SPATIO-TEMPORAL AWARENESS IN MOBILE WIRELESS SENSOR NETWORKS

By

Isaac Amundson

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

August, 2010

Nashville, Tennessee

Approved:

Professor Xenofon Koutsoukos

Professor Akos Ledeczi

Professor Julie Adams

Professor Yuan Xue

Professor Nilanjan Sarkar

PREFACE

Over the past decade we have witnessed the evolution of wireless sensor networks, with advancements in hardware design, communication protocols, resource efficiency, and other aspects. Recently, there has been much focus on *mobile* sensor networks, and we have even seen the development of small-profile sensing devices that are able to control their own movement. Although it has been shown that mobility alleviates several issues relating to sensor network coverage and connectivity, many challenges remain. Among these, the need for position estimation is perhaps the most important. Not only is localization required to understand sensor data in a spatial context, but also for navigation, a key feature of mobile sensors.

In this dissertation we develop techniques for the synchronization, localization and navigation of resource-constrained mobile sensors. Resource constraints prevent us from using traditional coordination methods, because these typically require bulky, expensive, and sophisticated sensors, substantial memory and processor allocation, and a generous power supply. The global positioning system is able to provide synchronization and localization information, however in many situations it cannot be relied on, and alternative methods are required. We focus on techniques that rely primarily on the sensor radio, since all wireless sensor nodes are equipped with them, thus requiring no additional hardware support. Specifically, we have developed a time synchronization protocol for heterogeneous sensor networks, localization techniques in which sensor nodes measure the phase and Doppler-shift in frequency of an RF interference signal, and waypoint navigation, in which the localization results are used to derive motion vectors for the mobile sensors.

ACKNOWLEDGMENTS

The work presented in this dissertation could not have been performed without the guidance, help, advice, and support from several people. First, I would like to thank my academic and research advisor, Professor Xenofon Koutsoukos, who took on a student with little background in computer science and somehow managed to get him to this point. Professor Koutsoukos has improved my theoretical computer science skills, taught me how to avoid unnecessary or redundant research directions, and has given me excellent feedback on my writing. I would also like to thank Professor Akos Ledeczi, who was involved one way or another in practically all of my work. Akos could always be counted on to give encouragement, helpful advice, and constructive criticism, all of which improved the quality of my experimental research and writing. And to the rest of my Ph.D. Committee, Professors Yuan Xue, Julie Adams, and Nilanjan Sarkar, thank you for providing me with feedback on my proposal and dissertation, as well as giving me advice and instruction during my time at Vanderbilt.

My research encompasses several areas of engineering, and oftentimes there were concepts that I had trouble understanding or that required clarification. When such problems came up, I could always rely on members of our research group to either give me a detailed explanation, or at the very least, point me in the right direction. Manish Kushwaha, who has been my research partner since the day I joined the sensornet group, was a great pleasure to work with. I could always rely on him to help me when I needed it, and together we developed several novel wireless sensor network applications. I would like to thank Branislav Kusy for bringing me up to speed with the intricacies of the time synchronization and localization protocols developed by our group before I became involved. Brano was largely responsible for two of the core methodologies I use in my work, Elapsed Time on Arrival and the Radio Interferometric Positioning System. Brano also gave me invaluable advice when it came time to write my dissertation proposal. I would also like to thank Peter Volgyesi, who was somehow able to answer any technical question that I asked him. No matter what type of operating system, programming language, or hardware device I was working with, Peter knew it inside and out and was always available to help me get it working. And finally there is Janos Sallai. Janos was the first person to give me a tutorial on mote programming before I even joined the sensornet group, and since then we have worked together on several projects. He always offered me an opinion or analysis or suggestion whenever I needed it, and no matter how often I stopped by his desk to ask a question, he would put down his work and give me his full attention. Discussions with Janos resulted in the development of TripLoc, the main contribution of this dissertation. It

was a great pleasure working with Janos, and I cannot thank him enough for his help these last few years. Finally, I would like to acknowledge the rest of our sensornet group for offering suggestions and comments in the lab and during our meetings. We have a terrific sensornet research group at ISIS, and this is only because it is comprised of enthusiastic, intelligent, and awesome people.

I could not have made it this far without having been mentored by some extraordinary scientists since my days as an undergraduate student at Antioch College. Dr. Leland Clark took me in when I was just beginning to take an interest in science, and his enthusiasm and fascination with natural processes (mostly biological) convinced me to give up a career in writing and pursue scientific study instead. Dr. Jeffrey Smith gave me an extraordinary opportunity by first hiring me on in his lab at the National Institutes of Health and then by luring me to Vanderbilt to automate his lab there. Finally, Dr. Kenneth Frampton encouraged me to pursue a graduate degree in engineering, and was the one who got me interested in wireless sensor networks in the first place.

During my time at Vanderbilt, both my children, Maya and Elliott, were born. These are the brightest and most enchanting people I will ever know. Even on the most grueling research days when nothing worked the way it was supposed to and I would come home in a miserable mood, I could always count on them to greet me with big smiles and hugs. Maya and Elliott are the most precious things in the world, and I am so thankful for having them in my life. I am also especially thankful for my wife, Suzanne, who is a very amazing (and patient) person. While in graduate school, my daily schedule, weekly schedule, conference schedule, graduation date, workload, and every other aspect of my life have been under constant flux. Somehow Suzanne managed to keep her sanity and help me maintain mine. Her support during this time has been tremendous. Suzanne means the world to me, and I do not think I can possibly convey with words how much I appreciate her and how fortunate I am to have her. Finally, I would like to thank my extended family and friends (i.e. my support group) for listening to me and always offering advice and encouragement.

The work presented in this dissertation was supported by a Vanderbilt University Discovery Grant, ARO MURI grant W911NF-06-1-0076, NSF CAREER award CNS-0347440, and NSF grant CNS-0721604. I would also like to thank Nashville Metropolitan Parks and Recreation and Edwin Warner Park for use of their space.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Wireless Sensor Networks

As the future of computing moves toward pervasive and ubiquitous platforms, we find ourselves in need of hardware, software, protocols and methodologies that promote their practical use. Wireless sensor networks (WSNs) are a perfect example of this developing technology. WSNs consist of large numbers of cooperating sensor nodes that can be deployed in practically any environment, and have already demonstrated their utility in a wide range of applications including environmental monitoring [1], transportation [2], and health care [3]. As WSNs and other embedded computing technologies continue to evolve, it is imperative that we stay abreast of the challenges that arise in order to enable a seamless transfer to general public utilization.

The greatest challenge we face when working with WSNs is their resource limitations [4], [5]. Memory, processor, communication range, power supply, and hardware quality have all been minimized in order to develop inexpensive, general-use sensor nodes with small form factors. The advantage of this approach is that hundreds of these devices can be deployed over a wide (possibly remote) area, at little cost, and handle tasks for which PC-class devices would not be well suited.

Due to the distributed nature of WSNs, we often encounter situations in which it is advantageous to network several different types of sensor platforms together. Such heterogeneous sensor networks (HSNs) are a promising direction for developing a diverse set of applications [6], [7], [8], [9]. The challenge with heterogeneity is that differences in hardware specification, OS design, and communication protocol add significant complexity to application development, and can hinder the overall performance of the sensor network deployment. Services are beginning to emerge that address these issues [10], [11], but because there can never be a single protocol that will function correctly for all possible present and future devices and operating systems, heterogeneity remains high on the list of sensor network challenges.

Mobile wireless sensor networks (MWSNs) are a particular class of WSN in which mobility plays a key role in the execution of the application. In recent years, mobility has become an important area of research for the WSN community [12], [13], [14]. Although WSN deployments were never envisioned to be fully static, mobility was initially regarded as having several challenges that needed to be overcome, including connectivity, coverage, and energy consumption, among others. However,

recent studies have been showing mobility in a more favorable light [15]. Rather than complicating these issues, it has been demonstrated that the introduction of mobile entities can resolve some of these problems [12]. In addition, mobility enables sensor nodes to target and track moving phenomena such as chemical clouds, vehicles, and packages [16].

Mobility in WSNs requires fine-grained coordination and navigation capability. These challenges must be addressed for the development of a versatile mobile sensor platform:

- **Localization.** One of the most significant challenges for MWSNs is the need for localization. In order to understand sensor data in a spatial context, or for proper navigation throughout a sensing region, sensor position must be known. Because sensor nodes may be deployed dynamically (i.e., dropped from an aircraft), or may change position during run-time (i.e., when attached to a shipping container), there may be no way of knowing the location of each node at any given time. For static WSNs, this is not as much of a problem because once node positions have been determined, they are unlikely to change. On the other hand, mobile sensors must frequently estimate their position, which takes time and energy, and consumes other resources needed by the sensing application. Furthermore, localization schemes that provide high-accuracy positioning information in WSNs cannot be employed by mobile sensors, because they typically require centralized processing, take too long to run, or make assumptions about the environment or network topology that do not apply to dynamic networks.

- **Navigation.** In MWSNs, position updates are often required for navigation through the sensing region. Robot navigation has been studied extensively, however, many of the published techniques are inappropriate for small-footprint mobile sensors because they are either too energy intensive, computationally expensive, or cost prohibitive.

- **Time Synchronization.** Not only do sensor networks require coordination in space, but coordination in time as well. Time synchronization is a critical component in many sensor network applications, including localization. Although several synchronization protocols have been developed, they tend to break down when implemented on networks of heterogeneous devices. This is because different synchronization protocols interact and behave differently when employed on devices with diverse hardware components, operating systems, and communication modalities. A methodology that provides accurate synchronization in one part of the network may perform undesirably in another part.

Contributions

The contributions described in this dissertation comprise methods for mobile sensor coordination including localization, navigation, and time synchronization.

Time Synchronization in Heterogeneous Sensor Networks. We have developed a protocol for achieving microsecond-accuracy synchronization in heterogeneous sensor networks [11], [17]. HSN synchronization becomes necessary when a diverse group of sensor nodes must coordinate with each other. We consider a multi-hop network consisting of Berkeley motes and mobile robots with embedded Linux PCs (the mote-PC connection is a dominant configuration for HSN applications). Time synchronization in both mote and PC networks have been studied independently, however, a sub-millisecond software synchronization method combining these two networks has not previously been developed.

Navigation based on RF Doppler Shifts. In addition to coordinating sensors in time, we have developed several techniques for spatial coordination in sensor networks. The first is a navigation method called *dNav* that measures RF Doppler shifts for control feedback [18], [19]. We show that a mobile sensor transmitting an RF sinusoidal signal is capable of navigating via waypoints based on the Doppler-shift in frequency observed at several stationary receiver nodes. This work demonstrates the feasibility of RF-based mobile sensor navigation, and highlights the challenges involved, as well as provides a direction for the evolution of RF-based navigation systems.

Determining Angular Separation in MWSNs. An extension to Doppler-based control has also been developed for determining angular separation between two landmarks [20]. Angular separation estimation only requires the sensor radio and wheel encoders, both of which are common to robotic wireless sensors, and hence no additional hardware is required. Our method uses the Doppler shift in radio frequency and the instantaneous velocity of a mobile node transmitting a sinusoidal signal to derive the angular separation between infrastructure nodes. The method does not require the positions of the infrastructure nodes, or the initial position of the mobile node, to be known. Knowledge of angular separation between pairs of beacons can be used for localization via triangulation, or for navigation without localization, using angle-based navigation techniques.

Rapid Distributed Localization from Radio Interferometric Measurements. We develop a localization method that utilizes the *phase* of the RF signal, rather than frequency [21]. The system, called TripLoc, determines angle-of-arrival by making radio interferometric measurements using the same technique as the Radio Interferometric Positioning System (RIPS) [22]. However, unlike RIPS, TripLoc is feasible for mobile sensor navigation. We bundle two radio-interferometric transmitter

nodes and a receiver node into a three-antenna array. The close proximity of the two transmitter nodes enables us to avoid the phase ambiguity problem that RIPS experiences. Another drawback with RIPS for navigation systems is that a single localization step can take several minutes, a latency that cannot sustain node mobility. Our method reduces this latency to approximately one second. In addition, RIPS requires PC processing for the localization algorithm, whereas the triangulation method we present can be implemented entirely on resource-constrained motes, in a distributed manner.

Using TripLoc for Mobile Sensor Navigation. We design TripNav, a waypoint navigation method for mobile wireless sensor networks, in which a mobile sensor traverses a series of waypoints using TripLoc to determine its current position. The development of TripNav is important, because it demonstrates that TripLoc is indeed suitable (sufficiently rapid and accurate) for mobile sensor navigation, one of the few RF-based distributed localization techniques that meet these criteria.

Dissertation Organization

In Chapter II, we provide a survey of mobile wireless sensor networks. The survey includes various coordination schemes relating to localization, navigation, and time synchronization. In Chapter III, we present a time synchronization protocol for heterogeneous sensor networks. Chapters IV and V describe localization and navigation techniques that measure the Doppler shift of an RF interference signal to determine spatial relationships. In Chapter VI, the phase of the interference signal is measured to determine either bearing or position, and in Chapter VII, we use this system for mobile sensor navigation. Finally, in Chapter VIII, we conclude.

CHAPTER II

BACKGROUND

One of the most significant challenges for WSNs is spatio-temporal awareness [23], [24]. Such network-wide coordination in both time and space is critical for the proper analysis of sensor data that is collected. Sensor nodes must have a common notion of time when detecting the occurrence of events, and without accurate location information, event detection will rarely be of any practical use. When the network only contains stationary nodes, localization is a one-time operation that can take place during network deployment. However, the localization problem becomes more complicated when mobile entities exist within the network. Instead of localizing only once at startup, mobile nodes require continuous localization for navigation throughout the sensing region.

At present, the most widely used method for spatio-temporal coordination is the NAVSTAR Global Positioning System (GPS) [25]. The system consists of approximately 24 satellites that orbit the planet, of which four are required to obtain location information (3 to determine 3D position, and 1 to resolve local clock uncertainty). The satellites continuously transmit messages that contain ephemeris data, transmission time, and vital statistics. Mobile receivers are then able to compute their location using lateration based on signal time of flight and orbital position data. Commercial-use GPS is accurate to within 10 meters (position) and 1 microsecond (time synchronization), is free to use anywhere on the planet and, for many mobile applications, is an ideal localization technology that should be taken advantage of. However, there are also several situations in which it will not work reliably. Because GPS requires line of sight to multiple satellites, mobile sensor networks that are deployed in urban environments, indoors, underground, or off-planet will not be able to use it. Furthermore, although GPS receivers are available for mote-scale devices, they are still relatively expensive, and therefore undesirable for many deployments. Therefore, this dissertation is presented from a GPS-*less* perspective (i.e., one that does not rely solely on GPS technology).

Mobile Wireless Sensor Networks

In this section, we provide a taxonomy of MWSNs, including the differences between MWSNs and WSNs, and the advantages of adding mobility.

MWSN Architectures

Mobile sensor networks can be categorized by flat, 2-tier, or 3-tier hierarchical architectures [12], as illustrated in Figure 1, and described below.

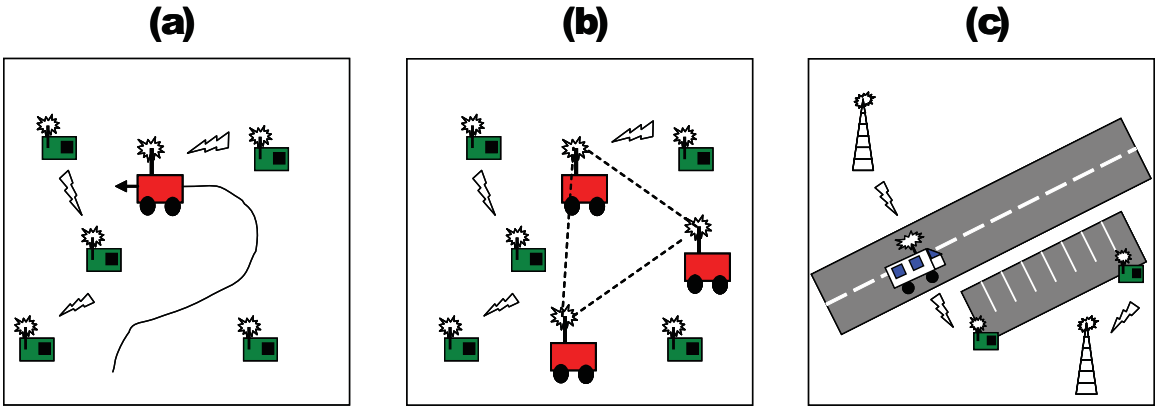


Figure 1: (a) Flat, (b) 2-Tier, and (c) 3-Tier MWSN architectures.

A flat, or planar, network architecture comprises a set of heterogeneous devices that communicate in an ad hoc manner. The devices can be mobile or stationary, but all communicate over the same network. Basic navigation systems such as [18] have a flat architecture, as pictured in Figure 1a.

The two-tier architecture consists of a set of stationary nodes, and a set of mobile nodes. The mobile nodes form an overlay network or act as data mules (see below) to help move data through the network. The overlay network can include mobile devices that have greater processing capability, longer communication range, and higher bandwidth. Furthermore, the overlay network density may be such that all nodes are always connected, or the network can become disjoint. When the latter is the case, mobile entities can position themselves in order to re-establish connectivity, ensuring network packets reach their intended destination. The NavMote system [26] takes this approach. The 2-tier architecture is pictured in Figure 1b.

In the three-tier architecture, a set of stationary sensor nodes pass data to a set of mobile devices, which then forward that data to a set of access points. This heterogeneous network is designed to cover wide areas and be compatible with several applications simultaneously. For example, consider a sensor network application that monitors a parking garage for parking space availability. The sensor network (first tier) broadcasts availability updates to compatible mobile devices (second tier), such as cell phones or PDAs, that are passing by. In turn, the cell phones forward this availability data to access points (third tier), such as cell towers, and the data are uploaded to a centralized database server. Users wishing to locate an available parking space can then access the database. The 3-tier architecture is pictured in Figure 1c.

At the node level, mobile wireless sensors can be categorized based on their role within the network:

Mobile Embedded Sensor. Mobile embedded nodes do not control their own movement; rather, their motion is directed by some external force, such as when tethered to an animal [27] or attached to a shipping container [28]. Typical embedded sensors include the MicaZ [29], XSM [30], Telos [31], and Mica2 [32] motes (Figure 2a).

Mobile Actuated Sensor. Sensor nodes can also have locomotion capability (for example, Robomote [33], Ragobot [34], Cotsbot [35], Millibot [36], and Micabot [37]), which enables them to move throughout a sensing region [18]. With this type of controlled mobility, the deployment specification can be more exact, coverage can be maximized, and specific phenomena can be targeted and followed. Figure 2b is a picture of such a mobile sensor.

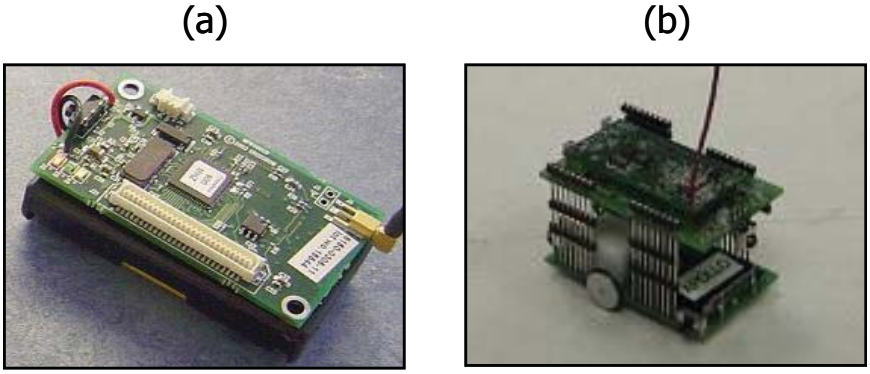


Figure 2: (a) The Mica2 sensor node. (b) The USC Robomote mobile actuated sensor.

Data Mule. Oftentimes, the sensors need not be mobile, but they may require a mobile device to collect their data and deliver it to a base station. These types of mobile entities are referred to as *data mules* [38]. It is generally assumed that data mules can recharge their power source automatically.

Access Point. In sparse networks, or when a node drops off the network, mobile nodes can position themselves to maintain network connectivity [38], [39]. In this case, they behave as network access points.

Advantages of Adding Mobility

Sensor network deployments are often determined by the application. Nodes can be placed in a grid, randomly, surrounding an object of interest, or in countless other arrangements. In many situations, an optimal deployment is unknown until the sensor nodes start collecting and processing

data. For deployments in remote or wide areas, rearranging node positions is generally infeasible. However, when nodes are mobile, redeployment is possible. In fact, it has been shown that the integration of mobile entities into WSNs improves coverage, and hence, utility of the sensor network deployment [39], [40]. Mobility enables more versatile sensing applications as well [15]. For example, Figure 3 illustrates a mobile sensor network that monitors wildfires. The mobile sensors are able to maintain a safe distance from the fire perimeter, as well as provide updates to fire fighters that indicate where that perimeter is currently located.

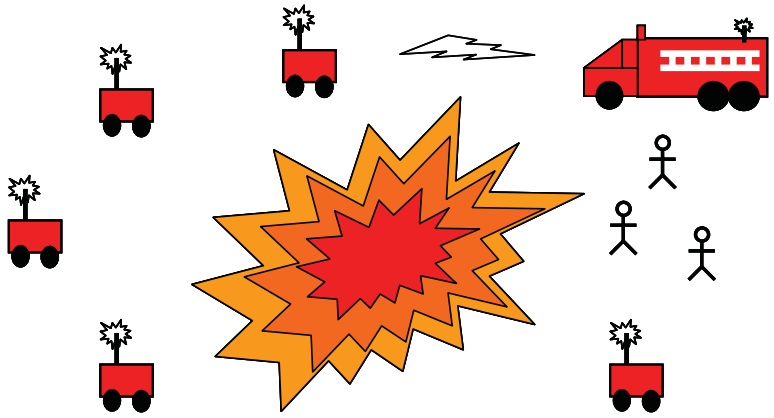


Figure 3: A MWSN that monitors wildfires. As the fire spreads, the mobile sensors can track it, as well as stay out of its way.

In networks that are sparse or disjoint, or when stationary nodes die, mobile nodes can maneuver to connect the lost or weak communication pathways. This is not possible with static WSNs, in which the data from dead or disconnected nodes would simply be lost. Similarly, when network sinks are stationary, nodes closer to the base station will die sooner, because they must forward more data messages than those nodes further away. By using mobile base stations, this problem is eliminated, and the lifetime of the network is extended [41].

Mobility also enables greater channel capacity and maintains data integrity by creating multiple communication pathways and reducing the number of hops messages must travel before reaching their destination [42].

Differences Between WSNs and MWSNs

In order to focus on the mobility aspect of wireless sensor networks, it is important to first understand how the common assumptions regarding statically-deployed WSNs change when mobile entities are introduced.

Localization. In statically deployed networks, node position can be determined once during initialization. However, those nodes that are mobile must continuously obtain their position as they traverse the sensing region. Localization requires additional time and energy, as well as the availability of a rapid localization service.

Dynamic Network Topology. Traditional WSN routing protocols [43], which describe how to pass messages through the network so they will most likely reach their destination, typically rely on routing tables or recent route histories. In dynamic topologies, table data become outdated quickly, and route discovery must repeatedly be performed at a substantial cost in terms of power, time, and bandwidth. Fortunately, there is an active area of research dedicated to routing in mobile ad hoc networks (MANETs), and MWSNs can borrow from this work [44].

Power Consumption. Power consumption models [45] differ greatly between WSNs and MWSNs. For both types of networks, wireless communication incurs a significant energy cost and must be used efficiently. However, mobile entities require additional power for mobility, and are often equipped with a much larger energy reserve, or have self-charging capability that enables them to plug into the power grid to recharge their batteries.

Network Sink. In centralized WSN applications, sensor data is forwarded to a base station, where it can be processed using resource-intensive methods. Data routing and aggregation can incur significant overhead. Some MWSNs use mobile base stations [41], which traverse the sensing region to collect data, or position themselves so that the number of transmission hops is minimized for the sensor nodes.

Localization in MWSNs

In this section, we provide a taxonomy of localization methods for MWSNs, as well as survey selected works representative of common MWSN localization. An extensive library of WSN localization research has been published within the past decade [46], [47], and many of these techniques can be applied to MWSNs. The localization techniques use diverse hardware, algorithms and signal modalities, which can be categorized along several different dimensions. We start by describing the three phases typically used in localization [48], [49], [50]: (1) *coordination*, (2) *measurement*, and (3) *position estimation*. We then focus on other aspects of MWSN-based localization, such as the effects of mobility, centralized versus distributed processing, and the environment.

MWSN localization is typically performed as illustrated in Figure 4. A group of nodes coordinate to initiate localization and synchronize their clocks, if necessary. One or more nodes then emit a

signal, and some property of the signal (e.g., arrival time, phase, signal strength, etc.) is observed by one or more receivers. Node position is then determined by transforming signal measurements into position estimates by means of a localization algorithm. In order to determine position, it is often necessary to enlist the help of cooperating sensor nodes that have been deployed into the environment at known positions a priori. These devices are referred to as *anchor*, *infrastructure*, or *seed* nodes. For example, in GPS, the infrastructure nodes are the satellites that orbit the planet. The position estimate may be relative to a set of stationary anchor nodes at known positions in a local coordinate system, or absolute coordinates may be obtained if the positions of the anchor nodes are known with respect to some global coordinate system (i.e., using GPS).

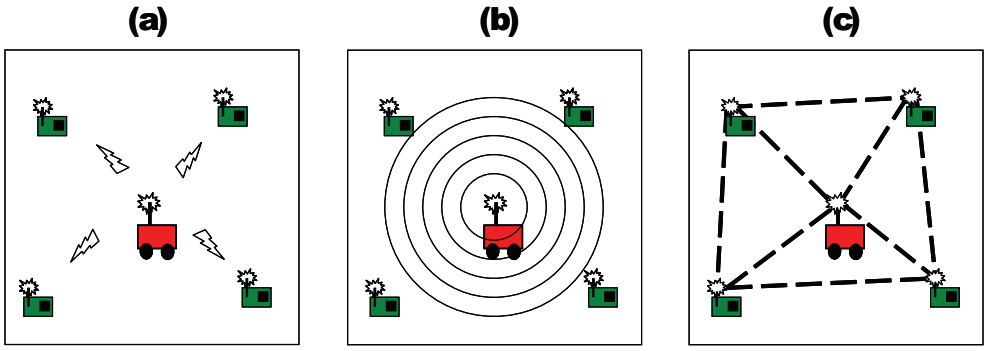


Figure 4: Localization phases: (a) coordination, (b) measurement, and (c) position estimation.

Coordination Phase

Prior to signal transmission, nodes participating in the localization typically coordinate with one another. Such coordination includes notification that the localization process is about to begin, as well as clock synchronization, which enables received signal data to be analyzed within a common timeframe.

Coordination using synchronization techniques such as RBS [51] and ETA [52] (see Time Synchronization, below) can encapsulate both notification and clock synchronization into a single message. These methods have microsecond accuracy and require transmission of only a single message. For example, the SyncEvent, one of the ETA primitives, declares a time in the future to begin the localization process. Encoded in the message is the timestamp of the message sender (typically the localization coordinator), which is inserted into the message immediately before transmission, thus reducing the amount of non-deterministic latency involved in the synchronization. All nodes within broadcast range will receive the message at approximately the same time instant, and assuming a negligible transit time of the radio signal through air, will be able to transform the sender

timestamp into their local timescale. This technique is used in several localization schemes, including [22], [53], [28], and [18].

Measurement Phase

The measurement phase typically involves the transmission of a signal by at least one node, followed by signal processing on the other participating nodes.

Signal Modalities. The choice of signal modality used by sensor nodes is important for accurate localization, and depends on node hardware, the environment, and the application. Because WSNs are developed to provide inexpensive wide-area observation capability, it is generally undesirable to add additional hardware to the sensor board, because this increases cost, weight, and resource utilization. Localization techniques will also perform differently in different environments. In humid environments, for example, radio signals perform worse than acoustic signals because moisture in the air absorbs and reflects the high frequency radio waves but does little to affect the vibrational sound waves. Finally, the application itself places some constraints on signal modality. A military application, for example, in which nodes must localize under stealth conditions, would be much better off using a silent modality such as radio frequency, rather than an audible one such as acoustic.

The acoustic modality typically employs either ultrasound or audible wave propagation. Several techniques have been published for each. Two early and commonly cited ultrasound localization techniques are Active Bats [54] and Cricket [55]. A more recent ultrasound approach can be found in [56], which also includes a survey on ultrasonic positioning systems and challenges. In the audible acoustic band, several novel localization systems have been developed, including beamforming [57], a sniper detection system [58], and generalized sound source localization [59].

Infrared (IR) signal attenuation is relatively high, requiring close proximity between transmitter and receiver. This is acceptable for most indoor localization techniques, however, outdoor localization becomes difficult, not only due to proximity issues, but also because the IR signal is difficult to read in the presence of sunlight. One of the earliest mobile localization systems is the Active Badge system [60], whereby a small electronic device (badge), carried by a user, emits a periodic IR identification signal. The signal is received by infrastructure nodes and centrally processed, allowing position information to be accessed by authorized users. Other IR localization methods can be found in [61] and [62].

Because all wireless sensor nodes have onboard radio hardware, radio frequency (RF) propagation has become a popular signal modality for localization. Signal properties such as strength, phase, or

frequency are analyzed to derive range or bearing data for position estimation. One benefit of using RF is that it has been shown to achieve localization accuracy on the order of centimeters, even in sparse networks [63]. On the other hand, because typical sensor node radios transmit at frequencies between 400 MHz to 2.4 GHz, sampling the raw signal for phase or frequency cannot be done with resource-constrained hardware. Instead, methods such as radio interferometry [22] must be used to generate a low frequency beat signal, as shown in Figure 5. The frequency and phase of the beat signal can then be measured by continuously observing the received signal strength indicator on the radio chip.

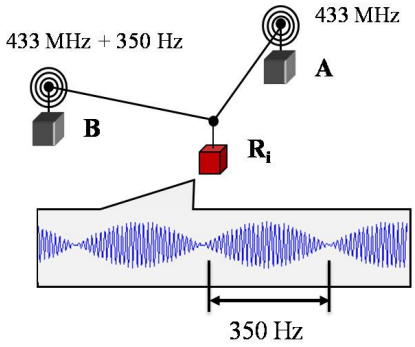


Figure 5: Radio interferometry. Two nodes transmit a sinusoidal signal at slightly different frequencies, which interfere to create a low-frequency beat signal that can be measured using resource-constrained sensor nodes.

The Lighthouse [64] and Spotlight [65] localization techniques use a light beacon to determine node position. Although both methods claim high accuracy, they require line of sight, a powerful light source that will perform well in lighted areas, and customized hardware for the light source.

Measurement Techniques. Several techniques exist for obtaining bearing, range, or proximity information based on signal measurement.

The angle-of-arrival (AOA) method [66], [67], [68], [57] involves determining the angular separation between two beacons, or a single beacon and a fixed axis. By determining the AOA at a certain number of sensor nodes, position can be determined by angulation methods, as outlined in the next section.

Localization by time-of-arrival (TOA) [69], [70], [71] measures the time a signal takes to arrive at some number of sensors. This requires knowing the time the signal was transmitted, and assumes tight time synchronization between sender and receiver. The signal will have known propagation properties, such as speed through air at sea level. The main drawback of this approach is that it is difficult to precisely record the arrival time of radio signals, since they travel close to the speed of

light. Therefore, it works best with an acoustic source. In addition, after transmitting the signal, the source must also make its transmit time known, incurring additional communication overhead. This can be avoided by employing a round-trip TOA method [72], whereby Node A transmits a signal to Node B. Upon signal reception, Node B transmits a signal back, and Node A observes the round-trip time, accounting for deterministic delay during the communication process.

Time-difference-of-arrival (TDOA) localization [53], [59] improves upon the TOA approach by eliminating the need to know when the signal was transmitted. Several time-synchronized nodes receive a signal, and look at the difference in arrival times (or difference in signal phase) at a specific time instant. Because the signal travels at a constant speed, the source position can easily be determined if there are a sufficient number of participating nodes.

Another localization method examines the received signal strength (RSS) of a message broadcast from a known location [73], [74]. Since the free-space signal strength model is governed by the inverse-square law, accurate localization is possible. Furthermore, measuring RSS typically does not involve any hardware modifications because most chips (e.g., RF, IR, etc.) provide software access to the amplitude of the received signal. Another use for RSS is *profiling* [75], [76], in which a map of RSS values is constructed during an initial training phase. Sensors then estimate their position by matching observed RSS values with the training data.

Recently, there have been several published techniques that determine the position of a node based on the observed frequency of arrival (FOA) of a signal [28], [18], [77], [78]. Signal frequency will undergo Doppler-shift when the transmitter and receiver are moving relative to one another. The observed Doppler-shift at multiple infrastructure nodes can be used to derive the position and velocity of the mobile node.

The above techniques provide the most accurate position estimates; however, it is oftentimes sufficient to only localize to a region. Such a region might be a room in a house, a floor in an office building, or a city block. This type of localization can be proximity-based, such as a node is located in Region A if an anchor in Region A is able to detect it. Another technique is to localize using hop count [79]. Because the approximate transmission range of the node radio is known, observing the number of message hops to a set of anchor nodes will constrain the target node to a specific region.

Localization Phase

The signal data obtained during the measurement phase can be used to determine the approximate position of the target node. Common localization techniques for MWSNs are based on ranging,

whereby distance or angle approximations are obtained. Because range data are often corrupted by noisy signal measurements, optimization methods are employed to filter the noise and arrive at a more accurate position estimate.

Lateration. When ranges between landmarks and the mobile node can be determined, lateration is used to estimate position [80], [81], [82]. Figure 6a illustrates the method. For two-dimensional localization, three range measurements from known positions are required. Each range can be represented as the radius of a circle, with the anchor node situated at the center. Without measurement noise, the three circles would intersect at exactly one point, the location of the target node. However, in the presence of noise, the three circles will overlap, and the target node will likely (but not necessarily) be contained within that region.

Angulation. When anchor bearings or angular separation between anchors and the mobile node can be obtained, angulation can be used to determine the position of the mobile node [83], [84], [67], [85]. This is pictured in Figure 6b. For *tri*-angulation, when two anchors are used, the target position will be identified as the third point in a triangle of two known angles (the bearings from each anchor), and the length of one side (the distance between anchor nodes). Often more than two anchor bearings are used, and target position is determined by the intersection of all bearings, as illustrated in the figure. In the presence of measurement noise, the bearings will not all intersect at the exact same point, but will instead define a region where the target node is likely to be.

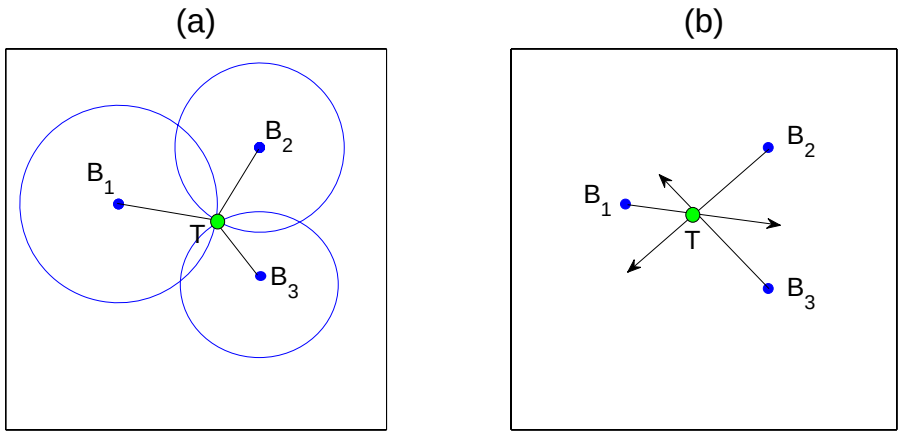


Figure 6: The position of a target node (T) is estimated based on the known positions of beacons (B_i) using (a) lateration or (b) angulation.

Cellular Proximity. An alternative approach is the range-free method [60], [86], [79], whereby a node is localized to the region in which it is detected. This method generally provides a more course-grained position estimate, and depends on the density of infrastructure nodes.

Dead Reckoning. A widely used localization technique for mobile robots is dead reckoning [16], [87], [88], [26]. Robots obtain their current velocity from wheel encoders or other means, and use this information in conjunction with the amount of time that has elapsed since the last update to derive current position and heading. The major drawback of this approach is that the position estimation accrues error over time, primarily because of noisy encoder data due to uneven surfaces, wheel slippage, dust, and other factors.

Estimation Methods. When measurement data is noisy, or the system is underdefined, state estimation methods can be used. There exist a number of estimation methods, but the two main approaches are: (1) maximum likelihood estimation (MLE) [89], which estimates the values of the state based on measured data only, and no prior information about the state is used, and (2) sequential Bayesian estimation (SBE) [90], which estimates state values based on measurements, as well as prior information.

MLE methods such as [91] and [92] find the estimates for the system state by maximizing the likelihood of the measured data. In other words, MLE picks the values of the system parameters that make the observed data "more likely" than any other values for the parameters. The data likelihood is computed using a measurement model that relates the measured data to the system state.

In SBE, the system state is iteratively estimated using the recursive Bayes rule, which states that the posterior is proportional to the product of the data likelihood and the predicted prior. Such methods are used in [76], [74], and [93]. Like MLE, the data likelihood is computed using a measurement model. The solution to SBE is generally intractable and cannot be determined analytically. Optimal solutions do exist in a restrictive set of cases, such as the Kalman Filter (KF) [94] and grid-based filters. More general suboptimal solutions exist, such as the Extended Kalman Filter (EKF) [95] and Particle Filter (PF) that approximate the optimal Bayesian estimation. The sequential Monte Carlo (SMC) [87] method is a PF that provides a suboptimal solution by approximating the posterior density by a set of random samples (also called particles) with associated weights. As the number of particles becomes very large, the particle filter approaches an optimal solution.

Table 1 summarizes common position estimation algorithms.

Algorithm	Signal modality	Measurement technique	Localization technique	Accuracy
Active Badge [60]	Infrared	Proximity	N/A	Room regions
Active Bat [54]	Ultrasound	TOA	Lateration	Centimeters
Cricket [55]	Ultrasound	Proximity	Lateration	Meter regions
RADAR [75]	Radio frequency	RSS	Angulation	Meters
RIPS [22]	Radio frequency	Phase	Genetic algorithm	Centimeters
dTrack [28]	Radio frequency	FOA	EKF	Meters
GPS [25]	Radio frequency	TDOA	Lateration	Meters

Table 1: Comparison of common position estimation algorithms.

The Radio Interferometric Positioning System

The radio interferometric positioning system (RIPS) [22] is an RF-based localization method upon which our work is based. We therefore provide a detailed overview of RIPS here.

The Radio Interferometric Positioning System was developed as a means for accurately determining relative position of a set of sensor nodes over a wide area by only using the onboard radio hardware. The randomness in the initial phases of the local oscillators on the transmitter and receiver nodes prohibits using the phase of the RF carrier directly for distance measurements. Instead, RIPS employs transmitter pairs at close frequencies for generating an interference signal. The envelope phase of the interference signal is independent of the phase of the receivers. The ambiguity in the phase at the transmitter side is eliminated by using a pair of receivers for measuring the phase *offset* between the receiver pairs.

RIPS was originally implemented on the COTS Mica2 mote platform [32]. These resource-constrained devices have a 7.4 MHz processor, 128 kB program memory (ROM), 4 kB RAM, ADC with 9 kSPS sampling rate, and a CC1000 tunable radio transceiver [96] that operates in the 433 MHz ISM band. Although the radio hardware is quite versatile for its size and cost, 433 MHz is too high to analyze the received signal directly. However, the phase and frequency of the envelope signal can be measured by making successive reads of the received signal strength indicator (RSSI). The phase of this low-frequency signal can be estimated with adequate accuracy, less than 5% error, using limited processing power and currently available time-synchronization methods ($< 5\mu s$) [97].

Figure 7 illustrates the approach. Two nodes, A and B, transmit pure sinusoids at respective frequencies f_A and f_B, such that $f_B < f_A$. The two signals interfere, resulting in a beat signal with frequency $|f_A - f_B|$. The phase offset between receiver pairs is measured, which is a linear combination of the distances between the four participating nodes:

$$\Delta\varphi \quad = \quad \frac{2\pi}{\lambda}(d_{AD} - d_{BD} + d_{BC} - d_{AC}) \; (mod \; 2\pi) \tag{1}$$

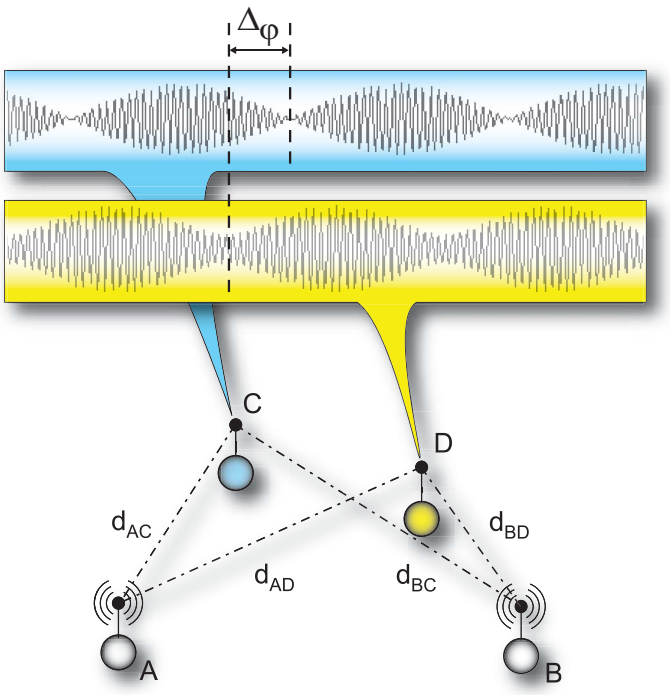


Figure 7: The radio interferometric positioning system.

Due to bandwidth limitations, the raw RSSI signal cannot be transmitted to the base station, and therefore must be processed on the mote. However, signal analysis on the resource-constrained mote hardware is non-trivial, involving both online and post-processing steps. During the online processing step, the RSSI signal is placed into a 256-byte buffer, one sample at a time upon each A/D converter interrupt. The clock speed and sampling rate limit the processing to approximately 820 CPU cycles per sample. Upon each interrupt the raw sample is incorporated into a moving average in order to filter out signal noise. Amplitude is determined by collecting the minimum and maximum peak values from the first 10% of the buffered samples, enough to cover at least one period. The amplitude is used as a signal quality indicator, as well as for determining a threshold for peak detection. The peaks are indexed, and are used to determine the frequency and phase of the signal in the post-processing step.

Post-processing is run after the buffer is full. The indexed peaks are used to determine the signal period, and outlying peaks are discarded. Signal frequency is determined by taking the reciprocal of the average period length. Similarly, signal phase is estimated by averaging the phases of the filtered peaks. Upon completion of the post processing step, the computed signal amplitude, frequency and phase are sent to the base station where the localization algorithm is run. The post-processing frequency and phase estimation of the RIPS algorithm takes less than 10,000 cycles per measurement and has an associated phase measurement error of 0.09 radian (approximately 5 degrees).

The distance measurement ($d_{AD} - d_{BD} + d_{BC} - d_{AC}$) is referred to as a *quad-range*. Because phase wraps to 0 at 2π, an ambiguity exists where an observed signal phase difference could correspond to several different quad-ranges. To resolve this, RIPS samples at multiple frequencies separated by 5 MHz, and searches for a unique quad-range that satisfies Equation (1) for each measured phase difference and corresponding wavelength.

A single quad-range is not sufficient to determine the positions of the four nodes involved in the radio interferometric measurement. Instead, a genetic optimization algorithm is used that takes into consideration all participating nodes in the sensing region. The algorithm is able to simultaneously remove bad measurements while accurately estimating the position of the sensors. The quad-ranges between a sufficient number of participating nodes constrain each node to a unique position in the sensing region. RIPS was shown to have an accuracy of 3 cm and a range of up to 160 meters. RIPS is likely the most accurate localization technique that can be performed on wireless sensor networks without additional sensor hardware support.

The Effect of Mobility on Localization

Typically, localization of mobile sensors is performed in order to track them, or for navigational purposes. However, when sensors are mobile, we encounter additional challenges and must develop methods to address them.

One of these challenges is localization latency. If the time to perform the localization takes too long, the sensor will have significantly changed its position since the measurement took place. For example, robot navigation requires periodic position estimates in order to derive the proper control outputs for wheel angular velocity. If the robot is traveling at 1 m/s and the localization algorithm takes 5 seconds to complete from the time the ranging measurements were taken, the robot might be 5 meters off from its estimated position.

Mobility may also impact the localization signal itself. For example, the frequency of the signal may undergo a Doppler shift, introducing error into the measurement. Doppler shifts occur when the transmitter of a signal is moving relative to the receiver, as illustrated in Figure 8. The resulting shift in frequency is related to the positions and relative speed of the two nodes. mTrack [53] takes this Doppler effect into account and uses it to refine its position estimate. Other approaches use the Doppler effect to directly solve for position and velocity [28], [18].

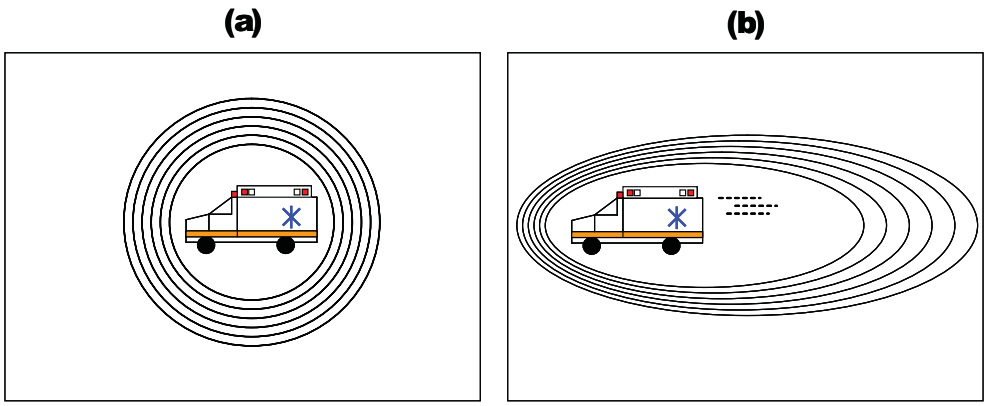


Figure 8: (a) The frequency of a signal does not change when the transmitter and receiver are moving at the same velocity. (b) However, when the transmitter and receiver are moving relative to one another, the signal will undergo a Doppler shift.

If the localization technique requires line of sight (LOS), there is the possibility that the mobile sensor will move from a position with good LOS, to a position with poor LOS. When this is the case, a dense network of nodes is required to ensure there is always LOS to the mobile node, wherever it may move.

Centralized vs. Distributed Algorithms

The resource constraints inherent in WSNs pose a challenge when it comes to executing certain localization algorithms, because they require extensive memory and processor bandwidth, especially when dealing with a large number of sensors, or when using complex statistical methods to estimate range or position [47]. A centralized localization algorithm runs on a base station, and all participating nodes must forward their measurement data to the base station. The advantage of the centralized approach is an algorithm can be designed that has more accuracy, precision, and can process greater amounts of data. On the other hand, base station processing suffers from the common pitfalls of centralization, such as poor scalability, single point of failure, data routing complexity, and greater power consumption (especially for nodes closer to the base station).

When nodes are mobile, the decision to use centralized or distributed processing becomes even more important. Mobility requires continuous and rapid localization. Although centralized localization techniques exist for mobile sensors [53], [28], they are usually not fast enough for certain applications, such as navigation. For example, mTrack [53] reports a latency of approximately 5 seconds. dNav [18], on the other hand, is distributed, and takes less than 1 second on average to return position and velocity estimates.

The Impact of Environment on Localization

The environment plays a significant role in the effectiveness of a localization method. As a result, there is no one localization method that will be accurate for all situations. Different environmental factors are listed below, as well as the effect they have on the aforementioned localization methods.

Ambient temperature, pressure, and humidity can affect localization accuracy, because these environmental factors directly impact the crystal oscillator in the transceiver. Furthermore, it has been well established that radio wave propagation is affected by precipitation, including moisture in the air, therefore localization techniques that use RF measurements can be impaired under these conditions [98]. One of the biggest problems with GPS is that it does not work reliably under water, indoors, or even when it is cloudy. This is because the GPS receiver requires line of sight to up to four satellites orbiting the planet [25].

At present there is a major effort underway to develop accurate localization methods in indoor environments. Indoor applications that require node position estimation are challenging because most propagation methods and measurement techniques suffer from multipath effects [99], where obstacles (e.g., walls, furniture, people, etc.) cause signal reflections that interfere with each other. In addition, many of the existing localization techniques that provide good accuracy outdoors, will not work indoors.

MWSN Applications with Localization Requirements

Although MWSNs are still in their infancy, several types of applications have already been developed in which localization plays an integral part. The applications fall under four main categories, (a) commercial, (b) environmental, (c) civil, and (d) military, however, most span more than just one of these domains.

Commercial. As MWSNs grow in popularity, we expect to see a burst of applications in the commercial sector that require some kind of positioning data.

- **Service Industry.** One such area is the service industry. Companies such as Skilligent [100] are developing software protocols for service robots that perform tasks such as basic patient care in nursing homes, maintenance and security in office buildings, and food and concierge service in restaurants and hotels. All of these applications require a mechanism for position estimation. Skilligent uses a visual localization system based on pattern matching. Objects

are used as landmarks, and are loaded into the system a priori, or dynamically at runtime. The robot learns its position by matching video images with landmark information.

- **Housekeeping.** The iRobot Roomba [101] is an automated vacuum cleaning robot for domestic use. The Roomba creates a map of the room as it moves by using feedback from a variety of bumper and optical sensors. Wheel encoders provide run-time position information that enable it to cover the entire room. The Roomba also uses a self-docking station to automatically recharge its batteries.

Environmental. MWSNs have become a valuable asset for environmental monitoring, thanks in part to their ability to be deployed in remote areas and for their ability to gather data over wide areas of interest.

- **Wildlife Tracking.** ZebraNet [27] is an early MWSN, in which mote-scale wireless devices were fitted to zebras for the purpose of tracking their movement. Due to the remote region, there was no cellphone coverage, so data was routed through the peer-to-peer network to mobile base stations. The zebras were not constrained to certain areas, and other than the small devices attached to their bodies, left undisturbed. To accomplish this level of tracking without the use of MWSNs would not be possible.

- **Pollution Monitoring.** A mobile air quality monitoring system is presented in [102]. Sensor nodes that measure specific pollutants in the air are mounted on vehicles. As the vehicles move along the roadways, the sensors sample the air, and record the concentration of various pollutants along with location and time. When the sensors are in the proximity of access points, the data are uploaded to a server and published on the web.

Civil. One of the areas that has great potential for MWSN utility is that of civil services. This includes those non-military municipal applications that keep society running efficiently and safely.

- **Pothole Detection.** In [103], a system is developed to detect potholes on city streets. Deployed on taxi cabs, the sensor nodes contain an accelerometer, and can communicate using either opportunistic WiFi or cellular networks.

- **Wireless E-911.** In North America, the Enhanced 911 emergency telecommunications service, or *E911* [104], was established to connect callers with emergency services in a manner that would associate a physical location with the phone number of the caller. *Wireless* E-911

is the second phase of the E911 service mandated by the FCC, which requires wireless cellular devices to automatically provide user location when the service is invoked. Wireless E-911 is an important requirement; however, its implementation is non-trivial and different carriers choose to use different methods including embedded GPS chips, multilateration and multiangulation based on the known locations of cell towers.

Military / Aerospace. One of the biggest promoters, as well as one of the biggest funders, of wireless sensor technology is the military. There is a clear interest in localization services, tracking friendly and hostile entities, and navigation of autonomous robots, and intensive research is carried out in this area.

- **Shooter Detection / Weapon Classification.** In [105], a soldier-wearable sensor system is developed that identifies the location of an enemy sniper and also identifies the weapon being fired. Each sensor consists of an array of microphones mounted on the helmet of a soldier. The sensor observes both the shock wave of the projectile, as well as the muzzle blast from the weapon, and based on TDOA, as well as properties of the acoustic signal, is able to triangulate the enemy position and classify the weapon type.

- **Autonomous Deployment.** In [106], an unattended aerial vehicle is used for sensor network deployment and repair. Such deployments aid the military in battlefield surveillance and command and control field operations.

Mobile Sensor Navigation

Navigation systems have been around for centuries, and have traditionally relied on celestial observation, such as measuring the angle between the horizon and the sun, or the relative positions of certain luminous stars [107]. More recently, navigation has been adapted to autonomous mobile robots, whereby the robot is responsible for traveling from region A to region B. Navigation is defined as "the process of determining and maintaining a course or trajectory to a goal location" [108]. In this definition, a *goal location* does not necessarily refer to the final destination of the mobile entity, because there may not be one (i.e., the mobile entity follows a circular route). Instead, the goal location describes a bounded region to where the mobile entity must proceed next.

Navigation methods have come a long way compared with their historical nautical counterparts. Although many new methods have been developed, especially during the past century, the challenges they address are not new. These challenges involve answering the following three questions:

(a) "Where am I?", (b) "Where are other places with respect to me?", and (c) "How do I get to other places from here?" Although it may not be necessary to answer all of these questions to accurately navigate through a region, modern navigation systems must be able to answer at least one [109]. Furthermore, in addition to determining a path between the current and goal positions, it may also be necessary to employ additional navigational tools, such as obstacle avoidance and search algorithms. These all require some form of sensory input that can ultimately be transformed into a motion vector to drive the mobile entity toward its goal position.

Typical sensors used for robot navigation include cameras, laser rangefinders, and sonar, to name a few. Figure 9 shows a robot equipped with these devices. They can be extremely accurate, however, they are bulky, consume a significant amount of energy, and can have substantial processing requirements. Consequently, they are inappropriate for resource-constrained mobile sensors. However, sensors do exist that fit on mote-scale devices. Such sensors include accelerometers, digital compasses, gyroscopes, and wheel encoders. Many navigation techniques have been developed that use data obtained from these types of sensors, as described below.

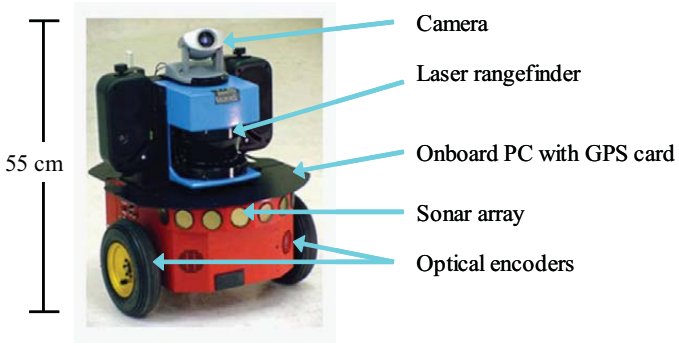


Figure 9: A mobile robot equipped with camera, laser rangefinder, sonar panels, GPS, and optical encoders.

Navigation Taxonomy

Navigational behavior falls into two categories, local and way-finding (also referred to as waypoint navigation) [108]. Each represents a strategy for reaching a goal location. The local strategy enables the mobile entity to reach the goal position by use of sensor input and, possibly, knowledge of the area. All sensor input needed to reach the goal position is available within the current sensor field of view [110]. On the other hand, way-finding requires the sequential observation of several landmarks in order to reach the goal [111]. Once the mobile entity is near the goal, it may switch to local navigation techniques to arrive at an exact position and orientation.

Local navigation includes the following four behaviors [110], [108]. These behaviors form a hierarchy in order of increasing complexity. It is assumed that if a mobile entity has the ability to perform one of these behaviors, it has the ability to perform those behaviors that precede it in the hierarchy.

- **Searching.** This is the most basic local navigation behavior, only requiring the capability for recognizing the goal. The mobile entity moves randomly, or based on some motion pattern (such as a grid search), but will only find the goal by chance. An example of this behavior is a search and rescue operation.

- **Direction-following.** In many situations, the goal can be found by following a path. The path could be a magnetic field, compass point, or celestial cue. An example would be a ship traveling to the New World from Europe by heading west. This requires the mobile entity to align itself along a given axis. Although the mobile entity may not have the goal in its field of view at all times, by following the path, the goal will be reached.

- **Aiming.** In the direction-following approach, the mobile entity follows a path to the goal position. In contrast, the aiming behavior requires the mobile entity to keep the goal in front of it at all times. The mobile entity reaches the goal by recognizing it from a distance, (for example, the goal may be emitting a beacon,) and moving toward it. An example of navigation by homing in on a lit airstrip at night.

- **Guidance.** Oftentimes the goal position may not be visible to the mobile entity, and it must be guided there based on a combination of sensory input and a known spatial relationship between the current position, the goal position, and the surrounding environment. An example of a guidance-based local behavior is directing a ship into port based on the known locations of buoys and the dock.

Way-finding navigation includes the following three behaviors [108]. Like local navigation, these behaviors fit into a hierarchy based on their complexity.

- **Recognition-triggered Response.** This behavior assigns a local navigation strategy to connect two locations. The local strategy is selected based on either a start or goal position and a known route between the two. The responses to the current or goal position are fixed, that is, the same local strategy will always be used for navigating between the two positions.

- **Topological.** The topological behavior overcomes the restriction of the recognition-triggered response. For example, if a route involves passing from one room to another, and the passage is blocked by a closed door, a mobile entity navigating by recognition-triggered response would be stuck at the door. The topological behavior enables the mobile entity to reach the goal position by using a complimentary strategy that may enable it to sidestep the blocked doorway and take an alternate route into the room.

- **Survey Navigation.** The above two way-finding strategies are restricted to traveling on known routes to reach the goal position. Survey navigation takes all known information about the environment into consideration, enabling additional uncharted routes to be taken, when necessary.

These navigation behaviors are summarized in Table 2.

Behavior	Characteristics
Search	Goal is reached by chance; Mobile entity must be able to recognize goal
Direction-following	Mobile entity reaches goal by following a path or direction
Aiming	Mobile entity orients toward goal by following an emitted signal
Guidance	Mobile entity is guided toward goal by features in the environment
Recognition-triggered response	Mobile entity follows a fixed route between endpoints, determined by location and local search method
Topological	Mobile entity can select a route between endpoints, based on combinations of known routes
Survey	Mobile entity can create new routes between endpoints, based on knowledge of the environment

Table 2: Local (top) and way-finding (bottom) navigation behaviors.

Navigation Approaches

There are two main approaches for using mobile sensors for navigation [112]. The first is dead reckoning, which was introduced earlier in this chapter. The second is reference-based, which typically requires an infrastructure that can be observed by the mobile entity in order to learn its orientation and position. We emphasize a particular class of reference-based navigation called angle-based navigation, which is used extensively in our work.

Dead reckoning uses onboard sensors to determine the distance traveled over a designated time interval. Distance can be obtained using odometry via encoders, or by inertial navigation techniques

such as accelerometers and gyroscopes. The main benefit of using dead reckoning systems is that no external infrastructure is required. Position can be inferred by integrating velocity, or doubly integrating acceleration, with respect to time; however, error will accrue unbounded unless the mobile entity can periodically adjust the error by using known reference positions. Navigation techniques that employ dead reckoning include [16], [87], [88], [113], and [26].

In reference-based systems, mobile entities use landmarks in the region for correct positioning and orientation. Landmarks can be active beacons, such as sensor nodes, satellites, and lighthouses, or structures in the environment, such as trees and buildings. A common use for reference-based systems is model-matching, also referred to as mapping. Mapping requires the ability to detect landmarks in the environment, and match them to a representation of the environment that was obtained a priori and stored in the memory of the mobile entity. For mobile robots, landmarks are typically detected using cameras. However, the landmarks do not need to be structural. RSS profiling [75], [76] is a type of model-matching technique. Simultaneous localization and mapping (SLAM) [114], [101] is also a type of mapping, in which the mobile entity builds a map of the environment at the same time as it determines its position. Similarly, simultaneous localization and tracking (SLAT) [115] is a technique to localize a mobile entity while keeping track of the path it has taken to arrive at its present position.

Time Synchronization

Time synchronization is a critical component in many wireless sensor network applications, most notably, localization and navigation. However, accurately synchronizing the clocks of all sensor nodes within the network is not a trivial task. This is especially true in sensor networks that contain mobile nodes, which may join and leave the network sporadically.

Synchronization Protocols

Synchronization protocols can be classified as *sender-receiver*, in which one node synchronizes with another, or *receiver-receiver*, in which multiple nodes synchronize to a common event. Both have their advantages, and each can provide synchronization accuracy on the order of microseconds using certain configurations. An in-depth survey on time synchronization in WSNs can be found in [116].

Several sender-receiver synchronization protocols have been developed for the Berkeley motes and similar small-scale devices that provide microsecond accuracy. Elapsed Time on Arrival (ETA) [52] provides a set of application programming interfaces for an abstract time synchronization service.

In ETA, sender-side synchronization error is essentially eliminated by taking the timestamp and inserting it into the message *after* the message has already begun transmission. On the receiver side, a timestamp is taken upon message reception, and the difference between these two timestamps is an estimate of the clock offset between the two nodes. The routing-integrated time synchronization protocol (RITS) [117] is an extension of ETA over multiple hops. RITS incorporates a set of routing services to efficiently pass sensor data to a network sink node for data fusion.

In the Flooding Time Synchronization Protocol (FTSP) [97], a single *root* node is selected to maintain a global timescale. The root floods the network with a timestamp representing its local time. At each hop, the timestamp is converted into the timescale of the local node, then forwarded to neighbors until the entire network is synchronized.

The Timing-sync Protocol for Sensor Networks (TPSN) [118] uses the round-trip latency of a message to estimate the time a message takes along the communication pathway from sender to receiver. This time is then added to the sender timestamp and transmitted to the receiver to determine the clock offset between the two nodes.

In [119], mote-PC synchronization was achieved by connecting the GPIO ports of a mote and a PDA. The PDA timestamped the output of a signal, which was captured and timestamped by the mote. The mote then sent the timestamp back to the PC, which was able to calculate the clock offset between the two. Although using this technique can achieve microsecond-accurate synchronization, it was implemented as a hardware modification rather than in software.

Reference broadcast synchronization (RBS) [51] is a receiver-receiver protocol that minimizes error by taking advantage of the broadcast channel found in most networks. Messages broadcast at the physical layer will arrive at a set of receivers within a tight time bound due to the almost negligible propagation time of sending an electromagnetic signal through air. Nodes then synchronize their clocks to the arrival time of the broadcast message.

Table 3 provides a comparison of some common synchronization protocols.

Protocol	Type	Accuracy	Global clock
ETA [52]	Sender-receiver	Microseconds	No
RITS [117]	Sender-receiver	Microseconds / hop	No
FTSP [97]	Sender-receiver	Microseconds / hop	Yes
GPS [25]	Sender-receiver	Hundreds of nanoseconds	Yes
TPSN [118]	Sender-receiver	Tens of microseconds	No
RBS [51]	Receiver-receiver	Microseconds	No
TSS [120]	Sender-receiver	Hundreds of microseconds	No

Table 3: Comparison of common time synchronization protocols.

CHAPTER III

TIME SYNCHRONIZATION IN HETEROGENEOUS SENSOR NETWORKS

Introduction

A common class of WSN applications detect physical phenomena through periodic sampling of the environment. In order to make sense of the individually collected samples, nodes pass their sensor data through the network to a centralized *sensor-fusion* node where it can be combined and analyzed. This process is referred to as *reactive data fusion*.

One important aspect of reactive data fusion is the need for a common notion of time among participating nodes. For example, acoustic localization requires the cooperation of several nodes in estimating the position of the sound source based on time-of-arrival data of the wave front [59], [58]. This may require up to 100-microsecond accurate synchronization across the network. Another example is acoustic emissions (AE), the stress waves produced by the sudden internal stress redistribution of materials caused by crack initiation and growth [121]. As the speed of sound is typically an order of magnitude higher in metals than in the air, the synchronization accuracy required for AE source localization can be in the tens of microseconds.

Existing WSN time synchronization protocols (e.g. [120], [51], [118], [97], [117]) perform well on the devices for which they were designed. However, these protocols tend to break down when applied to a network of *heterogeneous* devices. For example, RITS [117] was designed to run on the Berkeley motes, and assumes the operating system is tightly integrated with the radio stack. Attempting to run RITS on an 802.11 network can introduce an unacceptable amount of synchronization error because it requires low-level interaction with the hardware, which is difficult to attain in PCs. RBS [51], although portable when it comes to operating system and computing platform, is accurate only when all nodes have access to a common network medium. A combined network of Berkeley motes and PDAs with 802.11b wireless network cards, for example, would be difficult to synchronize using RBS alone because they communicate over different wireless channels. In addition, the communication pathway between a mote and PC is realized using a serial connection. Synchronization across this interface is essential when attempting to combine time-dependent sensor data from each device. Furthermore, it may be desirable to have several mote-PC gateways, since often it is not always possible to extract data fast enough through a single base station. In large networks, multiple gateways also enable data packets to reach a base station in a fewer number

of hops, thus minimizing delay and conserving energy. Although time synchronization across this interface has previously been explored [119], it has not been implemented in software.

The work presented in this chapter focuses on achieving microsecond-accuracy synchronization in heterogeneous sensor networks [11], [17]. HSNs are a promising direction for developing large sensor networks for a diverse set of applications [7], [8], [9]. We consider a multi-hop network consisting of Berkeley motes and Linux PCs, a dominant configuration in HSNs. Mote networks consist of resource-constrained devices capable of monitoring environmental phenomena. PCs can support higher-bandwidth sensors such as cameras, and can run additional processing algorithms on the collected data. In this sense, we model both motes and PCs as sensor nodes. We envision a scenario in which resource-constrained mobile sensors must coordinate in time and space with high-bandwidth devices. Such a scenario would be useful in multi-modal sensor fusion applications [6] that monitor traffic in urban areas, for example. Time synchronization in mote and PC networks have been studied independently, however, a sub-millisecond software method to synchronize these two networks has not yet been developed to the best of our knowledge.

The contributions of this work are as follows [11], [17]:

1. We develop a methodology for HSN time synchronization that utilizes a combination of existing synchronization protocols. The methodology incurs little overhead of network resources and has synchronization error on the order of microseconds.

2. We implement a time synchronization service for reactive data fusion applications, which allows the application developer to focus on aspects of sensor fusion and not the underlying synchronization and aggregation mechanisms.

3. To achieve accurate cross-platform synchronization, we develop a technique for synchronization between a mote and PC that is implemented completely in software, and remains effective when multiple mote-PC connections exist within the network.

4. We perform an analysis of various clock synchronization and clock skew compensation techniques on real-world hardware platforms.

5. Our HSN time synchronization service has been effectively demonstrated in multi-modal sensor fusion target tracking applications [6], [122], [123], [124].

Problem Statement

Each sensor node in the network is a computing device that maintains its own local clock. Internally, the clock is a piece of circuitry that counts oscillations of a quartz crystal, energized at a specific frequency. When a certain number of these oscillations occur, a *clock-tick* counter is incremented. This counter is accessible from the operating system and its accuracy (with respect to atomic time) depends on the quality of the crystal, as well as various operating conditions such as temperature, pressure, humidity, and supply voltage. When a sensor node registers an event of interest, it will access the clock-tick counter and record a *timestamp* reflecting the time at which the event occurred.

Some protocols synchronize nodes to a global clock, often referred to as *real-time* or *Coordinated Universal Time* (UTC). Irrespective of whether a given protocol synchronizes to UTC, it is often convenient to represent the occurrence of an event (such as a coordination request or the detection of some physical phenomenon) according to some universal time standard. We use the notation t to represent an arbitrary UTC time, and the notation t_e to represent the UTC time at which an arbitrary event e occurred. Because each node records a timestamp according to its own clock, we specify the local time on node N_i at which event e was detected by the timestamp $N_i(t_e)$.

Although the two timestamps $N_i(t_e)$ and $N_j(t_e)$ correspond to the same real-time instant t_e, this does not imply that $N_i(t_e) = N_j(t_e)$; the clock-tick counter on node N_i may be offset from N_j. Therefore, from the perspective of node N_i, we define the *clock offset* with N_j at real-time t as $\phi_j^i(t) = N_i(t) - N_j(t)$. It may be the case that the offset changes over time. In other words, the *clock rate* of node N_i, $\frac{dN_i(t)}{dt}$, may not equal the ideal rate ($\frac{dN_i(t)}{dt} = 1$). We define the ratio of clock rates of two nodes as the *relative rate*, $rr_j^i = \frac{dN_i(t)}{dN_j(t)}$. The relative rate is a quantity directly related to the *clock skew*, defined as the difference between clock rates, and is typically used by clock skew compensation methods. We refer to clock offset and clock rate characteristics as a node's *timescale*.

Practically all synchronization protocols can be implemented using *timescale transformation*. Rather than setting one clock to another, clocks advance uninhibited, and instead a reference table is maintained that keeps track of the clock offsets and clock drift between a node and its neighbors. The reference table is used to transform a clock value from one timescale to another, providing each node with a common notion of time. Clock adjustment is disadvantageous in WSNs because it leads to increased overhead and the possible loss of monotonicity [125].

Sources of Synchronization Error

Synchronization often requires passing timestamped messages between nodes. However, this communication has associated message delay, which has both deterministic and nondeterministic components that introduce error into the timescale transformation. We call the sequence of steps involved in communication between a pair of nodes the *critical path*. Figure 10 illustrates the critical path in a wireless connection. The critical path is not identical for all configurations, however, it can typically be characterized by the steps outlined in the figure (for more details, see for example [51], [118], [97]). Table 4 lists the main types of message delay.

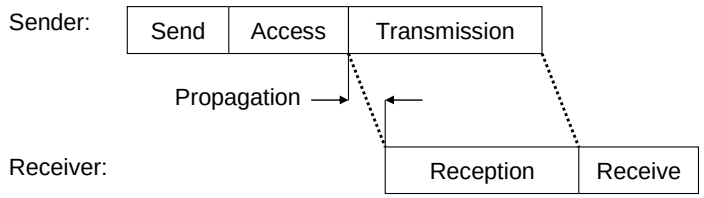


Figure 10: Critical path

Pathway component	Confidence	Delay
Send / Receive	Mostly nondeterministic	Hundreds of milliseconds
Access	Highly nondeterministic	Seconds
Transmission / Reception	Mostly deterministic	Tens of milliseconds
Propagation	Highly deterministic	Less than one microsecond

Table 4: Types of message delay incurred on the critical path for wireless communication.

In both mote and PC networks, the Send and Receive times are the delays incurred as the message is passed between the application and the MAC layer. These segments are mostly nondeterministic due to system call overhead and kernel processing. The Access time is the least deterministic segment in the critical path, and is the delay that occurs in the MAC layer while waiting for access to the communication channel. The Transmission and Reception times are the delays incurred from transmitting and receiving a message over the physical layer, bit by bit. They are mostly deterministic and depend on bit rate and message length. The Propagation time is the time the message takes to travel through physical space between the sender and receiver. Propagation delay is highly deterministic.

In heterogeneous sensor networks consisting of resource-constrained motes and resource-rich PCs, communication between the two is realized via serial connection. When this is the case, the Send and Receive times are similar to wireless communication, however, the Access time is nonexistent. Because the mote-PC connection does not have flow control, pending messages are immediately

transmitted without having to wait for an available channel. The Transmission time starts when the data on the sender is moved in 16-byte chunks to the buffer on the UART, and ends after the data is transmitted bit-by-bit across the serial port to the receiver. Similar to wireless networks, the Propagation time is minimal. The UART on the receiver places the received bits into its own 16-byte buffer. When the buffer is almost full, or a timeout occurs, it sends an interrupt to the CPU, which notifies the serial driver, and the data is transferred to main memory where it is packaged and forwarded to the user-space application.

In addition to the error from message delay nondeterminism, synchronization accuracy is also affected by clock skew when a pair of nodes operate for extended periods of time without updating their clock offset. For example, suppose at UTC time t, nodes N_1 and N_2 exchange their current clock values at local times $N_1(t)$ and $N_2(t)$, respectively. At some later time, an event e occurs that is detected and timestamped by N_1, which sends its timestamp $N_1(t_e)$ to N_2. If the clock rates on each node were equal, N_2 would simply be able to take the previously calculated offset and use it to transform the event timestamp $N_1(t_e)$ to the corresponding local time $N_2(t_e) = N_1(t_e) + \phi_2^1(t)$. However, if the relative rate $rr_2^1(t)$ is not equal to 1, but 1 + 20 ppm[1], for example, an attempt to convert $N_1(t_e)$ to the local timescale would result in an error of $20 * 10^{-6} * (N_2(t_e) - N_2(t))\mu s$. If the interval between the last synchronization and the event was one minute, the resulting error due to clock skew alone would amount to 1.2 milliseconds!

For accurate synchronization, it is therefore necessary to minimize the nondeterministic sources of error in the critical path, and account for the deterministic sources by appropriately adjusting the timescale transformation. Clock skew compensation is necessary for minimizing synchronization error when nodes run for long periods of time without updating their offset. With an estimation of clock offset and relative rate, a complete timescale transformation, which converts an event timestamp $N_j(t_e)$ from the timescale of node N_j to the timescale of N_i, can be defined as

$$N_i(t_e) = N_i(t_s) + rr_j^i(t_s)[N_j(t_e) - N_j(t_s)]$$

where $N_i(t_s)$ and $N_j(t_s)$ are the respective local times at which nodes N_i and N_j exchanged their most recent synchronization message, s.

[1]Parts per million (10^{-6}). A relative rate of 1 ppm means that one clock ticks $1\mu s/s$ faster than the other.

Clock Skew Compensation

Independent of synchronization protocol, there are several options for clock skew compensation. The simplest is to do nothing. In some applications, event detection and synchronization always occur so close together in time that clock skew compensation is unnecessary. However, when it does become necessary, nodes must exchange timestamps periodically to ensure their clocks do not drift too far apart. In resource-constrained WSNs, frequently exchanging timestamps may be undesirable because it can result in high message overhead. To keep message overhead to a minimum, nodes can exchange synchronization messages less frequently and instead maintain a history of their neighbors' timestamps. Techniques such as linear regression, exponential averaging, and phase-locked loops can then be used to produce an accurate estimate of clock offset at any time instant.

A linear regression fits a line to a set of data points such that the square of the error between the line and each data point is minimized overall. By maintaining a history of n local-remote timestamp pairs, node N_i can derive a linear relation $N_i(t) = \alpha + \beta N_j(t)$ and solve for the coefficients α and β. Here, β represents the estimation of $rr_j^i(t)$. A problem arises when attempting to improve the quality of the regression by increasing the number of data points. This can result in high memory overhead, especially in dense networks. However, it has been shown that sub-microsecond clock skew error can be achieved with as few as six timestamps in mote networks [97].

Exponential averaging solves the problem of high memory overhead by keeping track of only the current relative rate and the most recent neighbor-local synchronization timestamp pair. When a new timestamp pair is available, the relative rate is adjusted by

$$rr_j^i(t_k) = \alpha \left(\frac{N_i(t_k) - N_i(t_{k-1})}{N_j(t_k) - N_j(t_{k-1})} \right) + (1 - \alpha)(rr_j^i(t_{k-1})),$$

where α is a small weight constant that gives higher significance to the accumulated average. Because the relative rate estimate is partially derived from its previous value, there will be a longer convergence time before an accurate estimate is reached. This convergence time can be reduced by providing the algorithm with an initial relative rate, determined experimentally.

The phase-locked loop (PLL) is a mechanism for clock skew compensation used in the Network Time Protocol (NTP) [126], [127]. The PLL compares the ratio of a current local-remote timestamp pair with the current estimate of the relative rate. The PLL then adjusts the estimate by the sum of a value proportional to the difference and a value proportional to the integral of the difference. PLLs generally have a longer convergence time than linear regression and exponential averaging, but have low memory overhead. A diagram of a PLL implementation is illustrated in Figure 11.

The Phase Detector calculates the relative rate between two nodes and compares this with the output of the PLL, which is the previous estimate of the relative rate. The difference between these two values is the phase error, which is passed to the second-order Digital Loop Filter. Because we expect there to be some amount of phase error, we choose a filter with an integrator, which allows the PLL to eliminate steady-state phase error. To implement this behavior in software, a digital accumulator is used, and is represented by $y(t) = (K_1 + K_2)u(t) - 10K_2K_1u(t-1) + 10K_2y(t-1)$. The resulting static phase error is passed to the Digitally Controlled Oscillator (DCO). The DCO sums the previous phase error with the previous output, which produces the current estimate of relative clock rate, and is fed back into the PLL. Techniques for selecting gains K_1 and K_2 are presented in [126].

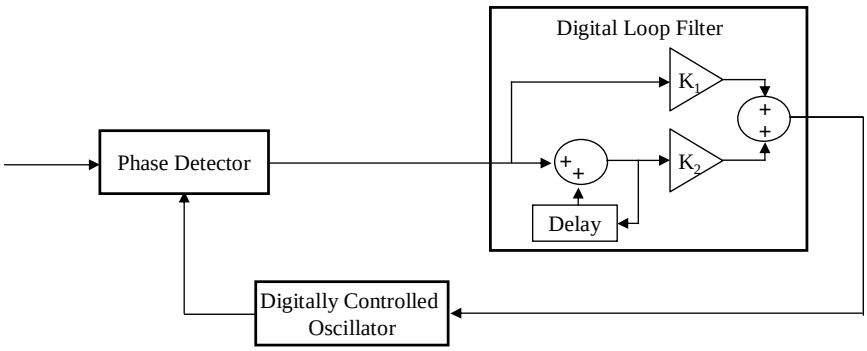


Figure 11: Phase-locked loop

Synchronization Methodology

The accuracy of a receiver-receiver synchronization protocol (such as RBS) is comparable to sender-receiver synchronization (such as RITS) in mote networks. However, receiver-receiver synchronization has greater associated communication overhead, which can shorten the lifetime of the network. Therefore, we selected RITS to synchronize the mote network in our HSN.

Synchronization of PC networks has been studied extensively over the past four decades, however, popular sender-receiver protocols such as NTP only provide millisecond accuracy. This is acceptable because PC users typically do not require greater synchronization precision for their applications. For microsecond-level synchronization accuracy in PC networks, a receiver-receiver protocol such as RBS outperforms sender-receiver protocols because it has less associated message delay nondeterminism. We therefore use RBS to synchronize our PC network.

To synchronize a mote with a PC in software, we adopted the underlying methodology of ETA and applied it to serial communication. On the mote, a timestamp is taken upon transfer of a

synchronization byte and inserted into the outgoing message. On the PC, a timestamp is taken immediately after the UART issues the interrupt, and the PC regards the difference between these two timestamps as the PC-mote clock offset. Serial communication bit rate between the mote and PC is 57600 baud, which approximately amounts to a transfer time of 139 microseconds per byte. However, the UART will not issue an interrupt to the CPU until its 16-byte buffer nears capacity (when the 14th byte is received) or a timeout occurs. Because the synchronization message is six bytes, reception time in this case will consist of the transfer time of the entire message in addition to the timeout time and the time it takes to transfer the data from the UART buffer into main memory by the CPU. This time is compensated for by the receiver, and the clock offset between the two devices is determined as the difference between the PC receive time and the mote transmit time.

Architecture

We have developed a PC-based time synchronization service for reactive data fusion applications in HSNs. Figure 12 illustrates the interaction of each component within the service. The service collects sensor data from applications that run on the local PC, as well as from other service instances running on remote PCs. It accepts event messages on a specific port, converts the embedded timestamps to the local timescale, and forwards the messages toward the sensor-fusion node. To maintain synchronization with the rest of the PC network, the service uses RBS. The arrival times of the reference broadcasts are stored in a reference table and accessed for timescale transformation. In addition, the service accepts mote-based event messages, and converts the embedded timestamps using the ETA serial timestamp synchronization method outlined above. The messages are then forwarded toward a sensor-fusion node. The service instance that resides on the sensor-fusion node transforms incoming timestamps into its local timescale before passing the event messages up to the sensor-fusion application.

Kernel modifications in the serial and wireless drivers were required in order to take accurate timestamps. Upon receipt of a designated synchronization byte, the time is recorded and passed up to the synchronization service in the user-space. The mote implementation uses the TimeStamping interface, provided with the TinyOS distribution [128]. A modification was made to the UART interface to insert a transmission timestamp into the event message as it is being transmitted between the mote and PC. The timestamp is taken immediately before a synchronization byte is transmitted, then inserted at the end of the message.

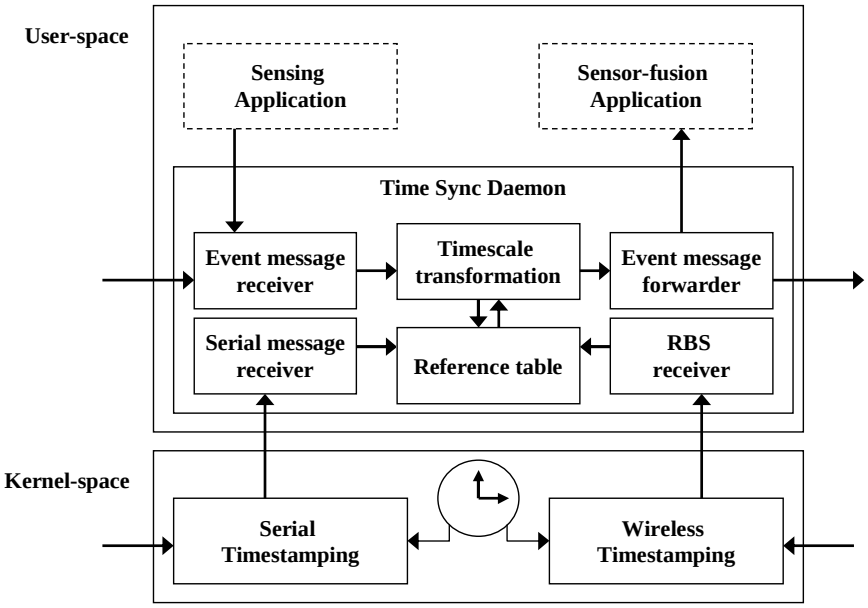


Figure 12: PC-based time synchronization service.

Experimental Evaluation

Our HSN testbed consists of seven Crossbow Mica2 motes and four MobileRobots Pioneer 3DX robots [129] with embedded Redhat Linux 2.4 PCs, as illustrated in Figure 13. In addition we employ a Linux PC to transmit RBS beacons. We chose this testbed because the challenges that arise are representative of practical HSN configurations such as hierarchical clustering and mote networks with multiple sinks. In addition, routing sensor data from a mote network to a PC base station is a dominant communication pathway in sensor network architectures.

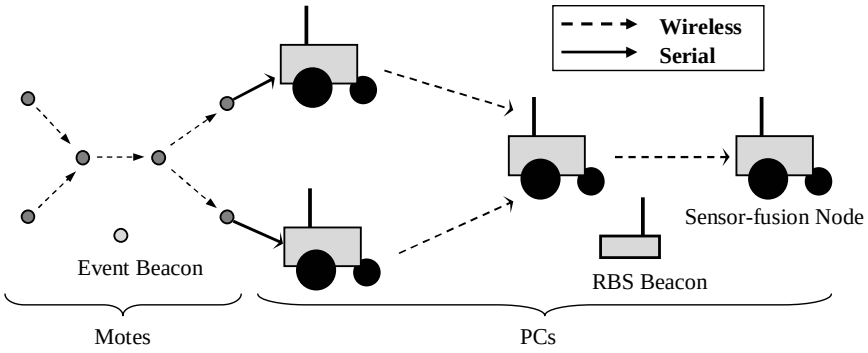


Figure 13: Our sensor network testbed. Arrows indicate communication flow.

The reference broadcast node transmits a reference beacon containing a sequence number once every ten seconds. The arrival of these messages are timestamped in the kernel and stored in a reference table. Simultaneously, a designated mote broadcasts event beacons, once every $4000 \pm \epsilon$

milliseconds, where ϵ is a random number in the range (0,1000). Six hundred event beacons are broadcast per experiment. The two outlying motes timestamp the arrival of the event beacon, and place the timestamp into a message. The message is routed over three hops in the mote network to the mote-PC gateways, using RITS to convert the timestamp to the local timescale with each hop. The message is next transferred from the mote network to the PC network over the mote-PC serial connections, and the event timestamp is converted to the timescale of the gateway PCs. The gateway PCs forward the message two additional hops to the sensor-fusion node. The experiment was repeated using the different clock skew compensation techniques in the PC network, as described in Chapter II. Because RITS synchronizes a single sender with a single receiver at the time of data transfer, and because event data is forwarded to the base station immediately after the event is detected or a data message is received, clock skew compensation in the mote network provides negligible improvement.

Sub-network Synchronization

We first performed a series of experiments to quantify synchronization error on the individual critical paths in the network. The results allow us to justify our selection of synchronization protocols for the entire HSN. Note that no clock skew compensation was performed for these initial tests. To determine synchronization error, we used the *pairwise difference* evaluation method. Two nodes, N_1 and N_2, simultaneously timestamp the occurrence of an event, such as a reference beacon. These timestamps are then transformed to the timescale of node N_3, and the absolute value of their difference represents the error in the timescale transformation.

Experimental results in the literature (e.g., [117], [52]) indicate that RITS works well for synchronizing the mote network. We confirmed this on a 3-hop network of Mica2 [32] motes. A beacon message was broadcast to two outlying motes, which timestamped its arrival and forwarded the timestamp to a network sink node 3 hops away. At each intermediate node, the timestamps were converted to the local timescale. Synchronization error was calculated as timestamps arrived at the network sink node and, over 100 synchronizations, the average error was $7.27\mu s$, with a maximum of $37\mu s$.

Based on the implementation described in [51], we synchronized our PC network using RBS. We used a separate transmitter to broadcast a reference beacon every ten seconds (slightly randomized) for 100 runs. Two PCs received reference broadcast r at local times $PC_1(t_r)$ and $PC_2(t_r)$, respectively. Synchronization error was $8.10\mu s$ on average, and $43\mu s$ maximum. Results are displayed in Figure 14a.

Figure 14b plots the synchronization error between two PCs using RITS. Every two seconds, PC_1 sent two synchronization messages to PC_2. Immediately before the command to send the first message was issued to the network interface controller on the sender, a transmission timestamp $PC_1(t_{tx})$ was taken in the kernel. This was found to be the latest possible time for the sender to take the timestamp. However, by the time the timestamp had been acquired, the message had already been transmitted, so a second message was needed to deliver the timestamp to the receiver. PC_2 recorded the timestamp $PC_2(t_{rx})$ in the kernel interrupt function upon receipt of the first message, and obtained the sender timestamp in the second message shortly after. The results show that we cannot expect consistent synchronization accuracy using RITS with the 802.11 networked Linux PCs. This is partly due to sender-side message delay error in RITS, which is nonexistent in RBS. In addition, the PC-based operating system is not tightly integrated with the network interface, and therefore the timestamping precision of the transmission and reception of synchronization bytes is degraded.

To synchronize the mote with the PC, we used the synchronization methodology described above. To evaluate synchronization accuracy, GPIO pins on the mote and PC were connected to an oscilloscope, and set high upon timestamping. The resulting output signals were captured and measured. The test was performed over 100 synchronizations, and the resulting error was $7.32\mu s$ on average. The results are displayed in Figure 15. The majority of the error is due to nondeterministic message delay resulting from jitter, both in the UART and the CPU. A technique to compensate for such jitter on the motes is presented in [97], however, we did not attempt it on the PCs.

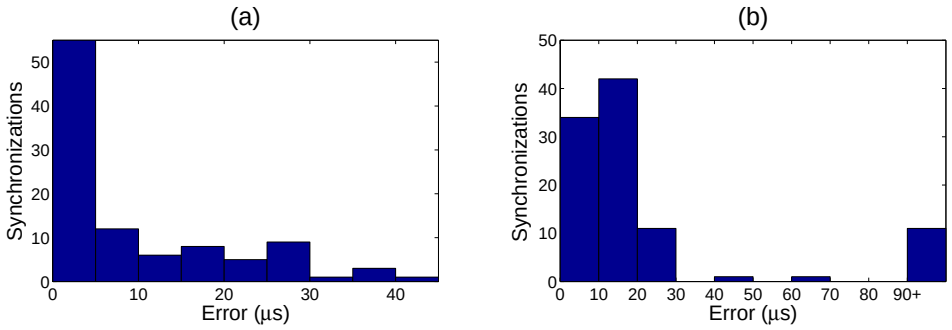


Figure 14: (a) RBS and (b) RITS synchronization error between two PCs.

HSN Synchronization Results

Figure 16 summarizes the synchronization error for each type of clock skew compensation technique under normal operating conditions, high network congestion, and high I/O load. To simulate a high

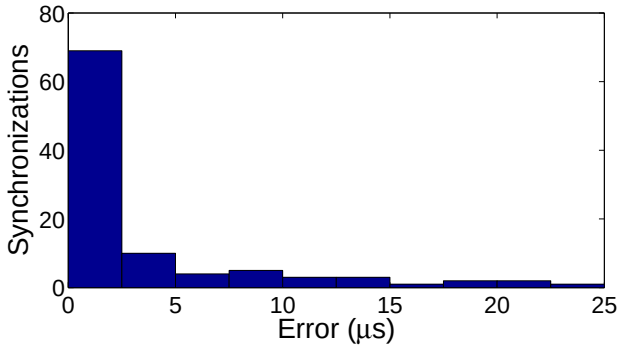


Figure 15: Mote-PC error using ETA.

network load, an extra mote and PC (not pictured in Figure 13) were introduced, each broadcasting messages of random size every 100 milliseconds. We first examined synchronization without clock skew compensation. The sensor-fusion node reported the average difference between the source timestamps as $12.05\mu s$, with a maximum of $270\mu s$. Next, we synchronized the PC network using linear regression as our clock skew compensation technique. As expected, there was a notable improvement in synchronization accuracy, with an average of $7.62\mu s$ error, and a maximum of $84\mu s$. Repeating the experiment using exponential averaging gives errors similar to linear regression. The average error recorded was $8.82\mu s$, with a maximum of $112\mu s$. The average synchronization error using phase-locked loops was $7.85\mu s$, with a maximum of $196\mu s$. For the digital loop filter, we used gains of $K_1 = 0.1$ and $K_2 = 0.09$, determined experimentally.

Memory overhead is minimal for each clock skew compensation technique. Nodes require 8 bytes for the current relative rate estimate for each neighbor within single-hop range. In the case of linear regression, a small history buffer for each neighbor is also required. Our implementation has no message overhead (except for the beacons transmitted by the RBS server). In fact, the only modification to the data message is the addition of a four-byte timestamp. Because these synchronization timestamps piggyback on data messages, no additional messages are required. Our methodology is therefore energy efficient, because message overhead directly impacts energy consumption. Convergence time depends on input parameters to the clock skew compensation algorithm. We found that, on average, it took the network 80 seconds to synchronize with linear regression, 200 seconds with exponential averaging, and 1300 seconds with phase-locked loops. Note that these convergence times reflect an inter-beacon delay of four seconds.

These results show that we are able to achieve microsecond-accurate synchronization in the HSN. Because error accrues with each timescale transformation, achieving this level of synchronization accuracy over a 6-hop network of heterogeneous devices is significant. Although the maximum

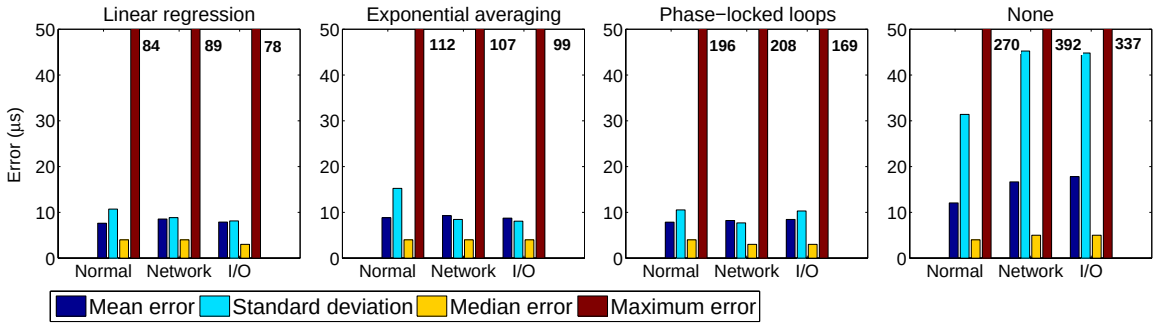


Figure 16: HSN synchronization error with different types of clock skew compensation under normal operating conditions (Normal), high network congestion (Network), and high I/O load (I/O).

synchronization error extends to tens of microseconds, it was principally caused by a small number of synchronization attempts in which prolonged operating system interrupt operations occurred. Although these prolonged interrupts were difficult to avoid, they did not occur frequently, and the worst-case synchronization error for each type of clock skew compensation technique was acceptable for most kinds of HSN data fusion applications. Furthermore, time synchronization was not affected by the most common types of nondeterministic behavior, such as network congestion and I/O operations.

The accuracy of the mote-PC synchronization is quite good. Although we did see error in the tens of microseconds, the maximum did not exceed $50\mu s$, and the average was below $10\mu s$. These results are significant, because they demonstrate that software-based, microsecond-accuracy synchronization can be achieved between mote and PC networks, enabling the development of HSN applications that require precision time synchronization. In addition, because this technique uses UART communication, it can easily be adapted for synchronization with other UART-supported peripheral sensing devices.

Conclusion

Time synchronization is an important and necessary component in most wireless sensor network applications. However, as we describe in this chapter, its implementation is non-trivial, especially when high-precision synchronization is required. In networks of heterogeneous devices, the problem is compounded by issues of system integration, and therefore alternative techniques must be employed to reduce synchronization error. We have shown that with certain configurations, microsecond accuracy can be achieved with careful selection of hardware, software, and network components. Our methodology is generally portable to other platforms, provided mechanisms exist within the target configuration that permit low-level timestamping.

CHAPTER IV

MOBILE SENSOR LOCALIZATION AND NAVIGATION USING RF DOPPLER SHIFTS

Introduction

In this chapter we address the problem of mobile sensor localization and navigation by describing a technique that uses RF Doppler shifts [28] for determining position and velocity. We design a system using inexpensive and commercially available sensor nodes, which eliminates the need for additional sensor hardware because only the sensor radio is used. The basic idea is that a mobile sensor transmits a sinusoidal signal at a specific frequency. Because the mobile sensor is moving, the observed signal frequency will be shifted due to the Doppler effect, and the shift is related to the relative speed and position of the transmitter and receivers. By using a sufficient number of stationary observers, the position and velocity of the mobile sensor can therefore be estimated.

Navigation is a challenging problem for resource-constrained mobile sensors. Typically, mobile robots utilize optical encoders, sonar arrays, laser range-finders, and GPS. These sensors enable the robots to determine their current location, as well as the correct motion vector to reach their destination. However, these types of devices can be large, heavy, expensive, power-intensive, and generate large amounts of data. As mobile sensors decrease in size and cost, these types of positioning sensors become difficult to implement. Furthermore, uneven terrain, motor degradation, and wheel slippage will often cause mobile entities to deviate from their intended trajectories. Motion controllers enable mobile entities to maintain a desired trajectory by minimizing position and velocity error in a stable manner.

We present the design of a motion control loop that uses observed Doppler shifts as feedback. An extended Kalman filter is used to determine the position and velocity of a mobile sensor in the presence of measurement noise. The position and velocity estimate is used by the controller to determine updated angular velocities for each wheel of the two-wheeled mobile sensor with differential steering. Our experimental results demonstrate the accuracy of this system with respect to maintaining the proper trajectory, as well as performing waypoint navigation.

The contributions of this work are as follows [18], [19]:

1. We develop an algorithm that estimates the position and velocity of a mobile node from RF Doppler-shifted frequency measurements. The position and velocity obtained from the

algorithm are used as control feedback, which keeps the mobile node from deviating from its target trajectory.

2. We implement the algorithm on a real-world resource-constrained WSN platform. All processing runs on two robot-mounted motes within the control loop, and requires no additional PC processing.

3. We perform several experiments using the implementation in order to demonstrate the viability and accuracy of the navigation algorithm.

System Overview

For our navigation system, which we refer to as *dNav*, the mobile sensor is mounted to a 2-wheeled mobile platform with differential steering. Similar to dTrack [28], the mobile node transmits a sinusoidal signal, which becomes Doppler shifted based on the relative speed between the transmitter and receiver. Receiver nodes send their observed Doppler-shifted frequencies back to the mobile node, which uses this data to compute the current estimate of its position, speed, and heading. These values are then used to calculate the trajectory error, which enables the mobile node to stay on course.

Although we apply the same underlying methodology, there are significant fundamental differences between dTrack and dNav. Unlike dTrack, for dNav we utilize the Doppler-shifted frequency information as feedback to *control* the mobile node. This approach requires the navigation algorithm to be implemented on the mobile sensor node, within the control loop. One benefit of dNav over dTrack is that the mobile node is aware of the angular velocity commands it sends to each wheel. This information can be used to arrive at a better position and velocity estimate, especially when sudden maneuvers take place.

In dNav, a typical sensing region consists of a set of anchored *receiver* nodes at known positions, as well as a stationary *assistant transmitter* node. The mobile node, also referred to as the *master transmitter*, moves around the sensing region, as in Figure 17. Periodically, the master synchronizes with the assistant and receiver nodes via a *SyncEvent* message [52]. In addition to serving as a synchronization point, the message provides a time in the near future for the master and assistant to transmit a signal, and for the receivers to listen for the transmission. The master and assistant transmit pure sine waves at frequencies that differ slightly, which generate a low-frequency beat signal upon arrival at the receiver antennas [22]. The observed beat frequency will be Doppler-

shifted, and the amount of Doppler-shift will be determined by the velocity of the mobile node and the relative positions of the stationary receivers.

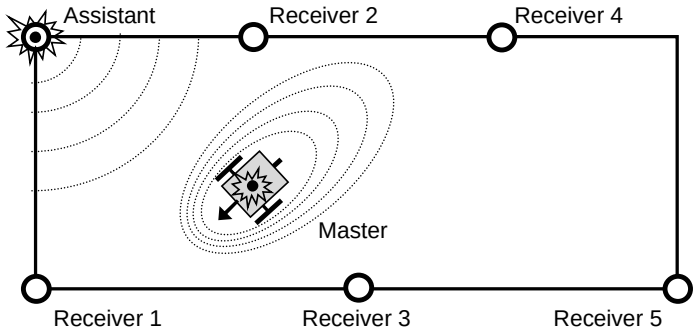


Figure 17: Sensing region for localization and navigation. Dotted curves represent radio wave propagation. The arrow on the master node indicates direction of travel.

The receiver nodes send the observed signal frequencies back to the master, and the observations are passed through an EKF in order to arrive at an estimated node trajectory in the presence of measurement noise. The output of the filter is the current estimate of the mobile node's position, speed, and heading, which are then used to calculate the trajectory error, based on reference speed and heading setpoints. The error is passed to the controller, which outputs updated left and right wheel angular velocities. Figure 18 illustrates the navigation system running on the mobile sensor. Each component of the architecture is presented in detail below.

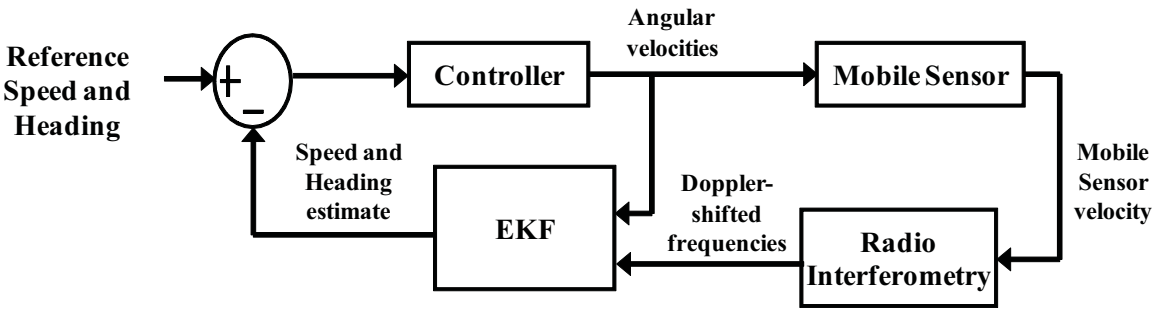


Figure 18: The dNav mobile sensor navigation system.

Mobile Sensor Kinematics

We use the following equations to describe the kinematic model of our two-wheeled mobile sensor with differential steering:

$$\dot{x} = \frac{r(\omega_r + \omega_l)}{2} cos\phi \tag{2}$$

$$\dot{y} = \frac{r(\omega_r + \omega_l)}{2} sin\phi \tag{3}$$

$$\dot{\phi} = \frac{r(\omega_r - \omega_l)}{2b} \tag{4}$$

where x and y constitute the mobile node's position, ϕ is the heading, r is the wheel radius, b is the distance between the hub center of the driving wheel and mobile platform axis of symmetry, and ω_r and ω_l are the right and left wheel angular velocities, respectively. The speed of the mobile node is the magnitude of the velocity, and in terms of angular velocity is represented as $|v| = \frac{r(\omega_r+\omega_l)}{2}$. For simplicity, this model does not take into account the effect of acceleration. For mobile entities with sufficiently low mass, acceleration does not have a major impact on the forward kinematics of the system.

Controller

To arrive at the angular velocities that will keep the mobile sensor on the reference trajectory, we use a controller that takes as input the speed and heading errors, $e_{|v|}$ and e_ϕ, respectively. Because ϕ wraps to 0 at 2π, we shift the heading error to fall between $-\pi$ and π:

$$e_\phi = \begin{cases} e_\phi - 2\pi & \text{if } e_\phi > \pi \\ e_\phi + 2\pi & \text{if } e_\phi < -\pi \\ e_\phi & \text{otherwise} \end{cases}$$

The controller contains two PI equations, one for each error component:

$$|v| = K_p e_{|v|} + K_i \int e_{|v|} dt \tag{5}$$

$$\dot{\phi} = K_p e_\phi + K_i \int e_\phi dt \tag{6}$$

The PI equations give us the updated speed and angular velocity, however, the mobile platform is commanded by specifying an angular velocity for each wheel. Consequently, we convert $|v|$ and $\dot{\phi}$ into individual wheel angular velocities, ω_l and ω_r, as follows:

$$\omega_l = \frac{|v| - b\dot{\phi}}{r} \tag{7}$$

$$\omega_r = \frac{|v| + b\dot{\phi}}{r} \tag{8}$$

These angular velocities constitute the drive input, u.

The effect of the above transformation is that both wheels will be set with an equal base velocity to compensate for the translational speed error. If heading error exists, the mobile node will minimize it by turning one wheel faster than the base velocity, and the other wheel slower, which will result in the node turning in the correct direction as it moves forward.

Doppler Tracking

A mobile node, T, moves through a sensing region with velocity $\overrightarrow{v}$ while transmitting a pure sine wave with frequency f_t and wavelength $\lambda_t = c/f_t$, where c is the speed of light. Simultaneously, a stationary node transmits a pure sine wave at a slightly lower frequency, f_a. A receiver node, R_i, observes the interference signal, with frequency f_i, which is Doppler-shifted. The amount of frequency shift is dependent on the relative speed of T and R_i, and is defined by

$$f_i = \widehat{f} - v_i/\lambda_t \tag{9}$$

where $\widehat{f} = f_t - f_a$ is the beat frequency, and v_i is the relative speed of T with respect to R_i. The relative speed is the projection of the mobile node velocity onto the unit position vector, $\overrightarrow{R_iT}$, pointing from R_i to T, and can be expressed in the form given by

$$v_i = v_x cos(\alpha_i) + v_y sin(\alpha_i) \tag{10}$$

where α_i is the angle of R_i from the viewpoint of T with respect to the x axis, $v_x = |v|cos\phi$, and $v_y = |v|sin\phi$ (see Figure 19).

Equation (9) allows us to compute the relative speed of the mobile node T and the node R_i if the difference between the two transmitted frequencies, $\widehat{f}$, is known. In practice, we find that the reported Doppler-shifted frequencies differ from the expected frequencies. Because of low-cost

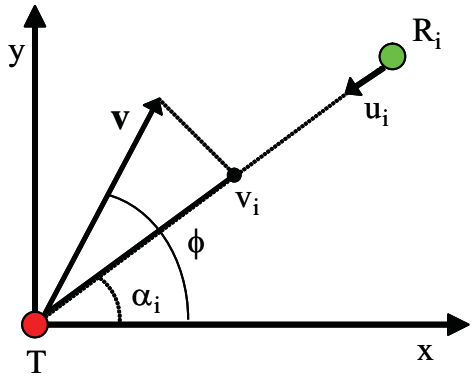


Figure 19: Mobile node T having velocity v transmits a signal. Sensor R_i measures the Doppler shift of the signal, which depends on v_i, the relative speed of T and R_i.

hardware, we are unable to know the exact transmission frequency, which has a tuning resolution of 65 Hz. We must therefore treat $\widehat{f}$ as an unknown variable. The noisy measurement data is therefore passed through an extended Kalman filter in order to arrive at an accurate mobile node position and velocity.

Extended Kalman Filter

The Kalman filter is a widely used technique for estimating the state of a dynamic system based on noisy measurement data. We use the *extended* Kalman filter [95] because we are dealing with the nonlinear mobile node dynamics (Equations (2) - (4)) and the nonlinear relationship between the measured frequency and the relative velocity (from Equation (10)).

The parameters that need to be estimated are the location (x, y), the speed, $|v|$, and heading ϕ of the mobile sensor. In addition, we must either estimate the frequency of the beat signal, or use a method to solve for these parameters that does not require it. We have developed two EKFs for each of these scenarios. The first, which we refer to as the 5-state EKF, contains five state variables, including the unknown beat frequency parameter $\widehat{f}$. The second, which we refer to as the 4-state EKF, does not include $\widehat{f}$.

EKF with Beat Frequency Estimation

We define our parameter vector X for the 5-state EKF as

$$X = \begin{bmatrix} x & y & |v| & \phi & \widehat{f} \end{bmatrix}^{\mathsf{T}}.$$

The parameter vector X is related to an observation vector z. We formalize this relation through a function H such that

$$z = H(X),$$

where the observation z_i is the signal measurement f_i, and the observation vector z is defined as

$$z = \begin{bmatrix} z_1 & z_2 & \dots & z_n \end{bmatrix}^{\mathsf{T}}.$$

The relative speed v_i of the nodes T and R_i is simply the projection of $\overrightarrow{v}$ onto $\overrightarrow{R_iT}$, which can be calculated using the velocity components $|v|$ and ϕ and coordinates (x, y) of the mobile node, and known quantities λ_t and the (x_i, y_i) and (x_{i+1}, y_{i+1}) coordinates of nodes R_i and R_{i+1}, respectively. We define our measurement function (for $i = 1, \dots, n-1$) as

$$H_i(X) = \widehat{f} - \frac{1}{\lambda_t} \frac{|v|((x_i - x)cos\phi + (y_i - y)sin\phi)}{\sqrt{(x_i - x)^2 + (y_i - y)^2}} \tag{11}$$

The EKF linearizes the estimation about the current mobile node state by applying the partial derivatives of the process and measurement functions. These functions take the form

$$X_k = F(X_{k-1}, u_{k-1}) + w_{k-1} \tag{12}$$

$$z_k = H(X_k) + \zeta_k \tag{13}$$

where X is the system state, z is the measurement vector, u is the process input, w is the process noise with covariance Q, ζ is the measurement noise with covariance R, and k is the current timestep. Both the process noise w and the measurement noise ζ are assumed to have "white noise" properties with zero mean, and their covariance matrices are determined experimentally. The state transition function F that governs the dynamics of the mobile node is a vector function and is given by

$$F = \begin{bmatrix} x_{k-1} + \Delta t \frac{r(\omega_r + \omega_l)}{2} cos(\phi_{k-1}) \\ y_{k-1} + \Delta t \frac{r(\omega_r + \omega_l)}{2} sin(\phi_{k-1}) \\ \frac{r(\omega_r + \omega_l)}{2} \\ \phi_{k-1} + \Delta t \frac{r(\omega_r - \omega_l)}{2b} \\ \widehat{f}_{k-1} \end{bmatrix} \tag{14}$$

where Δt is the time elapsed since the last time step. Recall that ω_l and ω_r comprise the process input u.

The EKF recursively estimates the system state in two phases. The first phase predicts the state at the current time step based on the state at the previous time step and the current process input. The second phase adjusts the prediction with actual measurement data obtained during the current time step. In addition, an error covariance matrix, P, is maintained, which is a measure of the accuracy of the estimated state, and is used to update the Kalman gain. Formally, these two phases are represented as

1. Prediction Phase

$$\begin{aligned} \hat{X}_k^- &= F(\hat{X}_{k-1}, u_k) \\ P_k^- &= A_{k-1} P_{k-1} A_{k-1}^T + Q \end{aligned} \tag{15}$$

   where $\hat{X}_k^-$ and P_k^- are the *a priori* state and covariance estimates for the current time step k, and A_{k-1} is the Jacobian of F with respect to X_{k-1}.

2. Update Phase

$$\begin{aligned} K_k &= P_k^- J_k^T (J_k P_k^- J_k^T + R)^{-1} \\ \hat{X}_k &= \hat{X}_k^- + K_k(z_k - H(\hat{X}_k^-)) \\ P_k &= (I - K_k J_k) P_k^- \end{aligned} \tag{16}$$

   where K is the Kalman gain, J is the Jacobian of H with respect to X, R is the covariance of the measurement noise, and I is the identity matrix.

The Jacobian matrices of F and H with respect to X are given below in Equations (17) and (18). Note that for $J_{[i,j]} = \frac{\partial H_{[i]}}{\partial X_{[j]}}$, we only need to calculate the partial derivative of H once for each column, because successive rows correspond to a different receiver node and can be calculated analogously.

$$A_k = \begin{bmatrix} 1 & 0 & 0 & \Delta t \frac{-r(\omega_r + \omega_l)}{2} sin(\phi_k) \\ 0 & 1 & 0 & \Delta t \frac{r(\omega_r + \omega_l)}{2} cos(\phi_k) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{17}$$

$$
\begin{aligned}
\frac{\partial H_i}{\partial x_k} &= -\frac{1}{\lambda}\frac{|v|_k(y_i-y_k)((x_i-x_k)sin\phi_k-(y_i-y_k)cos\phi_k)}{((x_i-x_k)^2+(y_i-y_k)^2)^{3/2}},\\
\frac{\partial H_i}{\partial y_k} &= -\frac{1}{\lambda}\frac{|v|_k(x_i-x_k)((y_i-y_k)cos\phi_k-(x_i-x_k)sin\phi_k)}{((x_i-x_k)^2+(y_i-y_k)^2)^{3/2}},\\
\frac{\partial H_i}{\partial |v|_k} &= -\frac{1}{\lambda}\frac{(x_i-x_k)cos\phi_k+(y_i-y_k)sin\phi_k}{\sqrt{(x_i-x_k)^2+(y_i-y_k)^2}},\\
\frac{\partial H_i}{\partial \phi_k} &= -\frac{1}{\lambda}\frac{|v|_k(-(x_i-x_k)sin\phi_k+(y_i-y_k)cos\phi_k)}{\sqrt{(x_i-x_k)^2+(y_i-y_k)^2}},\\
\frac{\partial H_i}{\partial \widehat{f}_k} &= 1
\end{aligned}
\tag{18}
$$

In [28], which used the 5-state EKF, it was observed that the filter did not respond well to sudden maneuvers of the mobile node, and so it was combined with a constrained nonlinear least squares algorithm. We are able to avoid this step because the mobile node is aware of any maneuvers that it makes, and provides this as input (u) to the EKF.

Extended Kalman Filter without Beat Frequency Estimation

Rather than treating the beat frequency $\widehat{f}$ as an unknown parameter in our localization algorithm, we can take the pairwise difference of frequency measurements, such that $\widehat{f}$ is subtracted out. Doing this provides an additional benefit of reducing the run-time complexity of the algorithm. Hence, we use the pairwise difference of frequency measurements as our observation.

From Equation (9), we see that f_i is a function of $\widehat{f}$, which is an unknown value. We would therefore like to solve X without having to deal with $\widehat{f}$. We can do this by taking the pairwise difference of the frequency measurements. Assuming n infrastructure nodes measure the Doppler-shifted radio signal, this yields $\binom{n}{2}$ pairwise differences. However, we only require $n-1$ pairwise differences, since no additional information can be acquired by using more. Therefore, we use $n-1$ observations z_i. We formally define the observation as

$$
z_i = f_{i+1} - f_i = \left(\widehat{f} - \frac{v_{i+1}}{\lambda_t}\right) - \left(\widehat{f} - \frac{v_i}{\lambda_t}\right) = \frac{v_i - v_{i+1}}{\lambda_t}
$$

and the observation vector z as

$$
z = \begin{bmatrix} z_1 & z_2 & \dots & z_{n-1} \end{bmatrix}^{\mathsf{T}}.
$$

The relative speed v_i of the nodes T and R_i is the projection of $\overrightarrow{v}$ onto $\overrightarrow{R_iT}$. Taking the difference of two such relative speeds, we get

$$
\lambda(f_{i+1} - f_i) = v_i - v_{i+1} = \overrightarrow{v} \cdot (\overrightarrow{u_i} - \overrightarrow{u_{i+1}}).
$$

Finally, $\overrightarrow{v} \cdot (\overrightarrow{R_iT} - \overrightarrow{R_{i+1}T})$ can be calculated using the velocity components $|v|$ and ϕ and coordinates (x, y) of the mobile node, and known quantities λ_t and the (x_i, y_i) and (x_{i+1}, y_{i+1}) coordinates of nodes R_i and R_{i+1}, respectively. Our measurement function (for $i = 1, \ldots, n-1$) is

$$\begin{aligned} H_i(X) &= |v| cos\phi \left(cos\left(tan^{-1}\left(\frac{y_i - y}{x_i - x}\right)\right) - cos\left(tan^{-1}\left(\frac{y_{i+1} - y}{x_{i+1} - x}\right)\right)\right) \\ &- |v| sin\phi \left(sin\left(tan^{-1}\left(\frac{y_i - y}{x_i - x}\right)\right) - sin\left(tan^{-1}\left(\frac{y_{i+1} - y}{x_{i+1} - x}\right)\right)\right). \end{aligned}$$

Experimental Evaluation

Figure 20 shows the sensor devices we use in our experiments. The mobile platform is a MobileRobots Pioneer 3DX [129]. Note that although the Pioneer is equipped with an onboard Linux PC, it was not powered on for these experiments, and all control operations were performed by a connected mote. In addition, the robot has optical encoders on each wheel, however, the measurement data were not made available to the controller at runtime.

Crossbow ExScal motes (XSMs) [30] were used to control the mobile platform, as well as for the dTrack implementation. Six motes were placed in a 20 by 30 meter sensing region (see Figure 21), elevated 1.5 meters from the ground. Five of these were receivers, and one was the assistant transmitter. The position of the assistant is not important, as long as it is stationary and the signal it transmits can be received by the participating receiver nodes. Another mote, the master transmitter, was fixed to the mobile platform, and communicated directly to the platform's microcontroller via a serial connection.

One additional mote was used to host the Kalman filter. Ideally, EKF functionality would be implemented on the same node as the controller. However, due to memory limitations, we made the design decision to use two nodes. We argue that this does not affect the scalability of the system, and with code optimizations it would be possible to implement the EKF on the controller node. The EKF mote was mounted to the mobile platform body and communicated with the controller node over the wireless radio interface. Tuning of the EKF was done experimentally, based on simulation and offline tests.

To obtain ground truth, we used measurements from the onboard optical encoders. In addition, the sensor field was recorded by video. We ran four sets of experiments. The first three experiments used the 5-state EKF, and the fourth used the 4-state EKF.

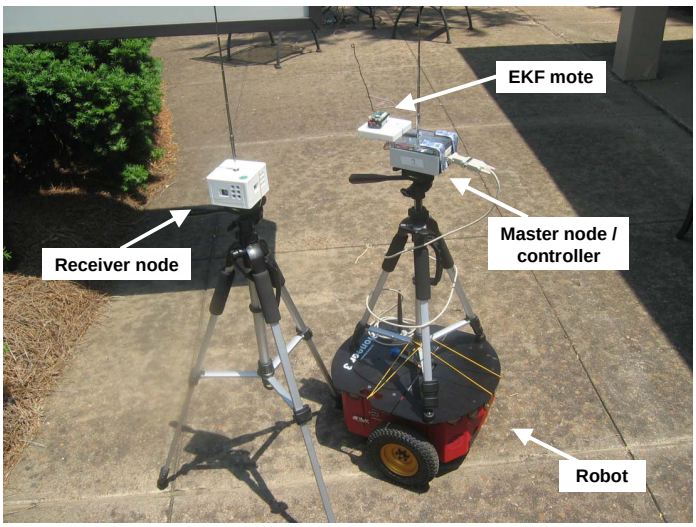


Figure 20: Sensor hardware used in our experiments.

EKF with Beat Frequency Estimation

In Experiment 1, we disabled receiver node feedback, and instructed the mobile sensor to drive in a 30-meter straight line without making corrections for motor inaccuracies or wheel slippage. In Experiment 2, the mobile sensor attempted to control its speed and heading along the same straight-line trajectory based on feedback from the dTrack system. For Experiment 3, we introduced a 90° mid-course maneuver to the mobile sensor trajectory.

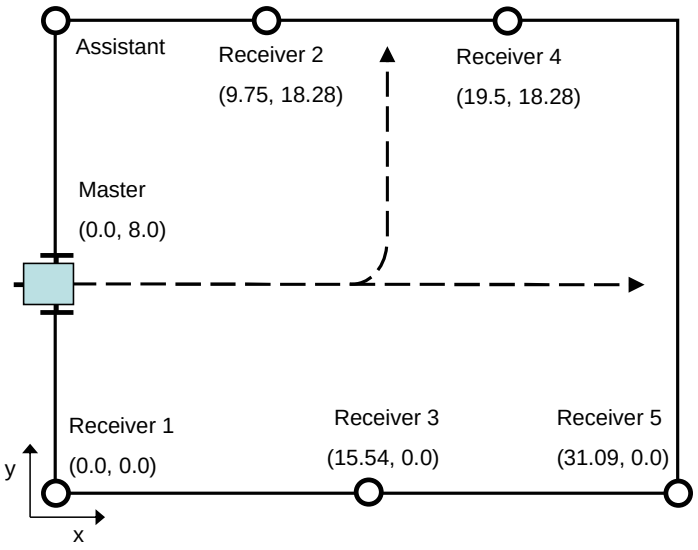


Figure 21: Experimental sensing region. Dashed arrows denote reference trajectories.

Figure 22 illustrates the desired and actual trajectories of the mobile sensor for both the open-loop and closed-loop sets of straight-line experiments, as well as for the 90° mid-course maneuver.

With each feedback cycle, we recorded the current position, speed, and heading, and compared these with the desired values. The average errors are listed in Table 5.

For the open-loop case, the average speed is very close to the desired value, because the translational velocity is set once at the beginning of the run, and the wheel motors are able to maintain a constant angular velocity fairly well. However, wheel slippage and uneven terrain caused the mobile platform to gradually veer off course, which contributed to the large position and heading error. The closed-loop case has much lower position and heading error than the open-loop, however its average speed error is higher. This is because the robot places a higher priority on correcting its heading than it does speed. For the maneuver, rather than having the robot come to a stop, rotate, then resume moving in the new direction at the desired speed, we instructed the robot to maintain its target speed while performing the turn. This resulted in a 1.23 meter overshoot of the desired turning point.

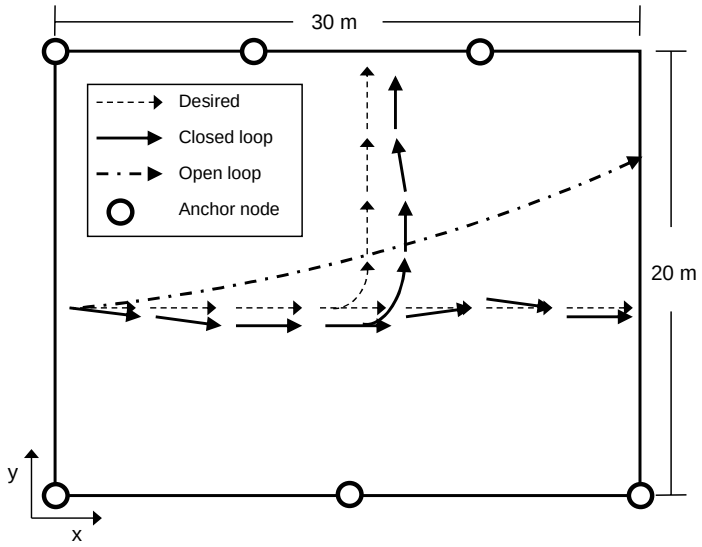


Figure 22: Robot trajectory under both closed-loop and open-loop experiments.

Experiment	Position error (m)	Speed error (m/s)	Heading error (rad)
1 (Open loop)	5.32	0.05	0.167
2 (Closed loop)	1.68	1.31	0.13
3 (Closed loop)	1.93	3.91	0.21

Table 5: Average position, speed, and heading error over 66 measurements for open-loop and closed-loop experiments.

EKF without Beat Frequency Estimation

Our goal for Experiment 4 is for the robot to navigate between a series of waypoints in the sensing region. Because of error in the EKF position estimate, we cannot rely on the robot reaching the exact waypoint coordinates. Therefore we selected an appropriate waypoint proximity region, in which the robot would consider the waypoint reached. An interesting situation arises in which the robot passes through the waypoint region between EKF updates, and is unaware it has done so. To prevent this, we had to make the waypoint region sufficiently large so that at the desired reference speed the robot would not pass over it. For these experiments, if the robot came within 2 meters of the waypoint, it would turn and proceed to the next waypoint.

Figure 23 illustrates the desired and actual paths of the robot during waypoint navigation. With each feedback cycle, we recorded the current position, speed, and heading, and compared these with the desired values. The average position, speed, and heading error was 1.89 m, 0.19 m/s, and 12.05°, respectively.

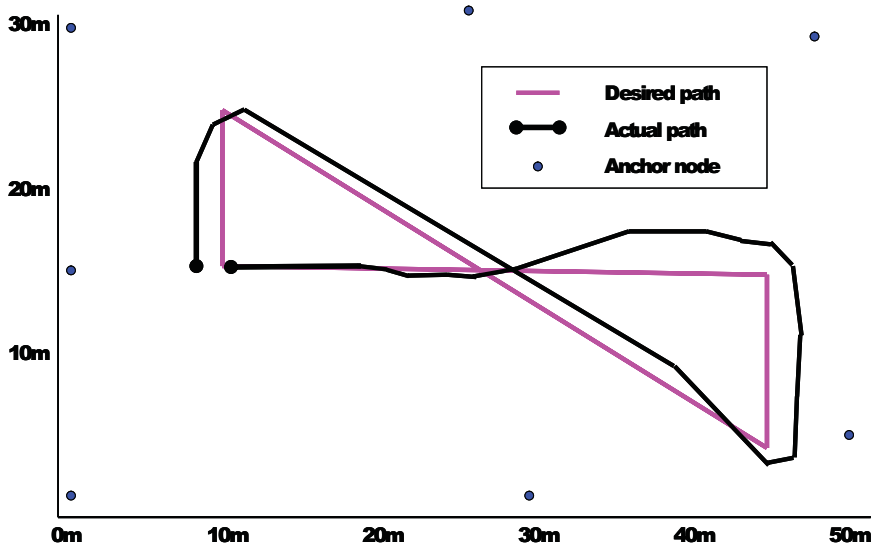


Figure 23: Desired and actual paths of the mobile sensor.

Discussion

By placing radio interferometry and the extended Kalman filter within the control loop, we are able to navigate between waypoints in a sensing region with a maximum position error of 2.51 meters. Factors that contribute to this error include measurement noise, model noise, placement of the assistant transmitter node, and execution time of the control loop. Model noise is primarily due to our approximation of the robot kinematic model, as well as inaccuracies in the process and

measurement error covariances provided to the EKF. In theory, the assistant transmitter can be placed anywhere in the sensing region, as long as its signal is received at all participating nodes. However, in practice, we find that its placement relative to the mobile transmitter does in fact alter the measurement results, because the power of the assistant signal can overwhelm or be overwhelmed by the mobile transmission, which will lead to distorted signal measurements at the receiver.

The benefit of using the 4-state versus the 5-state solution is apparent when considering the memory and latency constraints of the mobile sensor implementation. Table 6 shows the memory requirements and latency of each solution implemented in nesC. Although we did not perform waypoint navigation for our earlier version, and therefore cannot compare our localization results with a 5-state evaluation, we realize a significant memory savings of 208 bytes by only using 4 states. In addition, the latency of the system decreases by 49 ms. This is important, because in order to ensure an accurate controller response, our implementation is required to iteratively run within a small bounded timeframe. Table 7 displays the average and maximum execution times observed over 100 feedback cycles. From these execution times, we see that we can provide feedback to the robot controller at a rate of approximately 1 Hz.

	4-State	**5-State**
EKF Memory Usage	899 bytes	1107 bytes
EKF Latency	216 ms	265 ms

Table 6: Memory requirements and latency for the nesC implementation of the EKF algorithm.

Component	Average (ms)	Maximum (ms)
Radio interferometry	360	361
Frequency calculation	61	116
Routing	266	580
EKF	216	224
PI update	1.3	1.4
Robot control	56	133
Total	**960.3**	**1415.4**

Table 7: Execution time of each component.

Conclusion

We have developed a novel localization and navigation algorithm for mobile wireless sensor networks that utilizes Doppler shifts of the radio signal transmitted by a mobile node. We assume that a number of stationary infrastructure nodes are deployed around the mobile node and that the mobile node cooperates with the localization system. We showed that Doppler shifts can be measured

accurately using radio interferometry, enabling the infrastructure nodes to accurately determine the relative speed of the mobile node using low-cost hardware.

In implementing dNav, we have made two significant contributions. First, we show that this system can be implemented on mote hardware and that it does not require any PC processing. This is not a trivial task due to the limited memory and processing power available on the motes, as well as the intricacies of the system integration between the mote and the mobile platform. Second, we show that our system is capable of rapid localization, which enables a mobile node to accurately navigate through a sensing region. Accurate navigation requires the latency between feedback measurements to be small, which in turn requires tight synchronization between all components within the control loop.

CHAPTER V

DETERMINING ANGULAR SEPARATION IN MOBILE WIRELESS SENSOR NETWORKS

Introduction

In this chapter, we present a method for determining angular separation that only requires the sensor radio and wheel encoders, both of which are common to wireless mobile actuated sensors, and hence no additional hardware is required. Our method uses the Doppler shift in radio frequency and the instantaneous velocity of a mobile node transmitting a pure sinusoidal signal to derive the angular separation between infrastructure nodes surrounding the sensing region. Our method does not require the positions of the infrastructure nodes, or the initial position of the mobile node, to be known.

Arguably one of the biggest challenges for mobile sensors is navigation, where the mobile node must reach point B from point A. For the most basic mobile entities, navigation is typically solved using odometry, whereby the mobile device monitors the angular velocity of each wheel to approximate the distance traveled over a given time period. The angular velocity is often determined from feedback from optical encoders mounted on each wheel. The advantage of optical encoders is that they are small and can be mounted on almost any type of mobile platform. Most other sensors used for navigation (e.g., GPS, laser rangefinders, sonar, etc.) are either too large, heavy, expensive, or require too much power to operate over extended periods of time. When operating on a clean, level surface, optical encoders can be quite accurate. However, most environments contain dust that can interfere with the encoder readings. Additionally, odometry rapidly accrues error on uneven terrain due to wheel slippage and low tire pressure.

Recent work has shown that navigation is possible without knowing the current position of the mobile sensor [130]. In fact, all points in a plane are reachable if the angular separation between two pairs of beacons can be determined [66]. In many situations, navigating without having to determine position is advantageous because most localization methods require extensive PC processing, have high localization latency, and require the positions of infrastructure nodes to be known. In dNav (see Chapter IV), we used the Doppler shift in radio transmission frequency as control feedback to drive a mobile sensor. Although the mobile sensor was able to accurately navigate the sensing region, the localization algorithm relied on the use of an extended Kalman filter for noisy frequency

measurements. The EKF was run in realtime on the mobile platform, but its large size required execution on a separate mote, which communicated with the mobile node over a wireless interface. By eliminating the EKF, we will have a more compact and less time-consuming localization / navigation algorithm. The solution we present here is to only determine the angular separation between pairs of beacons, rather than the position and velocity of the mobile node. Knowledge of the angular separation between pairs of anchor nodes provides a means for angle-based navigation without localization [66], [130]. We show using real-world experimental results and in simulation that this method is accurate with an average angular separation error of 0.28 radian.

The contributions of this work are as follows [20]:

1. We develop a method for determining angular separation that only requires the sensor radio and wheel encoders, both of which are common to wireless mobile actuated sensors, and hence no additional hardware is required. Our method does not require the positions of the infrastructure nodes or the initial position of the mobile node to be known.

2. Because this method is intended for use with resource-constrained mobile sensors, it is rapid and can be implemented on mote-class devices.

3. We show using real-world experimental results and in simulation that this method is accurate with a moderate angular separation error.

System Overview

Consider a sensing region that contains multiple infrastructure nodes, as well as a mobile sensor that needs to travel from point A to point B. This scenario is illustrated in Figure 24.

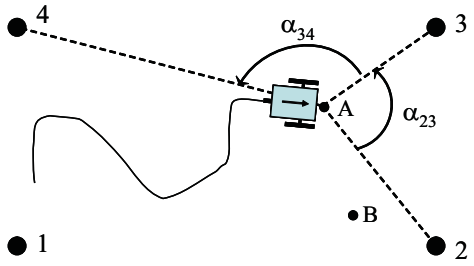


Figure 24: A mobile sensor moves through the sensing region. The node navigates based on angular separation of beacons (numbered 1 through 4).

In order to navigate toward point B, we need to know which direction to drive in, and for that we need to have some idea of the spatial relationship between the current position of the mobile node

(A) and the goal position (B). Determining angular separation between pairs of beacon nodes will provide us with such a spatial relationship. Often, angle information is determined using cameras, microphone arrays, or light pulses, all of which are not ideal for lightweight mobile sensors. We would like to estimate angular separation using only hardware that is widely available on sensor nodes. Specifically, we obtain this angle information using the sensor node radio and the optical encoders on the wheels.

The system works as follows. A mobile node, T, moving through the sensing region with velocity $\mathbf{v}$, collects angular velocity data from its wheel encoders. For mobile platforms with 2-wheel differential steering, the relationship between the translational speed and the wheel angular velocities is

$$|v| = \frac{r(\omega_r + \omega_l)}{2} \tag{19}$$

where the speed $|v|$ is the magnitude of the velocity $\mathbf{v}$, r is the wheel radius, and ω_r and ω_l are the right and left wheel angular velocities, respectively.

As the mobile node moves, it transmits an RF sinusoidal signal, which is observed by the receiver nodes. Because the mobile node is moving with respect to the stationary receivers, the RF signal will be Doppler-shifted. The amount of Doppler shift depends on the relative speed of the mobile and anchor nodes, as well as the wavelength and carrier frequency of the signal. This relationship is formalized as

$$f_i = f_{carrier} - \frac{v_i}{\lambda} \tag{20}$$

where f_i is the observed Doppler-shifted frequency at receiver R_i, $f_{carrier}$ is the transmission frequency of the carrier signal with wavelength λ, and v_i is the relative speed of mobile node T with respect to receiver R_i.

Figure 25 illustrates the geometry of a simplified setup. For now we will only consider two receiver nodes, R_i and R_j. The problem is to estimate the angular separation α_{ij} between the two receiver nodes based on the measured values of ω_r, ω_l, f_i, and f_j, and known values $f_{carrier}$ and λ.

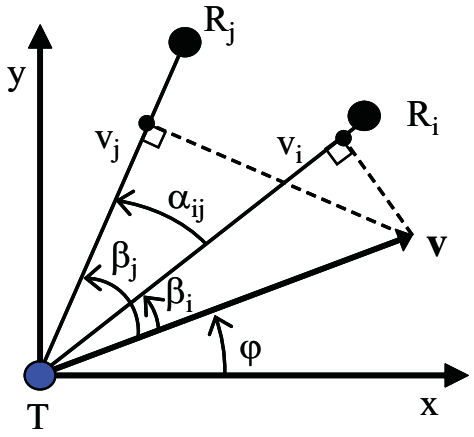


Figure 25: Geometry of simplified setup for determining angular separation between two receivers.

The relative speed, v_i, between the mobile node and receiver R_i is the scalar value resulting from the projection of $\mathbf{v}$ onto the position vector $\overrightarrow{TR_i}$, as

$$v_i = |v| cos\beta_i \tag{21}$$

where the speed of the mobile node, $|v|$, has a negative sign if T is moving toward R_i and positive sign otherwise, and β_i is the angle between the velocity vector $\mathbf{v}$ and the position vector $\overrightarrow{TR_i}$.

The relative speed is related to the received Doppler-shifted signal. By rearranging Equation (20), we have

$$v_i = \lambda(f_{carrier} - f_i). \tag{22}$$

Combining Equations (21) and (22), and rearranging, we can calculate β_i by

$$\beta_i = cos^{-1}\left(\frac{\lambda(f_{carrier} - f_i)}{|v|}\right). \tag{23}$$

Angular separation between two receiver nodes R_i and R_j can then be computed by subtracting one bearing from the other, as

$$\alpha_{ij} = \beta_j - \beta_i. \tag{24}$$

Frequency Estimation using Resource Constrained Hardware

Typical low-cost sensor hardware supports radios that transmit in the 400 MHz — 2.4 GHz range. These radios have a received signal strength indicator (RSSI) pin that can be accessed from software; however, we cannot sample the RSSI fast enough to determine the carrier frequency of the signal. Again, we use radio interferometry, in which a second node transmits a signal at a slightly lower frequency such that the two transmitted signals interfere, creating a low-frequency beat signal. The assistant transmitter can be positioned anywhere in or near the sensing region, as long as it is stationary, and its signal can reach all receiver nodes. The beat signal, which can be as low as a few hundred Hertz (350 Hz in our case), is sampled by making successive reads of the RSSI.

Another issue with the inexpensive radio chip is that the transmission frequency can differ from the nominal frequency by up to 65 Hz due to the tuning precision of the radio chip [96]. For this reason, we treat the transmission frequency as a random variable, which results in the beat frequency being a random variable as well. This poses a challenge, because we require knowledge of the beat frequency to compute the receiver bearings. Therefore, in order to determine receiver bearing, we use maximum likelihood (ML) estimation [89].

For ML estimation, we rewrite Equation (23) as

$$\begin{aligned} f_i &= F(\beta_i, f_{beat}) + \epsilon_i \\ &= f_{beat} - \frac{|v|}{\lambda} \cos \beta_i + \epsilon_i \end{aligned}$$

where $\epsilon_i \sim \mathcal{N}(0, \sigma_f)$ is the Gaussian noise in the observed Doppler-shifted frequency. The negative log-likelihood for f_i is given by

$$\ell_i(f_{beat}, \beta_i) = -\ln p(f_i | f_{beat}, \beta_i) = \frac{\parallel f_i - F(\beta_i, f_{beat}) \parallel^2}{\sigma_f^2}.$$

Assuming N receivers, the combined negative log-likelihood for $f_i, i = 1, \cdots, N$ is given by

$$\begin{aligned}
\ell(f_{beat}, \beta_1, \cdots, \beta_N) &= -\ln p(f_1, \cdots, f_N | f_{beat}, \beta_1, \cdots, \beta_N) \\
&= -\ln \prod_{i=1}^{N} p(f_i | f_{beat}, \beta_i) \\
&= \sum_{i=1}^{N} \ell_i(f_{beat}, \beta_i) \\
&= \sum_{i=1}^{N} \frac{\| f_i - F(\beta_i, f_{beat}) \|^2}{\sigma_f^2}.
\end{aligned}$$

The ML estimate can be obtained by minimizing the negative log-likelihood using the following

$$\frac{\partial \ell(f_{beat}, \beta_1, \cdots, \beta_N)}{\partial f_{beat}} = 0.$$

The partial derivative leads to the following result for the ML estimate for the beat frequency

$$\widehat{f}_{beat} = \frac{1}{N} \sum_{i=1}^{N} f_i + \frac{|v|}{\lambda N} \sum_{i=1}^{N} \cos \beta_i.$$

Note that the ML estimate, $\widehat{f}_{beat}$, is in terms of $\beta_i, i = 1, \cdots, N$. To solve for the angles, we iteratively compute the ML estimate and the angles. The two iterative steps are given below.

1. Computing the angles:
$$\beta_{i,k} = cos^{-1} \left(\frac{\lambda(\widehat{f}_{beat_{k-1}} - f_i)}{|v|} \right)$$

2. Computing the ML estimate for the beat frequency:
$$\widehat{f}_{beat_k} = \frac{1}{N} \sum_{i=1}^{N} f_i + \frac{|v|}{\lambda N} \sum_{i=1}^{N} \cos \beta_{i,k}$$

where $k = 1, \cdots, 10$ is the iteration index, and the ML estimate is initialized with the average of the observed Doppler-shifted frequencies, $\widehat{f}_{beat_0} = \frac{1}{N} \sum_{i=1}^{N} f_i$.

We show the convergence results for the beat frequency in Figure 26. We observed that the beat frequency estimate converges within a small number of iterations, hence we conservatively chose 10 iterations for the iterative algorithm. A theoretical analysis of convergence of the algorithm is beyond the scope of this work.

One obvious drawback with such a system is that it requires knowledge of the current speed of the mobile node. We argue that such information can easily be obtained by using optical encoders.

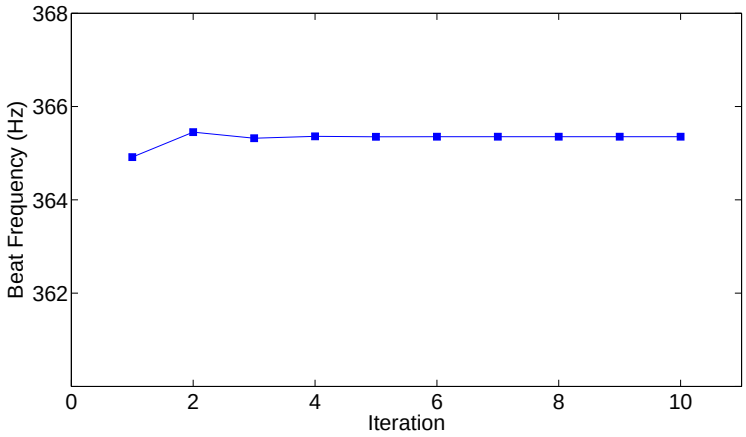


Figure 26: Convergence results for beat frequency estimate with the maximum likelihood estimation algorithm.

Such sensors are not always adequate on their own for dead reckoning because they accrue error over time. However, encoders are inexpensive, and provide good approximations of *instantaneous velocity*, which make them suitable for this type of system.

Implementation

Our mobile wireless sensor platform consists of ExScal motes (XSMs) [30] and a Mobile-Robots Pioneer 3DX [129] robot. All code was written in nesC [131] for the TinyOS operating system [128]. The XSMs use the Texas Instruments CC1000 radio chip [96], and transmit in the 433 MHz band. Note that although the Pioneer comes equipped with an onboard embedded PC, as well as a wide variety of sensors, only the instantaneous velocity, obtained from encoder data, is used, and all computation is performed on the attached mote.

Implementation Benchmarking

Mobile sensors require a rapid positioning algorithm, otherwise by the time the algorithm completes, the mobile node may be in a completely different location. We therefore provide a timing analysis of our algorithm implementation to demonstrate that its latency is acceptable for mobile sensor navigation. Our method for determining angular separation involves three major steps: (1) signal transmission/reception, (2) sending observed frequencies from the infrastructure nodes to the mobile node, and (3) running the angular separation estimation algorithm. We list the average and maximum latencies for these steps in Table 8. The most unpredictable of these steps is the time it takes the infrastructure nodes to send their observed frequencies to the mobile node. This latency

can grow relatively large because the nodes are all attempting to send messages at roughly the same time, resulting in back-off delays. However, even with this unpredictability, we can, on average, obtain angular separation information at a rate of 1.46 Hz, which is sufficient for mobile sensor navigation.

Step	Average (ms)	Maximum (ms)
Signal transmission	415	417
Routing	242	561
Angular separation algorithm	28	46
Total	**685**	**1024**

Table 8: Execution time for each step.

Because we are using resource-constrained sensor nodes, we are interested in minimizing the memory required to run the algorithm. dNav [18] required the use of two motes on the mobile platform, one hosting the controller, and the other hosting the EKF, leaving little space for the user application. Our current approach requires significantly less memory, using 2.9 kB of RAM and 49.6 kB of program memory (ROM).

Experimental Evaluation

Experimental Setup

Our setup consists of six XSM nodes, four of which act as stationary receivers and surround the sensing region. Another stationary node is designated the assistant transmitter, and is placed just outside the sensing region. The final mote is attached to the mobile platform. The mobile node moves around an uneven paved surface in an outdoor environment, mostly free of trees, buildings, and other obstacles. Figure 27 illustrates the experimental setup.

We direct the mobile node to move through the sensing region while transmitting a pure sinusoidal signal. The infrastructure nodes measure the frequency of the signal and report their observations back to the mobile node. At the beginning of each measurement round, the mobile node records its instantaneous velocity, obtained from the wheel encoders. This information is then used to derive the angular separation of the infrastructure nodes. Ground truth is manually measured at each timestep (the time at the beginning of each measurement).

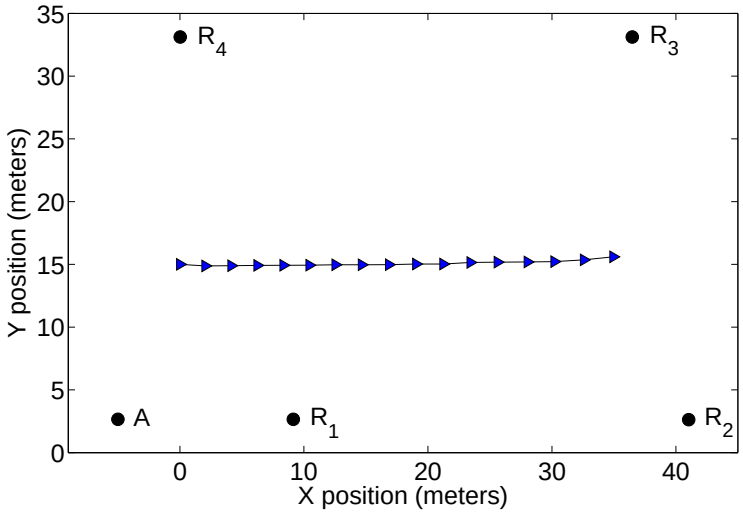


Figure 27: Experimental setup. Four infrastructure nodes ($R_1 \ldots R_4$) and the assistant transmitter (A) surround the sensing region. Triangles show the direction of travel of the mobile node at each timestep.

Experimental Results

Figure 28 shows the estimated versus ground truth angular separations for all pairs of adjacent beacons over the entire course. At present, this technique results in a moderately large error of 0.28 radian. We investigate the main sources of this error next.

Error Analysis

We have identified five main sources of error in this system. In order to understand the effect each source of error has on the performance of the system, we use a simulated experimental setup that consists of a single mobile sensor and single anchor node. Because the relationship between observed frequency and bearing is dependent on relative speed between the two nodes (see Equation (23)), we perform repeated simulations, rotating the anchor node around the mobile node at 1° increments. The mobile node is considered to be moving at a speed of 1 m/s with an orientation of zero degrees, and the distance between the mobile node and anchor is 10 meters. Figure 29 illustrates this setup.

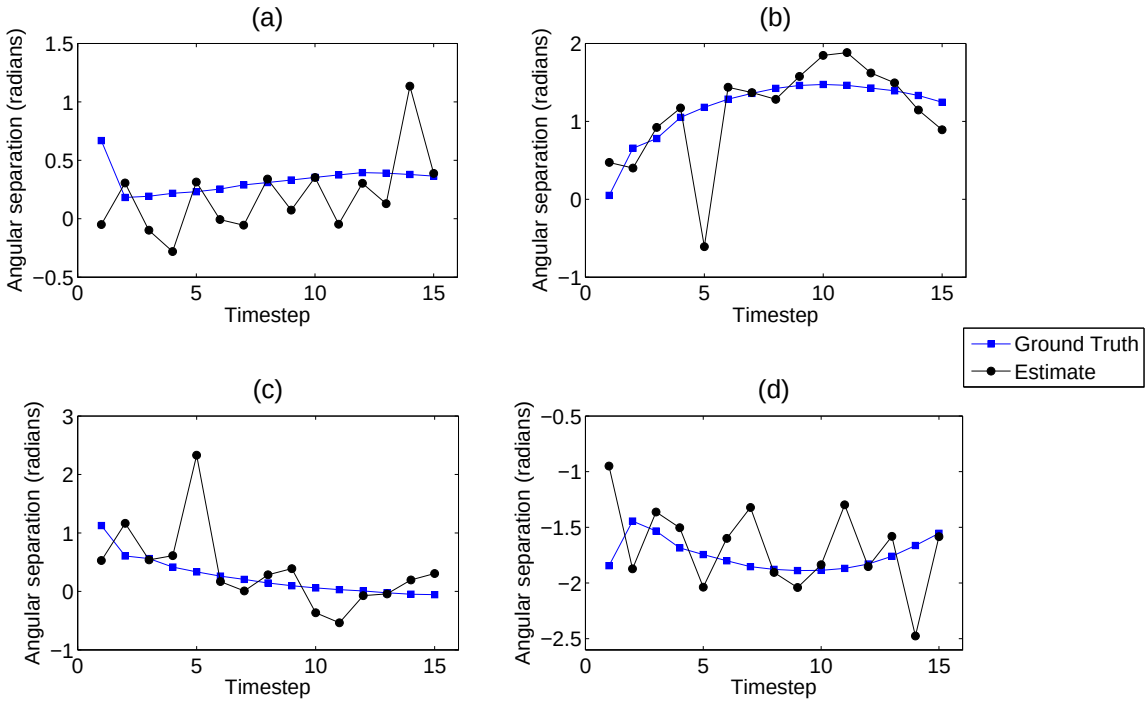


Figure 28: Ground truth versus estimated angular separation between receiver nodes (a) 1 and 2, (b) 2 and 3, (c) 3 and 4, and (d) 4 and 1, for each measurement as the mobile node traverses the sensing region.

Nonlinearity of the Bearing Estimation Equation

The relationship between bearing and observed frequency is

$$\beta_i \quad = \quad cos^{-1}\left(\frac{\lambda(f_{beat} - f_i)}{|v|}\right) \tag{25}$$

where β_i is the bearing to the mobile node from anchor node i, f_{beat} is the frequency of the beat signal, f_i is the Doppler-shifted frequency observed at anchor node i, and $|v|$ is the speed of the mobile node.

However, the error in computing the bearing β will vary due to the nonlinearity of Equation (25). Figure 30a shows the structure of the inverse cosine function ($y = cos^{-1}(x)$), and its derivative is pictured in Figure 30b. We can see that in general a small error in x will result in a large error in $cos^{-1}(x)$. This is especially true at the limits ($-0.8 \geq x \geq 0.8$). To avoid this problem, we can examine the argument to the inverse cosine, and if too large or small, discard the sensor data for the current measurement round. In practice, we found this gives us a marginal error reduction of approximately 11%, or 2°.

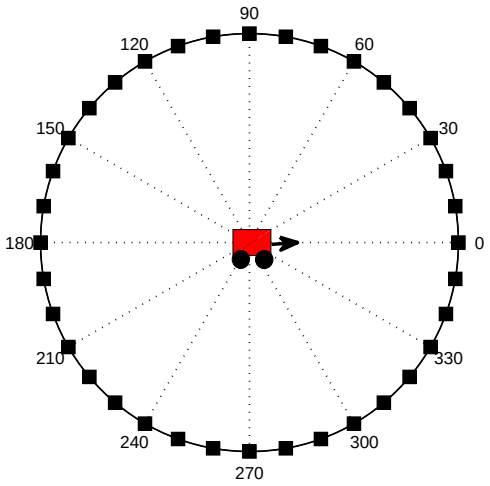


Figure 29: Simulation setup. Anchor nodes are placed at 1° increments (some nodes not pictured for clarity).

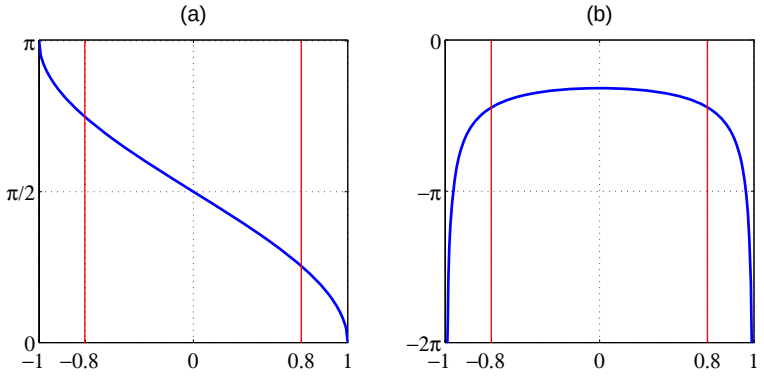


Figure 30: (a) The inverse cosine function and (b) its derivative.

Figure 31 is a polar plot of the expected arccosine argument under ideal conditions. We can see that the arccosine argument will exceed the bounds illustrated in Figure 30 when the anchor node is located between 325° and 35° and 145° and 215°. In light of this, for the remainder of this error analysis, we will discard all samples that fall within these ranges. Under non-ideal conditions (when there is noise in the system), the range of unusable anchor node measurements will increase. Therefore, we must deploy a redundant number of anchor nodes around the sensing region to ensure a sufficient amount of acceptable measurements at bearings that do not fall within these undesirable ranges.

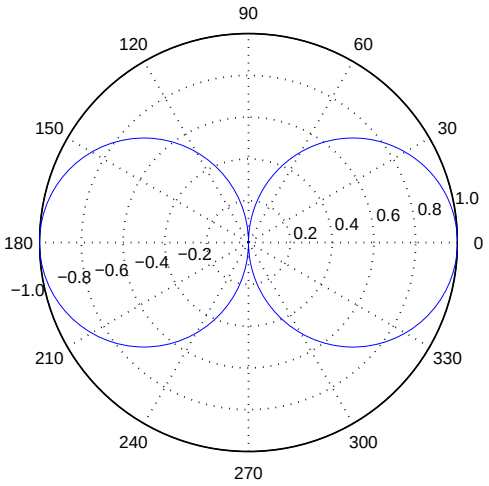


Figure 31: Expected arccosine argument under ideal conditions.

Measurement Noise

It was reported in [28] that the standard deviation of measured frequency is 0.21 Hz. Using this metric, we generate a dataset of 1000 samples for each anchor position. The dataset has a mean of the expected frequency and standard deviation of 0.21 Hz. For each anchor position, we would like to know the average error due to measurement noise we expect to see in the system. On average, the bearing error due to measurement noise is 6.9°. Figure 32 plots the error.

Noisy Encoder Data

Typical optical encoders have an instantaneous velocity error distribution with a standard deviation of approximately 1% of the speed. We examine how encoder error affects the bearing estimate. The results are displayed in Figure 33. On average, the bearing error due to encoder noise is 1.32°.

Unknown Beat Frequency

Because we are using low-cost hardware, one problem we encounter is that we are unable to know the exact beat frequency, f_{beat}. Although we instruct the nodes to transmit at specific frequencies, the actual transmission can differ from the nominal value by as much as 2 kHz. Because ultimately it is the beat frequency that is important, we can tune the transmitters to transmit at frequencies such that the interference signal (as measured by a participating receiver node) will have a beat envelope at the desired frequency. However, the radio hardware has a tuning resolution of 65 Hz,

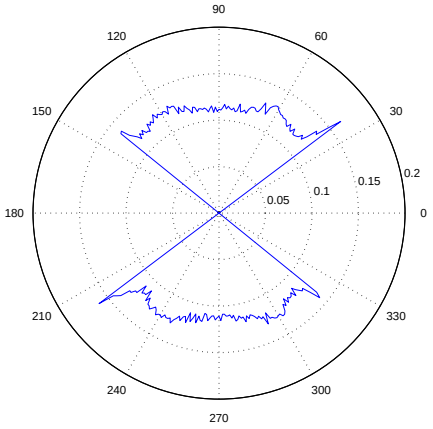


Figure 32: Bearing error due to measurement noise.

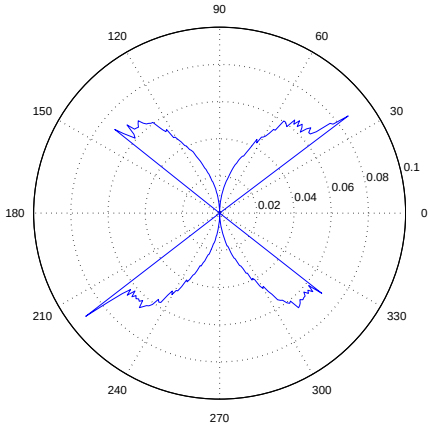


Figure 33: Bearing error due to wheel encoder noise.

so the actual beat frequency may differ from the desired frequency by up to 65 Hz. In addition, the transmission frequencies will drift from their tuned frequencies over time due to environmental causes such as temperature, humidity, and supply voltage, as well as imprecision in the radio crystal. We see the effects of a variable f_{beat} in Figure 34, in which we plot the beat frequency observed by a stationary receiver node over 100 successive measurements, 10 seconds apart. The figure illustrates the degree to which we are unable to estimate the beat frequency.

Because we do not know the beat frequency, we must solve Equation (25) using maximum likelihood estimation, where f_{beat} is the unknown parameter. The ML estimation introduces error into the bearing estimate. Because ML estimation approximates the unknown beat frequency by considering measurements made at all participating receivers, we cannot determine ML estimation error by simply rotating a single anchor receiver around the mobile node. Therefore, we must fix the positions of several receivers. We do this based on the error analysis we have performed so far. From the above analysis, we can see the best place to take measurements is by receivers on the periphery of the mobile node. We therefore select four anchor nodes at random, two on each side of the mobile node. We repeat this process 1000 times and determine the average bearing error. For each iteration, we generate a random beat frequency between 300 and 400 Hz. The error distribution is shown in Figure 35. The average error is 4.85°.

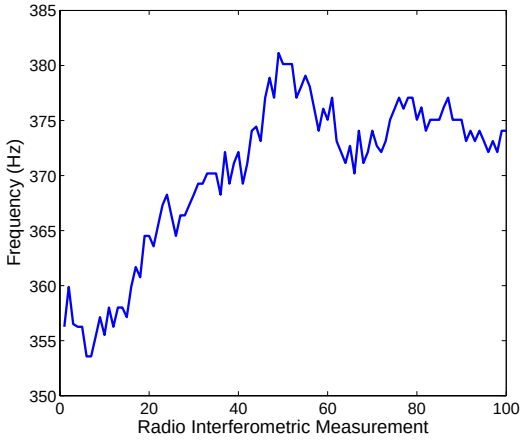


Figure 34: Frequency variation over 100 successive transmissions 10 seconds apart.

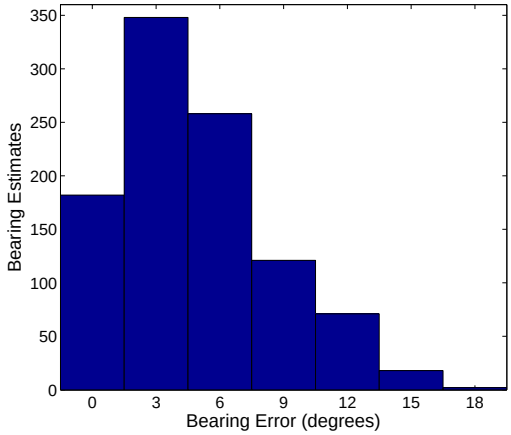


Figure 35: Bearing error due to maximum likelihood estimation.

System Latency

Mobile sensors require a rapid positioning algorithm, otherwise by the time the algorithm completes, the mobile node may be in a completely different location. We therefore provide a timing analysis of our algorithm implementation to demonstrate that its latency is acceptable for mobile sensor navigation. Our method for determining angular separation involves three major steps: (1) signal transmission/reception, (2) sending observed frequencies from the infrastructure nodes to the mobile node, and (3) running the angular separation estimation algorithm. We list the average latencies for these steps in Table 9.

Step	Average (ms)
Signal transmission	415
Routing	242
Angular separation algorithm	28
Total	**685**

Table 9: Average execution time for each step.

Based on the fixed speed of the robot at 1 m/s, we can determine the bearing error caused by algorithm latency. In 685 ms, the position of the mobile node will have changed by 0.685 m. The bearing error caused by this position change is 1.98° and is plotted in Figure 36.

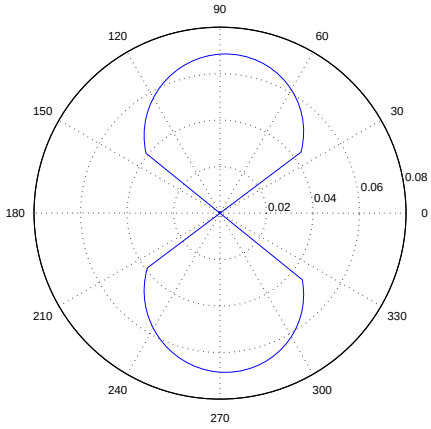


Figure 36: Bearing error due to system latency.

Conclusion

In this chapter we presented a method for determining angular separation using RF Doppler shifts and wheel encoder data in mobile sensor networks. Angular separation between multiple pairs of beacons can be used for navigation, without the need for localization.

Several implementation challenges were encountered while designing this system. Measuring Doppler-shifted frequencies required the use of radio interferometry. Radio hardware limitations caused the actual transmission frequency to be unknown. Because knowledge of the beat frequency was necessary for our algorithm, we used maximum likelihood estimation. Experimental results obtained using our method had an average error of 0.28 radian, which will provide course-grained navigation. However, in situations where such navigation is acceptable, our approach is fast, and requires less memory than other RF-based methods (e.g., [28], [53], [18]). This is because our algorithm is distributed, and therefore we expend no time routing data to a base station for analysis. In addition, determining angular separation from Doppler shifts and instantaneous velocity does not require complex statistical tools, such as a Kalman filter, reducing the overall memory footprint of the application.

CHAPTER VI

TRIPLOC: A RAPID DISTRIBUTED LOCALIZATION SERVICE

Introduction

In this chapter, we describe a novel method that uses radio interferometry to rapidly estimate the position of a mobile node. Radio interferometric positioning for mote-scale devices was first presented in [22], however, RIPS is not appropriate for many types of WSN applications. For instance, RIPS sensor nodes are used to measure the phase of the transmitted signal, but the localization algorithm runs on a resource-rich PC-class device. Although many WSN applications are centralized, it is often more desirable to run in a distributed manner, especially when mobility is involved. This will eliminate the possibility of a single point of failure, and the mobile node will not be in danger of moving out of range of the base station.

The basic idea of our technique, which we refer to as TripLoc, is to group together three of the four nodes involved in a typical radio interferometric measurement to form an antenna array, which acts as an anchor node. Two transmitters and one receiver are arranged in such a manner that their antennas are mutually orthogonal to minimize parasitic antenna effects (see Figure 37.) The measured phase difference between the receiver in the array and a target node constrains the location of the latter to a hyperbola. The bearing of the target node is estimated by computing the angle of the hyperbola asymptote, assuming the target node is not too close to the array. With a sufficient number of anchors, triangulation can then be performed to determine node position.

Although several techniques exist for determining node position based on bearing information [67], [83], [132], [133], [134], there are few options for actually measuring signal AOA in WSNs. Currently available methods for bearing estimation require a heavy-weight infrastructure [135], rotating hardware [78], [64], directional antennas [136], and/or expensive and sophisticated sensors [68]. Furthermore, such techniques typically require participating nodes to be stationary for extended periods of time. These constraints are often undesirable for WSN deployments, in which node size and cost must be kept to a minimum. An AOA approach that does not require additional hardware, runs on the nodes themselves, and is fast enough to support tracking in addition to static localization would be a major step forward.

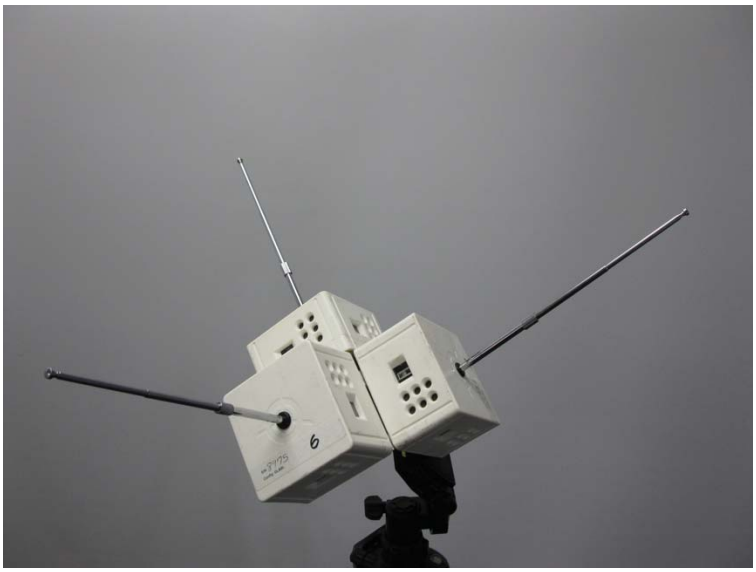


Figure 37: TripLoc antenna array implementation using three XSM motes. The name TripLoc comes from the fact that a triplet of nodes constitute an anchor node array in our localization scheme.

We present several new contributions for AOA-based RF localization in wireless sensor networks [21].

1. We describe an RF-based technique for determining node bearing and position.

2. We provide a detailed analysis that shows our bearing estimation and localization technique is robust to measurement noise and approximation error.

3. We design a real-world implementation using COTS sensor nodes, in which bearing estimation and triangulation is performed entirely on the resource-constrained motes without hardware support.

4. We present experimental results that show our approach can rapidly and accurately estimate node bearing and position.

System Overview

Radio Interferometric Measurements

Our system consists of stationary antenna arrays with cooperating wireless sensor nodes at unknown positions. The array contains three nodes, a master (M) and two assistants (A_1, A_2), as shown in Figure 38. We assume that the position of the midpoint of the array is known, as well as the distance between the antennas in the array. At a predetermined time, the master, M, and

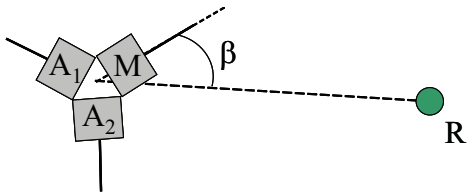


Figure 38: Array containing a master node (M) and two assistant nodes (A_1, A_2). A target node (R) computes its bearing (β) from the array.

one of the assistants, A_1, transmit a pure sinusoidal signal at slightly different frequencies, which interfere to create a low-frequency beat signal whose phase is measured by the other assistant in the array, A_2, and a receiver node, R, at an unknown position. Such a measurement is termed a radio interferometric measurement (RIM).

The difference in phase, $\Delta\varphi = \varphi_R - \varphi_{A_2}$, measured by receiver nodes R and A_2 is a linear combination of the distances between the transmitters and receivers,

$$\Delta\varphi \quad = \quad \frac{2\pi}{\lambda}(d_{MA_2} - d_{A_1A_2} + d_{A_1R} - d_{MR}) \ (mod\ 2\pi),$$

where λ is the wavelength of the carrier frequency, d_{MR} is the distance between the master node and target receiver node, d_{A_1R} is the distance between the assistant transmitter and the target receiver node, and d_{MA_1}, d_{MA_2}, and $d_{A_1A_2}$ are the respective distances between all pairs of nodes in the array. Note that the nodes in the array are equidistant from each other, and therefore $d_{MA_2} - d_{A_1A_2} = 0$, so the phase difference can be simplified:

$$\Delta\varphi \quad = \quad \frac{2\pi}{\lambda}(d_{A_1R} - d_{MR}) \ (mod\ 2\pi). \tag{26}$$

We denote the distance difference $d_{A_1R} - d_{MR}$ by d_{A_1MR} and refer to it as a *t-range*. From Equation (26), we see that if $-\frac{\lambda}{2} < d_{A_1MR} < \frac{\lambda}{2}$, the phase difference will fall in the interval $(-\pi, \pi)$. When this is *not* the case, the possible range of $\Delta\varphi$ will exceed 2π, which results in a modulo 2π phase ambiguity. To avoid this ambiguity, we would like the maximum possible distance difference to be less than $\frac{\lambda}{2}$. The maximum distance difference will occur when the receiver node is collinear with the transmitters M and A_1. d_{A_1MR} then corresponds to the distance between the master and assistant. Therefore, to eliminate the modulo 2π phase ambiguity, we require the distance between antennas in the array to be less than half the wavelength of the carrier frequency.

Having removed the modulo operation, we can rearrange Equation (26) so that known values are on the right hand side.

$$d_{A_1MR} = \frac{\Delta\varphi\lambda}{2\pi} \tag{27}$$

The t-range d_{A_1MR} defines an arm of a hyperbola that intersects the position of node R, and whose asymptote passes through the midpoint of the line $\overline{A_1M}$, connecting the master and assistant nodes. Figure 39 illustrates such a hyperbola with foci A_1 and M. The absolute value of the distance differences between the foci and any point on a hyperbolic arm is constant, formally defined as

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$$

where (x, y) are the coordinates of a point on the hyperbola, a is the distance between the hyperbola center and the intersection H of the hyperbola with the axis connecting the two foci, and b is the length of the line segment, perpendicular to the axis connecting the foci, that extends from H to the asymptote.

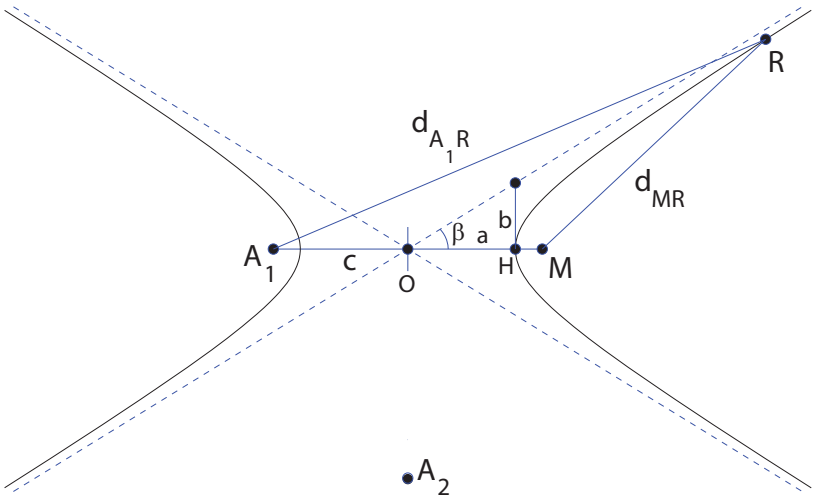


Figure 39: The t-range defines a hyperbola that intersects node R, and whose asymptote passes through the midpoint of the two transmitters in the array.

Bearing Approximation

The hyperbola in Figure 39 is centered at O, and the distance between O and either focus is denoted by c. Furthermore, it can be shown that $c^2 = a^2 + b^2$ [137]. From the figure, we see that the bearing

of the asymptote is $\beta = tan^{-1}(\frac{b}{a})$. Therefore, in order to solve for β, we must determine the values of b and a.

We can solve for a by observing that

$$d_{A_1R} - d_{MR} = d_{A_1H} - d_{MH}$$

because, by definition, the distance differences between the foci and all points on the hyperbola are constant. From Figure 39, we see that we can substitute $(c+a)$ for d_{A_1H} and $(c-a)$ for d_{MH}, and therefore

$$d_{A_1R} - d_{MR} = (c+a) - (c-a) = 2a.$$

From Equation (27), we know the value of $d_{A_1R} - d_{MR}$, which is the t-range, and therefore $a = \frac{d_{A_1MR}}{2}$. We can then solve for b, using $b = \sqrt{c^2 - a^2}$. In terms of known distances, the bearing of the asymptote is defined as

$$\beta = tan^{-1}\left(\frac{\sqrt{\left(\frac{d_{A_1M}}{2}\right)^2 - \left(\frac{d_{A_1MR}}{2}\right)^2}}{\left(\frac{d_{A_1MR}}{2}\right)}\right). \tag{28}$$

In Figure 39, we see the case where $d_{A_1MR} > 0$, and the position of R lies on the right arm of the hyperbola. If the phase difference is negative (i.e., $\varphi_R < \varphi_{A_2}$) then the position of R will lie on the left arm of the hyperbola. When this is the case, β is taken clockwise, and we must adjust it by subtracting it from π.

The line $\overline{A_1M}$ connecting the two foci is called the transverse axis of the hyperbola, and is a line of symmetry. This implies that although we know b, we do not know its sign, because mirrored positions on either side of the transverse axis will result in the same d_{A_1MR}. Therefore, the asymptote bearing β we obtain using this method could be either positive or negative. To find which bearing is correct, we can switch the roles of the assistant nodes in the array and perform another RIM. This will generate a different t-range, and hence another hyperbolic arm with foci A_2 and M.

Each hyperbola provides us with two angles $\pm\beta_i$, where β_i is the angle of the asymptote with the transverse axis, $\overline{A_iM}$. Of course, these angles will be offset from the global x-axis, because the orientation of $\overline{A_iM}$ may not be 0. Adjusting for this, one of the β_1 bearings, and one of the β_2 bearings will point in the same direction, which will approximate the actual bearing of R, as illustrated in Figure 40. Due to the position difference between the centers of the two hyperbolas, we do not expect these two angles to be equal, therefore we define a small threshold ϵ_β, such that if

$|\beta_1 - \beta_2| < \epsilon_\beta$, these two angles are considered a match. We then take the average of the two angles to obtain our bearing estimate, $\hat{\beta}$.

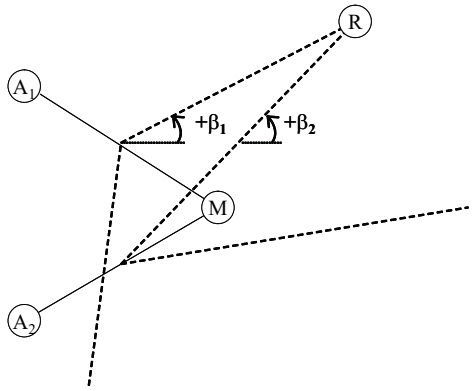


Figure 40: Determining the true bearing of R is accomplished by selecting $+\beta$ or $-\beta$ from each master-assistant pair, such that the difference between the two angles is below the threshold ϵ_β.

Because points on the hyperbola converge with the asymptote as their distance from the hyperbola center increases, the bearing approximation error is larger when R is close to the array. We therefore make the assumption that node R is a sufficient distance from the array. In our error analysis below, we show that this distance does not need to be very large when using small-aperture arrays.

Triangulation

Having approximated the bearing of target node R from a sufficient number of arrays, we can estimate its position using triangulation. Triangulation is the process of determining the position of an object by using the bearings from known reference positions. When two reference points are used (Figure 41a), the target position will be identified as the third point in a triangle of two known angles (the bearings from each reference point), and the length of one side (the distance between reference points).

We calculate the intersection of bearings using the following equations:

$$x = x_2 + cos(\alpha_2)\frac{y_2 - y_1 - tan(\alpha_1)(x_2 - x_1)}{cos(\alpha_2)tan(\alpha_1) - sin(\alpha_2)} \tag{29}$$

$$y = y_2 + sin(\alpha_2)\frac{y_2 - y_1 - tan(\alpha_1)(x_2 - x_1)}{cos(\alpha_2)tan(\alpha_1) - sin(\alpha_2)} \tag{30}$$

where (x, y) are the coordinates of the intersecting bearings (i.e., the position estimate of node R), (x_i, y_i) are the coordinates of array T_i, and α_i is the bearing of R from T_i.

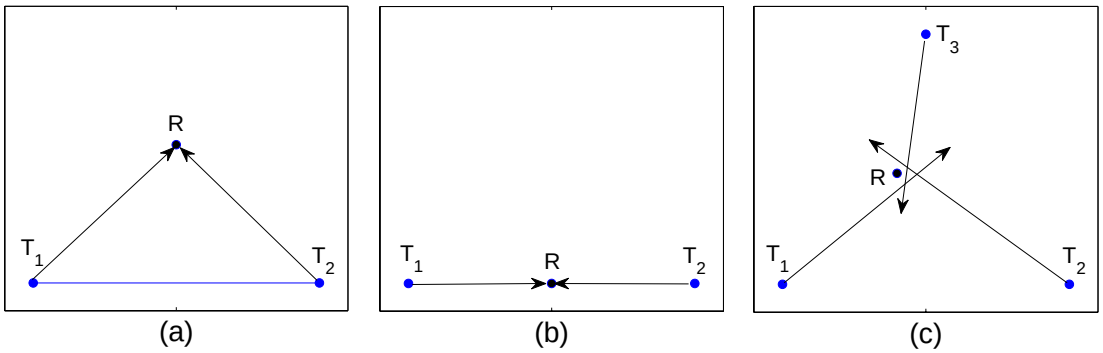


Figure 41: Triangulation. (a) As few as two bearings from known positions are required to estimate the position of a target. (b) Degenerate case where a third bearing is needed to disambiguate position. (c) Three bearings may not intersect at the same position.

When the target position is directly between the two reference points (Figure 41b), two bearings are not sufficient to determine position, because the target could be located at any point on that axis. Therefore, a third bearing is required to disambiguate. However, three bearings may not intersect at the same point if any one bearing is inaccurate (Figure 41c). Triangulation techniques are presented in [84], [67], [135], and [85], in which position estimation using more than two bearings is considered. The method we chose was a least squares orthogonal error vector solution based on [133] and [138], which is rapid, has low complexity, and still provides accurate position estimates from noisy bearing measurements.

Least squares triangulation using orthogonal error vectors works as follows. Figure 42 illustrates a simplified setup with a single TripLoc anchor (T_i) and target receiver node (R). The actual bearing from the anchor to the target is denoted as β_i and the estimate as $\widehat{\beta}_i$. Similarly, the vector pointing from the anchor position to the actual target position is denoted as $\mathbf{v}_i$ and the vector pointing to the estimated target position as $\widehat{\mathbf{v}}_i$. Finally, we denote the difference between the actual and estimated bearing vectors as the orthogonal error vector $\mathbf{e}_i$, such that $\mathbf{e}_i{}^{\mathsf{T}}\widehat{\mathbf{v}}_i = 0$.

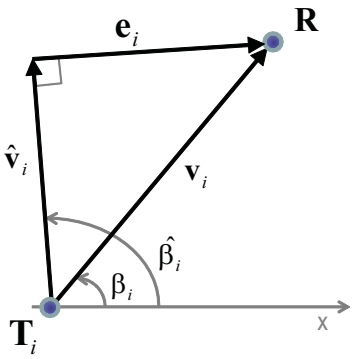


Figure 42: Least squares triangulation using orthogonal error vectors.

If we let $\mathbf{a}_i = \begin{bmatrix} sin\widehat{\beta}_i \\ -cos\widehat{\beta}_i \end{bmatrix}$, then the orthogonal error vector is formally defined as

$$\mathbf{e}_i = \|\mathbf{R} - \mathbf{T}_i\| \, sin(\widehat{\beta}_i - \beta_i)\mathbf{a}_i,$$

where $\|\mathbf{R} - \mathbf{T}_i\|$ is the distance between the target and anchor position vectors, $\widehat{\beta}_i - \beta_i$ is the Gaussian bearing noise with zero mean and variance σ_i^2, and $\mathbf{a}_i$ is the unit vector orthogonal to $\widehat{\mathbf{v}}_i$.

The position of the target can be represented as $\mathbf{R} = \mathbf{T}_i + \widehat{\mathbf{v}}_i + \mathbf{e}_i$. To remove $\widehat{\mathbf{v}}_i$, we multiply with the transpose of $\mathbf{a}_i$, resulting in

$$\mathbf{a}_i{}^{\mathsf{T}}\mathbf{R} = \mathbf{a}_i{}^{\mathsf{T}}\mathbf{T}_i + \eta_i,$$

where $\eta_i = \|\mathbf{R} - \mathbf{T}_i\| \, sin(\widehat{\beta}_i - \beta_i)$. Considering all anchors ($i = 1, \cdots, N$), we have a system of equations that take the form

$$\mathbf{AR} = \mathbf{b} + \eta$$

where $\mathbf{A} = [\mathbf{a}_1{}^{\mathsf{T}}, \mathbf{a}_2{}^{\mathsf{T}}, \cdots, \mathbf{a}_N{}^{\mathsf{T}}]^{\mathsf{T}}$ and $\mathbf{b} = [\mathbf{a}_1{}^{\mathsf{T}}\mathbf{T}_1, \mathbf{a}_2{}^{\mathsf{T}}\mathbf{T}_2, \cdots, \mathbf{a}_N{}^{\mathsf{T}}\mathbf{T}_N]^{\mathsf{T}}$. A least squares solution for estimating $\mathbf{R}$ is given by

$$\widehat{\mathbf{R}} = (\mathbf{A}^{\mathsf{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathsf{T}}\mathbf{b}$$

where $\widehat{\mathbf{R}}$ is the position estimate returned by the triangulation using noisy bearing measurements from N anchors.

Error Analysis

In this section, we present an error analysis of our bearing and position estimation techniques using TripLoc. It is important to note that, although we use phase differences as input to our bearing estimation algorithm, the algorithm is generalizable to small-aperture sensor arrays that can derive distance differences using *any* means. For instance, RF ultra wide band antenna arrays, acoustic or ultrasonic sensors, and other types of arrays that can yield time-difference-of-arrival measurements from (sufficiently) distant sources fall into this category. Therefore, in this section, we assume the inputs to be distance differences. Notice that the distance differences are linearly related to RIM measurements (see Equation (27)), and therefore the error sensitivity results presented below remain valid. In the generalized case, the same applies to TDOA measurements, from which the distance differences can be computed via multiplication of the respective signal propagation speed (speed of sound for acoustic, speed of light for RF).

Bearing Estimation Error Analysis

Typically, bearings are computed from distance differences by solving a nonlinear set of equations using iterative techniques. Such techniques are prohibitive on low-end microprocessors due to their computational cost. We make a set of assumptions that allows us to compute bearing estimates in a reasonably simple way. While our bearing estimation technique is computationally less expensive than traditional nonlinear optimization techniques, our simplifying assumptions introduce estimation errors, which we identify below.

- *Measurement noise.* The distance differences observed by the receiver nodes contain measurement noise. The measurement noise can be attributed to, for instance, non-ideal signal propagation, noise from the electrical circuitry of the receiver, sampling error and quantization error of the analog-to-digital converter.

- *Asymptote approximation.* For a pair of transmitters, we approximate the bearing of the receiver with the angles of the asymptotes of the hyperbola. This is a good approximation if the receiver is sufficiently far from the transmitters, because the hyperbola converges on its asymptote. However, for close receivers, errors due to this assumption will not be negligible.

- *Translation of bearing candidates.* At least two transmitter pairs are required to unambiguously compute the bearing because, for just one transmitter pair, the angles of both asymptotes are possible solutions. We refer to the two solutions as *bearing candidates.* Since, for a transmitter

pair, we compute the bearing candidates with respect to the midpoint of the segment defined by the two antennas, fusing bearing candidates from two different transmitter pairs is not possible without knowing the distance of the receiver. We use the far-field assumption (i.e., that the receiver is infinitely far from the transmitter array) to carry out the disambiguation and fusion of bearings, introducing an error this way.

We intentionally omit the analysis of array position and orientation errors and instead make the following assumptions:

- *Antenna configuration is known.* The transmitter locations are assumed to be given. It is assumed that the transmitter nodes are fabricated with a prescribed antenna separation.
- *Relative bearings.* We assume that the computed bearings are given in the local coordinate system of the array. Hence, the location and orientation errors of the array are not considered in the error analysis of the bearing estimation.

We first analyze the sensitivity of the bearing estimates to noise in the distance difference inputs. Second, we analytically derive the errors related to the asymptote approximation and to the translation of bearing candidates. These errors depend on the bearing and distance of the target receiver, relative to the transmitter array. Finally, we provide an analysis of the total bearing estimation error resulting from both noise in the inputs and the errors due to the asymptote approximation and the translation of bearing candidates.

Sensitivity of bearing to measurement noise.

A distance difference from a pair of transmitters in the array constrains the location of the receiver to one arm of a hyperbola, the foci of which are the positions of the two transmitters. For the sake of simplicity, let us assume that the two transmitters M and A_1 are located at $(c, 0)$ and $(-c, 0)$, respectively (see Figure 39). If the measured distance difference is positive, the receiver is constrained to the right arm of the hyperbola, while if the distance difference is negative, the receiver is located on the left arm. We approximate the bearing of the receiver using the asymptote angles as follows:

$$\hat{\beta} = \begin{cases} \pm tan^{-1}\left(\frac{\sqrt{c^2-a^2}}{a}\right) & \text{if } a > 0 \\ \pm\frac{\pi}{2} & \text{if } a = 0 \\ \pi \mp tan^{-1}\left(\frac{\sqrt{c^2-a^2}}{a}\right) & \text{if } a < 0 \end{cases} \tag{31}$$

We analyze the sensitivity of the bearing estimates $\hat{\beta}$ to noise in the distance difference by taking the partial derivative of Equation (31) with respect to the distance difference. To see what amplification effect an error in a given distance difference d produces on the bearing estimate, we need to evaluate the partial derivative at d.

Figure 43a shows the relation between the measured distance difference d and the bearing candidates $\hat{\beta}$ when the antenna separation is half the wavelength ($\frac{\lambda}{2}$). Notice the ambiguity of the bearing candidates. Figure 43b plots $\frac{\delta\hat{\beta}}{\delta d}$ for each solution of $\hat{\beta}$. This figure shows that when the absolute value of the measured distance difference is close to the antenna separation, the computed bearing candidates are very sensitive to measurement noise. For instance, if the distance difference is 80% of the antenna separation, an infinitesimally small error in the measurement will be amplified ten-fold in the bearing estimate.

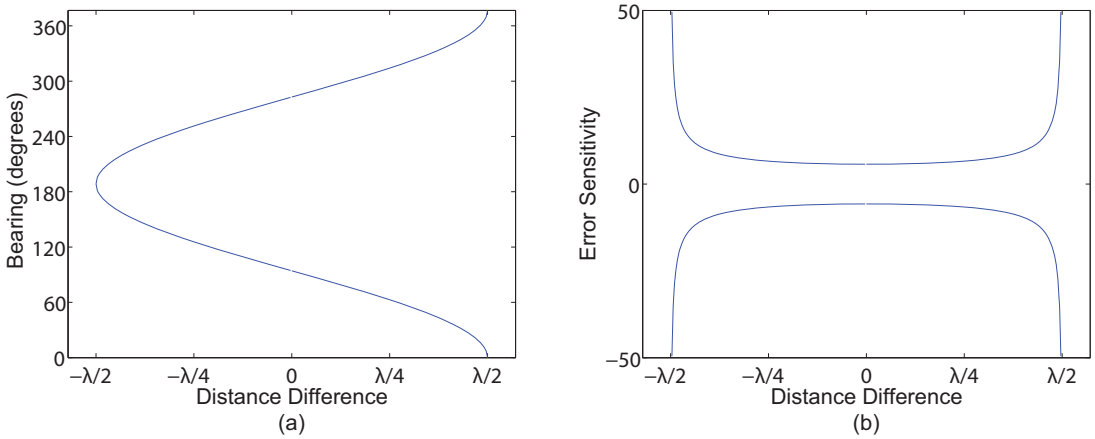


Figure 43: (a) Relationship between measured distance difference and computed bearing. (b) Sensitivity of the computed bearing to measurement noise.

Asymptote approximation.

The error of approximating a hyperbola with its asymptote is the difference between the approximated bearing $\hat{\beta}$ and the actual bearing β of the receiver. Assuming that the receiver R is located at (u, v), the actual bearing will be $\beta = tan^{-1}(\frac{v}{u})$. Hence, the error ϵ introduced by the asymptote assumption is

$$\epsilon = \hat{\beta} - \beta = \begin{cases} \pm tan^{-1}\left(\frac{v}{u}\right) \mp tan^{-1}\left(\frac{\sqrt{c^2-a^2}}{a}\right) & \text{if } a > 0 \\ 0 & \text{if } a = 0 \\ \mp tan^{-1}\left(\frac{v}{u}\right) \pm tan^{-1}\left(\frac{\sqrt{c^2-a^2}}{a}\right) & \text{if } a < 0 \end{cases} \tag{32}$$

Figure 44a shows the error introduced by the asymptote approximation when the receiver is located respectively one, two, and three times the antenna distance away from the midpoint of the segment connecting the two antennas. As expected, the error of the approximation decreases as the distance of the receiver from the transmitter array increases, that is, as the hyperbola converges on its asymptote. As we can see, the maximum error introduced by the asymptote assumption is less than 0.6°, as little as three antenna distances away.

Translation of bearing candidates.

For a pair of transmitter antennas, it is not possible to unambiguously approximate the bearing of the asymptote. Because the hyperbola arm has two asymptotes, the angle of either one can be the correct bearing estimate. Hence, we need two transmitter antenna pairs for disambiguation. Let us treat the bearing candidates (computed from the t-ranges of both transmitter antenna pairs) as vectors of unit length, with bases at the center of the hyperbolas, and whose angles are the computed bearing candidates. Since these vectors are given in the coordinate system of the respective hyperbolas, we need to transform them to the coordinate system of the array. This transformation includes a translation and a rotation. Then, we translate each vector such that its base is at the origin. Clearly, the bearing vector translated this way will not point directly toward the target receiver anymore, but if the receiver is sufficiently far from the transmitter array, the introduced angular error will be small. Finally, we disambiguate the bearing candidates by finding two that have approximately the same value.

Let us now express the angular error caused by the translation of bearing candidates. We assume that the transmitter is a uniform circular array of three antennas, with pairwise antenna distance of $\frac{\lambda}{2}$. The coordinate system of the array is set up such that the midpoint of the array is at the origin, and antenna M lies on the positive side of the x-axis. Let us consider only the correct bearing candidate (the other will be discarded later) for transmitter pair M and A_1. Furthermore, let us assume for now that the bearing candidate has no error. The difference between the actual bearing of the target receiver and the angle of the bearing candidate translated to the origin gives the angular error of the far-field assumption.

Figure 44b shows the error introduced by the far-field assumption when the receiver is located respectively one, two, and three times the antenna distance away from the midpoint of the segment connecting the two antennas. As we can see, as few as three antenna distances away, the maximum error introduced by the far-field assumption is less than 5°. In this particular antenna arrangement, the maximum errors are at 15° and 225°, respectively.

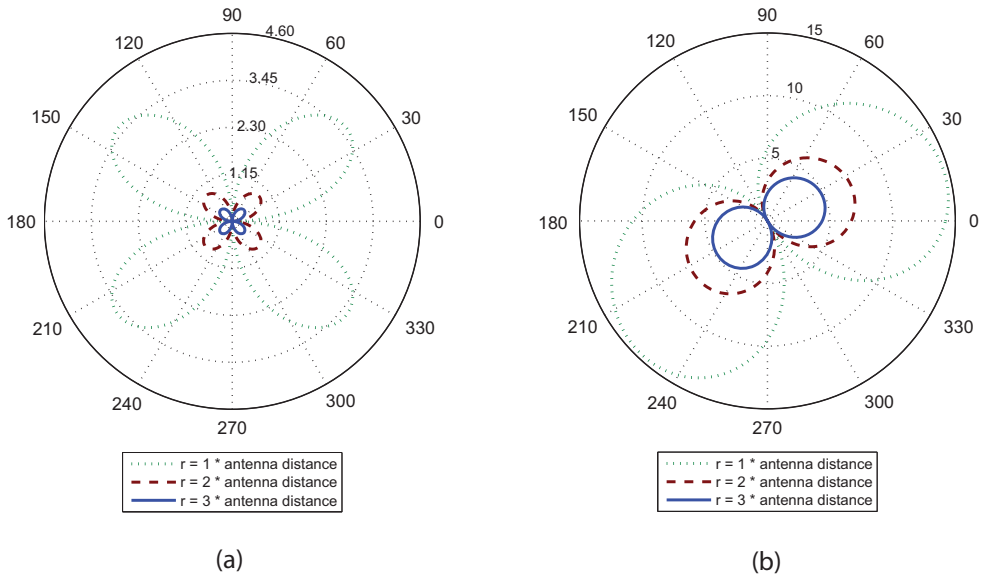


Figure 44: Error in bearing (in degrees) caused by (a) the assumption that the receiver lies on the asymptote, and (b) the assumption that the bearing from the midpoint of the segment connecting the two antennas equals the bearing from the origin of the array coordinate system.

Compound bearing estimation error.

Since one transmitter pair reports two bearing candidates, at least two transmitter pairs are required to resolve this ambiguity. For the sake of simplicity, let us assume that we have two transmitter pairs. Clearly, there must be one bearing candidate for each transmitter pair that is close to the true bearing. Except for some degenerate cases, the other two bearing candidates will be significantly different than the true bearing, and will not be close to each other (see Figure 40). Therefore, in order to disambiguate between the correct and incorrect bearing candidates, we take all possible pairs of bearing candidates, one from the first transmitter pair and the other from the second transmitter pair, and find the pair with the least pairwise angular difference. The reported bearing estimate is computed as the average of the two closest bearing candidates.

Figure 45 shows the bearing estimation errors considering the above three types of error sources, averaged over 500 simulation rounds. We added a Gaussian noise to the distance differences, with zero mean and standard deviation set to 5% of the antenna distance. The plot suggests that the expected bearing estimation errors are below 5°, and peak around 30°, 150°, 240° and 330°, exactly where the individual transmitter pairs exhibit high error sensitivity.

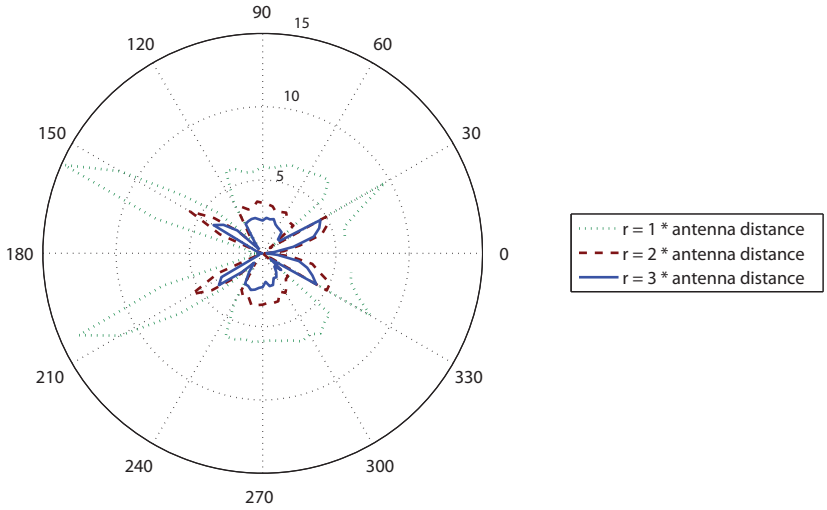


Figure 45: Absolute error of bearing estimation (in degrees) caused by noisy distance differences, averaged over 500 simulation rounds. The standard deviation of the distance difference errors is 5% of the antenna distance.

Position Estimation Error Analysis

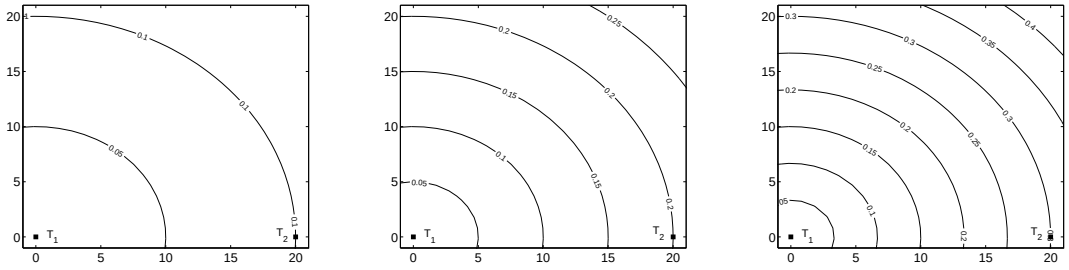
Our localization technique relies on triangulation to compute the position of the receiver nodes. Triangulation is a well studied localization method [84], [67], [85], [135], therefore we omit a detailed evaluation and refer the reader to the above references. Instead, we only present simulation results that demonstrate the effects of error from different sources. The simulation results will also serve as a reference to evaluate our experimental results.

We consider three different sources of error: position error of the arrays, orientation error of the arrays, and bearing estimation error.

Position error of the arrays

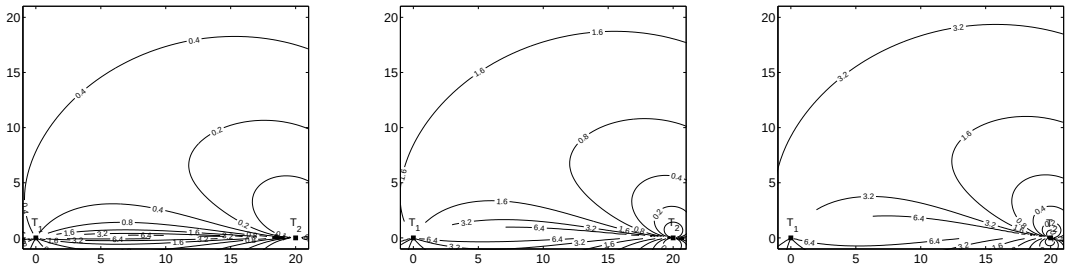
For simplicity, let us assume that we have two arrays T_1 and T_2, located at $(0, 0)$ and $(u, 0)$, respectively. We model the error of array positioning by varying u around a nominal value of 20 meters. Note that we can omit analysis of the effect of varying the y-coordinate of T_2, since an error in both coordinates of T_2 can be modeled by an error in the x-coordinate and an orientation error of T_1.

Figure 46 shows the localization error caused by error in the x-coordinate of array T_2, while all other sources of error are eliminated. As we can see, the maximum errors in the 20m by 20m sensing region are approximately $0.15m$, $0.3m$, and $0.5m$ for array position errors of $0.1m$, $0.2m$, and $0.3m$, respectively.



(a) (b) (c)

Figure 46: Location error (in meters) caused by (a) 0.1 m, (b) 0.2 m and (c) 0.3 m error in the x coordinate of array T_2, respectively.



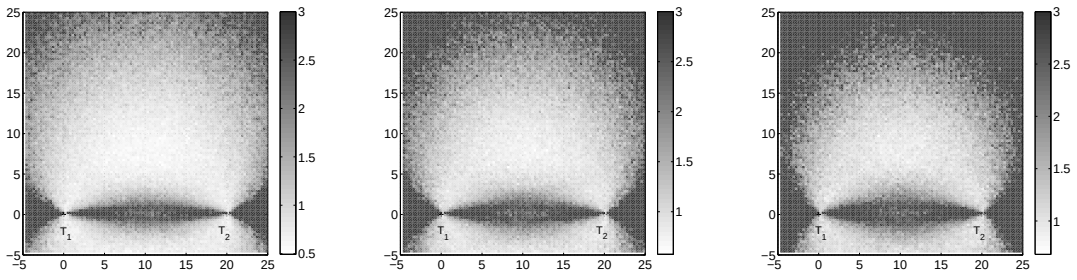
(a) (b) (c)

Figure 47: Location error (in meters) caused by (a) 1°, (b) 4°, and (c) 8° errors in the orientation of array T_2, respectively.

Orientation error of the arrays

To simulate the effects of orientation error, we vary the orientation of array T_2 around its nominal value. This is simulated by adding a constant to the bearings reported from T_2.

Figure 47 presents the localization error due to 1, 4 and 8 degrees of orientation error of T_2, eliminating all other error factors. As expected, if the receiver is close to the line defined by the two arrays, even a small orientation error will cause a considerable localization error. This is because the lines defined by the bearings intersect at an angle close to 180 degrees, and even a small error in the angles causes a considerable relocation of the intersection. The simulation results show that this particular configuration is very sensitive to orientation errors: with an orientation error of 1°, we see $0.5m$ localization errors in the sensing region. With an orientation error of 4°, the expected errors are close to $3m$, while with 8° orientation error, the localization errors become unacceptably large in most of parts of the sensing region.



(a) (b) (c)

Figure 48: Standard deviation of location error (in meters) caused by bearing noise with zero mean and standard deviation (a) $\sigma = 4$, (b) $\sigma = 5$ and (c) $\sigma = 6$ degrees, respectively.

Bearing estimation error

We now present simulation results to characterize the effect of bearing estimation errors on the computed locations. We assume that the location and the orientation of two antenna arrays are known. Furthermore, we assume that the bearing estimates have additive white noise. We model the noise with a Gaussian random variable with zero mean and a known standard deviation σ. It is assumed that the noise in the bearings from arrays T_1 and T_2 are uncorrelated.

Through 100 simulation rounds, we compute the standard deviation of localization errors for all points on a high-resolution grid covering the entire sensing region. Figure 48 shows the simulation results for bearing errors with standard deviation $\sigma = 4^\circ$, $\sigma = 5^\circ$ and $\sigma = 6^\circ$, respectively. The results suggest that standard deviation of the position errors is high when the receiver is close to the line connecting the two arrays, and increases towards the edge of the sensing region. The central part of the sensing region gives the lowest error variance. As we increase the standard deviation of the bearing error, the standard deviation of the location error also increases. As the standard deviation of the bearing errors increases, localization precision degrades. With 6° standard deviation of bearings, only the central part of the sensing region offers less than $1.5m$ location error variance.

Implementation

Our system is implemented on Crossbow ExScal motes [30], which use the Texas Instruments CC1000 radio chip [96], and transmit in the 433 MHz range. Three XSMs form a TripLoc array. Because the two transmitting antennas are so close to each other, they will suffer from parasitic effects [139]. To minimize this negative interference, we place the nodes in a mutually orthogonal configuration. The nodes are elevated approximately 1.5 meters to reduce ground-based reflections. The TripLoc antenna array is pictured in Figure 37.

All nodes in our system execute the same distributed application, coded in nesC [131], and run the TinyOS operating system [128]. All operations run locally, and there is no offline or PC-based processing. The entire application requires 3 kB of RAM and 57 kB of program memory (ROM).

Antenna Separation

The antenna separation of the TripLoc array is 35.35 cm. In order to avoid the modulo 2π phase ambiguity, we were required to transmit a signal with wavelength greater than 70.7 cm, (i.e., a frequency less than 424 MHz). This was possible, as we observed reliable CC1000 transmissions as low as 405 MHz. Ultimately, we chose to transmit at a carrier frequency of 409 MHz, corresponding to a wavelength of 73 cm.

Initialization

Before starting the localization process, system initialization is required.

Scheduling

First, we must create a schedule of the order that array anchors perform their RIMs. This schedule is broadcast to all anchors in the sensing region. During run-time, the anchors keep track of who is currently performing the RIM. The next anchor in the schedule will then know to begin its RIM when the previous one ends.

Tuning

One of the issues that arises when using the CC1000 is that the true carrier frequency can differ from that of the desired frequency by up to 2 kHz. In order to account for this, we must tune the radios by having the master and assistant transmit at the intended frequencies, while a third node monitors the signal. The master then changes the frequency in small steps. The receiver observes the resulting beat frequency, and when it matches the desired one, informs the master. From this point on, the master transmits at this new frequency, in order to maintain a consistent interference signal.

Calibration

Finally, we note that it is difficult to fix exact array orientations. If the array is not level, or is rotated slightly, the shape of the resulting hyperboloid will degrade the localization result. An easy way

around this is to place one or more sensor nodes at known positions and run the bearing estimation algorithm. Because all positions are known, bearings between the sensor nodes and arrays are also known. The difference between the actual bearings and the computed bearings indicate the true orientation of the arrays. By using this simple, yet effective technique, we saw a decrease in bearing error by 56%.

Radio Interferometric Measurements

A RIM consists of several tasks, executed sequentially according to a strict schedule. In order to transmit or receive the RF signal at the desired frequency, the radio must be acquired from the standard MAC layer. Once the radio is acquired, nodes are unable to use their radios for standard message passing, so each node participating in the RIM executes each task in lockstep. Therefore, before acquiring the radio from the MAC layer, the first step is to synchronize the participating nodes.

Time Synchronization

Because phase difference is used to determine bearing, each node must measure the signal phase at the same time instant. This requires a synchronization protocol with accuracy on the order of microseconds or better. To accomplish this, we use a SyncEvent [52], the same technique employed by RIPS and other radio interferometric based ranging methods. The SyncEvent is a message, broadcast by the master transmitter, that contains a time in the future to start the RIM. The time is specified according to the local clock of the master. The clock offset between the sender and receiver of this message is determined as follows. The sender inserts a timestamp at the end of the message *after* the message has already begun transmission. On the receiver side, a timestamp is taken upon message reception, and the difference between these two timestamps estimate the clock offset between the two nodes. The propagation time of the message through air is negligible. The accuracy of the SyncEvent was reported to be 1.4 μs. Clock skew is not an issue here because synchronization is performed at the beginning of the RIM, and relative clock rate does not noticeably change in the time it takes to perform the RIM.

Acquiring the Radio

The first task each node (transmitter and receiver) must perform after synchronization is acquiring the radio from the MAC layer. This involves switching radio drivers to enable the transmission/reception of a pure sine wave at the desired frequency, a feature the standard radio driver does not support.

Radio Calibration

Once the radio has been acquired, we must calibrate it for transmission at the desired frequency. To do this, we use previous calibration data, which avoids the time-consuming calibration step that automatically occurs in the restore task.

Signal Transmission / Reception

In order to determine the signal phase, we must sample enough RSSI values to reconstruct the signal. This requires a transmission/reception time that is long enough to sample about six periods, from which we can use filtering techniques to obtain a fairly smooth sinusoid. As each RSSI value is obtained, it is inserted into a 256-byte buffer. At the end of the sampling period, we traverse the buffer, keeping track of the minimum and maximum values, from which we can determine the signal amplitude and frequency. Once these signal characteristics are known, we can determine the phase of the signal at the specified time instant.

Restoring the Radio

Once the transmission is complete, the standard radio must be restored to its original state in order to exchange the phase data between nodes.

Bearing Approximation and Triangulation

After each array performs its RIMs, the assistant nodes broadcast their observed phase data, which are received by the target receiver nodes. The phase data packets are small, only 12 bytes, five of which comprise the standard active message header, two for the assistant ID, one for the sequence number, and four bytes for the measured phase. For redundancy, the data packets are broadcast three times; however, any messages that arrive after the communication deadline of 250 ms will be discarded. The communication deadline signals the start of the localization process, and incoming data after the algorithm is underway will cause unexpected errors. The localization method per-

forms both bearing approximation and triangulation and is optimized for memory and speed. The localization result is sent to a basestation for logging; however, for mote applications that require the position updates, no routing is required, and the position estimate is readily available.

Latency Analysis

Because we would like to use TripLoc to localize mobile sensors in addition to stationary nodes, each array must perform its RIMs as fast as possible so that the mobile sensor has not had a chance to significantly change its position. In order to keep the localization latency to a minimum, we provide an analysis of the different component execution times.

Table 10 lists the execution deadlines of the tasks presented above. These deadlines were chosen to give each task enough time to complete, assuming a reasonable amount of jitter, and were determined experimentally. Each array performs two RIMs, one for each master-assistant pair. For each array, only one synchronization message is broadcast at the beginning of the first RIM. When a RIM completes, a *measurementEnded()* event is triggered on each participating node. This includes nodes belonging to other arrays, even if they took no part in the phase measurement. The event enables the next array to begin its RIM without delay. The receiver nodes store their phase measurements for each array until all arrays have finished their RIMs. After all arrays have finished, the participating assistants broadcast their phase measurements, and the receiver nodes calculate their bearings to each array. We allow 250 ms for sending phase measurements. Localization takes no longer than 6.5 ms, after which the first array begins its next round of RIMs.

Task	Latency (ms)
Synchronize	163
Acquire	5.4
Calibrate	1.08
Transmit / Receive	63.2
Restore	49.91
Report phase	250
Localize	6.5

Table 10: Latency of localization tasks.

Figure 49 is a sequence diagram of these events using a setup of two arrays and a single receiver node. Because the target nodes act as receivers, no additional latency is incurred by introducing more to the sensing region. Each array sends one synchronization message and performs two RIMs, for a total time of 402.18 ms. After all arrays have performed their RIMs, an additional 256.5 ms is required for communication and localization. Therefore, localization using three arrays will take

1.46 seconds, and 1.86 seconds when using four arrays. It should be noted that this latency can be further decreased by removing subsequent synchronization messages after the first one. Because a round of RIMs takes a relatively short time, clocks will not experience a significant amount of drift. In addition, the time alloted to report phases is much greater than is actually needed, and can be shortened, if necessary. We chose to set it longer to ensure we received all measurement data for analysis and debugging purposes. From this latency analysis, we see that a sufficient number of arrays can perform RIMs to enable a receiver node to determine its position in under two seconds.

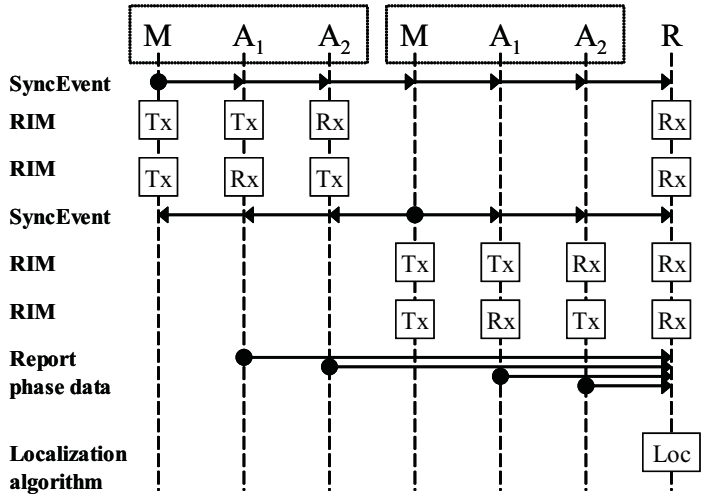


Figure 49: Sequence diagram of a RIM schedule with two arrays (dotted boxes) and a receiver node (R).

Experimental Evaluation

To evaluate the accuracy of TripLoc, we performed three experiments. The first two experiments measure the bearing accuracy of the system, and the third experiment measures the localization accuracy.

Bearing Accuracy

For Experiment 1, we measure the bearing accuracy of six receiver nodes, which are evenly spaced around the array every 60° at a distance of ten meters from the array center. This experiment demonstrates how the bearing error changes with respect to array orientation. For Experiment 2, we measure the bearing accuracy of 14 receiver nodes from three arrays surrounding the sensing region in an outdoor, low-multipath environment. This experiment is more representative of a real-world deployment with multiple anchors. Figure 50 illustrates our setup for both of these experiments.

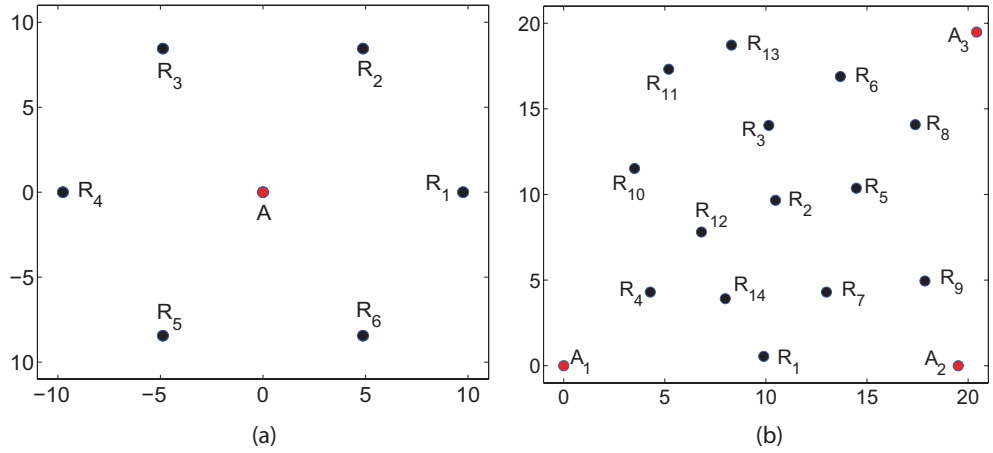


Figure 50: Experimental setup for bearing accuracy of TripLoc. (a) Experiment 1. Bearing accuracy of one array. Six receiver nodes ($R_1 \ldots R_6$) are placed 10 meters from array (A), with angular separation of 60°. (b) Experiment 2. Three arrays ($A_1 \ldots A_3$) surround the 20 x 20 m sensing region containing 14 target receiver nodes ($R_1 \ldots R_{14}$).

For Experiment 1, we perform 50 bearing estimates for each node surrounding the array. The average bearing errors are displayed in Figure 51a. For Experiment 2, we perform approximately 35 bearing estimates from each anchor to all target nodes, resulting in a total of 105 estimates per target and 1470 estimates total. Figure 51a shows the average error for each bearing from Experiment 1. The distribution of bearing estimate errors from Experiment 2 are shown in Figure 51b. The average bearing estimation error is 3.2° overall, with a 6.4° accuracy at the 90th percentile. The errors from both experiments are consistent with our bearing error analysis, above.

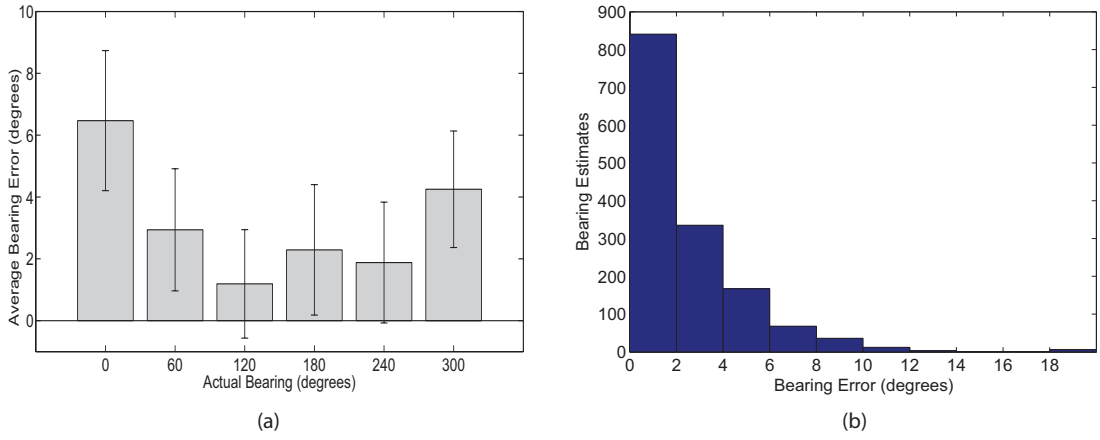


Figure 51: Experimental results. (a) Experiment 1 average bearing error with respect to array orientation (sample size of 50). (b) Experiment 2 bearing error distribution (sample size of 1470).

Localization Accuracy

For Experiment 3, we place four TripLoc array anchors at the corners of a 20 x 20 m sensing region containing 16 target receiver nodes. Each target node periodically estimates its position using the bearing estimation and triangulation techniques presented above. Figure 52 illustrates our setup for this experiment.

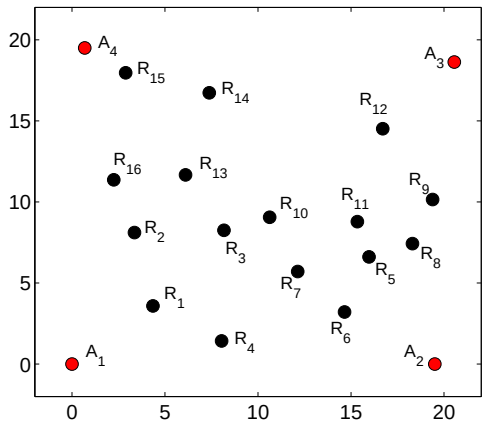


Figure 52: Experimental setup for localization accuracy of TripLoc (Experiment 3). Four arrays ($A_1 \ldots A_4$) surround the 20 x 20 m sensing region containing 16 target receiver nodes ($R_1 \ldots R_{16}$).

For Experiment 3, we perform 50 position estimates for each target node. Figure 53 shows the average position estimate for each target relative to its actual position. The distribution of position estimate errors are shown in Figure 54. The average overall position error is 1.13 m.

Note that, for the most part, position accuracy tends to worsen as the target node position approaches the perimeter of the sensing region. In this setup, the perimeter is where the bearings from at least one array are particularly sensitive to estimation error. If we only focus on the node locations in the center of the sensing region (at least 2.5 meters from the perimeter), the average position error drops to 0.62 meters. We therefore suggest placing the TripLoc anchors outside of the sensing region for the best overall accuracy.

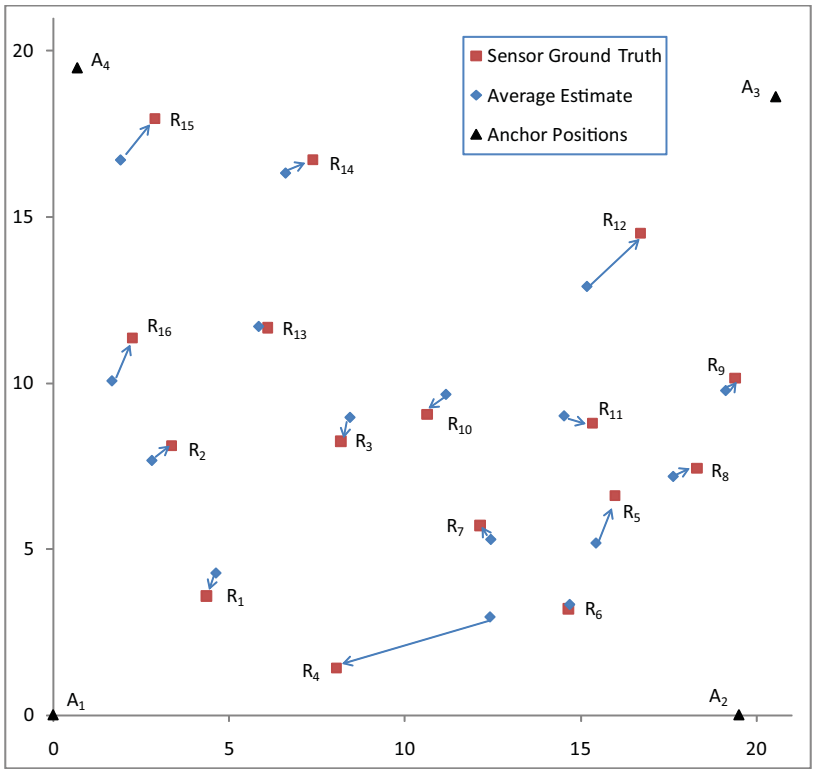


Figure 53: Experiment 3 average position estimate for each target relative to actual position. Arrows connect the estimated position of a target node with its actual location.

Conclusion

In this chapter, we present a method for rapid distributed bearing estimation and localization in WSNs. The array anchor in our system consists of three nodes, two of which transmit at frequencies that interfere to create a low-frequency beat signal. The phase of this signal is measured by the third node in the array, as well as by multiple target nodes at unknown positions. The phase difference defines a hyperbola, and bearing can be approximated by calculating the angle of the asymptote. Our experimental results show that this technique has an average bearing estimation accuracy of 3.2°, average position accuracy of 0.62 m, and measurements can be taken in approximately 1.5 sec.

Our system is designed to overcome several challenges in WSN AOA determination. The array prototype is easily constructed by fixing three motes together with antennas at orthogonal angles. It is comprised entirely of COTS sensor nodes, and no additional hardware is required because RIM-based ranging only requires use of the radio. Unlike other radio interferometric techniques, our system avoids the modulo 2π ambiguity, and therefore the need to perform RIMs on multiple channels, by separating the two transmitting antennas less than half the wavelength of the carrier frequency. Similarly, by constraining the location of one of the RIM receivers to the array, it becomes

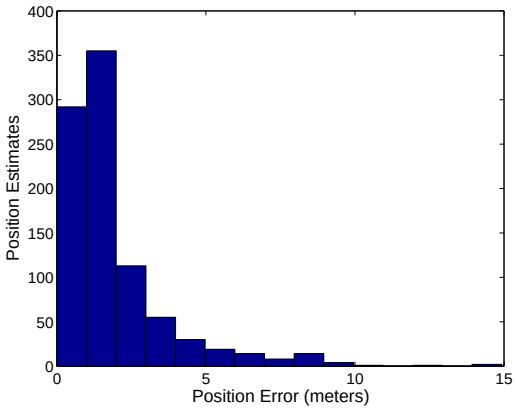


Figure 54: Experiment 3 position error distribution (sample size of 912).

possible to approximate the bearing of the other receiver without prolonged computation or having to rely on a base station for processing. Our experimental results demonstrate that the accuracy of our prototype implementation is on par with other state-of-the-art AOA techniques.

CHAPTER VII

MOBILE SENSOR NAVIGATION USING TRIPLOC

Introduction

In this chapter, we describe a navigation system called TripNav that uses TripLoc (presented in Chapter VI) to obtain position estimates. These estimates enable the mobile sensor to make the proper course corrections in order to arrive at a goal position. Typically, autonomous navigation is performed by robots that have cameras, laser rangefinders, and sonar sensors to collect range and bearing information that can be used to calculate position or angular separation between landmarks. However, these sensors are large, expensive, have considerable power requirements, and/or use a powerful computing platform to analyze sensor data. In order to implement angle-based navigation on a network of resource-constrained devices, we must use methods that are fast, accurate, and do not require a substantial amount of system resources.

We have developed a waypoint navigation system, in which a mobile sensor follows a path by navigating between position coordinates. As described in Chapter II, way-finding represents a major category of navigational behavior. Waypoint navigation scenarios could include automated transport routes and sentries that patrol a path along the perimeter of a secure area. For our research, the mobile sensor is provided with a target speed and a set of waypoints, and is instructed to pass by each waypoint in the order they are given. TripLoc anchors are used for localization because they can rapidly provide accurate position estimates and do not require additional hardware support. However, TripLoc does not enable the mobile sensor to determine its own orientation. To handle this, we employ a digital compass, which provides heading estimates, from which we can calculate the heading error of the mobile sensor with respect to the desired trajectory. A simple controller is then used to derive the necessary wheel speeds for maintaining the correct heading.

In Chapter IV, we presented another waypoint navigation system for MWSNs called dNav. Although both systems are RF-based and designed for resource-constrained mobile sensors, there are some fundamental differences between the two. For instance, dNav estimates mobile sensor position *and* velocity by measuring the Doppler shift in frequency of the radio interferometric beat signal, whereas TripNav estimates only position by measuring signal phase. As a simple rule of thumb, the frequency of a signal in the 433 MHz range will undergo a Doppler shift of approximately 1 Hz for every 1 m/s velocity difference between transmitter and receiver. When dealing with slow moving

mobile sensors, the Doppler shift can potentially be drowned out by measurement noise. In fact, we found that when moving less than 0.5 m/s, we could not reliably determine position or velocity based on Doppler shifts. Another issue with dNav is that it uses an EKF for estimating position and velocity in the presence of measurement noise. Although the filter provides accurate estimates, it consumes a substantial amount of memory, and we were required to implement it on a separate mote. TripNav is unable to directly estimate speed or heading; however, it is able to run on a single mote. Furthermore, it can still provide accurate navigation, even without the use of a filter.

The contributions of this work are as follows:

1. We describe a lightweight waypoint navigation system for resource-constrained mobile wireless sensor networks.
2. We perform error and timing analyses that show TripNav performs reliably.
3. We demonstrate that TripLoc is indeed suitable for mobile sensor navigation.
4. Our experimental results show that the navigation technique works reliably and has low trajectory error.

System Overview

Our waypoint navigation system consists of TripLoc infrastructure nodes and a mobile sensor that traverses a region in order to perform some task. In order for a mobile node to travel between waypoints, it is necessary to know the node's current position. By approximating the bearing of the mobile node from a sufficient number of landmarks, node position can be estimated using triangulation [133]. Figure 55 illustrates a simple waypoint navigation scenario.

Figure 55: Waypoint Navigation. A mobile entity traverses the sensing region by navigating between position coordinates.

The waypoint navigation system works as follows. A mobile sensor traverses the sensing region by moving from waypoint to waypoint. The waypoint coordinates are stored in the mote's memory.

The mobile sensor observes the phase of interference signals transmitted sequentially by anchor nodes at known positions within the sensing region. After observing a sufficient number of these measurements, the mobile sensor estimates its current position via triangulation. These position estimates are used to determine if the mobile node has reached the next waypoint. Heading error is determined using an onboard digital compass. The heading error is fed into a controller, which updates the wheel angular velocities and keeps the mobile sensor on the correct trajectory to intercept the waypoint. Figure 56 is a diagram of the control loop. We describe each step of this process below.

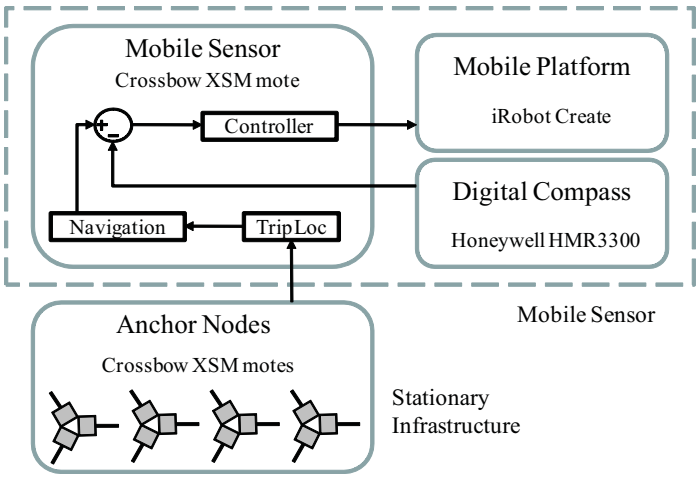


Figure 56: Control loop for waypoint navigation.

Mobile Sensor Kinematics

Similar to dNav (see Chapter IV), we use the following equations to describe the kinematic model of our two-wheeled mobile platform with differential steering:

$$\dot{x} = \frac{r(\omega_r + \omega_l)}{2} cos\phi \tag{33}$$

$$\dot{y} = \frac{r(\omega_r + \omega_l)}{2} sin\phi \tag{34}$$

$$\dot{\phi} = \frac{r(\omega_r - \omega_l)}{2b} \tag{35}$$

where x and y constitute the mobile node's position, ϕ is the heading, r is the wheel radius, b is the distance between the hub center of the driving wheel and mobile platform axis of symmetry, and ω_r and ω_l are the right and left wheel angular velocities, respectively. The speed of the mobile sensor is the magnitude of the velocity, and in terms of angular velocity is represented as $|v| = \frac{r(\omega_r+\omega_l)}{2}$.

Position and Heading Estimation

In order for a mobile node to travel between arbitrary waypoints, it is necessary to know its current position and heading. Using TripLoc, a node can determine its position with as little as two anchor arrays, the minimum required for triangulation. As presented in Chapter VI, TripLoc was shown to have an average position error of 0.62 meters.

Although TripLoc provides accurate localization, it is unable to estimate orientation. Therefore, to obtain heading, we use an onboard digital compass, which has an accuracy of approximately 1°.

Waypoint Navigation

The mobile node needs to follow a trajectory (reference heading) that will lead it to the next waypoint. The bearing from the node's current position to the waypoint is one such trajectory. However, localization error can cause the trajectory to be quite inaccurate, especially when the mobile node is close to the waypoint. Instead, we define the reference heading as the bearing from the previous waypoint (or the initial position estimate of the mobile node) to the next waypoint:

$$\phi_{Ref} = \begin{cases} tan^{-1}\left(\frac{w_{y_i}-\widehat{y}}{w_{x_i}-\widehat{x}}\right) & \text{if } i = 1 \\ tan^{-1}\left(\frac{w_{y_i}-w_{y_{i-1}}}{w_{x_i}-w_{x_{i-1}}}\right) & \text{if } i > 1 \end{cases}$$

where w_{x_i} and w_{y_i} are the coordinates of waypoint i and $\widehat{x}$ and $\widehat{y}$ are the estimated position of the mobile node. Initially, ϕ_{Ref} is computed based on the position of the mobile node and the first waypoint. After the mobile node has reached the first waypoint, ϕ_{Ref} is calculated once for each waypoint i at the time waypoint $i-1$ is reached.

Heading error is then determined by subtracting the mobile node's heading estimate, $\widehat{\phi}$, from the reference heading, ϕ_{Ref}:

$$\phi_{Err} = \phi_{Ref} - \widehat{\phi}. \tag{36}$$

To arrive at the angular velocities that will keep the mobile sensor on the reference trajectory, we use a PI controller that takes the heading error ϕ_{Err} as input. Because the heading wraps to 0 at 2π, we shift the heading error to fall between $-\pi$ and π:

$$\phi_{Err} = \begin{cases} \phi_{Err} - 2\pi & \text{if } \phi_{Err} > \pi \\ \phi_{Err} + 2\pi & \text{if } \phi_{Err} < -\pi \\ \phi_{Err} & \text{otherwise} \end{cases}$$

The controller then takes the following form:

$$\dot{\phi} = K_p \phi_{Err}(T) + K_i T_e \sum_{t=1}^{T} \phi_{Err}(t) \tag{37}$$

where K_p and K_i are constant proportional and integral gains, respectively, T is the current sample number, $\phi_{Err}(t)$ is the heading error for sample t, and T_e is the time elapsed from the previous sample. The output of the controller, $\dot{\phi}$, is the updated angular velocity of the mobile sensor; however, the mobile platform is commanded by specifying an angular velocity for each wheel. Consequently, we convert $\dot{\phi}$ into individual wheel angular velocities, ω_l and ω_r, as follows:

$$\omega_l = \frac{|v| - b\dot{\phi}}{r} \tag{38}$$

$$\omega_r = \frac{|v| + b\dot{\phi}}{r} \tag{39}$$

where ω_r and ω_l are the right and left wheel angular velocities, respectively. Here, r and b (defined above) are system parameters with known values. $|v|$ is an input parameter to this system and does not change even though the mobile platform may not actually achieve the desired value. This is because we are only interested in regulating our heading and not our speed.

The effect of the above transformation is that both wheels will be set with an equal desired base speed. If heading error exists, the controller will minimize it by turning one wheel faster than the base speed, and the other wheel slower, which will result in the mobile node turning in the correct direction as it moves forward. This type of controller has low run-time complexity and does not require a substantial amount of memory.

Error Analysis

In this section, we analyze the main sources of error in TripNav. We perform the analysis by generating a simulated setup and observing how the following error sources affect the results. The simulation engine models the dynamics of the mobile node and computes the ideal bearings from each TripLoc anchor at each timestep. Triangulation is then performed using the computed bearings. For the error analysis, Gaussian noise is added to the computed bearings and the position estimates, as described below for each source of error. In the simulation, we position TripLoc anchors at the corners of a 20 x 20 meter region. The mobile node follows a path that takes it around a 10 x 10 meter square within the sensing region. The desired speed of the mobile node is fixed first at 100 mm/s and then at 400 mm/s. This setup is identical to our real-world experimental evaluation, described in detail below and illustrated in Figure 61.

TripLoc Position Error

Although TripLoc position error can reach as high as several meters, it contributes relatively little to TripNav error. This is because TripLoc position estimates are only used to recognize waypoint proximity. The rest of the time, the digital compass is used to maintain the desired trajectory. To analyze the effect of TripLoc accuracy on TripNav, we simulate the system under ideal conditions, while adding Gaussian noise with zero mean and varying the standard deviation between zero and five meters. Figure 57 shows the simulated paths of the mobile node with different TripLoc accuracies. We see from the figure that even with large TripLoc error, the mobile node will still complete the circuit; however, the path it follows can be offset significantly from the desired path.

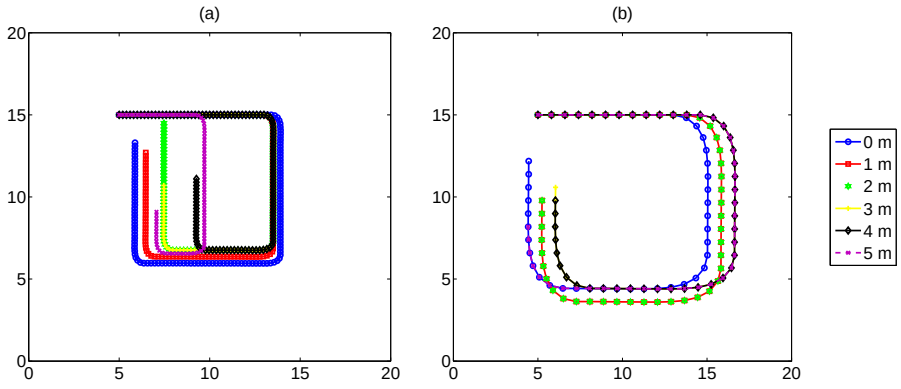


Figure 57: TripNav trajectories due to TripLoc position error when the mobile sensor speed is (a) 100 mm/s, and (b) 400 mm/s.

Digital Compass Measurement Noise

In order to compute the heading error of the mobile node, its current orientation must be known. To determine this, we use a digital compass. To understand how noisy compass sensor data affects navigation, we performed 100 simulated runs under ideal conditions, introducing a Gaussian noise to the compass heading with zero mean and a standard deviation of 0.5°, 1°, 2°, 3°, 4°, and 5°. Figure 58 shows the average associated position error for each. From the figure, we see that even a compass heading error as high as 5° does not contribute significantly to the position error. The digital compass we use in our real-world evaluation (see below) has an accuracy of about 1°.

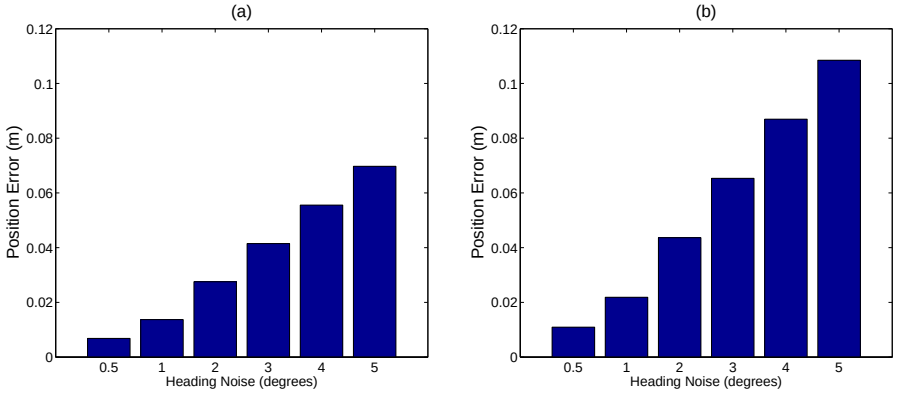


Figure 58: TripNav average position error due to digital compass sensor noise when the mobile sensor speed is (a) 100 mm/s, and (b) 400 mm/s.

Latency

Because the mobile node is in motion while running TripLoc, the accuracy of the position estimate will depend on the speed of the mobile node and the latency of the localization. In TripLoc, bearings from each anchor are estimated sequentially. Triangulation is then performed to determine position by finding the intersection of these bearing vectors. However, even if all other sources of error were absent from this system, these bearing vectors would still not intersect at a common point because each measurement is made from a slightly different physical location. In addition, once all measurements have been taken, the mobile node continues to change its location while phase data is being transmitted from the anchor nodes and the position estimate is computed. Therefore, the faster the TripNav control loop runs, the more accurate the position estimates will be, because the mobile node will not have had a chance to move far from the position where the localization algorithm was initiated.

To analyze how this affects the accuracy of the TripNav system, we simulate the system under ideal conditions while varying the number of TripLoc anchors. We performed 100 simulated runs, and averaged the position error for each localization. Figure 59 shows the average position error we can expect due to latency when we use two, three, and four TripLoc anchors. From the figure we can see that the latency incurred by increasing the number of anchors affects TripNav position accuracy on the order of centimeters.

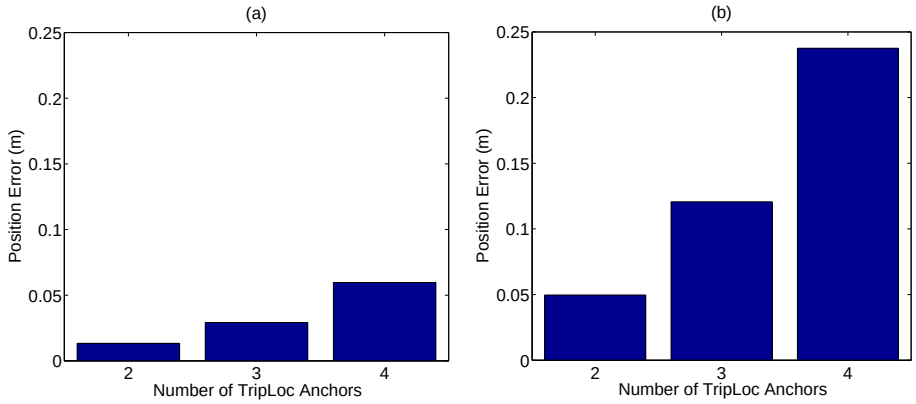


Figure 59: TripNav average position error due to latency using 2, 3, and 4 anchors when the mobile sensor speed is (a) 100 mm/s, and (b) 400 mm/s.

Implementation

Our mobile sensor is comprised of an XSM mote [30] attached to an iRobot Create robot [140], as pictured in Figure 60. All localization and control operations are performed on the mote, which communicates with the Create microcontroller over a serial interface. Mobile sensor heading is determined using a Honeywell HMC3300 digital compass [141]. The Create acts solely as a mobile platform and does not perform any computation or control independently of the mote. The TripLoc anchor node implementation is described in Chapter VI.

The Create is a small-profile mobile platform, only 7.65 cm tall. Fixing the XSM mote to the Create body becomes problematic because the localization transmission signal is influenced by ground-based reflections. For the TripLoc implementation and experimental evaluation, TripLoc anchors were elevated approximately 1.5 meters off the ground. Mounting the mote that high on the Create would cause the mobile platform to tip over. Instead, we built a mount out of lightweight PVC tubing that places the mote 85 cm off the ground. We determined this height was sufficient to minimize the effect of ground-based reflections. The mount is fixed to the Create body, and houses

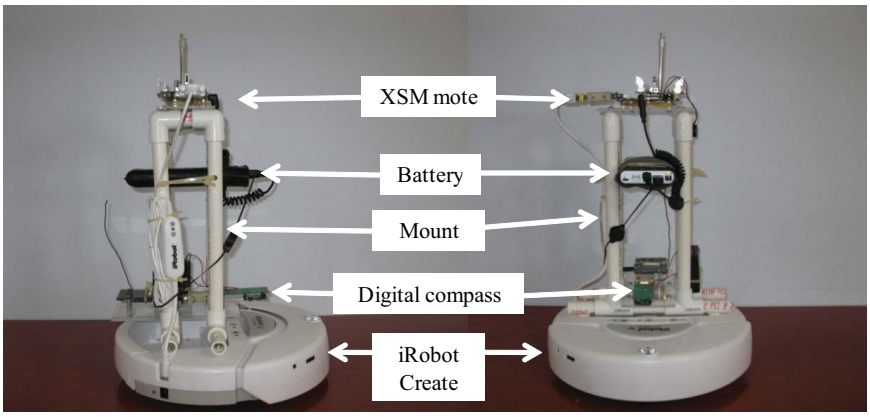


Figure 60: Our mobile platform. An XSM mote is attached to an iRobot Create via a serial connection. Although the Create has an onboard control module, it is disabled for our experiments, and all control operations are performed on the attached mote.

the XSM mote, the digital compass, as well as the connecting cable assembly for communicating with the Create, and a battery pack.

One of the main implementation challenges for TripNav is designing an accurate rapid localization system, as well as waypoint navigation logic and a mobile controller that is small enough to fit on a mote. Our TripNav implementation consumes approximately 3.1 kB RAM (out of an available 4 kB) and 60 kB of programming memory (120 kB available).

Evaluation

Performance Analysis

We placed four TripLoc anchors at the corners of a 20 x 20 meter region. The mobile sensor was given a series of four waypoint coordinates that instructed it to drive in a square. Once the mobile sensor reached the last waypoint (i.e., completed the circuit) it was instructed to come to a stop. Figure 61 illustrates this setup.

There are several tunable parameters for waypoint navigation using TripNav. Unlike dNav, TripNav only controls the heading of the mobile sensor and not its speed. Therefore an important system parameter is the *target drive speed* (the average translational speed of the sensor). The maximum speed of the Create is 500 mm/s. However, because we modified the Create with a sensor mount, the increased weight (as well as the terrain) limit the speed to about 450 mm/s. Because the controller specifies wheel speeds such that one wheel may rotate faster than the target speed and

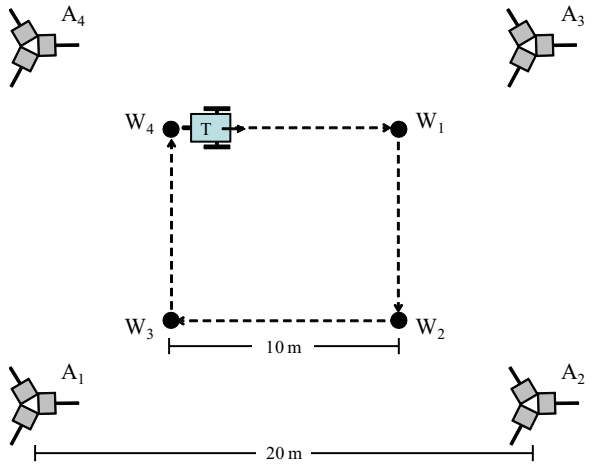


Figure 61: Waypoint navigation experimental setup. TripLoc anchors ($A_1 \ldots A_4$) surround the sensing region. The mobile sensor (T) is instructed to drive in a square, passing through each waypoint ($W_1 \ldots W_4$) before proceeding to the next.

the other slower, we set our maximum target drive speed to be 400 mm/s. For our experiments, we performed waypoint navigation with target drive speeds of 100 mm/s and 400 mm/s.

Because of localization noise and continuous movement, the mobile sensor will not always be able to land exactly on the waypoint, therefore, similar to dNav waypoint navigation, we chose a *waypoint range* that specifies how close the mobile sensor must be to a waypoint before being allowed to proceed to the next. For our experiments, we ultimately chose waypoint ranges of two meters when moving at 100 mm/s and three meters when moving at 400 mm/s. We found that if we increased the waypoint range beyond these values, the mobile sensor would still complete its circuit, however, the path it followed had a high average position error. When we decreased the waypoint range, the mobile sensor would occasionally miss the waypoint, then have to loop around in an attempt to reach it again.

Finally, to filter out inaccurate position estimates, we use a simple validation gate that approximates the distance traveled since the last position estimate by multiplying the elapsed time by the average wheel speed. If the distance difference between the current and previous position estimates is greater than the estimated travel distance plus a *position error constant* (to account for positioning and drive error), then the current position estimate is discarded. We chose a value of 2.5 meters for the position error constant.

We performed five waypoint navigation runs for both target drive speeds using TripNav. Figure 62 shows the average path of the mobile sensor over all runs. Note that the mobile sensor path does not intersect with the waypoints, and seems to stop short of the final waypoint. This is due

to the waypoint range setting, where the mobile sensor considers the waypoint reached if it comes within the specified range. On average, position and heading accuracy with respect to the desired trajectory was 0.95 m and 4.75° when traveling at 100 mm/s and 1.08 m and 5.05° when traveling at 400 mm/s.

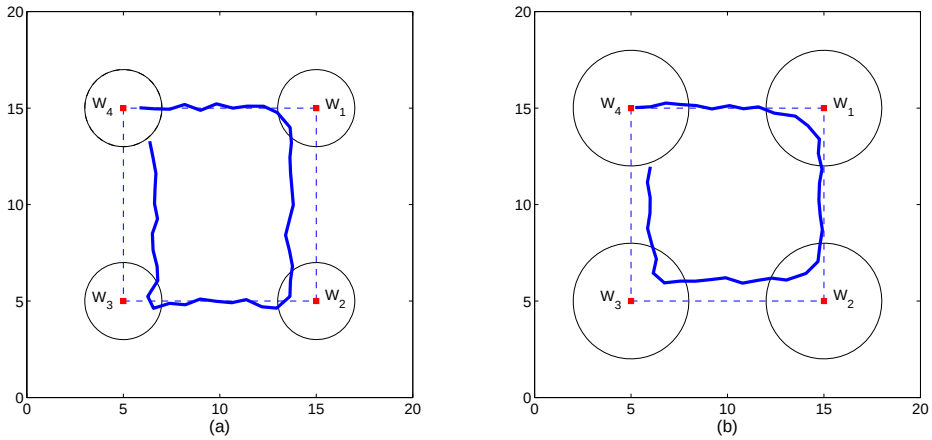


Figure 62: Waypoint navigation average position results when mobile sensor speed is (a) 100 mm/s, and (b) 400 mm/s. The dotted line represents the desired path. Waypoints are marked $W_1 \ldots W_4$, and the the surrounding circles represent the waypoint range of (a) 2 m and (b) 3 m.

Figure 63 displays the outermost and innermost positions along the circuit of the mobile sensor over all runs. These are not individual paths, but bounds on the mobile sensor's movement over all five runs. This shows that one TripNav run does not significantly vary from another.

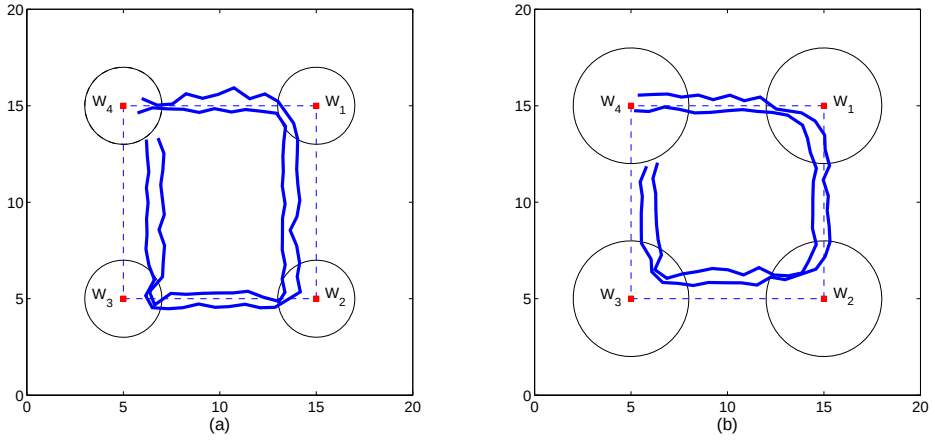


Figure 63: Waypoint navigation outermost and innermost path when mobile sensor speed is (a) 100 mm/s, and (b) 400 mm/s. The dotted line represents the desired path. Waypoints are marked $W_1 \ldots W_4$, and the the surrounding circles represent the waypoint range of (a) 2 m and (b) 3 m.

Latency Analysis

Because the mobile sensor is moving while estimating its position, localization must be performed rapidly, otherwise the mobile node will be in a different location by the time a result is returned. The speed of the entire localization process depends on the latency of each component within the TripNav system, and so we provide a timing analysis of those components here. A latency analysis of the individual TripLoc components is presented in Chapter VI.

Figure 64 shows a sequence diagram for each step in the TripNav control loop, in which two TripLoc anchors are used. Table 11 lists the average and maximum execution times over 100 iterations for the components pictured in Figure 56. Note that TripNav execution time depends on the number of participating TripLoc anchors, because bearing to each anchor is estimated sequentially. A minimum of two anchors is required for triangulation. The accuracy of the localization will improve with the addition of more participating anchors. We therefore provide execution times for three TripLoc scenarios, in which we vary the number of participating anchor arrays between two (the minimum required) and four (the number we use in our real-world evaluation).

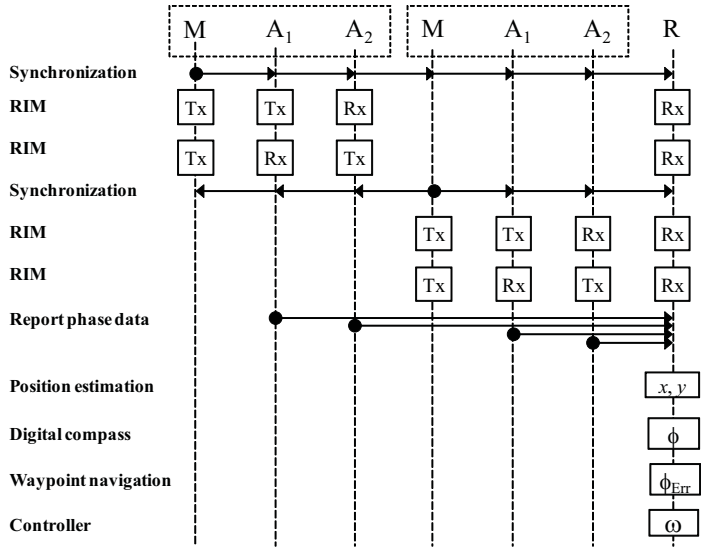


Figure 64: Sequence diagram of the TripNav control loop in which two TripLoc anchors are used.

Component	Average latency (ms)	Maximum latency (ms)
Digital compass	49.61	89.48
Waypoint Navigation	0.45	0.52
Controller	0.64	0.68
TripLoc (2 anchors)	888.72	956.22
TripLoc (3 anchors)	1283.81	1334.99
TripLoc (4 anchors)	1667.76	1734.48

Table 11: Latency of TripNav components.

Note that on average the digital compass takes approximately 50 ms to estimate heading. This estimation latency is in fact a limitation of the compass hardware, which provides heading estimates at a rate of approximately 8 Hz, or 125 ms. The 50 ms latency reflects the average time we must wait for the compass to deliver the next heading estimate. It is also worth noting that for TripLoc, we reduced the phase data routing time from 250 ms to 125 ms, which explains the slight latency reduction compared with the data presented in Chapter VI.

Conclusion

In this chapter, we described a method for waypoint navigation using resource-constrained mobile sensors. We mounted a mote and digital compass to a mobile platform, and used TripLoc as the underlying localization system. All computation and control operations were performed by the mote, and no offline or base station processing was required. Using this system, a mobile sensor was able to navigate a series of four waypoints, completing the circuit with an average trajectory deviation error of 0.95 m and 4.75°.

TripLoc is the key enabler of this system, thanks to its low latency, small memory footprint, and accurate position estimation algorithm. The work presented in this chapter demonstrates that TripLoc is indeed suitable for mobile sensor navigation. This is significant because TripLoc is one of the few RF-based distributed localization techniques that is rapid and accurate enough to support mobile sensor localization.

CHAPTER VIII

CONCLUSION

Spatio-temporal awareness in mobile wireless sensor networks entails new challenges that result from integrating resource-constrained wireless sensors onto mobile platforms. The localization methods and algorithms that provide greater accuracy on larger-footprint mobile entities with fewer resource limitations are no longer applicable. Similarly, centralized and high-latency localization techniques for static WSNs are undesirable for the majority of MWSN applications. In this dissertation, we presented a survey and taxonomy on spatio-temporal coordination for mobile wireless sensor networks. This includes integrating wireless sensors onto mobile platforms, time synchronization for heterogeneous sensor networks, mobile sensor localization, and finally, mobile sensor navigation. We described the problems we have studied in these areas, which includes HSN time synchronization, mobile sensor localization and navigation using RF Doppler shifts (dNav), angular separation determination using RF Doppler shifts and wheel encoder data, a rapid distributed localization service (TripLoc), and a TripLoc-based waypoint navigation system (TripNav).

One of the biggest challenges we currently face with RF propagation is multipath fading. Outdoor urban areas and building interiors are both major sources of multipath, and yet these are places where MWSNs have the greatest utility. At present, our localization techniques will only work outdoors in a wide-open area. An RF-based localization system that provides accurate results in high-multipath environments would be a major step forward. This is a future direction for MWSN localization, and we have already obtained encouraging preliminary results. In [142], we were able to demonstrate that precise RF indoor 1-dimensional tracking is indeed possible, and we are currently investigating how we can extend this technique to two and three dimensions. Such fine-grained RF-based localization would enable mobile sensors to navigate through hallways of burning buildings, help to evacuate shopping malls in the event of an emergency, and monitor the health of patients in every room of the house.

APPENDIX A

LIST OF ACRONYMS

Acronym	Meaning	Acronym	Meaning
A/D	Analog to Digital	PDA	Personal Digital Assistant
ADC	Analog to Digital Converter	PF	Particle Filter
AE	Acoustic Emissions	PI	Proportional-Integral
AOA	Angle of Arrival	PLL	Phase Locked Loop
COTS	Commercial Off The Shelf	PPM	Parts Per Million
CPU	Central Processing Unit	RAM	Random Access Memory
DCO	Digitally Controlled Oscillator	RBS	Reference Broadcast Synchronization
EKF	Extended Kalman Filter	RF	Radio Frequency
ETA	Elapsed Time on Arrival	RIM	Radio Interferometric Measurement
FCC	Federal Communications Commission	RIPS	Radio Interferometric Positioning System
FOA	Frequency of Arrival	RITS	Routing Integrated Time Synchronization
FTSP	Flood Routing Time Synchronization	ROM	Read Only Memory
GPIO	General Purpose IO	RSS	Received Signal Strength
GPS	Global Positioning System	RSSI	Received Signal Strength Indicator
HSN	Heterogeneous Sensor Network	SBE	Sequential Bayesian Estimation
IR	Infrared	SLAM	Simultaneous Localization and Mapping
ISM	Industrial, Scientific, and Medical	SLAT	Simultaneous Localization and Tracking
KF	Kalman Filter	SMC	Sequential Monte Carlo
LOS	Line of Sight	SPS	Samples Per Second
MAC	Media Access Control	TDOA	Time Difference of Arrival
MANET	Mobil Ad hoc Network	TOA	Time of Arrival
ML	Maximum Likelihood	TPSN	Timing Sync Protocol for Sensor Networks
MLE	Maximum Likelihood Estimation	TSS	Timestamp Synchronization
MWSN	Mobile Wireless Sensor Network	UART	Universal Asynchronous Receiver Transmitter
NTP	Network Time Synchronization	UTC	Coordinated Universal Time
OS	Operating System	WiFi	Wireless Fidelity
PC	Personal Computer	WSN	Wireless Sensor Network

Table 12: List of acronyms.

REFERENCES

[1] G. Werner-allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh, "Monitoring volcanic eruptions with a wireless sensor network," *in Proceedings of the Second European Workshop on Wireless Sensor Networks (EWSN)*, 2005.

[2] M. Tubaishat, P. Zhuang, Q. Qi, and Y. Shang, "Wireless sensor networks in intelligent transportation systems," *Wireless Communications & Mobile Computing*, vol. 9, no. 3, pp. 287–302, 2009.

[3] D. Malan, T. Fulford-jones, M. Welsh, and S. Moulton, "CodeBlue: An ad hoc sensor network infrastructure for emergency medical care," in *Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks*, 2004.

[4] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: Scalable coordination in sensor networks," *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking (MobiCom)*, pp. 263–270, 1999.

[5] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," *ACM SIGPLAN Notices*, vol. 35, no. 11, pp. 93–104, 2000.

[6] M. Kushwaha, I. Amundson, P. Volgyesi, P. Ahammad, G. Simon, X. Koutsoukos, A. Ledeczi, and S. Sastry, "Multi-modal target tracking using heterogeneous sensor networks," *Proceedings of the 17th International Conference on Computer Communications and Networks (ICCCN)*, August 2008.

[7] M. Yarvis, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu, and S. Singh, "Exploiting heterogeneity in sensor networks," in *Proceedings of the IEEE International Conference on Computer Communication (INFOCOM)*, vol. 2, Miami, FL, 2005, pp. 878–890.

[8] E. Duarte-Melo and M. Liu, "Analysis of energy consumption and lifetime of heterogeneous wireless sensor networks," in *Proceedings of the IEEE Global telecommunications conference (GLOBECOM)*, vol. 1, 2002, pp. 21–25.

[9] L. Lazos, R. Poovendran, and J. A. Ritcey, "Probabilistic detection of mobile targets in heterogeneous sensor networks," in *Proceedings of the 6th international conference on Information processing in sensor networks (IPSN)*. Cambridge, Massachusetts, USA: ACM, 2007, pp. 519–528.

[10] I. Amundson, M. Kushwaha, X. Koutsoukos, S. Neema, and J. Sztipanovits, "Efficient integration of web services in ambient-aware sensor network applications," in *Proceedings of the 3rd IEEE/CreateNet International Workshop on Broadband Advanced Sensor Networks (BaseNets)*, October 2006.

[11] I. Amundson, B. Kusy, P. Volgyesi, X. Koutsoukos, and A. Ledeczi, "Time synchronization in heterogeneous sensor networks," in *Proceedings of the 4th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, S. N. et al., Ed., vol. LNCS 5067. Santorini, Greece: Springer, June 2008, pp. 17–31.

[12] S. A. Munir, B. Ren, W. Jiao, B. Wang, D. Xie, and J. Ma, "Mobile wireless sensor network: Architecture and enabling technologies for ubiquitous computing," *Proceedings of the International Conference on Advanced Information Networking and Applications Workshops (AINAW)*, vol. 2, pp. 113–120, 2007.

[13] C. Chen and J. Ma, "MEMOSEN: Multi-radio Enabled MObile Wireless SEnsor Network," *Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA)*, vol. 2, pp. 291–295, 2006.

[14] M. Kohvakka, J. Suhonen, M. Kuorilehto, V. Kaseva, M. Hännikäinen, and T. D. Hämäläinen, "Energy-efficient neighbor discovery protocol for mobile wireless sensor networks," *Ad Hoc Networks*, vol. 7, no. 1, pp. 24–41, 2009.

[15] E. Ekici, Y. Gu, and D. Bozdag, "Mobility-based communication in wireless sensor networks," *IEEE Communications Magazine*, vol. 44, no. 7, pp. 56–62, July 2006.

[16] S. Tilak, V. Kolar, N. B. Abu-Ghazaleh, and K.-D. Kang, "Dynamic localization control for mobile sensor networks," in *Proceedings of the IEEE International Workshop on Strategies for Energy Efficiency in Ad Hoc and Sensor Networks*, 2005, pp. 7–9.

[17] I. Amundson, M. Kushwaha, B. Kusy, P. Volgyesi, G. Simon, X. Koutsoukos, and A. Ledeczi, "Time synchronization for multi-modal target tracking in heterogeneous sensor networks," in *Workshop on Networked Distributed Systems for Intelligent Sensing and Control*, Kalamata, Greece, 2007.

[18] I. Amundson, X. Koutsoukos, and J. Sallai, "Mobile sensor localization and navigation using RF doppler shifts," in *Proceedings of the 1st ACM International Workshop on Mobile Entity Localization and Tracking in GPS-less Environments (MELT)*. San Francisco, CA, USA: ACM, 2008, pp. 97–102.

[19] B. Kusy, I. Amundson, J. Sallai, P. Volgyesi, A. Ledeczi, and X. Koutsoukos, "RF doppler shift-based mobile sensor tracking and navigation," *ACM Transactions on Sensor Networks*, vol. 7, no. 1, 2011.

[20] I. Amundson, M. Kushwaha, and X. Koutsoukos, "On the feasibility of determining angular separation in mobile wireless sensor networks," in *Proceedings of the 2nd International Workshop on Mobile Entity Localization and Tracking in GPS-less Environments (MELT)*, R. Fuller and X. Koutsoukos, Eds., vol. LNCS 5801. Springer, 2009, pp. 115–127.

[21] I. Amundson, J. Sallai, X. Koutsoukos, and A. Ledeczi, "Radio interferometric angle of arrival estimation," in *Proceedings of the 7th European Conference on Wireless Sensor Networks (EWSN)*, S. S. et al., Ed., vol. LNCS 5970. Coimbra, Portugal: Springer, 2010, pp. 1–16.

[22] M. Maróti, B. Kusý, G. Balogh, P. Völgyesi, A. Nádas, K. Molnár, S. Dóra, and A. Lédeczi, "Radio interferometric geolocation," *Proceedings of the 3rd international conference on Embedded networked sensor systems (SenSys)*, pp. 1–12, November 2005.

[23] B. Kusy, "Spatio-temporal coordination in wireless sensor networks," *PhD Dissertation, Vanderbilt University*, 2007.

[24] I. Amundson and X. Koutsoukos, "A survey on localization for mobile wireless sensor networks," in *Proceedings of the 2nd International Workshop on Mobile Entity Localization and Tracking in GPS-less Environments (MELT)*, R. Fuller and X. Koutsoukos, Eds., vol. LNCS 5801. Springer, 2009, pp. 235–254.

[25] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins, *Global Positioning System: Theory and Practice, 4th ed.* Springer Verlag, 1997.

[26] L. Fang, P. J. Antsaklis, L. Montestruque, M. B. Mcmickell, M. Lemmon, Y. Sun, H. Fang, I. Koutroulis, M. Haenggi, M. Xie, and X. Xie, "Design of a wireless assisted pedestrian dead reckoning system - the NavMote experience," *IEEE Transactions on Instrumentation and Measurement*, vol. 54, no. 6, pp. 2342–2358, 2005.

[27] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet," pp. 96–107, 2002.

[28] B. Kusý, A. Lédeczi, and X. Koutsoukos, "Tracking mobile nodes using RF doppler shifts," in *Proceedings of the 5th international conference on embedded networked sensor systems (SenSys)*, Sydney, Australia, 2007, pp. 29–42.

[29] "Crossbow MICAz (MPR2400) Radio Module," http://www.xbow.com/.

[30] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, and D. Culler, "Design of a wireless sensor network platform for detecting rare, random, and ephemeral events," *Proceedings of the 4th international symposium on Information processing in sensor networks (IPSN/SPOTS)*, April 2005.

[31] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling ultra-low power wireless research," *Proceedings of the 4th international symposium on Information processing in sensor networks (IPSN/SPOTS)*, p. 48, April 2005.

[32] UC Berkeley, "Mica2," http://www.tinyos.net/scoop/special/hardware#mica2.

[33] K. Dantu, M. Rahimi, H. Shah, S. Babel, A. Dhariwal, and G. S. Sukhatme, "Robomote: enabling mobility in sensor networks," in *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN)*, 2005, p. 55.

[34] J. Friedman, D. C. Lee, I. Tsigkogiannis, S. Wong, D. Chao, D. Levin, W. J. Kaisera, and M. B. Srivastava, "Ragobot: A new platform for wireless mobile sensor networks," in *Proceedings of the International Conference on Distributed Computing in Sensor Systems (DCOSS)*, vol. LNCS 3560. Springer, 2005, p. 412.

[35] S. Bergbreiter and K. S. J. Pister, "CotsBots: An off-the-shelf platform for distributed robotics," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2003, pp. 27–31.

[36] J. H.B. Brown, J. V. Weghe, C. Bererton, and P. Khosla, "Millibot trains for enhanced mobility," *IEEE/ASME Transactions on Mechatronics*, vol. 7, no. 4, pp. 452–461, 2002.

[37] M. B. McMickell, B. Goodwine, and L. A. Montestruque, "MICAbot: a robotic platform for large-scale distributed robotics," in *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2003, pp. 1600–1605.

[38] R. Shah, S. Roy, S. Jain, and W. Brunette, "Data MULEs: modeling a three-tier architecture for sparse sensor networks," in *Proceedings of the 1st IEEE International Workshop on Sensor Network Protocols and Applications*, 2003, pp. 30–41.

[39] G. Wang, G. Cao, T. Porta, and W. Zhang, "Sensor relocation in mobile sensor networks," in *Proceedings of the IEEE International Conference on Computer Communication (INFOCOM)*, 2005, pp. 2302–2312.

[40] B. Liu, P. Brass, O. Dousse, P. Nain, and D. Towsley, "Mobility improves coverage of sensor networks," in *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, 2005, pp. 300–308.

[41] S. Gandham, M. Dawande, R. Prakash, and S. Venkatesan, "Energy efficient schemes for wireless sensor networks with multiple mobile base stations," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM)*.

[42] A. Kansal, A. A. Somasundara, D. D. Jea, M. B. Srivastava, and D. Estrin, "Intelligent fluid infrastructure for embedded networks," in *Proceedings of the 2nd international conference on Mobile systems, applications, and services (MobiSys)*, 2004, pp. 111–124.

[43] J. N. Al-Karaki and A. E. Kamal, "Routing techniques in wireless sensor networks: a survey," *IEEE Wireless Communications*, vol. 11, no. 6, pp. 6–28, 2004.

[44] M. Abolhasan, T. Wysocki, and E. Dutkiewicz, "A review of routing protocols for mobile ad hoc networks," *Ad Hoc Networks*, vol. 2, no. 1, pp. 1–22, 2004.

[45] Q. Wang, M. Hempstead, and W. Yang, "A realistic power consumption model for wireless sensor network devices," in *Proceedings of the 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks (SECON)*, vol. 1, 2006, pp. 286–295.

[46] J. Hightower and G. Borriello, "Location systems for ubiquitous computing," *IEEE Computer*, vol. 34, no. 8, pp. 57–66, 2001.

[47] G. Mao, B. Fidan, and B. D. O. Anderson, "Wireless sensor network localization techniques," *Computer Networks*, vol. 51, no. 10, pp. 2529–2553, 2007.

[48] R. R. Brooks, C. Griffin, and D. S. Friedlander, "Self-organized distributed sensor network entity tracking," *The International Journal of High Performance Computing Applications*, vol. 16, no. 3, 2002.

[49] D. Moore, J. Leonard, D. Rus, and S. Teller, "Robust distributed network localization with noisy range measurements," in *Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys)*, 2004, pp. 50–61.

[50] L. Girod, M. Lukac, V. Trifa, and D. Estrin, "The design and implementation of a self-calibrating acoustic sensing platform," *Proceedings of the 2nd international conference on embedded networked sensor systems (SenSys)*, pp. 71–84, November 2006.

[51] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *Proceedings of the 5th symposium on operating systems design and implementation (OSDI)*, pp. 147–163, 2002.

[52] B. Kusý, P. Dutta, P. Levis, M. Maróti, A. Lédeczi, and D. Culler, "Elapsed time on arrival: a simple and versatile primitive for canonical time synchronization services," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 2, no. 1, pp. 239–251, 2006.

[53] B. Kusý, J. Sallai, G. Balogh, A. Lédeczi, V. Protopopescu, J. Tolliver, F. DeNap, and M. Parang, "Radio interferometric tracking of mobile wireless nodes," *Proceedings of the 5th international conference on Mobile systems, applications and services (MobiSys)*, pp. 139–151, 2007.

[54] A. Harter, A. Hopper, P. Stegglesand, A. Ward, and P. Webster, "The anatomy of a context-aware application," *In Mobile Computing and Networking*, p. 5968, 1999.

[55] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The Cricket location-support system," *Proceedings of the 6th annual international conference on Mobile computing and networking (MobiCom)*, pp. 32–43, Aug. 2000.

[56] M. McCarthy, P. Duff, H. L. Muller, and C. Randell, "Accessible ultrasonic positioning," *IEEE Pervasive Computing*, vol. 5, no. 4, pp. 86–93, 2006.

[57] J. Chen, K. Yao, and R. Hudson, "Source localization and beamforming," *Signal Processing Magazine, IEEE*, vol. 19, no. 2, pp. 30–39, 2002.

[58] A. Lédeczi, A. Nádas, P. Völgyesi, G. Balogh, B. Kusý, J. Sallai, G. Pap, S. Dóra, K. Molnár, M. Maróti, and G. Simon, "Countersniper system for urban warfare," *ACM Transactions on Sensor Networks*, vol. 1, no. 1, pp. 153–177, Nov. 2005.

[59] S. M. Williams, K. D. Frampton, I. Amundson, and P. L. Schmidt, "Decentralized acoustic source localization in a distributed sensor network," *Applied Acoustics*, vol. 67, 2006.

[60] R. Want, A. Hopper, V. Falcao, and J. Gibbons, "The active badge location system," *ACM Transactions on Information Systems*, vol. 10, no. 1, pp. 91–102, 1992.

[61] E. Brassart, C. Pegard, and M. Mouaddib, "Localization using infrared beacons," *Robotica*, vol. 18, no. 2, pp. 153–161, 2000.

[62] J. Kemper and H. Linde, "Challenges of passive infrared indoor localization," in *Proceedings of 5th Workshop on Positioning, Navigation and Communication (WPNC)*, 2008, pp. 63–70.

[63] B. Kusý, G. Balogh, P. Völgyesi, J. Sallai, A. Nádas, A. Lédeczi, M. Maróti, and L. Meertens, "Node-density independent localization," *Proceedings of the 5th international conference on Information processing in sensor networks (IPSN/SPOTS)*, pp. 441–448, Apr. 2006.

[64] K. Römer, "The lighthouse location system for smart dust," in *Proceedings of the 1st International Conference on Mobile Systems, Applications and Services (MobiSys)*, 2003, pp. 15–30.

[65] R. Stoleru, T. He, J. A. Stankovic, and D. Luebke, "A high-accuracy, low-cost localization system for wireless sensor networks," *Proceedings of the 3rd international conference on Embedded networked sensor systems (SenSys)*, pp. 13–26, Nov. 2005.

[66] K. E. Bekris, A. A. Argyros, and L. E. Kavraki, "Angle-based methods for mobile robot navigation: Reaching the entire plane," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, New Orleans, LA, 2004, pp. 2373–2378.

[67] D. Niculescu and B. Nath, "Ad hoc positioning system (APS) using AOA," in *In Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2003, pp. 1734–1743.

[68] J. Friedman, Z. Charbiwala, T. Schmid, Y. Cho, and M. Srivastava, "Angle-of-arrival assisted radio interferometry (ARI) target localization," in *Proceedings of the Military Communications Conference (MILCOM)*, 2008, pp. 1–7.

[69] N. B. Priyantha, H. Balakrishnan, E. D. Demaine, and S. Teller, "Mobile-assisted localization in wireless sensor networks," in *Proceedings of the IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, vol. 1, 2005, pp. 172–183.

[70] J. Caffery, J.J., "A new approach to the geometry of TOA location," in *52nd IEEE Vehicular Technology Conference*, vol. 4, 2000, pp. 1943–1949.

[71] X. Wang, Z. Wang, and B. O'Dea, "A TOA-based location algorithm reducing the errors due to non-line-of-sight (NLOS) propagation," *IEEE Transactions on Vehicular Technology*, vol. 52, no. 1, pp. 112–116, 2003.

[72] A. Gunther and C. Hoene, "Measuring round trip times to determine the distance between WLAN nodes," in *Networking*, vol. LNCS 3462. Springer-Verlag, 2005, pp. 768–779.

[73] H. Lee, M. Wicke, B. Kusy, and L. Guibas, "Localization of mobile users using trajectory matching," in *Proceedings of the first ACM international workshop on Mobile entity localization and tracking in GPS-less environments (MELT)*, 2008, pp. 123–128.

[74] D. Madigan, E. Einahrawy, R. Martin, W.-H. Ju, P. Krishnan, and A. Krishnakumar, "Bayesian indoor positioning systems," *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, vol. 2, pp. 1217–1227, March 2005.

[75] P. Bahl and V. N. Padmanabhan, "Radar: An in-building RF-based user-location and tracking system," *Proceedings of the 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, vol. 2, pp. 775–784, Mar. 2000.

[76] A. Ladd, K. Bekris, A. Rudys, D. Wallach, and L. Kavraki, "On the feasibility of using wireless ethernet for indoor localization," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 3, pp. 555–559, June 2004.

[77] A. Ledeczi, P. Volgyesi, J. Sallai, and R. Thibodeaux, "A novel RF ranging method," in *Proceedings of the 6th Workshop on Intelligent Solutions in Embedded Systems (WISES)*, Regensburg, Germany, July 2008, pp. 1–12.

[78] H.-l. Chang, J.-b. Tian, T.-T. Lai, H.-H. Chu, and P. Huang, "Spinning beacons for precise indoor localization," in *Proceedings of the 6th ACM conference on Embedded network sensor systems (SenSys)*, Raleigh, NC, USA, 2008, pp. 127–140.

[79] D. Niculescu and B. Nath, "DV based positioning in ad hoc networks," *Journal of Telecommunication Systems*, vol. 22, pp. 267–280, 2003.

[80] D. Manolakis, "Efficient solution and performance analysis of 3-D position estimation by trilateration," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 32, no. 4, pp. 1239–1248, 1996.

[81] S. Genchev, P. Venkov, and B. Vidolov, "Trilateration analysis for movement planning in a group of mobile robots," in *Proceedings of the 13th international conference on Artificial Intelligence(AIMSA)*, 2008, pp. 353–364.

[82] P. M. Maxim, S. Hettiarachchi, W. M. Spears, D. F. Spears, J. Hamann, T. Kunkel, and C. Speiser, "Trilateration localization for multi-robot teams," in *Proceedings of the 6th International Conference on Informatics in Control, Automation and Robotics, Special Session on MultiAgent Robotic Systems (ICINCO)*, 2008, pp. 301–307.

[83] J. Esteves, A. Carvalho, and C. Couto, "Generalized geometric triangulation algorithm for mobile robot absolute self-localization," in *Proceedings of the IEEE International Symposium on Industrial Electronics (ISIE)*, 2003, pp. 346–351.

[84] C. McGillem and T. Rappaport, "A beacon navigation method for autonomous vehicles," *IEEE Transactions on Vehicular Technology*, vol. 38, no. 3, pp. 132–139, Aug 1989.

[85] M. Betke and L. Gurvits, "Mobile robot localization using landmarks," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 2, pp. 251–263, Apr 1997.

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

[87] L. Hu and D. Evans, "Localization for mobile sensor networks," in *Proceedings of the 10th annual international conference on Mobile computing and networking (MobiCom)*, 2004, pp. 45–57.

[88] P. Zhang and M. Martonosi, "Locale: Collaborative localization estimation for sparse mobile sensor networks," in *Proceedings of the 7th international conference on Information processing in sensor networks (IPSN)*, April 2008, pp. 195–206.

[89] S. M. Kay, *Fundamentals of Statistical Signal Processing, Volume I: Estimation Theory*. Prentice Hall, 1993.

[90] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[91] X. Kuang and H. Shao, "Maximum likelihood localization algorithm using wireless sensor networks," in *Proceedings of the 1st International Conference on Innovative Computing, Information and Control (ICICIC)*, vol. 3, 2006, pp. 263–266.

[92] M. Mendalka, L. Kulas, and K. Nyka, "Localization in wireless sensor networks based on zigbee platform," in *Proceedings of the 17th International Conference on Microwaves, Radar and Wireless Communications (MIKON)*, 2008, pp. 1–4.

[93] V. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello, "Bayesian filtering for location estimation," *IEEE Pervasive Computing*, vol. 2, no. 3, pp. 24–33, 2003.

[94] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[95] G. Welch and G. Bishop, "An introduction to the Kalman filter," Department of Computer Science, University of North Carolina at Chapel Hill, Tech. Rep. TR 95-041, 2004.

[96] Texas Instruments, "CC1000: Single chip very low power RF transceiver," http://focus.ti.com/docs/prod/folders/print/cc1000.html, 2007.

[97] M. Maróti, B. Kusý, G. Simon, and A. Lédeczi, "The flooding time synchronization protocol," *Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys)*, pp. 39–49, Nov. 2004.

[98] H. Sizun, *Radio wave propagation for telecommunication applications*. Springer, 2004.

[99] H. Hashemi, "The indoor radio propagation channel," *Proceedings of the IEEE*, vol. 81, no. 7, pp. 943–968, 1993.

[100] Skilligent, "Visual Localization System," http://www.skilligent.com/.

[101] iRobot, "Roomba vacuum cleaning robot," http://www.irobot.com.

[102] P. Völgyesi, A. Nádas, X. Koutsoukos, and A. Lédeczi, "Air quality monitoring with sensormap," in *Proceedings of the 7th international conference on information processing in sensor networks (IPSN)*, 2008, pp. 529–530.

[103] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan, "The pothole patrol: using a mobile sensor network for road surface monitoring," in *Proceedings of the 6th international conference on Mobile systems, applications, and services (MobiSys)*, 2008, pp. 29–39.

[104] Federal Communications Commission, "Enhanced 911," http://www.fcc.gov/pshs/services/911-services/enhanced911/.

[105] P. Völgyesi, G. Balogh, A. Nádas, C. Nash, and A. Lédeczi, "Shooter localization and weapon classification with soldier-wearable networked sensors," *Proceedings of the 5th International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pp. 113–126, 2007.

[106] P. Corke, S. Hrabar, R. Peterson, D. Rus, S. Saripalli, and G. Sukhatme, "Autonomous deployment and repair of a sensor network using an unmanned aerial vehicle," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2004, pp. 3602–3609.

[107] E. Taylor, "The haven-finding art: a history of navigation from Odysseus to Captain Cook," *Elsevier Scientific Publishing, New York, NY, USA*, 1971.

[108] M. Franz and H. A. Mallot, "Biomimetic robot navigation," *Robotics and autonomous Systems*, vol. 30, pp. 133–153, 2000.

[109] T. S. Levitt and D. T. Lawton, "Qualitative navigation for mobile robots," *Artificial Intelligence*, vol. 44, no. 3, pp. 305–360, 1990.

[110] O. Trullier, S. I. Wiener, A. Berthoz, J. arcady Meyer, and P. M. Berthelot, "Biologically-based artificial navigation systems: Review and prospects," *Progress in Neurobiology*, vol. 51, pp. 483–544, 1997.

[111] T. Prescott, "Spatial representation for navigation in animals," *Animal Behavior*, vol. 4, pp. 85–123, 1996.

[112] J. Borenstein, H. R. Everett, L. Feng, and D. Wehe, "Mobile robot positioning sensors and techniques," *Journal of Robotic Systems, Special Issue on Mobile Robots*, vol. 14, no. 4, pp. 231–249, 1997.

[113] J. Elwell, "Inertial navigation for the urban warrior," in *Proceedings of SPIE*, R. Suresh, Ed., vol. 3709, Jul. 1999, pp. 196–204.

[114] M. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, June 2001.

[115] C. Taylor, A. Rahimi, J. Bachrach, H. Shrobe, and A. Grue, "Simultaneous localization, calibration, and tracking in an ad hoc sensor network," in *Proceedings of the 5th International Conference on Information Processing in Sensor Networks (IPSN)*, Nashville, TN, 2006, pp. 27–33.

[116] K. Römer, P. Blum, and L. Meier, "Time synchronization and calibration in wireless sensor networks," in *Wireless Sensor Networks*, I. Stojmenovic, Ed. Wiley and Sons, 2005.

[117] J. Sallai, B. Kusý, A. Lédeczi, and P. Dutta, "On the scalability of routing-integrated time synchronization," *Proceedings of the 3rd European Workshop on Wireless Sensor Networks (EWSN)*, Feb. 2006.

[118] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems (SenSys)*, 2003, pp. 138–149.

[119] L. Girod, V. Bychkovsky, J. Elson, and D. Estrin, "Locating tiny sensors in time and space: A case study," in *Proceedings of the 2002 IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD)*, 2002, p. 214.

[120] K. Römer, "Time synchronization in ad hoc networks," in *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, ACM, Oct. 2001, pp. 173–182.

[121] M. Huang, L. Jiang, P. K. Liaw, C. R. Brooks, R. Seeley, and D. L. Klarstrom, "Using acoustic emission in fatigue and fracture materials research," *JOM*, vol. 50, no. 11, 1998.

[122] M. Kushwaha, S. Oh, I. Amundson, X. Koutsoukos, and A. Ledeczi, "Target tracking in heterogeneous sensor networks using audio and video sensor fusion," in *Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, 2008.

[123] M. Kushwaha, S. Oh, I. Amundson, X. Koutsoukos, and A. Ledeczi, "Target tracking in urban environments using audio-video signal processing in heterogeneous wireless sensor networks," in *Proceedings of the 42nd IEEE Asilomar Conference on Signals, Systems and Computers*, 2008.

[124] M. Kushwaha, S. Oh, I. Amundson, X. Koutsoukos, and A. Ledeczi, "Tracking in urban environments using sensor networks based on audio-video fusion," *Handbook of Ambient Intelligence and Smart Environments (AISE), Springer*, 2008.

[125] J. Elson and K. Römer, "Wireless sensor networks: a new regime for time synchronization," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 1, pp. 149–154, Jan. 2003, presented at the Workshop on Hot Topics in Networks (HotNets-I), October 2002.

[126] D. L. Mills, "Modelling and analysis of computer network clocks," Electrical Engineering Department, University of Delaware, Tech. Rep. 92-5-2, 1992.

[127] D. L. Mills, "Internet time synchronization: The network time protocol," *IEEE Transactions on Communications*, vol. 39, no. 10, pp. 1482–1493, 1991.

[128] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors," *Proceedings of the ninth international conference on achitectural support for programming languages and operating systems (ASPLOS-IX)*, pp. 93–104, Nov. 2000.

[129] MobileRobots, "Pioneer P3-DX," http://www.activrobots.com/ROBOTS/p2dx.html.

[130] K. Altun and A. Koku, "Evaluation of egocentric navigation methods," in *Proceedings of the IEEE International Workshop on Robot and Human Interactive Communication (ROMAN)*, August 2005, pp. 396–401.

[131] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler, "The nesC language: a holistic approach to networked embedded systems," *Proceedings of the ACM SIGPLAN conference on programming language design and implementation (PLDI)*, pp. 1–11, June 2003.

[132] P. Biswas, H. Aghajan, and Y. Ye, "Semidefinite programming based algorithms for sensor network localization," in *Proceedings of the 39th Asilomar Conference on Signals, Systems and Computers*, 2006, pp. 188–220.

[133] J. N. Ash and L. C. Potter, "Robust system multiangulation using subspace methods," in *Proceedings of the 6th international conference on information processing in sensor networks (IPSN)*, 2007, pp. 61–68.

[134] P. Rong and M. Sichitiu, "Angle of arrival localization for wireless sensor networks," in *Proceedings of the 3rd Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2006, pp. 374–382.

[135] A. Nasipuri and R. el Najjar, "Experimental evaluation of an angle based indoor localization system," in *Proceedings of the 4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, 2006.

[136] J. N. Ash and L. C. Potter, "Sensor network localization via received signal strength measurements with directional antennas," in *Proceedings of the Allerton Conference on Communication, Control, Computing*, 2004, pp. 1861–1870.

[137] K. Kendig, *Conics*. Mathematical Association of America, 1938.

[138] K. Doğançay, "Bearings-only target localization using total least squares," *Signal Processing*, vol. 85, no. 9, pp. 1695–1710, 2005.

[139] J. Carr, *Practical Antenna Handbook, Fourth Edition*. McGraw Hill, 2001.

[140] iRobot, "Create programmable robot," http://www.irobot.com/.

[141] Honeywell, "HMR3300 digital compass," http://www.magneticsensors.com/.

[142] J. Sallai, I. Amundson, A. Ledeczi, X. Koutsoukos, and M. Maroti, "Using RF received phase for indoor tracking," in *Proceedings of the Workshop on Hot Topics in Embedded Networked Sensors (HotEmNets)*, 2010.