Moving-Baseline Localization for Mobile Wireless Sensor Networks

by

Jun-geun Park

B.S., Seoul National University (2004)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2009

Author .....
Department of Aeronautics and Astronautics
January 29, 2009

Certified by .....
Seth Teller
Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by .....
David L. Darmofal
Associate Department Head
Chair, Committee on Graduate Students

Moving-Baseline Localization for

Mobile Wireless Sensor Networks

by

Jun-geun Park

Submitted to the Department of Aeronautics and Astronautics
on January 29, 2009, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

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.

Thesis Supervisor: Seth Teller
Title: Professor of Computer Science and Engineering

Acknowledgments

First and foremost, I would like to thank my thesis advisor, Professor Seth Teller, for his guidance and support. He always encouraged questions and discussions, from which I learned how to explore this exciting research field. My greatest debt is to him.

I am also grateful to Professor Erik Demaine for helpful feedbacks and discussions. His ideas brought new perspectives to my work and made it richer.

I would like to thank Professor Moe Win, Henk Wymeersch, Wesley Gifford, and Jamie Lien for their collaboration with UWB device characterization. Our measurement campaign would never been successfully completed without their help. I also thank the technical staff of Time Domain Corporation for useful discussions of their UWB devices.

My parents have been unfailingly supportive throughout my life. I cannot express how much I appreciate their unconditional support for everything I do.

I am truly indebted to Mina for all her love and support. Her smile has always been the best medicine even in my hardest times.

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

This thesis is an extended version of the following publication:

Jun-geun Park, Erik D. Demaine, and Seth Teller. Moving-Baseline Localization. In *IPSN '08: Proceedings of the 7th International Conference on Information Processing in Sensor Networks*, pages 15–26, 2008.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Location determination is a fundamental problem, attracting human attention since antiquity. Today, GPS (Global Positioning System [27]) 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 thesis.

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 [63]. 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 thesis 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.

1.1.1 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 information with, any node to which it can range. This choice of setting is motivated by existing devices with these capabilities, such as Crickets [50] and UWB (ultra-wideband) radios [37].

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 (Figure 1-1). Our method estimates a trajectory for each node that is consistent with recent range measurements involving that node.

In this thesis, 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

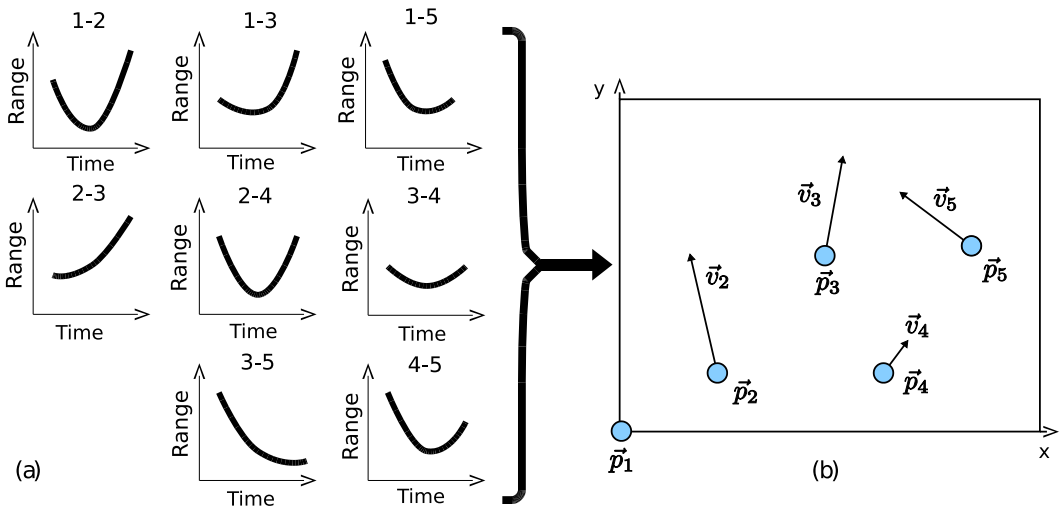


Figure 1-1: MBL as a local-to-global estimation problem. Available data (a) consists of a time series of range measurements at each node. Problem solution (b) consists of an estimate of all nodes' motions in each node's rest frame (solution for node 1 shown).

experiments with real UWB devices. We show that MBL can be solved in this setting through robust hyperbola fitting, trilateration, subgraph alignment, and change point detection. We implement and analyze the algorithm in simulation, and discuss its extension to less restricted settings.

1.1.2 Contributions and Organization of the Thesis

The main contributions of our work is as follows:

- We develop a distributed algorithm for mobile wireless sensor networks which estimates node trajectories rather than static position estimates.
- We quantify the benefits of trajectory estimation over static position estimation by comparing the MBL method with a static localization algorithm.
- We characterize a current-generation UWB radio to develop a faithful range error model.
- We evaluate the effects of various factors in mobile networks on the MBL algorithm through extensive simulation.

- We study existing localization algorithms and identify their limitation for mobile anchor-free networks

The remainder of this thesis is structured as follows. Chapter 2 reviews the current localization algorithms for wireless sensor networks and identifies the thrust of this work. Chapter 3 presents the MBL algorithm. In Chapter 4, we develop a range error model for UWB radios. Chapter 5 describes our experimental framework, evaluation metrics, and experimental results. We also discuss the method's behavior and future work. Finally, Chapter 6 concludes.

Chapter 2

Localization Algorithms for Wireless Sensor Networks

As research in wireless sensor networks proliferates both in scope and depth in recent years, the localization problem, an essential capability enabling other applications, has received a considerable amount of attention over the past decade. In this chapter, we study existing localization algorithms for wireless sensor networks and mobile network, and explain in what aspect our work is different from previous work. In Section 2.1, important properties of localization algorithms are identified. In Section 2.2, we review previous work on the localization problem. In Section 2.3, we describe the thrust of this thesis.

2.1 Properties of Localization Algorithms

Localization algorithms for wireless sensor networks can be classified according to a number of properties that each algorithm has. These properties often define the overall structure of a localization algorithm, and also impose limitations on it as well. In this section, we identify several properties related to the localization algorithms and explain their implication on the sensor network localization problem.

Anchor-Based versus Anchor-Free Many localization methods require *beacons* or *anchor* nodes that know their position on some absolute coordinate system

with very high or absolute certainty[1]. These methods are called *anchor-based* methods. In anchor-based methods, the localization problem can be formulated as finding a consistent set of locations of nodes in the network, given the information obtained from anchor-node relations as well as node-node relations. The anchors can be a form of dedicated devices emitting signals periodically [50], or a different type of sensor nodes that can acquire precise location of themselves from the outside of the network, for example, GPS. Since anchor-based methods require pre-deployed infrastructure, they may be impractical in certain application domains where anchors cannot be placed.

On the other hand, *anchor-free* methods refer to the localization methods that do not require specialized anchor nodes. Since no information from outside of the network is used, an anchor-free method itself does not have a mean to localize the network on the absolute reference frame. Instead, it recovers relative locations of nodes on a relative coordinate system centered on an arbitrary origin. If a sufficient number of anchor positions are given (at least three anchors in two-dimensional space), the output of an anchor-free method can be easily transformed to node locations in an absolute reference frame.

Centralized versus Distributed For *centralized algorithms*, computation occurs at one specific node or at a computer outside the network, whereas for *distributed algorithms*, computation load is distributed among nodes in the network. This classification is directly related to how the localization problem is formulated. If a problem is formulated as a global optimization such as convex optimization in [17], the computation must be performed in a centralized way. Because a computing node must have all the necessary information in centralized algorithms, the information required for computation, such as internode distances, must be relayed to the computing node. Therefore, centralized algorithms may suffer from many collisions and contention during wireless communication around the computing node [67]. This difficulty in communication

[1]Throughout the remaining chapter, we will simply call a beacon or an anchor node as an anchor, and a non-anchor node as a node.

may prevent the centralized algorithms from being scalable over the size of the network.

For this reason, distributed localization algorithms are more popular in wireless sensor networks. In distributed algorithms, each node splits up a computation job in some way; for example, each node is responsible only for finding its own coordinates [54], or every node computes node locations only in its vicinity and shares its solution with other nodes [42].

Proximity, Distance, Angle-of-Arrival. Localization algorithms use various types of information or measurements to infer locations of nodes in the network. While other types of information can be used (e.g., correlation between sensor data [46]), three major types of measurements will be described here – proximity, distance, and angle-of-arrival. *Proximity* is one of the simplest forms of information that a sensor node can obtain about its neighborhood. While proximity information only provides coarse location estimate, some localization methods such as [25] can estimate node positions with high granularity using multiple proximity measurements and *a priori* information about the proximity measurements, such as maximum detection range.

Another type of information used for localization is *distance*. Distance between sensor node is obtained in various forms, such as, received signal strength (RSS), time-of-arrival (ToA), or time-difference-of-arrival (TDoA) [32]. Estimating distance from RSS is based on the relation between RSS and distance whose relationship can be modeled as a path-loss equation:

$$P_r = c\frac{P_t}{d^\alpha}$$

where the path-loss exponent α is typically assumed to lie in the range from 2 to 5 [67]. Distance measurement through ToA uses the relationship between distance and signal propagation time when the signal's propagation speed through the transmission medium is known. If the transmitter and the receiver are

synchronized, inter-node distance can be calculated directly from a timestamp included in the ranging packet. TDoA uses two or more different sets of transmission pairs to eliminate the need of time synchronization in ToA. For example, the Cricket location support system [50] uses a combination of RF and ultrasound signal.

The last type of measurement is *angle-of-arrival* (AoA). AoA information can be acquired from directional antennae or an antennae array. Because of the need for multiple antennae, AoA information is less suitable for sensor networks in which the size and complexity of each node is restricted. However, it is possible to estimate sensor locations from only AoA information. The most basic technique to do it, triangulation, and its extensions have been studied in the context of wireless sensor network localization (e.g. [2]).

Static Network versus Mobile Network. Most of the existing localization algorithms for sensor networks do not consider node mobility explicitly, assuming that the network is *static*. While many sensor platforms are not likely to move actively like robots, there are certainly situations that sensors exhibit mobility, thus rendering the network *mobile*. For example, human-equipped sensors that move passively with people can be such an example. Mobile networks show different characteristics from static networks, such as changing topology, varying connectivity, and latency problem. Therefore, localization algorithms for mobile networks must be designed considering these different (and challenging) characteristics.

2.2 Previous Work

In this section, we review some of existing work on the localization problem in wireless sensor networks and mobile computing. Localization methods proposed by researchers are classified according to their main characteristics.

2.2.1 Proximity-based Approaches

Many simple yet effective localization algorithms are based on the proximity information. Proximity-based approaches often assume existence of large number of anchors in the region so that each node can approximate its location from single-hop proximity information. Bulusu *et al.* [8] assume an idealized radio model in which radio propagation is isotropic and identical for all radios. Based on this assumption, each radio can localize itself on the centroid of "reference points" (anchors) to which it is connected. Those reference points are densely and regularly placed throughout the region. They reported that the idealized radio model was valid in outdoor environments, but inappropriate for indoor environments.

The bounding box algorithm [57] and its variants, e.g. [14, 25], exploit knowledge on the communication range further and approximate location of a node as an intersection of feasible regions. The bounding box algorithm [57] uses rectangles centered at anchor positions. Another algorithm maintains the intersections of convex polygons to estimate node locations [14]. This algorithm can be adapted to mobile networks by "dilating" the convex polygons outward by the known maximum node speed.

The APIT (Approximate Point-In-Triangulation) method [25] is also based on proximity to anchors. It assumes a weak assumption on radio signal strength: signal strength decreases monotonically as range increases. A node first chooses three audible anchors, and determines if it is inside the triangle defined by the three anchors. It does so by observing if there is a neighbor whose signal strengths to anchors are all stronger or all weaker than its own signal strength. If then, the node is considered to be located outside of the triangle, because at least one of its neighbor nodes is consistently farther from or closer to the three anchors. Given a sufficient number of anchors, each node can estimate its position by computing the intersection of these triangles. Performance of APIT method is comparable to other range-free algorithms such as the DV-hop [44] or centroid method [8], but it requires: 1) that anchors are either placed densely or equipped with a high-powered transmitter; and 2) that nodes

are regularly placed. Otherwise, it can make an incorrect decision as to whether a node is inside a triangle or not.

Proximity-based approaches are simple and cost-effective because they do not require accurate ranging capability, and thus can operate on simple low-cost devices. However, the localization error is relatively large so they are most suitable to coarse-grained localization. Also, they often require high anchor density. For example, all the methods described in this section except [14] require multiple anchors to exist within communication range of each node, and [14] requires anchor density of at least 0.4 to obtain localization accuracy better than the communication range.

2.2.2 Lateration-based Approaches

Lateration refers to a class of geometric methods to locate an object given distance measurements from multiple reference positions [26]. In two-dimensions, a node position can be uniquely determined from three noncollinear node positions and distances. This technique is called *trilateration*. If more known nodes are available [54], determining a node position is equivalent to solving a set of quadratic equations, which is called *multilateration*[2]. If every node can hear from three or more anchors all the time, position determination becomes an easy problem of simple application of trilateration. However, if only a fraction of nodes are anchor nodes, or there are no anchors at all, more sophisticated localization methods are required.

One solution for the limited anchor availability is to perform lateration with approximate distances if direct distances to anchors are unknown. Particularly, the Hop-TERRAIN method [52] and the DV-hop method [44] share a similar idea to approximate distances to anchor nodes. First, anchor nodes initiate broadcast with their positions. Then, nodes can update hop counts to anchor nodes with these broadcast packets and estimate distances to anchors using the hop counts and the average distance per hop. The average distance per hop can be calibrated when an

[2]In a different usage, multilateration refers to a position estimation technique based on TDoA of signals from different reference locations. In this thesis, we refer to multilateration as defined above, following [54].

anchor receives a broadcast packet from another anchor with its position and hop count. A major drawback of these methods is, as indicated in [44], that they work well only with isotropic networks, because distance estimation by hop count is a poor approximation if the network topology is anisotropic.

Another approach is to build a network localization graph incrementally. The Ad-Hoc Localization System (AHLoS) [54] uses iterative lateration. Since lateration requires at least three anchors to locate a node, in AHLoS, nodes with sufficient connectivity to anchors (for unique determination of their own positions) first localize themselves and become anchor nodes to help localization of remaining nodes. While this iterative lateration makes it possible to localize sensor nodes given only a fraction of anchors as opposed to the dense anchor placement assumed in [8], it comes with cost — error accumulates as non-anchor nodes become anchor nodes. To combat this error propagation, recent iterative lateration algorithms employ error management schemes that selectively use reliable measurements [39, 66].

If there exists no anchor nodes in the network, incremental approaches can be used to recover relative coordinates of the nodes. Trilateration is often used as a subroutine in this case. For example, Capkun *et al.* [9] proposes a relative localization method, in which every node builds a local coordinate system and those local coordinate systems are aligned with each other. First, each node becomes the center of a local coordinate system, then determines positions of its one-hop neighbors by trilateration. After nodes build local coordinate systems, correction angles between them are computed, considering possible reflection. Then, the position of one node can be represented in any coordinate system defined by another node, by iteratively applying corresponding coordinate transformations. This idea of generating global solution from small local clusters, sometimes called "patch-and-stitch" [64], is common in many distributed localization algorithms [9, 42, 55]. Patch-and-stitch methods make it possible to localize multi-hop networks in an anchor-free setting, but introduce error compounding during "stitch" processes [64].

When making local clusters using trilateration, an algorithm may find an incorrect realization with a *flip configuration* if ranging is noisy. For instance, when node D is

trilaterated from known node positions A, B, C, a small measurement error in distance CD can induce an incorrect flip in the position of D along edge AB, producing a large error in the position computed for D. To avoid this flip configuration, one distributed localization method [42] uses "robust quadrilaterals", well-shaped 4-cliques in the network graph, as its elementary building block.

The algorithms described above share a common viewpoint to the network localization problem. That is, the problem is formulated as finding a weighted graph embedding whose edge weights match the distance measurements. A theoretical foundation for this problem is elucidated in terms of graph rigidity theory [18, 3]. Particularly, in [3], it is proven that unit disk graph reconstruction is NP-hard. This fact makes trilateration graphs, which are realizable in polynomial time, useful for localization. However, the trilateration-based approaches typically suffer from low node recovery at low node density [64, 24]. To alleviate this problem, Goldenberg *et al.* propose use of *bilateration* in which a set of possible positions are determined from two known positions rather than three or more. The resulting sweeps algorithm eliminates conflicting combinations of node positions as it proceeds. A different approach is to use a successive positions of a moving node virtual reference points [22, 49]. The moving node can provide sufficient measurements for localization when the information among static nodes is insufficient for unique determination of node positions.

2.2.3 Optimization or Learning-based Approaches

A different class of approach to the network localization problem is to formulate it as a global optimization problem. Doherty *et al.* [17] formulate the position estimation problem as a convex optimization problem given the known anchor coordinates, specifically second-order cone programming. Proximity, distance, or angular constraints are formed as convex constraints so that their intersection is still convex. Similarly, Biswas *et al.* [6] present a semi-definite programming based method, and Moses *et al.* [43] derive the maximum likelihood estimator for node positions under a Gaussian error model. Although this global optimization formulation is mathematically elegant, it has two major problems — it is unavoidably centralized and may

become computationally intractable as the size of the networks grows.

On the other hand, seeing the localization problem as a point estimation problem on node positions offers an insightful perspective on the localization error behavior. To this end, researchers have used the *Cramér-Rao bound*, a statistical lower bound on the covariance of an unbiased estimator, to characterize the error behavior [43, 53, 36]. In particular, Savvides *et al.* [53] analyze how network setup factors such as anchor placement, node density, or beacon density affect localization accuracy.

Optimization-based methods can also be used for anchor-free scenarios. Anchor-free localization (AFL) [48] seeks, by mass-spring relaxation, for an graph embedding whose edge lengths are consistent with inter-node distances. To prevent the optimization from falling into local minima, AFL algorithm employs a heuristic to make an initial "fold-free" assignment of node coordinates.

The localization problem has also been formulated as an instance of learning methods. Shang *et al.* [56] applied multidimensional scaling (MDS) to the localization problem. MDS is a set of techniques used for dimensionality reduction or visualization of high-dimensional data, finding a low-dimensional embedding in which "distances" between data points are preserved [58]. In general applications of MDS, the "distance" or dissimilarity between two data points is to be defined. However, the MDS-MAP method [56] utilizes the idea that, if each data point represents a node, range measurements between nodes directly give the very dissimilarity metric. Distances between non-neighbor nodes are approximated by shortest-path distances. Then, the resulting embedding found by MDS becomes an approximation of node positions on two- or three-dimensional space. The MDS-MAP has several advantages. It does not require anchor nodes and constructs a relative map. Also, it is a closed-form method (which does not require iterative computation). However, because it requires all-to-all distance information between nodes, it is centralized and does not work well with anisotropic networks in which the shortest-path distance may not be a close approximation to the real Euclidean distance. To overcome this problem, researchers attempt to develop distributed versions of MDS: by applying "patch-and-stitch" scheme [55], or by minimizing multiple local cost functions [13].

In addition, self-organizing maps [23], a neural network technique, and a variety of manifold learning algorithms (e.g. [46]) have been applied to the localization problem.

2.2.4 Bayesian Probabilistic Approaches

Most of the localization methods described so far solve for a single position estimate for each node. Accordingly, they cannot provide a measure of uncertainty in the solution arising from uncertainty in measurements or the method itself. Although bounding-box-like methods may provide a crude uncertainty metric, the area or volume of a bounding box, it does not tell which exact point in the box is "most likely" based on the current measurements.

In contrast, Bayesian inference offers a variety of estimation methods to infer posterior probability (also called belief) after evidences or measurements are taken into account. The posterior distribution can provide a single position estimate if needed, as well as characterize uncertainty in the estimate.

Ihler *et al.* [30] model sensor networks as graphical models (specifically Markov random fields) and applied nonparametric belief propagation which is a sample-based variant of well-known belief propagation [47]. In this work, each node is modeled as a variable in a graphical model. Since the belief propagation estimates a probability of interest by passing messages between the variable, this method is naturally suited to the distributed nature of sensor networks.

For mobile networks, sequential estimation methods have been applied. Sequential techniques for localization, such as extended Kalman filters or particle filters, have been widely used for robotics [20, 59]. However, sequential estimation for sensor networks must be treated differently from that for robots. Specifically, typical sensor platforms are expected to have limited or no knowledge over mobility, have limited sensing capability devoted to localization purpose, and have constrained computation capability [29]. Probabilistic localization techniques were adapted for sensor networks with considering these limitations. Particularly, particle filters (also known as Sequential Monte-Carlo localization when applied to localization problems) are popular techniques for sensor networks, because their computational load is adjustable and

implementation is easy while they can represent an arbitrary distribution. In particular, Hu and Evans [29] adapt the particle filters for mobile networks in range-free setting. Baggio and Langendoen [4] improve the method by Hu and Evans in the way the filter uses proximity information. Dil *et al.* [15] use both proximity and range information. Klingbeil and Wark [34] present a particle-filter-based indoor localization system that uses various sensing modalities including proximity, inertial sensors, and magnetic compasses.

One major property of sequential localization techniques is that they require *a priori* information about the environment (i.e. map). If such information is provided, i.e., beacons (any reference points such as anchor nodes) are placed widespread on the operation region, the sequential localization techniques are well-suited for a mobile sensor network localization task. However, if there are no anchors available in the network, such techniques cannot incorporate sensor measurements into the filter. In this sense, they are not suitable for relative localization.

2.2.5 Approaches in Other Areas

A variety of algorithms to provide location information have been developed in other areas than wireless sensor network research. In particular, localization problem has been extensively studied in robotics. Popular localization algorithms in robotics include extended Kalman filters [38], hidden-Markov model [20], and particle filters [19]. However, these estimation algorithms have several limitations: 1) they estimate self-location only, hence it requires another layer of protocol in order for one robot to have knowledge about others' locations; 2) they require knowledge about the map, or it must be obtained by exploration, which poses a problem as a difficult simultaneous localization and mapping (SLAM) problem; 3) they usually use a combination of multiple sensing modalities such as LIDAR, vision, and acoustic sensor. As stated in the previous section, unique characteristics of wireless sensor networks make it impractical to apply robotics techniques directly to general sensor networks localization problems.

Localization research in collaborative autonomous attempts to solve some of these

limitations. One 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 [35]. 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 [51]; no known long-range sensor can provide such measurements in the presence of complex occlusion.

Several localization 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 [21]. 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 [60]. All vehicles are thus subject to position errors that grow without bound [63].

2.3 Thrust of MBL

In the previous section, we reviewed various localization protocols for wireless sensor networks. The major body of the current localization research concentrates on the static network problem. Localization protocols for mobile networks is still a largely unexplored area of research. Although several methods are designed to work for mobile networks, they typically assume the existence of anchor nodes. This assumption is often unrealistic in unexplored or hostile environments.

The thrust of the present thesis 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. In principle, one can apply a localization algorithm designed for static networks to a mobile network, by repeating computation at each time instance. However, such an algorithm designed without explicit consideration of node mobility will not work as expected, suffering from latency and computational cost. In contrary, our goal is to estimate node positions and velocities together, reducing such difficulties.

Chapter 3

Moving-Baseline Localization

This chapter presents the Moving-Baseline Localization (MBL) method, a distributed algorithm to recover relative trajectory of each node rather than a single position at a fixed time. Section 3.1 describes the basic idea and gives an overview of the method. Sections 3.2–3.5 describe how each node constructs its own global view of the network from local range measurements. Section 3.6 explains the maintenance step of the MBL method in which each node detects motion change of other nodes and refines its view of the network. Section 3.7 studies the computational complexity of the MBL algorithm.

3.1 Overview

We assume that each node: has a unique integer identifier; can discover, range to, and communicate with nearby nodes, but has no prior information about its position; and maintains the time t, either locally or through a network synchronization method (e.g. [40]). 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 (Figure 3-1).

We start with the simplest instance of MBL: planar motion, with each node mov-

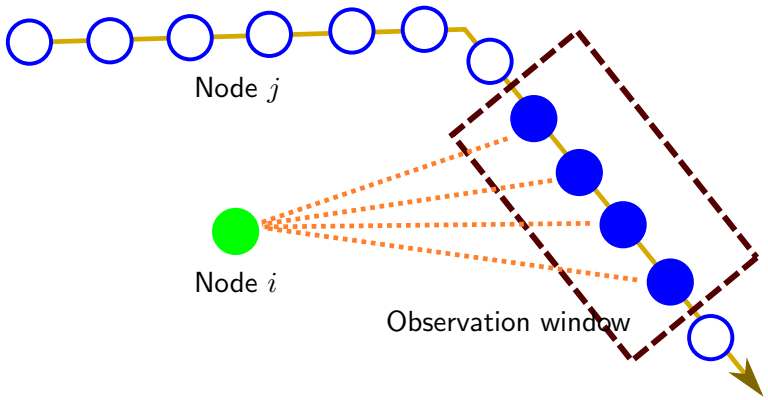


Figure 3-1: Time-series range data $r_{ij}(t)$.

ing along a straight-line path at constant speed. We can then cast the problem of recovering node trajectories as a low-dimensional optimization (Figure 3-2). Specifically, we must recover four DOFs (degrees of freedom) per node: the quantities $\vec{p}_i$ and $\vec{v}_i$ in the expression

$$\vec{L}_i(t) = \vec{p}_i + t \cdot \vec{v}_i \tag{3.1}$$

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. Our goal is to construct, from local range measurements, a global motion solution in which relative trajectory of each node is represented by Equation 3.1 up to a global isometry (i.e., an arbitrary rigid translation, rotation, reflection and inertial or constant-velocity coordinate transformation).

To this end, we utilize the fact that the distance between two points moving with constant velocities forms a hyperbola (Figure 3-3). From the viewpoint of node i, 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:

$$r_{ji}(t)^2 = m_{ji}^2 + s_{ji}^2(t - t_{ji}^c)^2 \tag{3.2}$$

where t_{ji}^c denotes the time at which nodes i and j make their closest approach, m_{ji} denotes the node separation distance at this time, and s_{ji} denotes the relative speed $\|\vec{v}_j - \vec{v}_i\|$ (Figure 3-3). We define $H_{ij} = (s_{ij}, t_{ij}^c, m_{ij})$ as the *motion hyperbola parameters* for nodes i and j. We assume that the range observation is symmet-

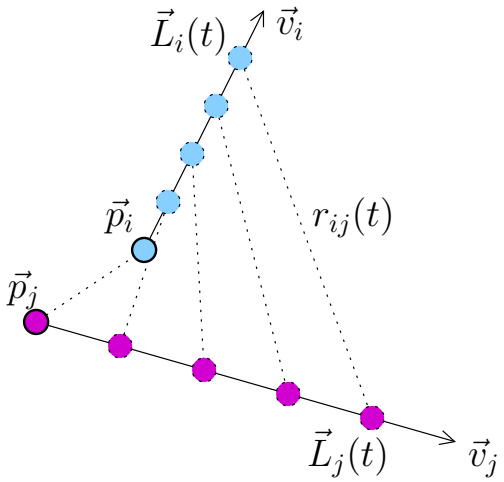


Figure 3-2: MBL recovers four DOFs per node.

ric, $r_{ji}(t) = r_{ij}(t)$, hence the motion hyperbola parameters are also symmetric, i.e. $H_{ij} = H_{ji}$. It should be noted that r_{ji} at any specific time can be computed from Equation 3.2. The motion hyperbola parameters serve as a mathematical abstraction of a relative trajectory, and can be estimated from a series of range observations. The MBL method constructs a global motion solution from all available motion hyperbola parameters. The subsequent sections explain how to accomplish this.

If the node motion is further generalized to a piecewise linear trajectory, the recovered motion solution would hold only for a finite duration before the node changes speed and heading. In this case, it is required to recompute a motion solution, and it is essential to detect these changes as soon as possible to minimize solution error. The detection is accomplished by use of a sliding observation window and a change point detection algorithm.

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.

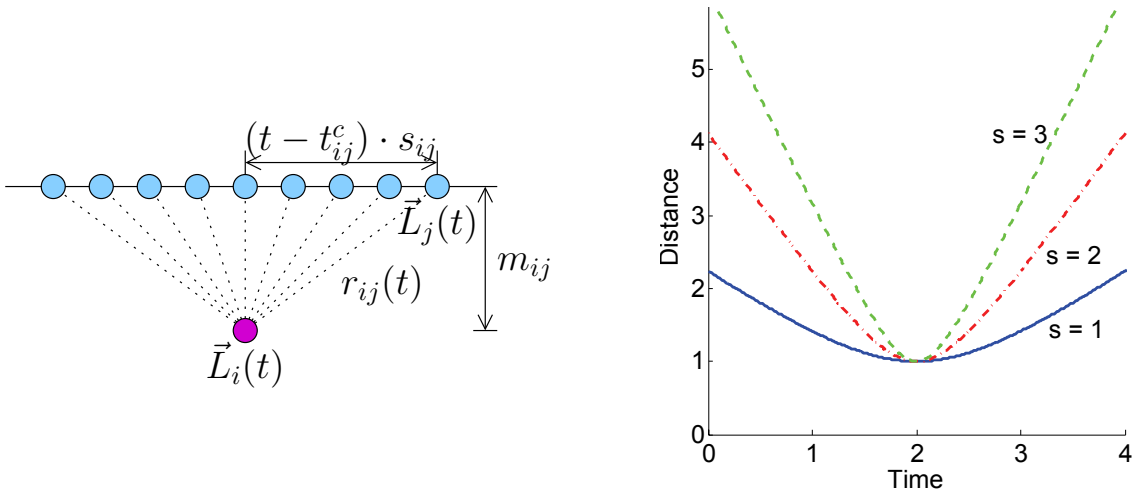


Figure 3-3: The distance between two points $\vec{L}_i(t)$ and $\vec{L}_j(t)$ moving with constant velocities traces a hyperbola with respect to time t.

3.1.1 MBL Algorithm at Each Node

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 one-hop 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 (Section 3.2), each node estimates the relative motion of each of its neighbors (Section 3.3), then constructs a local cluster by aligning computed positions and velocities (Section 3.4). Each node broadcasts its local cluster solution, enabling every other node to construct its own global view of the network (Section 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 it detects change in the observed error using a change point detection algorithm (Section 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. Also, there are many applications in which relative motion itself is useful, such as group maintenance or geometric routing [33, 41].

3.2 Hyperbola Estimation

In this section, we address the problem of estimating the motion hyperbola parameters between node i and j, $H_{ij} = (s_{ij}, t^c_{ij}, m_{ij})$, from time-stamped range measurements. We start by rewriting Equation (3.2) into a general quadratic form:

$$\begin{aligned} r^2_{ji}(t) &= m^2_{ji} + s^2_{ji}(t - t^c_{ji})^2 \\ &= s^2_{ji}t^2 - 2s^2_{ji}t^c_{ji}t + {t^c_{ji}}^2 + m^2_{ji} \end{aligned} \tag{3.3}$$

Now, suppose there are T range measurements in the most recent observation window. Given a sequence of T discrete range measurements between node i and j, $(r_n,\ t_n),\ n = 1, ..., T$, we consider the following quadratic model:

$$y_n := r^2_n = \alpha + \beta t_n + \gamma t^2_n + \epsilon_n \tag{3.4}$$

where ϵ_n is the error term. The problem of estimating motion hyperbola parameters in Equation (3.2) has been reformulated into a quadratic regression problem, in which y_n is a linear combination of parameters α, β, and γ. Once the estimates of regression coefficients are obtained from at least three measurements ($T \geq 3$), we can easily calculate the motion hyperbola parameters by comparing Equation 3.3 and 3.4 to

yield:

$$\hat{s}_{ji} = \sqrt{\hat{\gamma}} \tag{3.5}$$

$$\hat{t}^c_{ji} = \frac{\hat{\beta}}{-2\hat{\gamma}} \tag{3.6}$$

$$\hat{m}_{ji} = \sqrt{\hat{\alpha} - \frac{\hat{\beta}^2}{4\hat{\gamma}}} \tag{3.7}$$

where $\hat{x}$ denotes an *estimate* of x. It is to be noted that the relative speed s_{ji} and the distance at the closest approach m_{ji} must be physically nonnegative.

Parametric regression methods such as ordinary least-squares estimation are often used to estimate regression coefficients — $\hat{\alpha}$, $\hat{\beta}$, and $\hat{\gamma}$ in this case. However, the ordinary least-squares method is sensitive to the presence of outliers with significant leverage, because it minimizes the sum of squared residuals. This restricts usefulness of the ordinary least-squares method in our particular setting because of two reasons:

1. Range measurement data contain a nonnegligible amount of outliers (See Chapter 4);

2. The dependent variable y of the regression problem (3.4) is r^2, not r. If the ranging error were a Gaussian random variable with zero mean, the error term ϵ would not be Gaussian and would have a nonzero mean. Moreover, the effect of outliers in the error term ϵ is more magnified than that in the ranging error.

To guard the fitting result against harmful effect of outliers as well as to obtain more accurate estimates, we apply a *nonparametric robust quadratic fitting method* presented in [10] for estimation of the regression coefficients α, β, and γ. Assuming a quadratic model, the method takes the median value over all point estimators as the final estimator for each parameter. Because a small number of outliers have only marginal effect on the median value, the method performs well under the presence of outliers. That is, it naturally discards outliers when their fraction is not very significant. Moreover, the method is simple and non-iterative, thus more adequate for mobile sensor platforms than other robust regression methods or complicated

outlier rejection algorithms. In Appendix A, an analysis of the robust fitting method is presented with several examples.

The procedure of estimation for hyperbola motion parameters s_{ji}, t^c_{ji}, and m_{ji} is shown in Algorithm 1. 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.

It is to be noted that this procedure may return `failure` when the resulting parameters are physically impossible. For example, a parabola y with negative curvature $\gamma < 0$ will result in a complex-valued relative speed s_{ji}, which is physically invalid. In general, 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 (Figure 3-3) 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). Therefore, when `failure` occurs, a node must obtain additional range measurements to learn the true shape of the motion hyperbola.

Algorithm 1 Estimation of motion hyperbola parameters using nonparametric robust quadratic fitting method [10]. Note that a subscript i, j, or k points to an index in range measurements, not a node ID here.

$y_i = r_i^2,\ 1 \le i \le T.$

for all $\binom{T}{3}$ triplets of range observations, $(y_i,\ t_i)$, $(y_j,\ t_j)$, $(y_k,\ t_k)$ **do**
  $\tilde{\gamma}_{ijk} = \frac{1}{(t_k - t_j)}\{\frac{y_k - y_i}{t_k - t_i} - \frac{y_j - y_i}{t_j - t_i}\}$
end for
$\hat{\gamma} = \text{median}\{\tilde{\gamma}_{ijk}\}$

for all $\binom{T}{2}$ pairs of range observations, $(y_i,\ t_i)$, $(y_j,\ t_j)$ **do**
  $\tilde{\beta}_{ij} = \frac{y_j - y_i}{t_j - t_i} - \hat{\gamma}(t_j + t_i)$
end for
$\hat{\beta} = \text{median}\{\tilde{\beta}_{ij}\}$

for all T range observations $(y_i,\ t_i)$ **do**
  $\tilde{\alpha}_i = y_i - \hat{\beta} t_i - \hat{\gamma} t_i^2$
end for
$\hat{\alpha} = \text{median}\{\tilde{\alpha}_i\}$

if $\hat{\gamma} \ge 0$ and $\hat{\alpha} - \frac{\hat{\beta}^2}{4\hat{\gamma}} \ge 0$ **then**
  $\hat{s} = \sqrt{\hat{\gamma}}$
  $\hat{t}^c = \frac{\hat{\beta}}{-2\hat{\gamma}}$
  $\hat{m} = \sqrt{\hat{\alpha} - \frac{\hat{\beta}^2}{4\hat{\gamma}}}$
  return $\hat{s}$, $\hat{t}^c$, $\hat{m}$
else
  return `failure`
end if

3.3 Path Estimation Geometry

The parameters estimated in the previous section capture the relative position and motion of a pair of nodes. With three relations among three nodes i, j, and k, we can infer the relative motion of the *node triangle*, which we define as a set of linear trajectories of three connected nodes. A node triangle can be represented as a triangle of node positions at any time t with velocity vectors attached as in Figure 3-4(g).

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 (Figure 3-4(a)). Therefore, in node i's frame at any time t, node j has position $(m_{ji},\ s_{ji} \cdot (t - t^c_{ji}))$ and velocity $(0,\ s_{ji})$.

Likewise, the relative motion between nodes i and k can be established in i's frame (Figure 3-4(b)) up to an unknown reflection.

Now, we apply the remaining distance constraint r_{kj}, which can be calculated from Equation 3.2 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 (Figure 3-4(d)).

Figure 3-4(e) depicts relative velocities of all four cases in Figure 3-4(d) 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} (Figure 3-4(c)). By comparing the motions $\vec{v}^{(l)}_{kj}$ to $j\vec{k}$, $l = 1, 2, 3, 4$, with k's velocity, $\vec{v'}_{kj}$, to $j\vec{k}$ in Figure 3-4(c), one can identify the correct solution.

We do so by decomposing each vector $\vec{v}^{(l)}_{kj}$ in Figure 3-4(e) 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 (Figure 3-4(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

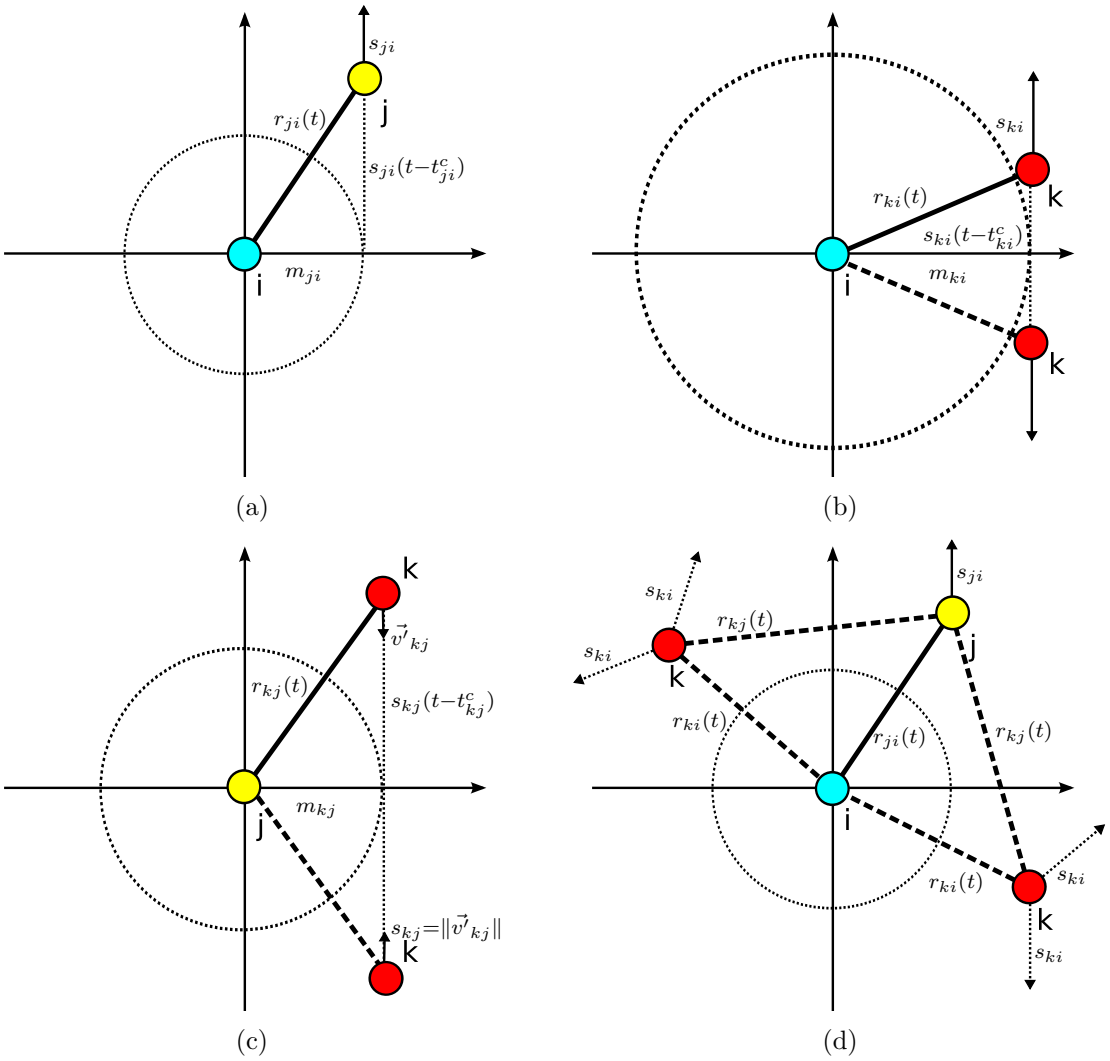


Figure 3-4: Recovering positions and velocities for nodes i,j,k in the relative coordinates of node i: (a) the relative motion of node j with respect to node i; (b) the relative motion of node k with respect to node i; (c) the relative motion of node k with respect to node j; (d) four possibilities for position and velocity of node k in the coordinates defined by (a).

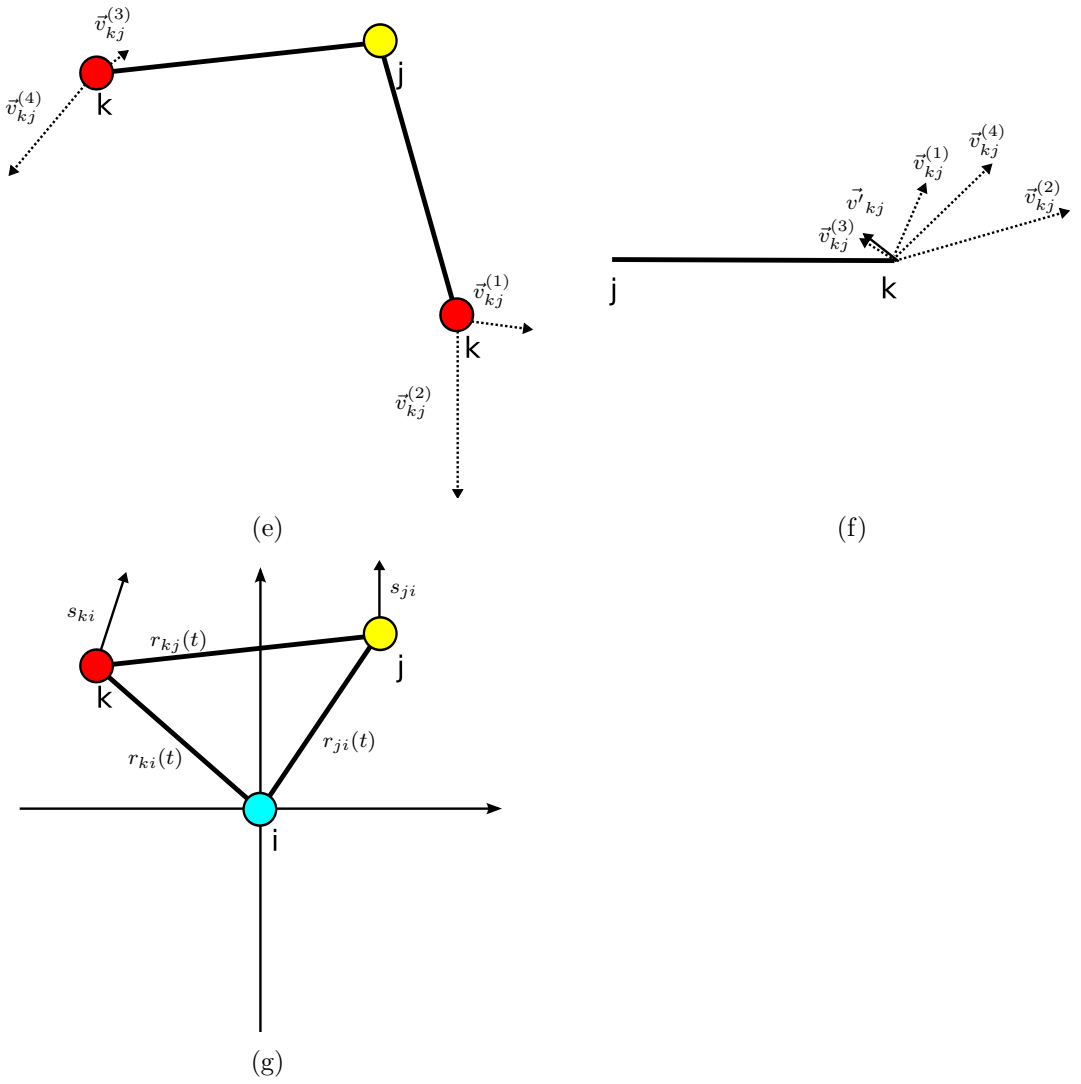


Figure 3-4: Recovering positions and velocities for nodes i,j,k in the relative coordinates of node i (contd.): (e) four possible relative velocities of node k with respect to j in (d); (f) the relative velocities of k to j compared to the known velocity $\vec{v}'_{kj}$; (g) the final triangle consisting of node i, j, and k.

defined by i and j (Figure 3-4(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:

$$v^*_{i-j-k} := \min_{l=1,2,3,4} \|\vec{v}_{kj}^{(l)} - \vec{v}'_{kj}\| < v^*_c \tag{3.8}$$

where v_c^* 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 $v_c^* = 0.5$ 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 in Section 3.3 shows how each node constructs node triangles from motion hyperbola parameters in its own frame. In the local cluster localization step, each node computes a local cluster at the specific time t from those node triangles. Each node becomes the origin of its local coordinates and estimates of relative locations and velocities of its neighbors using a process analogous to chained trilateration. In this step, we consider only node triangles that contains the self node i [1].

We first define a *trilateration dependency graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{C})$ where the vertex set $\mathcal{V}$ is a set of triangles constructed from node i and its one-hop neighbors at the specific time t, as in Section 3.3, and the edge $e_{ab} \in \mathcal{E}$ between vertex (node triangle) $\triangle_a$ and $\triangle_b$, $\triangle_a,\ \triangle_b \in \mathcal{V}$, is defined if:

1. triangle $\triangle_a$ and $\triangle_b$ share an edge; and

2. the unshared node in $\triangle_a$ is connected to that in $\triangle_b$.

That is, two vertices in the trilateration dependency graph are connected if one of the corresponding triangles can be localized by trilateration based on the other triangle in the underlying local cluster. In addition, we assign v_{i-j-k}^* in Equation 3.8, which indicates how consistent velocity estimates are, as an accuracy heuristic $c_a \in \mathcal{C}$ for each node triangle $\triangle_a \in \mathcal{V}$. Lower c_a indicates that the velocity estimates in triangle $\triangle_a$ are more consistent, giving a high possibility that $\triangle_a$ is more accurately constructed than others.

[1] There might exist other node triangles that do not contain the self node i. However, because all the triangles under consideration are constructed from one-hop neighbors, the number of those triangles is usually small and all one-hop neighbors can be localized using only the triangles containing node i

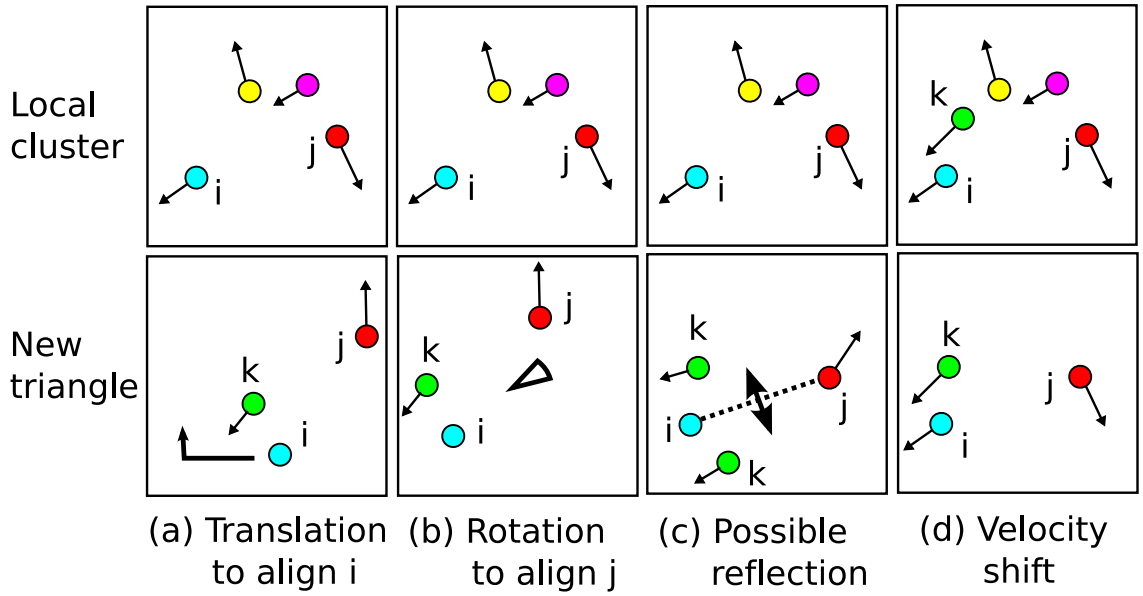


Figure 3-5: Aligning a local cluster and a new triangle sharing two nodes i and j by (a) translation, (b) rotation, (c) possible reflection, and (d) an inertial frame (velocity) shift.

For local cluster localization, each node repeatedly aligns triangles along a spanning tree in $\mathcal{G}$, adding a new node to the underlying local cluster at each step. We perform a greedy traversal on $\mathcal{G}$, based on the accuracy heuristic $\mathcal{C}$. Starting from the triangle with the lowest c, the local cluster localization algorithm visits a new triangle with the greedy strategy. If the new triangle contains a new node that is not localized yet, it is merged to a local cluster solution with the process described below. The local cluster localization step is described in Algorithm 2.

Now, let us consider adding a triangle $\triangle_{i-j-k}$ to a local cluster solution, given common nodes i and j (Figure 3-5). The procedure is as follows:

1. Translate the triangle in order to bring the positions of node i into alignment (Figure 3-5(a)).

2. Rotate the triangle, with velocity vectors, to bring the position of node j into alignment (Figure 3-5(b)).

3. Determine whether triangle must be flipped or not, using distances from k to the other nodes connected to k in the local cluster (Figure 3-5(c)).

4. Apply a velocity vector offset to equalize two views of $\vec{v}_i$ and $\vec{v}_j$ (Figure 3-5(d)).

5. Add the new node k to the local cluster.

Finally, it is to be noted that a trilateration dependency graph may be disconnected. This implies that if a node belongs only to a certain distinct connected subgraph $\mathcal{G}' \subset \mathcal{G}$, it can be localized via trilateration only if the local cluster localization algorithm starts from a triangle in $\mathcal{G}'$. Algorithm 2 does not guarantee that a local cluster found is maximal in terms of the number of recovered nodes, but can be tailored easily to yield the maximal one by repeatedly applying the algorithm on any remaining subgraph in $\mathcal{G}$ and replace the known "best" solution if the new local cluster solution is larger than the known one. In our implementation, however, this is not implemented because, for random deployment, the first solution was experimentally almost always identical to the best solution as the algorithm is most likely to start from a triangle in the maximal component of $\mathcal{G}$, yielding the maximal local cluster.

Algorithm 2 Local cluster localization.

Node i: The node self.
Neighbors: {N neighbors to be localized.}
Unlocalized: {A set of triangles that are not localized yet.}
TriangleQueue: {A priority queue of triangles.}
OUTPUT *LocalCluster*: {A set of (ID, position, velocity) tuples of localized nodes at time t.}

// Generate all triangles including node i and its neighbors.
for all node $j \in$ *Neighbors* **do**
  for all node $k \in$ *Neighbors*$\setminus\{j\}$ **do**
    Generate a triangle $\triangle_{i-j-k}$ at time t. {Section 3.3}
    if $\triangle_{i-j-k} \notin$ *Unlocalized* and $\|v^*_{i-j-k}\| < v^*_c$ **then**
      Add $\triangle_{i-j-k}$ to *Unlocalized*.
    end if
  end for
end for
if *Unlocalized* $= \emptyset$ **then**
  Return `failure`.
end if

Make a dependency graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

// Greedy traversal on $\mathcal{G}$
// 1: Initialization
From *Unlocalized*, pop $\triangle_0$ with the minimum $\|v^*\|$.
Initialize *LocalCluster* with $\triangle_0$.
Update *TriangleQueue* with neighbor vertices of $\triangle_0$ on $\mathcal{G}$.
// 2: Loop
while *TriangleQueue* $\neq \emptyset$ **do**
  From *TriangleQueue*, pop $\triangle' =< i, j, k >$ with the minimum $\|v^*\|$.
  Update *TriangleQueue* with neighbor vertices of $\triangle'$ on $\mathcal{G}$.
  if $\triangle'$ contains a new node $k \notin$ *LocalCluster* {i and j are common} **then**
    Merge $\triangle'$ into *LocalCluster* along i-j, adding k to *LocalCluster*.
  end if
end while

3.5 Global View Construction

This section describes how each node constructs a global motion solution of the network from the local clusters computed at each node as in Section 3.4. If a node receives a broadcast packet including local coordinates from another node, it saves the packet in a local cluster cache. The global view of the network is constructed these cached local clusters. Before performing global view construction, a node may check if each of the cached local clusters is outdated past a predefined "time-to-live" value and decide whether it is used for the global view construction or not.

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 [28].

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

$$(R', T') = \underset{R,T}{\operatorname{argmin}} \sum_i \left\| p_i^{(1)} - R(p_i^{(2)}) - T \right\|^2, \tag{3.9}$$

where R is a rotation (and possible reflection) and T is a translation. Equation 3.9 can be solved efficiently with linear complexity of the number of common nodes. The procedure is described below briefly. Refer to [28] for complete derivation.

Let N denote the number of common nodes, $\bar{p}^{(1)}$ and $\bar{p}^{(2)}$ denote the centroid of cluster 1 and 2 respectively. Then, the translation T' is simply the difference between

the centroid of 1 and the rotated centroid of 2:

$$T' = \bar{p}^{(1)} - R'(\bar{p}^{(2)}). \tag{3.10}$$

This translation is computed after we obtain R'. To find R', we first calculate the following matrix M:

$$M = \begin{bmatrix} S_{xx} & S_{xy} \\ S_{yx} & S_{yy} \end{bmatrix} \tag{3.11}$$

where

$$S_{xx} = \sum_{i=1}^{N} x^{(1)}x^{(2)} \ ; \ S_{xy} = \sum_{i=1}^{N} x^{(1)}y^{(2)} \ ; \ S_{yx} = \sum_{i=1}^{N} y^{(1)}x^{(2)} \ ; \ S_{yy} = \sum_{i=1}^{N} y^{(1)}y^{(2)}. \tag{3.12}$$

Now, the rotation matrix R' is computed as follows:

$$R' = M(M^T M)^{-1/2} \tag{3.13}$$

in which the matrix square root is computed via eigen-decomposition.

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 Adaptive Updates

Section 3.2–3.5 explain how each node computes a global view of the network under constant linear motion assumption. While the procedure described above can estimate relative positions of moving nodes more efficiently, by recovering velocities, than static localization methods, it should be able to deal with two types of changes in the

network: change in network topology and change in node motion. The first type of change, change in network topology, occurs when a node comes into or goes out of communication range of another. It also occurs when a node wakes up, goes into sleep mode, or is turned off. The second type of change, change in node motion, occurs when a node changes its moving direction, speed, or both.

The MBL method deals with these types of changes by recomputing its own local cluster estimate and propagating it to the network. Other nodes in the network can recompute or modify its global view estimate with the new local cluster estimate to learn about changes beyond their local neighborhood.

Each node employs a set of update rules to determine when it triggers a relocalization procedure against changes in the network. Table 3.1 summarizes the update rules. The first rule is to detect if a new node comes into the neighborhood. To prevent unnecessarily frequent relocalizations, we set a threshold of 3 node $\cdot$ sec, i.e., when one node persists for three successive seconds or three nodes persist for one second. The second rule, which detects a motion change, is explained in the following subsection. Finally, the third rule is used to learn changes in the network beyond one's sensing range. For the first and second event, a node relocalizes its local cluster as well as global solution of the network. For the third event, a node updates its global solution only.

In the following subsection, we describe how each node can detect motion changes in its local neighborhood in on-line fashion using a sequential change point detection algorithm.

Update rule	Description of a change
The counter tracking the number of new nodes exceeds 3 node $\cdot$ sec.	New nodes come into the neighborhood.
Detected a change (or prediction error) in motion via CUSUM algorithm.	This node detected a change in motion in the neighborhood.
Received a new local cluster packet.	Another node in the network detected change.

Table 3.1: Summary of update rules.

3.6.1 On-line Change Detection via CUSUM algorithm

Using the position and velocity estimates at a certain time, each node can compute relative location of other nodes in the network using Equation 3.1. The prediction error of any node's view, however, will grow over time because of two reasons. First, if other nodes change their motion to move in a different direction with a different speed, location predicted using Equation 3.1 is not valid any more. Second, even though node motions have not been changed, as uncertain trajectory estimates are extrapolated in time, error in the position estimate will grow linearly to the direction of velocity error.

We employ a change detection algorithm to deal with these situations. The basic idea is as follows. Suppose that a particular node, node A, has computed a local view. After that, node A will continue to measure distances between neighbors and itself. When measuring a distance to a neighbor node B, if A's estimate is still valid, prediction error of the distance measurements must be small and follow *a priori* ranging error distribution. If, otherwise, the prediction error is high and can be thought as not following the known ranging error distribution, node A determines change in node motion and triggers recomputation of its local view.

More specifically, the *on-line change detection* problem is formulated as follows [5]. Let r_k denote the measured distance and $\hat{r}_k$ denote the predicted distance to node B from node A. We consider a sequence of prediction errors (e_k), $k = 1, 2, ...$, where $e_k = r_k - \hat{r}_k$. Let us assume that, before the unknown change time t_0, e_k has a probability density function (pdf) $p_e(e; \theta_0)$, and after t_0, $p_e(e; \theta_1)$ with different parameters θ_0 and θ_1.

To detect a change in prediction error, we use the *two-sided cumulative sum* (two-sided CUSUM) algorithm [5]. We first describe the single-sided CUSUM algorithm to test between two hypotheses regarding the parameter of pdf:

$$\begin{aligned} \mathbf{H_0} &: \theta = \theta_0 \\ \mathbf{H_1} &: \theta = \theta_1. \end{aligned} \tag{3.14}$$

The CUSUM algorithm can be interpreted as a repeated application of the *sequential probability ratio test* (SPRT) [61]. After each measurement, the decision can be indecisive, in favor of $\mathbf{H_0}$, or in favor of $\mathbf{H_1}$ which indicates a change. If the decision is indecisive, the node takes the next measurement and the test continues. If the decision is to accept the null hypothesis $\mathbf{H_0}$, the test restarts[2]. If the decision is made in favor of $\mathbf{H_1}$, node A detects a change and triggers recomputation.

The decision is made based on the *log-likelihood ratio*:

$$s_k = \ln \frac{p_e(e_k; \theta_1)}{p_e(e_k; \theta_0)} \tag{3.15}$$

which represents the ratio of the likelihood that e_k is drawn with the parameter θ_1 over the likelihood with θ_0. The decision rule of CUSUM algorithm uses the following metric g_k, which can be written recursively as:

$$g_k = \sup(0,\ g_{k-1} + s_k). \tag{3.16}$$

Finally, the *alarm time*, at which the decision is made in favor of $\mathbf{H_1}$, is defined as:

$$t_a = \min\{k : g_k \geq h\}. \tag{3.17}$$

To apply Equation 3.16 and 3.17 to our change detection problem, probability density function of e, $p_e(e)$ must be known. In Chapter 4, a ranging error model of our target device, ultra-wideband radio, is developed and the resulting model is a mixture of small Gaussian error and large uniform error that models outliers. Because the Gaussian error is dominant in the ranging model and makes the CUSUM algorithm much simpler, we assume here that the pdf of e is Gaussian and the parameter of which change is to be detected is its mean. That is:

$$p_e(e; \theta) = \mathcal{N}(e; \theta, \sigma_R^2) \tag{3.18}$$

[2]In Equation 3.16, having a value $g_{k-1} + s_k$ below zero is interpreted as accepting $\mathbf{H_0}$. Replacing the negative g_k with zero means "restart".

where $\theta_0 = \mu_0$, and $\theta_1 = \mu_1 = \mu_0 + \nu$, where μ_0 and μ_1 are means before and after the change at t_0. It is further assumed that $\mu_0 = 0$, i.e. the pdf before change is zero-mean Gaussian.

The CUSUM algorithm above can be used if there is a single alternative hypothesis. In our setting, the ranging error can be either positive or negative. Therefore, we are interested in both positive and negative changes in the mean value. In this case, the solution is to use two CUSUM algorithms together [45]. If the mean value after change is either $\mu_0 + \nu$ or $\mu_0 - \nu$, the resulting two-sided CUSUM algorithm with Gaussian pdf assumption (Equation 3.18) is given by [5]:

$$g_k^+ = \sup(0, g_{k-1}^+ + e_k - \mu_0 - \frac{\nu}{2}) \tag{3.19}$$

$$g_k^- = \sup(0, g_{k-1}^- - e_k + \mu_0 - \frac{\nu}{2}) \tag{3.20}$$

$$t_a = \min\{k : g_k^+ \geq h \,\text{or}\, g_k^- \geq h\}. \tag{3.21}$$

The two-sided CUSUM algorithm above works well with the Gaussian ranging error assumption. However, if there are outliers in observations, the CUSUM algorithm will trigger false positive relocalization events. To combat this possibility, range errors are *winsorized* with a pre-defined threshold w [16]. That is, if the deviation of a range error from μ_0 is bigger than w, the error is replaced with $\mu_0 \pm w$. This *winsorization* technique makes false positive relocalization events occur less frequently. The resulting motion change detection algorithm is shown in Algorithm 3.

3.7 Complexity Analysis

In this section, we analyze the computational complexity of the MBL algorithm. Although the general problem of finding a graph embedding as well as unit-disk reconstruction problem are known to be NP-hard [3], it is also shown that trilateration graph realization can be done in polynomial time [18]. Our algorithm, which is based on trilateration, can be performed in polynomial time as well.

More specifically, Algorithm 1 examines $\binom{T}{3}$ triplets at maximum. Median-finding

Algorithm 3 Motion change detection algorithm using two-sided CUSUM algorithm.

$g^+ = 0$, $g^- = 0$

loop {for every new range measurement r to each node at time t}
  Compute range prediction $\hat{r}$ at time t from the local view estimate.
  $e = \hat{r} - r$.

  // Winsorization
  if $e > w$ **then**
    $e = w$
  else if $e < -w$ **then**
    $e = -w$
  end if
  // CUSUM
  $g^+ = \sup(0, g^+ + e_k - \mu_0 - \frac{\nu}{2})$
  $g^- = \sup(0, g^- - e_k + \mu_0 - \frac{\nu}{2})$
  if $g^+ > h$ or $g^- > h$ **then**
    $g^+ = 0$, $g^- = 0$
    Trigger `relocalization`.
  end if
end loop

is implemented using the randomized "quickselect" algorithm [12]. Although the worst-case complexity of the quickselect algorithm is $\mathcal{O}(l^2)$ where l is the number of elements, its expected runtime is $\mathcal{O}(l)$. Therefore the expected runtime of Algorithm 1 is $\mathcal{O}(T^3)$. This is comparable to the least squares method in which pseudoinverse computation has complexity of $\mathcal{O}(T^{2.807})$ [12]. If the length of an observation window is fixed, the fitting takes only a constant time.

For Algorithm 2, generating all node triangles take $\mathcal{O}(m^2)$ where m is the maximum node degree. The complexity of spanning tree traversal grows linearly with the numbmer of triangles, and each merging step takes $\mathcal{O}(1)$. Therefore, the overall complexity is $\mathcal{O}(m^2)$. This is achievable because only triangles including the self node are considered in this step.

For the global view construction step, there are n local clusters at maximum and any two clusters can have m shared nodes between them. Because the absolute orientation algorithm [28] has linear complexity in the number of common nodes (in Equation 3.11), the total complexity of this step is $\mathcal{O}(mn)$.

Chapter 4

UWB Radio Node Characterization

Starting in January 2007, we characterized a current-generation UWB radio, the Time Domain Corporation (TDC) PulsOn 210. We collected ranging measurements from the off-the-shelf UWB devices in indoor environments around MIT campus. The aim of the range data acquisition campaign was to gather range data from UWB radios in real indoor environments to develop a ranging error model that faithfully describes ranging behavior of the device in such environments. In Section 4.1, the overall characteristics of PulsOn 210 radio are described. In Section 4.2, we describe the setup for range measurement campaigns. In Section 4.3, we analyze the range data gathered from the campaigns and develop a ranging model of the UWB radio.

4.1 UWB Radio Characteristics

Time Domain Corporation PulsOn 210 radio is an ultra-wideband transceiver which operates at a center frequency of 4.7 GHz and a bandwidth of 3.2 GHz. The device has a small form-factor of 16.5 cm $\times$ 10.2 cm $\times$ 5.1 cm excluding antenna, which is suitable for a localization system. The TDC devices can range more than about 30 meters around obstacles in indoor environment. Range between a pair of radios is acquired via a two-way half-duplex ranging scheme, in which one radio sends a

Figure 4-1: Time Domain Corporation PulsOn 210 radio

ranging request and the other responds to the ranging request packet. The requester node measures a round-trip time-of-flight to calculate distance between the responder and itself. The radio compensates the round-trip time with electrical delays — the amount of time spent in internal logic — and leading-edge correction — the offset between where the signal preamble is acquired and the true leading edge of the signal — to make distance estimate more accurate. Ranging packets include a space for data payload; the estimated inter-device distance is piggybacked in the data packet so that the responder can also obtain the distance measurement. The data payload space can also be used for data communication.

The devices use a time-division scheme to schedule ranging requests, with the ranging protocol requiring about a few tens of milliseconds, yielding a maximum achievable ranging rate in any vicinity of approximately 30 Hz[1]. Our simulation in Chapter 5 uses a ranging frequency of 5 Hz to roughly capture the constraint that nearby nodes cannot range simultaneously.

4.2 Campaign Setup

A series of range data acquisition campaigns were performed in two different indoor locations of the Stata building, namely, the basement and the 3rd floor of the Stata

[1]Ranging rate is adjustable to obtain better ranging performance or larger maximum ranging distance.

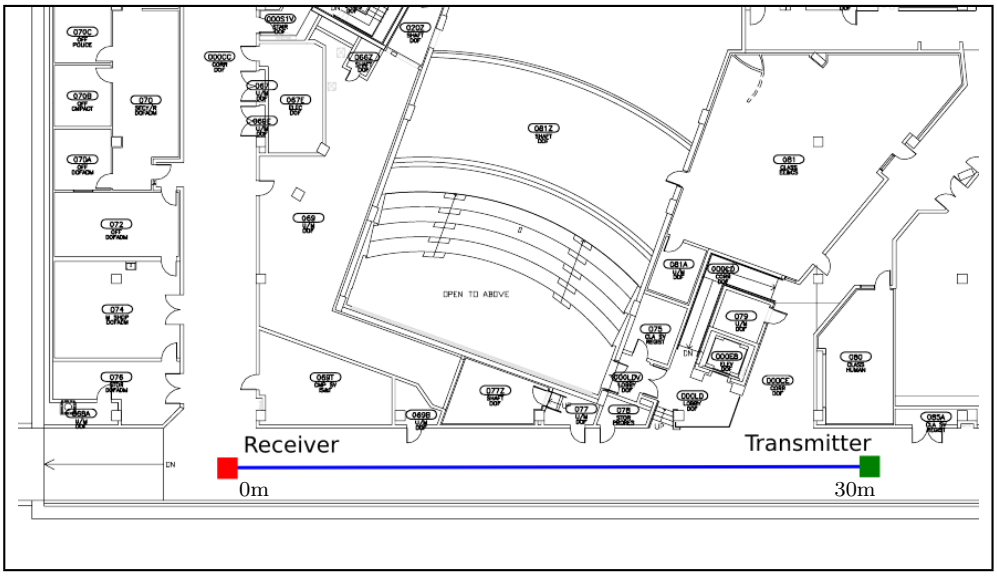
Location	Line-of-sight condition	Distance (m)	Granularity (m)
Stata basement hallway	LOS	1 - 30	1
CSAIL 3rd-floor	LOS	1.5 - 13.5	0.25
CSAIL 3rd-floor	Non-LOS	1.5 - 13.5	0.25

Table 4.1: Experiment conditions for range data acquisition campaign

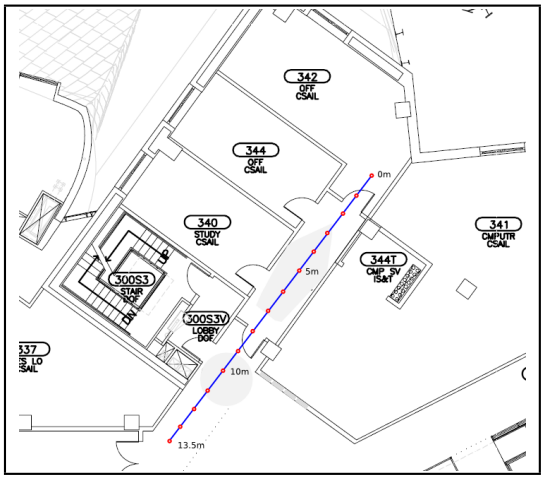
building. Range measurements were taken between two UWB radios by varying separation distance, fixing the receiver at one location and moving the transmitter so that the distance between the radios matches the ground-truth distance at each location. For each location, 1000 successful range measurements were collected. The radios were placed at the height of 89 centimeters from the ground. Table 4.1 summarizes environmental conditions for our data acquisition campaign.

Figure 4-2 shows locations where range measurement campaigns were conducted. The first campaign was conducted at a long and wide corridor in the basement of the Stata building, as marked in Figure 4-2(a). Since TDC PulsOn 210 radios can exchange ranging packets reliably up to approximately 30 meters without increasing transmission power beyond FCC regulation, we collected range data up to 30 meters with granularity of 1 meter. A data set acquired from this campaign constituted a basis for the ranging error model.

However, as shown in the next section (Section 4.3), range measurements gathered from the Stata basement campaign did not fully capture the ranging characteristics of UWB radios in indoor environments, because the wide corridor was considerably more open place than typical indoor spaces where walls, doors, and clutters exist nearby. In the latter types of environments, we can expect that reflection and loss of signal due to nearby objects adversely affect ranging performance of the devices. Therefore, we also conducted measurement campaigns at a narrow corridor on the third floor of the Stata building (Figure 4-2(b)). Along the experiment baseline, there were two doors with which we could evaluate the effect of non-line-of-sight (non-LOS) condition. Although we were able to collect range data only up to 13.5 meters in the third floor environment because of space limitation, these sets of measurement campaigns were also used for developing a reasonable ranging error model for simulation studies.



(a) Stata basement



(b) CSAIL 3rd-floor

Figure 4-2: Range measurement campaign locations.

4.3 Range Error Model

In this section, we develop a ranging error model for TDC PulsOn 210 UWB radios. We first define range error to be $\varepsilon = r_m - r$, where r_m is measured separation distance, and r is the ground-truth separation distance between radios.

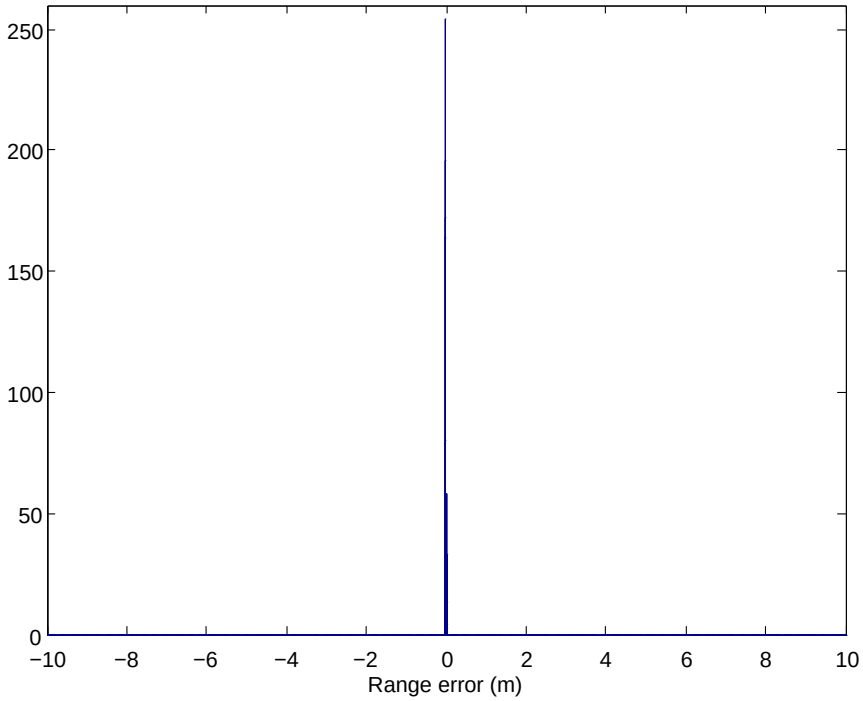
Figure 4-3, 4-4, and 4-5 show histograms of range error ε at $r = 7.5$ m in each environment in Table 4.1. From histograms over all data sets, we observed that that the data set of range measurements collected in Stata basement has few outliers, while that collected in CSAIL 3rd-floor either with LOS or non-LOS condition has a considerable number of outliers, whose associated range error extends up to ± 10 meters. While UWB time-of-flight ranging is generally accurate and resilient to multipath fading [65], large distance errors can occur when line-of-sight between radios is blocked [37], or when the amplitude of an impulse received along a directed path drops below the device's detection threshold [1]. On the other hand, if outliers are excluded, we could model range measurements as a Gaussian distribution in all three data sets. These observations lead to the following range error model (following [1]):

$$r_m = r + \varepsilon = r + \varepsilon_s + \varepsilon_l \tag{4.1}$$

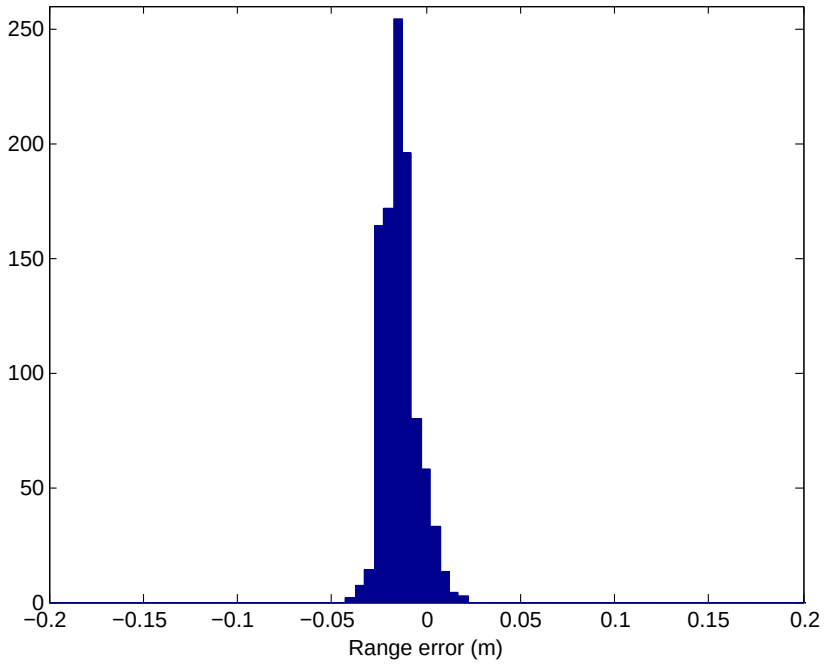
where ε_s is the small error with a Gaussian distribution and ε_l is the large error associated with outliers.

To model ϵ_s and ϵ_l separately, we partitioned observed range errors into small and large errors. For each data set, range measurements within ± 0.1 m from the median value were taken as the small errors, or inliers, and those outside this range were regarded as the large errors, or outliers. Figure 4-6, 4-7, and 4-8 show error bar plots with error bias and standard deviation, and fraction of outliers over all range measurement data at each distance. For the rest of this section, we will describe range error models for ϵ_s and ϵ_l in detail, then determine model parameters for each model with the measurement data.

First, we modeled the small error as a Gaussian random variable with a mean dependent on the true distance and standard deviation which is independent of the

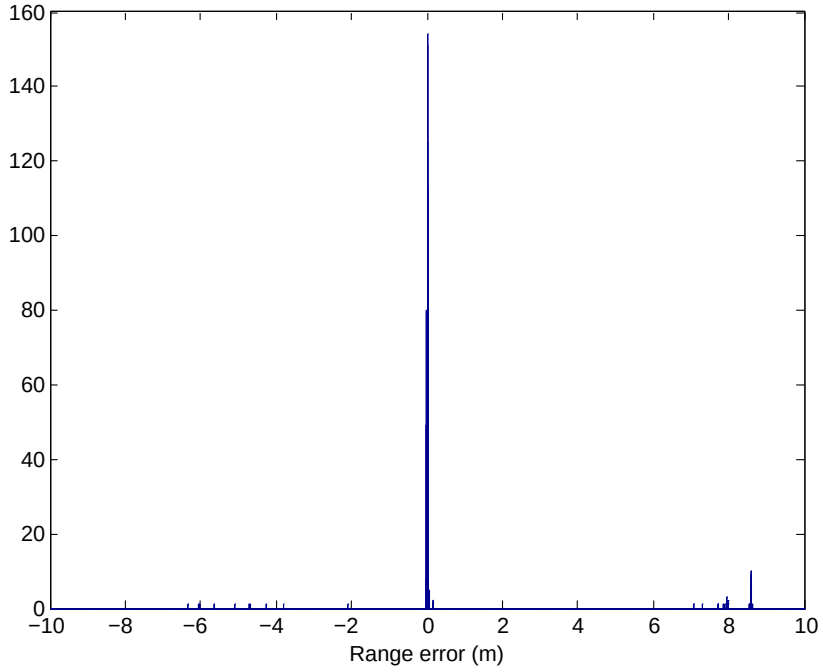


(a)

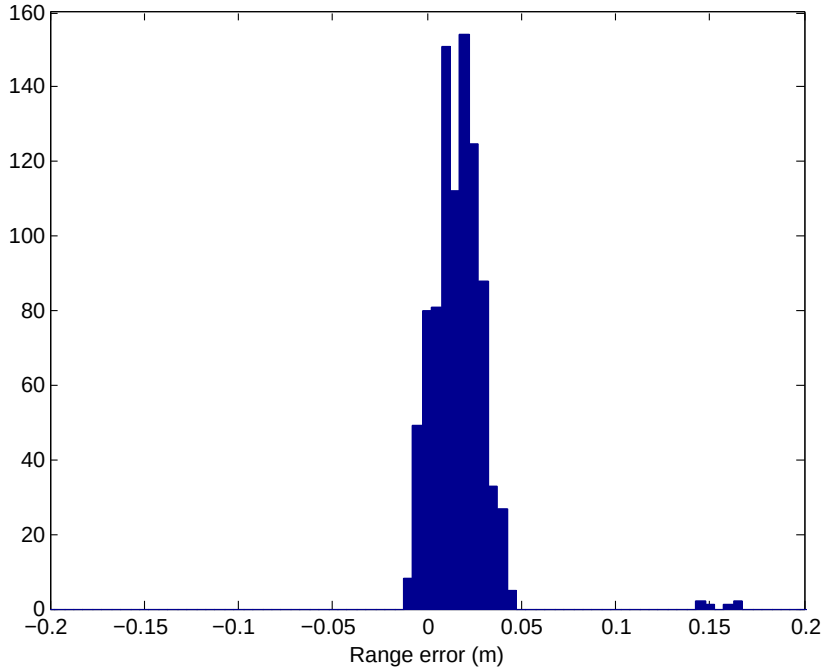


(b)

Figure 4-3: Histograms of range measurement error, $\varepsilon = r_m - r$, in Stata basement data at $r = 7.5$ m, at different x-axis scales: (a) $[-10\text{ m}, 10\text{ m}]$; (b) $[-0.2\text{ m}, 0.2\text{ m}]$.

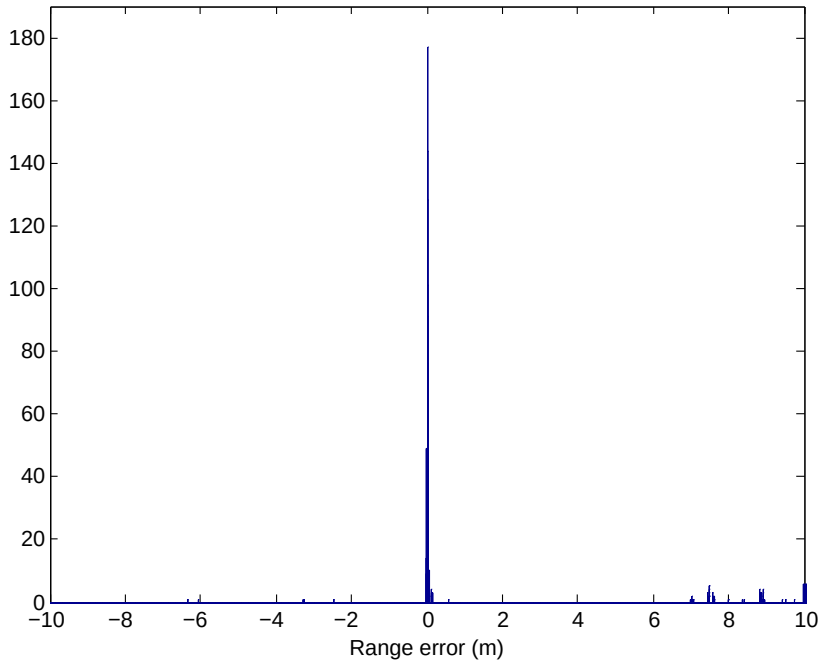


(a)

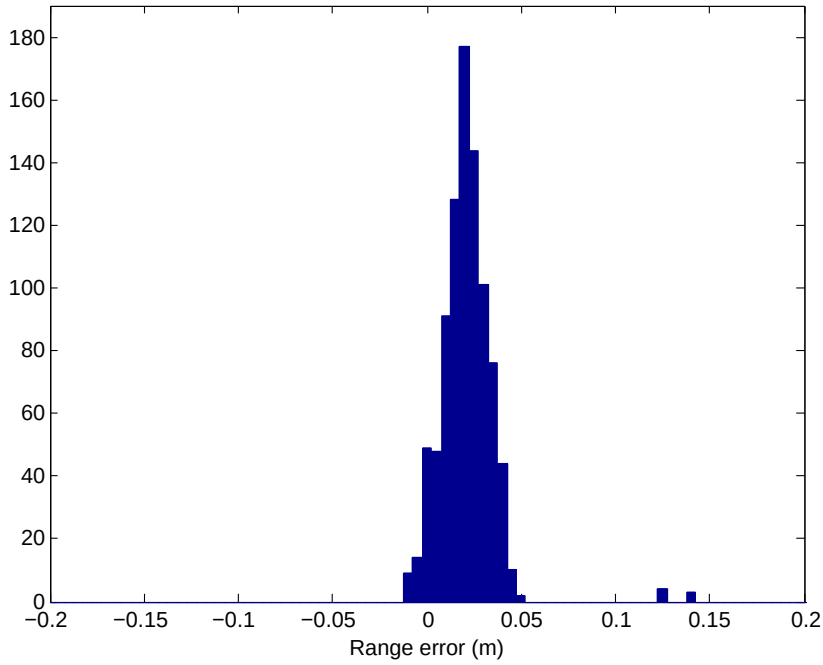


(b)

Figure 4-4: Histograms of range measurement error, $\varepsilon = r_m - r$, in CSAIL 3rd-floor LOS data at $r = 7.5$ m, at different x-axis scales: (a) $[-10\text{ m}, 10\text{ m}]$; (b) $[-0.2\text{ m}, 0.2\text{ m}]$.



(a)



(b)

Figure 4-5: Histograms of range measurement error, $\varepsilon = r_m - r$, in CSAIL 3rd-floor non-LOS data at $r = 7.5$ m, at different x-axis scales: (a) $[-10\text{ m}, 10\text{ m}]$; (b) $[-0.2\text{ m}, 0.2\text{ m}]$.

true distance as observed in Figure 4-6(a), 4-7(a), and 4-8(a). Therefore, the small error was modeled as:

$$\epsilon_s \sim \mathcal{N}(b(r),\ \sigma_\varepsilon^2) \tag{4.2}$$

with $b(r)$ represented as

$$b(r) = \nu_0\ \ln(1+r) + \nu_1 \tag{4.3}$$

where r is the true distance, ν_0 and ν_1 are bias model coefficients to be determined experimentally from range measurement data.

On the other hand, 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_o over $[-10, 10]$ meters and assigned its probability of occurrence as a binary random variable η:

$$\epsilon_l = \eta\ \epsilon_o \tag{4.4}$$

where η has probability mass function

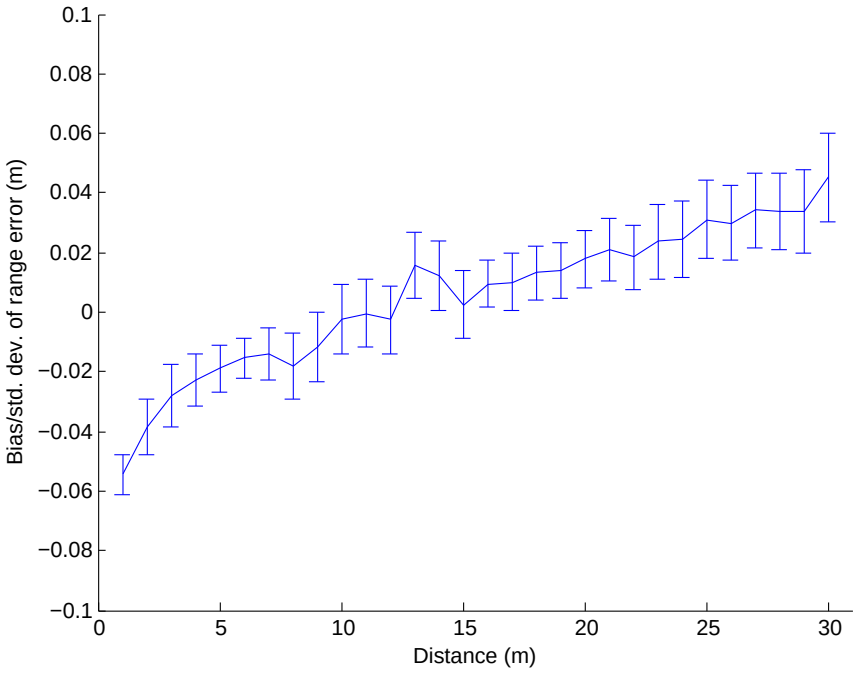
$$p_\eta(x) = \begin{cases} P_{OL} & \text{if } x = 1, \\ 1 - P_{OL} & \text{if } x = 0. \end{cases}$$

Here P_{OL} represents the probability of occurrence of large error and is determined from the measurement data. That is, we model an outlier that it occurs with the probability of P_{OL} and has additive large error ranging in $[-10, 10]$ meters.

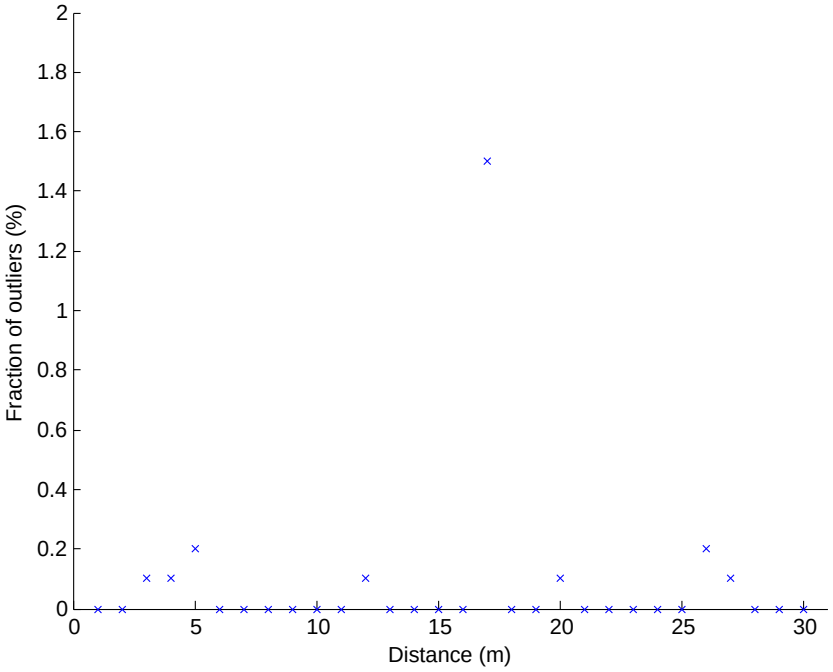
From the data sets, we determined *one* set of parameters ν_0, ν_1, σ_ε, and P_{OL}, which described a representative range error model for simulation. To this end, we

Data set	Bias (m)		Std. dev. σ_ε (m)		Outlier prob. P_{OL} (%)	
	ν_0	ν_1	Avg.	Max.	Avg.	Max.
Stata basement LOS	0.0323	-0.0786	0.0107	0.0149	0.08	1.5
3rd-floor LOS	0.0204	-0.0321	0.0107	0.0205	7.29	19.2
3rd-floor Non-LOS	0.0538	-0.0815	0.0112	0.0193	6.54	19.9

Table 4.2: Range error parameters from different data sets.

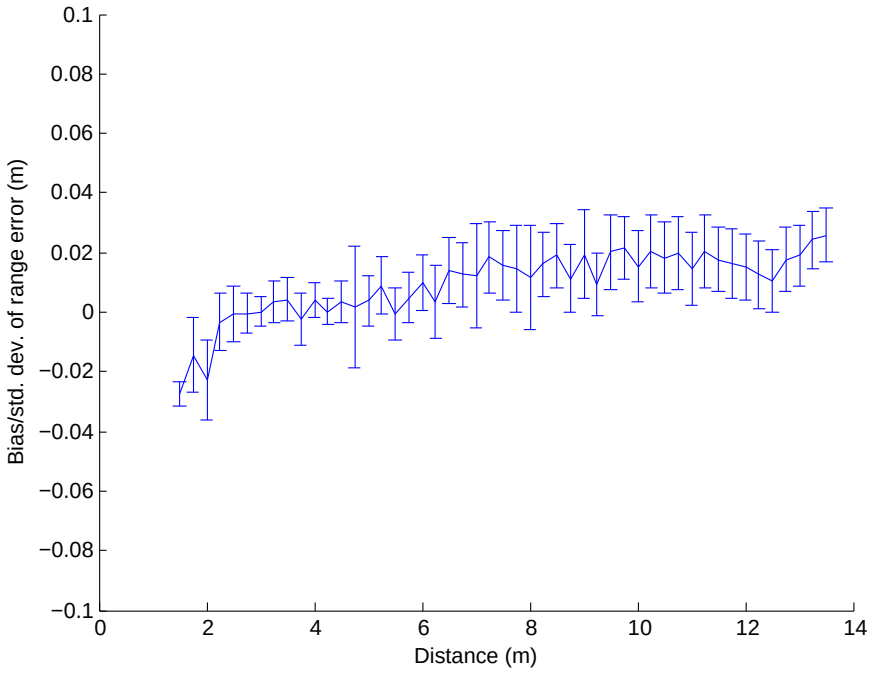


(a)

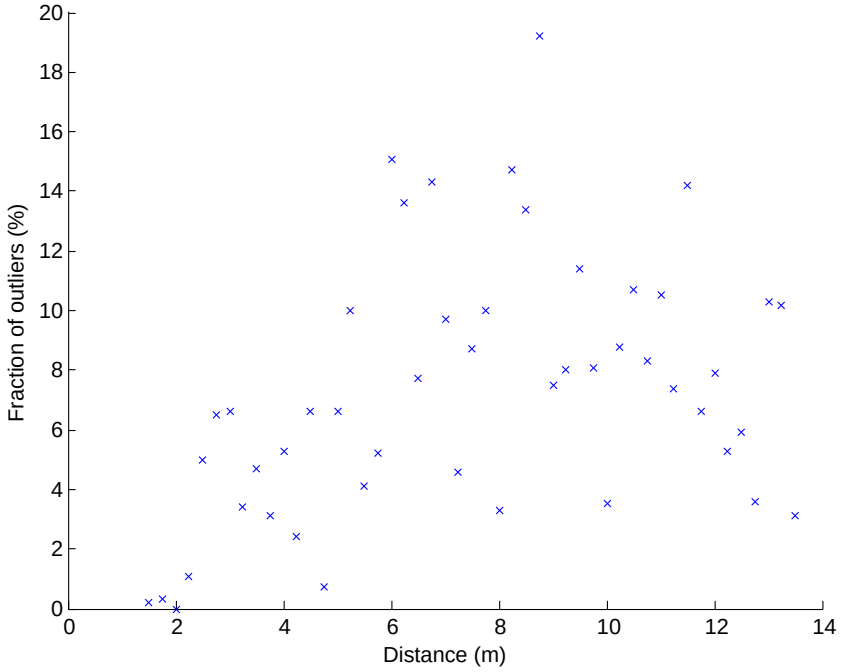


(b)

Figure 4-6: Range error characteristics for Stata basement LOS data set: (a) Bias and standard deviation of small errors; (b) Fraction of outliers (large errors) in the entire range data.

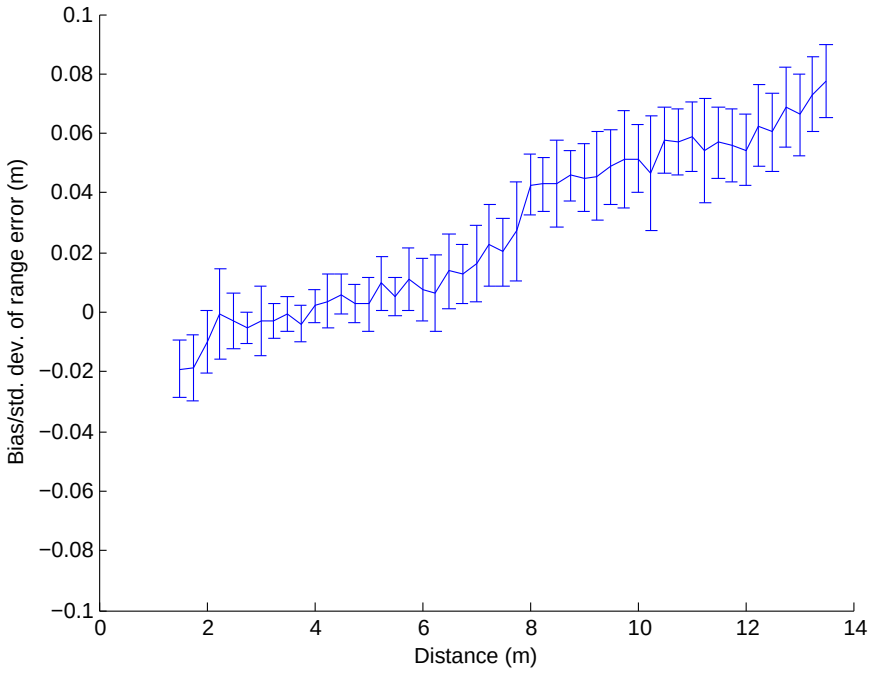


(a)

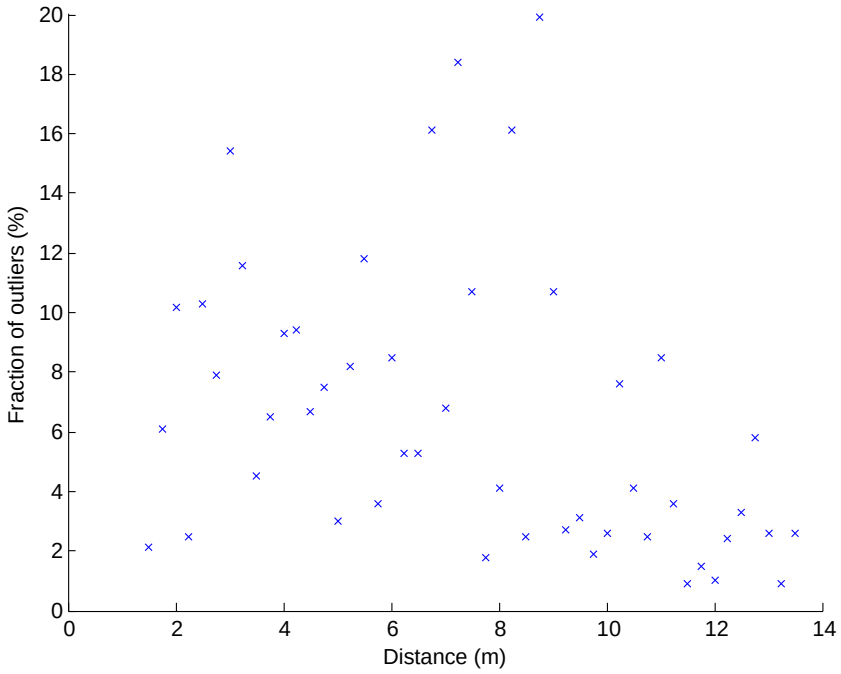


(b)

Figure 4-7: Range error characterstics for 3rd-floor LOS data set: (a) Bias and standard deviation of small errors; (b) Fraction of outliers (large errors) in the entire range data.



(a)



(b)

Figure 4-8: Range error characteristics for 3rd-floor Non-LOS data set: (a) Bias and standard deviation of small errors; (b) Fraction of outliers (large errors) in the entire range data.

first examined the parameters for each data, which is summarized in Table 4.2. These parameters are combined to yield a representative range model. For the error bias, we used bias model coefficients ν_0 and ν_1 from Stata basement LOS data, because it was obtained from the entire working range of a UWB radio as well as it fitted reasonably on the other range data. Standard deviation of small errors σ_ε and outlier probability P_{OL} was chosen conservatively as 0.03 m and 7 %, respectively, to reflect the effect of cluttered indoor environments.

Finally, we summarize our range error model for TDC PulsOn 210 UWB radios below:

$$r_m = r + \varepsilon = r + \varepsilon_s + \varepsilon_l \tag{4.5}$$

where

$$\epsilon_s \sim \mathcal{N}(b(r),\ 0.03^2)$$
$$b(r) = 0.0323\ \ln(1+r) - 0.0786$$

and

$$\epsilon_l = \eta\ \epsilon_o$$
$$\epsilon_o \sim \mathcal{U}(-10,\ 10)$$
$$p_\eta(x) = \begin{cases} 0.07 & \text{if } x = 1, \\ 0.93 & \text{if } x = 0. \end{cases}$$

This model was used throughout the simulation studies in the next chapter.

Chapter 5

Experimental Results

In this chapter, we evaluate the MBL method under a variety of simulation environments to measure its effectiveness for mobile wireless sensor networks. Section 5.1 describes how the simulation was performed. In Section 5.2, we study the temporal characteristics of the algorithm. Section 5.3 presents statistics that represent the overall behavior under varying conditions. Section 5.4 compares our method to MDS-MAP, a well-known relative localization technique. In Section 5.5, the effect of our update scheme is considered with a grid mobility model. Finally, in Section 5.6, the performance of the algorithm as well as its possible extensions are discussed.

5.1 Simulation Environment

5.1.1 MBL Simulator

We evaluated the proposed MBL method on a discrete-event simulator developed using Python and C (Figure 5-1). The MBL simulator was developed so as to accommodate various needs for validation and evaluation of the MBL method; it facilitated our understanding of the behavior of the MBL method through visualization of each part, and provided easy set-up of the simulation parameters for evaluation of the method. In addition, the use of both Python and C offered high level of flexibility in development as well as fast computation of numerical subroutines.

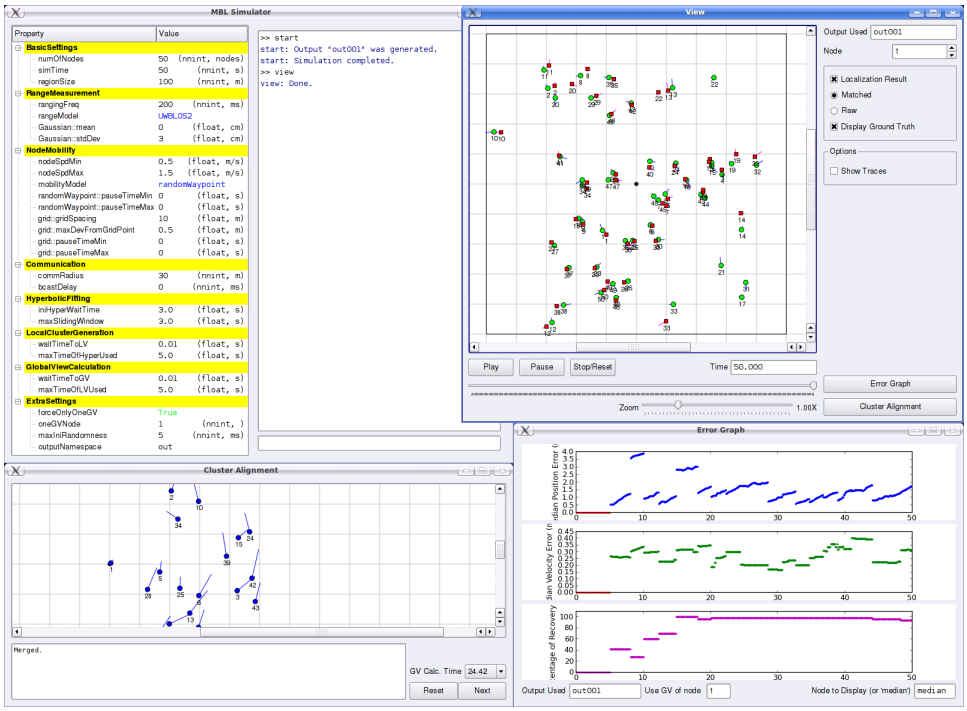


Figure 5-1: MBL simulator.

In the simulator, every node runs the same code and does not have any shared memory among nodes. All the communications between nodes are handled via the "arbiter" module, which determines if ranging and communication between any pair of nodes is feasible at a certain time based on locations of those nodes. The arbiter module also puts range error in every ranging packet according to the range error model explained in Chapter 4. The combination of discrete-event simulator and the use of arbiter module enabled us to capture temporal characteristics of the proposed algorithm, which is difficult to capture in a batch simulation.

5.1.2 Simulation Setup

In simulation, we adopted *unit-disk graphs* [11] to model the topology of networks. In the unit-disk graph model, two vertices (sensor nodes) are connected if two unit circles centered on each vertex intersect with each other. Although the unit-disk graph is a highly idealized model as omnidirectionality of signal propagation is assumed, it provides a convenient topology model for our simulation. It should be noted that

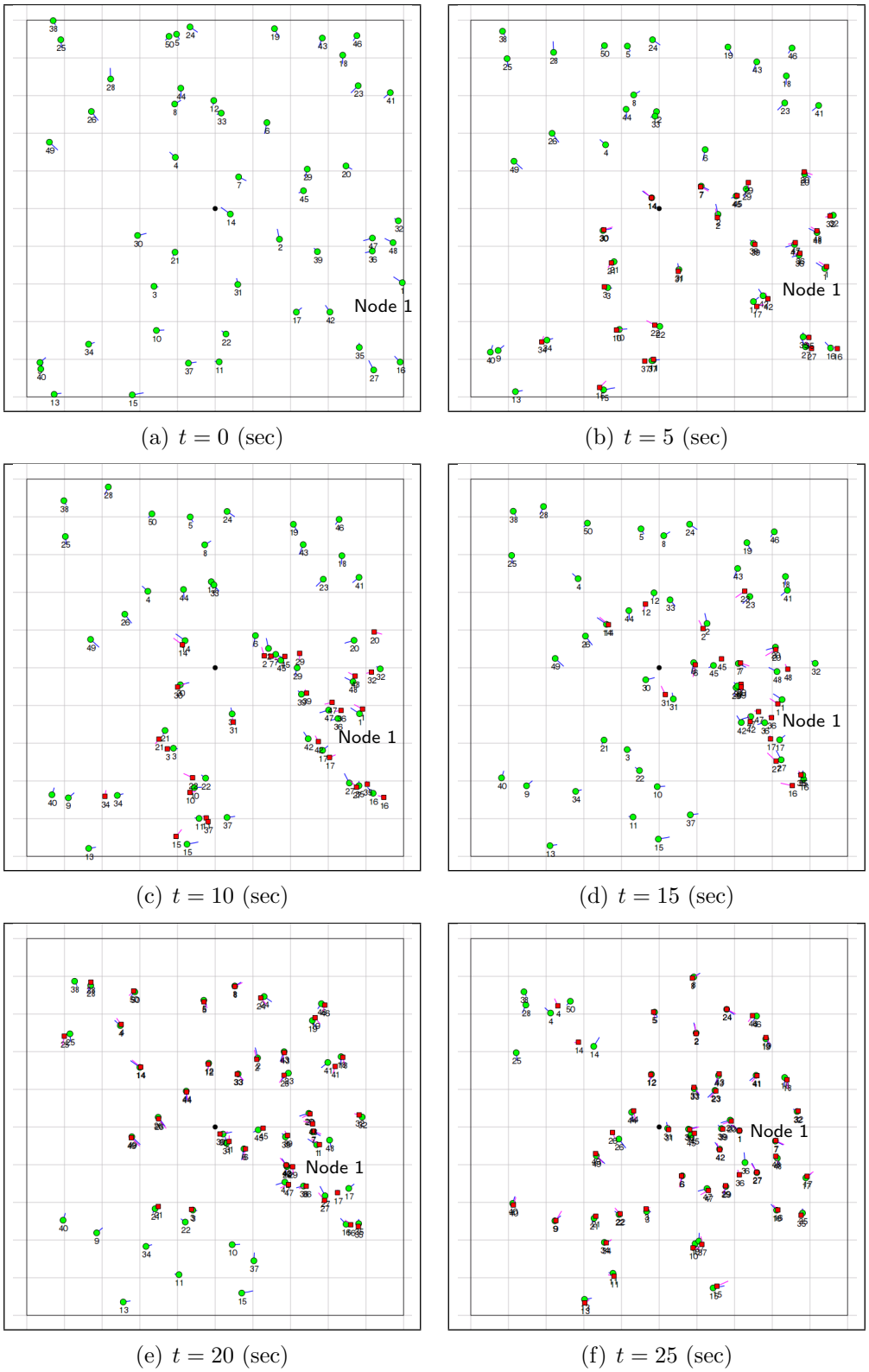
our method does not mandate such an omnidirectionality in communication — it is assumed only for simulation.

We considered two kinds of mobility models, *random waypoint mobility model* and *grid mobility model* for simulation. The random waypoint mobility model [31] is a commonly used mobility model for mobile wireless network research [7, 29]. In this model, each node moves from a waypoint to another waypoint with a randomly drawn velocity. After reaching the waypoint, the node randomly chooses the next waypoint in the area and proceeds to it with a new velocity. The grid mobility model is what we developed to depict a simplified mobility pattern of humans in indoor environment. In this model, nodes are constrained to move only on grid edges and can change direction or speed at corner points. The corner points have a pre-defined transition probability from which each node determines the next moving direction, such as north, east, west, and south. For both mobility models, we considered an average speed range from 0.5 m/s (slow walking) to 2.5 m/s (jogging).

The output of the MBL algorithm executed at a certain node is a relative position and velocity of other nodes in the network with respect to its own self-centered frame of reference. For comparison, position and velocity estimates are aligned to the ground-truth in least-squares sense. For the matching, we used the same routine that is used for the inter-cluster alignment (Section 3.5). After matching, evaluation metrics such as position error, velocity error, and percentage of recovery are computed every 0.1 second.

5.2 Network Example

This section illustrates the temporal characteristics of the MBL algorithm with an example (Figure 5-2). In this example, 50 nodes were simulated for 30 seconds in a 100 m $\times$ 100 m region, yielding an average node degree of 15. Nodes moved with a random waypoint mobility model. Initially, at $t = 5$ sec, upper nodes were not localized in the global view solution of node 1 because of spatial gap between upper nodes and lower nodes. Despite the average node degree was 15, which is enough



(a) $t = 0$ (sec)

(b) $t = 5$ (sec)

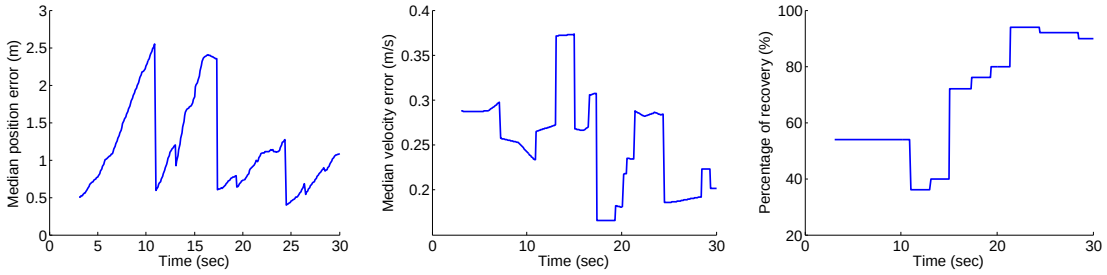
(c) $t = 10$ (sec)

(d) $t = 15$ (sec)

(e) $t = 20$ (sec)

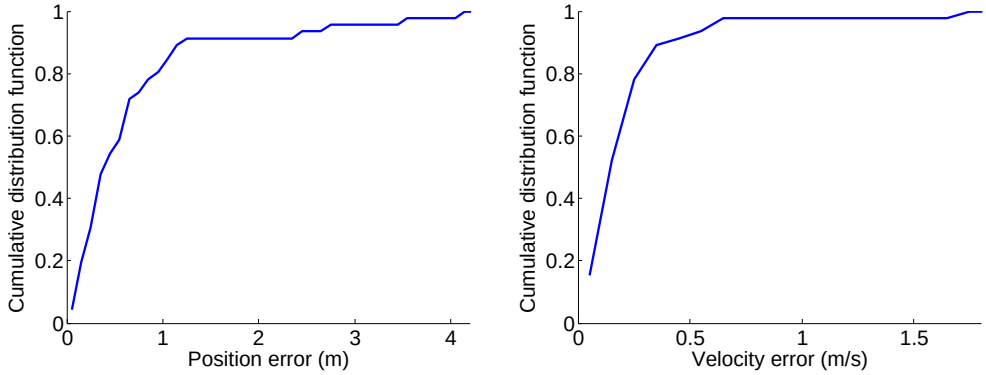
(f) $t = 25$ (sec)

Figure 5-2: MBL example. Each plot depicts ground truth (circle, green) and a computed view from node 1 (rectangle, red) at each time: (a) Initial node deployment; (b)-(d) Some nodes are not localized due to low connectivity; (e)-(f) Almost all nodes are localized as nodes are moving.



(a) (b) (c)

Figure 5-3: (a) Position error, (b) velocity error, and (c) node availability of the MBL example in Figure 5-2.



(a) (b)

Figure 5-4: Cumulative distribution function of (a) position error and (b) velocity error at $t = 25$ *sec*.

for localization in trilateration-based algorithms, a moving node in a mobile network may have occasionally had insufficient node degree due to its mobility. At $t = 15$ sec, nodes around the lower right corner were disappeared from the view because they became separated from node 1. At that time, local cluster information around the lower right region had been outdated in the local cluster cache of node 1. However, as nodes came closer, estimates of almost all nodes became available at $t = 25$ sec.

Figure 5-3 shows position, velocity, and percentage of recovery over time. Trajectory estimation was triggered according to prediction error as described in Section 3.6. Figure 5-4 shows cumulative distribution function of position error and velocity error at $t = 25$ sec respectively. More than 90% of nodes are localized within position error of ~ 1 m and velocity error of ~ 0.3 m.

5.3 Accuracy, Precision, Availability, and Efficiency

We assessed the performance of our MBL method using four 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. The *Efficiency* metric is represented as the average inter-computation time between relocalization events.

We simulated 50 nodes moving for 50 seconds with a random waypoint mobility model (with speed chosen uniformly in 0.5-1.5 m/s) in a degree-15 network (size 100 m $\times$ 100 m) while ranging at 5 Hz. Each node gathers range measurements for about 3 seconds whenever it needs to relocalize. We evaluated the Accuracy and Precision metrics for position and velocity 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 and Efficiency are shown in separate plots. Each box is computed from 100 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 and velocity error within 2 m and 0.3 m/s respectively for for ranging frequencies above 1 Hz (Figure 5-5(a) and 5-5(b)). As ranging becomes faster, localization result becomes more accurate. However, the effect of increasing ranging rate diminishes after 5 Hz because of the fixed size of the range observation window. The algorithm estimates trajectories for more than 80% of nodes when ranging occurs faster than 1 Hz (Figure 5-5(c)). For lower ranging frequencies, most nodes were unable to localize, collecting too few range samples within any observation window. Also, the algorithm relocalizes every 5 seconds on average except 1 Hz (Figure 5-5(d)). This does not mean that the motion change detection algorithm becomes more efficient at low ranging frequency. It is simply because, in 1 Hz ranging, the motion change detection algorithm requires

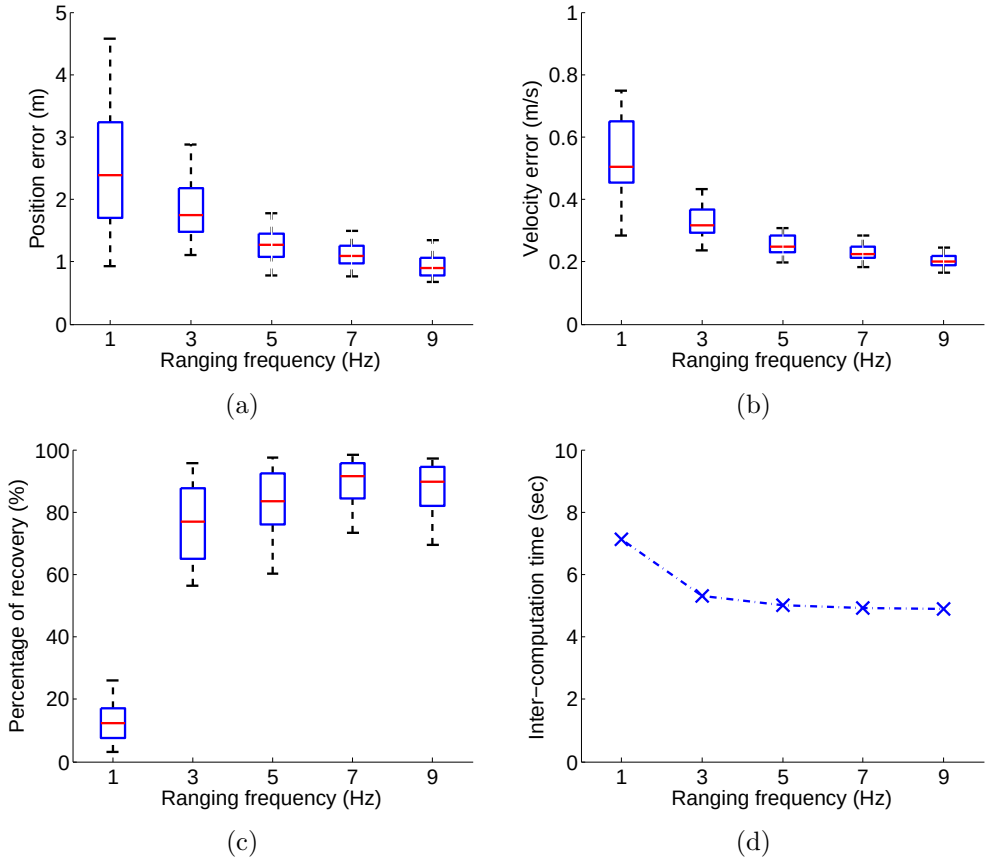


Figure 5-5: Time-averaged trajectory error, node availability, and computation efficiency as ranging rate increases.

longer time to make a decision as range observations arrive only once a second.

We also evaluated the growth and variation in recovered trajectory error as node speed increases (Figure 5-6). In general, faster node motions have both beneficial and harmful effect on the algorithm's position and velocity estimates. On one side, faster speed makes the slope of motion hyperbolae steeper, making the hyperbola fitting learn the overall shape of hyperbolae better. On the other side, it degrades the performance of the algorithm since fewer ranging observations can be gathered while any given node is within communication range. Additionally, at high speeds, we need to take account of faster divergence of true node motion after motion change from the predicted motion. We also observe decreasing efficiency at higher node speed in Figure 5-6(d), because motion change occurs more frequently.

Finally, we evaluated growth and variation in recovered trajectory error as the

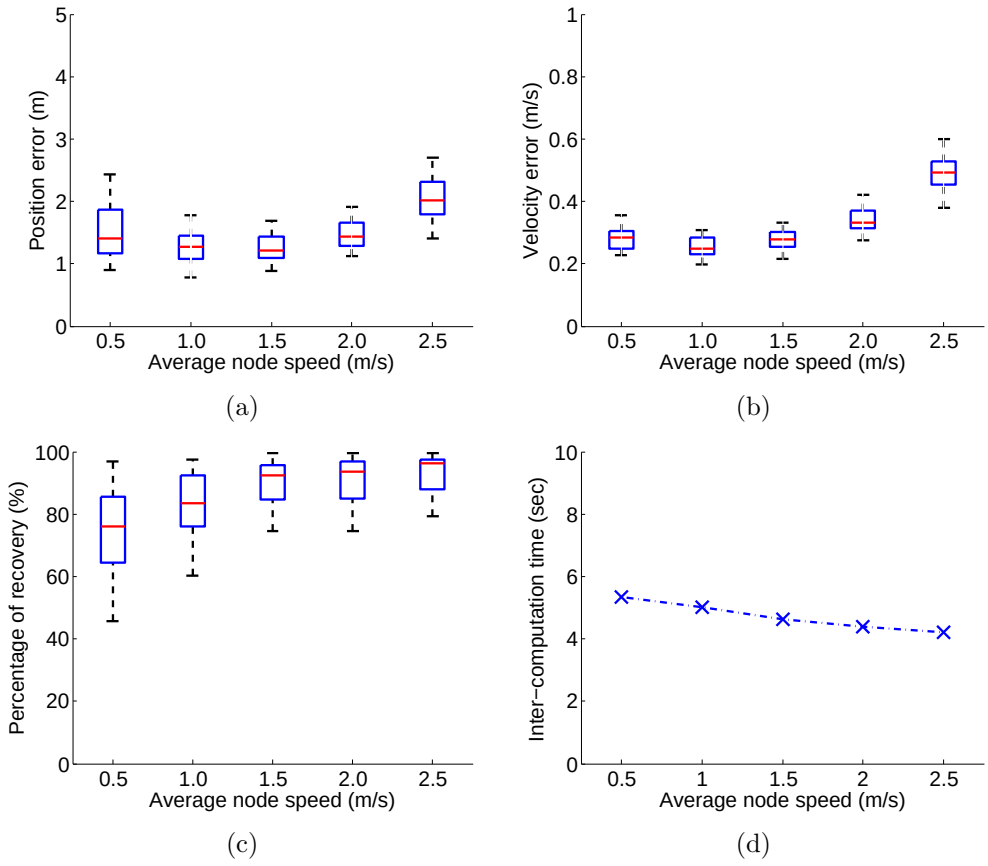


Figure 5-6: Time-averaged trajectory error, node availability, and computation efficiency as node speed increases.

node degree increases (Figure 5-7), 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 [42], with an availability transition similar to that observed in other settings [64]. In addition, it should be noted that at low densities such as degree of 5, any trilateration-based localization technique cannot achieve 100% availability. We confirmed through simulation that only 80% of randomly deployed network of degree 5 is a trilateration graph (a graph in which a trilateration

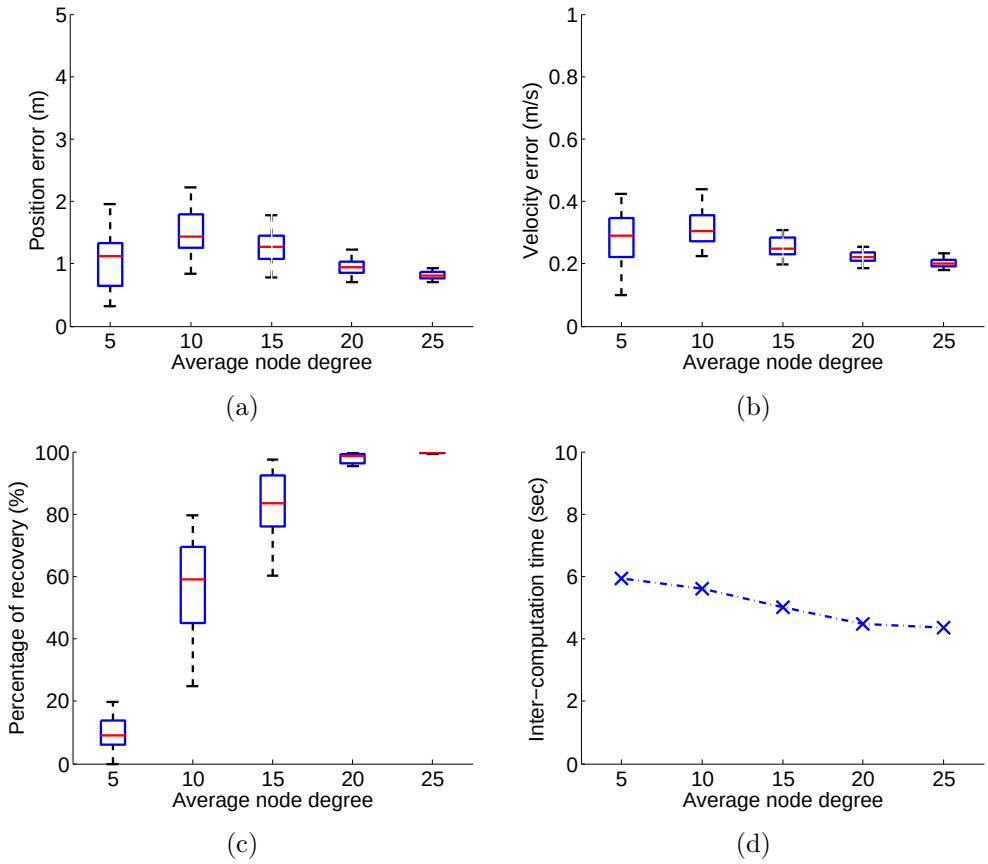


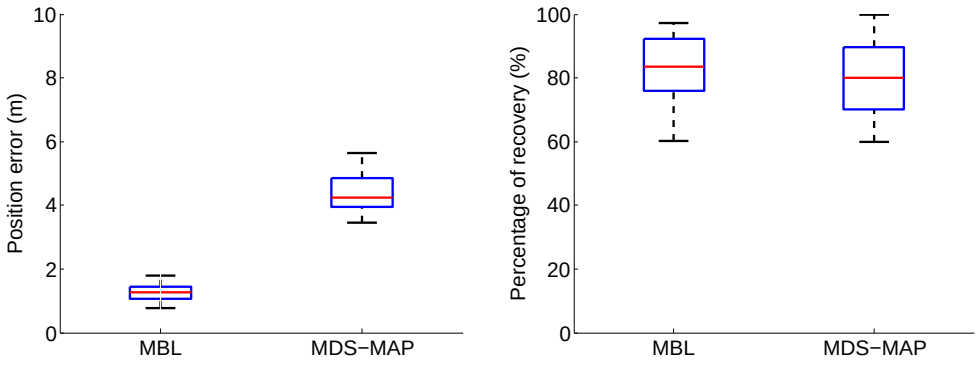
Figure 5-7: Time-averaged trajectory error, node availability, and computation efficiency as node degree increases.

ordering exists).

5.4 Comparison with MDS-MAP

In developing the MBL algorithm, one of the primary design considerations was the efficiency of the algorithm. The MBL algorithm started from an idea that recovering trajectories instead of location points would reduce the computational burden in mobile networks than repeating a static algorithm. In this section, we evaluate the efficiency of the MBL algorithm against the MDS-MAP [56] method, a well-known relative localization technique for static networks.

Both MBL and MDS-MAP method used the same base configuration of Section 5.3. Because the MBL method relocalized every 5 seconds on average under



(a) (b)

Figure 5-8: Comparison of MBL and MDS-MAP: (a) position error (b) node availability.

that configuration, MDS-MAP was set to run at the approximately same rate of 5 seconds. That is, we equalized efficiency of both methods and measured localization performance for each algorithm. Once MDS-MAP computed position estimates, they remained as a valid solution for the next 5 seconds. Note that MDS-MAP is a centralized algorithm and inadequate for practical mobile network locationing due to communication latency involved.

Figure 5-8 shows a comparison result. The MBL method localized the network with the median position error of 1.3 m, whereas MDS-MAP method obtained position error of 4.1 m, showing three times difference in accuracy. Node availability was similar. This inferior performance of MDS-MAP can be explained in two ways. First, MDS-MAP uses shortest-paths to approximate inter-node distances between non-neighbor nodes. This approximation induces a fair amount of error in the final location solution. Second, more importantly, a position estimate at a certain time is not valid at a different time for mobile nodes. Therefore, location estimates diverge much faster than in MBL which recovers node velocity together. Of course, it is possible to enhance accuracy by increasing the repeating rate in static methods , however, it will significantly increase computational and communication burden on the network.

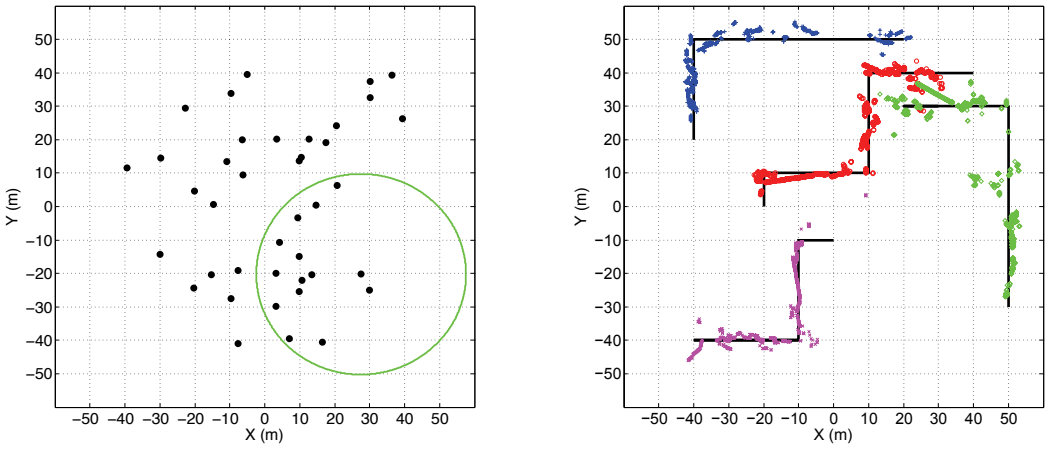
5.5 Frequent Updates

Finally, we considered the effect of our adaptive update rule that triggers localization only when observed ranges deviate significantly from prediction (Section 3.6). While this rule saves computation time, it might sacrifice about few meters of positioning accuracy before relocalization. Thus we compared MBL with adaptive update rules to that with periodic updates forced at 2 Hz.

We simulated a network of 40 nodes moving for 100 seconds within a square one hundred meters on a side (Figure 5-9(a)). For this example, we employed the grid mobility model described in Section 5.1.2. Each node repeatedly generates path segments by randomly selecting an axis-aligned velocity and distance (Figure 5-9(b)). We adjusted the velocity distributions to force motion transitions to occur in closely-spaced bursts separated by about 30 seconds (Figure 5-10(a)), 10 seconds (Figure 5-10(b)), and 3 seconds (Figure 5-10(c)) of transition-free motion.

Likewise, we ran the large-network experiment with periodic updates forced at 2 Hz. Figures 5-11(a), 5-11(b), and 5-11(c) show the corresponding plots of errorover time.

Table 5.1 summarizes the algorithm's performance for two different update schemes in each of the three regimes. In the table, velocity error is decomposed into speed and heading error. The computation cost grew roughly six-fold (58,221 vs. 10,071 triangles considered), but the errors 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.

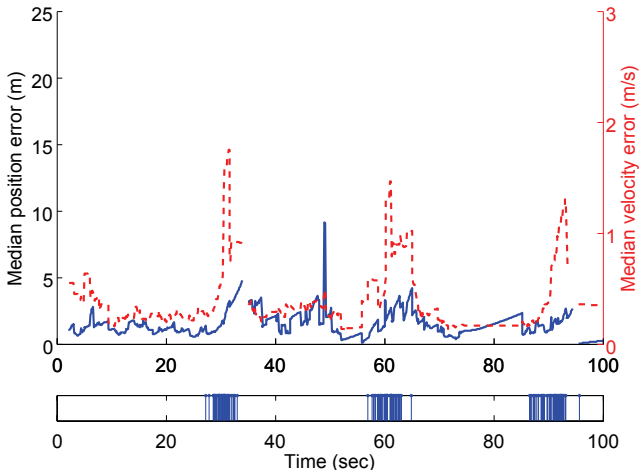


(a) (b)

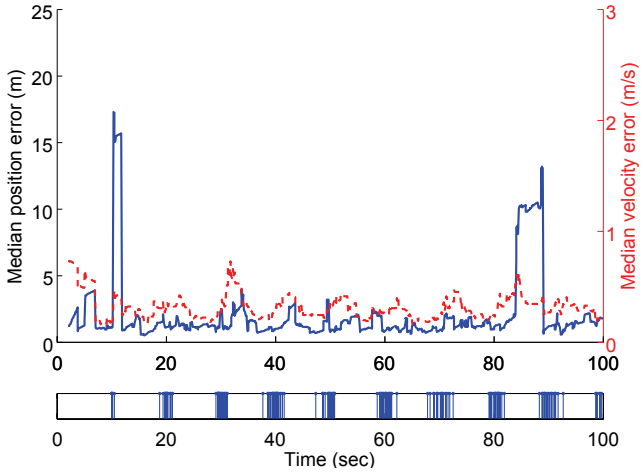
Figure 5-9: (a) True node positions at $t = 75$ seconds (dots), and ranging radius (circle). (b) True (black, solid line) and estimated (colored, dots) trajectories for four of 40 nodes.

		Adaptive updates			2 Hz updates		
		30 sec.	10 sec.	3 sec.	30 sec.	10 sec.	3 sec.
Position error (m)	Median	1.30	1.21	1.33	0.158	0.304	0.518
	Std. dev.	0.950	2.66	1.56	1.32	0.643	0.944
	Maximum	9.12	17.197	21.7	12.1	7.92	6.46
Speed error (m/s)	Median	0.136	0.137	0.174	0.111	0.120	0.173
	Std. dev.	0.124	0.139	0.179	0.107	0.085	0.129
	Maximum	0.771	1.05	1.36	0.645	0.732	1.57
Heading error (deg)	Median	10.0	8.94	13.9	5.69	8.59	13.5
	Std. dev.	17.2	12.3	10.6	37.0	8.29	11.3
	Maximum	103	76.0	84.6	161	81.6	64.9

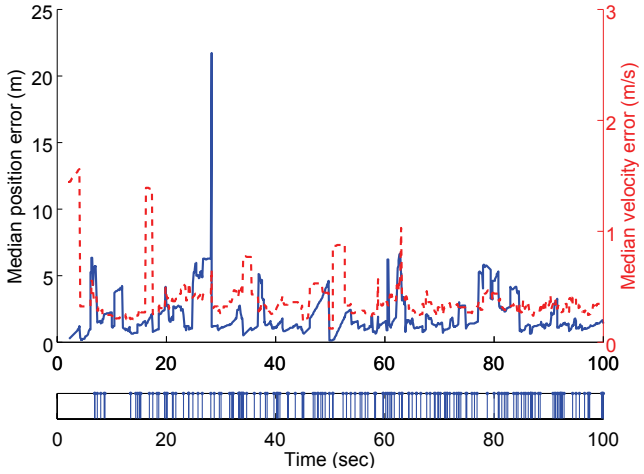
Table 5.1: Performance of MBL in two different update schemes.



(a)

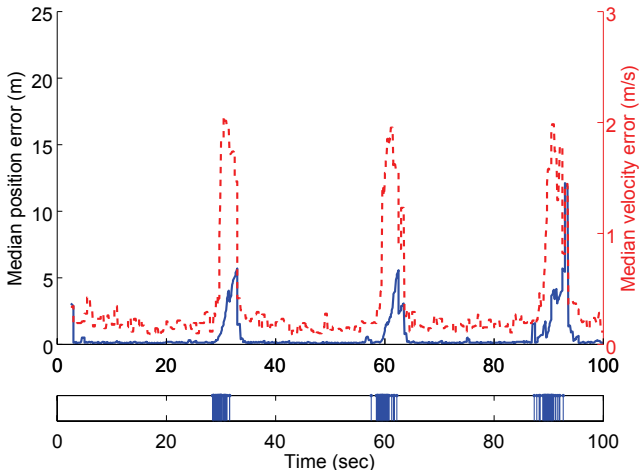


(b)

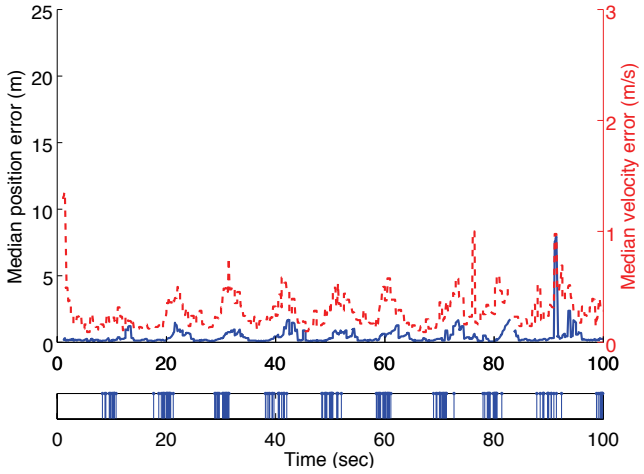


(c)

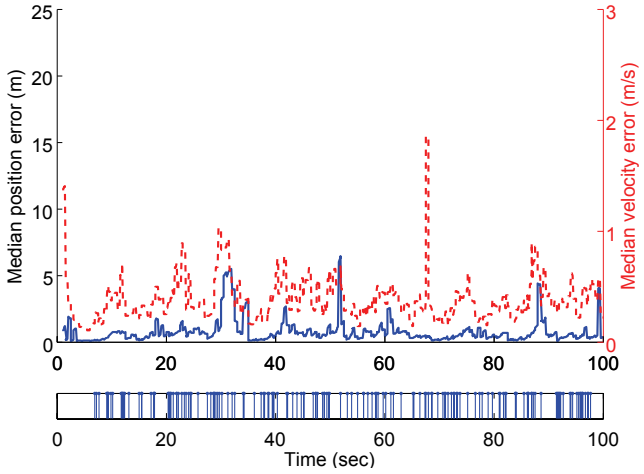
Figure 5-10: (a) 30-sec. smooth motion intervals. (b) 10-sec. smooth motion intervals. (c) 3-sec. smooth motion intervals.



(a)



(b)



(c)

Figure 5-11: (a) 30-sec. intervals, 2 Hz updates. (b) 10-sec. intervals, 2 Hz updates. (c) 3-sec. intervals, 2 Hz updates.

5.6 Discussion and Future Work

The proposed method in this thesis 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). We identified the effects of ranging frequency, node speed, and network density on the algorithm in Section 5.3.

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 still vulnerable to noise in the remaining data under the situations above, 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. For example, when range observations between two nodes remain constant (excluding outliers) for a significant duration of time, we can infer that the relative motion of one node is very small with respect to the other. In r vs. t plot, the constant relative motion is represented as a line instead of a hyperbola. Therefore, from the point of view of hyperbola fitting, this is a singular point, that is infeasible to recover. Instead, we can fix the trajectory of one node in the other's relative frame as a static point, by setting $s = 0$, $t = 0$, and $m = r$, where r is the observed constant range. This idea is not considered in this thesis to keep simplicity of the method, but is possibly a feasible idea to handle such situations.

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 [42]) 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.

Above all, the major challenge in mobile localization algorithms is the mobility itself. By comparing with a static localization algorithm, we confirmed that our moving-baseline approach is considerably more efficient than repeating static algorithms. However, the mobility still poses challenges to the localization problem. As shown in 5.3, varying node speed affects the performance in both beneficial and harmful way. Moreover, we are required to employ a change point detection scheme to detect topology change of the network, but there can be no single "one-for-all" set of parameters for the change point detection over a variety of speed ranges. One parameter set that fits to a certain speed range may result in many false positives or false negatives in change point detection under a different speed range.

Although any motion can be approximated as piecewise-linear motion to some degree, future algorithms must be able to handle more general motions. 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 [62]. 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.

Chapter 6

Conclusion

This thesis 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 0.2–2 m of their correct position (depending on the frequency of motion changes and recomputation) and within 0.3 m/s of their correct velocity. The method fails when ranging is too slow, the network is too sparse, or when node motions are too fast or too correlated.

We also demonstrated that node mobility is an important factor for the performance of localization algorithms. The varying topology of mobile networks was handled by proper use of adaptive update rules and change point detection algorithms. By comparing to a static localization scheme, we showed that estimating node trajectories made our proposed method far more efficient than estimating instantaneous

positions. Finally, as future work, we envision that future moving-basline algorithms will be able to handle more general motion and velocity profiles.

Appendix A

Analysis of Hyperbola Fitting

In this appendix, we study the behavior of hyperbola fitting, the basic ingredient of the MBL algorithm.

First, we compare the robust quadratic fitting method [10] with ordinary least squares method (Figure A-1). Due to the extreme outliers, least squares performed poorly, while robust quadratic fitting recovered an accurate hyperbola. Table A.1 compares the recovered motion hyperbola parameters to ground truth. In general, the robust quadratic method is superior to ordinary least squares if the underlying probability distribution is long-tailed such as the UWB ranging model developed in Chapter 4.

However, the robust quadratic fitting does not always find the correct motion hyperbola parameters. Figure A-2 exemplifies such a situation. Due to the large range error in the second sample, the fitted hyperbola is concave, which is physically impossible. In fact, any fitting method cannot find the correct parameters in that situation, because three samples uniquely define a quadratic equation. However, the

	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 A.1: Hyperbola estimation with least squares and robust quadratic fitting.

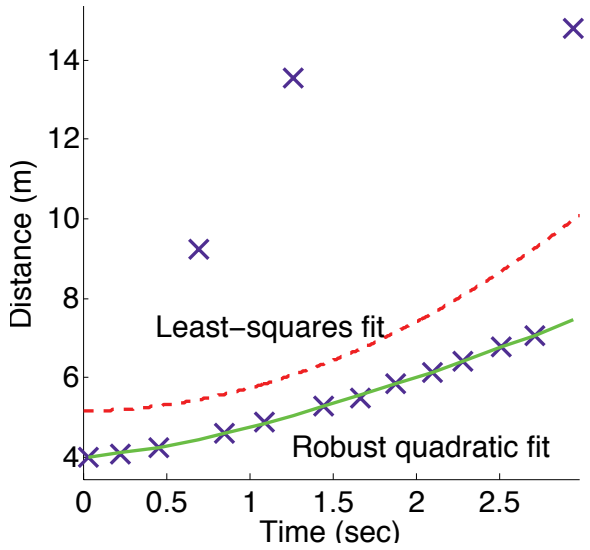


Figure A-1: Hyperbola fitting to noisy range data ($\times$ marks) with least squares (dashed) and robust quadratic fitting (solid).

fitting works well under certain situations. As a rule of thumb, it tends to recover the correct set of parameters if: 1) longer samples are available; 2) more samples are available; 3) samples are closer to the hyperbola vertex; 4) node speed is fast that range samples cover longer baseline within a fixed time interval. In other words, fitting becomes accurate if we can infer the entire shape of the hyperbola to be recovered better.

It is possible to formulate the hyperbola fitting in a Kalman filter form. We compare the performance of the robust hyperbola fitting with the Kalman filter form below. The quadratic equation we consider is:

$$r^2 = m^2 + (t - t^c)^2 s^2 = \gamma t^2 + \beta t + \alpha \tag{A.1}$$

Now, define the state vector $x = [\gamma \ \beta \ \alpha]^T$. Then the problem is an estimation of unknown deterministic state vector x. The corresponding state space model is:

$$x_{n+1} = x_n \tag{A.2}$$

$$z_n = r_n^2 = \begin{bmatrix} t_n^2 & t_n & 1 \end{bmatrix} x_n + v_n \tag{A.3}$$

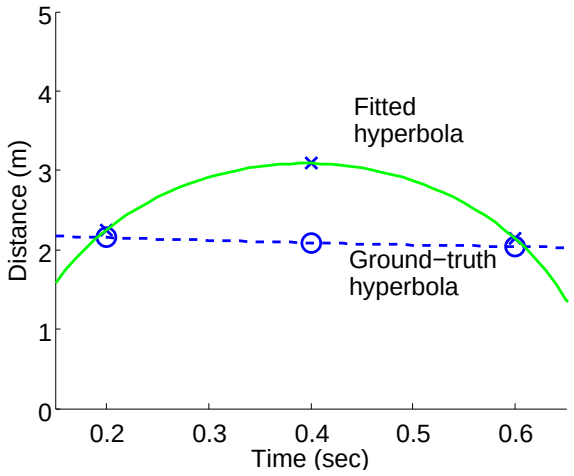


Figure A-2: Hyperbola fitting does not always find the true hyperbola.

where v is white measurement noise sequence with a priori variance Λ_{v_n}. Suppose $r_m = r + \epsilon$ where r_m is the measured range and r is the true range. For this analysis, assume that measurement error ϵ is zero-mean Gaussian with standard deviation of 0.03 m. Then the error v in terms of z is:

$$v = r_m^2 - r^2 = 2r_m\epsilon + \epsilon^2. \tag{A.4}$$

Here ϵ^2 follows a chi-squared distribution of degree 1 and its variance is $var(\epsilon^2) = \sigma^4 \cdot 2k = 1.62 \times 10^{-6}$. The expected value is also very small, making the ϵ^2 term is negligible.

Therefore, for simplicity, assume that error of y is $2r\epsilon$, which is a Gaussian. Its variance depends on r, so we take the expected value of variance.

$$\begin{aligned} E[var(2r\epsilon)] = E[E[var(2r\epsilon)|r]] &= \int_0^{r_{MAX}} 4r^2 var(\epsilon) p_r(r) dr \\ &= 4 \cdot 0.03^2 \cdot \frac{1}{r_{MAX}} \int_0^{r_{MAX}} r^2 dr \\ &= 1.08 \end{aligned} \tag{A.5}$$

with $p_r(r)$ being uniformly distributed and $r_{MAX} = 30$. We use this value (1.08) as

Λ_{v_n}.

We applied the standard Kalman filter on the formulation above, and compared its result with true motion hyperbola parameters as well as estimates from the robust quad fitting and least squares method. Specifically, we compared least squares with 3-second observation window, robust quadratic fit with 3- or 5-second observation window, and the Kalman filter. A result for estimate of t^c, the closest time of approach, is shown in Figure A-3. Although the Kalman filter converges to the true state, it takes 12 seconds for the convergence. On the other hand, the robust quadratic fit finds an estimate that is very close to the true state with only 3-second observation window. This is because the Kalman filter is not the optimal filter in general unless error distribution is Gaussian. We observed that our range error model itself is not Gaussian (containing outliers), and also the error term of our fitting problem is $r_m^2 - r^2$, not $r_m - r$.

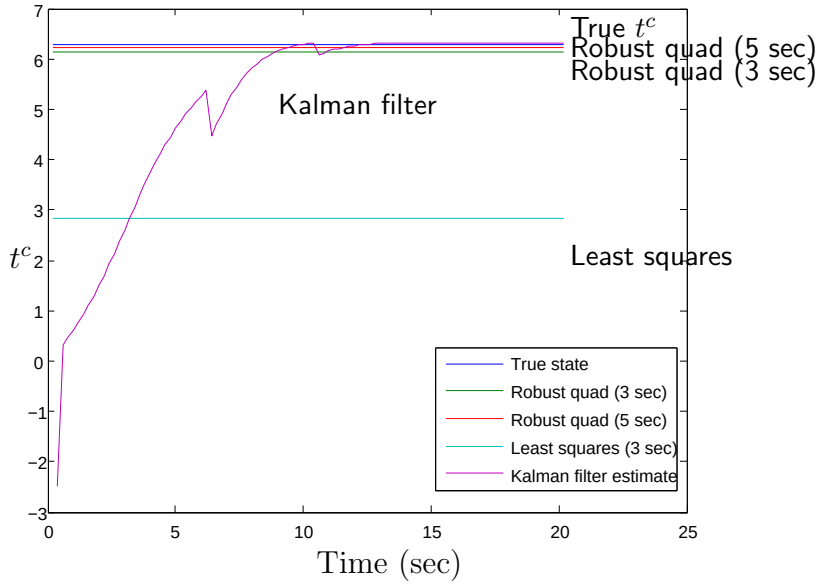


Figure A-3: Comparison of Kalman filter, robust quadratic fit, and least squares fit.

Bibliography

[1] Bardia Alavi and Kaveh Pahlavan. Modeling of The Distance Error for Indoor Geolocation. In *WCNC '03: Proceedings of the IEEE Wireless Communications and Networking Conference*, volume 1, pages 668–672, 2003.

[2] Joshua N. Ash and Lee C. Potter. Robust System Multiangulation using Subspace Methods. In *IPSN '07: Proceedings of the 6th International Conference on Information Processing in Sensor Networks*, pages 61–68, 2007.

[3] James Aspnes, Tolga Eren, David K. Goldenberg, A. Stephen Morse, Walter Whiteley, Yang Richard Yang, Brian D. O. Anderson, and Peter N. Belhumeur. A Theory of Network Localization. *IEEE Transactions on Mobile Computing*, 5(12):1663–1678, 2006.

[4] Aline Baggio and Koen Langendoen. Monte-Carlo Localization for Mobile Wireless Sensor Networks. In *MSN '06: Proceedings of the 2nd International Conference on Mobile Ad-hoc and Sensor Networks*, pages 317–328. Springer, 2006.

[5] Michéle Basseville and Igor V. Nikiforov. *Detection of abrupt changes: theory and application*. Prentice Hall, 1993.

[6] Pratik Biswas, Tzu-Chen Liang, Kim-Chuan Toh, Ta-Chung Wang, and Yinyu Ye. Semidefinite Programming Approaches for Sensor Network Localization With Noisy Distance Measurements. *IEEE Transactions on Automation Science and Engineering*, 3(4):360–371, 2006.

[7] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *MobiCom '98: Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 85–97, 1998.

[8] Nirupama Bulusu, John Heidemann, and Deborah Estrin. GPS-Less Low-Cost Outdoor Localization for Very Small Devices. *IEEE Personal Communications*, 7:28–34, 2000.

[9] Srdjan Capkun, Maher Hamdi, and Jean-Pierre Hubaux. GPS-free Positioning in Mobile Ad hoc Networks. *Cluster Computing*, 5(2):157–167, 2002.

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

[11] Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1-3):165–177, 1991.

[12] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Cliff Stein. *Introduction to Algorithms*. MIT Press, 2nd ed. edition, 2001.

[13] Jose A. Costa, Neal Patwari, and Alfred O. Hero. Distributed Weighted-Multidimensional Scaling for Node Localization in Sensor Networks. *ACM Transactions on Sensor Networks*, 2(1):39–64, 2006.

[14] Suprakash Datta, Chris Klinowski, Masoomeh Rudafshani, and Shaker Khaleque. Distributed Localization in Static and Mobile Sensor Networks. In *WiMob '06: IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, pages 69–76, 2006.

[15] Bram Dil, Stefan Dulman, and Paul Havinga. Range-Based Localization in Mobile Sensor Networks. In *EWSN '06: Proceedings of the Third European Workshop on Wireless Sensor Networks*, volume 3868, pages 164–179. Springer, 2006.

[16] W. J. Dixon. Simplified estimation from censored normal samples. *Annals of Mathematical Statistics*, 31:385–391, 1960.

[17] Lance Doherty, Kristofer S. J. Pister, and Laurent El Ghaoui. Convex Position Estimation in Wireless Sensor Networks. In *INFOCOM '01: Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1655–1663, 2001.

[18] T. Eren, D. K. 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 *INFOCOM '04: Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2673–2684, 2004.

[19] Dieter Fox, Wolfram Burgard, Hannes Kruppa, and Sebastian Thrun. A Probabilistic Approach to Collaborative Multi-Robot Localization. *Autonomous Robots*, 8(3):325–344, 2000.

[20] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Markov Localization for Mobile Robots in Dynamic Environments. *Journal of Artificial Intelligence Research*, 11(3):391–427, 1999.

[21] Lee Freitag, Mark Johnson, Matthew Grund, Sandipa Singh, and James Preisig. Integrated acoustic communication and navigation for multiple UUVs. In *OCEANS, 2001. MTS/IEEE Conference and Exhibition*, volume 4, pages 2065–2070, 2001.

[22] Aram Galstyan, Bhaskar Krishnamachari, Kristina Lerman, and Sundeep Pattem. Distributed Online Localization in Sensor Networks Using a Moving Target. In *IPSN '04: Proceedings of the Third International Symposium on Information Processing in Sensor Networks*, pages 61–70. ACM Press New York, NY, USA, 2004.

[23] Gianni Giorgetti, Sandeep K. S. Gupta, and Gianfranco Manes. Wireless Localization Using Self-Organizing Maps. In *IPSN '07: Proceedings of the 6th International Conference on Information Processing in Sensor Networks*, pages 293–302, 2007.

[24] David K. Goldenberg, Pascal Bihler, Ming Cao, Jia Fang, Brian D.O. Anderson, A. Stephen Morse, and Y. Richard Yang. Localization in Sparse Networks using Sweeps. In *MobiCom '06: Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*, pages 110–121, 2006.

[25] Tian He, Chengdu Huang, Brian M. Blum, John A. Stankovic, and Tarek Abdelzaher. Range-Free Localization Schemes for Large Scale Sensor Networks. In *MobiCom '03: Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*, pages 81–95, 2003.

[26] Jeffrey Hightower and Gaetano Borriello. Location Sensing Techniques. Technical Report UW-CSE-01-07-01, University of Washington, 2001.

[27] B. Hoffmann-Wellenhof, J. Collins, and H. Lichtenegger. *Global Positioning System: Theory and Practice*. Springer-Verlag, 1993.

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

[29] Lingxuan Hu and David Evans. Localization for Mobile Sensor Networks. In *MobiCom '04: Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*, pages 45–57. ACM Press New York, NY, USA, 2004.

[30] Alexander T. Ihler, John W. Fisher III, Randolph L. Moses, and Alan S. Willsky. Nonparametric Belief Propagation for Self-Calibration in Sensor Networks. In *IPSN '04: Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, pages 225 – 233, 2004.

[31] David B. Johnson and David A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. *Kluwer International Series in Engineering and Computer Science*, pages 153–179, 1996.

[32] Holger Karl and Andreas Willig. *Protocols and Architectures for Wireless Sensor Networks*. John Wiley & Sons, 2005.

[33] Brad Karp and H. T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *MobiCom '00: Proceedings of the 6th ACM/IEEE International Conference on Mobile Computing and Networking*, pages 243–254, 2000.

[34] Lasse Klingbeil and Tim Wark. A Wireless Sensor Network for Real-time Indoor Localisation and Motion Monitoring. In *IPSN '08: Proceedings of the 7th International Conference on Information Processing in Sensor Networks*, pages 39–50, 2008.

[35] Ryo Kurazume, Shigemi Nagata, and Shigeo Hirose. Cooperative positioning with multiple robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1250–1257, 1994.

[36] Erik G. Larsson. Cramer-Rao Bound Analysis of Distributed Positioning in Sensor Networks. *IEEE Signal Processing Magazine*, 11(3):334–337, 2004.

[37] Joon-Yong Lee and Robert A. Scholtz. Ranging in a Dense Multipath Environment Using an UWB Radio Link. *IEEE Journal on Selected Areas in Communications*, 20(9):1677–1683, 2002.

[38] John J. Leonard and Hugh F. Durrant-Whyte. Mobile Robot Localization by Tracking Geometric Beacons. *IEEE Tranactions on Robotics and Automation*, 7(3):376–382, 1991.

[39] Juan Liu, Ying Zhyang, and Feng Zhao. Robust Distributed Node Localization with Error Management. In *MobiHoc '06: Proceedings of the Seventh ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 250 – 261, 2006.

[40] Miklós Maróti, Branislav Kusy, Gyula Simon, and Ákos Lédeczi. The flooding time synchronization protocol. In *SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 39–49. ACM New York, NY, USA, 2004.

[41] Martin Mauve, Jorg Widmer, and Hannes Hartenstein. A Survey on Position-Based Routing in Mobile Ad Hoc Networks. *IEEE Network*, 15(6):30–39, 2001.

[42] David Moore, John Leonard, Daniela Rus, and Seth Teller. Robust Distributed Network Localization with Noisy Range Measurements. In *SenSys '04: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, pages 50–61, New York, NY, USA, 2004. ACM Press.

[43] Randolph L. Moses, Dushyanth Krishnamurthy, and Robert M. Patterson. A Self-Localization Method for Wireless Sensor Networks. *EURASIP Journal on Applied Singal Processing*, 4:348–358, 2003.

[44] Dragos Niculescu and Badri Nath. DV-Based Positioning in Ad Hoc Networks. *Telecommunication Systems*, 22:267–280, 2003.

[45] E. S. Page. Continuous Inspection Schemes. *Biometrika*, 41(1-2):100–115, 1954.

[46] Neal Patwari and Alfred O. Hero. Manifold Learning Algorithms for Localization in Wireless Sensor Networks. In *ICASSP '04: IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages 857–860, 2004.

[47] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, 1988.

[48] Nissanka B. Priyantha, Hari Balakrishnan, Erik D. Demaine, and Seth Teller. Anchor-Free Distributed Localization in Sensor Networks. In *SenSys '03: Proceedings of the First International Conference on Embedded Networked Sensor Systems*, pages 340–341, 2003.

[49] Nissanka B. Priyantha, Hari Balakrishnan, Erik D. Demaine, and Seth Teller. Mobile-Assisted Localization in Wireless Sensor Networks. In *INFOCOM '05: Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 1, pages 172–183. INSTITUTE OF ELECTRICAL ENGINEERS INC (IEEE), March 2005.

[50] Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan. The Cricket Location-Support System. In *MobiCom '00: Proceedings of the 6th ACM/IEEE International Conference on Mobile Computing and Networking*, pages 32–43. ACM Press New York, NY, USA, 2000.

[51] Stergios I. Roumeliotis and George A. Bekey. Collective Localization: a Distributed Kalman filter Approach to Localization of Groups of Mobile Robots. In *ICRA '00: Proceedings of the IEEE International Conference on Robotics and Automation*, volume 3, pages 2958–2965, 2000.

[52] Chris Savarese, Jan Rabaey, and Koen Langendoen. Robust Positioning Algorithms for Distributed Ad-Hoc Wireless Sensor Networks. In *USENIX '02: Proceedings of the USENIX Annual Technical Conference*, pages 317–328, 2002.

[53] Andreas Savvides, Wendy Garber, Sachin Adlakha, Randolph Moses, and Mani B. Srivastava. On the Error Characteristics of Multihop Node Localization in Ad-Hoc Sensor Networks. In *IPSN '03: Proceedings of the 2nd International Workshop on Information Processing in Sensor Networks*, pages 317–332. Springer, April 2003.

[54] Andreas Savvides, Chih-Chieh Han, and Mani B. Strivastava. Dynamic Fine-Grained Localization in Ad-Hoc Networks of Sensors. In *MobiCom '01: Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, pages 166–179. ACM Press New York, NY, USA, 2001.

[55] Yi Shang and Wheeler Ruml. Improved MDS-Based Localization. In *INFOCOM '04: Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2640–2651, 2004.

[56] Yi Shang, Wheeler Ruml, Ying Zhang, and Markus P.J. Fromherz. Localization from Mere Connectivity. In *MobiHoc '03: Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pages 201–212. ACM Press New York, NY, USA, 2003.

[57] Slobodan N. Simic and Shankar Sastry. Distributed Localization in Wireless Ad Hoc Networks. Technical report, UC Berkeley, 2001.

[58] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319–2323, 2000.

[59] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. Probabilistic Algorithms and the Interactive Museum Tour-Guide Robot Minerva. *The International Journal of Robotics Research*, 19(11):972, 2000.

[60] Jerome Vaganay, John J. Leonard, Joseph A. Curcio, and J. Scott Willcox. Experimental validation of the moving long base-line navigation concept. *Autonomous Underwater Vehicles, 2004 IEEE/OES*, pages 59–65, 2004.

[61] Abraham Wald. Sequential tests of statistical hypotheses. *Annals of Mathematical Statistics*, 16(2):117–186, 1945.

[62] Greg Welch, Gary Bishop, Leandra Vicci, Stephen Brumback, Kurtis Keller, and D'nardo Colucci. The HiBall Tracker: High-Performance Wide-Area Tracking for Virtual and Augmented Environments. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, pages 1–11, 1999.

[63] Louis Whitcomb, Dana Yoerger, and Hanumant Singh. Advances in Doppler-based navigation of underwater robotic vehicles. In *ICRA '99: Proceedings of the IEEE International Conference on Robotics and Automation*, volume 1, pages 399–406, 1999.

[64] Kamin Whitehouse and David Culler. A Robustness Analysis of Multi-hop Ranging-based Localization Approximations. In *IPSN '06: Proceedings of the 5th International Conference on Information Processing in Sensor Networks*, pages 317–325, 2006.

[65] Moe Z. Win and Robert A. Scholtz. On the Robustness of Ultra-Wide Bandwidth Singals in Dense Multipath Environments. *IEEE Communications Letters*, 2(2):51–53, 1998.

[66] Zheng Yang and Yunhao Liu. Quality of Trilateration: Confidence based Iterative Localization. In *Proceedings of the 28th International Conference on Distributed Computing Systems*, pages 446–453, 2008.

[67] Feng Zhao and Leonidas Guibas. *Wireless Sensor Networks: An Information Processing Approach*. Morgan Kaufmann, 2004.