IMPLEMENTATION OF COLLABORATIVE RF LOCALIZATION USING A SOFTWARE-DEFINED RADIO NETWORK

THESIS

Augustine A. Honore, 1Lt, USAF

AFIT/GE/ENG/09-20

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/ENG/09-20

IMPLEMENTATION OF COLLABORATIVE RF LOCALIZATION USING A SOFTWARE-DEFINED RADIO NETWORK

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

Augustine A. Honore, BSEE

First Lieutenant, USAF

March 2009

AFIT/GE/ENG/09-20

IMPLEMENTATION OF COLLABORATIVE RF LOCALIZATION USING A SOFTWARE-DEFINED RADIO NETWORK

Augustine A. Honore, BSEE

First Lieutenant, USAF

Approved:

[illegible]
Capt Ryan W. Thomas, PhD (Chairman) — 17MAR09 Date

[illegible]
LtCol Stuart H. Kurkowski, PhD (Member) — 17Mar 09 Date

[illegible]
Dr. Richard K. Martin (Member) — 17Mar2009 Date

AFIT/GE/ENG/09-20

Abstract

This thesis investigates the use of collaboration between sensor nodes that were tasked with localizing a radio frequency emitter. Localization is a necessary component for dynamic spectrum access. Using a set of software-defined radios as our sensors and a received signal strength-based maximum likelihood localization algorithm, we successfully localized transmitting nodes based on their received signal strength. Our experiment was conducted outdoors using a flexible topology that could be shaped into 21 sub-topologies that varied in size, and orientation with respect to the transmitters. This was made possible through application of a time shift concept and a post-processing technique. We were able to compare our real world results with the simulated results of the same topologies. Although our simulation results did not fully comply with our real world results, we observed some common trends regarding effective topology design.

Acknowledgments

In the beginning, there was Reginald Cooper, Ruolin Zhou, Dan Sundersingh, and Omer Mian. Each gave their time to help me get started with my research, and I am thankful for it. Without their efforts, I would not have made it as far as I have with the radios.

I would also like to thank Dr. Richard Martin for sharing his insight and his research, and for offering clever suggestions to help improve the value of my work. Thanks are due to LtCol Stuart Kurkowski for helping me find my writing voice – a key skill that I will continue to polish. Thanks also go to Maj Michael Saville for his patience and willingness to help me learn about the joys of the EM spectrum and antenna theory.

I owe a large debt for the combined support of Luis Oquendoclass, Benjamin Ramsey, and Anthony Larweck. Thank you for supporting me at every stage of the game and for helping me get through in one piece. Because of it, I am a better person. And my greatest appreciation goes to Capt Ryan Thomas for giving me the opportunity to break new ground and for providing the tools to get things done. Thank you all.

Augustine A. Honore

Table of Contents

List of Figures

List of Tables

IMPLEMENTATION OF COLLABORATIVE RF LOCALIZATION USING A SOFTWARE-DEFINED RADIO NETWORK

I. Introduction

1.1 Motivation

Although there used to be a general perception that the radio spectrum in the United States was becoming over-crowded, we now know that this perception was actually due to a spectrum access problem [1]. The Federal Communication Commission's (FCC) traditional approach to spectrum allocation worked well for a time when radio technologies were simpler. But compartmentalized spectrum assignment, which was once the solution, has now become a serious issue as the demand for available spectrum increases among an ever-growing number of users [2], [3], [4], [5].

Practical access of the unused spectra (also called whitespace), whose availability shifts dynamically in space and time, requires an adaptive solution. In the last decade, a bold solution was proposed – an autonomous agent that proactively makes decisions to assist a user – in order to make spectrum access and other communication-based tasks feasible despite dynamic and constrained operating environments. The cognitive radio (CR), as introduced by Mitola and Maguire [6], is centered entirely on a sole user, acting as a personal assistant that delivers end-user services through its ability to observe, adapt, and learn.

Dynamic spectrum access, however, is not a problem best solved through individual effort. A CR's estimation of its current radio frequency (RF) environment has been shown to become significantly more accurate when performed in cooperation with other CRs [7], [8]. A network of collaborating cognitive radios (a cognitive network) benefits from a shared representation of its RF environment by creating a more complete depiction of its dynamic surroundings. An essential part of this RF topography process identifies the presence of primary users, their spatial locations, and their antenna patterns, among other characteristics. In this way, primary users may continue to operate unimpeded, and the collaborating CRs (secondary users) can still productively share the limited medium.

1.2 Background

Previous research has examined the process of characterizing the RF environment by mapping spectrum usage in space, time, frequency, and code. The 5.1 dimensional RF topography developed by Martin and Thomas [9] uses simulation results to demonstrate their localization algorithm which identifies the presence, positions, and antenna patterns of primary users within a search space populated by CR nodes cooperating in a noisy environment. Using the received signal strength (RSS) obtained at each receiving sensor and known receiver positions, they have demonstrated how their algorithm can be used to improve decisions on spectrum availability in a dynamic spectrum access system.

1.3 Research Objectives

The objectives of this research are: to investigate whether Martin's proposed source localization algorithm [9] can be implemented in a real-time environment using a flexible hardware testbed, to examine the accuracy of source position estimates using the algorithm through hardware experimentation, and to compare the results obtained in real-world experiments against those achieved through simulation. Above all, this research sought to demonstrate that collaborative localization among cognitive radios enables network-wide dynamic spectrum access by helping to form a shared representation of the RF environment.

1.4 Research Scope

Our experiments were conducted using receivers and transmitters operating within the FM band (88 MHz to 108 MHz). The transmitting and receiving nodes were configured with omnidirectional monopole antennas. None of the nodes were mobile. The RF localization performed in our study was entirely RSSI-based; it did not incorporate any other localization approaches such as time-difference of arrival (TDOA) and angle of arrival (AOA). We limited our position estimation to two dimensions. Also, we chose to focus on the data collection process and not on the protocols that govern data exchange.

1.5 Assumptions and Limitations

Some assumptions were made and a few limitations were met in order to reach a purposeful end to this research. For example, all of the sensing nodes were assumed to be cooperating with trustworthy peers. Also, it was assumed that the nodes were able to exchange their information over a low-bandwidth, reliable channel that was set up beforehand. Although the simulations performed by Martin and Thomas included many randomly distributed sensing nodes, hardware costs limited our experiments to only a few sensing nodes and even fewer transmitting nodes.

1.6 Thesis Document Organization

The remaining sections of this document are arranged in the following order. Chapter 2 provides background information on the systems, ideas, and techniques used to perform our experiment. It also discusses the research of others in the field and how they contributed to this formal study. Chapter 3 details the sensor characterization process we used to ensure a homogeneous network and discusses the design of our experiment. Chapter 4 presents our performance criteria, our results, and provides a comparison between experimental results and simulation results. Chapter 5 summarizes our findings and lays a foundation for future work. The appendices are reserved for expanding on ideas that were briefly mentioned in the chapters, and several resources for any individual pursuing a similar line of research.

II. Background

2.1 Introduction

The purpose of this chapter is to provide background information about concepts directly related to our research and to discuss the efforts of others. Much of our research is described by the following subject areas: cognitive radio, cognitive networks, node localization, wireless sensor networks, and multisensor data fusion. In Section 2.2 we give a history of the cognitive radio and cognitive network by first exploring the software-defined radio. Section 2.3 provides a comparison of several wireless node localization techniques that are commonly used. As a core discipline of multi-agent collaboration, we define the components of a generalized wireless sensor network in Section 2.4. In Section 2.5 we expand this definition with a survey of the most relevant data fusion models and architectures. Each section provides a summary of research efforts within the field.

2.2 Cognitive Radio and Cognitive Networks

The beginnings of the cognitive radio concept are rooted in the areas of radio communications and artificial intelligence. These concepts formed the foundations of a cognitive network. In this section we present the ideas and events which led to the creation of both cognitive systems, and explain some of the ongoing research being done to mature these technologies.

The early 20th century work of Marconi gave the world a new way to communicate using what was then called a "wireless telegraph." For more than a century, radio technology has matured into an indispensible tool that permeates all areas of our lives. The radio that once brought *True Detective Mysteries* to numerous listeners on Sunday afternoons, and President Roosevelt's famous *Fireside Chats* (among others) has grown into a nearly trillion dollar business whose effects are seen in diverse areas from public safety, to personal communication, and everything in between [10], [11].

It was not until the mid-1980s that digital communications began to shape radio technology by combining hardware with the flexibility of software. One of the first major breakthroughs was the military software radio, SpeakEasy [12]. Developed at Rome Air Force Base, New York, it was intended to be a multiband, multimode, interoperable solution to the proliferation of incompatible radio units – a logistical nightmare. The military software radio was designed in response to the Department of Defense's long-standing question: 'How can the military ensure communication with its latest allies and global support structure, deny interception by [its] current enemies, take advantage of the rapid technology changes, and control the costs of military spending?' [12] The modularity, open architecture design, and upgradeability of the SpeakEasy system helped to pave the way for commercial development of what would be known as software-defined radio – a term coined by Mitola in his 1992 paper [13].

The software-defined radio (SDR) evolved from its military communications roots into a more accessible tool for commercial applications such as cellular infrastructure systems [14], [15]. This evolution was made in part by Wayne Bosner, of the Air Force Research Laboratories (AFRL) who founded the SDR Forum. Working in conjunction with the Institute of Electrical and Electronic Engineers (IEEE) P1900.1 group, they set out to develop hardware and software standards that would help ensure interoperability among all SDRs developed in industry, worldwide [16], [17]. Their pursuits brought together the software architecture, microprocessor, spectrum policy, and digital signal processing fields.

Then, in 1999 Mitola and Maguire formally introduced the cognitive radio (CR) [6] where they defined what would be an extension of the SDR that would use its awareness of internal and external influences in order smartly interact with the RF environment. The digital communications and artificial intelligence communities took an interest because much of the existing technology could be used to implement one almost immediately. Finding an expanded use for the CR concept, Thomas proposed a more collaborative agenda [18] in which networked devices would use their situation awareness to fulfill larger, network-wide objectives and thus realize the concept of an adaptive data network. Introduced as the Cognitive Network (CN), focus was shifted away from the individual device (or user) towards broader end-to-end decisions and goals.

2.3 Node Localization Techniques

Node localization is the process of determining position information for various wireless nodes in a network. Radio Frequency (RF) localization techniques are those methods that use signal measurements and signal processing to calculate position information for wireless nodes. The various RF localization techniques that exist today are unique according to their signal-measurement focus. For example, one approach measures the strength of a received signal in terms of a voltage or power. Other approaches use signal propagation time. Another uses incidence angles of received signals as they enter an array of antennas. Each of these major approaches has its own strengths and weaknesses which we compare below.

2.3.1 Received Signal Strength Indicated

Received Signal Strength Indicated (RSSI) is a measure of how strong a signal is when it arrives at a sensor. The Received Signal Strength (RSS, henceforth RSSI) is commonly taken as a voltage measurement, or equivalently calculated as a signal power (e.g. the magnitude squared). Measurements can be made from acoustic, RF or other types of signals without infringing on bandwidth or requiring complex hardware. However, RSS measurements, are known to vary unpredictably usually because of operating environment conditions [19]. The most influential sources of error are due to multipath propagation and shadowing. Multipath is a phenomenon that destructively (or constructively) combines signals of differing amplitude and phase orientations that have traversed multiple paths prior to arriving at the receiver. Shadowing is the attenuation that results when a signal is forced to go through or bend around obstacles such as walls or trees. Despite these hazards, the relative simplicity and low cost of RSSI-based techniques make them attractive solutions for localization tasks.

When RSSI values are taken using a *range-aware* approach, an effective propagation loss can be calculated at the receiving node given a known transmission power. Theoretical and empirical models can be applied to convert the propagation loss into a radial distance estimate [20]. However, when taken using a *range-free* approach, which makes no assumption about distance information, environmental effects can be significantly reduced as more sensing nodes are allowed to participate in the estimation process. In general, range-free approaches require anchor nodes – nodes that "know" their own position – that support regular (position-unaware) nodes in order to remotely sense a signal emitter.

2.3.2 Time of Arrival

Time of Arrival (TOA) is the measured time at which a *known* signal first arrives at a receiver. This measurement includes the time of transmission (the time it takes an RF source to "put" a signal into the environment) and the propagation delay (the time it takes the signal to move from a source antenna to a receiver antenna). The TOA is determined by calculating the cross-correlation between the received signals and the known transmitted signal. The location of the largest cross-correlation peak indicates when the line-of-sight

(LOS) signal arrived, and yields a time delay when taken with respect to a reference time. The location and height of the peak are greatly influenced by additive noise, which degrades the peak; and by self-interference from multipath signals, which obscures the peak of the LOS signal [19]. The separation distance between the transmitter and receiver is estimated by multiplying the time delay by a known propagation speed such as the speed of light or the speed of sound. Since TOA approaches are based on accurate timing, they generally require more sophisticated hardware and an absolute time reference.

2.3.3 Time Difference of Arrival

Time Difference of Arrival (TDOA) techniques, on the other hand, are relatively immune to timing errors because they calculate a signal's time delay by using the *difference* in arrival times of the same known signal received by two antennas. Thus, any internal clock bias experienced by either sensor is eliminated because the difference calculation ignores an absolute time reference [19]. This gives way for a less-costly asynchronous localization approach because specialized timing devices are unnecessary. Unlike the TOA method, which uses the time delay and propagation speed to calculate a distance, TDOA measurements define (or are solutions to) hyperbolas that lie between the transmitting node and the receiving node. When another TDOA measurement is performed by a different pair of sensing nodes, an additional hyperbola is created. The point at which the two hyperbolas intersect indicates the position estimate [20]. Position estimates using TDOA have been shown to yield better performance than TOA methods particularly in multipath environments [21].

2.3.4 Angle of Arrival

Rather than providing distance information, Angle of Arrival (AOA) measurements identify the direction of origin for a signal of interest. Using a specifically designed antenna array placed on a sensing node, TOA measurements and signal processing techniques are applied to calculate an arrival angle with respect to the sensor's orientation. Unlike the TDOA approach, direction estimate accuracy is dependent on a clear LOS path between the transmitter and receiver antennas [21]. Angle information can also be used to perform position estimates by calculating the point of intersection between two lines, drawn from two directional antennas, occurring at angles with respect to some reference orientation [22]. However, using angle information alone to determine position is not a common practice for RF localization. Instead, AOA measurements are used to supplement other localization techniques.

2.4 Wireless Sensor Networks

Wireless sensor networks (WSNs) are self-organizing, ad hoc networks made up of a large number of nodes that measure things. As such, nodes are typically designed to be low-cost, low-power, and capable of communicating over short distances. The general premise is to be able to observe phenomena by deliberately placing (or scattering) a group of collaborating nodes onto an area of interest and to have them transmit the sensed data wherever it is needed. Being wireless, it reduces installation costs; and being ad hoc, nodes may be removed or added just as easily. Applications for wireless sensor networks span many disciplines including the military [23], [24]; the environment [25],[26]; human health [27]; and commercial industry [28]. For example, an array of nodes can be distributed in an office building to measure temperature and human traffic in order to smartly conserve energy used for heating and lighting, which account for more than 50% of electricity consumed by office environments [29]. A related case study is given in [30].

As depicted in Figure 1, a sensor node is composed of four basic components: a sensing unit, a processing unit, a transceiver unit, and a power unit [31]. The sensing unit's sensor collects observed phenomena (temperature, humidity, pressure, etc.) as analog signals, and the analog to digital converter (ADC) digitizes the signals so that they can be processed. The transceiver links a sensing node to the other nodes in its immediate area so that data can be exchanged over the multi-hop network. Routing and collaboration decisions are calculated in the processing unit. The most important component, however, is the power unit. It governs all of the sensor's processes and is most often the leading hardware constraint.

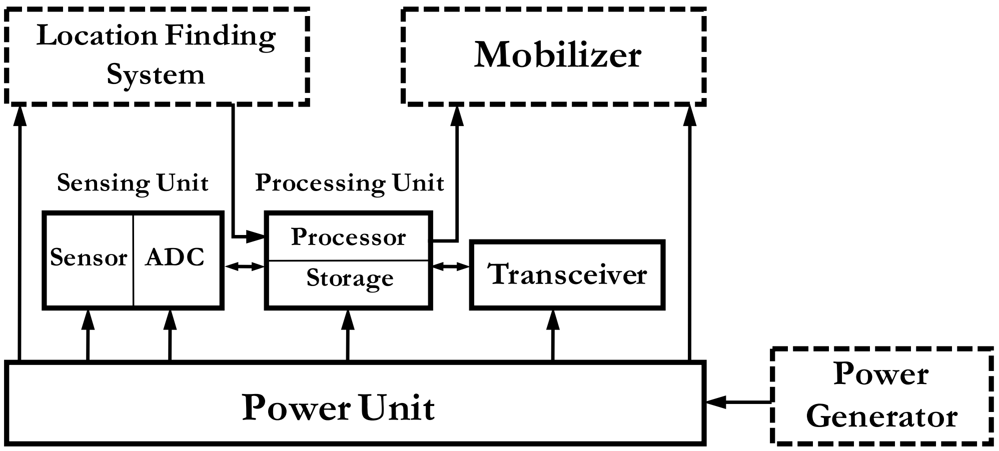


Figure 1: Components of a sensor node [31]

When being used for more specialized applications, sensor nodes can be made to include several other components (dotted outline in Figure 1). Miniature solar cells, vibration energy harvesters or other energy scavenging methods help reduce the power constraint and extend sensor persistence. Mobilizers allow a sensor to physically relocate itself (usually by crawling, rolling, or bounding). A location finding system enables a sensor to calculate its position relative to other sensors, and when used cooperatively, can help localize the source of phenomena (such as the position of a nearby moving tank).

The key to a WSN's success is a robust communication foundation. Depending on the sensing task, nodes may need to use protocols that combine power and routing awareness so that the least amount of power and bandwidth are used regardless of the amount of data that needs to be relayed [31]. Proactive [32] and reactive [33] routing algorithms can be applied to suit the type of sensor network so that the nodes can still cooperate effectively despite transient link states within a multi-hop network environment.

There are times when an end-user will need to retrieve more specific data from particular sections of a WSN or from several independent WSNs. Keeping this in mind, the authors of [34] address the importance of sharing sensor-derived data with external users. They propose a sensor network registry architecture whose usefulness they liken to a good web search engine that presents the most relevant results to a user query. When searching for a specific sensor network, there are two preferred methods: information gathering by collection, and information gathering by registration. The former is akin to a web crawler – an automated software agent that methodically searches web pages and *pulls* data to create entries for a search engine index. The latter (and their preferred) method takes into account the independence of sensor networks by allowing them to *push* data according to their own access policies.

According to their architecture, a sensor network registry would reply to a user's query and would be controlled by a sensor network operator (who would essentially establish general user permissions). Information about the sensor network is stored within the registry and is fetched by a query processor. Park, *et al.* insist that the query-reply process is relatively simple, but instead the difficulty lies in determining which information the sensor network registry should maintain, and how this information should flow within the system.

Their approach to determining the appropriate information to be stored is based on the types of queries that can be posed. By examining all 31 combinations of Who, What, When, Where, Why, and How (5W1H), and their significance to the user and operator, the authors formally establish a set of usable query parameters. Their parameters – operator, location, role, and sensor type – form the basis for an expandable "query grammar." In this way, sensor network queries can be tailored to be as general or specific as necessary, and the best possible answers can be provided.

2.5 Multisensor Data Fusion

Multisensor data fusion is the application of processing and reduction techniques to combine data from multiple sensors and various knowledge sources. The objective is to provide a better understanding of the phenomena under examination than what could be achieved by the use of a single sensor [35]. In the early 1980's, the U.S. military recognized a need to automate information processing for location, tracking, and identification of military entities such as tanks, missiles, and aircraft. By 1986 the Joint Directors of Laboratories (JDL) Data Fusion Working Group was formed to establish a fusion process model and a common language for military researchers and system developers to share.

The JDL data fusion process model shown in Figure 2 identifies five levels of data refinement that are applied iteratively; each level builds on the previous. From sensory data, entities are identified and then compared to reveal any relationships among them. Relationships form the basis of hypotheses which can be used to fulfill simple objectives (such as enhancing noisy surveillance footage) to more complex objectives (such as predicting enemy intent). Borrowing some of the ideas established by the JDL model, several other data fusion models were developed in the years following to fulfill military and non-military data processing needs [36], [37], [38]. Some of these needs include target tracking [39], autonomous robotics [40], and biomedical imaging [41].

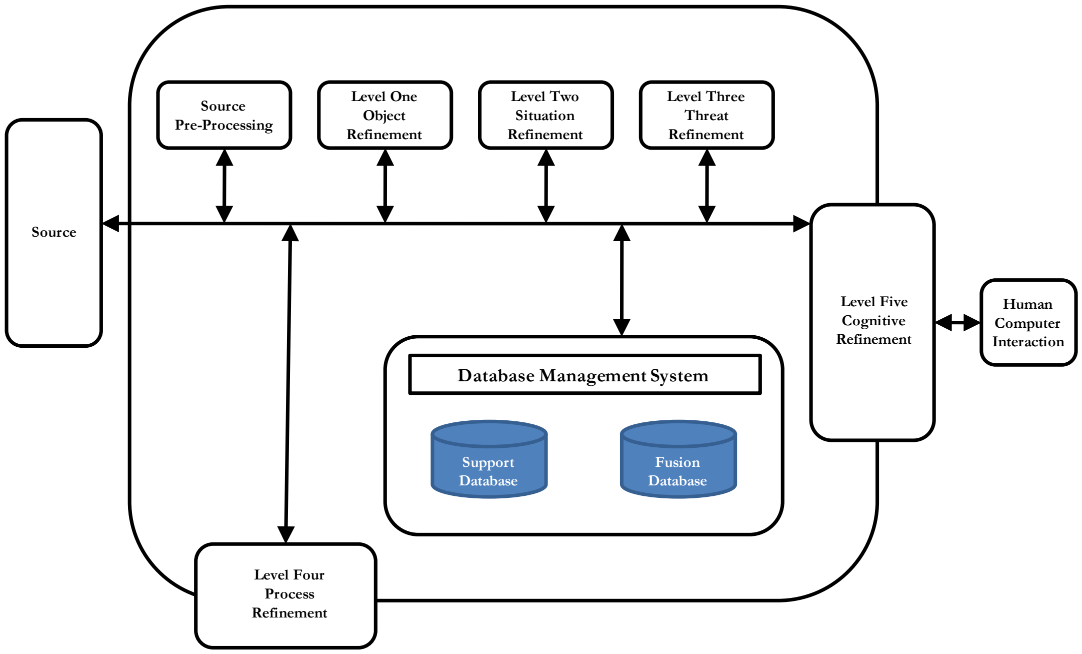

Source
Source Pre-Processing
Level One Object Refinement
Level Two Situation Refinement
Level Three Threat Refinement
Level Five Cognitive Refinement
Human Computer Interaction
Database Management System
Support Database
Fusion Database
Level Four Process Refinement

Figure 2: The JDL Fusion Model [42]

Although data fusion *models* specify the order and types of processes required for various data fusion applications, data fusion *architectures* are selected to specify how the sensing nodes will share their data, where the data is processed, and to what degree data is reduced. Traditionally, military data fusion architectures have been centralized – the sensing nodes transmit their raw data to be processed and reduced at a central location. Centralized fusion architectures usually demand a large amount of bandwidth. Decentralized architectures implement some data reduction (such as coordinate translation and image preprocessing) at the sensing nodes prior to transmitting. Although the raw data is reduced to state vectors, thereby reducing the bandwidth requirement, all subsequent processing is forced to rely on approximations made at the sensing nodes. Hybrid fusion architectures offer flexibility to choose a centralized or decentralized approach in response to network, data fidelity, or processing constraints. Hybrid architectures offer the flexibility of being able to command sensors to send raw data or reduced data as the situation requires. Thus, the bandwidth needed to transmit data and the power required to process data would increase and decrease appropriately. However, this flexibility comes at the price of process monitoring overhead which is required to determine when either operating mode is appropriate [42].

Multisensor data fusion yields several qualitative and quantitative benefits. Generally, an array of sensors provides extended spatial and temporal coverage over an area or a phenomenon. As a result, the probability of successfully detecting objects and events is increased. Joint information from multiple sensors reduces the set of hypotheses about a target or event, thus reducing ambiguity [43]. Particularly among sensors of the same type, multisensor data fusion results in improved resolution.

In order to reap the benefits of multisensor data fusion, there are several things to consider - most of which should be introduced early on in the system design phase. Some of these considerations include:

- There is no substitute for a good sensor.
- Downstream processing cannot make up for errors (or failures) in upstream processing.
- There is no perfect fusion algorithm that is optimal under all conditions.
- The data fusion process is not static but rather iterative and dynamic, and continually in need of refinement [43].

Regardless of the size or application intended for a data fusion system, these and other ideas must be seriously considered to avoid inaccurate estimation and poor data interpretation.

2.6 Summary

In this chapter, we provided background information for several key concepts that related to our research. We began with a brief history behind the cognitive radio and cognitive network. Then, we compared various node localization approaches used today. We also surveyed research within the wireless sensor networks and multisensor data fusion fields.

III. Methodology

3.1 Introduction

The purpose of this chapter is to describe the tools and processes used to conduct our experiments. Although we performed experiments using two different approaches – a real-time approach and a post-processing approach – only the latter will be discussed here since it is the more mature of the two. However, we address the real-time approach in Appendix B and offer some suggestions to improve it.

In Section 3.2 we introduce our major tools: the GNU Radio Development software and the Universal Software Radio Peripheral (USRP). Section 3.3 introduces our data collection and data reduction methodologies. The hardware characterization procedure we used prior to experimenting is outlined in Section 3.4. Our node localization algorithm is explained in Section 3.5. And in Section 3.6 we provide details regarding our collaboration experiments.

3.2 The GNU Radio Development Software and Universal Software Radio Peripheral

Together the GNU Radio development software and USRP software-defined radio form the core components of our research implementation. In the following sections we give a brief history of both components, and explain two important parameters that governed how RF signals were captured – the decimation rate and the Fast Fourier Transform (FFT) size.

3.2.1 GNU Radio

GNU Radio is a free [44] software development toolkit specializing in signal processing and is maintained by Eric Blossom. It was originally conceived as a means to acquire high-definition television signals. Over time, it has evolved into an empowering tool that helps people learn about and explore new ways of using the electromagnetic (EM) spectrum [45]. In the four years since its creation, GNU Radio has grown into a widely used cross-platform package that supports software-defined radio systems. Part of its success is derived from a flexible, process block abstraction which allows software developers to manipulate signals by appending a series of individual signal processing events. Written primarily using the Python programming language, GNU Radio applications declare the linkages between signal processing events (also called signal processing blocks). The signal processing blocks themselves and performance-critical algorithms are implemented in the C++ programming language. Typically they are imported at the very beginning of a Python script. See Appendix A for a short tutorial of the coding structure.

3.2.2 USRP

The GNU Radio project developed the USRP as a relatively low cost ($800) software radio under the direction of Eric Blossom and a team led by Matt Ettus [45], [46]. It too gained wide adoption through flexibility – offering a hardware platform that is easily reconfigured by adding or removing interchangeable daughterboards, each designed to operate within specific bands of the EM spectrum (DC to 5.9 GHz). For our research we used the USRP version 1 hardware as shown in Figure 3. Receiver and transmitter daughterboards are affixed to the USRP motherboard which houses four analog-to-digital converters (ADCs) and four digital-to-analog converters (DACs), and a field-programmable gate array (FPGA) for high-speed floating point signal processing. A USB 2.0 controller is the sole interface between the radio hardware and the radio software (which resides on a host computer). These components are identified in Figure 4 and listed in more detail in Table 1 and Table 2.

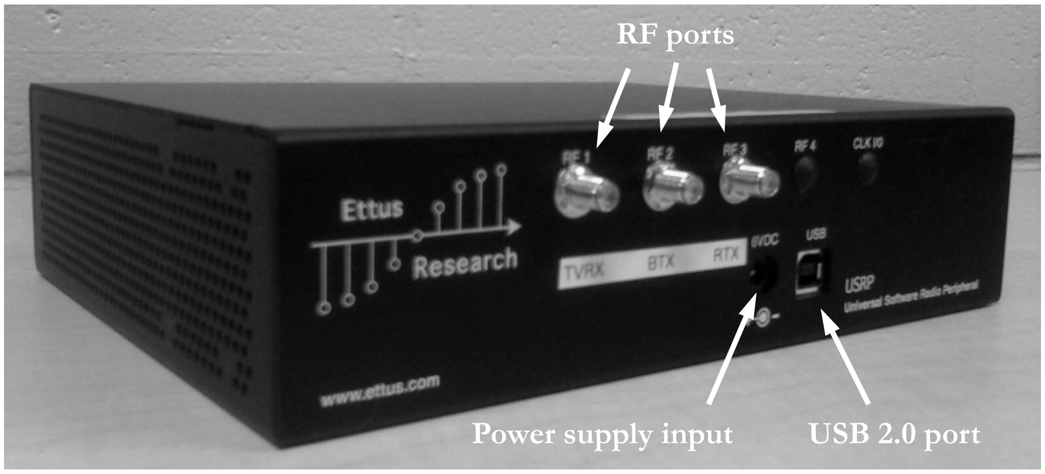


Figure 3: USRP version 1 hardware enclosure displaying external interfaces

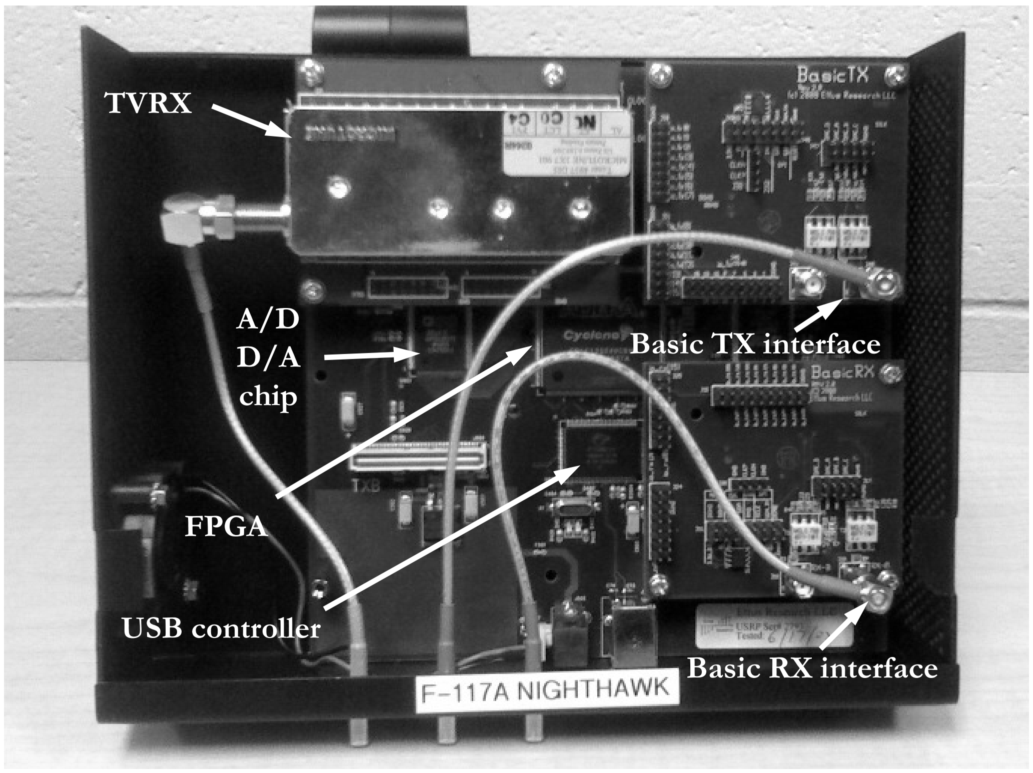


Figure 4: Top-down view of URSP version 1 main board and daughterboard components

Table 1: USRP Motherboard specifications [46]

USRP Motherboard	
Dimensions	2 1/8 " x 7 " x 8 1/4 " (with enclosure)
	1 1/2 " x 6 1/4 " x 7 " (without enclosure)
FPGA	EP1C12 Q240C8 Altera Cyclone
A/D	4 x AD9862 12-bit, 64 MS/s, Bandwidth: 32 MHz
D/A	4 x AD9862 12-bit, 128 MS/s, Bandwidth: 32 MHz
Interface	High-speed USB 2.0, 480 Mb/s
Power Requirements	6 Volts DC, 0 ~ 3.5 Amps

Table 2: USRP Daughterboard specifications (partial list) [46]

USRP Daughterboards (partial list)		
Name	*Operating Bana*	*Notes*
Basic RX	1 MHz - 250 MHz	Receive only
		No mixers, filters, or amplifiers present
Basic TX	1 MHz - 250 MHz	Transmit only
		No mixers, filters, or amplifiers present
TVRX	50 MHz - 860 MHz	Receive only
		Automatic Gain Control
		Based on standard TV tuner module

3.2.3 Decimation and FFT size

The decimation rate and the FFT size are two fundamental parameters that affect how signal data is represented before they are manipulated by a Python script. But before we explain these parameters, we must discuss the USRP's sampling process. All of the USRP components are tied to the FPGA as shown in Figure 5, and are driven by the motherboard's clock which operates at 64 million cycles per second (MHz). Analog signals received by a daughterboard are streamed to the ADC where they are digitized. Once digitized, the stream of signal bits is passed through the FPGA to the USB 2.0 controller. Finally, the signal bits are streamed via the USB 2.0 cable to the host computer where they are manipulated.

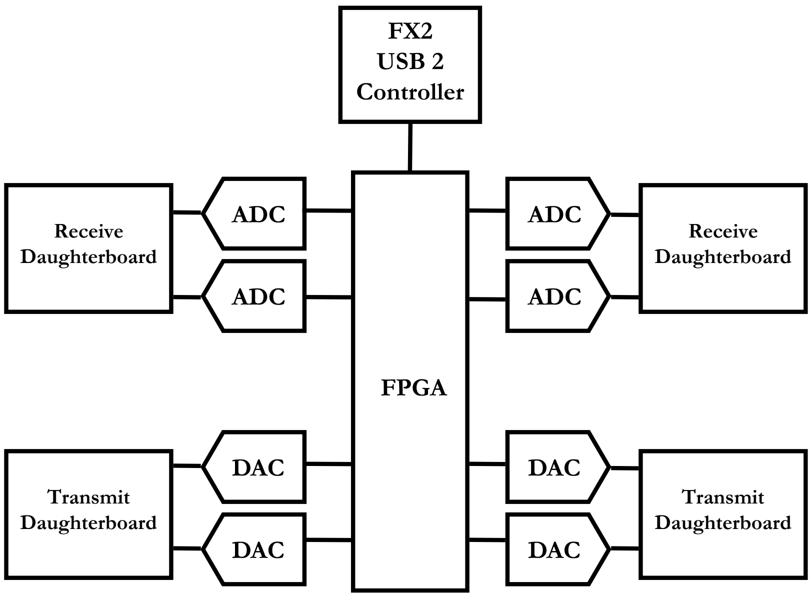


Figure 5: Block diagram of the USRP hardware interfaces [47]

The decimation rate *d* is a user-defined, positive integer which specifies the sampling rate that the FPGA applies to a received signal. This rate is a fraction of the ADC's 64 MHz sampling rate and is usually specified as a base-2 value (2, 4, 8, etc.). Thus, a decimation rate of 4 instructs the FPGA to sample a digitized received signal at a rate of:

$$\frac{64\times10^{6}\text{cycles/sec}}{4\text{ cycles/sample}}=16\times10^{6}\text{samples/sec} \qquad (1)$$

which is more commonly written as 16 Mega-samples/sec (MS/s). Alternately stated, given a decimation rate $d = 4$, the FPGA will take every fourth sample of a signal that was originally captured at 64 MS/s and discard the other samples. What results is a digitized version of the received analog signal that has been effectively sampled 16 million times per second.

The FFT size is a base-2, positive integer that affects a sampled signal in both the time and frequency domains. In the time domain, the FFT size specifies the *number of samples* to be taken from the input signal. This same value also defines the *number of frequency bins* a digital signal will be represented by when it is converted to the frequency domain. It is most commonly represented as *N* in the discrete Fourier transform (DFT) equation:

$$X(m) = \sum_{n=0}^{N-1} x(n) * e^{-(j2\pi m)/N} \qquad (2)$$

where

$X(m)$ =	*the mth DFT output*
$x(n)$ =	*the discrete time signal*
N =	*number of frequency bins, and the number of samples of the discrete time signal*

The ADC (which deals with received signals in the time domain) uses the FFT size to determine the *number of samples* to take of the input signal, whereas the decimation rate is used to specify the *rate* at which those samples are taken. In our data manipulation code (which primarily deals with received signals in the frequency domain), the FFT size determines the number of frequencies used to represent the received signal as it is transformed into the frequency domain.

For the purposes of our research, we needed to find a decimation rate and an FFT size that would represent the received signals in sufficient detail for manipulation. Our general goals were to sample incoming signals quickly enough to avoid aliasing, and to make the frequency bins sufficiently narrow. Together, these goals intended to ensure that the received signals were not misrepresented so that an automated algorithm could accurately identify occupied stations (as explained in Appendix B).

3.2.4 Determining the Decimation Rate and FFT Size

In many of the GNU Radio scripts included with the development package, the default decimation value is preset to 8. Using the default value as a starting point, we began a comparison of different decimation rates given an arbitrary, fixed FFT size of 512 points. To make our comparison, we tuned a USRP to a center frequency of 92 MHz and observed the resulting power spectral density (PSD) plots given decimation settings of *d* equal to 4, 8, and 16. Figure 6 depicts the time-averaged PSDs for each of the three settings. One of the first observations we made was that the widths (viewable bandwidths) of the plots vary. As the decimation rate was increased, the viewable bandwidth of the USRP decreased. For example, given a decimation of 4, the viewable bandwidth is 16 MHz ([84, 100] MHz), whereas for a decimation of 16, the viewable bandwidth is only 4 MHz ([90, 94] MHz). Just as the decimation rate affects the rate of sampling in the time domain, it also affects the sampled bandwidth in the frequency domain by taking a fraction of the maximum viewable bandwidth (as set by the USRP sampling rate). Thus a bandwidth of [-*Fs*, +*Fs*] is reduced to [-*Fs*/*d*, +*Fs*/*d*].

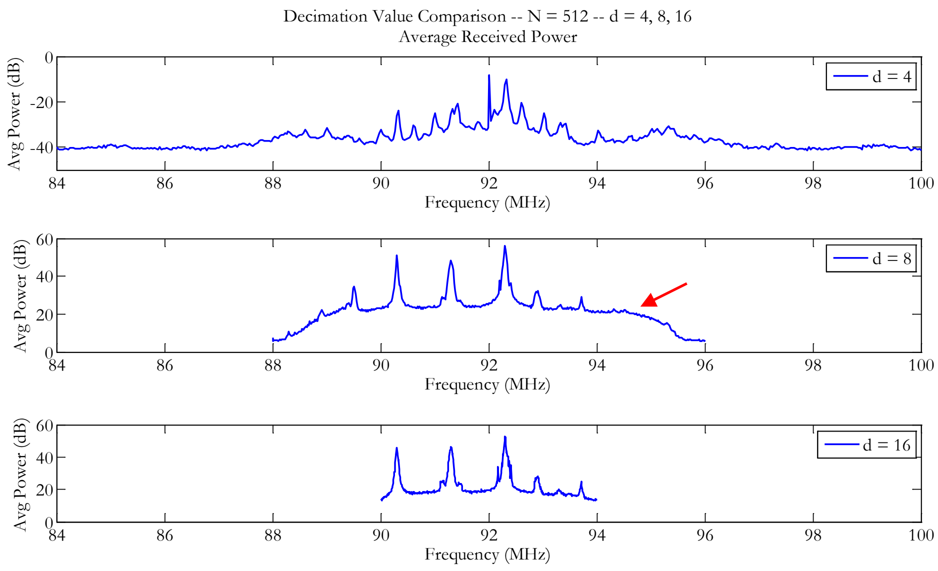


Figure 6: Decimation value Comparison for *d* = 4, 8, 16 (Note: The sharp peak occurring at 92 MHz for the *d* = 4 case may be the result of a nearby electronic device or an air conditioning unit. During our research we observed that these devices tend to emit energy around the upper portion of 91 MHz.)

Additionally, Figure 6 shows that the shapes of the plots vary in two important ways. First, the level of detail given by each plot increases as the decimation value increases, which can be partially attributed to the fact that regardless of the length of the x-axis, each case is represented by a fixed number of points. Although the upper plot gives a cursory view of how many stations may exist between a much larger band of frequencies, it does not capture some of the nuances that would help us precisely determine where stations begin and end. Second, we see that the shape of the noise floor is more prominent in the '$d = 8$' condition (as indicated by the arrow). By raising the middle 80% of the bandwidth (by 10 dB) our automated station identification algorithm (as described in Appendix B) would be adversely affected because it relies on a comparison between a station's supposed average power (across its bandwidth) and the total average power contained in the viewable bandwidth. Thus, the stations residing on the edges of the viewable band would be unfairly dwarfed and ignored. Given these observations, we decided to select 16 as our decimation value, and to continue to use it as we explored our choice for an appropriate FFT size.

Using a similar approach, we ran trials using FFT sizes above and below our starting point. Figure 7 depicts time-averaged PSD plots for cases where N equals 256, 512, and 1024. The most significant observations we made were influenced by the levels of detail given by each plot. As expected, the larger the FFT size, the greater the detail that can be displayed. For example, the first plot depicts a general outline of the occupied channels for N equal to 256. Although this plot is useful for confirming the existence of strong radio stations, it would be difficult to discern the full widths of particularly weak channels, as is the case at 91.3 MHz. By doubling the number of frequency bins used to represent the signal, the '$N = 512$' case shows improvement in the level of detail that describes where channels begin and end, as well as the upper and lower sideband widths (as seen at 92.3 MHz – an

HD radio station). The bottom plot of Figure 7 shows the '*N* = 1024' case, which does not appear to be a significant improvement over the previous. The additional data points provide more detail mostly to the noisy areas, but this offers us no additional value. As a result, we decided to use an FFT size of 512.

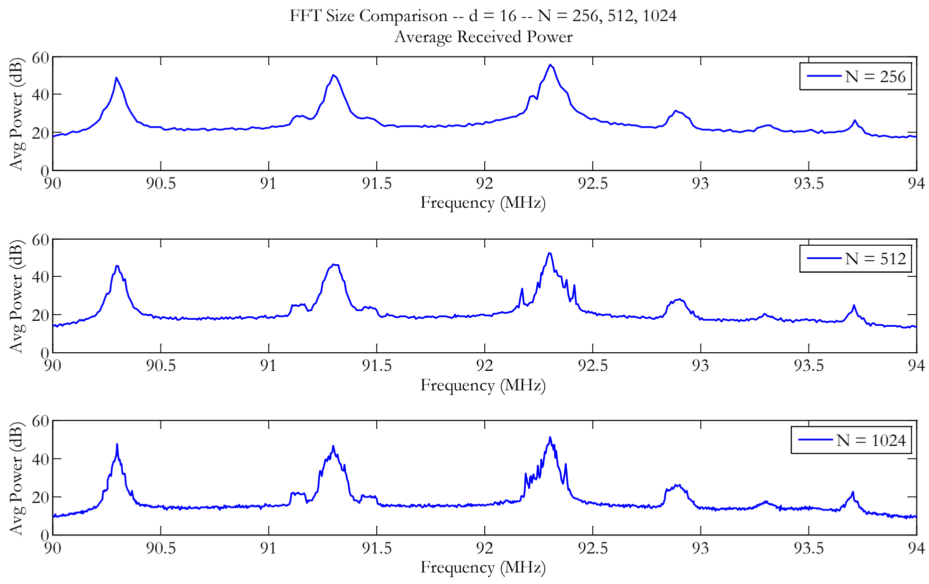


Figure 7: FFT Size Comparison for *N* = 256, 512, 1024

Based on what we observed in Figure 6 and Figure 7 we decided to use a decimation of 16 and an FFT size of 512. All subsequent tests and experiments operated under these two parameter settings.

3.3 Data Collection and Data Reduction Methodologies

Our post-processing approach organized the data collection and data reduction procedures as two distinct steps. First the USRP hardware was used to capture signal data using a GNU Radio Python script. Then, our MATLAB script was applied to convert, calculate, and extract RSSI information from the signal data. The algorithms found in the `usrp_capture_nsamples.py` and `Data_to_PSD.m` scripts are explained here.

Also, we describe an unexpected problem that we encountered during initial data collection attempts, and give the corrective solution we applied to eliminate its effects.

3.3.1 USRP Data Collection

The `usrp_capture_nsamples.py` script was packaged with the GNU Radio development software. The purpose of the script was to use the USRP hardware to capture complex signal data and then store them into a binary-encoded file according to the arguments it specified: decimation rate, FFT size, tuning frequency, and file name. Since the script performs only one iteration each time it is executed, we needed to find a way to run as many successive iterations as necessary to continuously capture RF signals. Borrowing heavily from a wrapping script created by Reginald Cooper [48], we implemented a process that imported the signal capture program as a function, so that it could be called repeatedly until interrupted by the user. The pseudo code of shows that during each iteration, a new file name was formatted to include the current system time (as Epoch time), and an iteration count value as in: `data_1231832819.02_991.bin`. Given our standard FFT size of 512 points, each data file was approximately:

$$512\,\frac{\text{samples}}{\text{iteration}} \times 4\,\frac{\text{bytes}}{\text{signal}} \times \left(1\ \text{signal}_{\text{in-phase}} + 1\ \text{signal}_{\text{quadrature}}\right) = 4096\ \text{bytes} \qquad (3)$$

or 4 kilobytes in length.

```
while iteration_index >= 1

    declare USRP parameters (decim, FFT_size, tune_freq)
    create filename as file_path + 'data_' + current Epoch time + '_' + iteration_index + '.bin'

    call usrp_capture_nsamples
    store returned values to file

    increment iteration_index
```

Figure 8: Pseudo code for data collection wrapper function

Figure 9 depicts signal data for one capture iteration as a pair of in-phase and quadrature phase signals that are 128 microseconds in duration. The signal duration can be verified using the following expression:

$$512 \text{ samples} \times \left(64\,\frac{\text{MS}}{\text{s}} \times \frac{1}{16}\right)^{-1} \times 2 = 128\ \mu\text{sec} \tag{4}$$

where the in-phase and quadrature phase sampling are treated as independent, interleaved events [49], hence the factor of two.

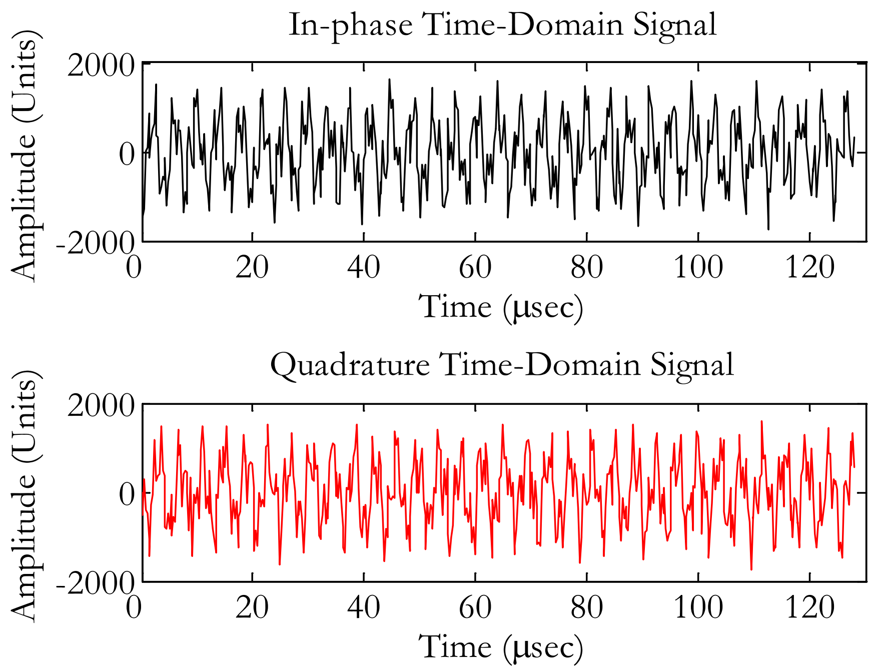


Figure 9: Typical in-phase and quadrature received FM signals

After the desired number of iterations are completed (usually determined by elapsed time), what remains is a folder of identically-sized signal data that is ready for reduction. It should be noted, however, that the data collection folders can grow very large (as in number of files) after only several minutes of data collection. For example, a 30-second collection period yields nearly 500 files. Data collections lasting several minutes could not feasibly be transferred to another workstation for data reduction because more files require more pre-write disk activities. Thus, we forced fewer files to be created by adding a brief sleep period (0.15 sec) at the end of the capture iteration. This helped to reduce the file creation rate to approximately 140 files per 30-second period, or about 4.7 files per second.

3.3.2 MATLAB Data Reduction

Our data reduction script was designed to take the signal data files collected by the USRP and reduce them to a time-varying list of received power values given a station of interest. First, the binary files were read and their complex signal data were extracted as two separate signals - the real (in-phase) and imaginary (quadrature phase) parts. As given in Figure 10, the reduction process continued with the creation of a whole signal (represented in rectangular form), which was transformed into the frequency domain by applying the Fast Fourier Transform function. Once the signal was converted to a frequency domain representation, a power spectral density (PSD) was calculated in order to determine the received power at each frequency. Line 4 in Figure 10 merely shifts the spectral representation from a [0, 2π] display into a more intuitive [-π, +π] display.

```
1   whole_signal = real_part + ( j * imaginary_part )
2   whole_signal_FFT = fft( whole_signal ) / length( whole_signal )
3   whole_signal_PSD = abs( whole_signal_FFT )$^2$
4   whole_signal_PSD = fftshift( whole_signal_PSD )
```

Figure 10: Pseudo code that converts a complex signal (written in rectangular form) and transforms it into a PSD using the magnitude squared of the Fourier transform

For each file, the conversion process was performed by using the expression for the DFT (as given in Section 3.2.3), and an expression for the power spectral density:

$$X_{PSD}(m) = |X(m)|^2 \tag{5}$$

where the magnitude of the DFT signal is taken and then squared. Pictorially, the transformation is demonstrated in Figure 11.

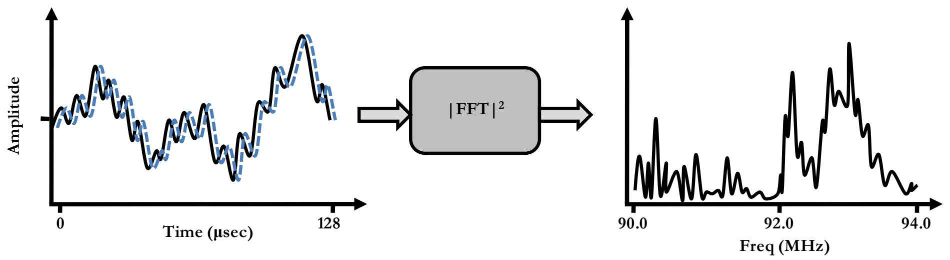


Figure 11: Complex time domain signals are converted to a power spectral density in the frequency domain using the magnitude-squared of the signal's Fourier Transform..

Using a PSD calculated from a single file made it easy to identify some of an FM station's features such as peak power. When combined with the PSDs of all subsequent files it became possible to see how the strength of a station (henceforth, channel) fluctuated over time. When viewed as an animation, the time-varying PSD revealed pervasive noise that also fluctuated over time. The additive effects of the noise made the PSDs appear jagged, thereby making it difficult to precisely identify the lower and upper frequencies of the radio channels. In order to minimize these effects we appended a process that took the average of the PSDs at regular intervals in time. By using a time-averaging process, as shown in Figure 12, the resulting PSD shape became smoother, and radio channels were more readily identified. (This also made the automated station detection algorithm easier to implement.)

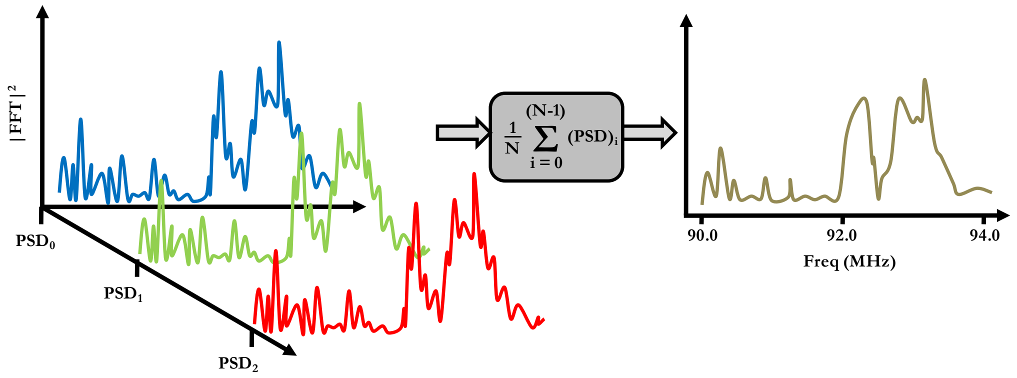


Figure 12: Several noisy PSDs are averaged over time to produce a less noisy PSD – a spectral summary – from which channels can be more readily identified.

We manually extracted received signal data by first using the time-averaged PSD to note the lower and upper frequencies that defined a channel of interest. Then, the frequency values were translated into start and end indices for a subset of columns within a time-averaged-PSD matrix. Using the column indices, we summed the received power values along each row of the matrix. Finally, what resulted was a time-varying vector of channel RSSI values.

3.3.3 Unexpected Problem: Ringing

Only after we began the data reduction process in MATLAB were we able to identify a problem in the data collection process. As depicted in Figure 13, there were some instances in which the complex signals we subject to an abnormal ringing effect within the first 23 samples (or 5.75 μsec) of data. In the frequency domain, these large, narrow impulses transformed into broad spectral densities that dwarfed all other PSDs. Given the transient nature of the ringing and their presence only at the beginning of some sampling iterations, we suspected that somewhere in the USRP a power surge occurs when it is commanded to start sampling (via the `usrp_capture_nsamples` script). Our solution was to extend the number of samples we would normally collect by specifying an intermediate FFT size of

512 + 23 = 535 points in the data collection code, and then we removed the first 23 samples for all signal data files as they were imported into MATLAB. The FFT size used in the data reduction code remained unchanged.

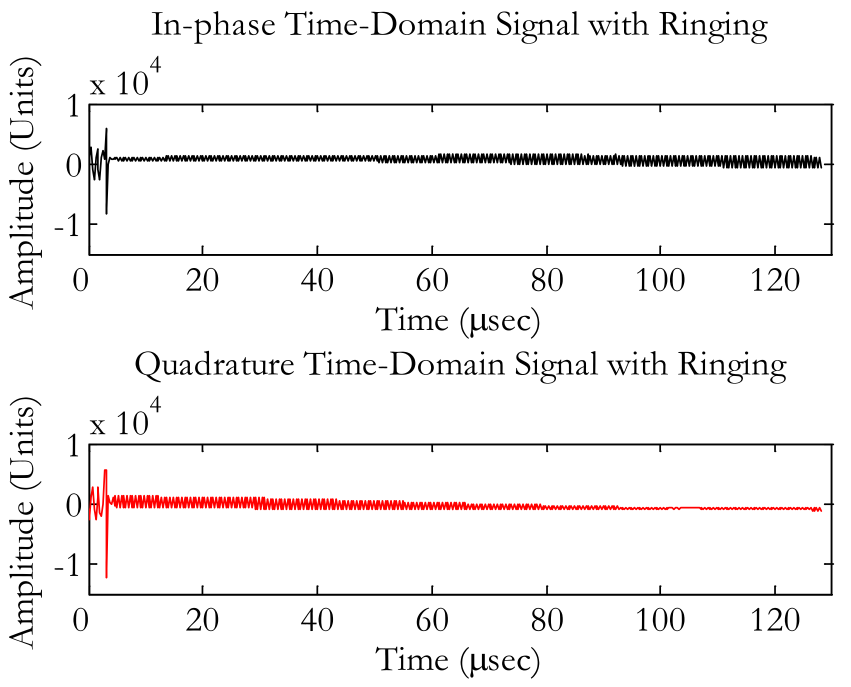


Figure 13: In-phase and quadrature received FM signals with ringing

3.4 Hardware Characterization

Aside from external sources of measurement error, particularly multipath fading and shadowing for RSSI-based applications, it is also important to recognize internal sources of error. Energy-based localization techniques greatly depend on how closely a sensor set responds given the same input conditions [19]. The RSSI technique we implemented is of no exception. Thus, we developed a procedure that helped to determine the uniformity of our sensing nodes. Using the same input signal applied to each sensor, we were able to make a comparison by overlaying their frequency responses onto a single plot.

The hardware set up for our characterization procedure is shown in Figure 14. Each sensor was given a -3 dBm (158.3 mV$_{0\text{-peak}}$) sine wave from a signal generator (Agilent E4438C) via a SubMiniature version A (SMA) cable – first to the Basic RX port and then to the TVRX port. The option to connect all sensors to the signal generator simultaneously using SMA splitters was deferred in favor of connecting each radio one at a time. This decision helped to ensure a more uniform received signal among all sensors. It also eliminated the need to characterize losses across each splitter – a time-consuming process.

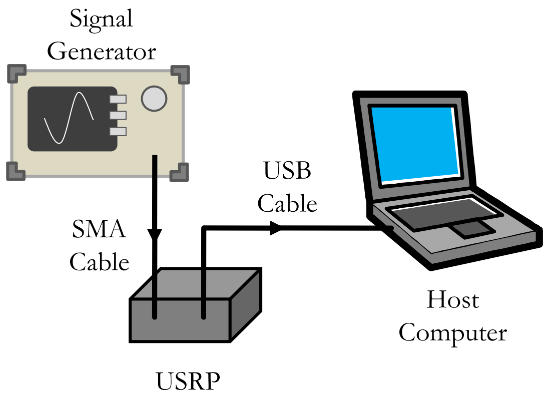


Figure 14: USRP hardware characterization setup

In anticipation of outdoor experiments, we centered a 4 MHz band of frequencies about 92 MHz to perform our characterization test because it was the least crowded by local radio stations. The -3 dBm signal was swept through nine evenly spaced frequencies within this band (90, 90.5, 91 MHz, etc.) and dwelled at each frequency for one minute before advancing to the next. Figure 15 depicts the power received by one of our software radios over time (increasing from right to left), and across the band of frequencies (increasing from front to back). Each sample was obtained using a decimation rate of 16 cycles per sample and an FFT size of 512 points.

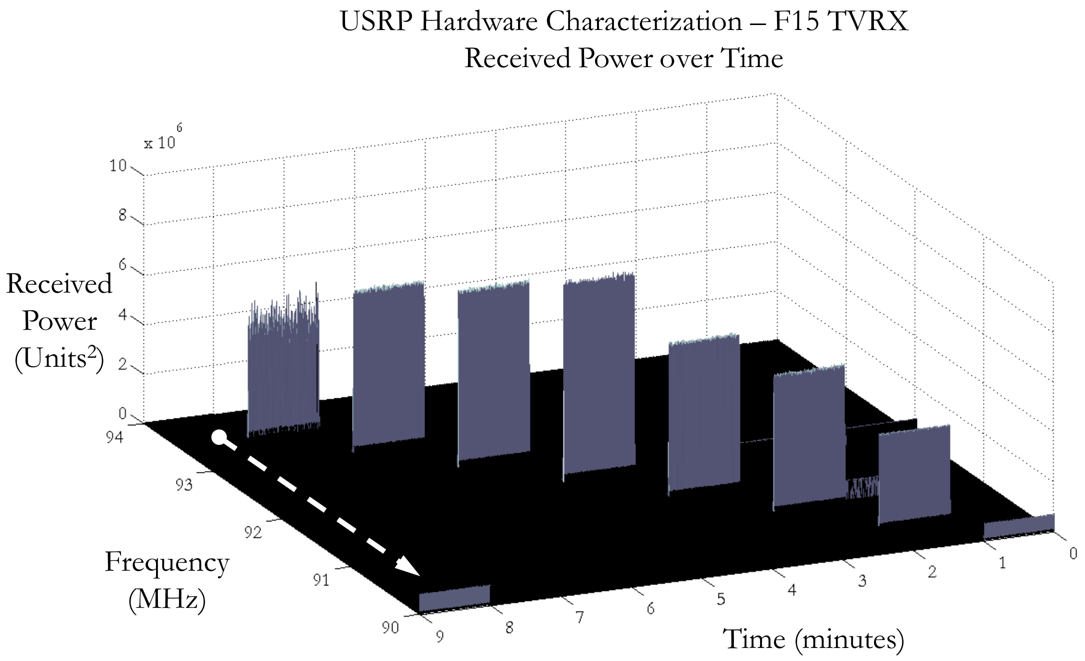


Figure 15: Time-varying plot of the characterization signal's PSD as received by the USRP labeled F15; arrow indicates signal energy shifted in frequency due to clipping effects

Once all of the radios finished sampling the characterization signal, the sample data for both their Basic RX and TVRX daughterboards were reduced by extracting only those areas where the swept signal was present – the gray pillars of data in Figure 15. On average, each pillar within the sweep band formed a channel approximately 23 kHz wide for the Basic RX daughterboards, and approximately 70 kHz wide for the TVRX daughterboards. The channels occurring at 90 MHz and 94 MHz were not relied on because they were subject to clipping effects. Additionally, the signal energy intended for 94 MHz was displaced to the other side of the viewable band. This is true for all sample data sets, and is highlighted in Figure 15 with an arrow. Subsequent activities were designed to avoid the bounding frequencies.

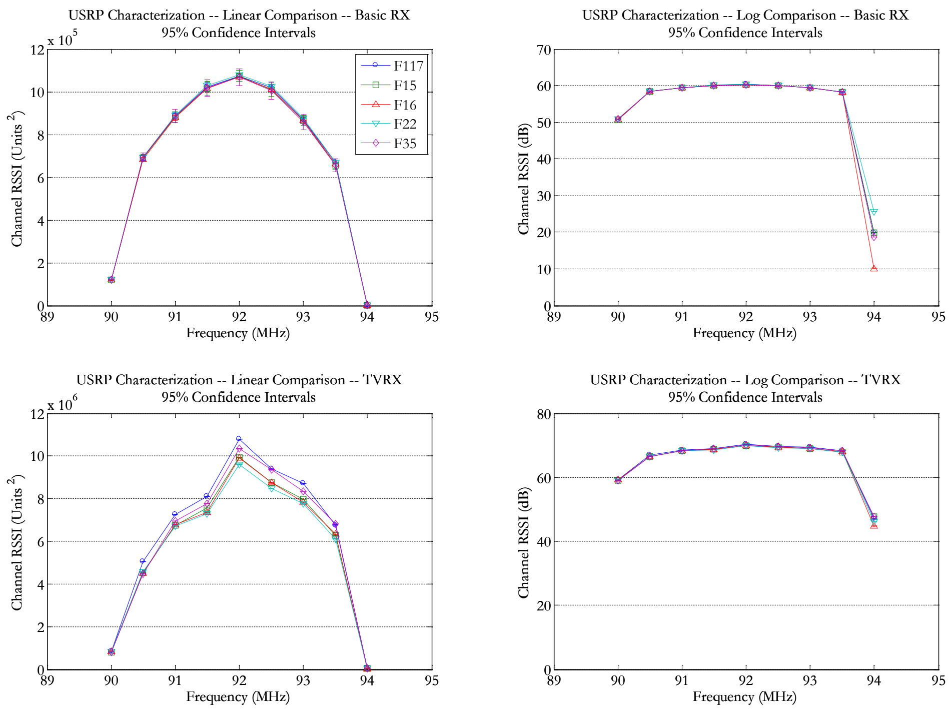


Figure 16: USRP Hardware Characterization Linear and Log-Scale Plots: Basic RX daughterboard (top row), TVRX daughterboard (bottom row).

After computing the average received signal power within all channels (as in Section 3.3), we were able to compare the channel RSSI values for both types of daughterboards by creating the plots of Figure 16. The linear scale plots of Figure 16 show that the channel RSSI values for the TVRX daughterboards are, on average, an order of magnitude greater than those of the Basic RX daughterboards. In the log-scale plots, this order-of-magnitude difference is represented as a 10 dB gain. The TVRX daughterboard gains are due to their built-in RF front end circuitry which amplifies received signals as they are translated to an intermediate frequency [50]. (The Basic RX daughterboard does not have an RF front end -- see Table 2.) We believe that because of manufacturing tolerances for the analog front end components, gains are not applied identically between the TVRX daughterboards, hence the slight variation in the TVRX linear-scale plot. Overall, Figure 16 reveals that the respective daughterboards respond similarly (within 0.46 dB) when given the same input signal.

Despite the general variation of its mean values, the TVRX daughterboard was selected as the primary interface for conducting our experiments. Since gains are applied by an RF front end, sensor arrays based on the TVRX daughterboard could be made to encompass a larger search area, thereby affording some additional topology design flexibility.

3.5 RSSI Localization Implementation

In [9] Martin and Thomas derived a new sensor localization algorithm that applies a Maximum Likelihood (ML) approach to estimate a transmitter's position, orientation, beam width, and transmit power using RSS measurements. A large portion of their paper's focus was centered on transmitter directionality, as previous research generally ignored non-uniform antenna gain patterns. To demonstrate their algorithm, they created a MATLAB simulation which modeled sensing and transmitting nodes that operated within a log-normal fading environment, and then applied their ML approach to various distributions of wireless nodes. Although we were unable to devise a suitable directional antenna that functioned within the FM band, we were fortunate to be able to borrow the portion of their code (`findomni2.m`) that implemented their localization algorithm against omnidirectional nodes (which they used for performance comparisons). Here we will discuss how their localization algorithm works, and address two considerations we made prior to designing our experiment.

3.5.1 RSSI Localization Algorithm

In general, the localization algorithm for omnidirectional nodes uses the same approach as for directional nodes. Beforehand, all nodes are arranged within a rectangular coordinate plane. The sensing nodes (whose positions are known) observe the received power from a transmitter located at some unknown point in the plane. Given a similar scenario, the `findomni2` function takes the following arguments: each sensing node's x and y coordinates, the RSSI value observed by each sensing node, and two vectors that define the boundaries and number of points within a rectangular search space (one vector for each dimension). As an example, Figure 17 depicts a search space that has been defined around a simple topology of nodes.

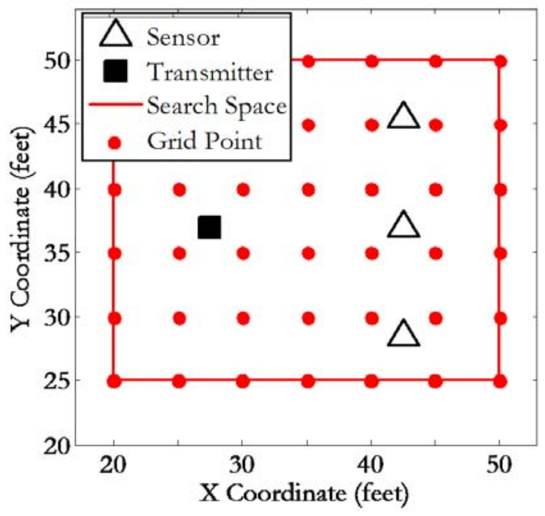


Figure 17: Example node localization search space

Using the observed RSSI values and the locations of the nodes, every point in the grid is evaluated to identify the likelihood that a transmitter resides there. Two types of calculations are performed at each grid point, one at a time -- preliminary calculations and likelihood calculations. The preliminary calculations identify the mean distances (and mean squared-distances) between all sensing nodes and the current search point, the variance of the sensors' distances to the current search point, the average RSSI value received by the sensor network, and the mean power-distance product - an average of the power received by each sensor, scaled by their separation distance to the current search point.

The likelihood calculations are performed in two phases. First, estimates of the best possible transmitter characteristics are calculated, which assumes that a transmitter exists at the current search point. Then, an error is calculated between the supposed transmitter characteristics (what was observed) and the best possible transmitter characteristics (what would have been observed). These calculations (as derived in [9]) are expressed as:

$$P0_{best} = \frac{\left(\mathrm{mean}(\boldsymbol{p}) \times \mathrm{mean}(\boldsymbol{d}^2) - \mathrm{mean}(\boldsymbol{p} \times \boldsymbol{d})\right)}{\mathrm{var}(\boldsymbol{d})} \tag{6}$$

$$np_{best} = \max\left(\frac{\mathrm{mean}(\boldsymbol{p}) \times \mathrm{mean}(\boldsymbol{d}) - \mathrm{mean}(\boldsymbol{p} \times \boldsymbol{d})}{\mathrm{var}(\boldsymbol{d})}, 0\right) \tag{7}$$

$$fiterror = \mathrm{norm}\left(\boldsymbol{p} - P0_{best} - (np_{best} \times \boldsymbol{d})\right) \tag{8}$$

where

$P0_{best}$ =	*the transmitter power that would have been observed*
np_{best} =	*the calculated path loss exponent*
fiterror =	*the normalized difference between the power observed and the power that would have been observed*
$\boldsymbol{p}$ =	*the power received by the sensing nodes (arranged as a vector)*
$\boldsymbol{d}$ =	*the distances between each node and the current search point (arranged as a vector)*

The computed error is a direct representation of the likelihood that a transmitter (with similar observed properties) exists at the current search point. After all search points have been evaluated, the computed error values form a matrix whose entries coincide with the search grid. Therefore, the search point that bears the lowest error value represents the most likely position of the transmitter. Figure 18 illustrates a sample fit error matrix taken as a surface and viewed from the side.

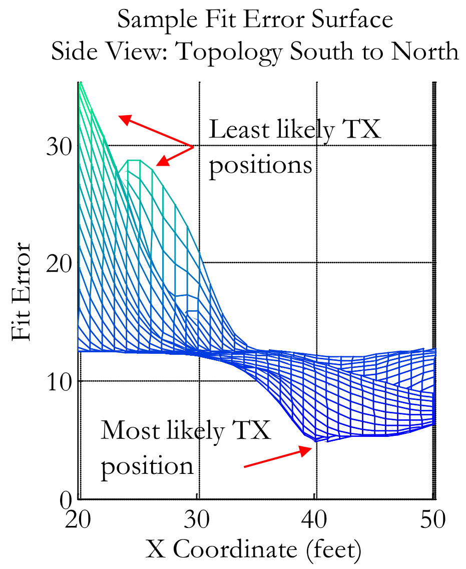


Figure 18: Example fit error surface viewed from the side

3.5.2 Usage Considerations

Before integrating the localization algorithm into our experiments, we made note of some usage considerations. Our first concern regarded how fine the search grid would be "drawn." Having fewer points meant having fewer cumulative calculations. However a fairly coarse grid would yield poor *position estimate* resolution. On the other hand, a very fine grid would greatly increase the *search space* resolution, but may do so needlessly since RSSI measurements can fluctuate greatly. Thus, we resolved to define our search grid points to be evenly separated by 1 foot in both directions.

Our second concern regarded sensor placement within the search space. The grid in Figure 17 does not include points where either the sensors or, more importantly, the transmitter lie. In fact, a search grid of this type would distort position estimates as none of the possible positions are correct solutions. Instead we resolved to design our experiment such that the transmitters would be placed on top of a grid point. In this way, we could accurately determine whether a position estimate was correct, and if not, calculate a valid position error. Although the sensor positions need not coincide with the search grid, we designed our sensor topologies to have any given sensor placed no farther than half a unit (or half a foot) away from a search point, in both the x and y directions.

3.6 Collaboration Experiments

Our research intended to use real-world experimentation to demonstrate collaborative localization, and thereby validate (or challenge) simulation results based on the quality of their position estimates. The nature of our experiment design was influenced by the tools we had available, particularly the number of software-defined radios we could use. In this section, we explain how we applied a time shift concept to amplify our post-processing approach, which turned our seemingly small number of nodes into a flexible network of 21 collaborative topologies. However, we also share the measures we put in place to help ensure time shift validity. Then, we describe the conditions of our experiment during the execution phase. Finally, we discuss how Martin and Thomas' simulation was configured in order to repeat our real-world experiments under a simulated RF environment.

3.6.1 Overview

We had six USRP software-defined radios available to us. They were divided into two roles; five of the radios were declared sensing nodes, and one radio was declared a transmitting node. The sensing nodes ran the `usrp_capture_nsamples.py` script to collect signal data. And the transmitting node broadcasted audio signals using an existing FM transmission program (`fm_tx4.py`) that came preloaded with the GNU Radio development package. To differentiate the USRPs, the sensing nodes were named after U.S. military F-series aircraft - "F15," "F16," "F22," "F35," and "F117" - while the transmitting node was designated "TX." When referred to in our topology legend in Appendix C, the sensing nodes are identified according to a number, from one through five, respectively. For example, "F15" appears as "1," "F16" appears as "2," and so forth.

3.6.2 Time Shift Concept

The number of SDRs on hand was a strong limiting factor that affected many facets our design. Our sensor set would determine the spatial diversity, reliability, and performance of our topologies. Having too few sensing nodes would severely limit the number of shapes and sizes of our sensing topologies. Also, sparse topologies would be more likely to suffer in the event of a poorly performing node. Thus, we needed to find a sensible way to expand our design options so that we could increase the likelihood of achieving sensible results.

Unlike the real-time approach, which interleaves the data collection and reduction processes with every iteration, the post-processing approach separates these events into two distinct phases. This distinction provided an opportunity well-suited for experimental analysis. Since data collection and data reduction did not occur concurrently, we were able to conduct multiple small-topology experiments at different points in time. Then, we combined the data from the experiments as if they occurred concurrently. Finally, we applied our data reduction process to the accumulated data. For example, sensing nodes were arranged as shown in Topology A in Figure 19 and collected signal data from a transmitter located at a nearby position, unknown to them. Then, the sensing nodes were rearranged to observe the same transmitter from different locations (as depicted in Figure 19, Topology B).

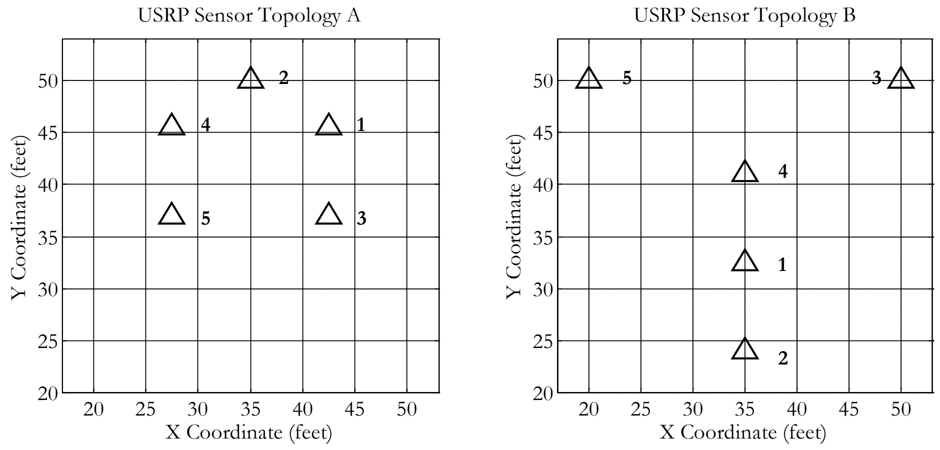


Figure 19: Sensor configurations used for time shift

By combining the signal data from our five sensing nodes, that sensed the same transmitter (each from two independent locations), we essentially emulated a ten node topology that acted upon two transmitters (separately). Figure 20 depicts our combined sensor topology, and overlays the locations where we placed our transmitter node. (See Figure C1 for an enlarged topology legend that shows the sensor identities.)

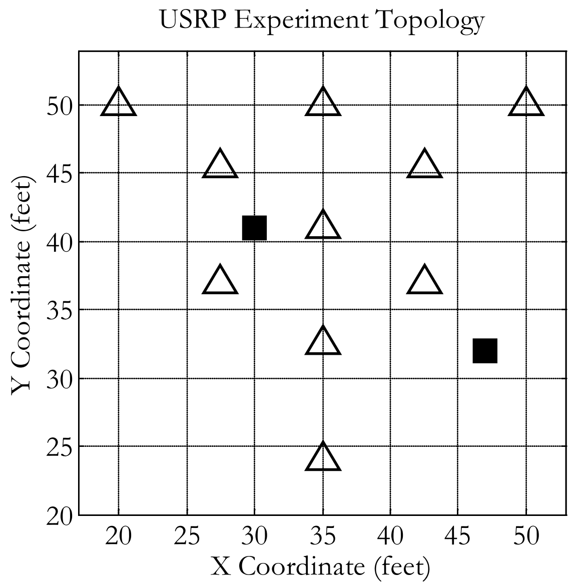


Figure 20: Complete sensor topology as a result of applying time shift

The flexibility of the post-processing approach allowed us to capitalize further on the time shift concept. By excluding the data collected by some sensors, we could select a number of sub-topologies from our emulated set of 10 nodes. For example, Figure 21 shows how a Rectangle topology was formed when we excluded signal data from six nodes. Repeating this process, we were able to identify 21 sub-topologies that varied by the number of nodes, shape, and size (as in perimeter). The sub-topologies took the following forms: Triangle, Rectangle, Hexagon, Line, and one topology that included all of the sensors. With the exception of the Line and All-Sensors topologies, every other topology type was varied by excluding, and then including, the node located in its center. For example, the Rectangle topology in Figure 21 was taken as shown, and again with the sensor located at coordinate (35, 41). All 21 sub-topologies are depicted in Appendix C with overlays of the transmitter locations.

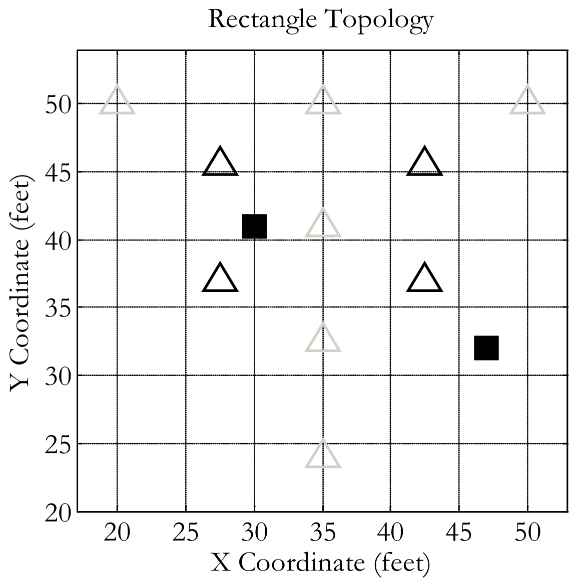


Figure 21: The nodes of a rectangular sub-topology are selected while the remaining sensors are excluded (subdued)

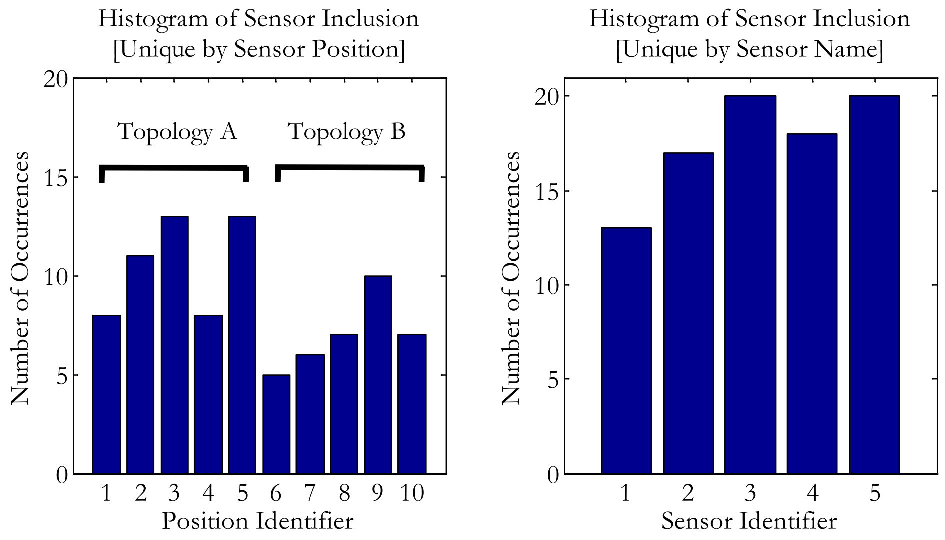


Figure 22: Histograms of sensor inclusion where F15 = [1, 6], F16 = [2, 7], F22 = [3, 8], F35 = [4, 9], and F117 = [5, 10]

The histograms of Figure 22 show how often each *sensor position*, and how often each *sensor* was used to form our sub-topologies. Along the x-axis of the leftmost histogram, sensor positions are listed using the following convention. As before "F15's" position is designated by "1" and "F117's" position is designated by "5." Positions 6 through 10 represent the time shifted positions of the same five sensors - position 6 is "F15's" second position, position 7 is "F16's" second position, and so on. In general, the first sensor positions (Topology A in Figure 19) were used more often than the second sensor positions (Topology B in Figure 19). The rightmost histogram in Figure 22 shows the frequency of sensor inclusion. It indicates that "F22" and "F117" were tied as the two most-frequently-included sensors. Both histograms helped us identify the critical dependencies of our topology choices.

3.6.3 Design Measures for Time Shift Validity

Although time shifting added a great deal of flexibility to our post-processing approach, its benefit would be moot unless our experiment design included an accurate data-alignment process. When we first introduced our data collection process (in Section 3.3.1), we explained that each signal data file was time stamped with the current system time. As listed in Figure 8, this timestamp was associated with the system time immediately before the signal capture function call. In this way, each signal capture could be identified according to when it was collected. However, in order to align the separate sets of data, they needed to be time stamped with respect to a common time reference.

As part of our first design measure, we established a common time reference by creating a wired, local area network (LAN). The host computers for all sensing nodes were joined to the same subnet as the transmitter's host computer, which acted as the network time protocol (NTP) server. Under this architecture, the sensors' host computers would synchronize their system clocks by polling the time server upon system boot-up and periodically thereafter. Using an independent digital clock, we performed a simple test to confirm that all system time clocks were accurate to at least the nearest second.

Using an NTP server allowed us to mark every signal data file with a timestamp based on a common-reference clock. Thus, we were able to keep our experimental procedure simple. Each node was commanded to start signal capture, one at a time. Once all of the nodes were capturing, we activated the transmitting node to broadcast an audio file for 61 seconds. After all trials and signal collections were complete, we merely needed to identify a common signal event (for example, the first peak of the received signal), and note its associated timestamp. So long as the transmitter emitted the same 61 seconds of audio during each trial, this alignment process was valid regardless of where or when a given sensor node made a signal collection. Without ignoring the fact that an emitted signal would be received at different times by antennas at two different locations, we performed a calculation to further justify our decision to align data in this fashion. Under free space propagation of RF signals across separation distances no larger than 33 feet, the time differences of arrival among the sensing nodes are on the order of $(9.8 \times 10^{8}$ ft/sec$)^{-1} \times 33$ ft $\approx 33.7 \times 10^{-9}$ seconds, or tens of nanoseconds. This figure of merit is much smaller than our timekeeping precision, and is therefore negligible.

Our final design measure in support of time shift regarded how we implemented timestamps during the data reduction process. As stated before, the timestamps were given as Epoch time, which denotes the number of seconds since midnight of January 1, 1970 [51]. This number was given with decimal seconds, as in `1231832819.020`. To help simplify our alignment and time-averaging processes, we ignored partial-second increments by truncating the timestamps to whole numbers of seconds. Doing this changed the "resolution" of contiguous data captures to be relative to the nearest second. For example, signal captures that occurred at `1231832819.020` and then at `1231832819.35` may have just as well occurred in the reverse order. Thus, when we applied our time-averaging process to datasets collected by any given node, we specified an averaging interval of one second.

3.6.4 Experiment Execution

Our collaboration experiment was conducted outdoors in an uncovered parking lot. Our equipment setup was no less than 50 feet away from vehicles or other large RF reflective objects. Weather conditions were more accommodating than usual for a typical Ohio winter: a high temperature in the low 40s, clear skies, 70% humidity, and winds from the South-southwest averaging 10 mph [52]. Our first order of business was to mark the sensor and transmitter positions on the parking lot surface since we were going to reposition the nodes during the four phases of our experiment (as given in Figure 23). Each node was elevated 12 inches above the ground using plastic storage containers to minimize RF ground effects. The power and networking cables were routed along the ground to a cart that carried our portable power unit and network router. This cart was located at what would have appeared as coordinate (10, 10) in our topology diagrams - far enough to have little to no influence on the experiment devices.

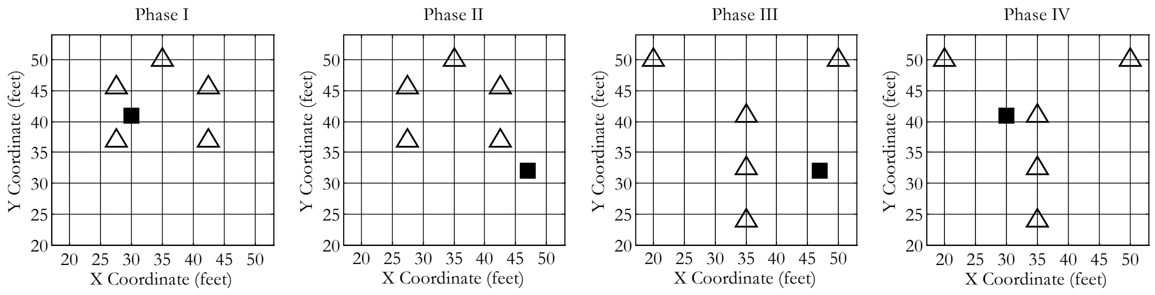


Figure 23: Four phases of the outdoor experiment

Before we began Phase I of our experiment, we followed a list of pre-test checks: (1) all network cable connections are secured and active, (2) all host computer clocks are synchronized to the nearest second, (3) all antennas are upright, fully-extended and positioned squarely above their mark, and (4) all sensor nodes detect the test audio broadcast from the transmitter. This set of pre-test checks helped to ensure that the sensor layout matched what we designed and that the equipment was functioning as expected. Between phases, this list was reduced to check-items (2) and (3).

At the beginning of every phase, each sensing node was remotely started using a remote desktop application. Once all of the nodes had begun collecting signal data, the transmitter node was remotely activated, and a digital timer was started. After 61 seconds had elapsed, the transmitter was turned off and then the sensing nodes were commanded to stop signal collection. Upon completion of the final phase, all signal data were retrieved from the host computers, and organized in preparation for data reduction (as discussed in Section 3.3.2).

3.6.5 Topology Simulation

The simulation software, which we borrowed from [9], utilized a two-step approach. First, assuming a log-normal fading environment, received power values were generated for a randomly-placed network of sensors that were observing a transmitter. The transmitter's position was unknown to the sensor nodes. Then, the localization routine was applied to estimate various characteristics of the unknown transmitter based on the received power values and locations of the observing nodes. We modified the software so that we could simulate the performance of our sub-topology configurations.

The first change we made was to create a wrapper function that iteratively invoked the simulation software much like our data reduction process invoked the localization algorithm, `findomni2.m`. Then, we adjusted the code that generated the simulated sensor positions to, instead, read the sensor positions that were based on our topological design. Then, the fading model variance parameter was adjusted to what we felt was comparable to the outdoor environment at the time of our experiment. This term (given in dB) governed how heavily the fading model was applied. A variance of 4 dB corresponded with an uncluttered environment (such as a desert), and a variance of 12 dB was associated with considerable levels of shadowing and multipath (such as an urban environment). Originally we selected a variance of 4 dB, but after reviewing the simulated data, we reduced this value to 3 dB because it yielded position error figures that were more on par with our experimental data. The final change we made was to disable the antenna shaping code and thus make the transmitter an omnidirectional emitter.

3.7 Summary

In this chapter we described the tools we used to conduct our experiments. We also discussed how these tools were incorporated into our data collection, data reduction and hardware characterization procedures. A description of our localization algorithm was provided, as well as the details regarding how our collaboration experiments were conducted.

IV. Data Analysis

4.1 Introduction

Our data analysis process takes position estimation results from our node localization experiment -- conducted using actual hardware in an uncluttered outdoor environment -- and compares them with results for the same experiment which was reenacted in a simulated environment. The purpose of this chapter is to review all of the facets that helped us form a comparison between the topologies we tested and between the different environments they were tested in. In Section 4.2 we list, define, and justify the performance metrics we used in our comparison. In Section 4.3 we share our general hypotheses about topology performance based on the size of a given topology and its node distribution with respect to an emitter. Section 4.4 provides our observations and performance comparisons between the topologies as they operated in the outdoor environment (henceforth, real world), first according to their "helpfulness" (in terms of absolute error), and then with regard to their "effectiveness"(in terms of normalized error). In Section 4.5, we compare the real-world results with the simulated results to identify similar and dissimilar trends between them.

4.2 Performance Metrics

Our performance metrics were derived from the position estimates that were generated by our localizing sub-topologies. Using the known transmitter locations, we calculated position errors by evaluating

$$\text{error}_{\text{radial}}=\sqrt{(\Delta x^2 + \Delta y^2)} \tag{9}$$

which is written more explicitly as,

$$\text{error}_{\text{radial}} = \sqrt{(x_{\text{guess}} - x_{\text{true}})^2 + (y_{\text{guess}} - y_{\text{true}})^2} \qquad (10)$$

where *xguess* and *yguess* are the estimated transmitter coordinates; and *xtrue* and *ytrue* are the transmitter's actual coordinates. By performing this calculation for all position estimates, we created two time-varying error vectors for each sub-topology – one error vector with respect to each transmitter location. For example, Figure 24 compares the radial position errors for a triangle and a hexagon topology as they vary over time.

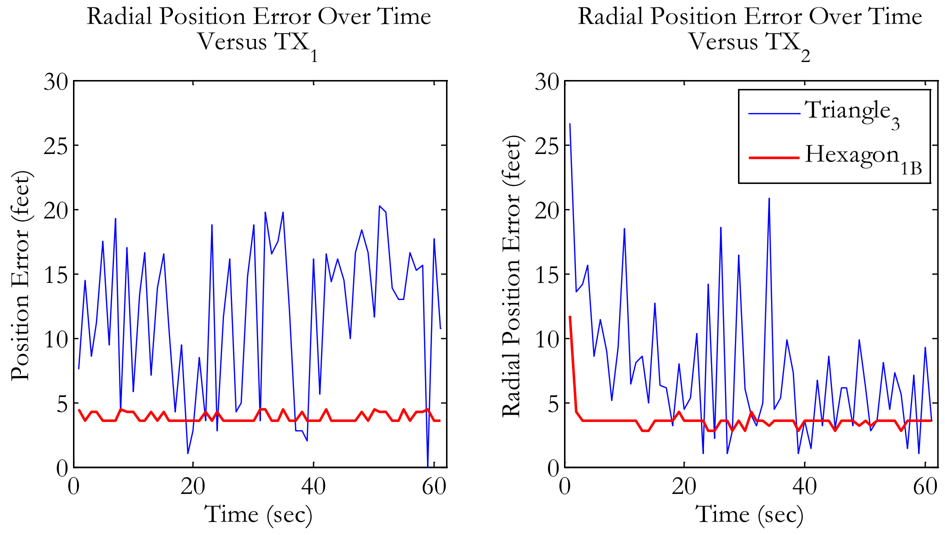


Figure 24: Time-varying radial error comparison between Triangle_3 and Hexagon_1B

Since the radial position errors varied over time (as seen in Figure 24), we calculated the mean position errors for our topologies to *quantify* their performance. Then, we sought to *qualify* each topology's performance according to how precise (or consistent) its position estimates were, regardless of whether they were correct or not. This was done by calculating the variance of the position errors. By combining these two metrics – mean position error and position error variance – we were able to make absolute comparisons between our topologies according to how accurate and how consistent they were with their localization attempts.

4.3 General Observation Guidelines

As can be observed in Appendix C, our topologies varied by size, shape, and orientation with respect to the transmitter locations. Some topologies surrounded a TX node, and some did not. Some topologies were very sparse, while others were relatively dense. These and other cursory observations were considered when we formed our short-list of guidelines which we used to gauge our data's correctness. First, our general belief was that as more nodes surrounded an emitter, the average position error should be relatively lower because there would be more independent observations of the same phenomena as opposed to a sparse network. Second, for those topologies that had the option available, adding the center node should reduce the topology's position error variance. A node added to a topology's center would increase the topology's spatial diversity without disrupting its symmetry as opposed to adding the node somewhere beyond the topology's perimeter. And third, we believed that topologies which had a symmetrical distribution of nodes about (or near) an emitter would have lower average position error values than those topologies that were not. Symmetry would offer positive redundancy which would help reduce ambiguity, and thereby produce more consistent estimates.

4.4 Real World Results

Using the performance metrics we defined (in Section 4.2) we constructed the plots of Figure 25 and Figure 26 to compare topology performance against both transmitter locations. This type of plot shows the mean *absolute* error for each topology (shown as bars), as well as the variance associated with their position error distributions (shown as square-ended stems). The mean absolute error plots can be used to look at topology performance on an individual basis or between topologies of the same size (as in number of nodes).

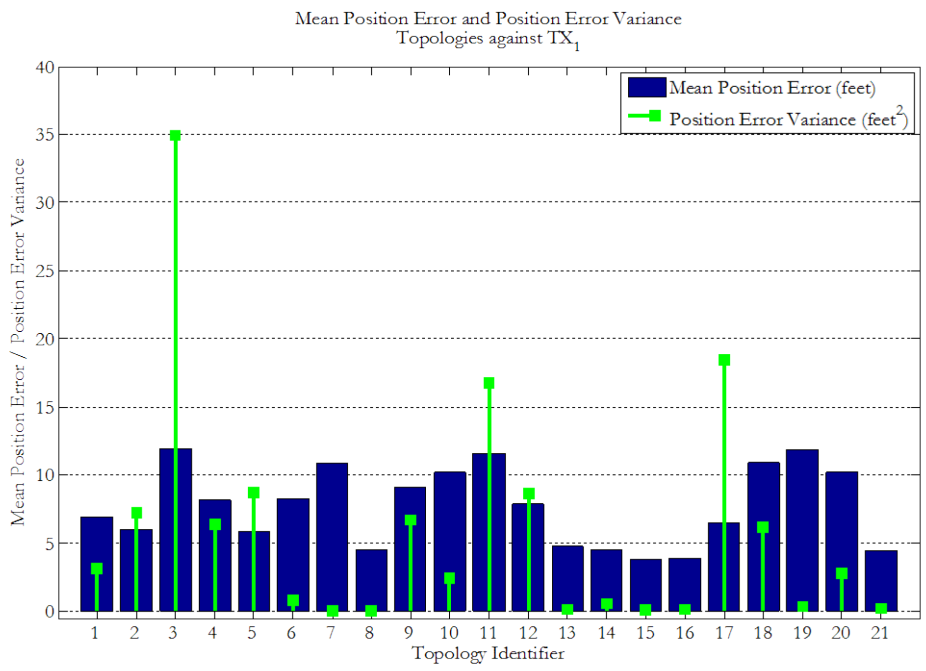


Figure 25: Mean absolute position error for all real world topologies against TX_1

For example, topologies 1 through 5 (the 3-node Triangles) can be directly compared with topologies 11 and 12 (the 3-node Small Triangles), but not with topologies 6 through 10 (the 4-node Triangles). Using mean absolute error is but one way to make comparisons, such as determining which topology had the largest position error or which topologies were statistically similar. These kinds of comparison are good; however, a normalized comparison is more meaningful because it takes into account topology size. In this way, smaller topologies would be praised for exemplary performance, and larger topologies would be penalized for not performing better than their peers. Thus, we begin our comparisons using measures of mean absolute error to determine which topologies were the "most helpful", and then continue with an analysis of those topologies that were "most effective."

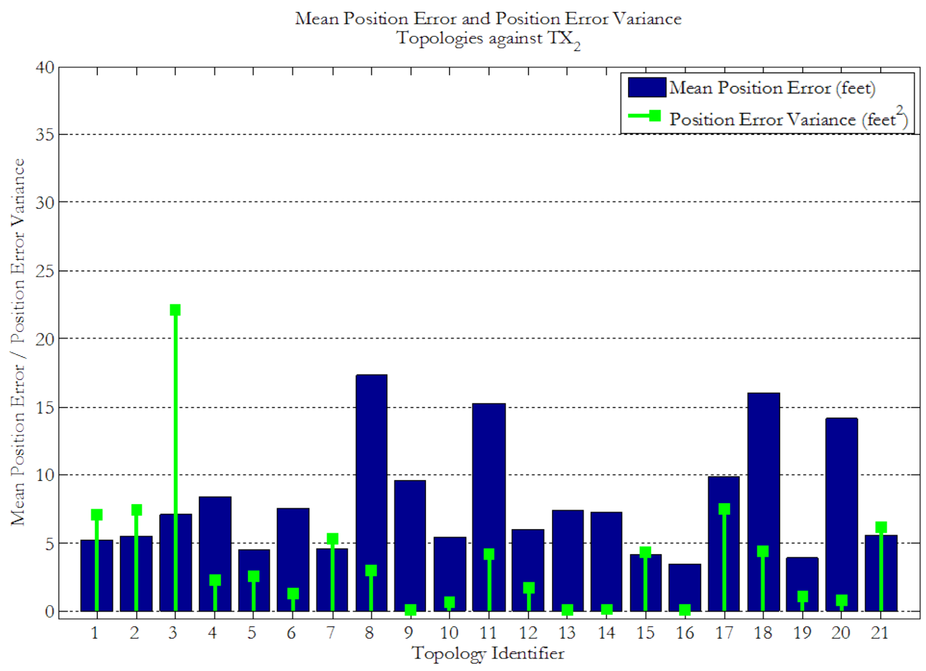


Figure 26: Mean absolute position error for all real world topologies against TX_2

As we delved deeper into the data to understand why particular results were the way they were, we spotted a problem. Looking again at the right plot of Figure 24, we saw that the position errors for both topologies started with a large decrease before carrying on as they should. This initial drop only occurred for some sets of data and for other sets there was a large change at the end of the time series. We were hesitant to believe that these events were caused by misalignment of the signal data. Instead, we believed it was caused either by a single, or a pair of "misbehaving nodes." (We discuss our efforts to identify these nodes later.) To avoid skewing our data as a result of a few outliers, we removed the first and last position estimate for all data sets before we made our comparisons.

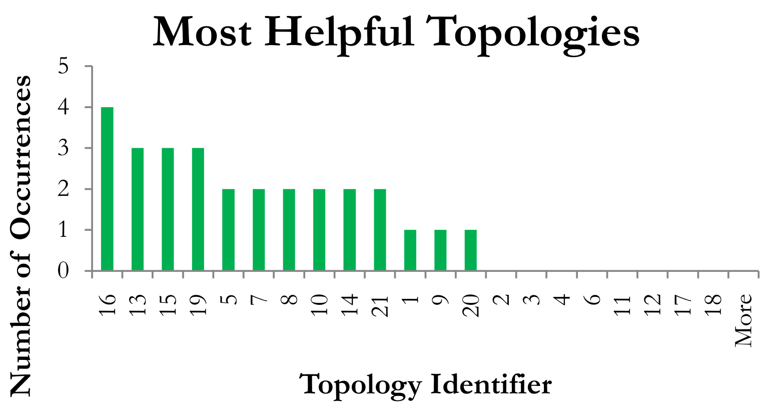
Comparing the plots of Figure 25 and Figure 26 we see that Triangle_3 (ID: 3) and Triangle_3B (ID: 8) have the largest mean absolute errors against TX_1 and TX_2 – 11.97 feet and 17.32 feet, respectively. The topology layouts (in Appendix C) show that Triangle_3 did not surround TX_1 and Triangle_3B's nodes were far removed from TX_2 (approximately 20-32 feet). On the other hand, Hexagon_1 (ID: 15) and Hexagon_1B (ID: 16) produced the smallest mean errors – 3.79 feet and 3.5 feet, respectively. Both hexagon variants were spatially diverse and were symmetric. For example, symmetry with respect to TX_1 can be seen if a horizontal line is drawn across the middle of Hexagon_1, and a line drawn diagonally across Hexagon_1B (from top left to bottom right) reveals its symmetry with TX_2.

Continuing with Figure 25 and Figure 26, Triangle_3 produced the largest variances in both cases – 34.96 feet^2 and 22.13 feet^2, respectively. But most surprising of all, against TX_1, Triangle_3B had the smallest error variance, which was very close to zero. We verified an approximate value of 2×10^{-29} feet^2. Referring to the topology layouts, we saw that Triangle_3B's nodes surrounded TX_1 and were relatively close (approximately 4-13 feet). Rectangle_1 (ID: 13) had the smallest error variance against TX_2 – 0.07 feet^2.

Additionally, we wanted to see how performance changed when topologies were switched from localizing TX_1 to localizing TX_2. More specifically, we wanted to observe the topologies that had the most dramatic changes to better understand how transmitter positioning played a role. The largest increase in position error was by Triangle_3B (12.85 feet) since it was initially close to TX_1 and then relatively far from TX_2. The Line_3 topology (ID: 19) had the largest decrease in position error (7.95 feet), which came to us as a surprise. Having a mean position error of 3.96 feet placed it within rank of the hexagon and rectangle topologies. This went contrary to our observation guidelines, which did not look favorably upon line topologies. The line topologies were not spatially diverse, they did not surround the transmitter, and they almost never had some form of symmetry about the transmitter (with the exception of Line_3 versus TX_1). However, from this dramatic performance improvement we learned that spatial diversity does not only apply to a degree of node scatter taken with respect to two dimensions, but it also refers to a range of separation distances taken along some axis, or a single dimension. In the case of Line_3 versus TX_2, this axis can be drawn such that it connects all of the node positions and terminates at the transmitter's location.

The largest error variance increase (or decrease in estimate precision) was by the All Sensors topology (5.96 feet^2). This was expected since the transmitter's position was moved from being surrounded by the topology's nodes to being located outside of the topology's perimeter. The largest decrease in error variance was by Small Triangle_1 (12.57 feet^2). Being more than 14 feet farther from TX_2 (than to TX_1) and that fact that it occupied a relatively small area gave Small Triangle_1 a large decrease in stability performance. In general, the hexagon topologies yielded smaller position errors, and the error variances for the rectangle topologies were consistently small. Both results agree with our first and third observation guidelines.

After we finished our initial survey, all of the numerical data for Figure 25 and Figure 26 were compiled into a table so that the topologies could be ranked. Our goal was to summarize the data according to how well the topologies helped to locate a transmitter by being both accurate (by having a low mean position error) and stable (by having a low error variance). We arranged the data into four columns as shown by the solid vertical lines in Table D1 in Appendix D. Then, we divided the rankings into thirds. For the top-third and bottom-third rankings, we created two histograms which represented the "best of the best" and the "worst of the worst" based on how frequently each topology appeared in the four columns belonging to the upper and lower rankings, respectively. We provide these histograms in Figure 27.



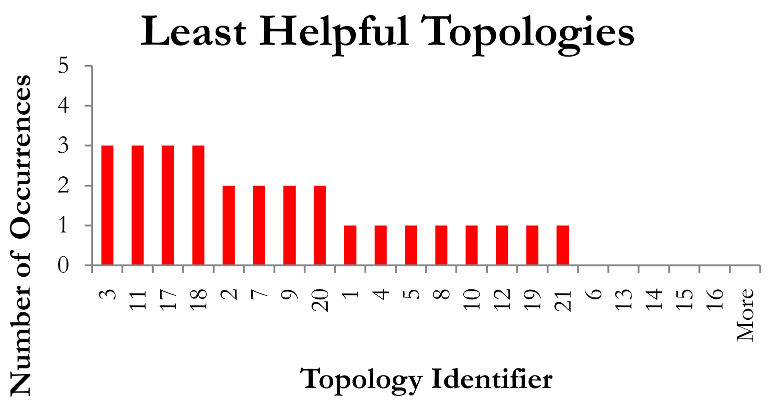


Figure 27: Histograms of the most helpful and least helpful sensor topologies

The Hexagon_1B (ID: 16) and Triangle_3 (ID: 3) topologies were ranked as the absolute best and absolute worst topologies, respectively. The time-varying position error plots in Figure 24 provide a qualitative idea of how different the two topologies performed. The hexagon's error was nearly constant for the duration of the trials, and the triangle's error was very erratic. Their qualities of performance are compared further when the position estimates are viewed as scatter plots. Figure 28 shows relatively benign scatter plots given by the Hexagon_1B topology, while Figure 29 depicts scatter plots of Triangle_3's position estimates. (The outliers were kept in both figures for emphasis.) The shapes of the position estimates in Figure 29 seemed to point towards the top-leftmost sensor (F117_B) instead of forming a relatively Gaussian distribution. In an effort to find out why the position estimates were scattered in this fashion, we took a look at the received power of the topologies' nodes.

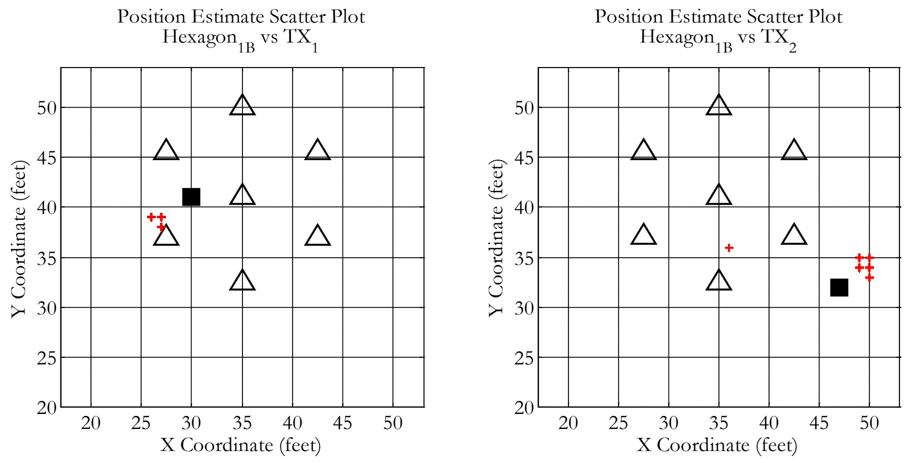


Figure 28: Scatter plot of Hexagon_1B's position estimates (59 estimates each)

The plots of Figure 30 depict the received power for the Triangle_3 (top row) and Hexagon_1B (bottom row) topologies according to the distance between their nodes and the transmitter. In keeping with communications theory, we expected to see an exponential decay of the received power. To make this trend more apparent, we converted the power values into the log domain and applied a linear fit to each case.

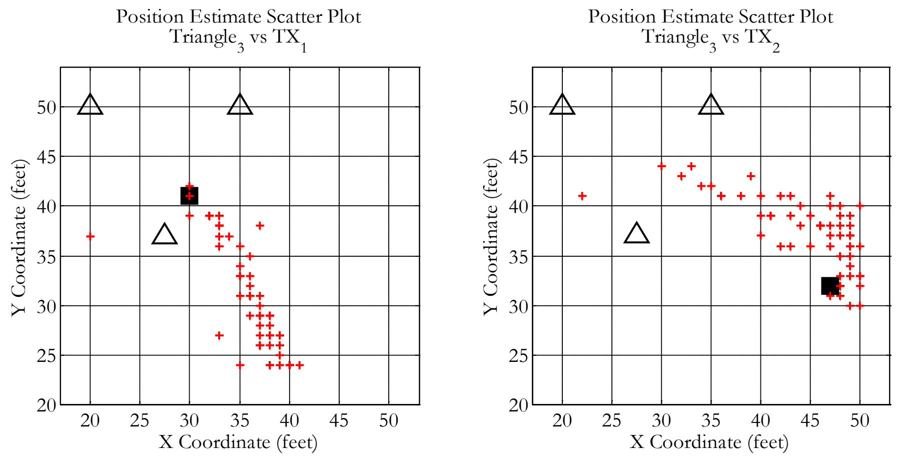


Figure 29: Scatter plot of Triangle_3's position estimates (59 estimates each)

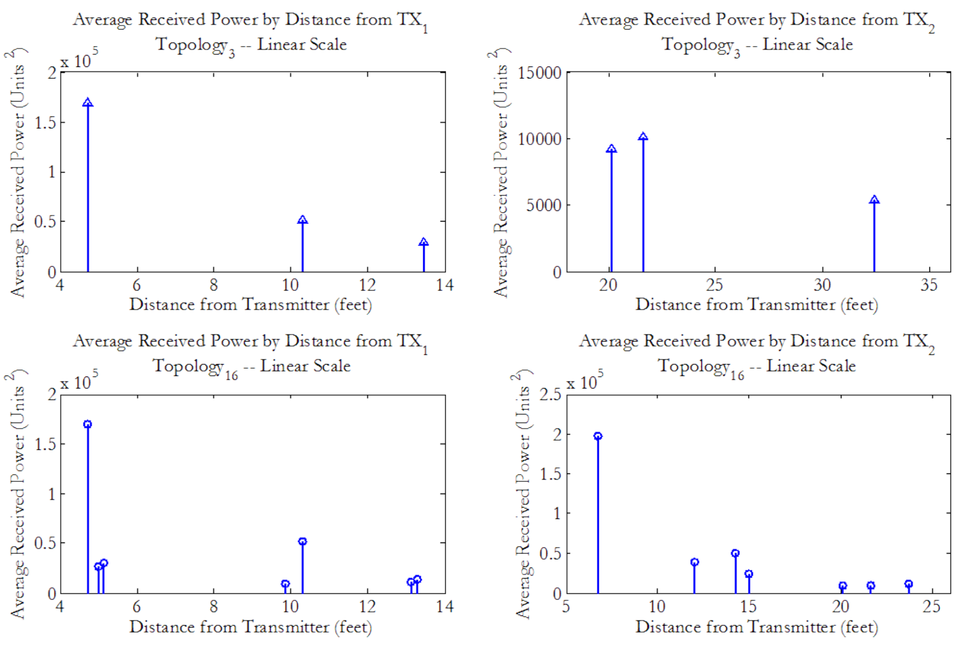


Figure 30: Received power by distance for Triangle_3 (top) and Hexagon_1B (bottom)

The log-domain equivalent plots for Figure 30 are shown in Figure 31 and Figure 32, as well as the residual plots from the linear fit tests. The linear fit has a negative slope in all cases, and the residuals are nearly symmetric about their respective zero-error lines. These indications suggest good exponential decay. However, we were concerned with the abnormally large power values at distances 10.3 and 14.2 feet in Figure 30's bottom-left and bottom-right plots, respectively. Both values belonged to F16_A. Going even further, we examined the node populations for the worst-of-the-worst topologies (from Figure 27) and found that of the four worst performing topologies, the most frequently-used node was F16_A.

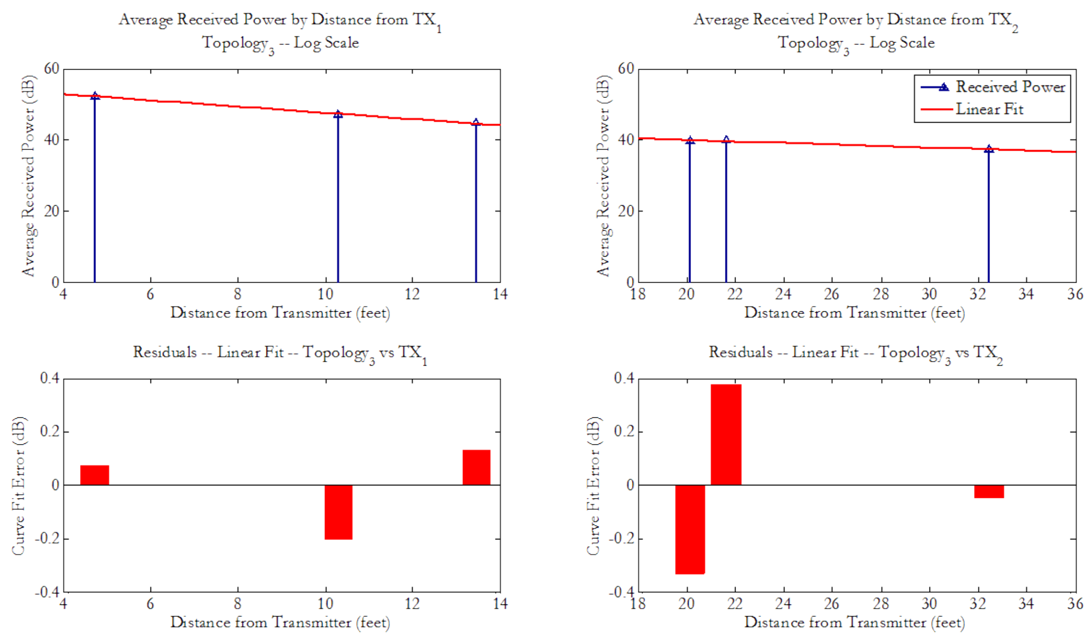


Figure 31: Log-domain representation of the average received power of Triangle_3's nodes with linear fit curves and residual their associated residuals

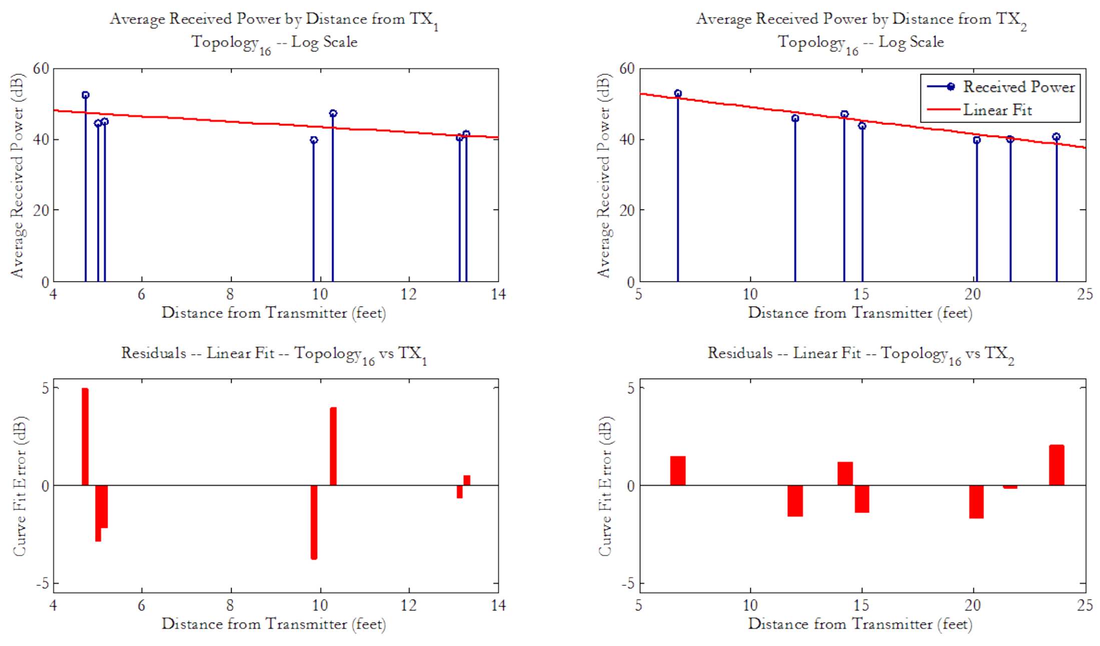


Figure 32: Log-domain representation of the average received power of Hexagon_1B's nodes with linear fit curves and residual their associated residuals

Given what we saw here, we concluded that the F16_A node may have been faulty. The USRP characterization data (Figure 16) showed near identical responses among the USRPs in a lab environment, however unlikely it seemed, there may have been something unknown to us that affected the F16 USRP at its Topology_A location. Since the sensor inclusion histogram (Figure 22, right) did not show F16 (ID: 2) to be one of the most frequently used nodes, we continued our comparisons of the real world data.

As we stated earlier, using a mean absolute position error is just one way to make comparisons. To expand our analysis we took into account topology size so that we could normalize the position error values, and thus compare any pair of topologies, regardless of their sizes. This kind of comparison is instrumental for doing a cost-benefit analysis. In effect it would allow us to find which topology gave the most value using the least resources under our experiment conditions. To form the normalized position error and normalized variance values we evaluated:

$$\text{error}_{\text{norm}} = \sqrt{\text{number of nodes } \times \text{mean}(\,\text{error}_{\text{radial}}^{2}\,)} \qquad (11)$$

and

$$\text{variance}_{\text{norm}} = \sqrt{\text{number of nodes } \times \text{var}(\,\text{error}_{\text{radial}}^{2}\,)} \qquad (12)$$

where the number of nodes have been used to scale the mean squared error and the variance of the square error. The results of our calculations are depicted in Figure 33 and .
The topology rankings are listed in Table D2 in Appendix D.

Applying the same ranking method as described earlier, we sorted the topologies according to their normalized mean position errors and their normalized error variances. We called this normalized ranking a measure of topology effectiveness, and they are reflected in Figure 33 and Figure 34. The least effective topologies were Line_2 and Small Triangle_1. Line_2's poor performance was expected since its node positions did not line up alongside the transmitter in either position (as Line_3's nodes were arranged against TX_1). The nodes of Small Triangle_1 surrounded TX_1 in an almost ideal fashion – the transmitter was located at its center. Therefore, we expected that this supposedly ideal arrangement would reflect favorably in the topology's position estimates, especially since the nodes were no more than 5 feet away. But this turned out not to be the case. Upon revisiting the localization code, we realized that the variance of the distances between each node and the transmitter were close to zero. Since this value was taken as a denominator term, it reflected as an unstable topology, analytically. When viewed in this regard, Small Triangle_1 performed as it should have against TX_1. Small Triangle_1 was expected to fare poorly since it was very far removed from TX_2.

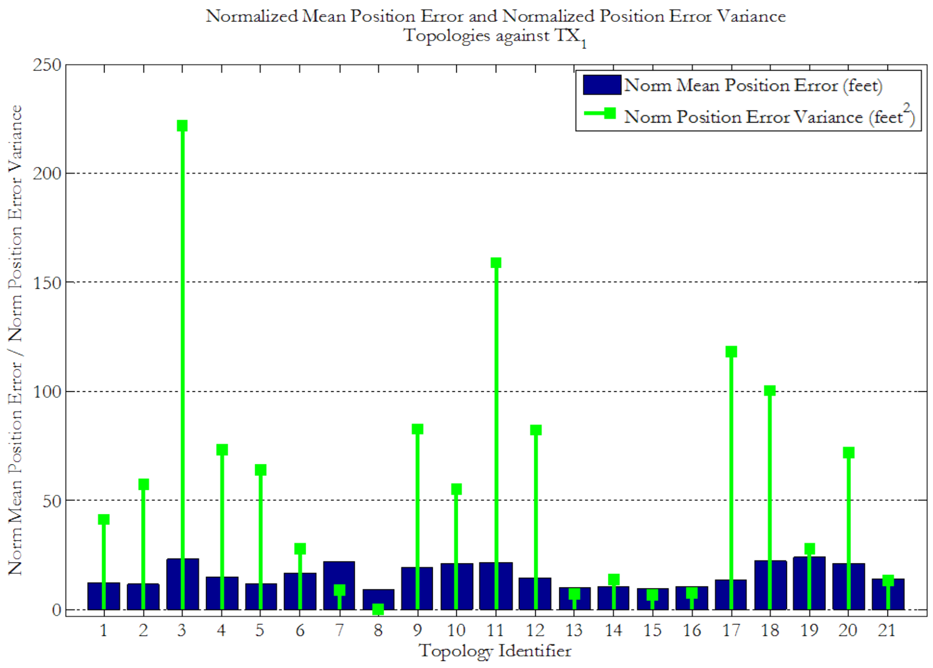


Figure 33: Normalized position error for all real world topologies against TX_1

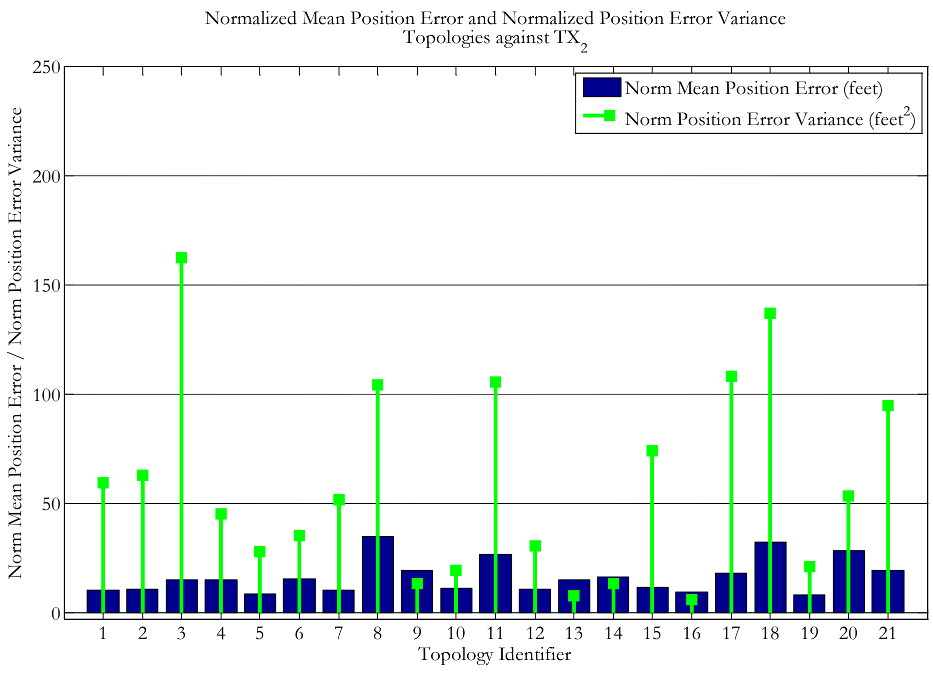


Figure 34: Normalized position error for all real world topologies against TX_2

The most effective topology was Hexagon_1B. It was consistently more stable and more accurate than the other topologies. In contrast, despite having a relatively low mean position error (approximately 4 feet), Hexagon_1's error variance (4.37 feet^2), which was comparable with Line_2's error variance (4.43 feet^2), translated into a large normalized width. This goes to show that if a topology with many nodes does not perform significantly better, it should be penalized. As the rankings indicate, if given the choice to select either the hexagon with the center node or without, the more effective choice would be to include the center node because it added significant value.

4.5 Simulation Results

Using the same metrics and the same topologies, we created similar position error plots for the simulation results. Figure 35 and Figure 36 were used to draw comparisons as we had done before, and we made note of any similarities between the trends given by the simulated data with what we saw from the real world data. Part of the value in emulating our topologies using the simulation was to be able to see how performance would change given an environment where the nodes were identical. An initial look at the un-normalized mean position errors revealed that the topologies' results occurred in clusters and that similar topologies were performed similarly (with the exception of Line_1). For example, Triangles 1 through 5 (the 3-node triangles) had similar mean position errors and were separate from Triangles 6 through 10 (the 4-node triangles). The clustering of the errors was a good indication that there were distinctions between the different types of topologies. However, the nearly monotonic mean error values suggested a trend that disregarded the various topology orientations with respect to the transmitter; it did not seem correct that Triangle_1 (ID: 1) and Triangle_2 (ID: 2) should ever have comparable mean position error values.

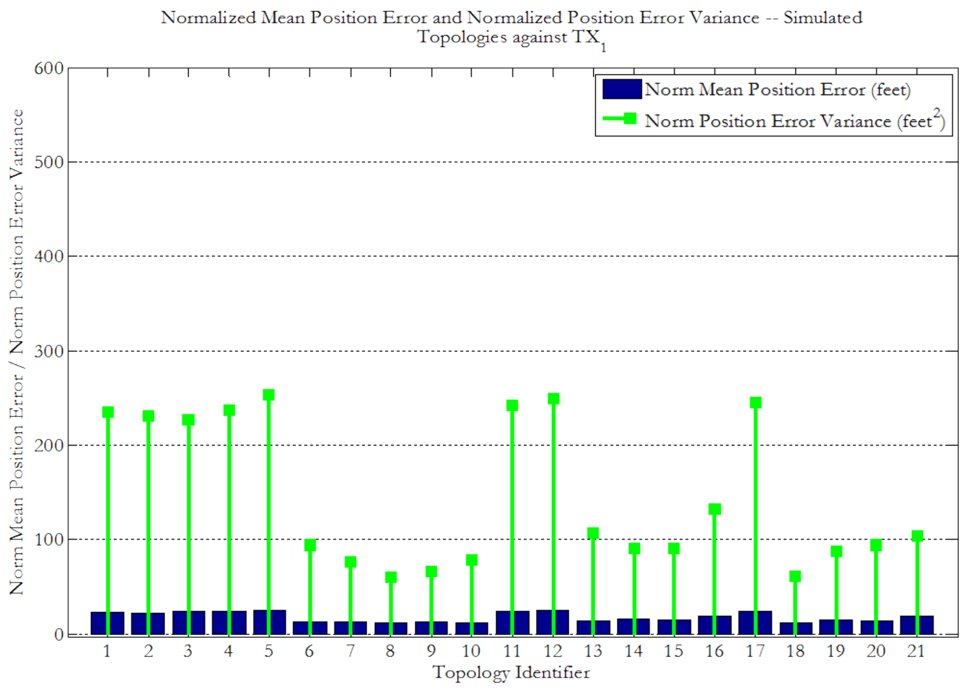


Figure 35: Mean absolute position error for all simulated topologies against TX_1

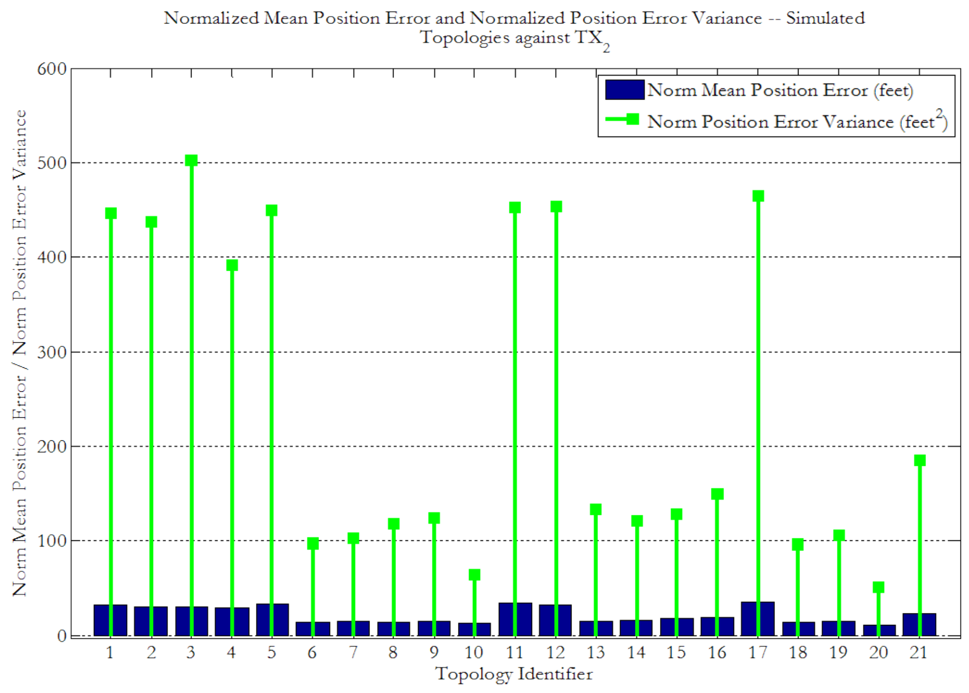


Figure 36: Mean absolute position error for all simulated topologies against TX_2

According to the simulated data, Triangle_5 (ID: 5) and Line_1 (ID: 17) had the largest mean position errors against TX_1 and TX_2, respectively. This claim against Triangle_5 does not match the real world data, which indicated that Triangle_5 was in the third tier for "Most Helpful" topology. On the other hand, the claim regarding Line_1 did correspond with the experimental results, which had it ranked as the "Least Helpful" topology. A general comparison showed that the position error variance for the simulation data was five times greater than the real world position error variance, on average. This comparison convinced us that the simulation results were not as close to the real world data as we hoped.

V. Conclusions and Recommendations

5.1 Introduction

The purpose of this chapter is to summarize the conclusions of our research. We begin with a review of our research objectives. Then, we discuss the significance of what our efforts accomplished. Finally, we provide our recommendations for future work.

5.2 Review of Research Objective

As introduced in Section 1.3, our first objective was to demonstrate that collaborative localization could be implemented in real time. We were successful in implementing a real-time process (Appendix B), however we chose to discuss our more flexible post-processing approach in Chapters 3 and 4. This approach not only proved to be highly reconfigurable, but also an excellent learning tool. Our second objective was to examine the position estimate accuracy of our testbed. We gave a comparison of position estimation performance for all of our topologies using mean absolute error plots. Additionally, we compared topology performance using a normalized metric that took into account the topology's size. Overall we showed our hexagon topology was the most effective (with mean position errors near 4 feet), and that our triangle and line topologies were the least effective (with mean position errors approaching 12 feet). Finally, our third objective was to compare simulation results with real-world results. Unfortunately, our comparisons showed that these results were not in total agreement. We are led to believe that despite our efforts to emulate a larger sensor array, the simulation code we used was better equipped for much larger sensor networks (for S = 100 nodes instead of S = 10 nodes).

5.3 Significance of Research

This research broke new ground by actually demonstrating principles of collaborative localization -- a topic more commonly explored using simulation experiments alone. A major benefit of our research is that the flexible sensor network implemented here could be modified to extend more concepts that, to date, still only exist on paper. Modern military communications devices are being extended to support the Network-Centric Warfare (NCW) vision, in which network cohesion and information sharing take precedence. As the capabilities of information resources are extended to support automated collaboration, all users of this technology would benefit from the higher quality of information that it would offer. Our sliver of research has shown that it can be done.

5.4 Recommendations for Future Research

The research effort presented in this thesis can be extended in several ways. First, we recommend that our experiments are repeated using directional antennas at each transmitter location. In this way, more compelling arguments for the benefits of collaborative location can be made through a real-world demonstration. Next, we recommend that the sensor network undergo evolutionary modification that includes: removing the LAN requirement for time syncing, reconfiguring the USRPs' hardware and software to operate in another (or multiple) frequency bands, and having the radios implement a common resource map to transfer digital data among each other. The objective would be to construct a scenario in which a successful outcome depends on how well the radios can collaborate. Finally, we recommend the creation of a shared language (similar in style to semantic web) that would be extensible by design, so that a more formal process can be put in place whereby the radios could query each other and an observer could query the sensor network.

5.5 Summary

In this chapter, we presented the conclusions of this research. Also, we reviewed the objectives that we intended to meet. We discussed the significance of our research and finally, offered some recommendations for future research.

Appendix A. GNU Radio Sample Code

The GNU Radio software includes several development tools and programming libraries to interact with the USRP hardware. For example, the Python scripting language serves as the primary language used to command the hardware. In general, this is done by calling libraries, classes and signal processing blocks; and then linking those pieces together in a more human-readable, object-oriented programming (OOP) style. Since this process takes some time to get acclimated to, we strongly recommend any GNU Radio and USRP hardware newcomer to follow the series of tutorials created by Dawei [53].

An excerpt of code is provided in Figure A1. This code was taken from a Python script (`usrp_wfm_rcv.py`) which uses the USRP hardware to capture wideband FM radio transmissions, and then plays the received audio through a computer's speakers.

```
#!/usr/bin/env python

from gnuradio import gr, gru, eng_notation, optfir
from gnuradio import audio
from gnuradio import usrp
from gnuradio import blks
from gnuradio.eng_option import eng_option
from gnuradio.wxgui import slider, powermate
from gnuradio.wxgui import stdgui, fftsink
from optparse import OptionParser
imprt usrp_dbid
import sys
import math
import wx
```

Figure A1: Excerpt code from `usrp_wfm_rcv.py` showing import statements

Line 1 is an optional statement that allows the script to be executable from the command line. The text that follows in lines 3 through 14 are import statements. They invoke modules and packages that would be used throughout the script and are typically called upon at the beginning. Modules are files that contain Python definitions and statements. Packages are collections of modules that have similar functions. There also exist sub-packages that group even more closely related modules together. For example, Line 9 of Figure A1 states: from the `wxgui` subpackage (of the larger `gnuradio` package) import the `stdgui` and `fftsink` modules. Note that it is not required to import all modules in a sub-package. Here only two modules were needed from `wxgui`, and so only two were imported.

Classes in Python operate much the same way as classes do in other OOP languages. Figure A2 shows the `wfm_rx_graph` class declaration which defines the user interface and signal processing routine.

```
class wfm_rx_graph (stdgui.gui_flow_graph):
    def __init__(self, frame, panel, vbox, argv):
        stdgui.gui_flow_graph.__init__ (self, frame, panel, vbox, argv)

        parser = OptionParser(option_class = eng_option)
        parser.add_option("-R", "--rx-subdev-spec", type="subdev", default=None,
                          help="select USRP Rx side A or B (default=A)")
        parser.add_option("-f", "--freq", type="eng_float", default=100.1e6,
                          help="set frequency to FREQ", metavar="FREQ")
        parser.add_option("-g", "--gain", type="eng_float", default=None,
                          help="set gain in dB (default is midpoint)")
        parser.add_option("-D", "--audio-device", type="string", default="",
                          help="pcm device name. E.g., hw:0,0 or surround51 or /dev/dsp")

        (options, args) = parser.parse_args( )
```

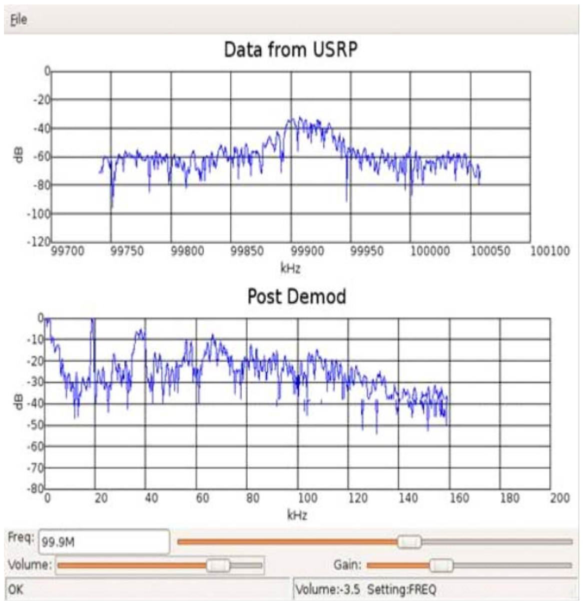
Figure A2: Excerpt code from `usrp_wfm_rcv.py` showing class declaration

In Line 30, the class declaration utilizes a package-module relationship to define a new class (`wfm_rx_graph`) that is derived from the `gui_flow_graph` module (a sub-module of the `sdgui` module imported in Line 9 of Figure A1).

In the lines following, `wfm_rx_graph` is given a set of initial conditions (Lines 31 and 32), defines command-line option assignments (Lines 34 – 42), and calls a parsing function to sort the user-input options (Line 44). These options are listed at the command line and may specify parameter such as the type of receiver daughterboard to be used and the center frequency it will be tuned to. For example, the following statement may be entered at the command line:

```
./usrp_wfm_rcv.py   -R  B   -f  99.9M
```

This command would invoke the `usrp_wfm_rcv.py` script to use the receiving daughterboard located on Side B of the USRP motherboard (likely a Basic RX or TVRX daughterboard), and to tune its center frequency to 99.9 MHz. What results is a display similar to the one shown in Figure A3.



Freq: 99.9M

OK

Volume:-3.5 Setting:FREQ

Figure A3: Graphical display of usrp_wfm_rcv.py tuned to 99.9 MHz

Joining all of the software pieces is performed easily using and intuitive structure. Line 89 of Figure A4 sets the audio device (by default the host computer's sound card) as the audio sink. Line 92 joins the radio, filter, signal processing blocks, volume adjustment control, and sound card in the order they are passed in the connect function.

```
89  audio_sink = audio.sink( int(audio_rate), options.audio_device)
90
91  self.connect( self.u, chan_filt, self.guts, self.volume_control, audio_sink )
```

Figure A4: Excerpt code taken from usrp_wfm_rcv.py showing audio sink declaration and software block connect statement

This concludes our introductory look at GNU Radio code. However, several other online sources are available to get started. Here they are listed in descending order according to our personal preference: [53], [54], [55], [56].

Appendix B. Real-time Collaborative Localization

In this section we discuss the central software tool used to perform our experiments, the RSSI Localization Code. The code is shared between two files. The first file makes use of an already-existing GNU Radio script (`usrp_spectrum_sense.py`) by appending our station detection algorithm, RSSI algorithm, and sensor collaboration procedures. The second file is a Python port of the omnidirectional node localization function which was originally written as M-code by Martin in support of the simulations described in [9].

The localization experiments are conducted using a sensor array which observes an emitter node. The emitter node transmits audio much like a local radio station by mixing audio signal from a song with a carrier tone. As such, each sensor must be able to identify any potential stations within its viewable band that need to be localized. This process should be performed iteratively and consistently among all members of the sensor array.

The `usrp_spectrum_sense.py` script is provided upon installation of the GNU Radio development toolkit. Its primary functions are to continuously sweep through the entire FM band (88 MHz - 108 MHz) at evenly-spaced center frequencies, collect a stream of complex signal data, and calculate the magnitude squared of the signal in the frequency domain – a power spectral density (PSD). As Figure 11 depicts, each sweep-iteration in the time domain ends with an un-normalized PSD calculation for a sampled signal 128 μsec in duration. Although the example script provides PSD data for all frequencies in the FM band, we only need a sub-band of frequencies to be monitored in order to carry out our experiments. Thus, the tuning algorithm was changed to remain fixed on a user-defined center frequency.

Determining the presence of a radio station requires a systematic approach of distinguishing potential channels of interest versus noise. We devised an algorithm that calculates the average PSD given an averaging interval and then uses a sliding window to single out clusters of frequencies with relatively high energy – potential radio stations. The time averaging process depicted in Figure 12 shows several PSDs occurring at regular time intervals being reduced to a single PSD whose shape is smoother and less noisy. Given a five-second time interval, approximately 430 PSDs will be used to form an average. Once a time-averaged PSD is produced, stations are more readily identified because a significant portion of the noise is averaged out.

The sliding-window process relies on two concepts: (1) a fixed number of contiguous data points (the window), and (2) movement of the window's boundaries along an axis (the sliding motion). A window placed anywhere along the frequency axis of the time-averaged PSD will enclose a cluster of received power values. If this window is aligned with a station, the sum of the received power values is declared the RSSI of that station and the center x-value of the window is declared the station's center frequency. But, to ensure that our algorithm only acknowledges valid stations, each realization of the sliding window must go through a vetting process, and the window must be made appropriately wide.

Every time-averaged PSD has an associated average power that is calculated across its entire viewable bandwidth (a global average power). For each sliding window position, the average power bounded by the window (a local average power) is compared to the global average power. If the local average power is equal to or greater than the global average power, we are confident that a station resides partially or completely within the window. Any window position for which the local average power is less than the global average power is ignored. Figure B1 identifies typical window alignment conditions.

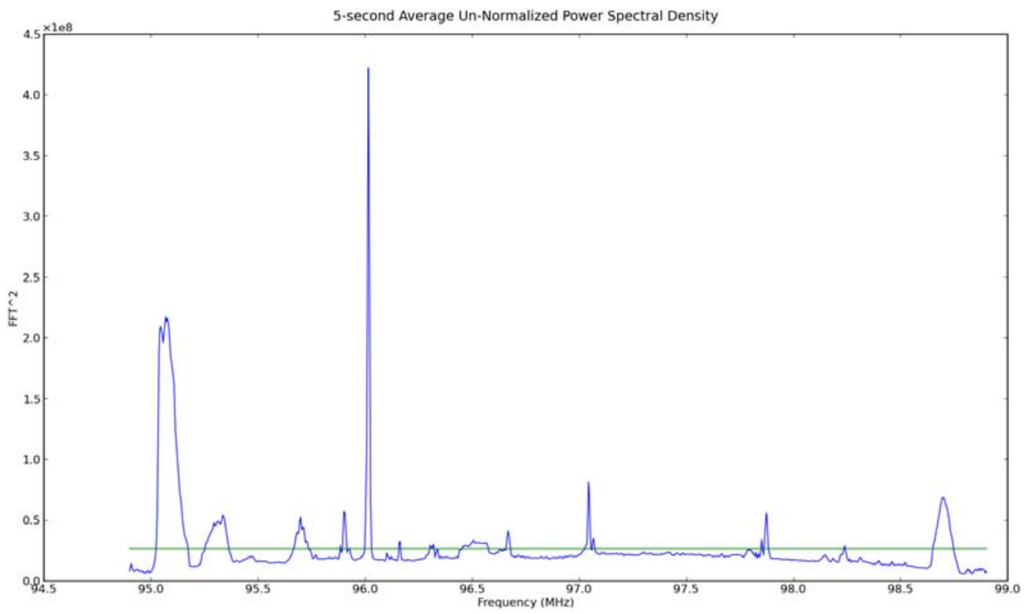


Figure B1: Typical un-normalized PSD showing the global average power across the viewable band

Figure B2 shows the transition from a time-averaged PSD to a pseudo-RSSI plot once the sliding window has traversed the entire frequency axis. By using this average power comparison technique, we see that weaker signals and noise are excluded.

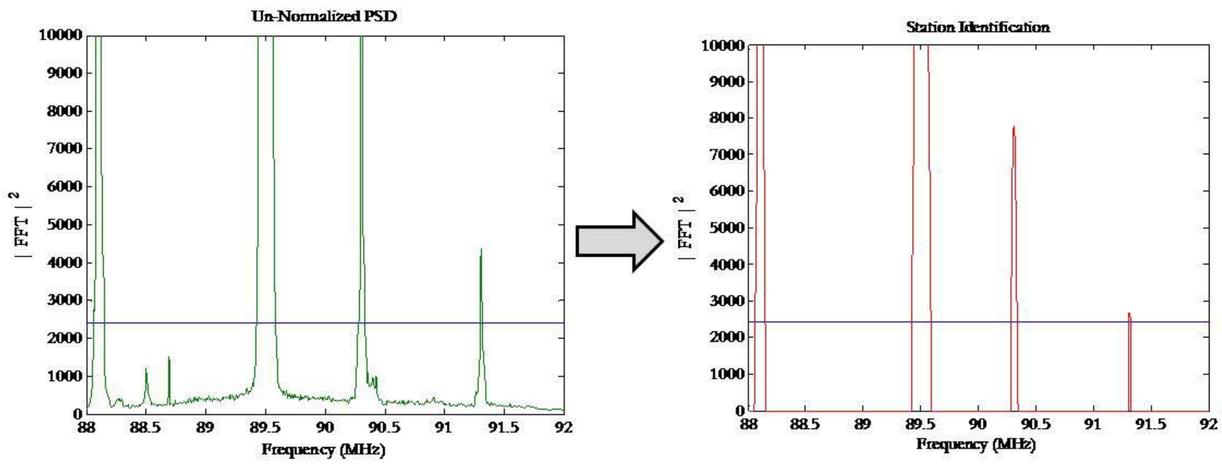


Figure B2: Time-averaged PSD (left) transition to a pseudo-RSSI plot (right)

A brief comparison of station detection results for several window widths is offered in Figure B3. The differences in window size affect the number of stations that are detected successfully. Once possible stations are identified, the center x-values are retrieved. Finally, summations are performed using the values adjacent to each center frequency (the only remaining non-zero values).

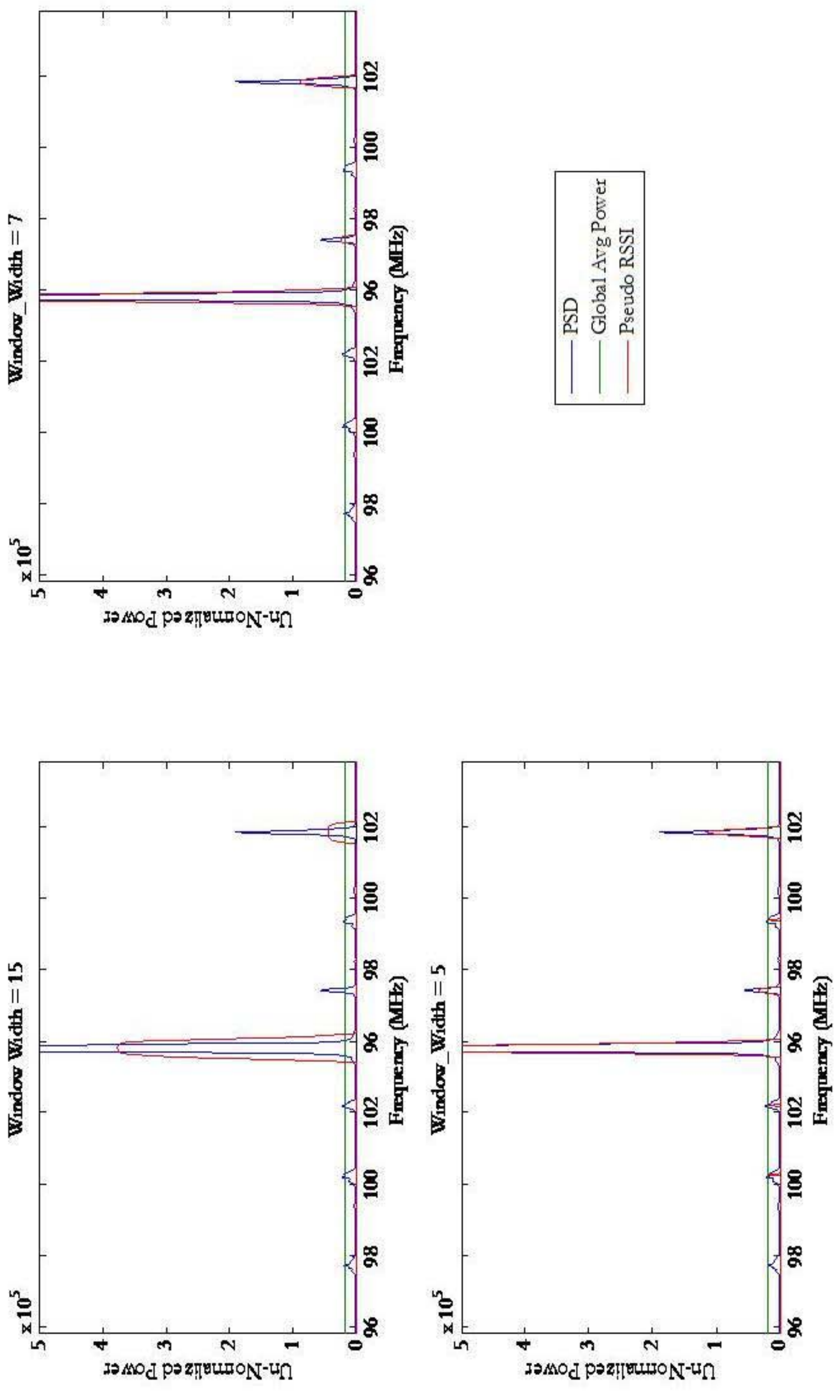


Figure B3: Sliding window width comparison with pseudo-RSSI plot overlay

Appendix C. Sensor Topologies

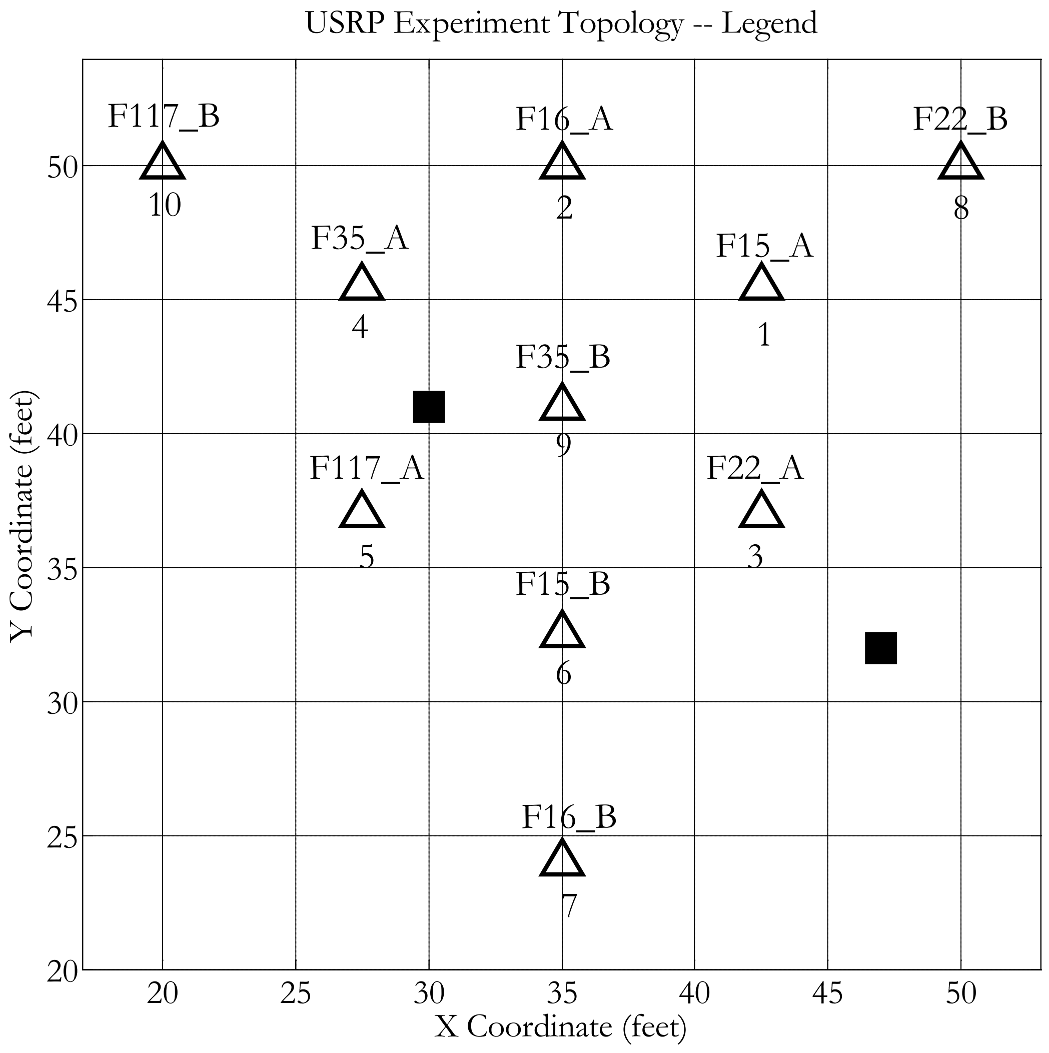
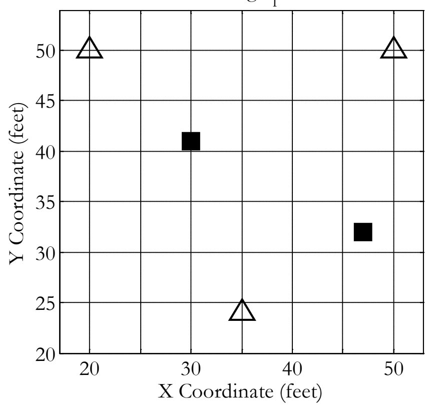
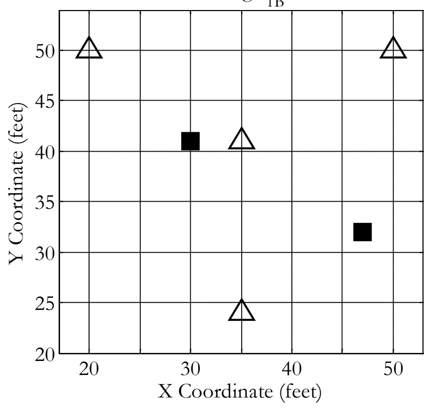


Figure C1: Topology legend showing sensor identities

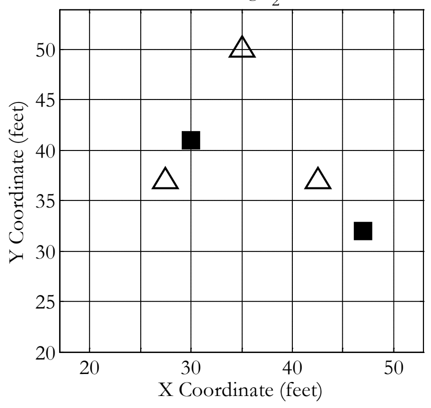
Triangle$_1$



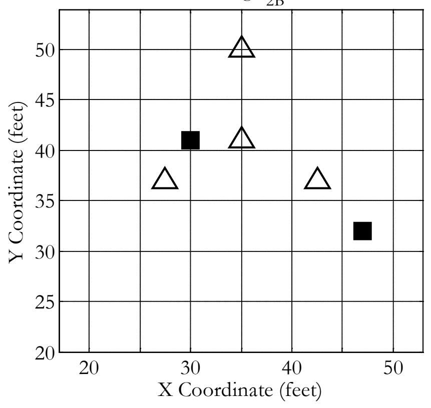
Triangle$_{1B}$



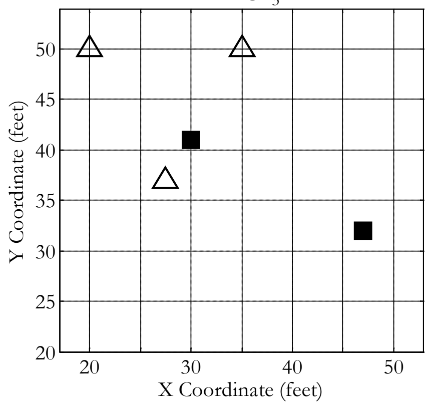
Triangle$_2$



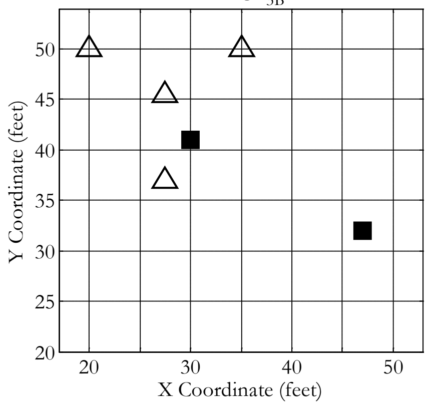
Triangle$_{2B}$

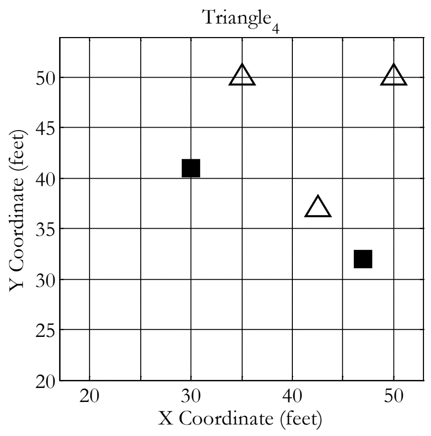


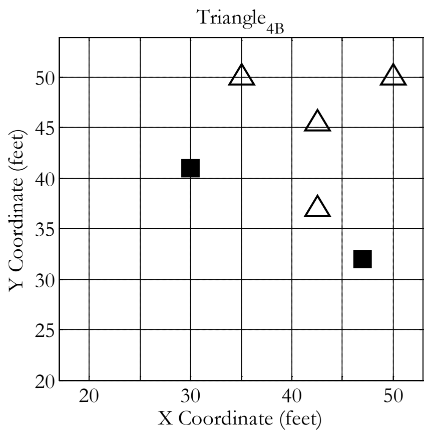
Triangle$_3$

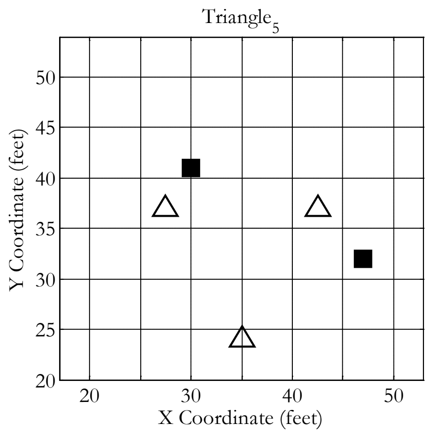


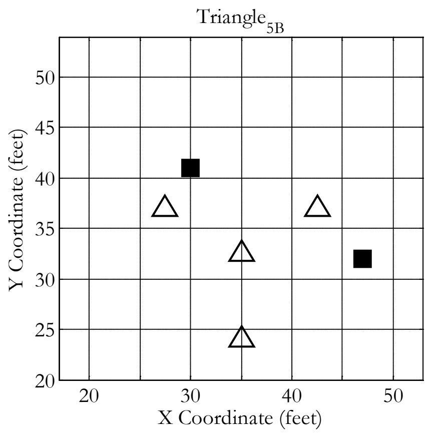
Triangle$_{3B}$

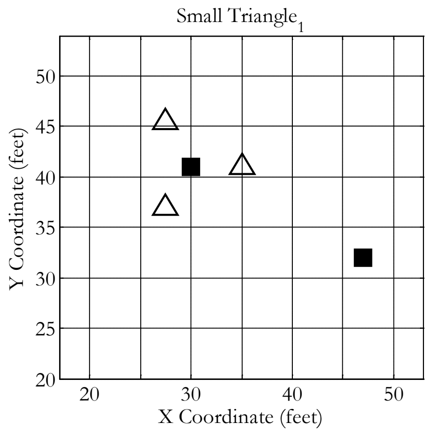


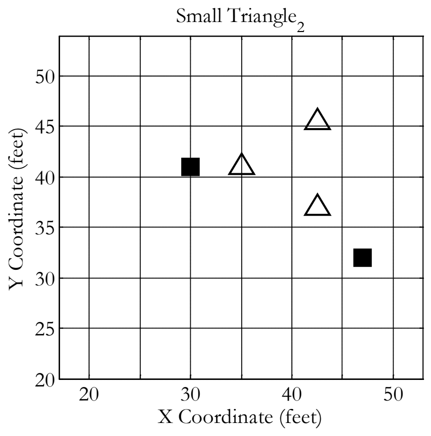

Triangle$_4$
Y Coordinate (feet)
X Coordinate (feet)
20
25
30
35
40
45
50

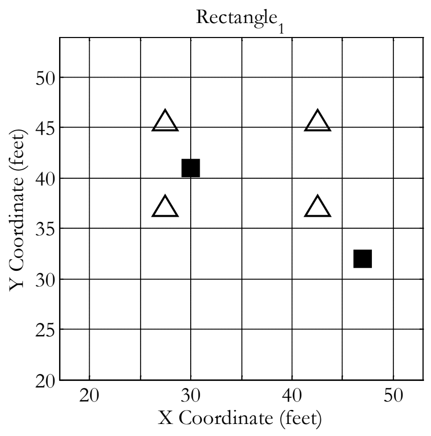

Triangle$_{4B}$
Y Coordinate (feet)
X Coordinate (feet)

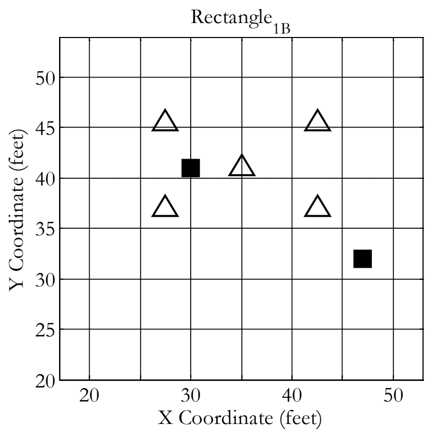

Triangle$_5$
Y Coordinate (feet)
X Coordinate (feet)

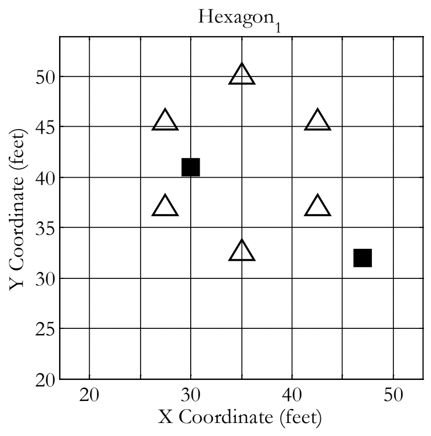

Triangle$_{5B}$
Y Coordinate (feet)
X Coordinate (feet)

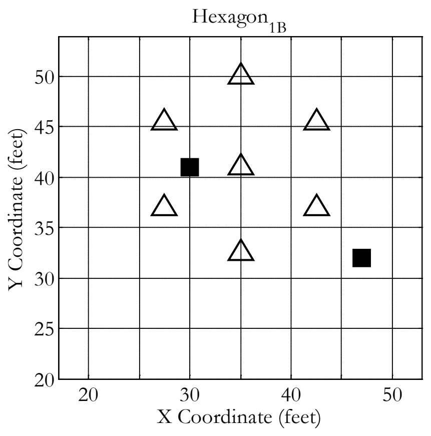

Small Triangle$_1$
Y Coordinate (feet)
X Coordinate (feet)

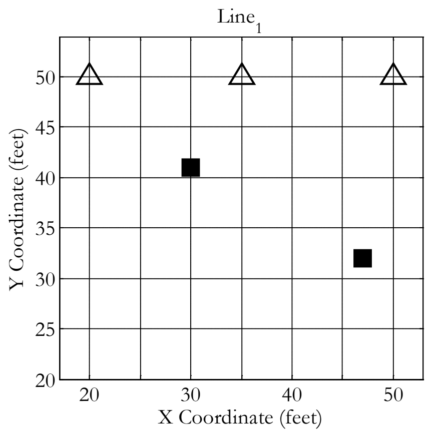

Small Triangle$_2$
Y Coordinate (feet)
X Coordinate (feet)

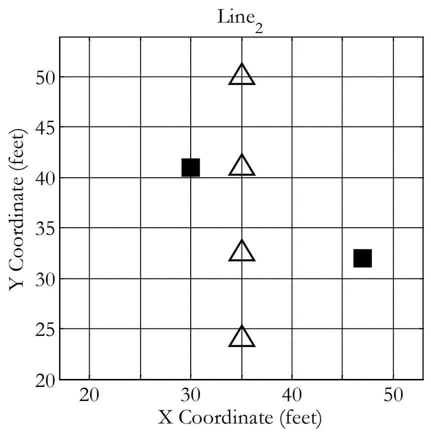

Rectangle$_1$
Y Coordinate (feet)
X Coordinate (feet)
20
25
30
35
40
45
50


Rectangle$_{1B}$
Y Coordinate (feet)
X Coordinate (feet)


Hexagon$_1$
Y Coordinate (feet)
X Coordinate (feet)


Hexagon$_{1B}$
Y Coordinate (feet)
X Coordinate (feet)


Line$_1$
Y Coordinate (feet)
X Coordinate (feet)


Line$_2$
Y Coordinate (feet)
X Coordinate (feet)

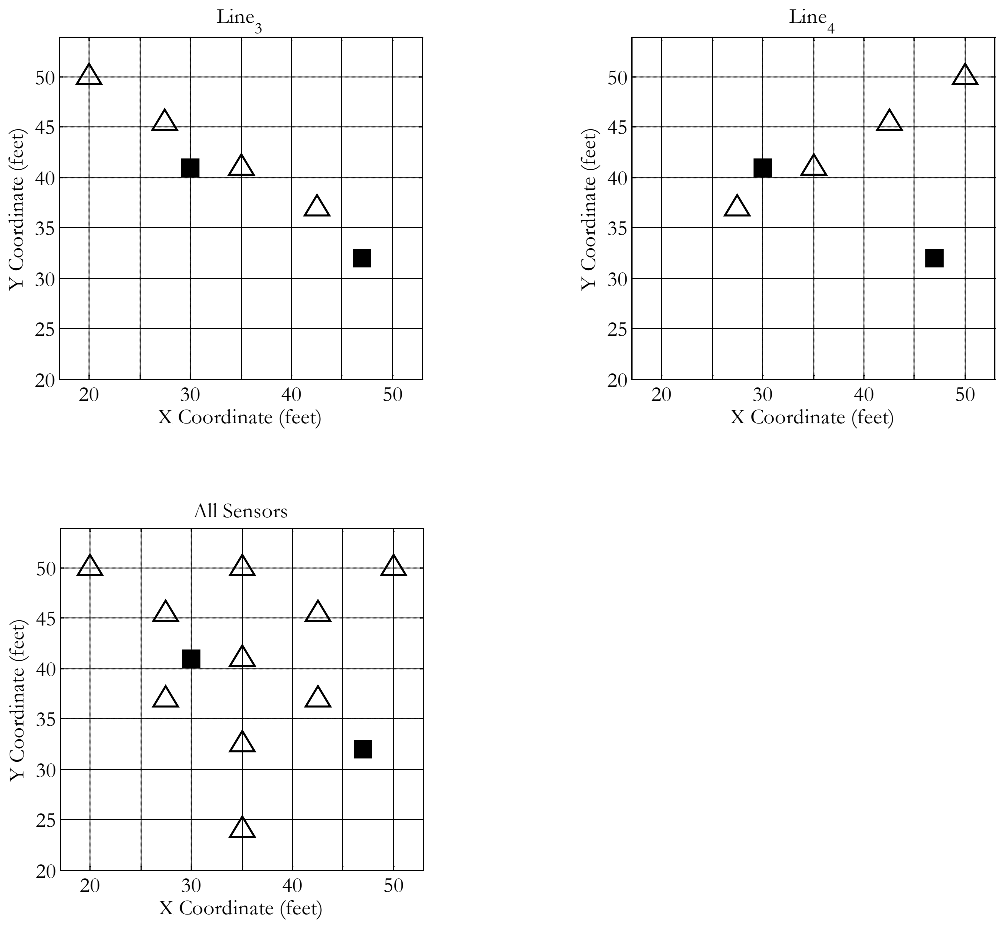


Figure C2: Sensor topologies with sensor locations shown as triangles and transmitter locations as squares

Appendix D. Topology Rankings

Table D1: Topology rankings using real world data

	ACCURACY (mean position error)				STABILITY (position error variance)			
	Against TX1		Against TX2		Against TX1		Against TX2	
RANK	Topology	Mean Error	Topology	Mean Error	Topology	Variance	Topology	Variance
1	15	3.7851	16	3.4969	8	0	13	0.1361
2	16	3.8742	19	3.9639	7	0.106	16	0.1714
3	21	4.4043	15	4.1711	15	0.1734	9	0.176
4	8	4.4721	5	4.5138	16	0.1827	14	0.2176
5	14	4.4874	7	4.5874	13	0.209	10	0.4269
6	13	4.7751	1	5.2602	21	0.2275	20	0.4773
7	5	5.883	10	5.4406	19	0.2965	19	0.536
8	2	6.0491	2	5.5109	14	0.3846	6	0.5836
9	17	6.4806	21	5.6101	6	0.4617	12	0.6694
10	1	6.9204	12	6.0102	10	0.7911	4	0.7743
11	12	7.8911	3	7.1398	20	0.8504	5	0.815
12	4	8.1566	14	7.2281	1	0.8974	8	0.8849
13	6	8.3275	13	7.3737	18	1.2699	11	1.0455
14	9	9.1528	6	7.5511	4	1.2851	15	1.0669
15	20	10.2453	4	8.4658	9	1.3206	18	1.0752
16	10	10.2566	9	9.582	2	1.3784	7	1.1767
17	7	10.8327	17	9.9171	12	1.5	21	1.2662
18	18	10.8892	20	14.1901	5	1.5044	1	1.3626
19	11	11.6025	11	15.2708	11	2.0898	2	1.393
20	19	11.9167	18	16.0315	17	2.1901	17	1.3999
21	3	11.9714	8	17.3203	3	3.0173	3	2.4006

Table D2: Topology ranking using normalized real world data

	ACCURACY (mean position error)				STABILITY (position error variance)			
	Against TX1		Against TX2		Against TX1		Against TX2	
RANK	Topology	Mean Error	Topology	Mean Error	Topology	Variance	Topology	Variance
1	8	8.9443	19	8.1967	8	0	13	7.8173
2	15	9.3083	5	8.2852	13	7.4665	16	8.1412
3	13	9.5846	16	9.2937	15	8.4581	9	13.5931
4	14	10.1722	1	10.1997	7	9.2233	14	14.9856
5	16	10.2932	7	10.2511	16	10.2008	10	19.503
6	5	11.3779	2	10.6341	14	15.66	19	21.1357
7	2	11.4581	12	10.6509	21	20.9803	5	24.399
8	1	12.3611	10	11.0069	19	28.1883	12	27.041
9	17	13.4278	15	11.4092	6	28.5459	6	35.9264
10	21	13.9976	13	14.7568	1	36.2594	4	39.8855
11	12	14.5701	3	14.7711	2	50.3413	1	52.3639
12	4	14.7746	4	14.8928	10	55.9041	7	52.7111
13	6	16.7514	6	15.2716	5	56.4046	20	54.2762
14	9	19.0111	14	16.1901	4	64.5759	2	55.5537
15	10	20.7422	17	17.8112	12	72.7113	15	92.6852
16	20	20.7552	9	19.1763	20	73.1738	11	93.1902
17	11	21.291	21	19.3715	9	84.2739	17	95.3259
18	7	21.6693	11	26.6827	18	102.4515	8	106.4256
19	18	22.3303	20	28.4408	17	104.3838	18	139.5096
20	3	23.0875	18	32.3341	11	140.3247	3	143.3983
21	19	23.8612	8	34.8109	3	195.9913	21	152.5331

Table D3: Topology ranking using simulated data

	ACCURACY (mean position error)				STABILITY (position error variance)			
	Against TX1		Against TX2		Against TX1		Against TX2	
RANK	Topology	Mean Error	Topology	Mean Error	Topology	Variance	Topology	Variance
1	21	5.1896	20	4.9924	9	1.2202	20	1.1182
2	10	5.2941	10	5.8392	18	1.2786	10	1.2632
3	15	5.4234	14	6.1013	8	1.3335	15	1.518
4	8	5.589	8	6.2603	21	1.4635	18	1.5373
5	18	5.6732	18	6.269	14	1.4775	6	1.5691
6	6	5.7202	21	6.4252	15	1.5639	19	1.5974
7	7	5.8916	6	6.4489	7	1.5688	7	1.5987
8	20	6.0417	16	6.4644	10	1.6004	21	1.6608
9	9	6.068	15	6.5404	19	1.6176	14	1.7014
10	13	6.1814	13	6.6697	13	1.7184	9	1.7061
11	16	6.2797	19	6.7218	6	1.7352	8	1.7535
12	14	6.3338	9	6.8331	16	1.7581	16	1.8012
13	19	6.8189	7	6.9852	20	1.7961	13	1.8982
14	2	11.8661	3	14.85	3	2.4599	4	3.8823
15	1	12.4499	4	15.1855	17	2.703	17	3.9818
16	17	12.7192	2	15.6584	11	2.7506	11	4.1858
17	4	12.7209	1	16.7918	4	2.7836	5	4.2118
18	11	12.8941	12	17.0073	2	2.8021	2	4.2361
19	3	13.1468	5	17.3369	1	2.8185	1	4.2694
20	12	13.2218	11	18.0146	5	2.8756	12	4.3373
21	5	13.2304	17	18.9017	12	2.9139	3	4.8076

Table D4: Topology ranking using normalized simulated data

	ACCURACY (mean position error)				STABILITY (position error variance)			
	Against TX1		Against TX2		Against TX1		Against TX2	
RANK	Topology	Mean Error	Topology	Mean Error	Topology	Variance	Topology	Variance
1	10	12.2792	20	10.8893	8	61.4206	20	52.1065
2	8	12.3206	10	12.6679	18	62.2494	10	65.472
3	18	12.3864	18	13.8882	9	67.2809	18	98.2556
4	9	13.0293	8	14.2543	7	78.6881	6	99.0538
5	7	13.2665	6	14.2662	10	80.0974	7	103.9888
6	6	13.2793	19	14.8072	19	90.1743	19	107.1703
7	20	13.9539	9	15.1892	20	95.6038	8	120.2211
8	13	14.0508	13	15.2427	6	96.4126	9	126.0681
9	19	15.0164	7	15.2893	14	103.338	13	136.1069
10	15	15.2271	14	15.5165	13	108.7081	14	138.1652
11	14	15.5493	15	17.5741	15	114.0277	15	159.9327
12	21	18.7128	16	19.4483	21	166.4936	16	202.1052
13	16	18.9133	21	22.7365	16	178.6767	21	298.7552
14	2	22.6124	4	29.3679	3	200.4101	4	345.6858
15	1	23.5574	3	30.3857	2	203.9151	2	386.3385
16	17	23.8342	2	30.639	1	207.6708	1	394.2617
17	4	23.9417	1	32.439	4	208.9645	5	396.9483
18	11	24.1752	12	32.875	11	213.9479	11	400.3386
19	3	24.2288	5	33.2051	17	216.7458	12	400.6844
20	5	24.8749	11	34.2342	12	220.2961	17	410.5908
21	12	24.9117	17	35.3745	5	224.2536	3	444.0298

Bibliography

[1] Anonymous "Spectrum policy task force report," Federal Communications Commission, Tech. Rep. Technical Report 02-135, Nov, 2002.

[2] Anonymous (2008, Apr). Wireless data drives high growth in U.S. telecommunications industry says new report. *Fierce Wireless* [Online]. *2009(01/11),* pp. 1. Available: http://www.fiercewireless.com/press-releases/wireless-data-drives-high-growth-u-s-telecommunications-industry-says-new-report

[3] A. Singh. (2008, Oct). Telecommunications future growth predicted. *TMCnet* [Online]. *2009(01/11),* pp. 1. Available: http://ipcommunications.tmcnet.com/topics/ip-communications/articles/43864-telecommunications-future-growth-predicted.htm

[4] K. Gerwig. (2008, Feb). Telecom market heading for healthy growth, TIA projects. *SearchTelecom* [Online]. *2009(01/11),* pp. 1. Available: http://searchtelecom.techtarget.com/news/article/0,289142,sid103_gci1302583,00.html

[5] J. Ewing. (2008, May). Telecom's last great growth markets. *BusinessWeek* [Online]. *2009(01/11),* pp. 1. Available: http://www.businessweek.com/magazine/content/08_22/b4086056645511.htm

[6] J. Mitola III and G. Q. Maguire Jr. (1999, Cognitive radio: Making software radios more personal. *Personal Communications, IEEE 6(4),* pp. 13-18.

[7] S. M. Mishra, A. Sahai and R. W. Brodersen, "Cooperative Sensing among Cognitive Radios," *Communications, 2006. ICC '06. IEEE International Conference on,* vol. 4, pp. 1658-1663, 2006.

[8] C. R. C. M. da Silva, W. C. Headley, J. D. Reed and Youping Zhao, "The application of distributed spectrum sensing and available resource maps to cognitive radio systems," *Information Theory and Applications Workshop, 2008,* pp. 53-57, 2008.

[9] R. K. Martin and R. W. Thomas, "Algorithms and Bounds for Estimation of Directionality of Primary Spectrum Users," Submitted to *IEEE Trans. Wireless Comm,*

[10] L. Genco. (2009, Old-time radio: The golden years. [Online]. *2009(01/10),* pp. 1. Available: http://www.old-time.com/golden_age/index.html

[11] Anonymous (2009, VSS forecast communications industry sectors 2006 - 2011. [Online]. *2009(01/10),* pp. 1. Available:

http://www.marketingcharts.com/television/communications-spend-to-reach-1-trillion-in-08-internet-to-surpass-all-ad-segments-in-2011-1206/vss-forecast-communications-industry-sectors-2006-2011jpg/

[12] R. I. Lackey and D. W. Upmal. (1995, Speakeasy: The military software radio. *Communications Magazine, IEEE 33(5),* pp. 56-61.

[13] J. Mitola III. (1992, Software radios-survey, critical evaluation and future directions. *Telesystems Conference, 1992. NTC-92. , National* pp. 13/15-13/23.

[14] D. Wilson, "Emerging technology: Will software-defined radio shake up communications?" *Electronics Design, Strategy, News,* vol. 2009, pp. 1, Aug 21. 2007.

[15] A. Shah and V. Bose. (1999, Accelerating evolution of the cellular infrastructure using software radios. [Online]. Available: http://vanu.com/wp-content/resources/publications/1999accelerating_evolution_of_cellular_infrastructure.pdf

[16] B. Fette, *Cognitive Radio Technology.* ,1st ed.New York: Elsevier, 2006, pp. 656.

[17] J. Mitola III. (2008, Nov). Joseph mitola III homepage. [Online]. *2009(02/25),* pp. 1. Available: http://web.it.kth.se/~maguire/jmitola/

[18] R. W. Thomas, "Cognitive networks," *New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005. 2005 First IEEE International Symposium on,* pp. 352-360, 2005.

[19] N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero III, R. L. Moses and N. S. Correal. (2005, Locating the nodes: Cooperative localization in wireless sensor networks. *Signal Processing Magazine, IEEE 22(4),* pp. 54-69.

[20] G. Jordt, "Evaluation of energy costs and error performance of range-aware, anchor-free localization algorithms for wireless sensor networks," *Department of Electrical and Computer Engineering,* pp. 11-12, March. 2006.

[21] M. Aatique, "Evaluation of TDOA Technique fo Position Location in CDMA System," *Master's Thesis at Virginia Polytechnic Institute and State University,* pp. 4-5, August. 1997.

[22] C. Drane, M. Macnaughtan and C. Scott. (1998, Positioning GSM telephones. *Communications Magazine, IEEE 36(4),* pp. 46-54, 59.

[23] J. Heyer and L. C. Schuette. (2004, Sept). NRL - unattended ground sensor network. [Online]. *2009(2/9),* pp. 1. Available: http://www.nrl.navy.mil/content.php?P=04REVIEW185

[24] Anonymous (2006, Jan). Unattended ground sensors. *Defense Update* [Online]. *2009(2/9),* pp. 5. Available: http://defense-update.com/features/du-1-06/feature-ugs.htm

[25] I. Hakala, M. Tikkakoski and I. Kivela. (2008, Wireless sensor network in environmental monitoring - case foxhouse. *Sensor Technologies and Applications, 2008. SENSORCOMM '08. Second International Conference on* pp. 202-208.

[26] R. Bharadwaj, S. Mukhopadhyay, M. Peralta, K. Shenai and S. Majumder. (2008, Cognitive distributed networks in environmental e-science. *Future Trends of Distributed Computing Systems, 2008. FTDCS '08. 12th IEEE International Workshop on* pp. 192-198.

[27] Garam Park and Yoo Jaeheung. (2008, Suggesting infection causes monitoring system based on wireless sensor network for hospital infection control. *Advanced Communication Technology, 2008. ICACT 2008. 10th International Conference on 1*pp. 642-647.

[28] J. Markoff. (2008, Jul). Can't find a parking spot? check smartphone. *NY Times* [Online]. *Business*Available: http://www.nytimes.com/2008/07/12/business/12newpark.html?ex=1373601600&en=9e06e6d3c756ca1a&ei=5124&partner=permalink&exprod=permalink

[29] Anonymous (2008, Sept). 2003 CBECS detailed tables: Summary. [Online]. *2009(2/9),* pp. 1. Available: http://www.eia.doe.gov/emeu/cbecs/cbecs2003/detailed_tables_2003/detailed_tables_2003.html#enduse03

[30] T. Carlon. (2005, Sept). Building systems program: Wireless. [Online]. *2009(2/9),* pp. 1. Available: http://www.buildingsystemsprogram.pnl.gov/wireless/in_building.stm

[31] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks,* vol. 38, pp. 393, 2002.

[32] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum and L. Viennot. (2001, Optimized link state routing protocol for ad hoc networks. *Multi Topic Conference, 2001. IEEE INMIC 2001. Technology for the 21st Century. Proceedings. IEEE International* pp. 62-68.

[33] C. E. Perkins and E. M. Royer. (1999, Ad-hoc on-demand distance vector routing. *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA '99. Second IEEE Workshop on* pp. 90-100.

[34] J. Park, J. Han, K. Kang and K. H. Lee, "The Registry for Sensor Network Discovery," *Engineering Complex Computer Systems, 2007. 12th IEEE International Conference on,* pp. 129-137, 2007.

[35] P. K. Varshney. (1997, Multisensor data fusion. *Electronics & Communication Engineering Journal 9(6),* pp. 245-253.

[36] M. Bedworth and J. O'Brien. (2000, The omnibus model: A new model of data fusion? *Aerospace and Electronic Systems Magazine, IEEE 15(4),* pp. 30-36.

[37] B. Dasarathy, *Decision Fusion.* Los Alamitos, CA: IEEE Computer Society Press, 1994, pp. 381.

[38] O. Kessler and B. Fabian, "Estimation and ISR process integration," Defense Advanced Projects Research Agency, Washington, D.C., 2001.

[39] D. Smith and S. Singh. (2006, Approaches to multisensor data fusion in target tracking: A survey. *Knowledge and Data Engineering, IEEE Transactions on 18(12),* pp. 1696-1710.

[40] S. Thrun. (2006, Winning the DARPA grand challenge: A robot race through the mojave desert. *Automated Software Engineering, 2006. ASE '06. 21st IEEE/ACM International Conference on* pp. 11-11.

[41] A. Toga and P. Thompson, "The role of image registration in brain mapping," *Image and Vision Computing,* vol. 19, pp. 3-24, 2001.

[42] D. L. Hall and J. Llinas. (1997, An introduction to multisensor data fusion. *Proc IEEE 85(1),* pp. 6-23.

[43] D. L. Hall and S. A. H. McMullen, "Introduction to multisensor data fusion," in *Mathematical Techniques in Multisensor Data Fusion* ,2nd ed.Anonymous MA: Artech House, 2004, pp. 1-35.

[44] Anonymous (2008, Dec). The free software definition. [Online]. *2009(01/14),* pp. 1. Available: http://www.gnu.org/philosophy/free-sw.html

[45] Q. Norton. (2006, Jun). GNU radio opens an unseen world. *Wired* [Online]. *2009(1/10),* pp. 2. Available: http://www.wired.com/science/discoveries/news/2006/06/70933

[46] Ettus Research. (2008, Nov). Ettus research order page. [Online]. *2009(01/10),* pp. 1. Available: http://www.ettus.com/orderpage.html

[47] E. Blossom. (2004, Nov). Exploring GNU radio. [Online]. *2009(2/7),* pp. 1. Available: http://www.gnu.org/software/gnuradio/doc/exploring-gnuradio.html

[48] Anonymous (2008, Dec). ECE team takes first place in smart radio challenge '08. [Online]. *2009(02/25),* pp. 1. Available: http://www.ece.cmu.edu/news/story/2008/12/ece_team_takes/

[49] R. Peter. (2008, Jun). Usrp/FAQ/Intro/FPGA. [Online]. *2009(02/18),* pp. 1. Available: http://www.gnuradio.org/trac/wiki/UsrpFAQ/Intro/FPGA

[50] Anonymous (2001, Sept). 4937 DI5 RF tuner module advance data sheet. [Online]. Available: http://www.comsec.com/usrp/microtune/4937-DI5-3x8899-2.pdf

[51] Anonymous (2009, Epoch converter. [Online]. *2009(02/21),* pp. 1. Available: http://www.epochconverter.com/

[52] Anonymous (2009, Jan 22). History for wright-patt AFB, OH. *Weather Underground* [Online]. *2009(01/26),* pp. 1. Available: http://www.wunderground.com/history/airport/KFFO/2009/1/22/DailyHistory.html?req_city=NA&req_state=NA&req_statename=NA

[53] D. Shen. (2006, 28 Feb 06). USRP tutorials. [Online]. *2009(10 Jan 09),* pp. 1. Available: http://www.nd.edu/~jnl/sdr/docs/tutorials/

[54] J. Blum. (2008, Sept). Learning by example - josh knows | GNU radio. [Online]. *2009(01/17),* pp. 1. Available: http://www.joshknows.com/?key=gnuradio#example

[55] R. Kern. (2008, Nov). NumPy for matlab users. [Online]. *2009(01/17),* pp. 1. Available: http://www.scipy.org/NumPy_for_Matlab_Users

[56] F. Abbas. (2007, Nov). Simple user manual for GNU radio 3.1.1. [Online]. *2009(01/17),* Available: http://www.ece.jhu.edu/~cooper/SWRadio/Simple-Gnuradio-User-Manual-v1.0.pdf

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 074-0188*

1. REPORT DATE *(DD-MM-YYYY)* 27-03-2009	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From – To)* March 2008 – March 2009

4. TITLE AND SUBTITLE IMPLEMENTATION OF COLLABORATIVE RF LOCALIZATION USING A SOFTWARE-DEFINED RADIO NETWORK	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Honore, Augustine, A., First Lieutenant, USAF	**5d. PROJECT NUMBER** 09 - 265
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology, Graduate School of Engineering and Management (AFIT/EN), 2950 Hobson Way, Building 640, WPAFB OH 45433-8865	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GE/ENG/09-20
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/RYRE AFMC, Attn: Mr. Vasu Chakravarthy, 2241 Avionics Circle, WPAFB AFB, OH 45433 DSN: 785-5579 (vasu.chakravarthy@wpafb.af.mil)	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/RYRE
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This thesis investigates the use of collaboration between sensor nodes that were tasked with localizing a radio frequency emitter. Localization is a necessary component for dynamic spectrum access. Using a set of software-defined radios as our sensors and a received signal strength-based maximum likelihood localization algorithm, we successfully localized transmitting nodes based on their received signal strength. Our experiment was conducted outdoors using a flexible topology that could be shaped into 21 sub-topologies that varied in size, and orientation with respect to the transmitters. This was made possible through application of a time shift concept and a post-processing technique. We were able to compare our real world results with the simulated results of the same topologies. Although our simulation results did not fully comply with our real world results, we observed some common trends regarding effective topology design.

15. SUBJECT TERMS

Collaboration, Cognitive Radio, Cognitive Network, Localization, Received Signal Strength (RSS), Software-Defined Radio

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Ryan Thomas, Capt, USAF (ENG)
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-6565, ext 4613 (ryan.thomas@afit.edu)
U	U	U	UU	99	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18