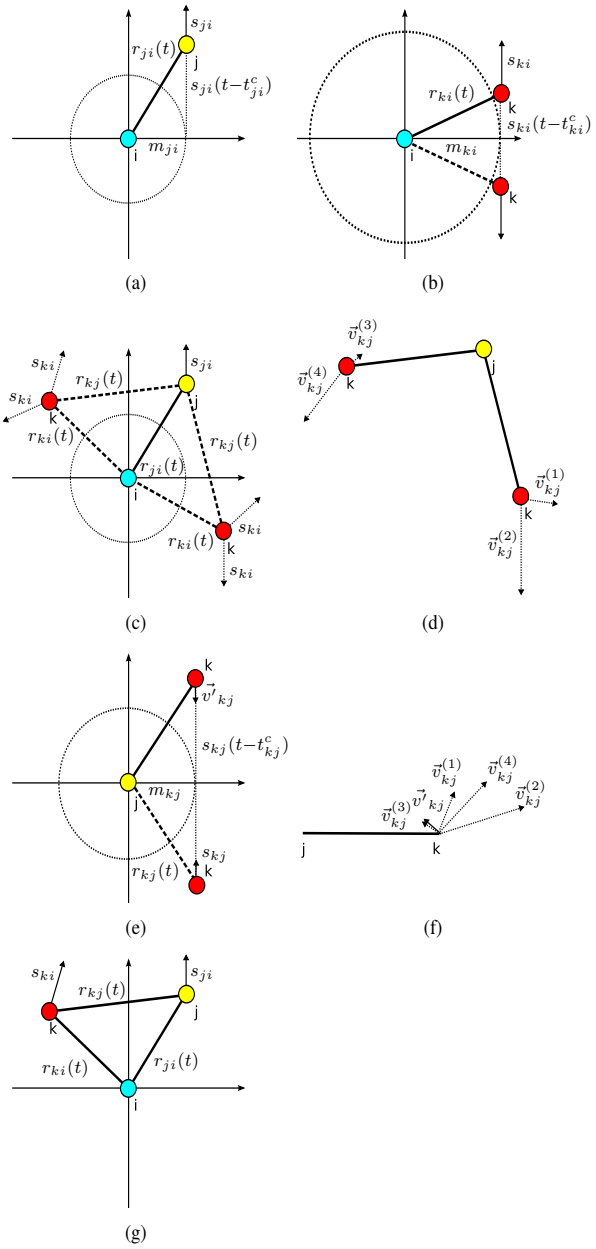


Figure 5. Recovering positions and velocities for nodes i,j,k in the rest frame of node i.

three relations among three nodes i, j, and k, we can infer the relative motion of the node triangle. Our approach, for each node i, amounts to fixing i at its own origin and determining the motions of j and k in i's frame. First, we exploit the fact that, when two nodes i and j move linearly, at the time of closest approach t^c_{ji} the velocity of j with respect to i with magnitude s_{ji} must be tangent to a circle of radius m_{ji} centered at node i (Fig. 5(a)). Therefore, in node i's frame at any time t, node j has position $(m_{ji}, s_{ji}(t - t^c_{ji}))$ and velocity $(0, s_{ji})$.

Likewise, the relative motion between nodes i and k can

be established in i's frame (Fig. 5(b)) up to an unknown reflection.

Now, we apply the remaining distance constraint r_{kj}, which can be calculated from Eqn. 2 when H_{kj} is known, to solve for the position of node k in the frame defined by nodes i and j. This step yields two possible positions for node k, and two possible relative velocities for nodes i and k, giving a total of four possibilities (Fig. 5(c)).

Fig. 5(d) depicts relative velocities of all four cases in Fig. 5(c) with respect to the node j. This ambiguity can be resolved by considering the relationship between nodes j and k, i.e. the H_{kj} (Fig. 5(e)). By comparing the motions $\vec{v}_{kj}^{(l)}$ to $j\vec{k}$, $l = 1, 2, 3, 4$, with k's velocity, $\vec{v'}_{kj}$, to $j\vec{k}$ in Fig. 5(e), one can identify the correct solution.

We do so by decomposing each vector $\vec{v}_{kj}^{(l)}$ in Fig. 5(d) into its vector projection on $j\vec{k}$ and its (nonnegative) orthogonal component, and calculate the magnitude of the corresponding difference vector from the decomposed $\vec{v'}_{kj}$ in the jk frame (Fig. 5(f)). The vector associated with the smallest difference is chosen as the final solution. This disambiguation determines the position and velocity of node k in the frame defined by i and j (Fig. 5(g)).

If the motion hyperbola parameters were exact, this procedure would always select the correct motion. In practice, however, estimation error corrupts the recovered positions and velocities of nodes j and k, making the disambiguation step imperfect. We employ the following heuristic to suppress erroneous estimates:

$$\min_{l=1,2,3,4} \|\vec{v}_{kj}^{(l)} - \vec{v'}_{kj}\| > C_v \tag{4}$$

where C_v is a selection threshold. Any triangle that does not meet this criterion is not used for local cluster construction. Because $\vec{v}_{kj}^{(l)}$ and $\vec{v'}_{kj}$ are both estimated values for which parametric distributions are generally unknown, we selected the 81^{st}-percentile value $C_v = 0.5$ m/s empirically from a Monte Carlo simulation (for 81 percent of the triangle construction steps in the simulation, one of $\vec{v}_{kj}^{(l)}$ matches $\vec{v'}_{kj}$ within 0.5 m/s).

3.4. Local Cluster Localization

The algorithm above constructs each triangle in its own frame. Next, each node localizes its neighbors using a process analogous to chained trilateration.

Let us consider adding a triangle (i, j, k) to a local cluster, given common nodes i and j (Fig. 6). To align i and j, we (1) translate the triangle in order to bring the positions of i into alignment; (2) rotate the frames to bring the positions of j into alignment; (3) determine whether the triangle must be flipped or not, using distances from k to the other nodes in the local cluster; and finally (4) apply a vector offset to equalize the two views of $\vec{v}_i$ and $\vec{v}_j$. By repeatedly aligning triangles along shared edges in an arbitrary spanning tree,

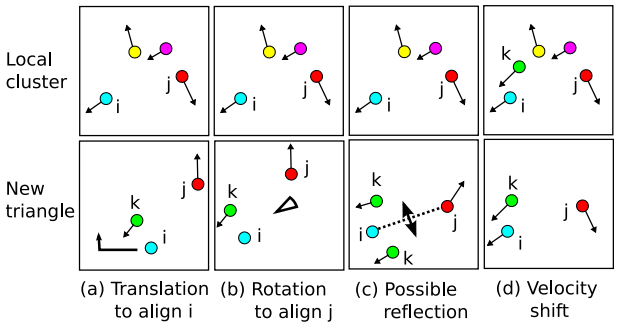


Figure 6. Aligning a local cluster and a new triangle sharing two nodes i and j by (a) translation, (b) rotation, (c) possible reflection, and (d) an inertial frame (velocity) shift.

each node computes a position and motion estimate for all other nodes in the local cluster (pseudocode in Alg. 1).

Algorithm 1 Local cluster localization

```
INPUT Node i: The node self.
INPUT Neighbors: {Neighbors to be localized.}
OUTPUT LocalCluster: {A set of (ID, position, velocity) tuples of localized nodes.}

InNodes = ∅: {Nodes already localized.}
Add arbitrary nodes j_0, k_0 to InNodes.
Initialize LocalCluster as arbitrary triangle (i, j_0, k_0).

for node j ∈ InNodes do
    for node k ∈ Neighbors\InNodes do
        if H_ji, H_ki, H_kj are available then
            Construct triangle (i, j, k).
            Merge (i, j, k) into LocalCluster.
            Add k to InNodes.
        end if
    end for
end for
```

3.5. Global View Construction

To construct a consistent view of the network, we repeatedly find the best alignment for each pair of local clusters that share three or more noncollinear nodes along an arbitrary spanning tree, each time adding a local cluster to the MBL solution. The alignment operation requires extrapolating the cluster to be added to a common time, then finding the translation, rotation, reflection, and velocity offset that transform positions and velocities within one cluster to those within another. An efficient method for solving this "absolute orientation" problem is known [9].

We consider the absolute orientation problem of aligning a cluster (denoted 2) to another cluster (denoted 1). We denote node i's position and velocity in cluster 1 and 2 as

$(P_i^{(1)}, V_i^{(1)})$ and $(P_i^{(2)}, V_i^{(2)})$ respectively. As in local cluster localization, we solve for the Euclidean transformation (translation, rotation, and reflection) and velocity offset separately. We recover the Euclidean transformation from $P^{(2)}$ to $P^{(1)}$ by minimizing the sum of squared residuals

$$(R', T') = \underset{R,T}{\operatorname{argmin}} \sum_i \left\| P_i^{(1)} - R(P_i^{(2)}) - T \right\|^2, \quad (5)$$

where R is a rotation (and possible reflection) and T is a translation. Eqn. 5 can be solved efficiently in closed form by eigen-decomposition [9], in this case of 2×2 matrices.

After solving for (R', T'), we solve

$$\begin{aligned} V' &= \underset{V}{\operatorname{argmin}} \sum_i \left\| V_i^{(1)} - R'(V_i^{(2)}) - V \right\|^2 \\ &= \sum_i V_i^{(1)} - \sum_i R'(V_i^{(2)}) \end{aligned}$$

to yield the velocity offset V' that best shifts velocities in cluster 2 to align with those of cluster 1.

3.6. Local Cluster Update

Using the method above, each node can compute a view of the network using Eqn. 1. The error of any node's view, however, will grow over time as uncertain trajectory estimates are extrapolated in time. One simple way to combat growing error would be for each node to relocalize the network continuously. However, this would waste computation and communication resources when observed and predicted ranges match. In our implementation, each node triggers relocalization whenever the most recent three range measurements disagree with prediction by more than one meter. After relocalization, each node broadcasts its new local cluster solution. (We require three mismatching ranges, rather than just one, to combat false positive relocalization events due to one or two outlier range measurements.)

4. Experimental Results

We evaluated the proposed method on a discrete-event simulator developed using Python and C (Fig. 7).

4.1. UWB Radio Node Characterization

Starting in January 2007, we characterized a current-generation UWB radio, the Time Domain Corporation (TDC) PulsOn 210. While UWB time-of-arrival ranging is generally accurate and resilient to multipath fading [25], large distance errors can occur when line-of-sight between radios is blocked [12] or when the amplitude of an impulse received along a direct path drops below the device's detection threshold [1]. We observed these phenomena during our experiments, and found (following [1]) that we could model the distance errors in practice as a sum of a high-probability small error and a low-probability large error as:

$$\hat{d} = d + \epsilon = d + \epsilon_s + \epsilon_u \quad (6)$$

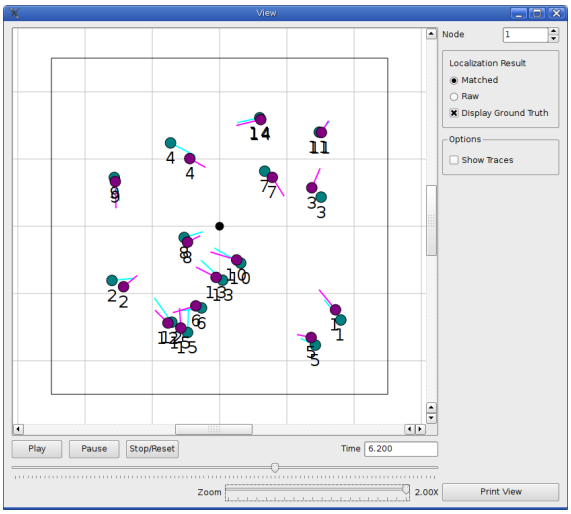


Figure 7. MBL simulator, showing true (green) and estimated (magenta) trajectories.

where $\hat{d}$ is the measured distance, d is the true distance, ϵ_s is the small error, and ϵ_u is the large error. We used this model throughout our simulation studies.

To model ϵ_s and ϵ_u separately, we partitioned observed range errors into small and large errors. We modeled the small error as a Gaussian random variable with a mean dependent on the true distance (Fig. 8(a)) and standard deviation of 3cm (Fig. 8(b)). Therefore, the small error is modeled as $\epsilon_s \sim N(b(d), 0.03)$ with $b(d)$ determined experimentally as

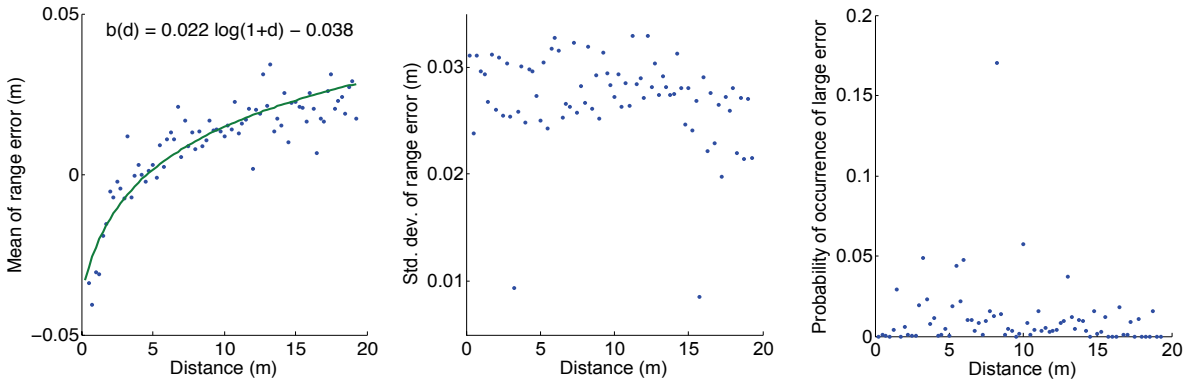
$$b(d) = 0.022 \ln(1 + d) - 0.038.$$

We observed that large errors occurred with low probability and did not follow any characteristic parametric distribution. Thus we modeled the large error simply as a uniform random variable ϵ_l over $[0, 10]$ meters and assigned its probability of occurrence as a binary random variable η: $\epsilon_u = \eta\, \epsilon_l$ where η has probability mass function

$$p_\eta(x) = \begin{cases} P_{OL} & \text{if } x = 1, \\ 1 - P_{OL} & \text{if } x = 0. \end{cases}$$

Here P_{OL} represents the probability of occurrence of large error and is chosen conservatively as 0.05 (Fig. 8(c)).

TDC devices can range within a radius of about 30 meters around obstacles with the above error characteristics. The devices use a time-division scheme to schedule ranging requests, with the ranging protocol requiring about 50 milliseconds, yielding a maximum achievable ranging rate in any vicinity of approximately 20 Hz. Our simulation uses a ranging frequency of 5 Hz to roughly capture the constraint that nearby nodes cannot range simultaneously. We adopted these parameters in our simulation using a unit-disk graph model and a variety of node speeds from slow walking (at



(a) Bias (b) Standard deviation (c) Probability of large error

Figure 8. Ranging behavior of a pair of commercially available UWB devices.

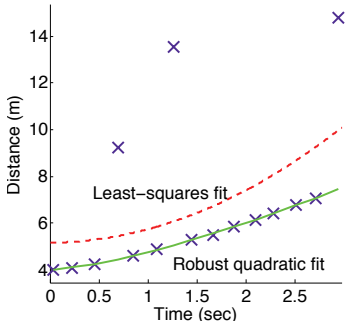


Figure 9. Hyperbola fitting to noisy range data (× marks) with least squares (dashed) and robust quadratic fitting (solid).

0.5 m/s or ∼1 mph), walking (at 1 m/s or ∼2 mph), and jogging (at 3 m/s or ∼6.5 mph).

4.2. Hyperbola Fitting

We used simulated noisy range data drawn from Eqn. 6 to compare the hyperbola fitting method [2] with ordinary least squares (Fig. 9). Due to the extreme outliers, least squares performed poorly, while robust quadratic fitting (§ 3.2) recovered an accurate hyperbola. Table 1 compares the recovered motion parameters to ground truth.

	True	Robust fitting	Least squares
s	1.869	1.874	0.7382
t^c	−0.4492	−0.4381	−0.02734
m	3.890	3.924	2.274

Table 1. Hyperbola estimation with least squares and robust quadratic fitting.

4.3. Trajectory Refinement

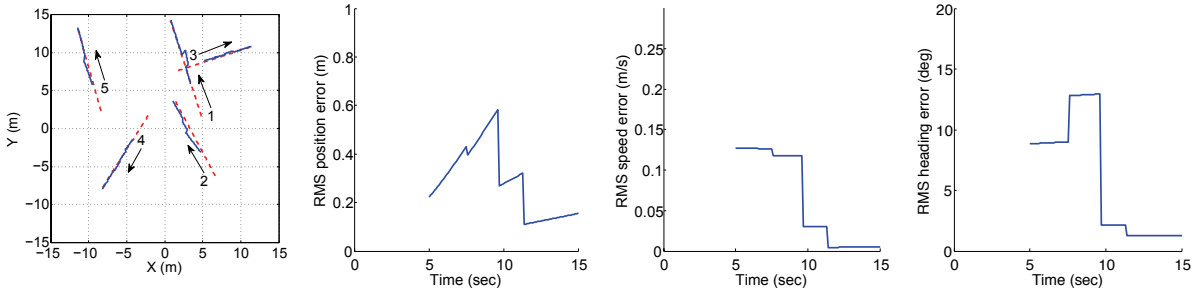
This section illustrates the temporal characteristics of the MBL algorithm in a small-network example (Fig. 10). Fig. 10(a) shows the evolution of node 1's global view (manually aligned to ground truth for comparison). Because the first global view calculation was scheduled to occur at $t = 5$ seconds, trajectories were available only after that time. Trajectory re-estimation was triggered according to prediction error as described earlier. Figures 10(b), 10(c), and 10(d) respectively show the error in position, speed, and heading estimates over time.

4.4. Accuracy, Precision, and Availability

We assessed the performance of our MBL method using three quantitative performance metrics. The *Accuracy* metric characterizes the median error in trajectory estimation with respect to ground truth. The *Precision* metric characterizes the standard deviation in recovered trajectory parameters. The *Availability* metric characterizes the fraction of nodes for which trajectories are successfully recovered.

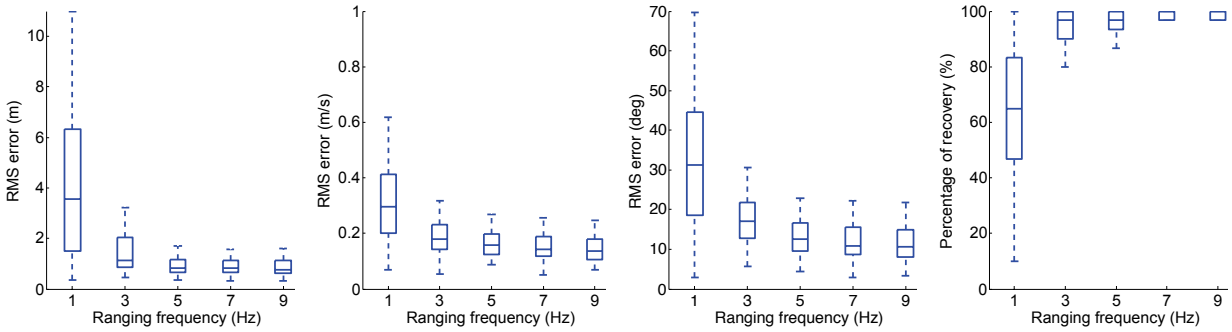
We simulated 50 nodes moving for 15 seconds with a randomly selected constant velocity (uniform in 0.5-1.5 m/s, average 1 m/s) in a degree-15 network while ranging at 5 Hz. Each node gathers range measurements for about 3 seconds and computes an initial view of the network about 5 seconds from the start of simulation. We evaluated the Accuracy and Precision metrics for position, speed, and heading over a variety of ranging sample frequencies, average node speeds, and average node degrees. Both metrics are shown as box plots, in which the box center represents Accuracy and the box height represents Precision. Availability is shown in a separate plot. Each box is computed from 50 Monte-Carlo simulation runs. Because the MBL algorithm revises its trajectory estimates over time, we computed each metric using time-averages.

Our MBL algorithm recovers trajectories with position, speed, and heading RMS error around 1 meter, 0.2 m/s, and 15° respectively, for ranging frequencies above 1 Hz (Fig. 11). Likewise, the algorithm estimates trajectories for almost all nodes when ranging occurs faster than 1 Hz (Fig. 11(d)). For lower ranging frequencies, most nodes were unable to localize, collecting too few range samples



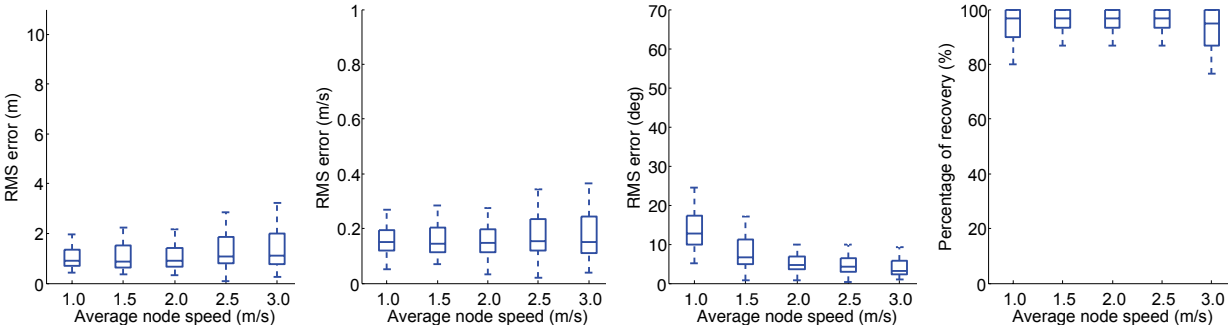
(a) Ground truth (red) and estimated (blue) trajectories (b) Position error (c) Speed error (d) Heading error

Figure 10. A small-network MBL example.



(a) Position error (b) Speed error (c) Heading error (d) Percent nodes recovered

Figure 11. Time-averaged trajectory error and node availability as ranging rate increases.



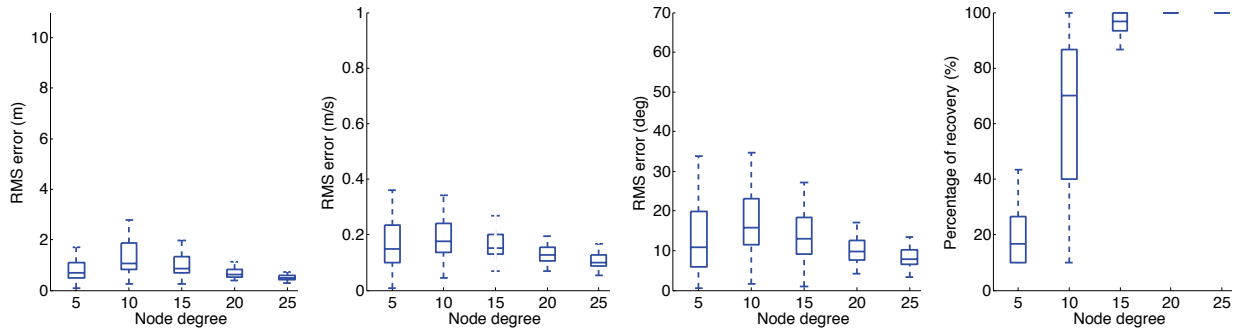
(a) Position error (b) Speed error (c) Heading error (d) Percent nodes recovered

Figure 12. Time-averaged trajectory error and node availability as node speed increases.

within any observation window.

We also evaluated the growth and variation in recovered trajectory error as node speed increases (Fig. 12). As expected, faster node motions generally degrade the algorithm's position and speed estimates and availability since fewer ranging observations can be gathered while any given node is within communications range. However, increasing node speed does have one beneficial effect: a longer effective triangulation baseline between successive range measurements that yields more accurate heading estimates.

Finally, we evaluated growth and variation in recovered trajectory error as the node degree increases (Fig. 13), from five (sparse) to twenty-five (dense). The availability transitions rapidly from low to high at approximately degree ten. Because our algorithm employs a thresholding heuristic when generating triangles, low-degree networks tend to produce insufficiently many shared triangles for propagation of localization information. In other words, the algorithm prefers to achieve high-quality localization, even if only part of the network can be localized, than to localize the entire network with lower accuracy. This behavior is similar to that observed with earlier "robust quadrilateral" criteria [14], with an availability transition similar to that observed in other settings [24].



(a) Position error (b) Speed error (c) Heading error (d) Percent nodes recovered

Figure 13. Time-averaged trajectory error and node availability as node degree increases.

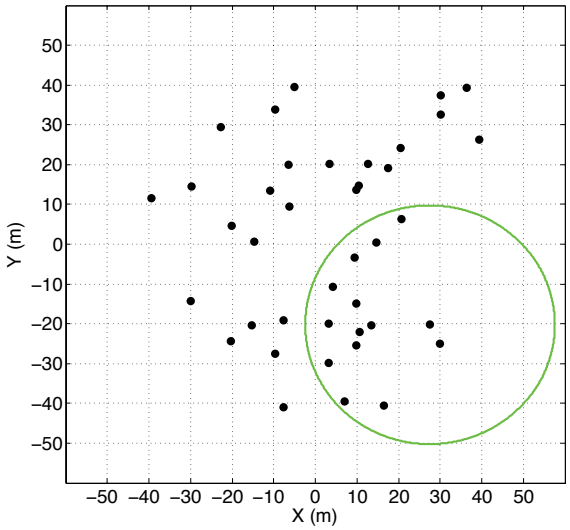


Figure 14. True node positions at $t = 75$ seconds (dots), and ranging radius (circle).

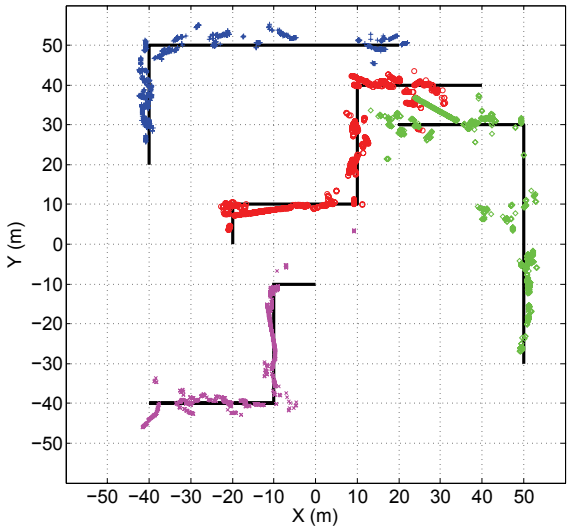


Figure 15. True (black) and estimated (colored) trajectories for four of 40 nodes.

4.5. Large-Network Example

We simulated a network of 40 nodes moving for 100 seconds within a square one hundred meters on a side (Fig. 14). Each node repeatedly generates path segments by randomly selecting an axis-aligned velocity and distance (Fig. 15). We adjusted the velocity distributions to force motion transitions to occur in closely-spaced bursts separated by about 30 seconds (Fig. 16), 10 seconds (Fig. 17), and 3 seconds (Fig. 16) of transition-free motion. Table 2 summarizes the algorithm's performance in each of the three regimes.

4.6. Faster Ranging

We also simulated operation of a hypothetical future UWB device capable of ranging an order of magnitude faster than today's devices (i.e., at 50 Hz vs. 5 Hz). In the most difficult regime where nodes change direction roughly every 3 seconds, we found that with a 1-second (vs. 3-second) observation window, the algorithm localized nodes to within 1.05 m on average (about 25% better).

		30 sec.	10 sec.	3 sec.
Position error (m)	Median	1.30	1.21	1.33
	Std. dev.	0.950	2.66	1.56
	Maximum	9.12	17.197	21.7
Speed error (m/s)	Median	0.136	0.137	0.174
	Std. dev.	0.124	0.139	0.179
	Maximum	0.771	1.05	1.36
Heading error (deg)	Median	10.0	8.94	13.9
	Std. dev.	17.2	12.3	10.6
	Maximum	103	76.0	84.6

Table 2. Performance of large-network MBL.

4.7. Frequent Updates

Finally, we considered the effect of our adaptive update rule that triggers localization only when observed ranges

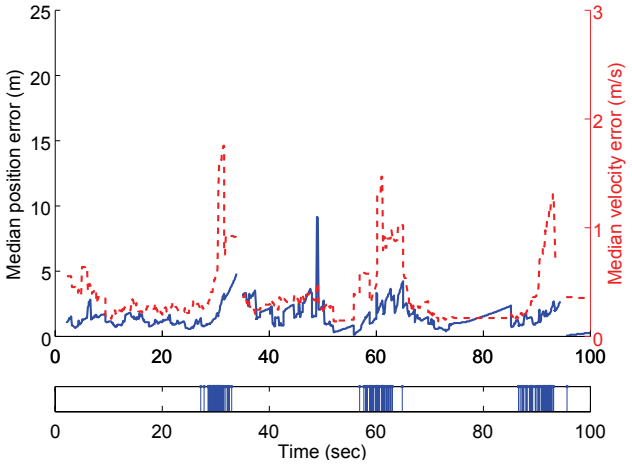


Figure 16. 30-sec. smooth motion intervals.

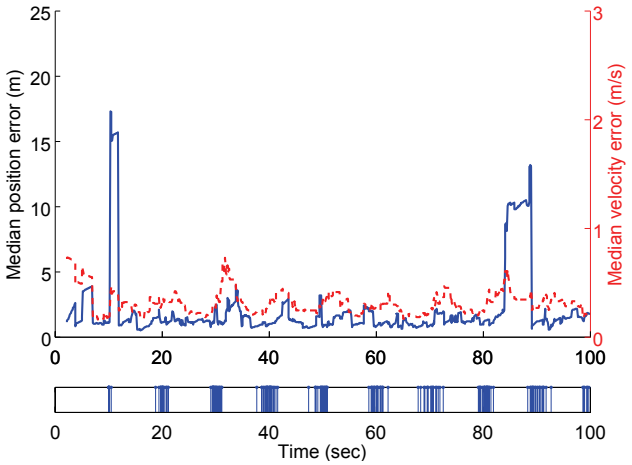


Figure 17. 10-sec. smooth motion intervals.

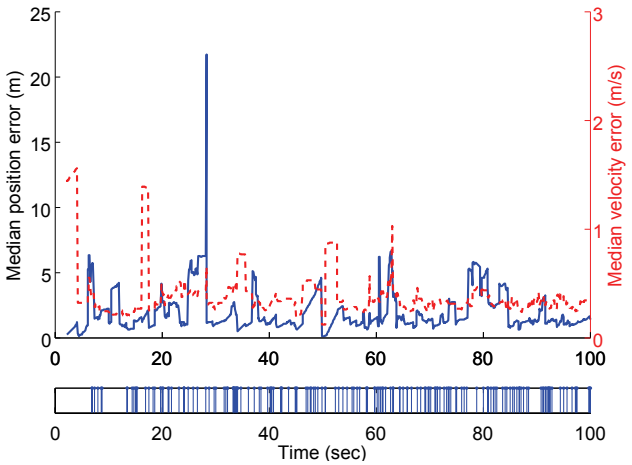


Figure 18. 3-sec. smooth motion intervals.

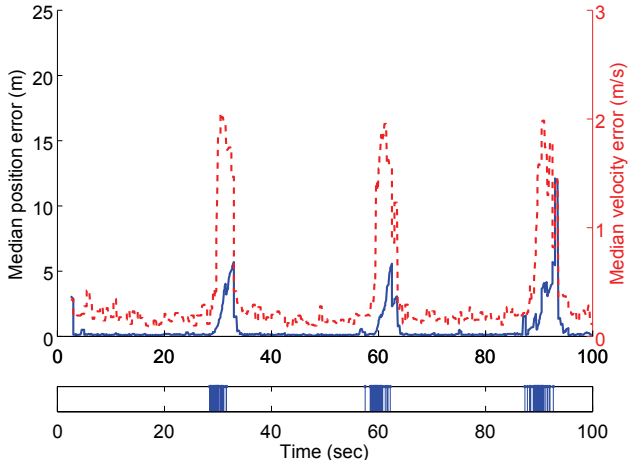


Figure 19. 30-sec. intervals, 2 Hz updates.

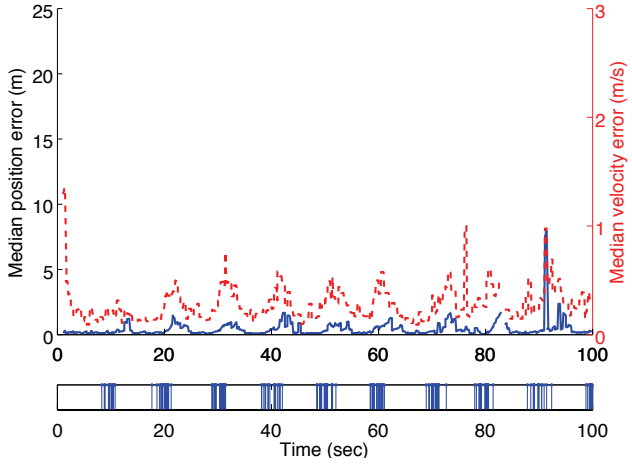


Figure 20. 10-sec. intervals, 2 Hz updates.

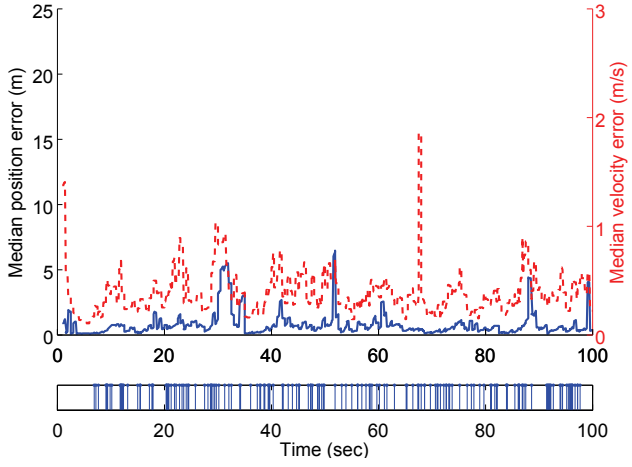


Figure 21. 3-sec. intervals, 2 Hz updates.

deviate significantly (1 m) from prediction (§3.6). While this rule saves computation time, it sacrifices about 1 m of positioning accuracy. Thus we ran the large-network experiment with periodic updates forced at 2 Hz. Table 3 summarizes the resulting algorithm's performance in each of the three regimes. Figs. 19, 20, and 21 show the corresponding plots of error over time. The computation cost grew roughly six-fold (58,221 vs. 10,071 triangles considered), but the er-

rors are significantly smaller and the peaks caused by motion changes are clearer. For 90% of the estimates, positions are within 1.4 m, speeds are within 0.3 m/s, and headings are within 25° of ground truth.

		30 sec.	10 sec.	3 sec.
Position error (m)	Median	0.158	0.304	0.518
	Std. dev.	1.32	0.643	0.944
	Maximum	12.1	7.92	6.46
Speed error (m/s)	Median	0.111	0.120	0.173
	Std. dev.	0.107	0.085	0.129
	Maximum	0.645	0.732	1.57
Heading error (deg)	Median	5.69	8.59	13.5
	Std. dev.	37.0	8.29	11.3
	Maximum	161	81.6	64.9

Table 3. Large-network MBL, 2 Hz updates.

5. Discussion

Our proposed method recovers motion trajectories well over a range of (simulated) operating conditions, but fails when ranging is too slow, when nodes move too quickly, when relative motions are too small, or when the network is sparse (i.e. when too few nodes lie within ranging radius).

Even when the ranging rate and network density are adequate, two real-world factors prevent our algorithm from achieving perfect instantaneous estimates of all node trajectories. The first is measurement noise. Even small ranging errors of a few centimeters degrade recovered motion and alignment parameters, producing trajectory estimates that lose accuracy over time. The second factor is latency of communication and computation; it takes time for any change in a node's motion to be sensed by other nodes and incorporated into their computations, and for the results to propagate throughout the network. We envision adopting a "best effort" methodology (as in [14]) in which each node frequently broadcasts its latest information to its neighbors, by piggybacking alignment information onto ranging pulses (which would be exchanged frequently in any case). In this way, updated motion solutions will propagate rapidly through the network, and every node will have not perfect, but at least reasonably timely, estimates of the motion of all other nodes within its connected component.

At the heart of our solution is a hyperbola fitting method for estimating the relative motion between two nodes. The fitting method that we use removes outliers, but is vulnerable to noise in the remaining data, which we believe reduces the quality of predictions based on the fit. Further smoothing may reduce the system's noise sensitivity, yielding better predictions and ultimately improved end-to-end localization. Another weakness of the fitting method becomes evident when the relative motion of two nodes is small, making the computed time of closest approach ambiguous and sensitive to noise. It may improve matters to detect and handle this scenario explicitly.

When devices move along complex motion paths, we can introduce higher-order parametric terms, e.g., time-dependent acceleration terms, and estimate them using additional range measurements. Also, a combinatorial algorithm could determine where best to split motion paths into lower-dimensional segments.

We also envision integration of inertial sensing to handle transient loss of range measurements, due to channel contention, intervening material and attenuation, or excessive distance to neighbors. Transient errors in trajectory estimation, when node velocities change over short time scales, can also be smoothed using filtering [22]. Inertial data could also be used to stabilize, for each user, the coordinate frame in which that user's MBL solution is displayed.

The network itself can provide predictive feedback to help ensure some minimum quality of service. For example, leaders could receive guidance to slow down, laggards to speed up, so as to keep the network sufficiently dense for operation in the regime required by MBL.

We believe that the fundamental parameters determining the method's performance in any real-world setting include ranging rate, ranging radius, ranging noise, maximum node speed and acceleration, expected network density, inter-node communications latency, and the computational resources available at each node. We hope to discover the quantitative relationship among these parameters and use it predictively, for example to determine what user motions are recoverable using a given UWB device with a specified behavior, or conversely to select among some set of available UWB devices given some characterization of the group's motion.

6. Conclusion

This paper described a method for localizing a network of moving, range-capable nodes. The method is the first to our knowledge to estimate persistent node trajectories, rather than instantaneous node positions. This choice enables the method to make good use of time-windowed range data, although at a cost of increased latency in system response to changes in the motions of individual nodes.

The proposed method combines three computations to achieve localization: hyperbola fitting; a form of trilateration; and subgraph alignment. We implemented each component within a simulation model informed by the ranging characteristics of a commercially available UWB radio. When nearby node pairs can achieve sustained ranging frequencies of 5 Hz over distances up to about 30 m with standard deviation of a few centimeters, our method localizes nodes moving at typical walking speeds to within 0.2–1 m of their correct position (depending on the frequency of mo-

tion changes and recomputation), within 0.2 m/s of their correct speed, and within 15° of their correct heading. The method fails when ranging is too slow, the network is too sparse, or when node motions are too fast or too correlated.

Our preliminary results, based largely on simulation data, are promising but not determinative. We hope to evaluate our method soon using actual UWB ranging devices and real-world group motions.

Acknowledgements

We thank the technical staff of Time Domain Corp. for useful discussions of their UWB devices. We also thank Moe Win, Henk Wymeersch, Wesley Gifford, and Jaime Lien for their help with UWB device characterization.

Jun-geun Park is supported by a fellowship from the Kwanjeong Educational Foundation, and by the National Science Foundation through award NSF ITR ANI-0205445.

References

[1] B. Alavi and K. Pahlavan. Modeling of the ToA-based distance measurement error using UWB indoor radio measurements. *IEEE Communications Letters*, 10(4):275–277, 2006.

[2] S. Chatterjee and I. Olkin. Nonparametric estimation for quadratic regression. *Statistics and Probability Letters*, 76(11):1156–1163, 2006.

[3] S. Datta, C. Klinowski, M. Rudafshani, and S. Khaleque. Distributed localization in static and mobile sensor networks. In *IEEE Int'l Conf. on Wireless and Mobile Computing*, pages 69–76, 2006.

[4] B. Dil, S. Dulman, and P. Havinga. Range-based localization in mobile sensor networks. In *Third European Workshop on Wireless Sensor Networks*, volume 3868, pages 164–179. Springer, 2006.

[5] T. Eren, D. Goldenberg, W. Whiteley, Y. R. Yang, A. S. Morse, B. D. O. Anderson, and P. N. Belhumeur. Rigidity, computation, and randomization in network localization. In *Proc. IEEE INFOCOM*, pages 2673–2684, March 2004.

[6] L. Freitag, M. Johnson, M. Grund, S. Singh, and J. Preisig. Integrated acoustic communication and navigation for multiple UUVs. In *Proc. MTS/IEEE Oceans*, pages 290–294, Honolulu, HI, USA, Sept. 2001.

[7] D. K. Goldenberg, A. Krishnamurthy, W. C. Maness, Y. R. Yang, A. Young, A. S. Morse, A. Savvides, and B. D. Anderson. Network localization in partially localizable networks. In *Proc. IEEE INFOCOM*, pages 313–326, Miami, FL, March 2005.

[8] B. Hoffmann-Wellenhof, H. Lichtenegger, and J. Collins. *Global Positioning System: Theory and Practice, Fourth Edition*. Springer-Verlag, 1997.

[9] B. K. P. Horn, H. M. Hilden, and S. Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *Journal of the Optical Society of America A*, 5(7):1127–1135, 1988.

[10] L. Hu and D. Evans. Localization for mobile sensor networks. In *Proc. MobiCom*, pages 45–57. ACM Press New York, NY, USA, 2004.

[11] R. Kurazume, S. Nagata, and S. Hirose. Cooperative positioning with multiple robots. In *Proc. IEEE Int'l Conf. in Robotics and Automation*, pages 1250–1257, Los Alamitos, CA, USA, May 1994.

[12] J.-Y. Lee and R. Scholtz. Ranging in a dense multipath environment using an uwb radio link. *IEEE J. Selected Areas in Communications*, 20(9):1677–1683, Dec. 2002.

[13] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The flooding time synchronization protocol. In *Proc. SenSys*, pages 39–49, New York, NY, USA, 2004. ACM Press.

[14] D. Moore, J. Leonard, D. Rus, and S. Teller. Robust distributed network localization with noisy range measurements. In *Proc. SenSys*, pages 50–61, New York, NY, USA, 2004. ACM Press.

[15] N. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller. Mobile-Assisted Localization in Wireless Sensor Networks. In *Proc. InfoCom*, pages 172–183, March 2005.

[16] N. Priyantha, A. Chakraborty, and H. Balakrishnan. The Cricket Location-Support System. In *Proc. MobiCom*, Boston, MA, Aug. 2000.

[17] N. B. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller. Anchor-free distributed localization in sensor networks (poster abstract). In *Proc. SenSys*, pages 340–341, Los Angeles, California, USA, November 5–7 2003.

[18] S. Roumeliotis and G. Bekey. Synergetic localization for groups of mobile robots. In *Proc. IEEE Decision and Control*, pages 3477–3482, Sydney, Australia, Dec. 2000.

[19] A. Savvides, W. Garber, S. Adlakha, R. Moses, and M. B. Srivastava. On the error characteristics of multihop node localization in ad-hoc sensor networks. In *Proc. IPSN*, pages 317–332, Palo Alto, CA, April 2003.

[20] E. Stump, B. Grocholsky, and V. Kumar. Extensive representations and algorithms for nonlinear filtering and estimation. In *Proc. WAFR*, New York, NY, July 2006.

[21] J. Vaganay, J. Leonard, J. Curcio, and S. Willcox. Experimental validation of the moving long base-line navigation concept. In *AUV 2004*, pages 555–556, Nagoya, Japan, June 2004.

[22] G. Welch, G. Bishop, L. Vicci, S. Brumback, K. Keller, and D. Colucci. The HiBall tracker: High-performance wide-area tracking for virtual and augmented environments. In *Proc. ACM Symp. on Virtual Reality Software and Technology*, December 1999.

[23] L. Whitcomb, D. Yoerger, H. Singh, and J. Howland. Advances in Underwater Robot Vehicles for Deep Ocean Exploration: Navigation, Control and Survey Operations. In *The Ninth Int'l Symposium on Robotics Research*, Springer-Verlag, London, 2000.

[24] K. Whitehouse and D. Culler. A robustness analysis of multi-hop ranging-based localization approximations. In *Proc. IPSN*, pages 317–325, Nashville, TN, April 2006.

[25] M. Z. Win and R. A. Scholtz. On the robustness of ultra-wide bandwidth signals in dense multipath environments. *IEEE Communications Letters*, 2(2):51–53, 1998.

[26] Y. Xu, Y. Ouyang, Z. Le, J. Ford, and F. Makedon. Mobile anchor-free localization for wireless sensor networks. In *Distributed Computing in Sensor Systems*, 2007.