Geolocation Using Smartphone Sensors

A Major Qualifying Project Report

Submitted to the Faculty of

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree in Bachelor Science

In

Electrical and Computer Engineering

By

Ryan Darnley

Nathaniel Peura

Justin Seeley

Sponsored By:

Matt Beals

Vito Mecca

Jonathan Watson

Date: October 9, 2018

Project Advisor:

_____

Professor Edward A. Clancy

This material is based upon work supported by the Department of the Air Force under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of the Air Force.

Abstract

Geolocation is the process of estimating an object's location on Earth [Djuknic and Richton, 2001]. Most commonly, geolocation performs in a "direct" way, i.e., geolocation involves the manipulation of Global Navigation Satellite System (GNSS) data to determine the location of a GNSS receiver. A less common geolocation application is the location estimation of an object of interest in an "indirect" way, i.e., the geolocation of an object of interest not co-located with any equipment such as a GNSS receiver. Referred to as "indirect geolocation" in this report, this method of location estimation utilizes existing knowns and reference points to determine the final geographic location of an object of interest in the distance.

This report presents a method of indirect geolocation leveraging the pose – location and orientation – of multiple observers. Using a GNSS receiver to determine location and an Inertial Measurement Unit (IMU) to determine orientation, this indirect geolocation method creates an imaginary intersection point from each observer's pointing vector. MEMS technology has reduced the size, weight, and cost of GNSS and IMU systems, however, these advantages have drawbacks in accuracy. Thus, in an attempt to find a less expensive and application specific device that has adequate sensors and sensor fusion capabilities, this report explores the feasibility of using smartphones to perform indirect geolocation.

The developed method of indirect geolocation uses an intersection algorithm to estimate the location of an object of interest given two or more observers' location and orientation. Equipped with algorithms for both two-dimensional (2-D) and three-dimensional (3-D) space, simulations were used to characterize algorithm performance. These simulations showed that there exists an optimal angle of 90^{o} that minimizes intersection mean RMS error. They also showed that position error creates a bias in the intersection error, while orientation error affects the intersection error as a function of distance. Ultimately such work culminated in a series of field tests where location estimations commonly formed within 3 meters of the truth value with both observers over 30 meters in the distance.

Statement of Authorship

Over the course of the summer of 2018 at MIT Lincoln Laboratory, Nathan and Ryan completed sections of the two-dimensional intersection algorithm and initial Kalman Filter design as detailed in Chapter 3. All group members contributed equally during the scheduled 9-week period at MIT Lincoln Laboratory.

All sections were reviewed by all group members, but the primary authors for each section are as follows:

Ryan Darnley: Section 2.1, 2.2, 2.3, 3, 4.1, 6.1, 6.2, 6.3

Nathan Peura: Section 2.1, 2.2, 2.3, 3, 4.3, 4.4, 5.2, 5.3

Justin Seeley: Section 2.1, 2.2, 2.4, 4.2, 4.4, 5.1, 5.3

All Team Members: Abstract, Executive Summary, Introduction, Discussion, Conclusion

Acknowledgements

There are many people to whom we can attribute the success of our project including, but not limited to, Professor Edward Clancy, Professor George Heineman, Vito Mecca, Jonathan Watson, Matthew Beals, and the entire MIT Lincoln Laboratory community. Most notably, we would like to thank Group 108 for creating a challenging but welcoming work environment. Professor Edward Clancy provided his direct support and suggestions as our advisor. Professor Heineman directed the MIT Lincoln Laboratory MQP site. Vito Mecca, Jonathan Watson, and Matthew Beals provided valuable advice and input throughout the progression of the project.

Executive Summary

Introduction

Geolocation is the determination of an object's location on Earth. In its most standard application, geolocation is the process of determining the geographical location of a measuring device. Referred to as direct geolocation in this report, this process usually leverages a Global Navigation Satellite System (GNSS) receiver, which estimates its location using information transmitted by satellites. Less commonly, geolocation can also be the process of determining the geographical location of an object of interest not co-located with a measuring device. Referred to as indirect geolocation in this report, this method utilizes known locations and orientations to a common target to deduce this target's location. Specifically, indirect geolocation is the focus of this report.

In order to perform indirect geolocation, the location and orientation, data reflecting the pose – location and orientation – of two devices with respect to a target must be collected. In practice, several different sensor systems can provide location and orientation data; however, this report focuses on two: the Global Positioning System (GPS) receiver for location and the inertial measurement unit (IMU) for orientation. A myriad of products that contain GPS and IMU sensor systems are available for purchase. Often called Attitude and Heading Reference Systems (AHRS), these devices utilize integrated circuit (IC) technology to create small, robust pose estimation systems. Despite their effectiveness, these systems are often heavily customized for industry and military applications. Thus, their inflexible and expensive nature lend these systems to be undesired by the more ordinary customer. Consequently, this report analyzes the feasibility of using a more abundant and flexible platform to perform indirect geolocation: smartphones.

In order to test the practicality and effectiveness of smartphone-based indirect geolocation systems, this report first explores the theory of indirect geolocation and the requirements necessary to perform it. Specifically, this section highlights two major requirements: a pose algorithm and an intersection algorithm. In order to satisfy the first requirement of indirect geolocation, creation of a pose algorithm, the report discusses the concept of sensor fusion by means of an extended Kalman Filter (EKF). With the first requirement satisfied, the report then discusses the second requirement, the intersection algorithm in both two-dimensional (2-D) and three-dimensional (3-D) space. Finally, having satisfied the two requirements necessary to perform indirect geolocation, the report discusses experimental results obtained via simulations and field tests.

Indirect Geolocation

As prior stated, indirect geolocation is the process of determining the geographical location of an object of interest not co-located with a measuring device such as a GNSS receiver. For clarification, Figure 1 illustrates a typical indirect geolocation application. The object of interest, represented by the buildings, has no co-located equipment to determine its location. Meanwhile, two observers, A and B, each have devices capable of determining their respective pose. If both observers orient their equipment so that they are both facing the object of interest, each device creates a pointing vector indicative of its orientation. When elongated infinitely, this pointing vector passes through the object of interest.

Ultimately, by utilizing the known pose of the equipment for two observers, the location of the object of interest can be determined by finding the intersection between each equipment's pointing vectors. The theory of indirect geolocation, however, becomes much more complicated when errors are presented.

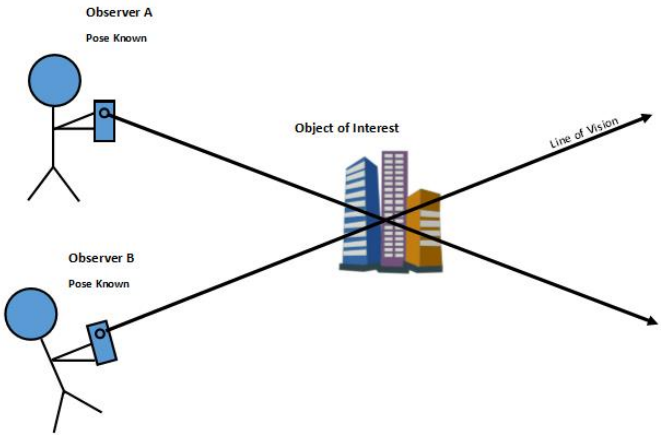

Figure 1: Indirect Geolocation

Sensor Fusion

Typically, smartphones are equipped with a variation of a GNSS receiver and the components of an IMU (accelerometer, gyroscope, and magnetometer). Unfortunately, the hardware used by smartphones is far from perfect. Corrupted by noise, drift, and external distortions, smartphone sensors typically produce inadequate pose estimations when used without filtering. If the strengths of individual sensors are leveraged together, however, the computed pose estimation can be drastically more accurate. Thus, by performing a process known as sensor fusion, more accurate pose estimations can be generated. Although many different algorithms can be used for sensor fusion, this report focuses on the extended Kalman Filter (EKF). A nonlinear variation of the conventional Kalman Filter, the EKF leverages a process model and a measurement model to create an optimal state estimate.

The proposed EKF had the process model based on the gyroscope and the measurement model based on the accelerometer and magnetometer. As each model produced its own orientation estimate, the EKF fused the two models together so that it attained a minimum distance error from truth. In order to fuse the respective strengths of each model, the EKF represented each model with a precision factor. Known as the covariance, this matrix determined the trustworthiness and reproducibility of each orientation estimate. As such, creating a gain factor based on the ratio between each model's covariance matrix, a more optimal orientation estimate was created.

Two-Dimensional Intersection Algorithm

A two-dimensional intersection algorithm was created that determined the unique intersection point shown in Figure 1. Specifically, the algorithm used many trigonometric principles to determine the intersection point and was ultimately validated using a zero-error model. To characterize the algorithm's performance in an environment with errors, Monte Carlo simulations were then run. First, a simple scenario was used for the Monte Carlo simulation, as shown in Figure 2. Denoted as observers A and B, each observer had two types of error applied to them – position and orientation. The position error applied to both observers was zero-mean Gaussian noise with a standard deviation of 2.5 meters. Meanwhile, the orientation error was zero-mean Gaussian noise with a standard deviation of 1 degree. Each Monte Carlo simulation was subsequently run for 25,000 iterations.

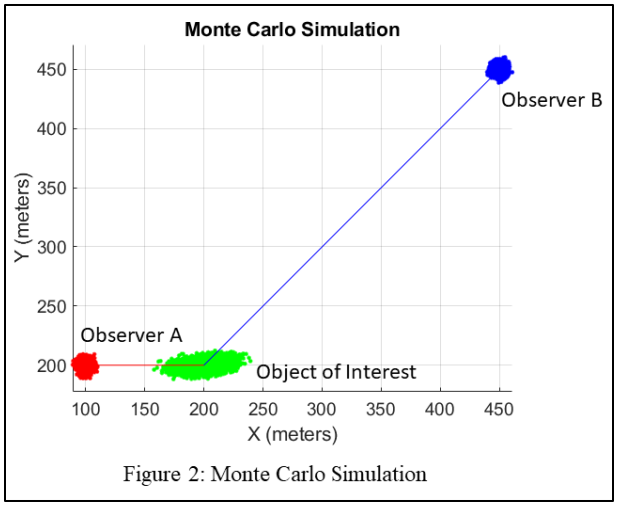

Figure 2: Monte Carlo Simulation

From all simulations the intersection point cloud around the object of interest was characterized. The individual X and Y components of the intersection cloud were found to be Gaussian distributed. Furthermore, the RMS error was found to follow a Rayleigh distribution, but only when the angle between the observers was 90°.

Following, a series of simulations was performed, each using distinct geometry, to characterize the algorithm in a more real-world environment where the observers could be anywhere in the two-dimensional plane and have various orientations. The findings for the 2-D Monte Carlo simulations can be seen after the 3-D Intersection Algorithm section below.

Three-Dimensional Intersection Algorithm

To supplement the two-dimensional intersection algorithm, a three-dimensional intersection algorithm was created to add the ability to geolocate points in three-dimensional space. As three-dimensional vectors are unlikely to intersect in 3-D space, a least-squares pseudoinverse method was used to find the approximation or "closest point" of intersection between observer pointing vectors. Ultimately, scenarios were used to validate the functionality of the least-squares method 3-D algorithm using simple geometry and distance graphs.

Simulations were used to characterize the performance of the algorithm with noisy inputs similar to those of real-world measurements. In order to simulate a noisy environment, a simple Monte Carlo simulation of 25,000 iterations was created to model real-world sensor errors. In this scenario, two observers were looking at an object directly between them – a target at (50, 50, 0) – with observer A having an angle of 45° and observer B having an angle of -45°. Each observer had a 2.5 meter uncertainty in the X and Y location, a 5 meter uncertainty in the Z location, and 2.5 degrees of uncertainty in their pitch and yaw angle. These uncertainties were standard deviations of independent zero-mean Gaussian distributions. One example scenario is shown in Figure 3 below.

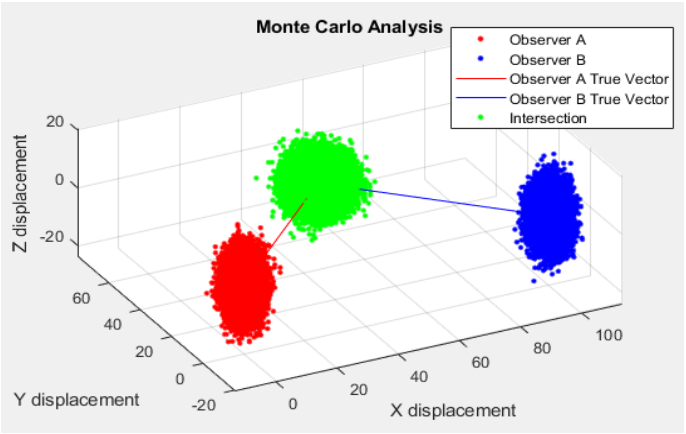

Figure 3: 3-D Intersection Monte Carlo

After each simulation, the intersection point cloud around the object of interest was characterized. The individual X, Y, and Z components of the intersection cloud were found to be Gaussian distributed.

Following, a series of simulations were performed to characterize the algorithm in a more real-world environment where the relationships between intersection point cloud distribution, intersection angle, and distance were found. The following section details the findings from the 2-D and 3-D Monte Carlo simulations.

Two-Dimensional and Three-Dimensional Intersection Algorithm Findings

The two-dimensional and three-dimensional intersection algorithms were characterized using Monte Carlo simulations. It was found that these two algorithms possessed similar properties, and the following observations were made.

1. The optimal angle between observers to minimize intersection mean RMS error was 90^o.
2. An optimal angle range from 75^o to 105^o only increased the mean RMS error by 2-3% of the minimum mean RMS error at 90^o for all distances and for standard location error of 2.5m and orientation error of 1.0^o.
3. Position error created an offset in the intersection mean RMS error that was independent of distance, but not the angle between observers.
4. Orientation error increased the magnitude of the intersection mean RMS error depending on distance and the angle between observers.
5. At longer distances, the intersection mean RMS error due to orientation error dominated the intersection mean RMS error due to position error.

These findings can be used as guidelines during field tests in order to obtain the optimal indirect geolocation solution.

Field Tests

In order to relate the simulations to real-world testing the proposed indirect geolocation system was tested at Wachusett Mountain in Princeton, Massachusetts. Using a United States Geological Service (USGS) marker as the truth reference with known latitude and longitude coordinates, a series of thorough and controlled tests were performed. Among the many locations and scenarios tested, the field tests can be broken down into two general categories: static and dynamic.

The results of one of the static field tests can be seen in Figure 5. In the figure, the red markers represent Observer A and the green markers represent Observer B. Both observers were located over 30 meters from the object of interest (yellow marker) and oriented towards it. As seen by the distribution of the estimated geolocation positions (blue markers), the 90 degree static test achieved a highly precise cluster of results (Standard Deviation = 0.389 meters in latitude and 0.401 meters in longitude). In terms of accuracy, however, the system was less refined. With an estimated intersection 3.8 meters from the truth marker, the presence of GPS bias ultimately created a location offset. An inaccuracy seen regularly within the field tests, the GPS accuracy behaved as a limiting factor in the indirect geolocation testing.

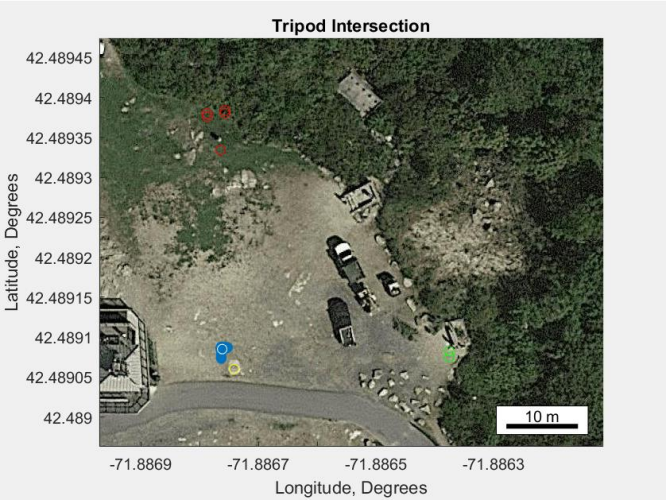

Figure 5: Static Field Test

Conclusion

Overall, this report analyzed the feasibility of utilizing smartphones for indirect geolocation applications. In order to find intersections in 2-D space, first a closed form algorithm was developed for mapping the intersection points of two coplanar vectors in 2-D space. Running simulations to characterize the performance of the system, the next task was the creation of a 3-D space intersection algorithm. Using least-squares principles, a 3-D intersection algorithm was created which provided unique, optimal solutions. These simulations produced several findings for the 2-D and 3-D algorithms. First, to minimize intersection RMS error, an angle between observers of 90^{o} was required. Second, position error acted as a bias, while the effect of orientation error on the intersection solution was dependent on the distance of the observer from the object of interest. Third, at longer distances the orientation error had more of an effect on system accuracy than the position error. With these findings in mind, field tests were performed with the Samsung Galaxy J7. Using an extended Kalman Filter (EKF), gyroscope, magnetometer, and accelerometer data were ultimately fused together to create an optimal orientation estimation.

In conclusion, the fusion of the intersection and orientation algorithms produced an indirect geolocation system. In order to test the performance of the system using smartphone sensors in a real-world environment, a series of field tests were conducted. Performed in both static and dynamic scenarios, these field tests illustrated the feasibility of smartphone-based indirect geolocation applications in the future.

Table of Contents

Chapter 1: Introduction

Indirect geolocation systems greatly aid in applications where the location of an inaccessible object of interest is desired. Imagine for example, a search and rescue mission where victims are stranded throughout a disaster area. In need of data that can help rescue teams navigate themselves to the victims quickly a system, such as indirect geolocation, can be used.

The aforementioned scenario describes the usage of an indirect geolocation system by which an observer is able to obtain the location of an object of interest while not co-located with that object. Given the severity of the scenario, such applications most likely utilize highly customized and expensive equipment. For the more ordinary consumer and less urgent situation, however, such elaborate, high-end systems are impractical [Al-Hamad and El-Sheimy, 2014]. Thus, the question follows: how practical is it to create a functioning and effective indirect geolocation system on an inexpensive, abundant, and flexible device?

Smartphones are inexpensive and abundant. According to a study performed by the Pew Research Center, 77% of American adults own a smartphone [Pew Research Center, 2018]. While models vary by manufacturer, the majority of smartphones contain a Global Navigation Satellite System (GNSS) receiver and an Inertial Measurement Unit (IMU). These two units provide the two main requirements for indirect geolocation—location and orientation. Unfortunately, smartphone sensors are not as accurate as the sensors used in customized, military-grade indirect geolocation systems. Furthermore, differing quantities of memory and processing capabilities create large margins between the computational ability of smartphones. Thus, these issues raise the question: how feasible is it to create a smartphone-based indirect geolocation system?

Previous work using smartphones for indirect geolocation has shown promising results. Specifically, smartphones have been shown to use sensor fusion to help alleviate the issues caused by the inaccuracy of their sensors. This fusion maximizes the strengths and reduces the weaknesses of each sensor to obtain an overall better measurement of location and orientation. Fusion methods are continuously improving and changing based on implementation, with one such method being the Kalman Filter [Solin *et al,* 2018]. In addition to sensor fusion, smartphones have also been shown to be capable of performing as a mobile mapping system,

subsequently utilizing the camera for photogrammetry to determine the location of an object of interest [Al-Haman and El-Sheimy, 2014]. Thus, given these examples, smartphones generally possess adequate hardware and software necessary to perform indirect geolocation.

This report explores the feasibility of using two Samsung Galaxy J7 smartphones to determine the geographic coordinates of an object of interest not co-located with either smartphone. The distance from each device to the object of interest ranges from 20 meters to several km, with increasing inaccuracy as the distance becomes larger. A prototype indirect geolocation system is presented which consists of two observers with smartphones oriented towards an object of interest. A sensor fusion algorithm, an extended Kalman Filter (EKF), is presented and implemented, which allows a more accurate estimation of each smartphone's location and orientation, as well as an error estimate of the geolocation solution. Using each smartphone location and orientation, a three-dimensional least-squares intersection algorithm is used to determine the location of the object of interest. This algorithm is validated through several zero-error models and also characterized using Monte Carlo simulations. Lastly, field tests were conducted and the entire system was evaluated using real-world data.

Chapter 2: Background

In this chapter, Section 2.1 introduces the theory of geolocation and its two variations: direct and indirect geolocation. As indirect geolocation is the focus of this report, Section 2.1.2 subsequently discusses the data and processes necessary to perform indirect geolocation. Following, Section 2.2.1 details the sensors and data necessary to perform indirect geolocation. As these necessary sensors have several sources of error, Section 2.3 introduces the concept of sensor fusion. Specifically, this section discusses the implementation of the Kalman Filter, a commonly used sensor fusion algorithm, to estimate a more accurate location and orientation. Lastly, Section 2.4 presents common methods that can be used to validate and characterize models such as a Kalman Filter and an indirect geolocation algorithm.

2.1 The Theory of Geolocation

Geolocation is the identification or estimation of the geographic location of an object of interest. As defined by this report, there are two different types of geolocation: direct geolocation and indirect geolocation. This section discusses these two variations in depth and details the calculations that are involved with each.

2.1.1 Direct Geolocation

Direct geolocation is the location estimation of an object of interest co-located with a measuring device. Represented in Figure 2-1, knowledge of the device's current location is the only necessary requirement to perform direct geolocation.

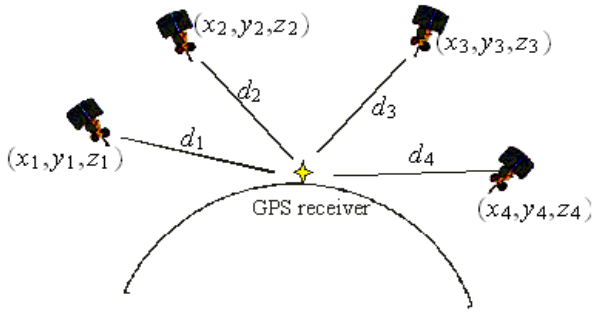

Figure 2-1: Direct Geolocation of a GPS Receiver

As seen in Figure 2-1, direct geolocation uses information transmitted by navigation satellites to pinpoint a receiver's location. There are several systems of navigation satellites that exist and they are called Global Navigation Satellite Systems (GNSS). Commonly used in applications such as Google Maps on smartphones, direct geolocation applications are abundant. More detail describing direct geolocation is provided in Section 2.2.1.

2.1.2 Indirect Geolocation

Indirect geolocation is the estimation of the location of an object of interest not co-located with any measuring device. In order to perform indirect geolocation, knowledge of one or more "truth" references of an observer's location and orientation are necessary. Ordinarily, location is obtained using a Global Navigation Satellite System (GNSS) and orientation using an inertial measurement unit (IMU). Indirect geolocation consists of finding the distance and orientation of an object from one or more observers whose locations are known. Ultimately, these three requirements – location, orientation, and distance – can be used to estimate the location of the object of interest.

A method commonly used in robotics is simultaneous location and mapping (SLAM). SLAM is a mapping algorithm theory that commonly uses vision, such as a single camera mounted to a robot, and properties of that vision, such as the parallax effect, to map the surrounding area. Despite its effectiveness though, SLAM has its own flaws. First, 3-D implementations are computationally intensive as the algorithms frequently collect a significant amount of undesired location points. Second, unless on-board processing is performed, the data link used to transfer SLAM data requires a significant bandwidth [Mendes *et al*, 2016]. Therefore, the following subsection will discuss a less computationally expensive two and three-dimensional indirect geolocation method that does not require a high amount of bandwidth, but requires at least two observers.

Indirect Geolocation Intersection in Two-Dimensional Space

Two-dimensional (2-D) indirect geolocation uses the 2-D location of two devices and each device's respective heading, or yaw, angle to ultimately calculate a final location using an intersection algorithm. Each device is pointed towards an object of interest by an observer, known as observer A and B. Both observers' "line of sight" coplanar vectors ultimately intersect

at a point in the distance. As this point is represented in a 2-D plane and only dependent on observer location and heading (yaw), the point ultimately represents a 1D intersection. Following, is the derivation for the 2-D intersection algorithm.

Observer A and observer B are pointed towards an object of interest off in the distance. Observer A's position is defined by (X_A, Y_A) and orientation is defined by Θ_A with respect to the positive x-axis. Observer B's position is defined by (X_B, Y_B) and orientation is defined by Θ_B also with respect to the positive x-axis. These six inputs (X_A, Y_A, Θ_A, X_B, Y_B, & Θ_B) are the inputs to the two-dimensional intersection algorithm.

The system and coordinate axis are shown below in Figure 2-2.

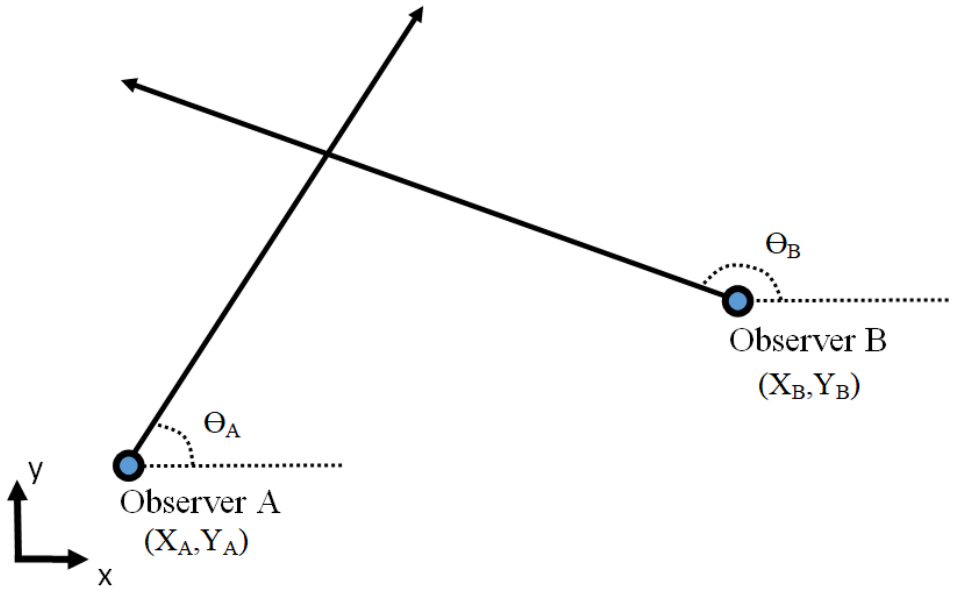

Figure 2-2: Observer A and Observer B Position and Angle Configuration

First, ensure that observer B is always on the positive x-axis in relation to observer A. If not, the label for observer A and observer B should be switched to ensure that the correct equations are used. Observer B does not have to have a greater Y coordinate than observer A as depicted in Figure 2-2. If $Y_B < Y_A$, then the angles Θ_A and Θ_B will change accordingly and the derivation remains the same.

Next, calculate the vector angle of observer B with relation to the negative x-axis, θ_B^O, at observer B, shown in Figure 2-3. This value can be found using Equation 2-1 seen below.

$$\theta_B^O = 180^\circ - \theta_B \qquad \text{(Eq. 2-1)}$$

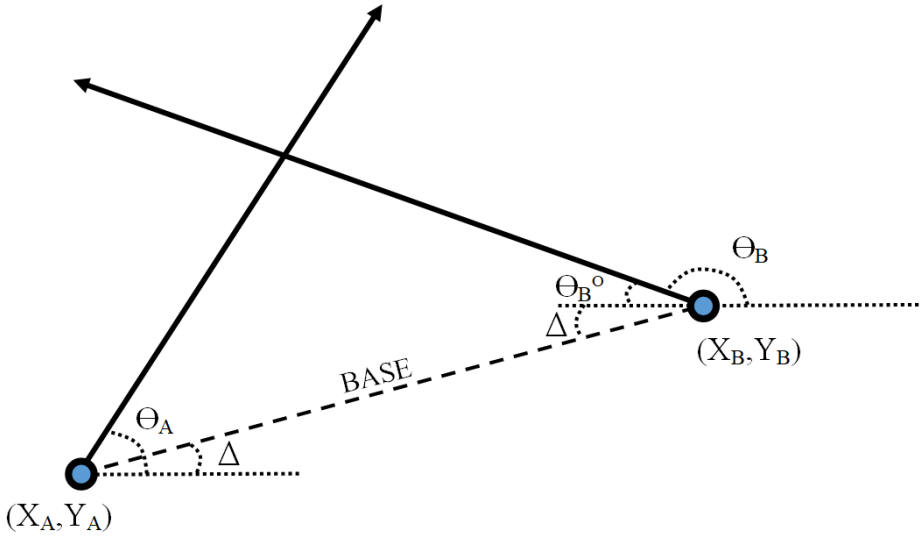

Figure 2-3: θ_B^O and Δ Angle Configuration

The BASE distance and angle Δ can be isolated in a triangle seen in Figure 2-4 below. The horizontal lines that extend from X_B and X_A are parallel, so therefore the angle Δ for both observers is equivalent due to the alternate interior angles theorem.

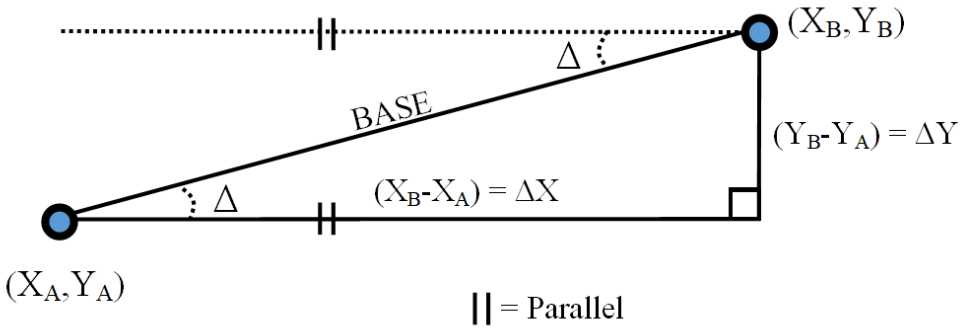

Figure 2-4: Isolated Triangle to Calculate BASE and Δ

The BASE distance between observer A and observer B can be found using Equation 2-2 seen below.

$$\text{BASE} = \sqrt{(X_B - X_A)^2 + (Y_B - Y_A)^2} \quad \text{(Eq. 2-2)}$$

The angle Δ can then be calculated with trigonometry properties using Equation 2-3 seen below, where ΔY and BASE are shown in Figure 2-3.

$$\Delta = \sin^{-1}\left(\frac{\Delta Y}{\text{BASE}}\right) \quad \text{(Eq. 2-3)}$$

The angle Δ allows for the calculation of the angles Θ_A' and Θ_B' seen in Figure 2-5 below.

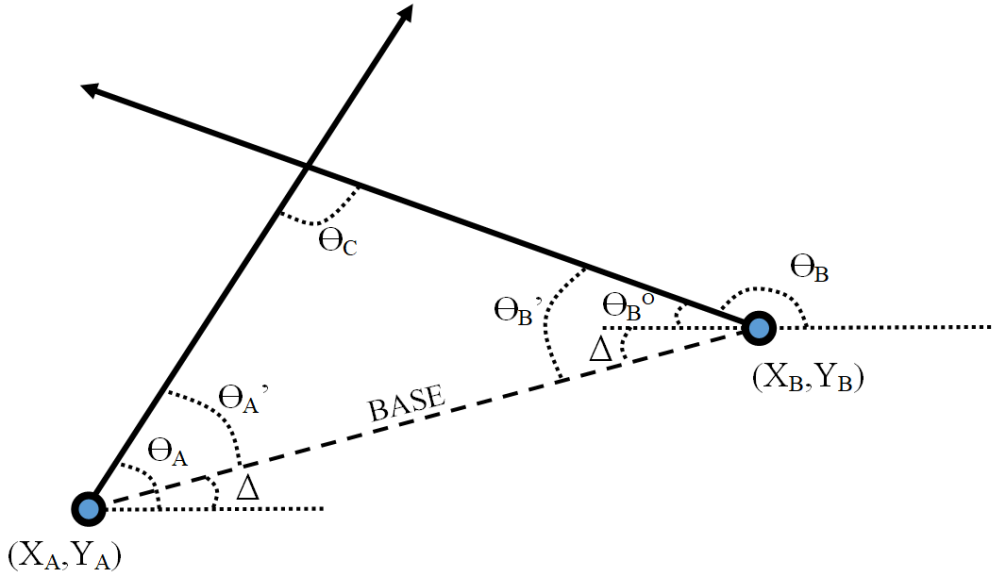

Figure 2-5: Angles Θ_A' and Θ_B' Configuration

The calculation for each of these angles is seen in Equation 2-4 and 2-5.

$$\theta'_A = \theta_A - \Delta \quad \text{(Eq. 2-4)}$$

$$\theta'_B = \theta_B + \Delta \quad \text{(Eq. 2-5)}$$

With these two known angles, the last angle, Θ_C, can be calculated using Equation 2-6, because the sum of the angles of a triangle is 180°.

$$\theta_C = 180° - \theta'_A - \theta'_B \quad \text{(Eq. 2-6)}$$

Using the law of sines, the distances from observer A and observer B to the intersection point, d_A and d_B, can be calculated for the triangle with sides d_A, d_B, and BASE. This triangle is shown in Figure 2-6 below.

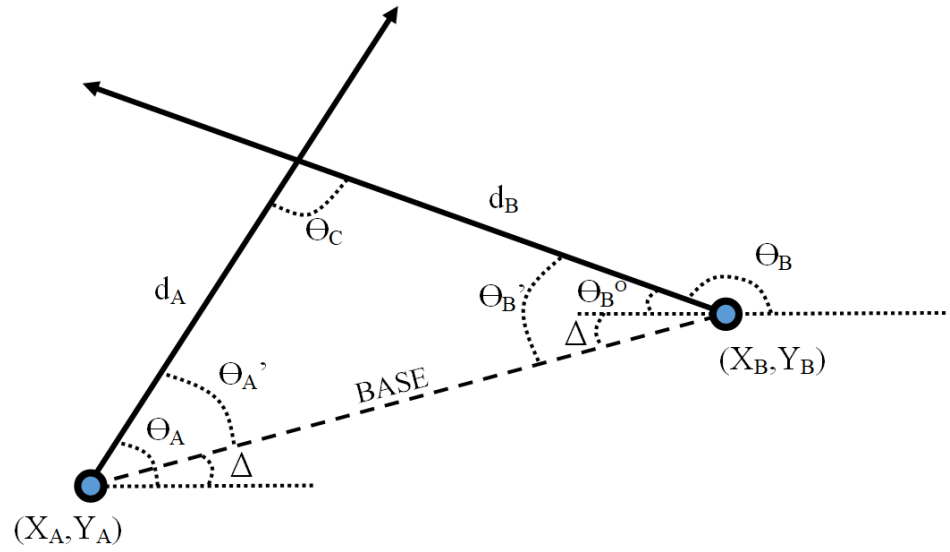

Figure 2-6: Intersection Distance d_A and d_B Configuration

The following equations can be created using The Law of Sines and they are shown in Equations 2-7.

$$\frac{BASE}{\sin(\theta_C)} = \frac{d_A}{\sin(\theta_B')} = \frac{d_B}{\sin(\theta_A')} \qquad \text{(Eq. 2-7)}$$

Equation 2-7 can then be re-arranged to obtain Equations 2-8 and 2-9 below. These equations can then be separately solved for d_A and d_B respectively using the BASE and angles solved for earlier.

$$d_A = BASE \cdot \frac{\sin(\theta_B')}{\sin(\theta_C)} \qquad \text{(Eq. 2-8)}$$

$$d_B = BASE \cdot \frac{\sin(\theta_A')}{\sin(\theta_C)} \qquad \text{(Eq. 2-9)}$$

The distance from each observer to the intersection point is the last value needed to solve for the intersection point. The intersection point (X_i, Y_i) can be calculated using the properties of a right triangle and trigonometry functions with respect to observer A. This triangle can be seen in Figure 2-7 below.

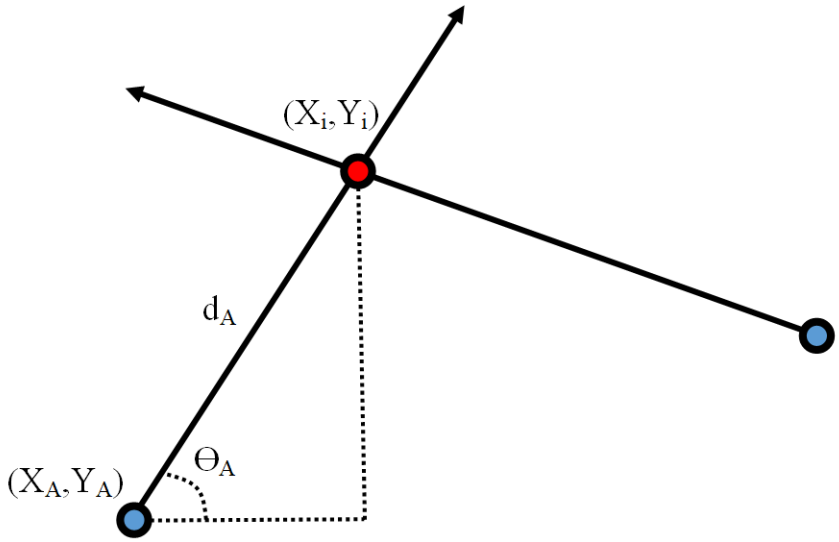

Figure 2-7: Right Triangle with Respect to Observer A and Intersection Point

Figure 2-8 below shows how the X component can be analyzed as the difference of the X intersection point and observer A X point.

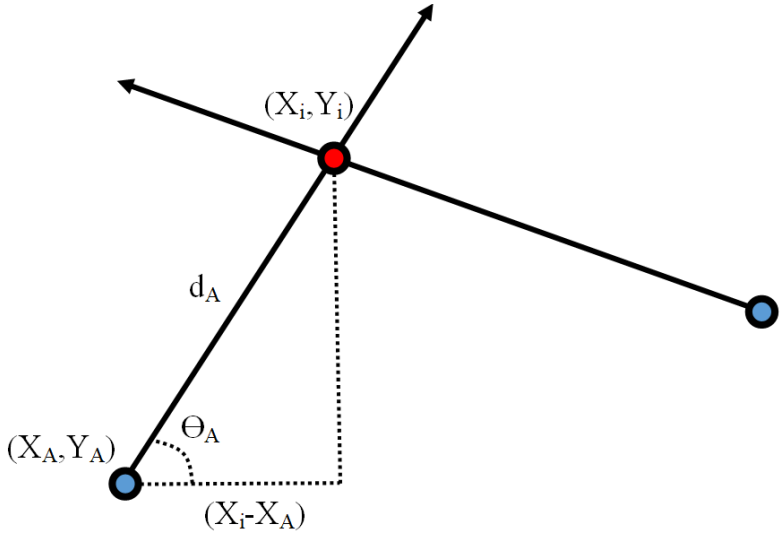

Figure 2-8: X Displacement of Observer A and Intersection Point

Equations 2-10, 2-11, and 2-12 show the process of how to find the X intersect point. Equation 2-10 uses trigonometry of the right triangle to create a ratio of X displacement to distance.

$$\cos(\theta_A) = \frac{X_i - X_A}{d_A} \quad \text{(Eq. 2-10)}$$

The equation can be re-arranged to solve for X_i, the x-coordinate of the intersection point in Equation 2-11 and 2-12.

$$(X_i - X_A) = d_A * \cos(\theta_A) \quad \text{(Eq. 2-11)}$$

$$X_i = d_A * \cos(\theta_A) + X_A \quad \text{(Eq. 2-12)}$$

Figure 2-9 below shows how the Y component can be analyzed as the difference of the Y intersection point and observer A Y point.

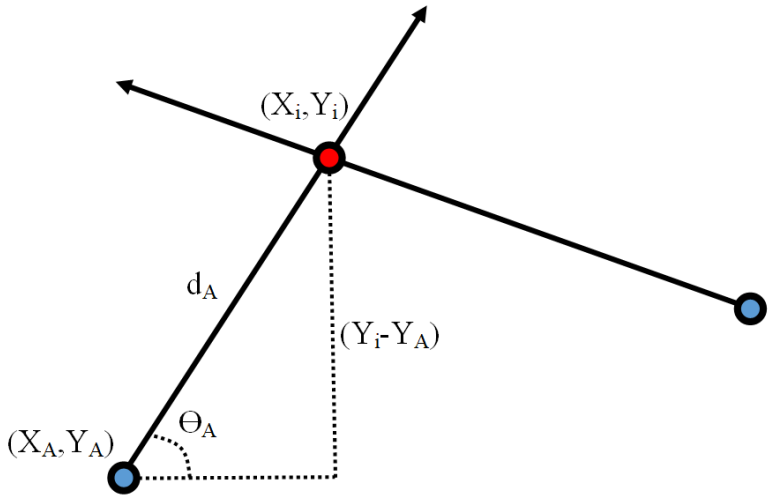

Figure 2-9: Y Displacement of Observer A and Intersection Point

Equations 2-13, 2-14, and 2-15 show the process of how to find the Y intersect point. Equation 2-13 uses trigonometry of the right triangle to create a ratio of Y displacement to distance.

$$\sin(\theta_A) = \frac{Y_i - Y_A}{d_A} \quad \text{(Eq. 2-13)}$$

The equation can be re-arranged to solve for Y_i, the y-coordinate of the intersection point, as shown through Equations 2-14 and 2-15.

$$(Y_i - Y_A) = d_A * \sin(\theta_A) \quad \text{(Eq. 2-14)}$$

$$Y_i = d_A * \sin(\theta_A) + X_A \quad \text{(Eq. 2-15)}$$

The resulting (X_i, Y_i) point is the intersection point created from the vectors of observer A and observer B. As will be discussed in the next section though, additional input data can transform this simple scenario into a three-dimensional model of an object of interest.

Indirect Geolocation Intersection in Three-Dimensional Space

Three dimensional (3-D) indirect geolocation integrates pitch into the two-dimensional method, providing the 3-axis orientation in the Earth frame. Pitch is the only additional orientation measurement needed to convert the intersection from 2-D to 3-D because in the Earth frame, roll does not have an effect on the orientation vector that is perpendicular from the device. The roll rotates around the perpendicular axis, having no effect on the orientation angle. The pitch and yaw provide the ability to locate specific points in 3-D space rather than being limited to projecting a 3-D world into a 2-D plane. To perform the 3-D calculation, the information of each observer's 3-axis position and 3-axis orientation in the Earth frame are needed. With this information, a least-squares pseudo inverse is performed to calculate the 3-D intersection. The intersection uses the least-squares method because vectors in 3-D are not likely to intersect. The least-squares method calculates the unique point that is equidistant from each vector, and at a minimum total distance from all vectors. Following is the derivation of the least-squares intersection algorithm for two observers that have a location and an orientation.

To begin the derivation of the least-squares intersection algorithm, each 3-D vector can be written with a starting point (location) and a unit vector (orientation). These define the starting point and direction of each vector. In 3-D space, the starting point, or location, of observer A is defined as $[x_A, y_A, z_A]$ meters and the starting point of observer B is defined as $[x_B, y_B, z_B]$ meters. These can be combined into the starting point matrix, P_{start}, shown in Equation 2-16.

$$P_{start} = \begin{bmatrix} x_A & y_A & z_A \\ x_B & y_B & z_B \end{bmatrix} \quad \text{(Eq. 2-16)}$$

The variables, $[n_x, n_y, n_z]$ are the components of the unit vector, *n*, in radians, that must be normalized to a magnitude of 1. Equation 2-17 below shows the calculation to normalize the unit vector. The resulting vector u is the normalized vector.

$$u = \frac{n}{sqrt(n_x^2 + n_y^2 + n_z^2)} \quad \text{(Eq. 2-17)}$$

The variables, $[u_{xA}, u_{yA}, u_{zA}]$ are the components of the normalized unit vector, u, whose magnitude is equal to 1 and describe the orientation of observer A. For observer B, $[u_{xB}, u_{yB}, u_{zB}]$ similarly describe the orientation of observer B.

$$u = \begin{bmatrix} u_{xA} & u_{yA} & u_{zA} \\ u_{xB} & u_{yB} & u_{zB} \end{bmatrix} \quad \text{(Eq. 2-18)}$$

The goal of a least-squares algorithm is to find the point, which minimizes the sum of squared distances between the point and a set of lines, to compute an approximate intersection. To minimize the distance between a line and potential intersection point, this distance must be defined in an equation. The distance, *d*, from a point on a line, *a*, with direction vector, *u*, to a point *p* can be given by Equation 2-19, where *I*, is the identity matrix.

$$d = (p - a)^T (I - uu^T)(p - a) \quad \text{(Eq. 2-19)}$$

Equation 2-19 can then be minimized with respect to point *p* to produce the least-squares solution to the intersection of lines. The sum of squared distances, D_j, from the point a_j on line with unit vector u_j to the point *p* is defined in Equation 2-20 below, where *j* is the integer

identifier for each line and *K* is the number of total lines. For simplicity and this project's application, K is equal to 2.

$$D_j = D(p; A, u) = \sum_{j=1}^{K} D(p; a_j, u_j) = \sum_{i=1}^{K} (p - a_j)^T (I - u_j u_j^T)(p - a_j) \qquad \text{(Eq. 2-20)}$$

The objective is to find the point *p* that minimizes the distance *D* equation above, which is shown in Equation 2-21.

$$\hat{p} = argmin_p \, D(p; a_i, u_i) \qquad \text{(Eq. 2-21)}$$

This equation consists of the variable, or point, *p* as a quadratic. By taking the derivative with respect to *p*, Equation 2-20 can be reduced to a linear system of equations, which is shown in Equation 2-22.

$$\frac{\partial D}{\partial p} \sum_{i}^{K} D_j^2 = \frac{\partial D}{\partial p} \sum_{j}^{K} [\,(p - a_j)^T * (p - a_j) - [(p - a_j^T) * u_j]^2] \qquad \text{(Eq. 2-22)}$$

The partial derivative of this equation can be seen in Equation 2-23 below.

$$\frac{\partial D}{\partial p} = \sum_{j}^{K} [\,2 * (p - a_j) - 2 * [(p - a_j)^T * u_j] * u_j] = 0 \qquad \text{(Eq. 2-23)}$$

Equation 2-23 can be rearranged to isolate *p* and pull it out of the summation, using the derivations seen in the LS Line Intersect [Tan, 2015]. The resulting equation can be seen in Equation 2-24 below.

$$\sum_{j}^{K} (p - a_j) = \sum_{j}^{K} [u_j * u_j^T] * (p - a_j) \qquad \text{(Eq. 2-24)}$$

With this step, a linear system of equations is produced. Let,

$$R = \sum_{j}^{K} \left[u_j * u_j^T - I \right] \quad \text{(Eq. 2-25)}$$

$$q = \sum_{j}^{K} [u_j * u_j^T - I] * a_j \quad \text{(Eq. 2-26)}$$

With the variables defined in Equations 2-25 and 2-26, Equation 2-24 can be rewritten as

$$Rp = q \quad \text{(Eq. 2-27)}$$

To solve for p, the system can be solved directly by dividing by R, or taking the inverse, seen in Equation 2-28 below.

$$\hat{p} = R^{-1} q \quad \text{(Eq. 2-28)}$$

The generalized inverse R^{-1} has guaranteed existence, but not uniqueness. So, the Moore-Penrose pseudo-inverse can be used to guarantee uniqueness. When R is nonsingular, the pseudo-inverse of R is equivalent to the inverse. Otherwise, the pseudo-inverse, $R^{\dagger}$, computes the least-squared unique solution that is minimum distance from the point $\hat{p}$ to every intersecting line [Courrieu, 2008]. The resulting equation can be seen in Equation 2-29 below.

$$\hat{p} = R^{\dagger} q \quad \text{(Eq. 2-29)}$$

To find the matrix *R* as defined in Equation 2-25, the unit vectors, u_i, and identity matrix, I, are needed. This matrix calculation can be seen in Equation 2-30 below.

$$R = \sum_{j}^{K} [u_j^T * u_j - I] = \begin{bmatrix} R_{xx} & R_{xy} & R_{xz} \\ R_{yx} & R_{yy} & R_{yz} \\ R_{zx} & R_{zy} & R_{zz} \end{bmatrix} \quad \text{(Eq. 2-30)}$$

To find the matrix q as defined in Equation 2-26, the starting points, unit vectors, and identity matrix are needed. This matrix calculation can be seen in Equation 2-31 below.

$$q = \sum_{j}^{K} [u_j * u_j^T - I] * a_j = \begin{bmatrix} q_{xx} \\ q_{yx} \\ q_{zx} \end{bmatrix} \quad \text{(Eq. 2-31)}$$

The intersection point, p, can then be calculated by finding the Moore-Penrose inverse of R and q. This intersection point calculation can be seen in Equation 2-32 below.

$$p = R^{\dagger} q = \begin{bmatrix} R_{xx} & R_{xy} & R_{xz} \\ R_{yx} & R_{yy} & R_{yz} \\ R_{zx} & R_{zy} & R_{zz} \end{bmatrix}^{\dagger} \begin{bmatrix} q_{xx} \\ q_{yx} \\ q_{zx} \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \quad \text{(Eq. 2-32)}$$

The resulting point p is the least squares intersection point of the vectors.

2.2 Sensors Required For Indirect Geolocation

As presented in Section 2.1.2, indirect geolocation systems require the location and orientation of multiple reference points, or observers. Known as *pose*, an observer's location and orientation can be determined using many different sensor systems such as cameras and ultra-wideband systems [Kok *et al*, 2017]. For simplicity, however, this report focuses on the Global Navigation Satellite System (GNSS) receiver and the inertial measurement unit (IMU) to perform indirect geolocation [Gikas and Perakis, 2016].

2.2.1 Sensor Physics and Operation

Global Navigation Satellite System

A Global Navigation Satellite System (GNSS) consists of several constellations of satellites in space continuously transmitting data. Small chip receivers located on Earth can receive said data, and ultimately use those data to calculate its position on Earth. While orbiting the globe, each satellite specifically transmits information describing itself and the whole constellation of satellites. Providing data such as its unique ID, almanac, ephemeris, and time of transmission, the satellites help GNSS receivers perform direct geolocation. Ultimately, the datum that enables GNSS receivers to determine their location is the time of transmission.

The receiver compares the time of transmission to its internal time and calculates the total transmission time from satellite to receiver. In this calculation there are many variables taken into consideration such as the effects of the atmosphere on the signal transit time. With the total transmission time, the GNSS receiver can then determine the distance to each satellite because all radio signals travel in a vacuum at the speed of light(300,000 km per second). Equation 2-33 describes the relationship between the speed of light (c), distance to each satellite (d), and transmission time from satellite to receiver (t).

$$d = c \cdot t \qquad \text{(Eq. 2-33)}$$

The calculated distance is a pseudo-range because the receiver only knows where the satellite is and how far away it is from the satellite. This limited knowledge subsequently creates a sphere around the satellite, indicating possible locations for the GNSS receiver. When a receiver calculates four or more pseudo-ranges from different satellites, the receiver can then perform a process known as trilateration to determine its location on the Earth. Shown in Figure 2-10, the trilateration method is used to determine a GNSS receiver's location by finding an overlapping region of the potential spherical locations. [Groves, 2013; Djuknic & Richton, 2001]

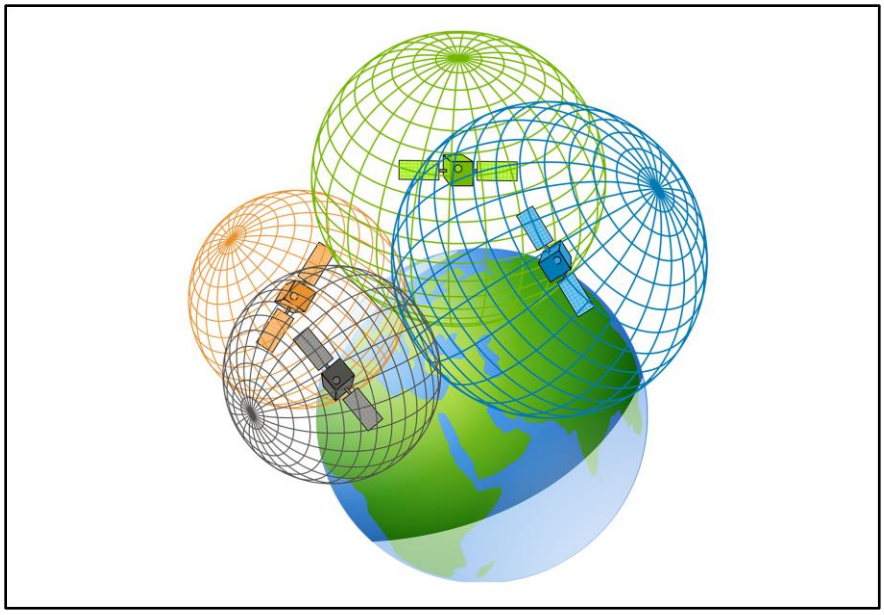

Figure 2-10: GPS Trilateration with Uncertainty [GISGeography]

An example of trilateration with four different satellites is shown in Figure 2-10 above. Alone, no satellite provides sufficient data for a location solution. When a GNSS receiver

calculates the distance from each satellite, the receiver is not able to determine its orientation with respect to each satellite and therefore could be anywhere on the surface of each sphere. As the GNSS receiver collects more satellite data from different satellites, however, an overlapping area is found where the surface of these spheres all intersect and thus an estimated location is determined. Ultimately, with the data provided by four satellites, a GNSS receiver can estimate its location anywhere on Earth.

Unfortunately, due to sensor inaccuracies, there is uncertainty throughout the system. If the receiver's distance calculation for each satellite has an error of a few meters, that error directly affects the receiver's location calculation. Specifically, the United States government has acknowledged this uncertainty and supports findings stating that the GPS error of smartphones is ± 4.9 meters [van Diggelen & Enge, 2015].

Inertial Measurement Unit

Often paired with a GNSS receiver is an Inertial Measurement Unit (IMU). Although there are different variations, an IMU frequently possesses 9 degrees of freedom (9DOF) from three sensors each in a 3-axis configuration. Consisting of a 3-axis accelerometer, 3-axis gyroscope, and 3-axis magnetometer, the IMU can help create a location and orientation estimate of a device if given the proper initial conditions.

In general, there are two main categories of IMU designs: gimballed and microelectromechanical systems (MEMS). Gimballed systems utilize gimbals to isolate the IMU from the movement of the attached system. Largely mechanical, these systems offer simplicity and accuracy at the cost of size and weight [Li *et al*, 2013]. On the contrary, MEMS IMUs take an intrinsically mechanical action, such as rotation, and generate a corresponding electrical signal. Often packaged into systems a few millimeters in size, MEMS technology allows for compact sensor implementations, such as those within smartphones. Due to their implementation with smartphones, the focus of this report will be on MEMS systems. [Barret, 2014]

Accelerometers

MEMS accelerometers have two main categories: mechanical, but manufactured as MEMS, and vibrating element. Mechanically based accelerometers measure the change in position of a known mass to determine the forces, and thus, acceleration on that mass.

Meanwhile, vibrating element accelerometers, such as surface acoustic wave (SAW) accelerometers, utilize a lever arm with a mass on it. When the mass has a force applied to it, the lever arm's resonant frequency changes. Subsequently, by measuring the change in frequency, the force applied to that mass can be determined. This implementation is considered more accurate and more appropriate for an inertial navigation system [Maenaka, 2008]. An example implementation is shown below in Figure 2-11 [Woodman, 2007].

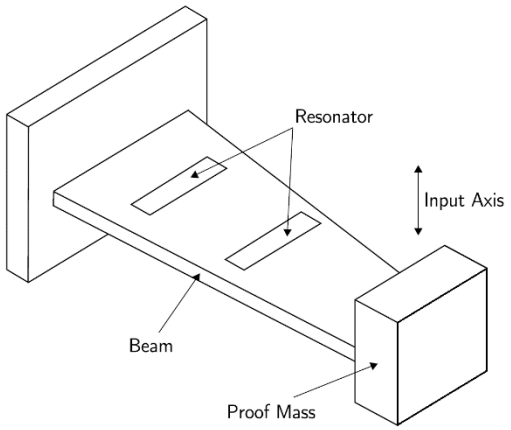

Figure 2-11: MEMS Surface Acoustic Wave Accelerometer Diagram [Woodman, 2007]

Gyroscope

A MEMS gyroscope, often called a Coriolis vibratory gyroscope (CVG), contains a vibrating mass that allows for the measurement of rotational acceleration, or rate of direction change, to be found. This change can be found using Newton's first law, the conservation of momentum: vibrating objects continue to oscillate in the same plane; any deviation from the plane can be used to detect a change in direction. When undergoing rotation, the gyroscope measures the Coriolis Effect, which states that a mass moving within a rotating system experiences an external force called the Coriolis force. This force is perpendicular to the direction of motion and to the axis of rotation. An example of a one degree of freedom MEMS gyroscope is seen in Figure 2-12 below. [Vu *et al*, 2011]

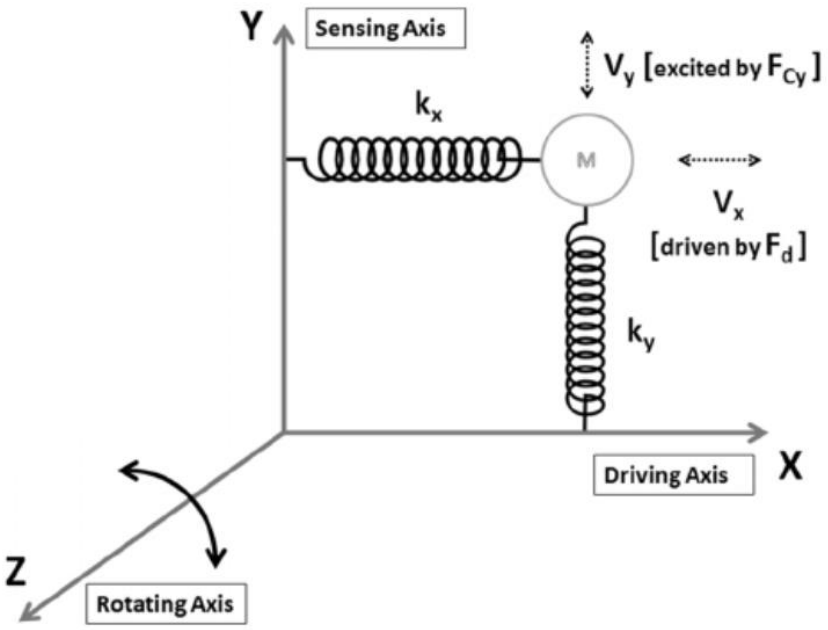

Figure 2-12: Coriolis Vibratory Gyroscope Diagram [Vu *et al*, 2011]

The one degree of freedom describes the independent measurement this device will be measuring. Since the z-axis is the rotating axis and measurement to capture, this axis acts as the degree of freedom. The driving axis attached to the x-axis is used to vibrate the mass in one direction. When an external force perpendicular to the x and y plane occurs, oscillations are produced in the y-axis direction through the energy transfer of the Coriolis force. By measuring this force, the rate of rotation around the z-axis can be found [Vu *et al*, 2011]. One IMU typically consists of multiple gyroscopes or multiple-axis gyroscopes to capture a movement with 3 degrees of freedom.

Magnetometer

Most MEMS magnetometers utilize the Lorentz force, which is the force exerted by an electromagnetic field on a moving charge, or current. Figure 2-13 shows a MEMS magnetometer system designed by M. Pierre Courtois at the Université Catholique de Louvain. [Said *et al.*, 2013]

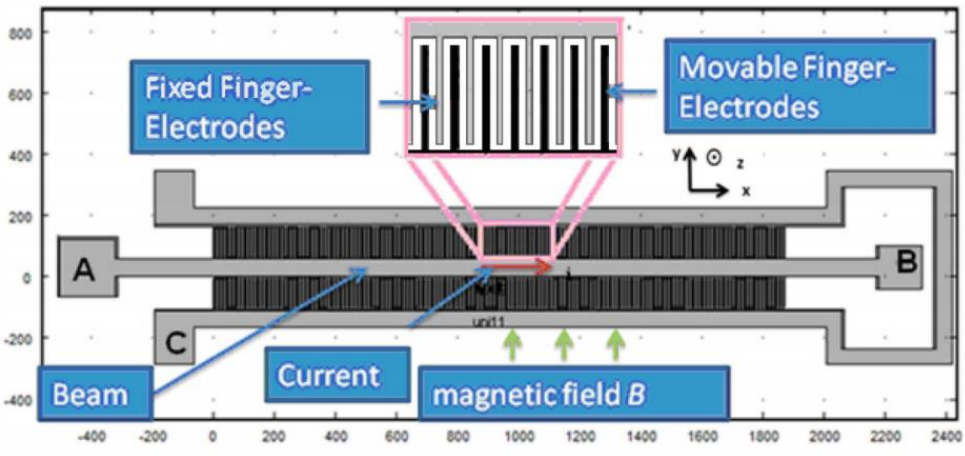

Figure 2-13: MEMS Magnetometer Diagram [Said *et al.*, 2013]

When a magnetic field is applied across the central beam, the beam moves proportionally in the z-axis. This movement produces a change in capacitance across the finger electrodes on both sides of the beam. The strength of the magnetic field is a function of this change in capacitance [Said *et al.*, 2013].

Determining Orientation with Inertial Measurement Unit Sensors

Each sensor within an IMU—accelerometer, gyroscope, and magnetometer—is typically seen in a 3-axis configuration, taking measurements about the device's x, y, and z direction. Using these measurements, the IMU's orientation can be found, most often in terms of roll, pitch, and yaw [Luinge *et al.*, 2005]. Figure 2-14 below shows how roll, pitch, and yaw are typically aligned to each of the three axes.

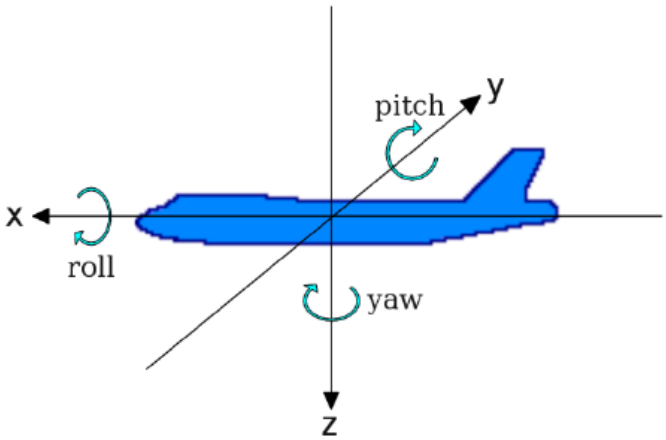

Figure 2-14: Orientation in the XYZ Plane as Roll, Pitch, and Yaw [FitzGerald, 2015]

Each sensor within the IMU provides data that can be used to calculate the device's pitch, roll, and yaw. A gyroscope measures the device's angular rate. Integrated over time, this measurement can determine the device's angle with respect to an initial orientation [Groves, 2013]. An accelerometer can be used to measure translational acceleration. This measurement can be integrated once with respect to time to produce a 3-axis change in velocity, or twice to produce a 3-axis change in position. As the 3-axis accelerometer also measures the constant force of gravity, it can be also used to determine the direction of Earth's gravity vector. By knowing the direction of Earth's gravity vector, or which way is down, the device's roll and pitch can be calculated [Titterton *et al.*, 2004]. A magnetometer can be used to measure the device's relative magnetic field strength and can be calibrated to determine the direction of magnetic North. The direction of North can be used to determine the device's yaw compared to North, also commonly referred to as azimuth or heading [Groves, 2013]. Figure 2-15 below summarizes the measurement and orientation contribution of each IMU component, with Δ meaning, "change in".

IMU Component	Measurement Uses (3 Axes)	Orientation Contribution(s)
Gyroscope	Δ Angle	Roll, Pitch and Yaw
Accelerometer	Acceleration Δ Velocity Δ Position	Roll and Pitch
Magnetometer	Δ Angle from North	Yaw

Figure 2-15: Measurement and Orientation Contribution of each IMU Component

2.2.2 Strengths and Weaknesses of Sensors

Ideally, a GPS and gyroscope paired with an accurate 3-D initial orientation would be the only sensors necessary to determine location and orientation. Unfortunately, sensors with high accuracy and precision are expensive and not seen in any current smartphone [Jin *et al.*, 2011]. Consequently, current smartphone sensors will frequently produce inaccurate device locations and orientations without any prior filtering of the measurement data.

Global Navigation Satellite System

Although a GNSS receiver can determine an estimated location, it also suffers from multiple issues. GNSS receivers typically collect information from satellites at a much slower rate than IMUs. Given the sampling rate inconsistencies, the GNSS can produce gaps in

coverage between samples. In addition to the coverage gaps, the low power satellite signals can be easily disrupted by obstructions, resulting in an inaccurate location or no location estimate at all. These obstructions can include tunnels, metal walled buildings, or almost anything that prevents the receiver from having a clear view of the sky [Groves, 2013].

Inertial Measurement Unit

IMUs suffer from several problems as well. For gyroscopes, their readings often contain significant drift. In order to calculate orientation from angular rate, the gyroscope measurement must be integrated. As noise and inaccuracies are commonly present within the measurement, the integration of these errors will compound and ultimately create drift. The level of drift varies depending on the gyroscope; however, for all systems it grows with time. Thus, due to drift, gyroscopes are accurate in determining angular change with short, jerky movements but become inaccurate when determining orientation over a long period of time [Madgwick, 2010].

While gyroscopes are best in short, sporadic movements, accelerometer readings are best under static or constant acceleration conditions. Different from drift, accelerometers are subject to high levels of noise that can produce inaccurate approximations of roll and pitch. The high amount of noise disrupts the short, jerky measurements of the gravity vector and an accurate roll and pitch orientation cannot be determined. Contrary to the gyroscope, the accelerometer does not deviate over time because no integration is required to produce roll and pitch estimates [Sabatini, 2006].

Last for the IMU, magnetometer readings are subject to high levels of noise and magnetic interference. Magnetic interference is a limitation that can affect accurate yaw orientation in both short-term and long-term measurements. The noise affects accurate yaw orientation in the short term but does not deviate over time, so measurements in the long term will be accurate if no interference is present because there is no integration required [Sabatini, 2011].

High Performance IMU Compared to Consumer Grade

A high performance IMU should perform considerably better than a common smartphone IMU. Figure 2-16 shows the specifications for VectorNav's high performance tactical series IMU. This IMU is likely used for industrial or military applications and is not targeted towards ordinary consumers.

Smartphones contain less accurate IMU's, but smartphones are not intended to be extremely accurate due to the broad range of uses and cost; Figure 2-17 shows specifications for smartphone accelerometers and gyroscopes in the market today and for the magnetometer in the Samsung Galaxy S4. The Samsung Galaxy S4 is chosen because the specific part numbers used within the smartphone are publicly available and there are no market specifications on magnetometers. The VectorNav IMU has a clear performance advantage over those commonly found in smartphones, specifically in the bias and noise specifications for the accelerometer and gyroscope.

IMU	Accelerometers	Gyroscopes	Magnetometers
Range	± 15g	± 490 °/s	± 2.5 Gauss
In-Run Bias Stability	< 10 µg	< 1 °/hr	-
Noise Density	0.040 mg/√Hz	3.24 °/hr /√Hz	140 µGauss/√Hz

Figure 2-16: IMU Specifications for VectorNav Tactical Series [VectorNav, 2016]

IMU	Accelerometers	Gyroscopes	Magnetometers
Range	-	-	± 49 Gauss
In-Run Bias Stability	< 14.3 - 25.3 mg	< 21.96 - 33.84 °/hr	-
Noise Density	0.25 - 2.2 mg/√Hz	36 - 216 °/hr /√Hz	-

Figure 2-17: IMU Specification Averages for Various Smartphones [Kos *et al*, 2016; AsahiKASEI, 2013]

In order to produce an accurate three-dimensional orientation using an IMU, the three inertial sensors must be used together in a method called sensor fusion. Sensor fusion algorithms take advantage of the complementary strengths and weaknesses of the three sensors to produce a more accurate orientation. The gyroscope captures quick, high frequency movements with which the accelerometer and magnetometer struggle. The accelerometer and magnetometer capture static or low frequency movements, which can prevent drift errors from the gyroscope. This fusion results in a more accurate roll, pitch, and yaw orientation with less effect from the detrimental sensor errors that are native to hardware solutions. [Sabatini, 2011]

2.3 Sensor Fusion

Systems dependent on a single sensor commonly suffer from sensor deprivation, limited coverage, imprecision, and uncertainties. Consequently, in an attempt to mitigate the influences of these undesired effects, sensors often undergo a process known as sensor fusion: a process in which individual sensor strengths are combined in order to mitigate their individual weaknesses. Leveraging measurement redundancies and the laws of probability to increase system robustness and confidence, sensor fusion creates more optimal system estimates. [Elmenreich, 2002]

Many different algorithms perform variations of sensor fusion. For simplicity, this report focuses on the Kalman Filter (KF). Specifically, this report discusses two variations of the KF: the Linear Kalman Filter (LKF) and the Extended Kalman Filter (EKF).

2.3.1 The Kalman Filter[1]

Prior to discussing the equations and principles used in the Kalman Filter (KF), it is important to get a better understanding of the filter's purpose. To do so, a hypothetical scenario describing the purpose and application of the Kalman Filter is briefly described. Dynamic systems, such as the motion of a car for example, may require the knowledge of said vehicle's position and velocity. Referred to as sub-states in this report, these desired system variables are arranged into a column vector as seen in Equation 2-34. In the equation, x_k, is a 2-by-1 column vector representing the state of the system. Its two specific sub-states, p_k and v_k, represent the unknown position and unknown velocity, respectively.

$$x_k = \begin{bmatrix} p_k \\ v_k \end{bmatrix} \qquad \text{(Eq. 2-34)}$$

To estimate the car's current state (i.e., the car's current position and velocity), many algorithms, such as the complementary filter, rely exclusively on sensor data. Analyzing data from the GPS receiver and speedometer, which measure position and velocity respectively, the algorithm produces an estimation of the vehicle's current position and velocity. Theoretically, such measurement synthesis algorithms like the complementary filter provide accurate state

[1] Information describing the principles and equations of the Kalman Filter came from the following sources: [Feng *et al*, 2017], [Groves, 2013].

estimates. Realistically, however, sensors are prone to inherent biases, scaling errors, axial misalignments, noise, and other undesirable features; all of which corrupt the legitimacy of their readings. Consequently, filters, such as the complementary filter, are heavily reliant on data which can be potentially corruptible.

In an attempt to increase the robustness and accuracy of its state estimations, the Kalman Filter uses two different models to describe the dynamics of the system. Similar to the complementary filter, one of the models used by the Kalman Filter is the measurement model. In order to supplement the measurement model, however, the KF also uses a process model. Designed to represent the theoretical propagation of the state (i.e., car's position and velocity) with respect to time, the process model uses equations that represent the dynamics of the state. Thus, leveraging all possible knowns, the KF is capable of calculating an optimal state estimate. [Kok *et. al*, 2017]

The fundamental principles of the KF are modeled in Equations 2-35 to 2-40. The first stage of the KF, the prediction of the state x_k^-, is shown in Equation 2-35. Specifically, this stage is known as the *a priori* phase.

$$x_k^- = A_{k-1}\hat{x}_{k-1} \qquad \text{(Eq. 2-35)}$$

In this first step, as modeled by Equation 2-35, the KF creates an initial guess of the current state. Based on the previous timestep's optimized state estimate, $\hat{x}_{k-1}$, the filter propagates the previous estimate with an N-by-N transition matrix, A_{k-1} (where N represents the number of sub-states). Designed to represent the theoretical dynamics of the state, the conventional transition matrix contains a linear system of equations describing the system's theoretical behavior. Ultimately, by propagating the best previous estimate with the appropriate dynamic equations, a new guess of the current state can be made.

As each iteration of the KF is an estimation, there is a corresponding uncertainty with each approximation. Modeled as an N-by-N covariance matrix, this uncertainty parameter represents external uncertainties, system noise, and all other parameters that might alter the legitimacy of the estimate. Specifically, the covariance matrix helps determine the variance of

each sub-state and the correlation between all sub-states. Further explanation of the *a priori* covariance matrix can be seen in Equation 2-36.

$$P_k^- = A_{k-1}P_{k-1}A_{k-1}^T + Q_{k-1} \quad \text{(Eq. 2-36)}$$

The *a priori* covariance matrix estimate, P_k^-, is a function of the transition matrix from Equation 2-35, the previous timestep's optimized covariance, P_{k-1}, and a process noise parameter, Q_{k-1}. Just as the state vector, $\hat{x}_{k-1}$, undergoes its own propagation in Equation 2-35, its corresponding covariance matrix, P_{k-1}, must undergo its own propagation. As seen by the pre and post-multiplication of P_{k-1} by the transition matrix, this propagation shows the change in the state covariance matrix with respect to time. Due to the presence of Q_{k-1}, however, the *a priori* covariance estimate must also model an additional parameter, the process noise. Used to represent the increasing uncertainty due to the absence of data in between discrete-time sampling periods, the addition of Q_{k-1} ensures that the *a priori* covariance estimate is either equivalent or greater than the previous timestep's optimized value $\hat{x}_{k-1}$.

Having used Equations 2-35 and 2-36 to create initial estimates for both the state vector and its covariance matrix, the process model portion of the Linear Kalman Filter is complete. For maximum accuracy, however, the Kalman Filter now implements its measurement model. While the process model from Equations 2-35 and 2-36 was based on theory such as the laws of kinematics, the measurement model is based on actual sensor measurements. A means of providing tangible data and a different perspective to the state estimate, the measurement model helps update and optimize the *a priori* state and covariance estimates.

The implementation of the measurement model is represented by Equation 2-37. Using the *a priori* state estimate previously generated by Equation 2-35, the measurement model attempts to predict the values that the sensors will measure. As the monitored sub-state parameters may differ from the parameters measured by the sensors, the measurement model also uses a measurement relationship matrix, H_k. A means of connecting the state estimates to the measurement data, this linear matrix helps conjoin the two different models.

$$z_k = H_k x_k^- \quad \text{(Eq. 2-37)}$$

With the measurement model implemented, Equation 2-38 can be performed to compute the Kalman gain. Acting as the weighted component to the filter, the Kalman gain updates recursively to determine which of the two models is more trustworthy.

As seen in Equation 2-38, the Kalman gain is a ratio of covariance matrices represented by P_k^- and R_k. P_k^-, as described by Equation 2-36, represents the *a priori* covariance value for the system. Representative of the process model's current estimate of uncertainty, this value indicates the trustworthiness of the system prior to the input of the measurement model. In order to represent the influence of the measurement model, P_k^- is propagated by the measurement matrix H_k. Given the imperfect nature of sensors, however, an additional term is added as a covariance matrix to embody sensor error. This term, R_k, is an N-by-N matrix that embodies the variances of each sensor. For simplicity, a hypothetical 3-by-3 measurement noise covariance matrix, R_k, is provided with Equation 2-38. Representative of the covariance matrix of a tri-axial sensor, this matrix identifies the variance of each sensor axis, while also indicating zero correlation between the axes.

$$K_k = \frac{P_k^- H_k^T}{H_k P_k^- H_k^T + R_k} \qquad \text{(Eq. 2-38)}$$

where,

$$R_k = \begin{bmatrix} \sigma_{xx}^2 & 0 & 0 \\ 0 & \sigma_{yy}^2 & 0 \\ 0 & 0 & \sigma_{zz}^2 \end{bmatrix}$$

As seen by Equation 2-38, the Kalman gain is calculated as a function of uncertainty. Based entirely on the system's covariance matrices, the gain determines how much of each model (process and measurement) the system should trust to create an optimized state estimate.

Having calculated the Kalman gain, the input from the process and measurement models can be appropriately filtered. As seen in Equation 2-39, the Kalman gain value, K_k, weighs between the actual measurement values, $\hat{z}_k$, and the theoretical measurement values, $H_k x_k^-$. Known as the measurement residual, the difference between these two values is weighted by the gain and then added to the *a priori* state estimate, x_k^-, to create an optimized *a posteriori* state estimate, $\hat{x}_k$.

$$\hat{x}_k = x_k^- + K_k(\hat{z}_k - H_k x_k^-) \quad \text{(Eq. 2-39)}$$

Lastly, having optimized the system's state estimation, the system's covariance estimation must also be optimized. Using the same Kalman gain value, K_k, the measurement matrix, H_k, the *a priori* covariance matrix, P_k^-, and an N-by-N identity matrix, I, the *a posteriori* covariance matrix, P_k, can be calculated as seen in Equation 2-40.

$$P_k = (I - K_k H_k)P_k^- \quad \text{(Eq. 2-40)}$$

2.3.2 The Extended Kalman Filter[2]

The extended Kalman Filter (EKF) operates using the same principles as modeled by the conventional, or linear Kalman Filter from Equations 2-35 through 2-41. Unlike the simpler linear Kalman Filter, however, the EKF is designed to approximate nonlinear system models. Using both a process model and a measurement model to create a refined state estimation, the EKF utilizes Bayesian estimation principles.

In order to create an optimal state estimate of a nonlinear system, the EKF operates using a nonlinear state model. Additionally, the filter assumes that all process and measurement noise is zero-mean Gaussian and that the measurement noise is additive. Similar to the Kalman Filter description in Section 2.3.1, the EKF has both a prediction step and an update step. The prediction step of the EKF is modeled by Equations 2-41 and 2-42. Very similar to Equations 2-35 and 2-36 of the linear Kalman Filter, the EKF predicts its next state and covariance using a process model and its most recent state and covariance values. Unlike Equations 2-35 and 2-36, however, the process model that describes the state space of the system is nonlinear. The *a priori* estimate, shown in Equation 2-41, uses the non-linear model f_{k-1} instead of the linear system model A_{k-1}.

$$x_k^- = f_{k-1}(\hat{x}_{k-1}) \quad \text{(Eq. 2-41)}$$

[2] Information describing the principles and equations of the Extended Kalman Filter came from the following sources: [Feng *et al*, 2017], [Kok *et al*, 1997], [Groves, 2013].

As seen in Equation 2-41, the *a priori* state prediction, x_k^-, is a function of the previous time step's state estimate, $\hat{x}_{k-1}$. Then, written in function notation to express the nonlinearities of the system model f_{k-1}, Equation 2-41 has the previous state estimate undergo nonlinear propagation.

$$P_k^- = F_{k-1}P_{k-1}F_{k-1}^T + Q_{k-1} \quad \text{(Eq. 2-42)}$$

where,

$$F_{k-1} = \frac{\partial f_{k-1}(\hat{x}_{k-1}, w_{k-1})}{\partial \hat{x}_{k-1}} | w_{k-1} = 0 \quad \text{(Eq. 2-43)}$$

Above, w_{k-1} is defined as the process noise. Having created an *a priori* state estimation using a nonlinear function, the next step is the calculation of the *a priori* covariance matrix. Unlike the equivalent operation for the LKF, however, this calculation, as seen in Equation 2-42, is more complicated. In order to propagate the previous time step's covariance matrix, P_{k-1}, the nonlinear system function must be temporarily linearized. As seen in Equation 2-43, the transition matrix, F_{k-1}, is calculated by taking the Jacobian of the nonlinear state function. Thus, by linearizing the process model about the most recent time step's optimal state estimation, $\hat{x}_{k-1}$, a transition matrix similar to that of Equation 2-35 can be calculated.

Having theorized the next state and covariance values using the nonlinear process model, the system next incorporates the measurement model. Similar in principle to Equation 2-37 from the LKF, the measurement model of the EKF creates an estimate as to what the measured sensor values, z_k, will be based on the *a priori* state estimate, x_k^-. Unlike Equation 2-37 though, the measurement model, as expressed in Equation 2-44, can be nonlinear in nature. Thus, function notation is used to describe the potentially nonlinear nature of the measurement model due to measurement relationship function, h_k.

$$z_k = h_k(x_k^-) \quad \text{(Eq. 2-44)}$$

With the implementation of the measurement model, the Kalman gain can be computed. Using the same principles as those used for the LKF, the Kalman gain of the EKF is calculated as a ratio of covariances. In order to propagate the *a priori* covariance matrix, P_k^-, with the

measurement model, however, the nonlinear measurement model must be linearized. Thus, undergoing the process shown in Equation 2-45, the Jacobian of the measurement model, Equation 2-46, function is taken in order to compute the Kalman gain, K_k.

$$K_k = \frac{P_k^- H_k^T}{H_k P_k^- H_k^T + R_k} \quad \text{(Eq. 2-45)}$$

where,

$$H_k = \frac{\partial h_k(x_k^-)}{\partial x_k^-} \quad \text{(Eq. 2-46)}$$

Finally, with the weighing factor calculated, the EKF can create optimized state and covariance estimates. Using the same equation for the *a posteriori* state estimate as seen with the LKF (Equation 2-47), the EKF weighs the difference between the actual measurement values, $\hat{z}_k$, and their theoretical values, $h_k(x_k^-)$. Indicating the level of trust the system has in the process and measurement models, this weighted residual is then added to the *a priori* state estimate, x_k^-, to ultimately compute an updated state estimate, $\hat{x}_k$.

$$\hat{x}_k = x_k^- + K_k(\hat{z}_k - h_k(x_k^-)) \quad \text{(Eq. 2-47)}$$

Following the same principles as outlined in the LKF, execution of Equation 2-48 computes the updated *a posteriori* system covariance matrix.

$$P_k = (I - K_k H_k) P_k^- \quad \text{(Eq. 2-48)}$$

2.4 Model Validation and Evaluation

This report discusses the design of intersection algorithms and a Kalman Filter. In practice, it is important to validate and characterize the performance of algorithms and filters before they are implemented into the real-world. This section first presents the zero-error model, which is used to prove that an algorithm performs as expected when given a scenario with no errors. This proof is known as validation. Second, the Monte Carlo simulation is introduced, which introduces error distributions to systems, so that the system performance can be characterized.

2.4.1 Zero-Error Model for Validation

A zero-error model is used to characterize the performance of an algorithm when no errors are introduced into the system. This model is important because ultimately the system will need to be characterized when errors are introduced. When that error propagates throughout the system, the evaluator has to be confident that the output error can be attributed solely to the input error and not a mistake in the algorithm. As such, a zero-error model removes variables and ambiguity from the equation. An extremely simple case is the algorithm for determining a point *y* on a line, given an input *x* and characteristics of that line, shown in Equation 2-49.

$$y = mx + b \quad \text{(Eq. 2-49)}$$

The output can be defined as *y*. Meanwhile, the input is *x*, with *m* and *b* acting as constants that characterize the line. To create a zero-error model, first these variables must be defined, for example:

$m = 1$

$b = 1$

$x = 5$

Analytically, the output *y* is determined by creating an analytical model and algebraically solving for the point on the line. First, a triangle with the expected result is shown in the top right corner of Figure 2-18. The result is shown as the point *(x3, y3)*. The y-intercept of the line is shown as *(x1, y1)* and the point to form a right triangle between the two points is shown as *(x2, y2)*.

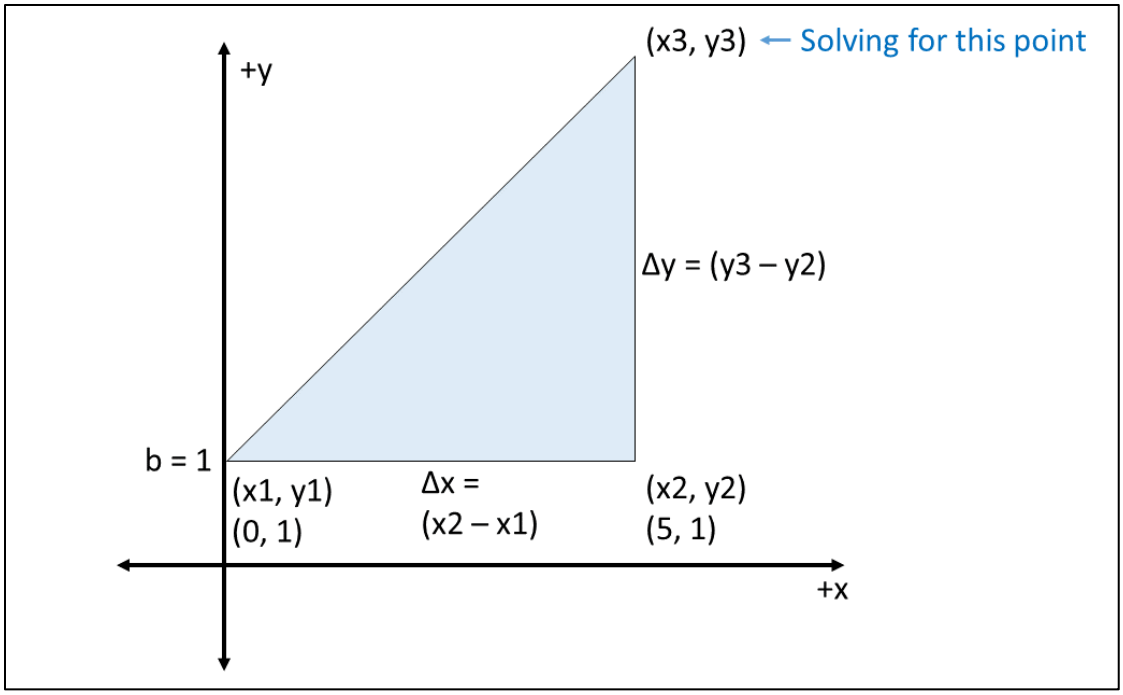

Figure 2-18: Line Analytical Model

$$m = \frac{\Delta y}{\Delta x} \qquad \text{(Eq. 2-50)}$$

When Equation 2-50, the slope equation, is solved for $m = 1$, it is shown that $\Delta x = \Delta y$. Consequently, this means the triangle shown in Figure 2-18 is an isosceles right triangle, where the side parallel to the x axis is equal in length to the side parallel to the y axis. Since Δx is equal to five, Δy must also be equal to five due to this property. Then, five is added to the initial *x1* value of 0, and to the initial *y1* value of 1 to get the result of a point at (5, 6). Now, that result is compared to the algorithm implemented into MATLAB and Figure 2-19 shows the algorithm solution and it is shown to be (5, 6) as well.

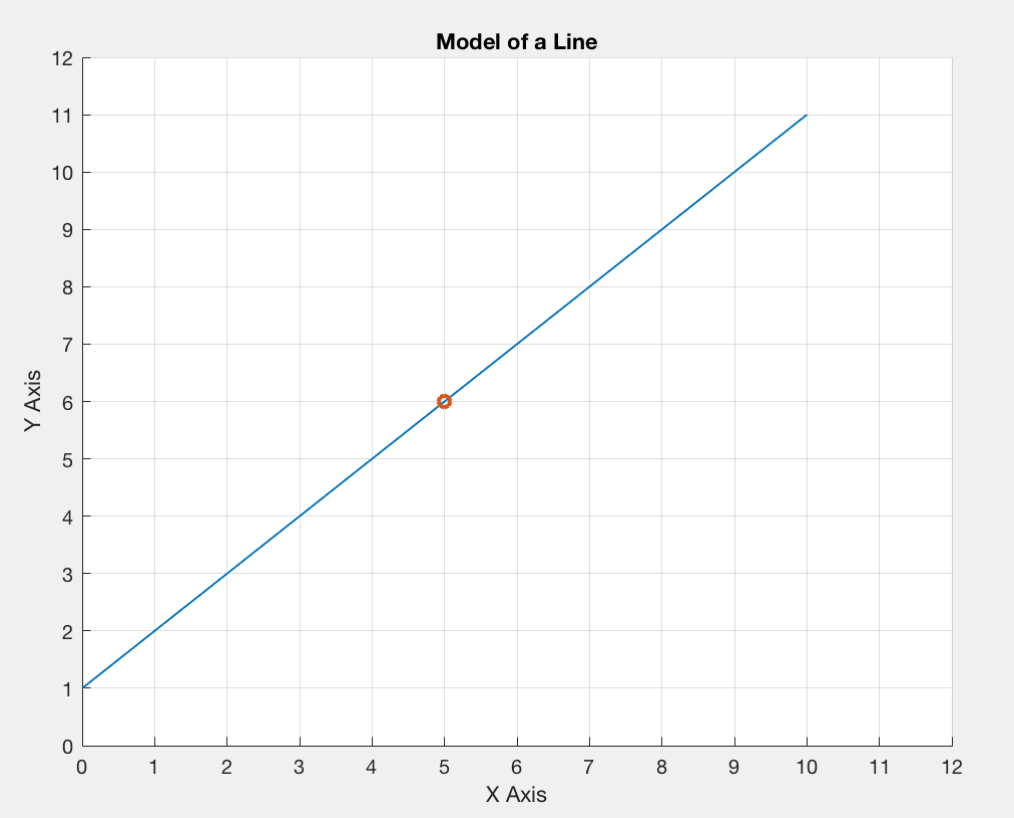

Figure 2-19: Zero-Error Model of a Line

This zero-error modeling process can be extended to more complicated algorithms to ensure that the algorithm functions as expected in zero error conditions. A zero-error model also does not have to consist of one case. Often, varieties of cases are tested to examine bounds and otherwise increase the evaluator's confidence that their algorithm is correct.

2.4.2 Monte Carlo Simulation for Evaluation

In the real world, measurement errors exist. Despite knowledge of their existence, however, it is often difficult to quantify the influences of these errors when developing an algorithm. As a result, this report utilizes techniques that introduce simulated errors as input uncertainties into a system to characterize system performance in the presence of errors without the complications of testing in the real world. A Monte Carlo simulation is one such technique, which commonly uses the Gaussian distribution to model input errors.

Gaussian Distribution

A Gaussian distribution, also known as a normal distribution, is one of the most commonly known probability distributions. As represented by its histogram in Figure 2-20, the standard Gaussian distribution has zero mean and a standard deviation of one.

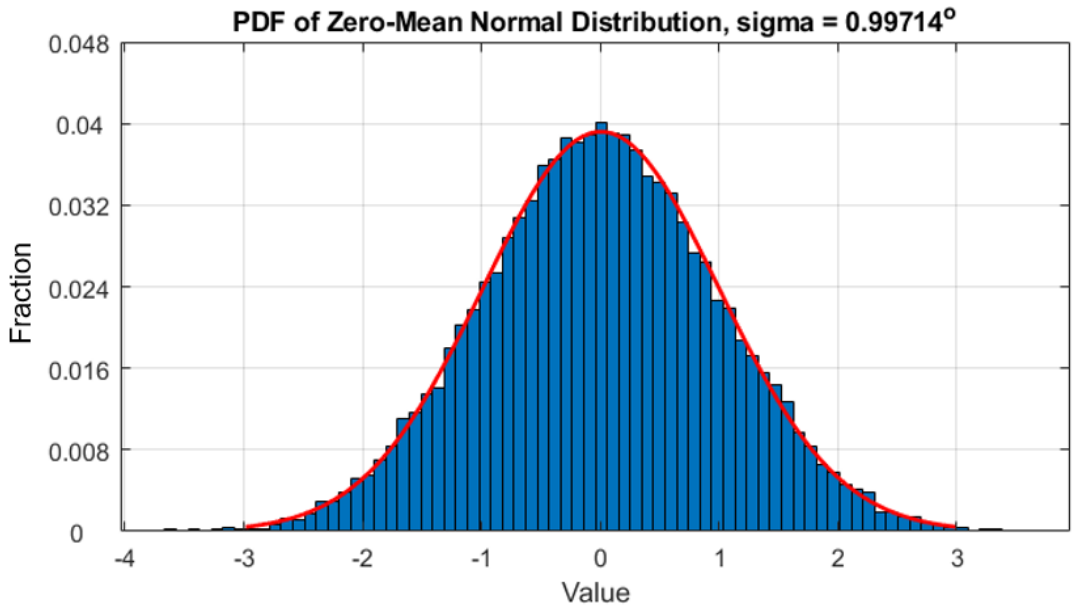

Figure 2-20: Zero-Mean Gaussian Distribution

By varying the standard deviation, different distributions can be created that produce errors of different magnitudes.

The standard deviation (σ) defines the percentage of values that will fall within a certain range. *One-*σ is the percentage of values that fall between $-\sigma$ and $+\sigma$ from the mean of the distribution. For a normal distribution, that percentage is 68.2%. *Two-*σ is the percentage of values that fall between $-2(\sigma)$ and $+2(\sigma)$ from the mean of the distribution, which is 95.4%. Similarly, *three-*σ is 99.7% of values. As the value of σ increases, the percentage of values that fall within that range does as well. [Dowdy *et al*, 2004]

The ability to choose a standard deviation that will introduce errors of a likely magnitude into a system is very useful. For example, as shown in Section 2.2.1, GPS has an error of

approximately ± 5 meters. To create a zero-mean normal distribution in which 95.4% of errors will fall within ± 5 meters, the standard deviation must equal five divided by two, or 2.50 meters.

Monte Carlo Simulation Architecture

A Monte Carlo simulation introduces a distribution of errors, or uncertainties, into a model iteratively and characterizes the overall performance. The simulation allows the mapping of input uncertainties to output uncertainties, sometimes allowing a closed-form solution to be found. The Monte Carlo simulation ultimately allows the characterization of an algorithm's or system's performance.

The architecture of a Monte Carlo simulation is very simple. First, there must be a model to simulate, with a corresponding dataset for the inputs to the model. Second, there must be an error distribution. This distribution is often a Gaussian distribution. The scenario is then run in a loop, each time errors are applied to the system inputs from the error distribution. This process ensures that over the length of the Monte Carlo simulation a Gaussian distribution of errors was applied. The output can then be characterized with those inputs errors.

Monte Carlo Example Implementation

Following the aforementioned example of the zero-error model of a line, error is now introduced into the system. With *x* defined as the input, a zero-mean, independent Gaussian distribution with a standard deviation of 0.50 is applied the *x* variable. The resulting output *y* is shown for 1000 iterations as small circles in Figure 2-21.

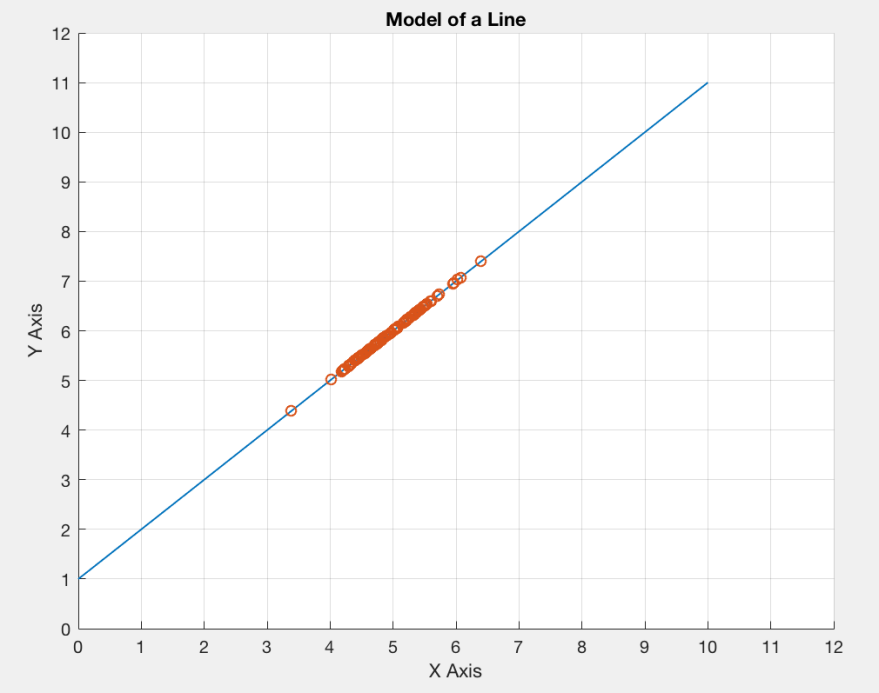

Figure 2-21: Monte Carlo Simulation of Line Model

The distribution of y is seen following the line model. There is a higher concentration in the middle centered on an x-axis value of five. The input uncertainty and output uncertainty relationship can also be determined. When calculated, the standard deviation of the distribution of *y* is found to be 0.50. This result is the same as the standard deviation of the input, as the closed-form error solution can be made for this scenario, shown in Equation 2-51.

$$\sigma_y = m \cdot \sigma_x \qquad \text{(Eq. 2-51)}$$

Being able to predict the output standard deviation given an input standard deviation is very useful when transitioning to a real-world application where it is time consuming and difficult to iteratively test a system to characterize its performance.

Chapter 3: Preliminary Work Related to Project

Two group members worked on the early stages of this project during a summer research internship at MIT Lincoln Laboratory. During this time, the members researched the theory of indirect geolocation and acknowledged flaws in current systems. Keen to utilize a system that could perform accurate indirect geolocation, while still cheap and flexible in application, the two group members studied the feasibility of using smartphones for indirect geolocation applications.

Equipment and Process

For security reasons at the Laboratory, the group members were unable to use a smartphone for testing purposes while at the Laboratory. Consequently, the team created a customized testing approach. Acting as a smartphone surrogate, the team utilized a microcontroller equipped with multiple accelerometers, gyroscopes, magnetometers, a GPS receiver, and no camera. Readily available at the lab, while also encompassing the cheap inertial and location sensors found within smartphones, the microcontroller was an acceptable surrogate for testing. As the microcontroller possessed different data-link possibilities compared to a smartphone, the team performed post-processing algorithms for simplicity. Using a micro SD card to store all sensor data, the team performed two tests to represent the two observers. The data would then be brought to a master PC which would perform all necessary parsing and testing.

Sensor Fusion

In order to optimize location and orientation estimates of the microcontroller sensors, a series of sensor fusion algorithms were tested. The first method of sensor fusion was a Madgwick Filter. Entirely open-source, the Madgwick Filter operated as a complementary filter. Much simpler to implement than a Kalman Filter, the Madgwick Filter utilized the different effective frequency ranges of sensors to create an optimal estimate. For example, the accelerometer was most accurate in the low frequency range, while the gyroscope was most accurate in the high frequency range. As these ranges complemented one another, the complementary filter combined the two measurements to create an optimized value estimate.

Although simple to implement, the complementary filter had significant flaws. Unlike the Kalman Filter, the complementary filter relied solely on measurement data. Thus, the absence of a process model meant that the complementary filter was less stable. Additionally, the complementary filter did not keep track of an error covariance matrix like the Kalman Filter. As a result, it became much more difficult to track the uncertainty of the estimates.

Having experimented with the complementary filter, the two group members began to focus on variations of the Kalman Filter. As the estimation of pose was inherently nonlinear, the members could not use the linear Kalman Filter (LKF) to create a state estimate. Instead, the team focused on using an extended Kalman Filter (EKF) due to its easier implementation and its non-linear estimation. Less complicated to implement than Kalman Filter alternatives like the Unscented Kalman Filter (UKF) and still capable of producing accurate results, the EKF was most appropriate for implementation in a smartphone surrogate.

The EKF used by the team was modeled by the algorithm proposed by Kaiqiang Feng, *et. al*. [Feng *et al*, 2017]. Utilizing the same EKF principles as described in Section 2.3.2, the algorithm based the process model on the gyroscopic measurements.

Having created the process model, the measurement model was entirely dependent on the accelerometer, magnetometer, and most recent state estimate. Using a principle known as "Two-Step Geometrically Intuitive Correction," the measurement model uniquely was not linearized. Instead, the measurements received geometric corrections based on their actual values and their theoretical values. Representing the measurement as a four-dimensional quaternion, in order to avoid "gimbal lock" and singularities, the measurement values underwent the correction process described below.

The theoretical unit vectors for the body's gravity and magnetic field were calculated using the current time-step quaternion rotation matrix and the Earth frame gravity and magnetic field vectors. Having created the values that the sensors should theoretically measure, the filter then created unit vectors of the actual measurement values. Using the laws of the dot product, the differential angle between the theoretical and actual measurement values was subsequently calculated. Following, by using the laws of the cross product, the rotation axis common to both pairs of vectors was found. Ultimately, by finding the error written as a quaternion and then performing quaternion multiplication, the corrected measurements were calculated.

By separating the effects of magnetic distortion from the accelerometer, this correction method helped create more optimal roll and pitch measurements. More important, however, was the increased computation efficiency experienced due to the simplification of the Kalman Filter model. Ultimately, such sensor fusion created a more optimal 4D quaternion orientation estimate.

Intersection

Using the determined observer orientations, the next stage of the indirect geolocation process was vector intersection. Due to time constraints, the team designed an intersection algorithm in two-dimensional (2-D) space that found the single intersection point between two 2-D vectors. Looking at the surface of the Earth in a planar sense, the team ignored any effects due to the Earth's curvature. Creating "observation vectors" indicative of each observer's orientation, the algorithm extended these vectors until they intersected with one another. The algorithm then determined this intersection point in both a Latitude, Longitude, Altitude (LLA) reference frame and an Earth-Centered Earth-Fixed (ECEF) reference frame.. In order to ensure intersection, the vectors were coplanar with the intersection surface, as well as, non-parallel and non-antiparallel. Resultantly, the altitude of both observers was assumed to the same as the altitude of the intersection point.

Results

One of the proof of concept experiments conducted over the summer is modeled as in Figure 3-1. In this experiment observer A and observer B both pointed themselves towards a radar dish on the roof of a building on the MIT Lincoln Laboratory campus. The distance to the radar dish was 78 meters and the distance between observers was 50 meters. The location and orientation of observer A and B was recorded and the 2-D intersection was found throughout the test.

The red central dots for both observers show their estimated locations from the GPS receiver. Meanwhile, the blue central dot next to the object of interest shows the estimated location from the 2-D intersection algorithm. The ellipsoids around observer A, observer B, and the intersection represent each entity's respective sigma uncertainty value for location. Sigma uncertainty is the confidence for the measurement, in this case, location assuming Gaussian

distributed errors. The smallest ellipse indicates that there is 68.2% confidence that the location lies within that circle. The second smallest ellipse indicates a similar 95% confidence. The last ellipse shows 99.7% confidence.

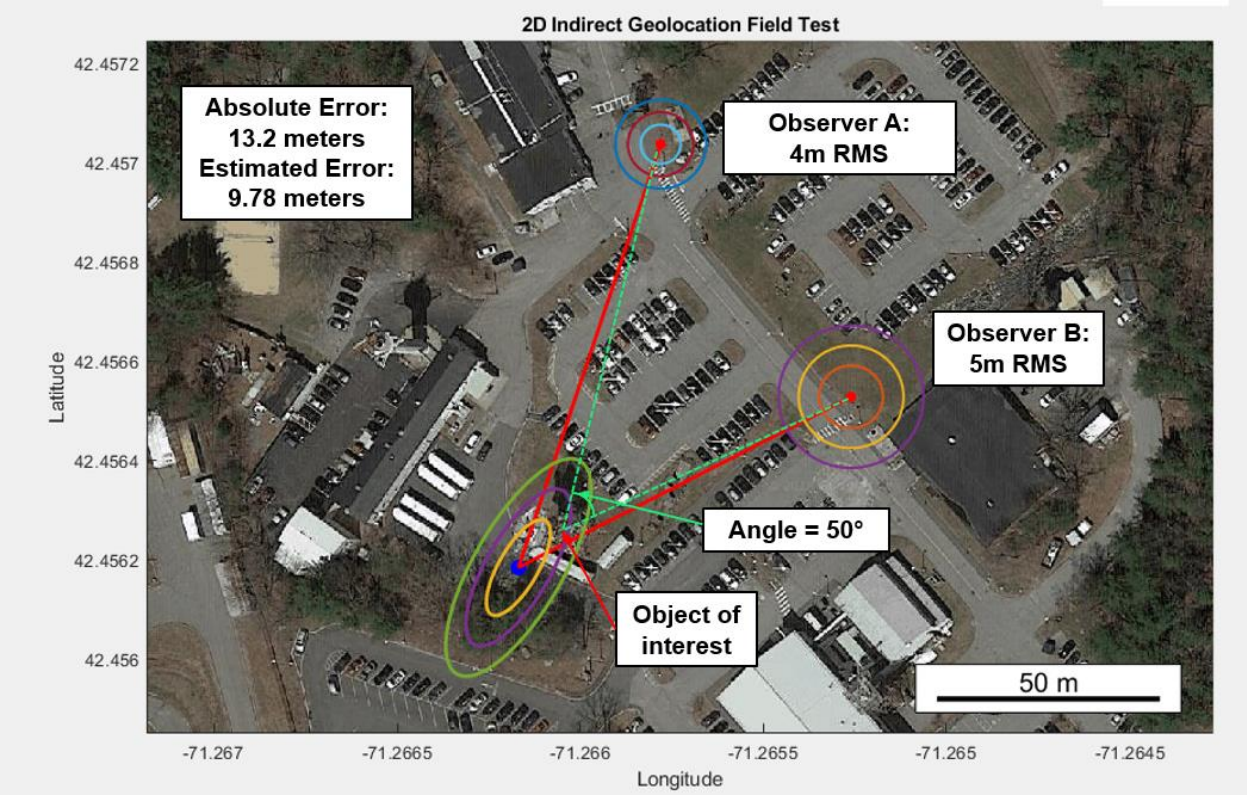

Figure 3-1: Testing Result using Surrogate Smartphones

Conclusions

The preliminary work performed for this project provided great intuition for future work. Familiarizing two of the team members with the nuances of the Kalman Filter and the difficulties of mass data analysis, the work helped the team avoid potential pitfalls. Additionally of importance is the proof of concept provided by this experimentation. Although the system used was not a smartphone, it possessed many of the qualities present within a smartphone. Obtaining an answer within the two sigma bounds, although not as optimal as desired, showed the plausibility of leveraging smartphones to perform indirect geolocation. Thus, this work helped inspire future work, while identifying the feasibility of the task.

Chapter 4: Kalman Filter Design and Algorithm Validation

In this chapter, both the design of the Kalman Filter and the validation of the intersection algorithms are presented. The extended Kalman Filter design section consists of the construction and implementation of the pre-filtering and filter state equations. Then, the two-dimensional and three-dimensional algorithms are both validated by comparing hand analysis to zero-error model algorithms created in MATLAB.

4.1 Designing the Extended Kalman Filter

In order to achieve an optimal state estimate of orientation, an extended Kalman Filter (EKF) manipulated data from the gyroscope, magnetometer, and accelerometer.

Pre-filtering:

Prior to entering the Kalman Filter, the raw gyroscope, magnetometer, and accelerometer data underwent a pre-filtering and calibration stage. The first step of the stage was NaN correction.

NaN Correction:

While testing, it was determined that the gyroscope, magnetometer, and accelerometer produced NaNs, or Not-a-Number values for approximately 2 percent of all data. Most likely due to some periodic processing algorithm done by the phone, the existence of these NaN values did little to corrupt the legitimacy of the signal. Unfortunately, as such values were often unreadable by MATLAB, however, their presence often threatened divergence with the extended Kalman Filter (EKF). Consequently, in order to ensure system functionality, NaN elimination was performed using smoothing.

The smoothing method replaced each NaN with the most recent time-step's non-NaN value. As the frequency of NaN occurrences was low with respect to the sampling rate and dynamics of the tests, this method of smoothing did not have a significant effect on the data. Had the frequency of NaNs been greater, however, such smoothing could have skewed data metrics.

Low-pass Filtering of the Accelerometer and Magnetometer

After undergoing NaN correction, the data next went through low-pass and high-pass filters. As prior stated in Section 2.2.2, the accelerometer, magnetometer, and gyroscope were all prone to sensor errors. Especially prevalent in cheaper units, such as those within smartphones, sensor errors often consisted of noise, sensor drift, sensor bias, sensor misalignment, sensor perpendicularity, and external distortion effects. Although subject to all errors, the accelerometer and magnetometer present within the phone were most prone to high frequency noise. Thus, in order to help eliminate some of the high frequency noise, data from both the accelerometer and magnetometer were passed through a fourth-order Butterworth low-pass filter with a cut-off frequency of 25 Hz.

High-pass Filtering of Gyroscope

A different procedure was used to better filter the gyroscope data. Naturally, as explained in Section 2.2.2, the gyroscope did not suffer from the same high frequency noise effects as did the accelerometer and magnetometer. Instead, the gyroscope was susceptible to low frequency noise. Due to the fact that the gyroscope measured angular rate and not orientation directly, all sensor inaccuracies by the gyroscope were integrated. A phenomenon known as drift, the orientation estimate generated by gyroscope integration increasingly diverged from truth over time in the presence of noise. An error function with respect to time, the presence of drift corrupted gyroscope measurements within seconds if left uncorrected. Thus, in order to mitigate the effects of gyroscopic drift by means of low frequency noise, all gyroscope data were first passed through a fourth-order Butterworth high-pass filter with a cut-off frequency of 0.001 Hz.

Magnetometer Calibration:

Having performed pre-filtering of the three sensors and NaN correction, the final stage prior to implementing the EKF was sensor calibration. The magnetometer, a device essential for establishing heading, measured Earth's ambient magnetic field strength. When used in an environment absent of hard and soft iron materials, the magnetometer could very accurately measure the ambient magnetic field strength. Unfortunately, when exposed to magnetic interference, the magnetometer was unable to differentiate between Earth's magnetic field and

distortion. As such, the magnetometer had to be corrected for both a translational offset and an inaccurate scaling using a transformation matrix and scaling matrix, respectively.

In order to get a better understanding of the errors for the Galaxy J7 smartphone's magnetometer, two pre-calibration tests were performed. For the first test, a data sweep was performed. Conducted in an outdoors environment away from potentially magnetic material, it was assumed that the only magnetic interference present was from the other hardware components within the phone (hard iron distortion). The data sweep consisted of the phone continuously changing orientation so that it had theoretically pointed in every direction while testing. From the data sweep, the magnetometer readings as seen in Figure 4-1 were created.

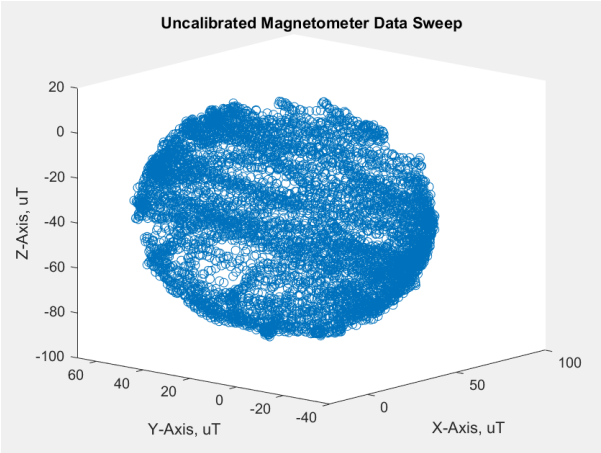

Figure 4-1: Uncalibrated Magnetometer Data Sweep

	X-Axis (uT)	Y-Axis (uT)	Z-Axis (uT)
Mean	31.4669	12.3527	-43.1674
Range	103.0050	101.3990	101.6740

Figure 4-2: Uncalibrated Magnetometer Data Sweep Metrics

As seen in Figure 4-1 and Figure 4-2, the data sweep created a roughly spherical distribution. Supportive of a uniform scaling, this spherical distribution indicated little to no soft iron distortion. In addition to the distribution, the range of the values with respect to each axis further supported the theory that the scaling matrix was uncorrupted. As the magnetic field strength of the Earth ranges from approximately -50uT to 50uT depending on orientation, these values supported a pure scaling matrix. This matrix, however, is only an estimate because the true scaling matrix is unknown.

Despite the seemingly accurate scaling matrix, the distribution did appear to have some error. As can be seen in the offset from the origin from Figure 4-1, as well as the mean values from Figure 4-2, the data distribution experienced a translational error. Indicative of a hard iron bias, such error was likely due to magnetometer biases and the surrounding hardware within the phone. Fortunately, such bias was easily corrected with a linear translation.

In order to better understand the performance of the magnetometer in a non-ideal environment, the second test introduced magnetic interference into the system. The same data sweep was performed as in the first test; however, an audio speaker was placed near the phone. Indicative of a less controlled environment experiencing magnetic distortion, this test specifically targeted the axial scaling abilities of the magnetometer. Figure 4-3 illustrates the magnetometer data.

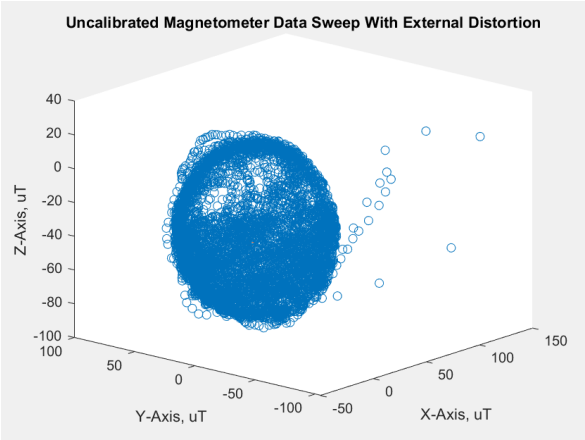

Figure 4-3: Uncalibrated Magnetometer Data Sweep With Interference

	X-Axis (uT)	Y-Axis (uT)	Z-Axis (uT)
Mean	17.3791	12.3527	-43.1674
Range	190.1250	176.2500	126.7490

Figure 4-4: Uncalibrated Magnetometer Data Sweep With Interference Metrics

Evident in Figures 4-3 and 4-4, the presence of magnetic interference severely corrupted the accuracy of the distribution. As seen by the “egg” shape of the data distribution and the dramatic increase in range values, the scaling matrix of the magnetometer was altered. The presence of hard iron effects additionally varied the linear offset experienced by the data. Of most importance, however, was the distortion of the scaling matrix due to soft iron effects.

Having established the need to correct for magnetic offset and more specifically scaling, the magnetometer data underwent a calibration. In order to convert the point cloud of data, as seen in Figure 4-3, into a three-dimensional geometric shape, the data first underwent a spherical fit. Specifically, this fit helped characterize the distribution of the point cloud.

With the center, radii, and eigenvectors of the point cloud determined from the spherical fit function, the data then received its first adjustment, where the center was shifted as shown in Equation 4-1.

$$Magnetometer\_x - Center\_x = New\_Magnetometer\_x(i)$$

$$Magnetometer\_y - Center\_y = New\_Magnetometer\_y(i) \qquad \text{(Eq. 4-1)}$$

$$Magnetometer\_z - Center\_z = New\_Magnetometer\_z(i)$$

With each axial array of data adjusted by the scalar offset of the data distribution, a linearly translated data sweep was generated as seen in Figure 4-5. Much closer to the origin, as can be seen by the mean values of Figure 4-6, this translation helped accounted for some of the offset due to hard iron effects.

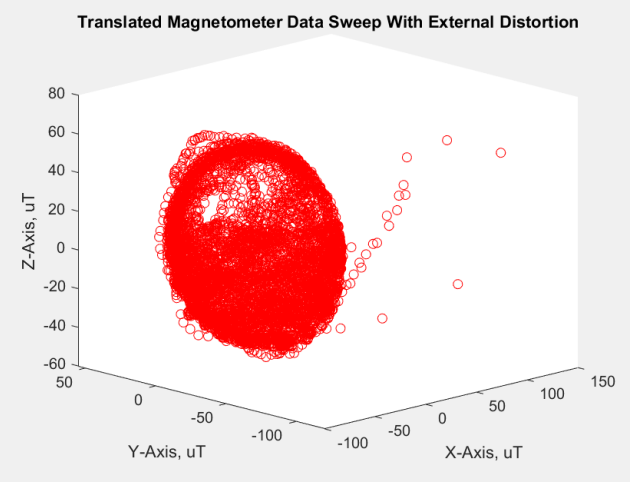

Figure 4-5: Translation of Offset Magnetometer Data

	X-Axis (uT)	Y-Axis (uT)	Z-Axis (uT)
Mean	-9.4204	-3.3177	-5.6302
Range	190.1250	176.2500	126.7490

Figure 4-6: Translation of Offset Magnetometer Data Metrics

With most of the hard iron effects corrected, the next step was the calibration of the scaling matrix. Using the calculated radii from the spherical fit, a new scaling matrix was created. Then, using the eigenvectors, a transformation matrix was created in order to map the spherical axes to the coordinate system. Ultimately, by multiplying the new scaling matrix with the transformation matrix, a final fully compensated matrix was created.

As seen in Figures 4-7 and 4-8, the new matrix centered the sphere very close to the origin. The data still experienced a slight offset due to assumptions made by the spherical fit algorithm; however, such offset was negligible in comparison to the previous offset values. Of more importance was the correction of the shape of the data distribution. Possessing much more of a spherical shape as opposed to the "egg" shape from earlier, the scaling matrix helped compensate for the distortion effects of soft iron. Furthermore, the scaling matrix also helped reduce the range of each axis to values closer to the range of Earth's magnetic field strength.

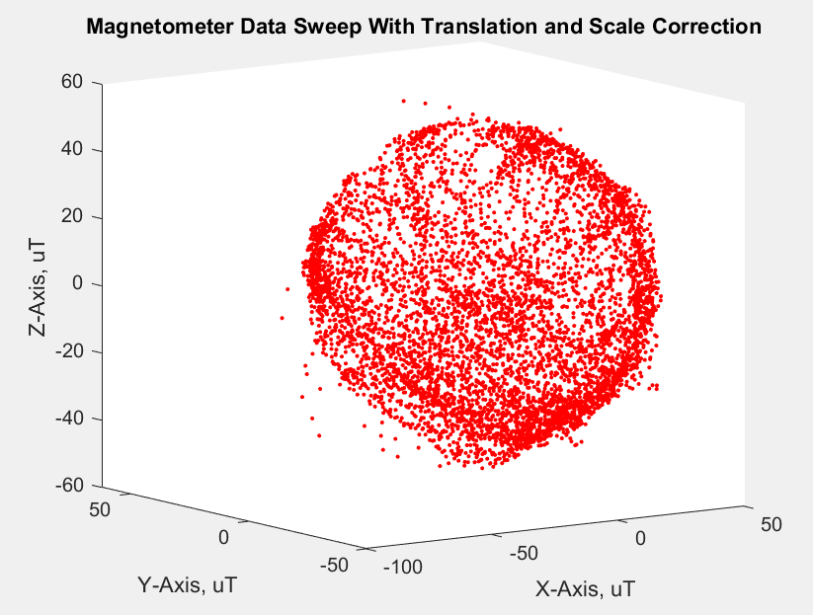

Figure 4-7: Fully Calibrated Magnetometer Data Sweep

	X-Axis (uT)	Y-Axis (uT)	Z-Axis (uT)
Mean	-0.2942	-2.3870	-2.6020
Range	126.5507	108.9374	108.6303

Figure 4-8: Fully Calibrated Magnetometer Data Sweep Metrics

Gyroscope Calibration:

Meanwhile, for the gyroscope, the calibration was much simpler. Due to the fact that the gyroscope measured angular rate, a stationary gyroscope theoretically measured zero degrees per second across all three axes. Thus, by placing the gyroscope on a flat surface and letting it collect data, the sensor inaccuracies of the gyroscope were exposed. Although the high-pass filter helped mitigate some of the low frequency noise, the presence of sensor biases still created drift. Inherent to the sensor and independent about each of the sensor's three axes, sensor bias created a permanent offset that very slowly changed with time. As seen in Figure 4-9, the sensor biases native to the Galaxy J7 used in testing were as follows.

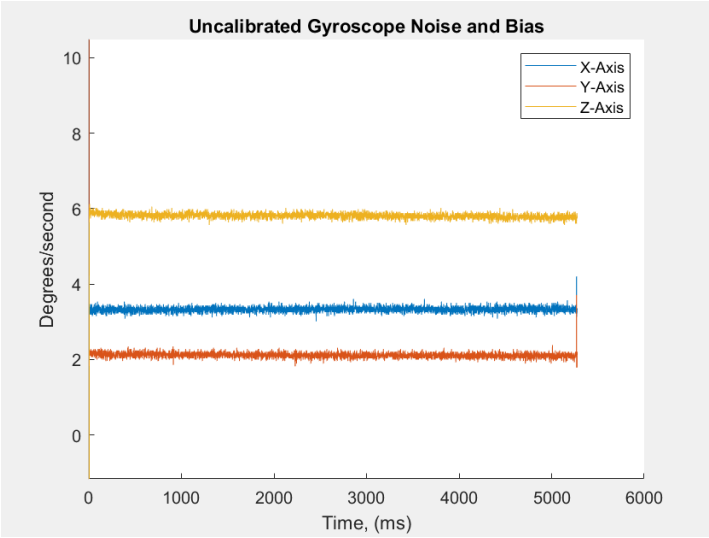

Figure 4-9: Gyroscope Sensor Biases

Having captured the approximate biases from the stationary gyroscope, the angular rates were integrated to generate relative orientation values. As the gyroscope was stationary, each of the three axes was expected to report values of zero degrees for the duration of the report. From Figure 4-10, however, a linear drift of all angles occurred.

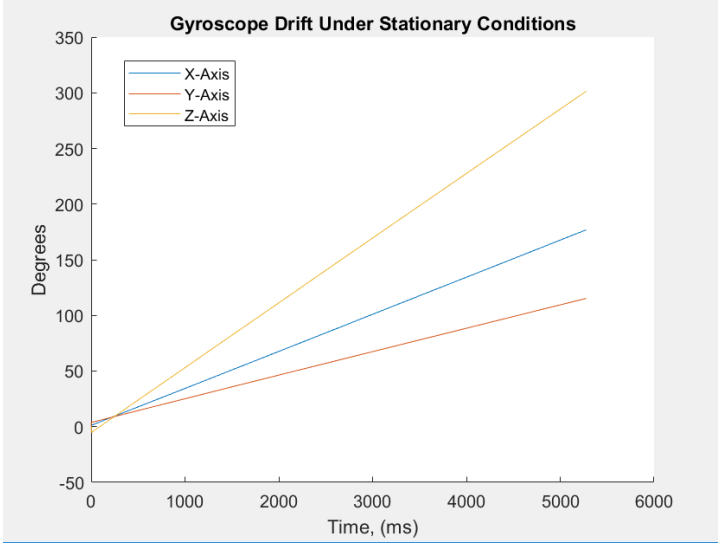

Figure 4-10: Gyroscope Drift

With the gyroscope bias eliminated by a linear offset, Figure 4-9 transformed into Figure 4-11 and Figure 4-10 transformed into Figure 4-12.

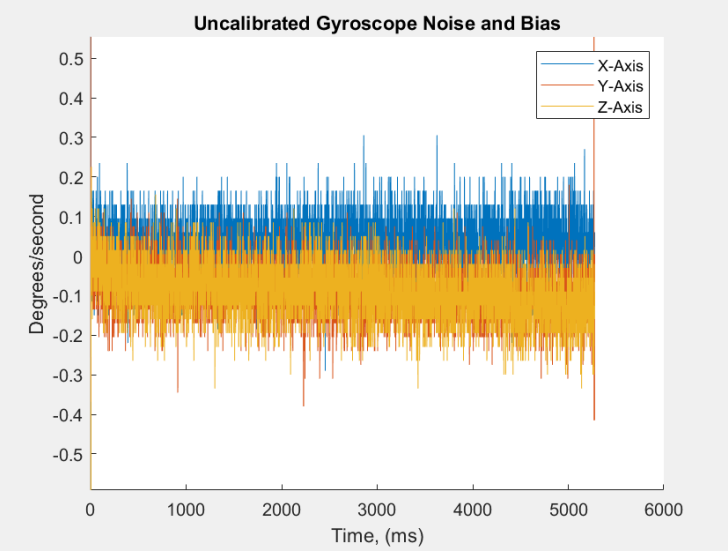

Figure 4-11: Gyroscope Sensor Bias After Calibration

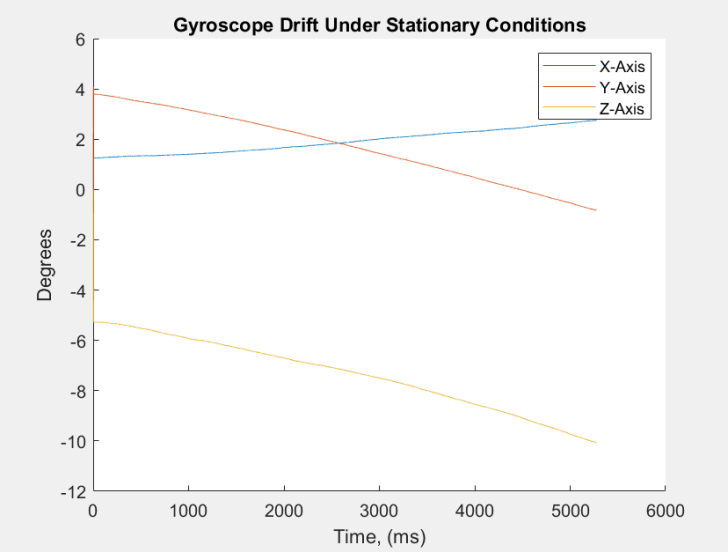

Figure 4-12: Gyroscope Drift With Bias After Calibration

Consequently, the gyroscope drift significantly decreased in amplitude, most notably along the Z-Axis.

Extended Kalman Filter

State:

For maximum computational efficiency, the proposed extended Kalman Filter (EKF) only monitored the four-dimensional quaternion, q, which represented the orientation. Modeled in Equation 4-2, the quaternion state vector was comprised of a scalar value, q_0, and a vector value, $[q_1, q_2, q_3]$.

$$x = [q_0, q_1, q_2, q_3] \quad \text{(Eq. 4-2)}$$

For additional system accuracy, the state could have also contained parameters for gyroscope and magnetometer biases (${\delta_{gyro}}^b, {\delta_{mag}}^b$). Helpful in mitigating the negative effects of noise and drift, these additional sub-states would have provided increased knowledge regarding the effectiveness of the system. As the data already passed through significant prefiltering and calibration though, the increased complexity of monitoring these additional parameters was unnecessary for the marginal expected improvement in performance.

Process Model:

As stated in Section 2.3.2, the process model of the EKF was based on Equation 4-3.

$$\dot{x}(t) = f(x,t) + g(w,t) \quad \text{(Eq. 4-3)}$$

In this continuous-time equation, *x* represented the state vector of the system, *w* represented the Gaussian noise of the system, $\dot{x}$ represented the time derivative of the state vector, and *f* and *g* represented nonlinear functions.

Indicative of the relationship between a state and its time derivative, Equation 4-3 was remodeled to calculate the quaternion orientation as seen in Equation 4-4.

$$\dot{q}(t) = \frac{1}{2}[\Omega \times]q(t) \quad \text{(Eq. 4-4)}$$

where,

$$\frac{1}{2}[\Omega \times] = \frac{\partial f(x,t_k)}{\partial x}$$

As the state *x* from Equation 4-3 only monitored the quaternion orientation, all instances of *x* were replaced with *q.* Additionally, the noise parameter, *w*, and its nonlinear function, *g*, were withdrawn and moved to an additional equation. The most important change between equations, however, was the replacement of the nonlinear function f with the term $\frac{1}{2}[\Omega \times]$. The 4-by-4 skew matrix of the gyroscope's angular rates, $\frac{1}{2}[\Omega \times]$ temporarily linearized the system matrix.

$$[\Omega \times] = \begin{pmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{pmatrix} \quad \text{(Eq. 4-5)}$$

As seen in Equation 4-5, the skew matrix was comprised of the gyroscope's angular rates ($\omega_k = [\omega_{x,k}, \omega_{y,k}, \omega_{z,k}]$). Due to the fact that the gyroscope only performed discrete-time measurements, however, Equations 4-4 and 4-5 had to be remodeled. Using the Van Loan procedure, Equation 4-4 was rewritten as Equation 4-6. Assuming the time between iterations (dt) was appropriately small, Equation 4-6 calculated the state transition matrix, $\emptyset_k$. Based on the

principle that $f(x, t_k) \approx f(x, t_{k-1})$, $\emptyset_k$ represented the discrete-time linearization of the nonlinear function f. Furthermore, Equation 4-5 was rewritten as seen in Equation 4-7, where the subscript *k* represented the discrete-time sampling index.

$$\emptyset_k \approx \exp([\Omega \times]_k * dt) \quad \text{(Eq. 4-6)}$$

$$[\Omega \times]_k = \begin{pmatrix} 0 & -\omega_{x,k} & -\omega_{y,k} & -\omega_{z,k} \\ \omega_{x,k} & 0 & \omega_{z,k} & -\omega_{y,k} \\ \omega_{y,k} & -\omega_{z,k} & 0 & \omega_{x,k} \\ \omega_{z,k} & \omega_{y,k} & -\omega_{x,k} & 0 \end{pmatrix} \quad \text{(Eq. 4-7)}$$

Ultimately, using this state transition matrix, $\emptyset_k$, and the previous time-step's *a posteriori* state estimate, $\hat{x}_{k-1}^{+}$, the current time-step's *a priori* state estimate, x_k^{-}, was calculated as seen in Equation 4-8.

$$x_k^{-} = \emptyset_k * \hat{x}_{k-1}^{+} \quad \text{(Eq. 4-8)}$$

Measurement Model:

While the process model was based on the gyroscope data, the measurement model was based on the accelerometer ($a = [a_{x,k}, a_{y,k}, a_{z,k}]$) and magnetometer ($m_k = [m_{x,k}, m_{y,k}, m_{z,k}]$) data. In regard to the accelerometer, its ability to measure external specific force, including the force of gravity, allows it to act as a tilt sensor. Specifically, the accelerometer was capable of measuring both pitch and roll.

As seen in Equation 4-9, the first step of the measurement model was the estimation of pitch.

$$pitch = \sin^{-1}\left(\frac{-a_{y,k}}{|a_k|}\right) \quad \text{(Eq. 4-9)}$$

Taking the inverse sine of the ratio between the y-axis acceleration, $a_{y,k}$, and the magnitude of the acceleration vector, $|a_k|$, a pitch angle approximation was generated in radians. In a similar manner, the roll was estimated as seen in Equation 4-10.

$$roll = \sin^{-1}\left(\frac{a_{x,k}}{|a_k|}\right) \quad \text{(Eq. 4-10)}$$

Having created satisfactory pitch and roll estimates for static or quasi-static conditions, the next step was the implementation of the magnetometer. Providing a reference to magnetic north, the magnetometer helped create an azimuth (yaw) approximation which connected the body frame of the sensors to the inertial frame of the Earth. As seen in Equation 4-11, the computation of the azimuth angle was broken into several parts. Using the tilt data provided by the accelerometer (roll and pitch), the magnetometer data helped determine the Earth's magnetic field components. As these data helped create a planar representation of Earth's magnetic field, the azimuth could then be computed using the laws of trigonometry.

$$azimuth = \tan^{-1}\left(\frac{y}{x}\right) \quad \text{(Eq. 4-11)}$$

where,

$$y = -m_{x,k} * \cos(roll) + m_{z,k} * \sin(roll)$$

$$x = m_{x,k} * \sin(pitch) * \sin(roll) + m_{y,k} * \cos(pitch) + m_{z,k} * \sin(pitch) * \cos(roll)$$

With the angle estimations computed in terms of radians, the measurement model then converted the angles calculated in the measurement model into quaternions. Ultimately, this conversion allowed the process model quaternion estimate to be compared to the measurement model quaternion estimate, z_k.

Noise:

In both the process and measurement models, noise and inaccuracies corrupted the orientation estimations. Due to random sensor noise, sensor biases, external interference, and axis misalignment, uncertainties altered system accuracy. For this specific application, the prefiltering and calibration most likely reduced most uncertainties with sensor biases, axis misalignment, and external interference; however, the sensors were still prone to some noise variances.

It was assumed that the noise of the accelerometer (Σ_{acc}), magnetometer (Σ_{mag}), and gyroscope (Σ_{gyro}) followed an independent zero-mean Gaussian (normal) distribution. Additionally, it was assumed that each axis possessed independent variance, σ^2, with no correlation between axes. Thus, the noise matrices were modeled as seen in Equation 4-12 with values of zero at all off-diagonal elements.

$$\Sigma_{acc} = \begin{bmatrix} \sigma_{a,x}^2 & 0 & 0 \\ 0 & \sigma_{a,y}^2 & 0 \\ 0 & 0 & \sigma_{a,z}^2 \end{bmatrix}$$

$$\Sigma_{mag} = \begin{bmatrix} \sigma_{m,x}^2 & 0 & 0 \\ 0 & \sigma_{m,y}^2 & 0 \\ 0 & 0 & \sigma_{m,z}^2 \end{bmatrix} \quad \text{(Eq. 4-12)}$$

$$\Sigma_{gyro} = \begin{bmatrix} \sigma_{\omega,x}^2 & 0 & 0 \\ 0 & \sigma_{\omega,y}^2 & 0 \\ 0 & 0 & \sigma_{\omega,z}^2 \end{bmatrix}$$

Just as the process and measurement models propagated the previous iteration's state vector to a current-time value, the models also propagated their respective noise matrices. In order to model the covariance matrix of the measurement model, R_k, the accelerometer and magnetometer covariance matrices were combined. As shown in Equation 4-13, the new 6-by-6 covariance matrix, $\Sigma_{a,m}$, modeled both sensor uncertainties.

$$\Sigma_{a,m} = \begin{bmatrix} \sigma_{a,x}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{a,y}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{a,z}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{m,x}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{m,y}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{m,z}^2 \end{bmatrix} \quad \text{(Eq. 4-13)}$$

In order to propagate the covariance matrix from Equation 4-13, the measurement model was used. Given the nonlinearity of the measurement model, the measurement model was temporarily linearized. As seen in Equation 4-14, the Jacobian or first-order partial derivative of the measurement model was taken for linearization.

$$J = \frac{\partial q_{meas}}{\partial \{a_x, a_y, a_z, m_x, m_y, m_z\}} \quad \text{(Eq. 4-14)}$$

To better illustrate the meaning of the Jacobian, the linearization equation is expanded in Equation 4-15. Specifically, the partial derivatives of the Jacobian were estimated using first backward differences.

$$J = \begin{bmatrix} \frac{\partial q_0}{\partial a_x} & \frac{\partial q_0}{\partial a_y} & \frac{\partial q_0}{\partial a_z} & \frac{\partial q_0}{\partial m_x} & \frac{\partial q_0}{\partial m_y} & \frac{\partial q_0}{\partial m_z} \\ \frac{\partial q_1}{\partial a_x} & \frac{\partial q_1}{\partial a_y} & \frac{\partial q_1}{\partial a_z} & \frac{\partial q_1}{\partial m_x} & \frac{\partial q_1}{\partial m_y} & \frac{\partial q_1}{\partial m_z} \\ \frac{\partial q_2}{\partial a_x} & \frac{\partial q_2}{\partial a_y} & \frac{\partial q_2}{\partial a_z} & \frac{\partial q_2}{\partial m_x} & \frac{\partial q_2}{\partial m_y} & \frac{\partial q_2}{\partial m_z} \\ \frac{\partial q_3}{\partial a_x} & \frac{\partial q_3}{\partial a_y} & \frac{\partial q_3}{\partial a_z} & \frac{\partial q_3}{\partial m_x} & \frac{\partial q_3}{\partial m_y} & \frac{\partial q_3}{\partial m_z} \end{bmatrix} \quad \text{(Eq. 4-15)}$$

The Jacobian linearized the measurement model about the linearization point, q_{meas}. Having computed the Jacobian, J, the measurement model covariance matrix, R_k, was then computed as in Equation 4-16.

$$R_k = J\Sigma_{a,m}J^T \quad \text{(Eq. 4-16)}$$

Similar to the measurement model covariance matrix, R_k, the process noise covariance matrix, Q_k, was calculated by the propagation of the gyroscope covariance matrix, Σ_{gyro}. Unlike the measurement model, in which the system was linearized via the Jacobian, the process model utilized matrix Ξ_k. Related to the 4-by-4 skew matrix, $[\Omega \times]$, matrix Ξ_k was the linearized approximation to the nonlinear function g from Equation 4-3. Thus, the process noise covariance matrix Q_k, was calculated as seen in Equation 4-17.

$$Q_k = \left(\frac{\Delta t}{2}\right)^2 \Xi_k \Sigma_{gyro} \Xi_k^T \quad \text{(Eq. 4-17)}$$

where,

$$\Xi_k = \begin{pmatrix} q_1 & q_2 & q_3 \\ -q_0 & -q_3 & -q_2 \\ q_2 & -q_0 & -q_1 \\ -q_2 & q_1 & -q_0 \end{pmatrix}$$

Kalman Loop:

Having created both a process and measurement model both accompanied by their respective noise matrices, the optimization process via the Kalman Filter was performed. As prior written in Equation 4-18, the first step of the EKF was the state prediction using the process model.

$$x_k^- = \emptyset_k * \hat{x}_{k-1}^+ \quad \text{(Eq. 4-18)}$$

As seen in the restatement of Equation 4-8, the previous time-step state estimate, $\hat{x}_{k-1}^{+}$, was propagated by the 4-by-4 matrix $\emptyset_k$. Using that same transition matrix and the process noise covariance matrix, Q_k, the predicted covariance was then recursively updated as in Equation 4-19.

$$P_k^- = \emptyset_k {P_{k-1}}^+ {\emptyset_k}^T + Q_k \quad \text{(Eq. 4-19)}$$

With the *a priori* predictions of both the state, x_k^-, and covariance, P_k^-, the next step of the EKF algorithm was the determination of the Kalman gain, G_k. As modeled in Equation 4-20, the Kalman gain was the filter's weighting factor that was dependent on the current state covariance matrix, P_k^-, and the current state measurement noise R_k. Weighing their relative uncertainties with one another, the Kalman gain determined whether the process or measurement model was more trustworthy.

$$G_k = \frac{P_k^-}{P_k^- + R_k} \quad \text{(Eq. 4-20)}$$

With an optimized weight for the filter, the state correction was next performed (Equation 4-21). This updated estimate, known as the *a posteriori* estimate, reflected the inputs from both the measurement and process models. Leveraging the data provided by both models and the Kalman gain from Equation 4-19, the filter deduced a linear approximation of the optimal state estimate.

$$\hat{x}_k^+ = x_k^- + G_k(z_k - x_k^-) \quad \text{(Eq. 4-21)}$$

Ultimately, the Kalman gain weighing factor was used to generate the uncertainty of the optimized state estimate from Equation 4-21. Proven to be smaller or at least the same value as the *a priori* covariance prediction in Equation 4-18, this *a posteriori* covariance was calculated using Equation 4-22.

$$P_k^+ = (I_{4x4} - G_k)P_k^- \quad \text{(Eq. 4-22)}$$

The Kalman Filter full procedure is shown in Figure 4-13.

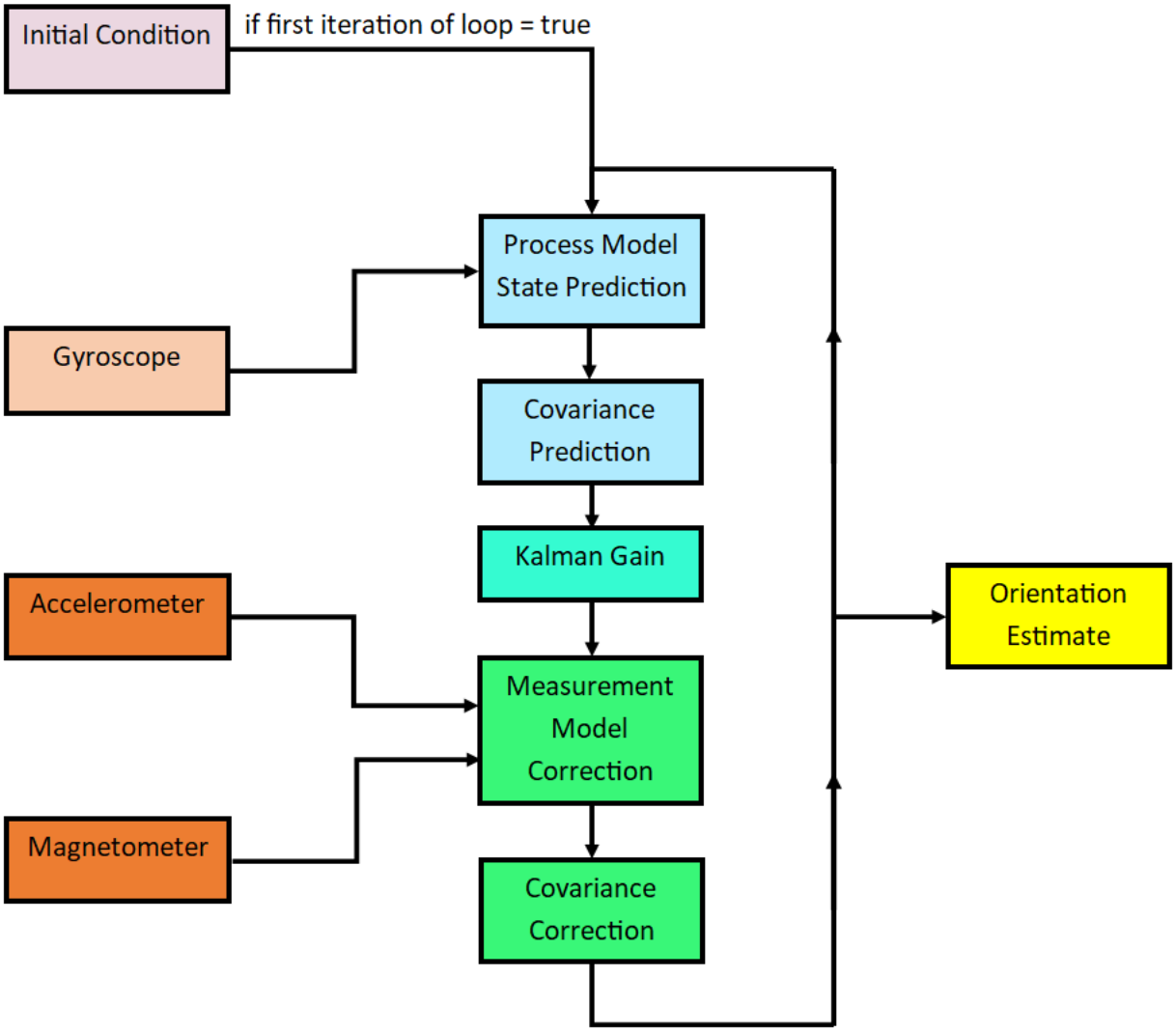

Figure 4-13: Kalman Filter Procedure for Optimized Orientation Estimation

4.2 Validating the Two-Dimensional Intersection Algorithm

The two-dimensional (2-D) intersection algorithm described in Section 2.1.2 was implemented in MATLAB as a standalone function. Figure 4-14 shows the four inputs and single output of the intersection algorithm.

Figure 4-14: Two-Dimensional Intersection Algorithm System Block

The designed intersection algorithm required horizontal position and orientation as inputs for both observers. In the real-world these inputs would be latitude-longitude and yaw, however, for testing and simulation they were defined on a 2-D x-y coordinate plane with an orientation angle measured with respect to the +y-axis. Therefore, in total there were six inputs required for the intersection algorithm.

The first step in characterizing the algorithm was to create a zero-error model. As described in Section 2.4.1, zero-error models helped validate that an algorithm produced the expected results in a zero-error environment. As such, these zero-error models were designed to model several scenarios in which both observers are oriented towards the same object of interest. Ultimately, the algorithm inputs and outputs for each scenario were determined analytically and then compared to the MATLAB algorithm output to validate the intersection points produced from the algorithm.

Methods

An example for analytically determining each scenario's inputs and outputs is shown below. The intersection algorithm inputs included the 2-D coordinates and orientation (angle with respect to the +y-axis) of both observers. Processing said input, the algorithm then produced a single output which was the object of interest's 2-D coordinates. The first task in creating each scenario in the zero-error model was to determine locations for both observers and the object of interest. To begin, a coordinate system was established as a 2-D plane, shown in Figure 4-15. In

the coordinate system, the x and y-axes were perpendicular, with a 0^o theta angle corresponding to the +y-axis.

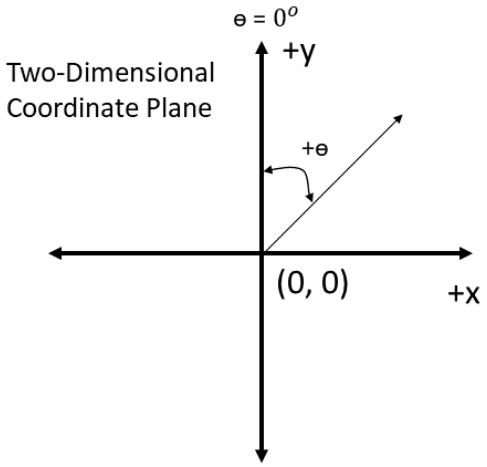

Figure 4-15: Two-Dimensional Coordinate Plane

Having created a coordinate system, the location of the observers were then chosen within this 2-D plane. For example, one scenario had observer A at (-146.4102 m, 0 m) and observer B at (-100 m, -100 m). The object of interest was chosen to be (200 m, 200 m). Figure 4-16 shows this scenario.

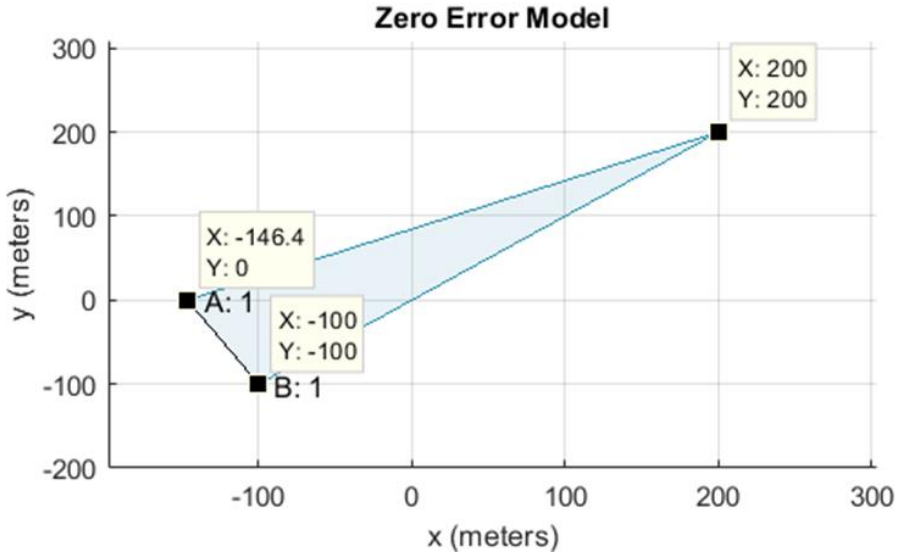

Figure 4-16: Scenario with Two Observers and Object of Interest

The only variable remaining, orientation, was then solved for by creating two triangles, one for each observer, with the hypotenuse representing the pointing vector towards the object of interest. The two triangles are shown below in Figure 4-17.

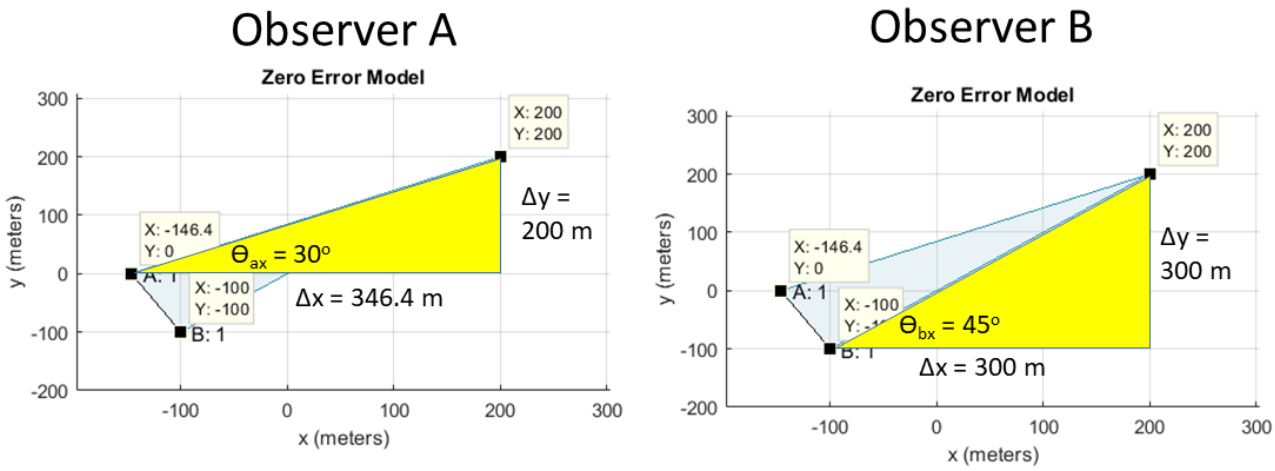

Figure 4-17: Triangles Used to Solve for Orientation

Using trigonometry, the orientation for each observer with respect to the +x-axis, θ_{ax} and θ_{bx}, were found. Equation 4-23 shows how this angle was found using the tangent function, the change in vertical position, Δy, and the change in horizontal position, Δx, for observer A.

$$\theta_{ax} = \tan^{-1}\left(\frac{\Delta y}{\Delta x}\right) \quad \text{(Eq. 4-23)}$$

Then each observer's angle was shifted by 90° to align 0° with the +y-axis as shown in Equation 4-24.

$$\theta_{ay} = 90^{o} - \theta_{ax} \quad \text{(Eq. 4-24)}$$

Having determined both observers' orientations, all inputs and outputs for this zero-error scenario were found. For further validation, this process was completed for an additional 9 scenarios that covered a range of different situations. Appendix A shows the complete data set used to construct the full zero-error model while Figure 4-18 shows a representative model graphically.

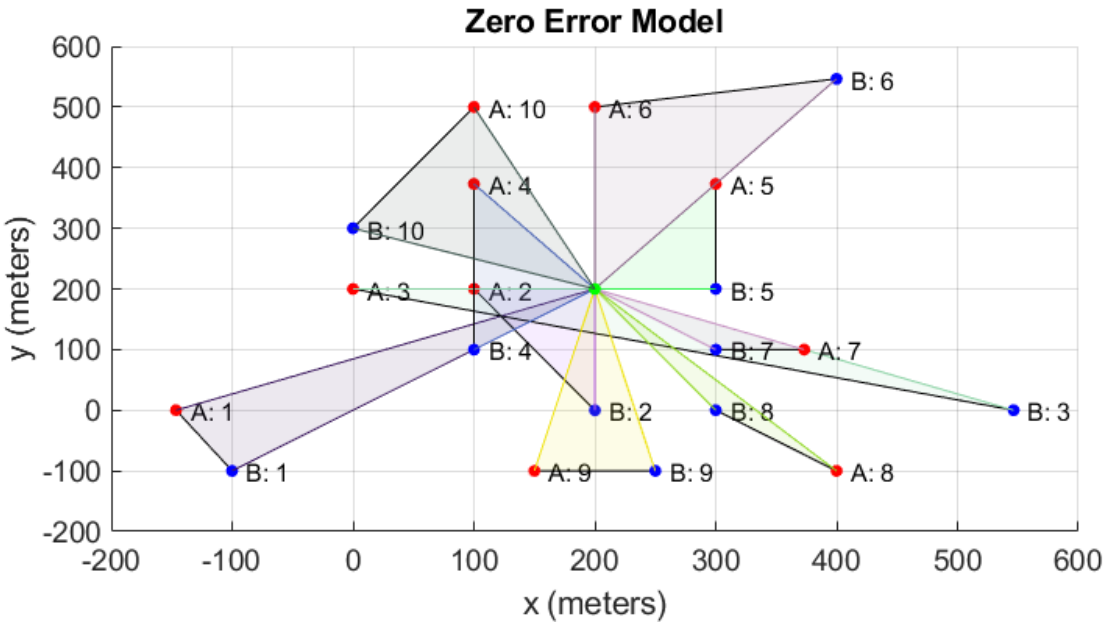

Figure 4-18: Zero-Error Model with Ten Scenarios

In Figure 4-18 above, points are labeled as observer A or B with the appropriate number pair to which they belong. For example, in the bottom left of Figure 4-18, points "A: 1" and "B: 1" are a pair of observers. For consistency, the intersection point was the same for all scenarios, (200 m, 200 m). The same scenarios were then used in the 2-D intersection algorithm to produce a set of intersection points, which was then compared to this model, which will be discussed in the results. Additionally, the algorithm also determined if both observers' orientations were parallel and consequently returned NaN in MATLAB for the parallel lines would never intersect.

Results

The results of the validation of the 2-D intersection algorithm consisted of comparing the distance of each intersection point to the actual truth value of (200 m, 200 m). Visually, the zero-error model, shown in Figure 4-18**,** appears to have all scenarios intersect at the correct location. When the exact error is examined, however, small magnitude errors become evident as shown in Figure 4-19.

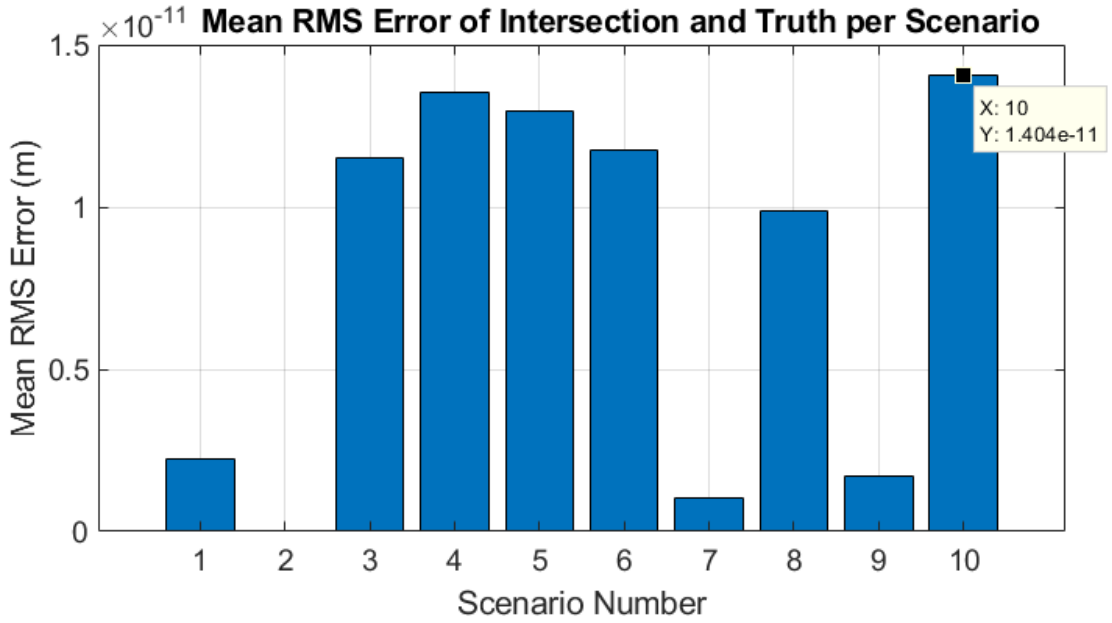

Figure 4-19: Bar Graph of Mean RMS Error in X and Y

These errors range from zero to 1.404e-11 meters, with only one scenario having exactly zero error.

Discussion

The validation of the two-dimensional intersection algorithm was completed in MATLAB, which has the capability to determine its estimated precision error. For the personal computer used to conduct the zero-error models, a MATLAB error of 2.22e-16 was observed. This value is much smaller than the maximum error presented in Figure 4-19 above; therefore, the simulation result error is not due to the precision of the personal computer. The source is most likely precision error in the zero-error model as the values used, shown in Appendix A, were only accurate to four decimal places. Also, scenario 2 shown in Appendix A, which had no decimal approximation for its input values, produced a perfect result with exactly zero error. With the only error in the intersection algorithm being due to precision error, the two-dimensional algorithm was validated.

4.3 Validating the Three-Dimensional Intersection Algorithm

Validation Methods of Three-Dimensional Intersection Algorithm

The three-dimensional (3-D) intersection method described in Section 2.1.2 was validated using a zero-error model implemented in MATLAB. This model consisted of scenarios that were chosen to represent different, simple configurations that two observers might encounter during testing. These included: 30, 60, and 90 degrees between both observers' pointing vectors. The following section details the scenarios that were tested.

The zero-error model was produced to ensure that the 3-D intersection calculation was implemented correctly in MATLAB. The first four scenarios can be grouped together, due to similar observer locations and yaw angles. Each yaw angle was with respect to the positive y-axis, with the positive x-axis being 90 degrees, and the negative x-axis being -90 degrees. Furthermore, the true location of the target was listed when the vectors intersect, but was listed as N/A when the least-squares approximation was utilized. These scenarios can be seen in Figure 4-20 below.

	Observer A			Observer B		
Scenario	X,Y,Z Position	Yaw Angle	Pitch Angle	X,Y,Z Position	Yaw Angle	Pitch Angle
1	(0 m, 0 m, 0 m)	45°	0°	(100 m, 0 m, 0 m)	-45°	0°
2	(0 m, 0 m, 0 m)	45°	20°	(100 m, 0 m, 0 m)	-45°	20°
3	(0 m, 0 m, 0 m)	45°	20°	(100 m, 0 m, 0 m)	-45°	0°
4	(0 m, 0 m, 0 m)	45°	0°	(100 m, 0 m, 0 m)	-45°	20°

Figure 4-20: 3-D Intersection Scenarios 1 to 4 Zero-Error Model

These four scenarios can be used to show that the 3-D intersection algorithm worked correctly with a 90 degree angle between the intersection of both observers' pointing vectors. The first scenario consisted of an intersection on a 2-D plane, as there was no pitch angle added for either observer and each observer was at the same Z position. Ultimately, this scenario showed that the intersection worked correctly in the 2-D plane with no least-squares approximation. The second scenario consisted of the same location and yaw angle as the previous, but there was an added pitch angle to each observer. Since the two observers had the same Z position and pitch angle, their two vectors were set to intersect. Consequently, this scenario showed that the 3-D

intersection function worked correctly, without the use of the least-square approximation. The third and fourth scenarios consisted of the same location and yaw angle as scenarios 1 and 2, but differing pitch angles. Therefore, these vectors would not intersect and the least-squares approximation was implemented. As such, the third and fourth scenarios tested the use of the least-squares approximation.

The next four scenarios, 5-8, tested similar situations with a different yaw angle configuration and location for observer A and observer B. The true location of target was listed when the vectors intersected, but was listed as N/A when the least-squares approximation was utilized. The configurations can be seen in Figure 4-21 below.

	Observer A			Observer B		
Scenario	X,Y,Z Position	Yaw Angle	Pitch Angle	X,Y,Z Position	Yaw Angle	Pitch Angle
5	(10 m, 10 m, 10 m)	30°	0°	(100 m, 10 m, 10 m)	0°	0°
6	(10 m, 10 m, 10 m)	30°	20°	(100 m, 10 m, 10 m)	0°	20°
7	(10 m, 10 m, 10 m)	30°	20°	(100 m, 10 m, 10 m)	0°	0°
8	(10 m, 10 m, 10 m)	30°	0°	(100 m, 10 m, 10 m)	0°	20°

Figure 4-21: 3-D Intersection Scenarios 5 to 8 Zero-Error Model

These four scenarios were used to test that the 3-D intersection worked with a 30 degree angle between the each observer. These testing configurations were similar to the scenarios 1-4, just with different yaw angles and starting coordinates for both observers.

The next four scenarios, 8-12, tested similar situations as 4-8, but with a different yaw angle configuration. The true location of target was listed when the vectors intersected, but was listed as N/A when the least-squares approximation was utilized. The configurations can be seen in Figure 4-22 below.

	Observer A			Observer B		
Scenario	X,Y,Z Position	Yaw Angle	Pitch Angle	X,Y,Z Position	Yaw Angle	Pitch Angle
9	(10 m, 10 m, 10 m)	60°	0°	(100 m, 10 m, 10 m)	0°	0°
10	(10 m, 10 m, 10 m)	60°	20°	(100 m, 10 m, 10 m)	0°	20°
11	(10 m, 10 m, 10 m)	60°	20°	(100 m, 10 m, 10 m)	0°	0°
12	(10 m, 10 m, 10 m)	60°	0°	(100 m, 10 m, 10 m)	0°	20°

Figure 4-22: 3-D Intersection Scenarios 9 to 12 Zero-Error Model

These four scenarios were used to show that the 3-D intersection worked with a 30 degree angle between the each observer's pointing vector. These testing configurations were similar to the scenarios 5-8, just with different yaw angles.

The last three scenarios, 13-16, tested similar situations with a different yaw angle configuration, but the same location for observer A and observer B as scenarios 5-8. The true location of target was listed when the vectors intersected, but was listed as N/A when the least-squares approximation was utilized. The configurations can be seen in Figure 4-23 below.

	Observer A			Observer B		
Scenario	X,Y,Z Position	Yaw Angle	Pitch Angle	X,Y,Z Position	Yaw Angle	Pitch Angle
13	(0 m, 0 m, 0 m)	45°	0°	(100 m, 0 m, 20 m)	-45°	0°
14	(0 m, 0 m, 0 m)	45°	20°	(100 m, 0 m, 20 m)	-45°	20°
15	(0 m, 0 m, 0 m)	45°	20°	(100 m, 0 m, 20 m)	-45°	0°
16	(0 m, 0 m, 0 m)	45°	0°	(100 m, 0 m, 20 m)	-45°	20°

Figure 4-23: 3-D Intersection Scenarios 13 to 16 Zero-Error Model

These four scenarios were used to test that the 3-D intersection least squares method worked with two observers with different starting Z coordinates. The angle configuration was similar to scenarios 1-4, except for the difference in Z coordinate.

To validate the correctness of the least squares method, geometry could not be used. Instead, the distance from each vector to the intersection point was used to determine the correctness of the intersection. As described in Section 2.1.2, the least-squares method minimized the distance from the intersection to each vector. As a result, when the distance was minimized between the two vectors, the distance was the same for each vector. Therefore, the least-squares intersection occurred at the distance intersection.

Ultimately, these scenarios were analyzed using two distinct techniques: simple geometry was used to analyze the intersection point when the vectors directly intersected and vector distance graphs were used to analyze the correct intersection position of the least-squares calculation when the vectors did not intersect.

Validation Results of Three-Dimensional Intersection Algorithm

The first scenario seen in Figure 4-20 was analyzed using simple geometry as well as the zero-error model on MATLAB. The geometry analysis can be seen below in Figure 4-24 to 4-26.

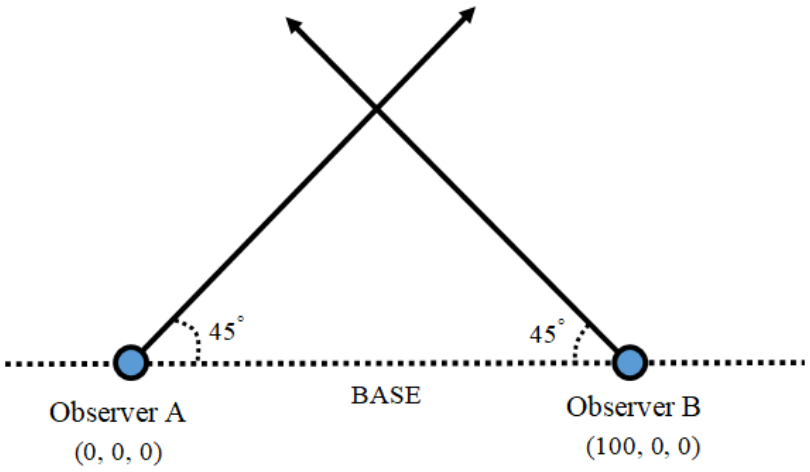

Figure 4-24: Base Scenario 1

From this scenario, the base distance was calculated using the difference in distance between observer A and B. This value is found from Equation 4-25 below.

$$BASE = \sqrt{(X_B - X_A)^2 + (Y_B - Y_A)^2 + (Z_B - Z_A)^2} \quad \text{(Eq. 4-25)}$$

Using this equation and after entering the X, Y, and Z location for observer A and B, the BASE distance was found to be 100. Next, the angle, Θ_C, and the distance, d_A, from the observer A, to the intersection point was found, shown in Figure 4-25.

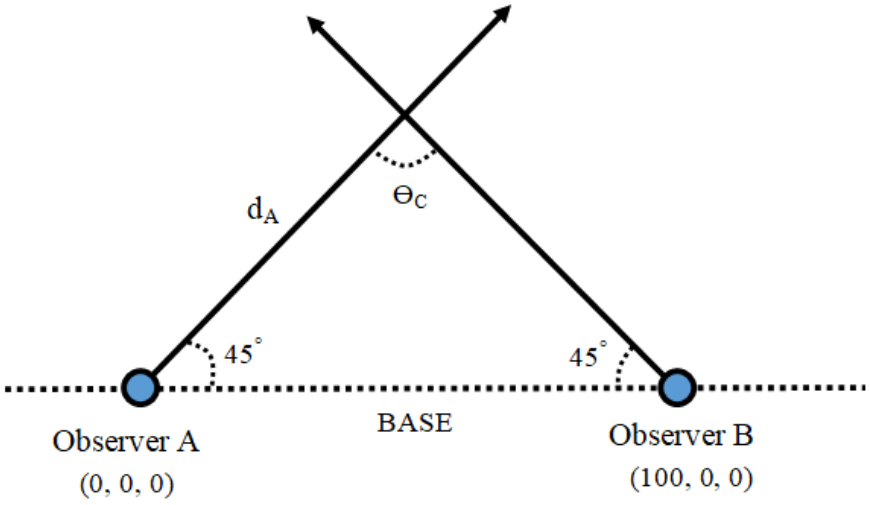

Figure 4-25: Scenario 1 d_A Configuration

The angle Θ_C can be found using Equation 4-26 below.

$$\theta_C = 180 - \theta_A - \theta_B \quad \text{(Eq. 4-26)}$$

Using this equation and after entering the angle for each observer within the triangle, Θ_C was found to be 90 degrees. With this value, the distance d_A was then found using the law of sines in Equation 4-27 below.

$$\frac{d_A}{\sin(\theta_B)} = \frac{BASE}{\sin(\theta_C)} \quad \text{(Eq. 4-27)}$$

After rearranging this equation, the distance d_A was calculated. Specifically, Equation 4-28 was used to determine this distance.

$$d_A = BASE\frac{\sin(\theta_B)}{\sin(\theta_C)} \quad \text{(Eq. 4-28)}$$

The distance d_A was equal to 70.7107. With this distance, the intersection point was calculated. The following Figure 4-26 shows the geometry that was necessary to find the intersection point.

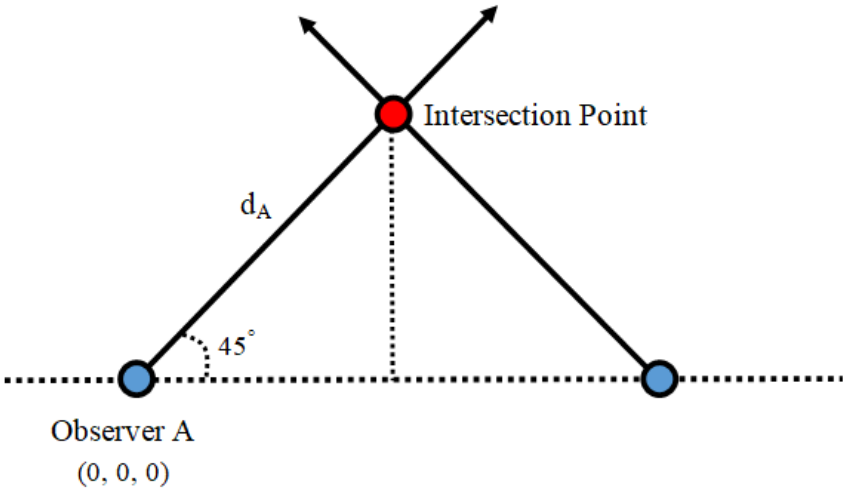

Figure 4-26: Scenario 1 Intersection

Then, using Figure 4-26, the X and Y coordinates of the intersection point were found using the following equation, with Θ_A being 45° Θ_A.

$$X_i = \cos(45°) * d_A \quad \text{(Eq. 4-29)}$$
$$Y_i = \sin(45°) * d_A \quad \text{(Eq. 4-30)}$$

Using Equations 4-29 and 4-30, the intersection point occurred at (50, 50, 0). Next, the scenario was analyzed using MATLAB. The resulting plot can be seen in Figure 4-27 and 4-28 below.

Figure 4-27: Scenario 1 MATLAB Zero Error Model View A

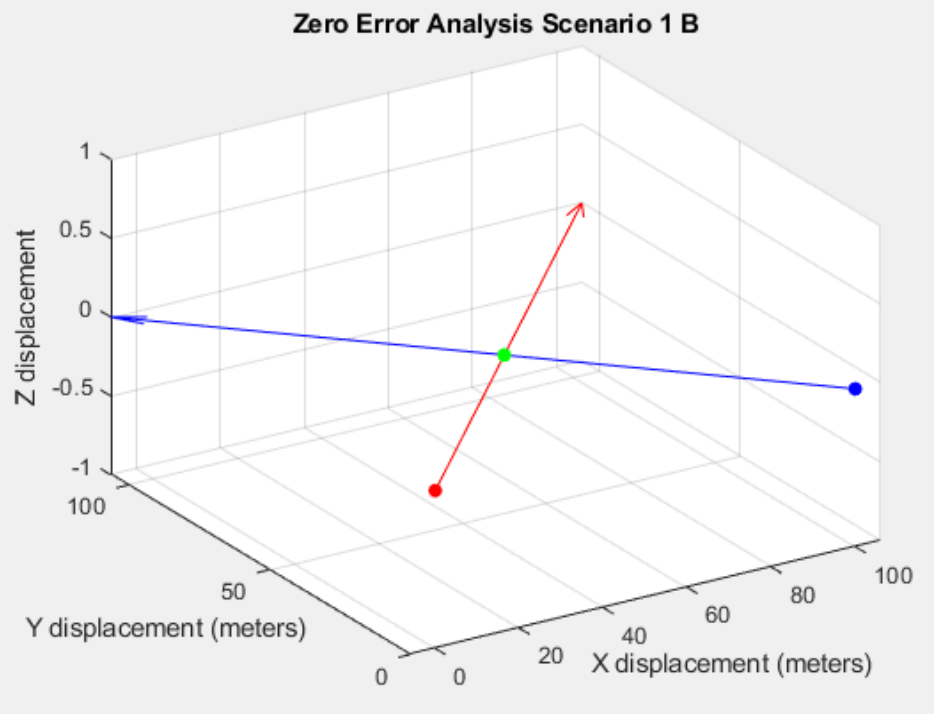

Figure 4-28: Scenario 1 MATLAB Zero Error Model View B

The resulting intersection, determined through calculations using the MATLAB algorithm, occurred at (50 m, 50 m, 0 m). This MATLAB intersection was found to match up with the intersection that was calculated using geometric analysis.

The second scenario seen in Figure 4-29 was analyzed using simple geometry as well as the algorithm on MATLAB. This scenario was similar to the analysis of scenario 1, only with an added pitch component. Due to the position and yaw being the same, and the pitch being the same for each observer, the two vectors were set to intersect. Therefore, the X, Y intersect was the same as in scenario 1. Following the intersect determination, geometric analysis was used to calculate the Z intersection coordinates. The d_A was the distance along the 2-D plane and helped determine the Z intersection coordinate from Equation 4-31, with ρ being the pitch angle.

$$Z\ Intersection = d_A * \tan(\rho) \qquad \text{(Eq. 4-31)}$$

The resulting Z coordinate intersection was 25.736 m; thus, the intersection point occurred at (50 m, 50 m, 25.736 m). Next, the scenario was analyzed using MATLAB. The resulting plot can be seen in Figure 4-29 and 4-30 below.

Figure 4-29: Scenario 2 MATLAB Zero Error Model View A

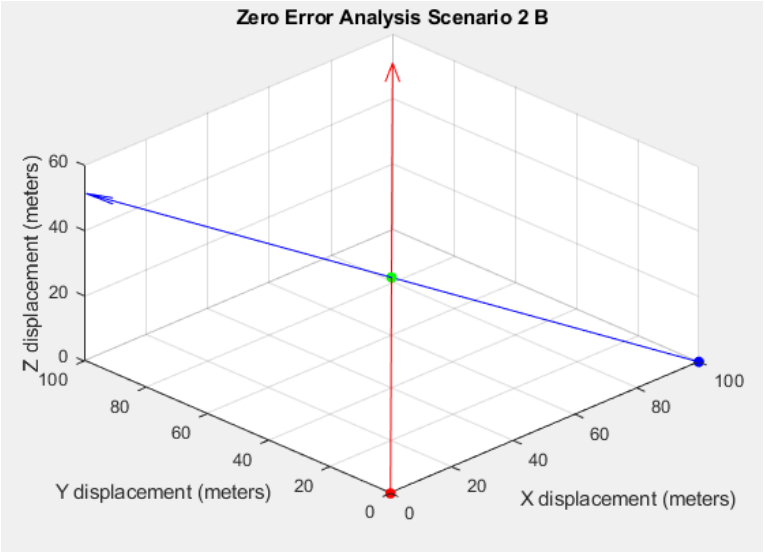

Figure 4-30: Scenario 2 MATLAB Zero Error Model View B

The resulting intersection, determined through calculations using the MATLAB algorithm, occurred at (50, 50, 25.7366). This MATLAB intersection was found to match up with the intersection that was calculated using geometric analysis.

The third scenario seen in Figure 4-20 was analyzed using a different method than simple geometry, as the vectors did not truly intersect. The analysis graph can be seen in Figure 4-31

below, with distance from each vector being on the y-axis and the index, or position, along each vector being on the x-axis.

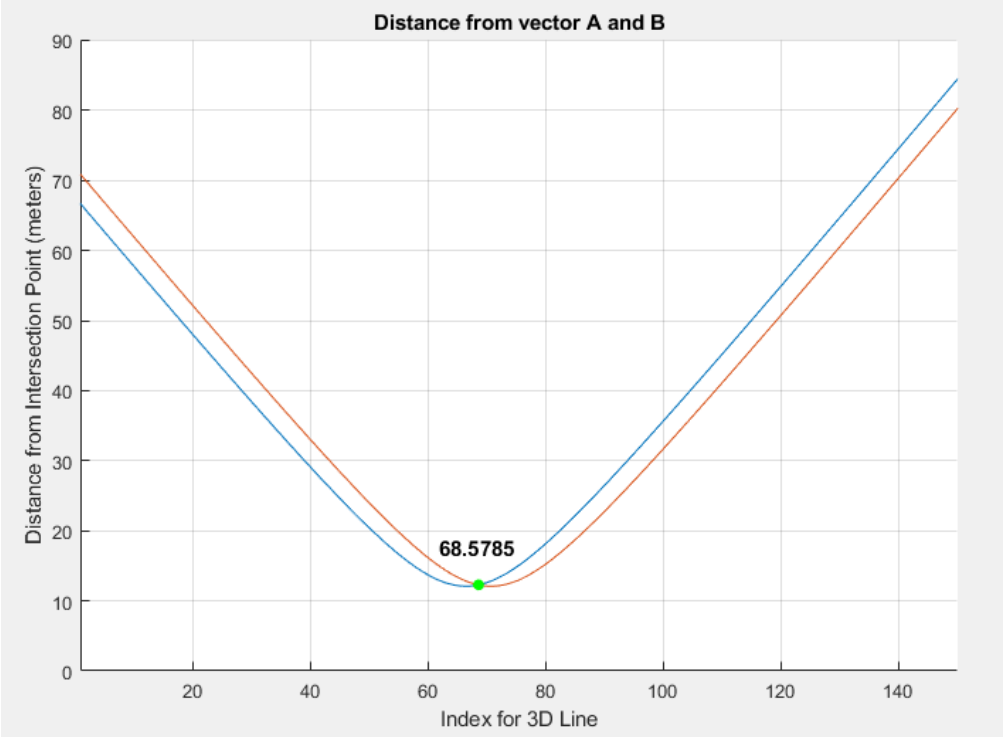

Figure 4-31: Scenario 3 Distance to Least Squares Intersection

The distance intersection seen in Figure 4-31 occurred at the point (45.5679, 45.5679, 23.4552) for Vector A and (51.5077, 48.4923, 0) for Vector B. The least-squares intersection found in MATLAB was expected to be directly in between these two points. Next, this scenario was analyzed using MATLAB. The resulting plot can be seen in Figure 4-32 and 4-33 below.

Figure 4-32: Scenario 3 MATLAB Zero Error Model View A

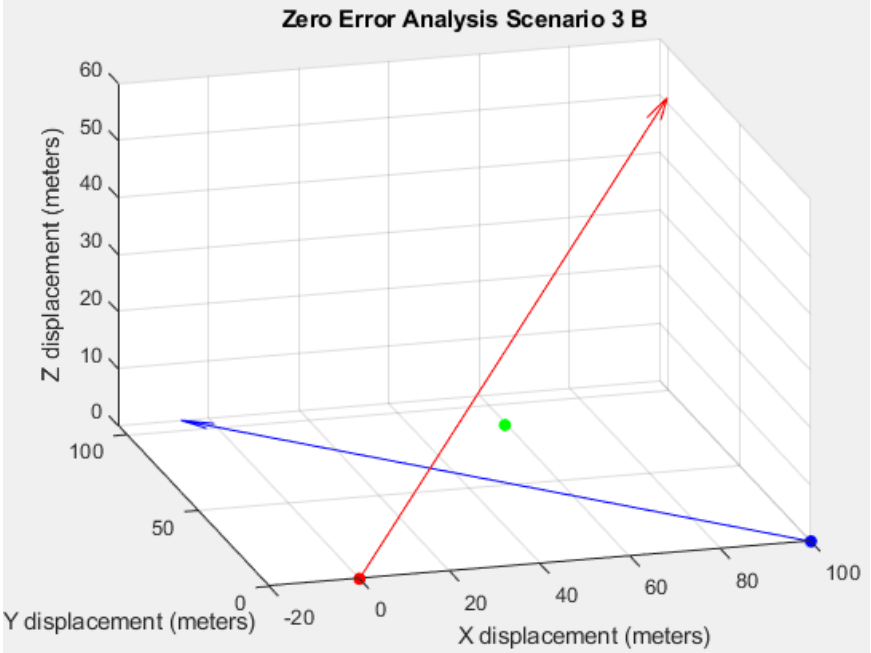

Figure 4-33: Scenario 3 MATLAB Zero Error Model View B

The resulting least-squares intersection using the MATLAB zero error model occurred at (47.076 m, 47.076 m, 11.363 m). This MATLAB intersection was found to occur in between the

intersection of Vector A and Vector B, which was the goal of the least squares approximation. Therefore, the least-squares approximation was found to be working correctly.

The fourth scenario seen in Figure 4-20 was analyzed using the same distance method as scenario three. The analysis graph can be seen in Figure 4-34 below.

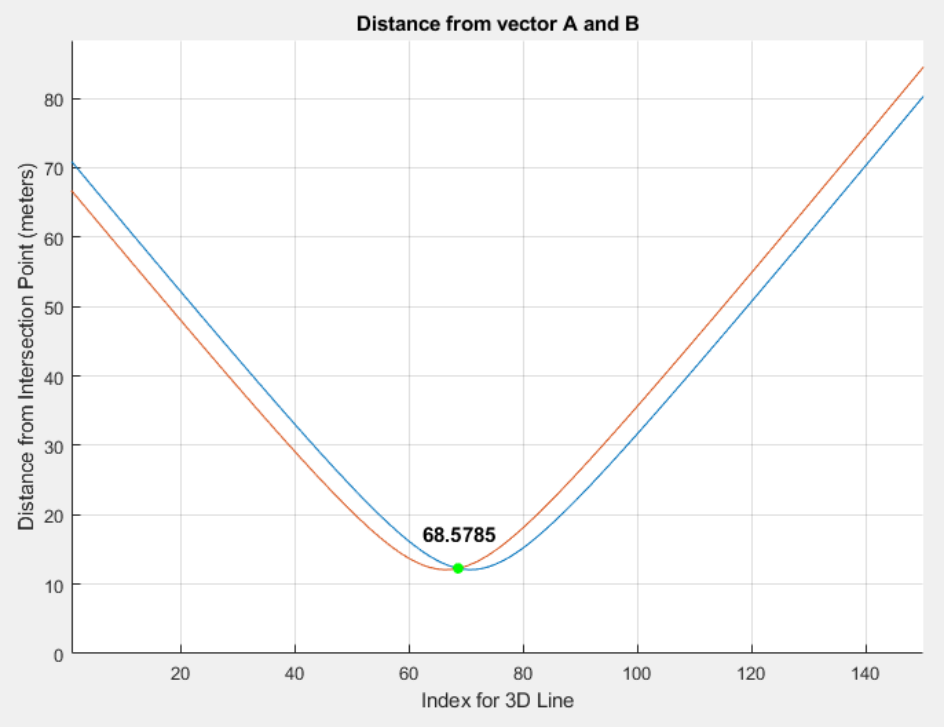

Figure 4-34: Scenario 4 Distance to Least Squares Intersection

The distance intersection for Vector A is (48.4923, 48.4923, 0) and the position for Vector B at this intersection was (54.4321, 45.5679, 23.4552). The least-squares intersection found in MATLAB was expected to be directly in between these two points. Next, the scenario was analyzed using MATLAB. The resulting plot can be seen in Figures 4-35 and 4-36 below.

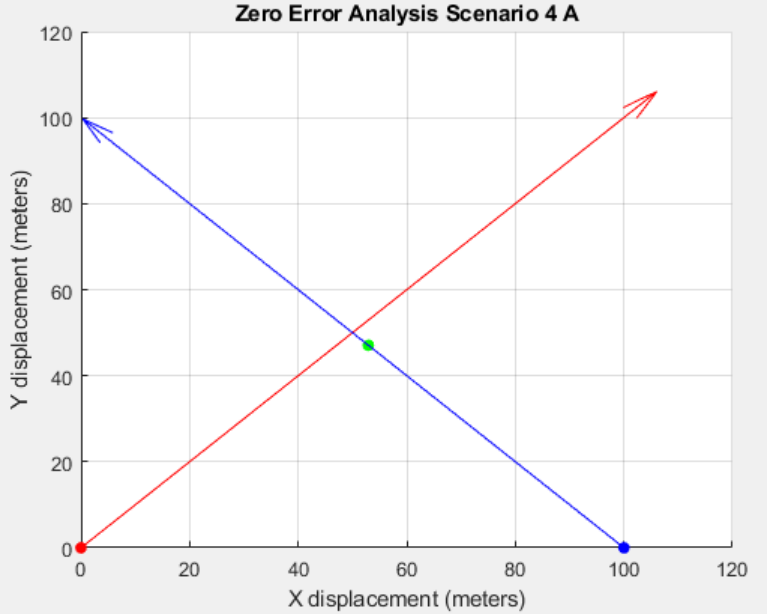

Figure 4-35: Scenario 4 MATLAB Zero Error Model View A

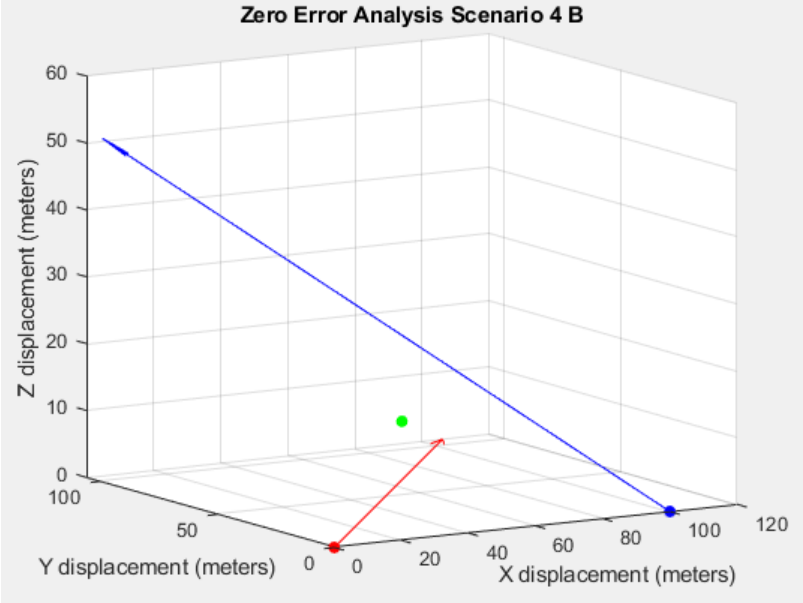

Figure 4-36: Scenario 4 MATLAB Zero Error Model View B

The resulting least-squares intersection, determined through calculations using the MATLAB algorithm, occurred at (52.9244, 47.0756, 11.3630). This MATLAB intersection was found to occur in between the intersection of Vector A and Vector B, which was the goal of the least-squares approximation. Therefore, the least-squares approximation was found to be working correctly.

The results from scenarios 5-12 can be seen in Appendix B. The thirteenth scenario seen in Figure 4-23 was analyzed using a similar method as scenario 3 and 4, as the vectors did not directly intersect. The analysis graph can be seen in Figure 4-37 below.

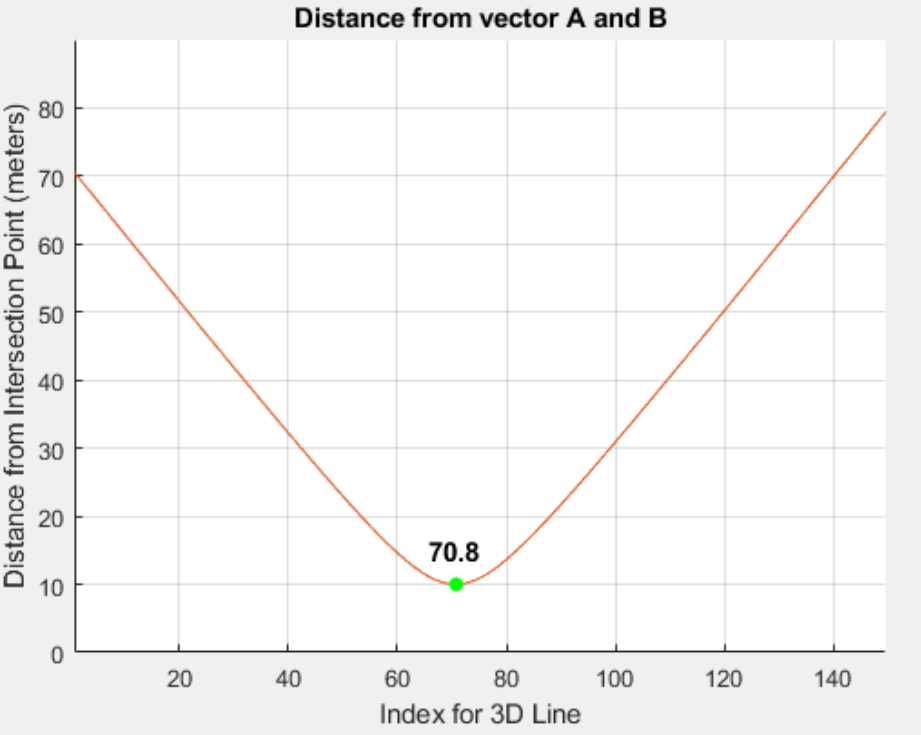

Figure 4-37: Scenario 13 Distance to Least Squares Intersection

The position for Vector A at this intersection was (50.0632, 50.0632, 0) and the position for Vector B at this intersection was (49.9368, 50.0632, 20.0000). The distance from vector A and vector B was the same in Figure 4-37 since both observer A and observer B's pitch angles were the same, therefore, the distance lines overlapped one another. Next, the scenario was analyzed using MATLAB. The resulting plot can be seen in Figures 4-38 and 4-39 below.

Figure 4-38: Scenario 13 MATLAB Zero Error Model View A

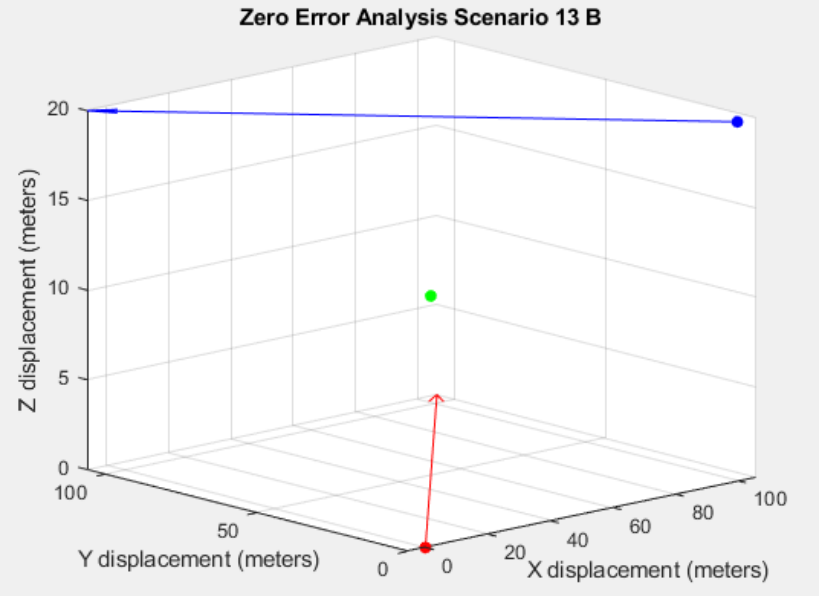

Figure 4-39: Scenario 13 MATLAB Zero Error Model View B

The resulting least-squares intersection, determined through calculations using the MATLAB algorithm, occurred at (50, 50, 10). This intersection point was found to very similar to the middle of each intersection point specified in the distance analysis.

The fourteenth scenario seen in Figure 4-23 was analyzed using a similar method to the third, fourth, and thirteenth scenarios, as the vectors did not intersect. The analysis graph can be seen in Figure 4-40 below.

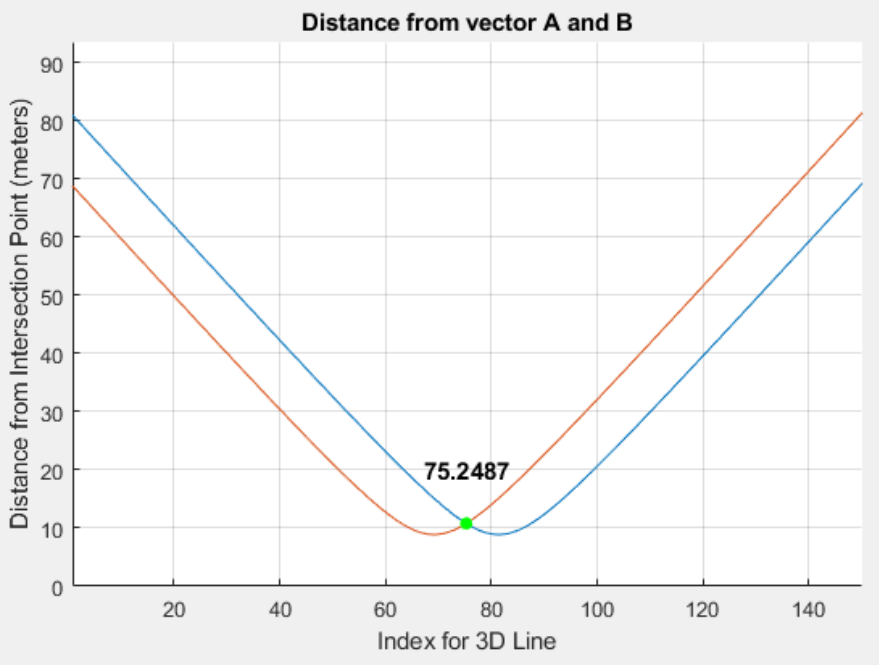

Figure 4-40: Scenario 14 Distance to Least Squares Intersection

The position for Vector A at this intersection was (50.0000, 50.0000, 25.7366) and the position for Vector B at this intersection was (50.0000, 50.0000, 45.7366). The least-squares intersection found in MATLAB was expected to be directly in between these two points. Next, the scenario was analyzed using MATLAB. The resulting plot can be seen in Figures 4-41 and 4-42 below.

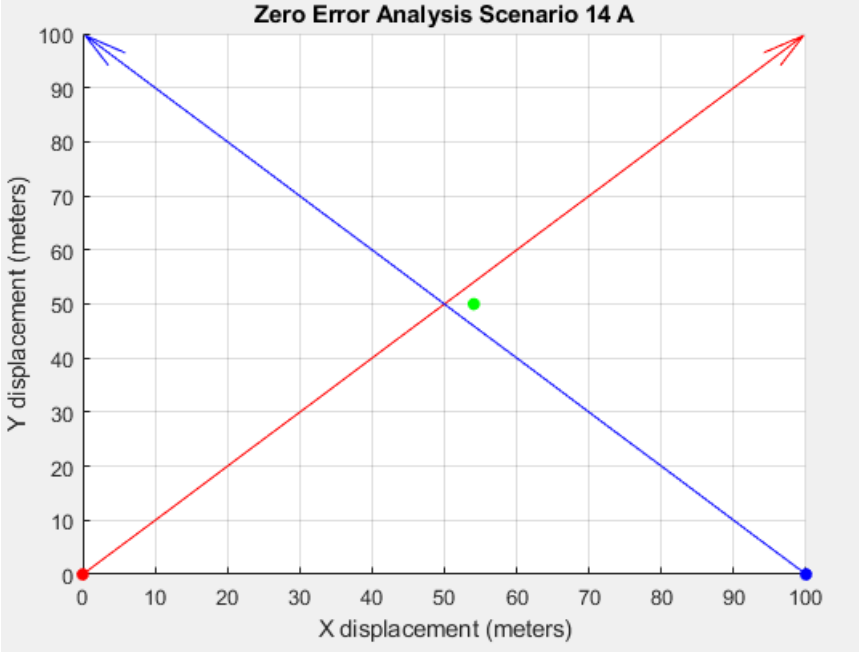

Figure 4-41: Scenario 14 MATLAB Zero Error Model View A

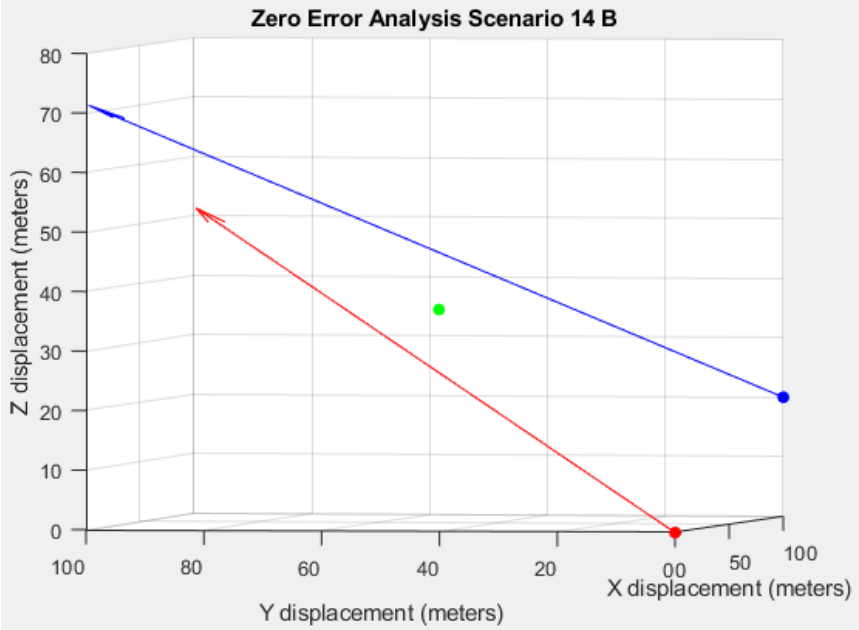

Figure 4-42: Scenario 14 MATLAB Zero Error Model View B

The resulting least-squares intersection, determined through calculations using the MATLAB algorithm, occurred at (54.0692, 50, 35.7366). This intersection was found to be very similar to the middle of each intersection point specified in the distance analysis. Therefore, the least-squares approximation produced the correct intersection point for this scenario. The results from

the remaining scenarios, 15 and 16, can be seen in Appendix B. After each scenario was analyzed and reproduced with the MATLAB algorithms, it was found that each intersection occurred in the correct position. Therefore, it was determined that the 3-D intersection worked correctly.

Discussion

Scenarios were analyzed using two distinct techniques: simple geometry was used to analyze the intersection when the vectors directly intersected and vector distance graphs were used to analyze the correct intersection position of the least-squares calculation when the vectors did not intersect. These two techniques were used to validate the output of the MATLAB 3D intersection algorithm. Direct intersection using simple geometry was used when the vectors directly intersected and was compared to the intersection using the MATLAB algorithm. These direct intersections were seen in scenarios 1, 5, and 9. The calculated intersection points directly matched up with the points found using the MATLAB algorithm, which showed that the intersection, without the approximation, worked correctly. Next, the least squares intersection calculation was shown to produce the correct intersection point by comparing the distance between two vectors and the calculated intersection. In this method, the correctness of the approximate intersection had to be determined after the least-squares method was calculated and an intersection was found. With this intersection point, the distance from intersection to each observer's vector was used to determine if the intersection point occurred at both the same distance and minimum distance from both vectors. This assertion was found to be true for every non-intersecting scenario that was tested. After every scenario was tested, it was determined that every 3-D intersection scenario computed with MATLAB matched up with the theoretical expected point of intersection. Due to this, it was concluded that the 3-D intersection was validated.

4.4 Discussion

In this chapter, the design of the extended Kalman Filter and the validation of the two-dimensional intersection and three-dimensional intersection were detailed. Designed to make an optimal orientation estimate, the EKF leveraged all sensor data provided by the magnetometer,

accelerometer, and gyroscope. With a built-in function determining the appropriate variance values for each sensor axis, the EKF created custom noise measurements for each individual test.

Second, the two-dimensional algorithm was validated by comparing ten analytical intersection results obtained via geometry and intersection algorithm outputs. In scenarios where there was no rounding, the error between the analytical and actual results was exactly zero. However, when inputs with four decimal places of accuracy were used, a maximum error of 1.404e-11 was observed. This significantly small error showed that the algorithm was validated.

Lastly, the three-dimensional algorithm was validated using simple geometry and vector distance graphs to compare to the MATLAB algorithm. When the observers' vectors had a direct intersection, and were not skew, the geometry analysis intersection was observed to directly match up with the intersection found from the MATLAB algorithm. When the observers' vectors did not have a direct intersection, vector distance graphs had to be used to determine the distance from the least-squares intersection to each observer's vector. The intersection point was found to be in the correct position if it was at both the same distance and minimum distance from both vectors. This was determined to be true for every non-intersecting scenario that was tested. After the validation analysis was complete, the two-dimensional intersection and three-dimensional intersection algorithms were found to be working correctly. Future work in the validation of both the 2-D and 3-D intersection algorithm could involve examining more edge cases.

Chapter 5: Algorithm Performance with Measurement Error

This chapter presents the methods, results, and discussion in characterizing the performance of the two-dimensional intersection algorithm and three-dimensional intersection algorithm.

5.1 Characterization of the Two-Dimensional Intersection Algorithm

The knowledge that the two-dimensional (2-D) intersection algorithm performed as expected with zero errors was valuable, however, in the real world there are measurement errors that cause the resulting intersection location estimation to be inaccurate. To model these errors, noise was added to each observer's orientation and location. Monte Carlo simulations, as shown in Section 2.4.2, were conducted to simulate the noisy environment that would be seen in the real world. This section presents several Monte Carlo simulations that were performed to characterize the performance of the 2-D intersection algorithm. These simulations were produced using MATLAB and the results were analyzed graphically as well as quantitatively.

5.1.1 Static Scenario Monte Carlo

Methods

The first Monte Carlo simulation was a static scenario where a simple noise distribution was applied to the location and orientation of two observers pointing at an object of interest. The same coordinate system as the zero-error model shown in Figure 5-1 was used for the two-dimensional Monte Carlo simulations. Within the two-dimensional plane, an indirect geolocation scenario was then established with two observers and a single object of interest. The truth values for this scenario are shown in Figure 5-1 below.

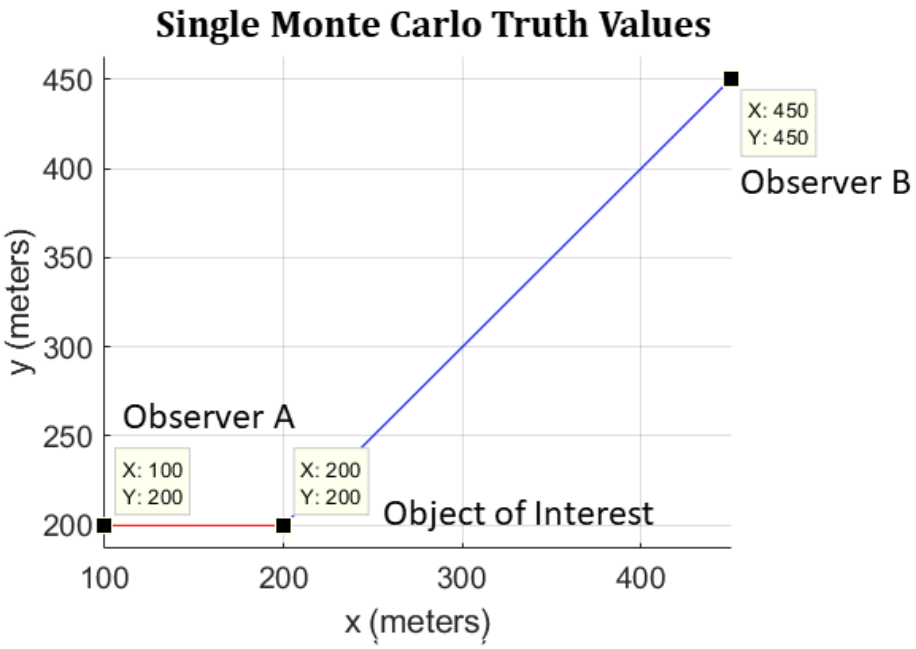

Figure 5-1: Truth Values for Monte Carlo Simulation

As shown in Figure 5-1, observer A was located at (100 m, 200 m) with an orientation of 90^{o} degrees with respect to the positive y-axis. Observer B was located at (450 m, 450 m) with an orientation of -135 degrees with respect to the positive y-axis. For the remainder of Section 5.1, orientation will be defined as the angle with respect to the positive y-axis. The object of interest was located at (200 m, 200 m). Each Monte Carlo simulation had six input variables (four for locations and two for orientations) that could have error applied to them in order to observe how that error propagates through the system and affects the intersection solution.

To introduce errors, independent Gaussian zero-mean noise distributions were applied to the position and orientation of both observers; these noise distributions were applied by adding their values directly to each observer's truth values. For all simulations conducted in this report, zero-mean Gaussian distributions were used as the error distributions. The position distribution has a standard deviation of 2.5 meters. The standard deviation value of 2.5 meters was chosen so that 95.4% of the errors would fall within ± 5 meters, the standard GPS error as presented in Section 2.4.2. Figure 5-2 shows this position error distribution for observer A. The orientation distribution has a standard deviation of 1 degrees, which was based on data from field tests. Figure 5-3 shows this distribution also for observer A. Observer B has the same error distribution for both location and orientation.

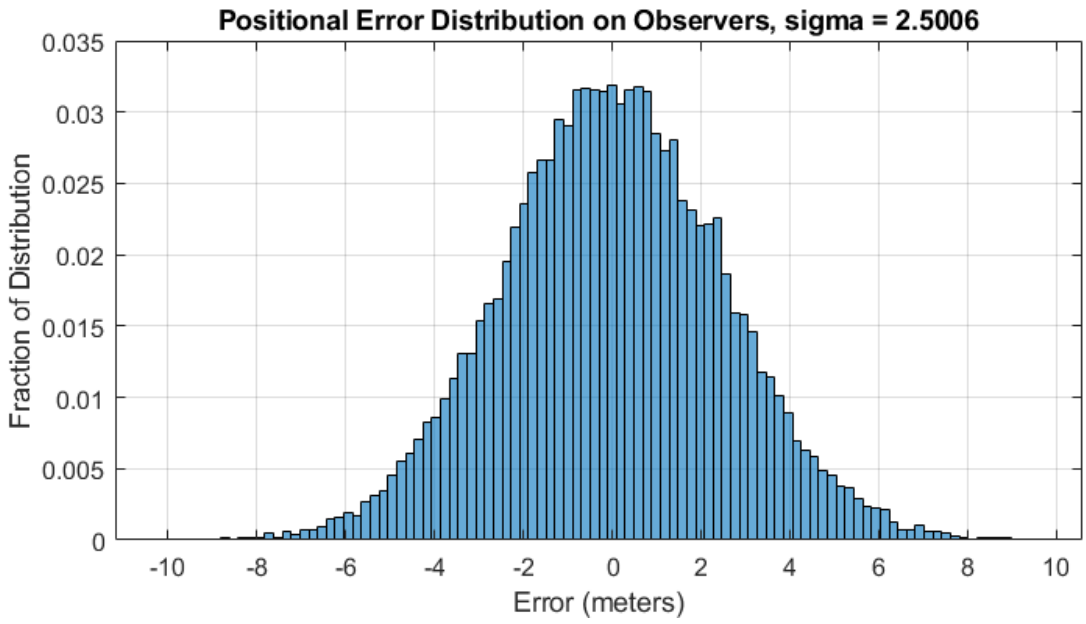

Figure 5-2: Distribution for Observer A Position Error

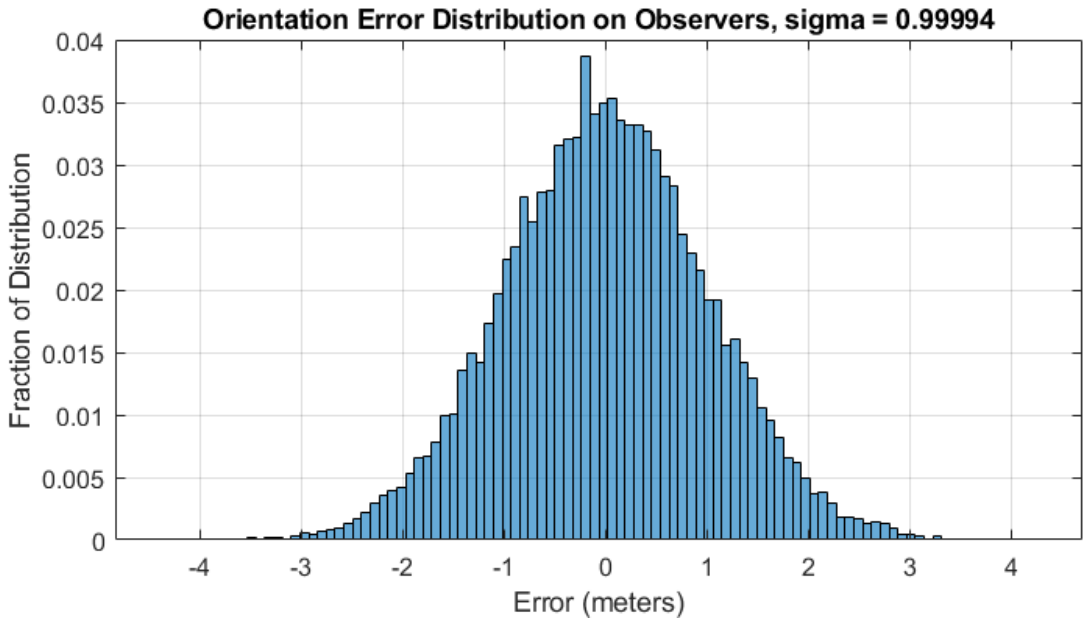

Figure 5-3: Distribution for Observer A Orientation Error

Then, the Monte Carlo simulation was run where the distribution was applied over a run of 25,000 iterations (n = 25,000). The only variable that was changed for each run of the simulation was that a different error instance was applied to each input.

Results

Figure 5-4 shows the system after one Monte Carlo simulation.

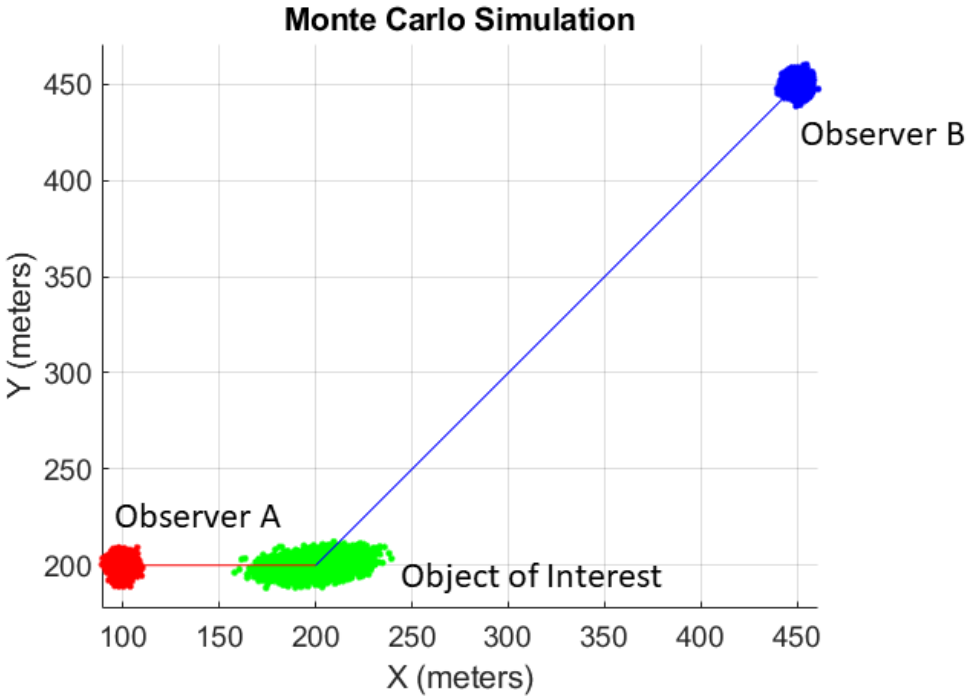

Figure 5-4: Single Scenario Monte Carlo Points

There were "clouds" of points on both observers and the intersection solutions. The position clouds for both observers were known to fit a Gaussian distribution in the x and y coordinates. The cloud of intersection points, however, was unknown and was characterized by several methods.

One of the characteristics of the intersection cloud was the standard deviation and mean of the cloud for both the x and y coordinates. Figure 5-5 shows the distribution of the x-coordinates and Figure 5-6 shows the distribution of the y-coordinates.

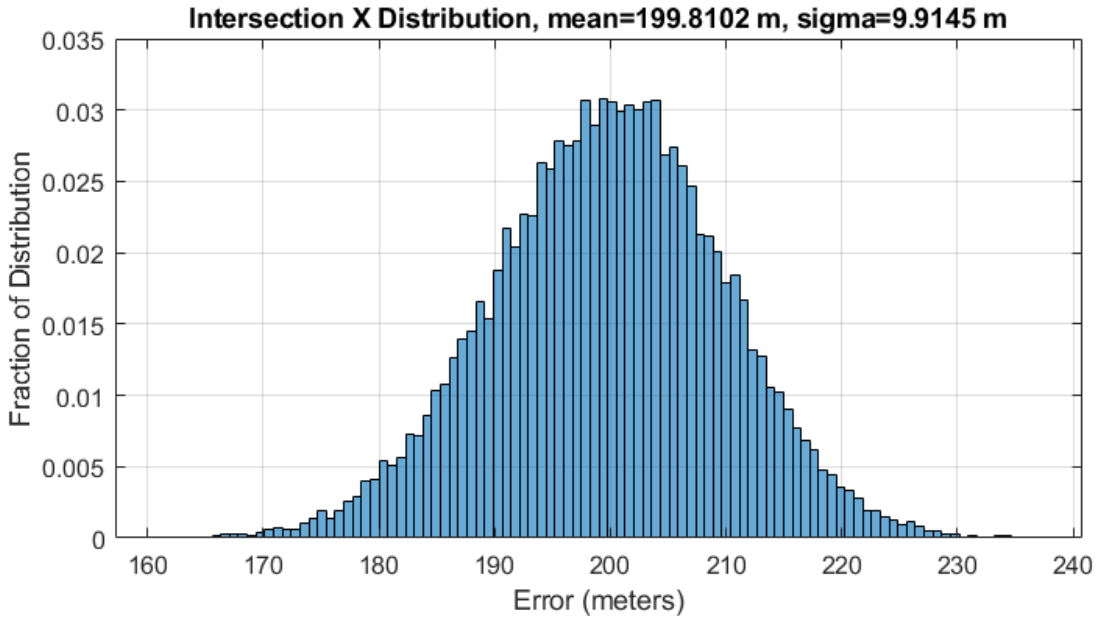

Figure 5-5: Intersection X Distribution

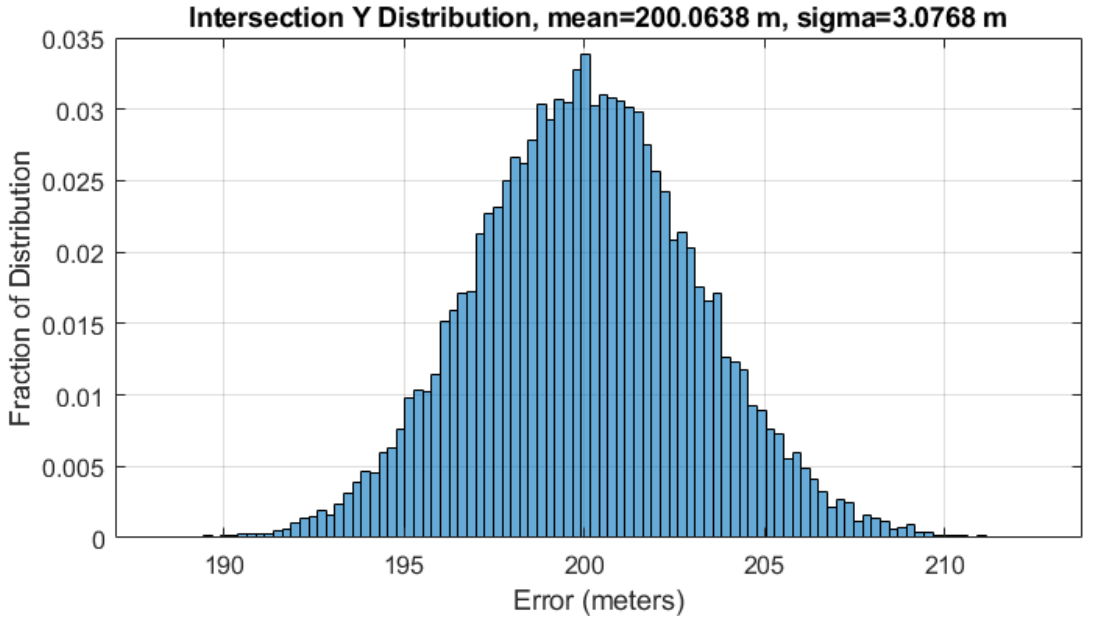

Figure 5-6: Intersection Y Distribution

Both distributions were centered on the truth value of (200 m, 200 m) with a standard deviation, θ_x,of 9.8897 m for the x-coordinate and a standard deviation, θ_y, of 3.0396 m for the y-coordinate. The overall standard deviation was then found by taking the square root of the sum of the squares of both the X and Y standard deviation values, which was 10.3463 m.

Another metric of the intersection cloud that was examined was the distribution of distances from each intersection point to the truth value, known as RMS error. For the truth position of (x_t, y_t) and the Monte Carlo intersection point (x_m, y_m) the distance, d, was found for all 25,000 intersection points in the Monte Carlo simulation by Equation 5-1.

$$d = \sqrt{(x_m - x_t)^2 + (y_m - y_t)^2} \qquad \text{(Eq. 5-1)}$$

The value, d, could also be called the RMS Error, and its distribution is shown in Figure 5-7.

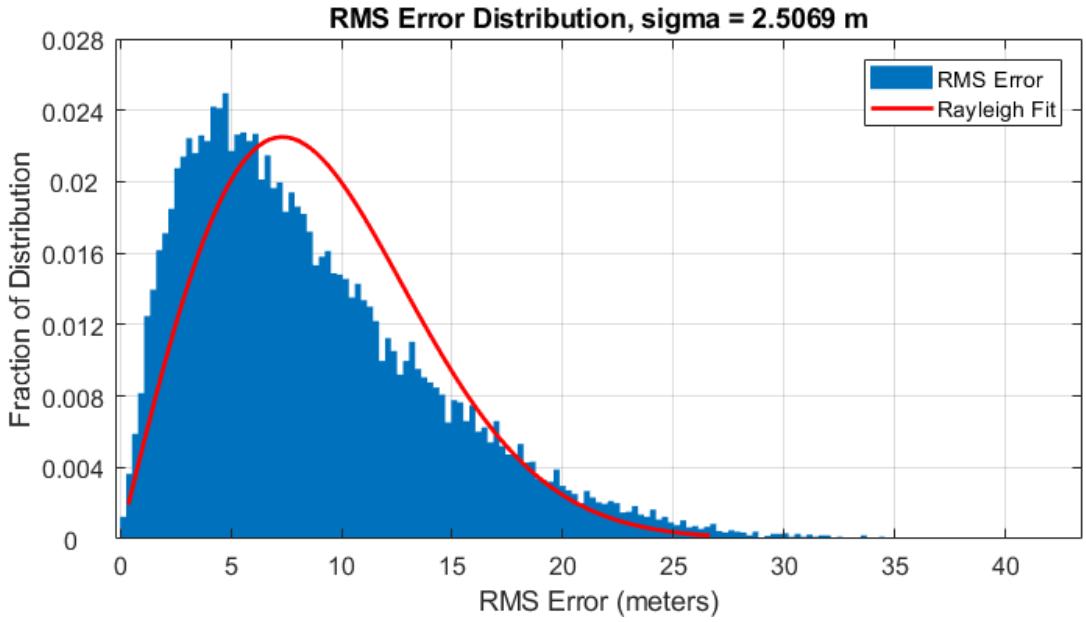

Figure 5-7: RMS Error Distribution for 2-D Static Scenario

This distribution looked to follow a Rayleigh distribution, which is a distribution of the positive square root of the sum of the square of two independent random variables. Both random variables are normally distributed with the same mean and standard deviation. The non-linear fit shown in Figure 5-7 is an attempt to fit the distribution to a Rayleigh distribution, but as shown in Figure 5-5 and 5-6 the X and Y distributions are not equal. Therefore, the distribution in Figure 5-7 did not follow a Rayleigh distribution well. When the same static scenario Monte Carlo was run, but with zero-error on the observer's orientation, the RMS error with Rayleigh fit shown in Figure 5-8 was obtained.

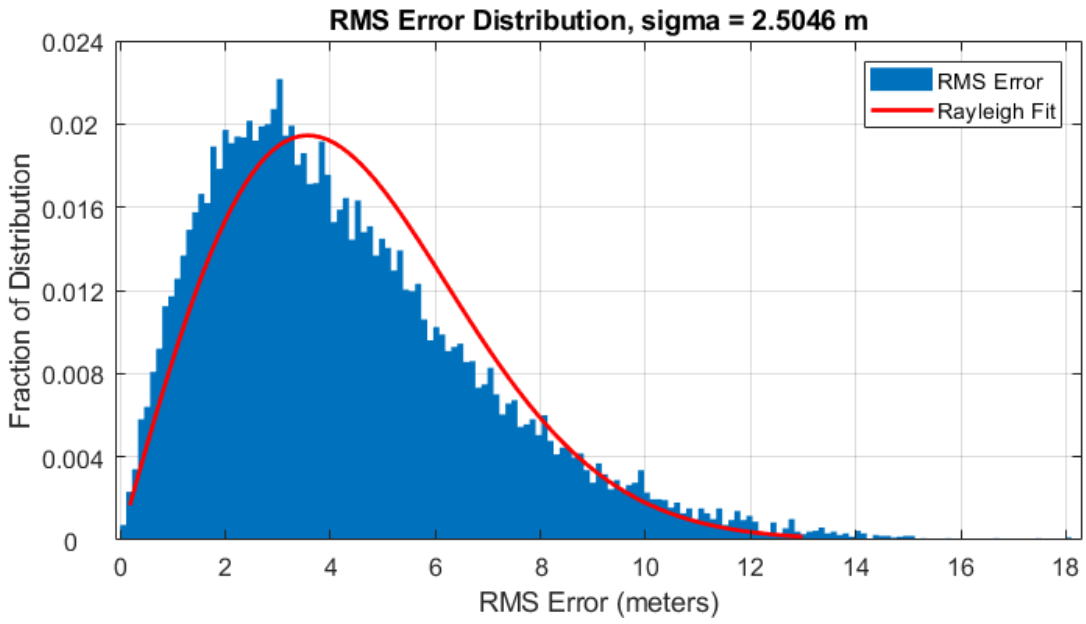

Figure 5-8: RMS Error Distribution with Zero Orientation Error

The fit was closer when the degrees of freedom were reduced, however, another requirement of the Rayleigh distribution is that the two random variables that create it are independently and identically Gaussian distributed. The two random variables in these simulations are the intersection point coordinates as shown in Equation 5-1 and could only be identically distributed when the angle between observers is 90^o. A scenario in which the angle between observers was 90^o and zero orientation error was run and the resulting RMS Error distribution is shown in Figure 5-9.

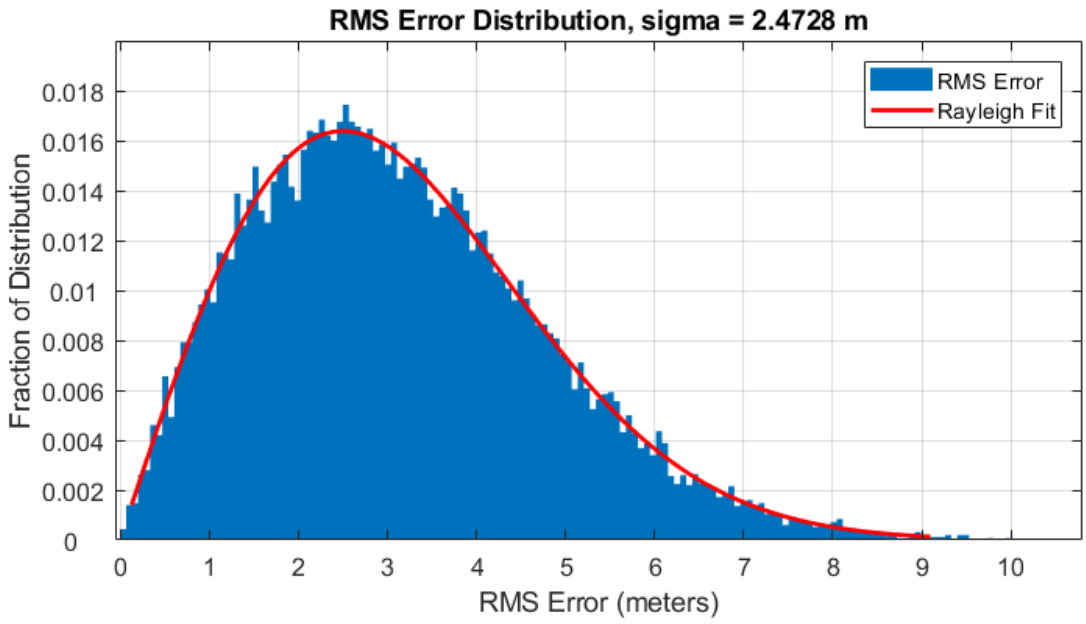

Figure 5-9: RMS Error Distribution with Zero Orientation Error and Observer Angle of 90°

The Rayleigh distribution fit shown in Figure 5-9, fits the RMS distribution.

Discussion

The results of the two-dimensional static scenario Monte Carlo demonstrated how the Monte Carlo results for the intersection algorithm could be characterized. There were two characterizations, standard deviation and RMS error. When both observers had Gaussian errors applied to them, the errors in the X and Y coordinates of the intersection cloud were also seen to be Gaussian, seen in Figures 5-5 and 5-6. When the RMS error distribution was found using Equation 5-1, it was not Gaussian, because the error distributions in the X and Y direction were not identical. In a specific scenario, however, when the angle between observers was 90° and there is no orientation error, the RMS error distribution was shown to follow a Rayleigh distribution.

5.1.2 Varying Distance Monte Carlo

Methods

In the previous Monte Carlo simulation, the observers' average location and orientation were kept constant. In the real world, however, these values vary and as such, simulations were conducted that varies the location of observer A. The series of simulations began with the same

scenario presented shown in Figure 5-1, however, observer A's x-coordinate is varied by 20 meters each Monte Carlo run until observer A is a total distance of 900 meters from the object of interest. For these series of simulations, distance refers to the absolute value of the x-coordinate difference between observer A and the object of interest. Observer B remains stationary throughout all simulations. Figure 5-10 shows the truth locations for the series of Monte Carlo simulations performed.

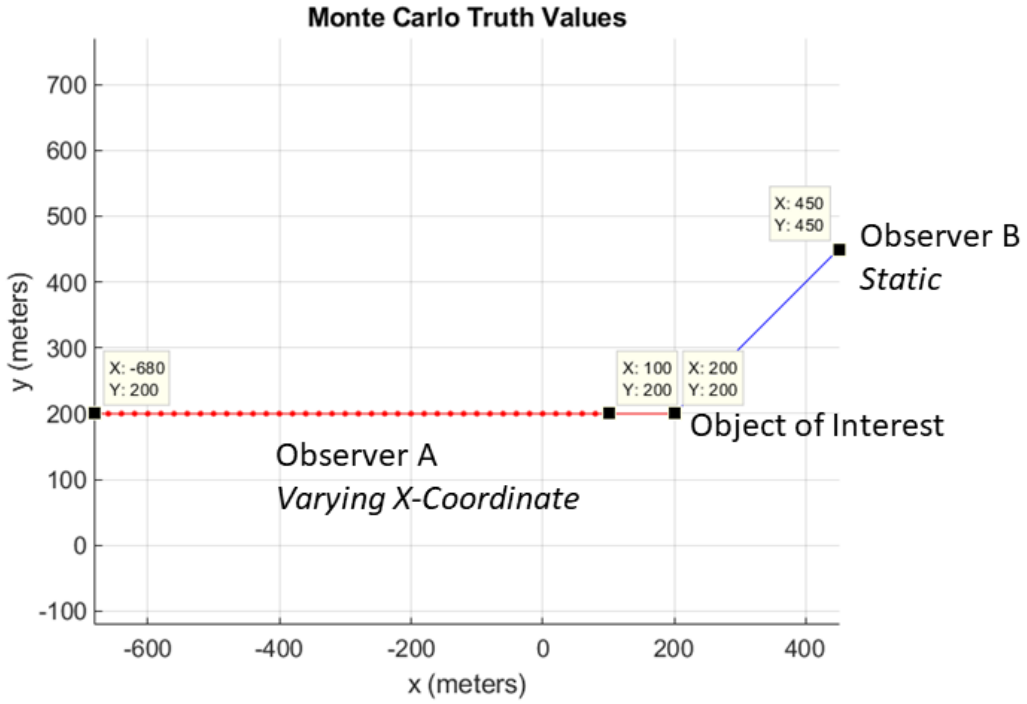

Figure 5-10: Varying Distance Monte Carlo Truth

Both observers' orientations remained constant throughout the simulations at 90° and 225° for A and B, respectively. This scenario formed an angle between the observers of 135°. Observer A's distance from the object of interest ranges from 100 m to 900 m in steps of 20 m. For both observers, the position standard deviation is 2.5 m and the orientation standard deviation is 1°. Shown in Figure 5-11, is the first of 20 Monte Carlo simulations with error clouds on the observers and intersection.

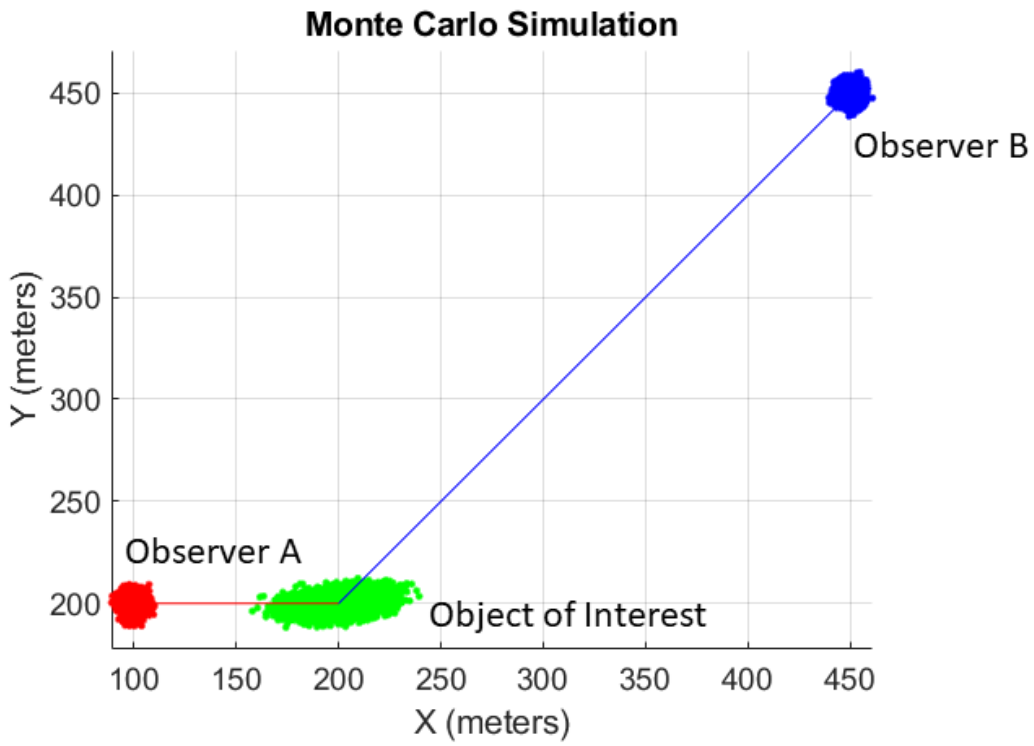

Figure 5-11: Monte Carlo Simulations Showing Observer and Intersection Distributions

The series of simulations shown in Figure 5-10 was then run multiple times, but with different error magnitudes on both observers' location and orientation. Then, a comparison was made between the effects of position error to orientation error. Figure 5-12 shows the different conditions in a table.

Run #	Position Standard Deviation (meters)	Orientation Standard Deviation (degrees)
1	2.5	0
2	2.5	0.5
3	2.5	0.7
4	2.5	1.0
5	2.5	1.2
6	2.5	1.5
7	2.5	2.0
8	0.001	0.0
9	0.001	1.0
10	0.001	1.2
11	0.001	1.5
12	0.001	2.0

Figure 5-12: Table of Variations of Position and Orientation Error Distributions

Results

The distance of observer A from the object of interest versus the mean RMS error of the intersection for the first series of simulations is shown in Figure 5-13. In this series, observer A was iteratively moved 20 m away from the object. Figure 5-13 showed the mean RMS error to be a function of the distance with a non-linear relationship.

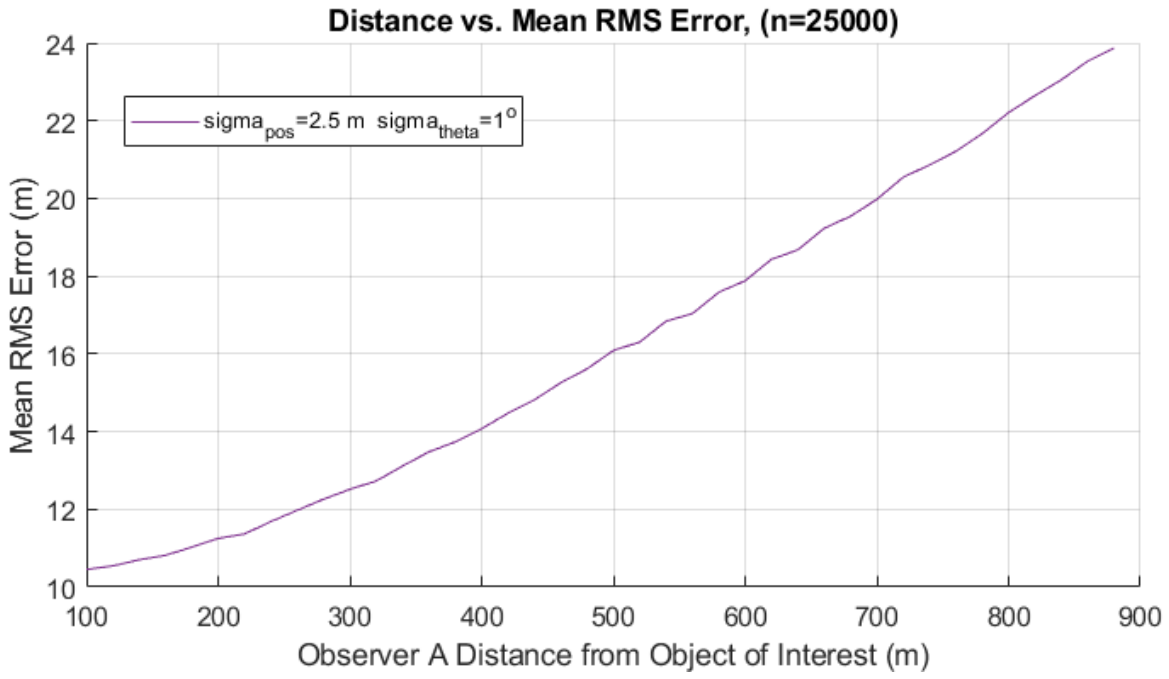

Figure 5-13: Distance vs. Mean RMS Error

This comparison was also conducted for each of the runs with different errors on each observer, described by the table in Figure 5-12. Figure 5-14 shows all of these runs.

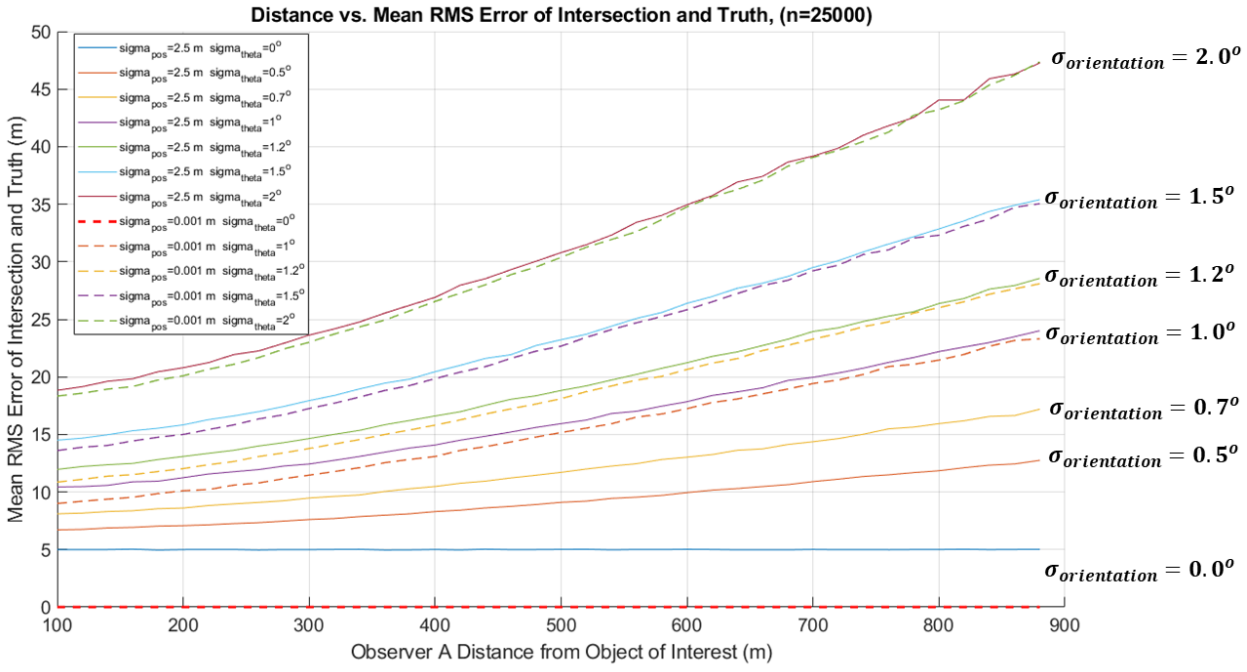

Figure 5-14: Distance vs. Mean RMS Error with Varying Observer Error

In Figure 5-14, position error seemed to act as a bias in the intersection mean RMS error. This bias is shown below in Figure 5-15.

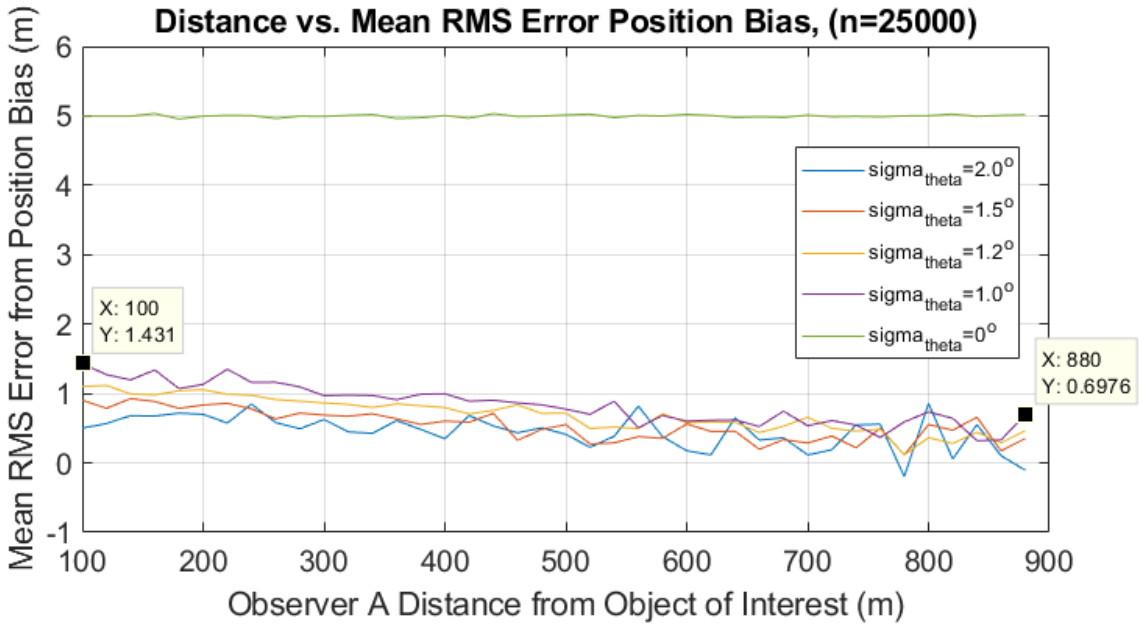

Figure 5-15: Mean RMS Error Position Bias

There are two distinct groupings of lines in Figure 5-15. The first consists of the bias for no orientation error. That bias remains constant over the entire run around 5 m, even when observer A is 900 m away from the object. The other four, however, show a clear trend downwards and the lines themselves are different magnitudes. The bias for an orientation error sigma of 1.0° falls from 1.431 m to 0.698 m. Also, generally the higher the standard deviation of the orientation error distribution, the lower the mean RMS error position bias. To explore this relationship more, the percent of each bias to the total mean RMS error was examined, shown in Figure 5-16.

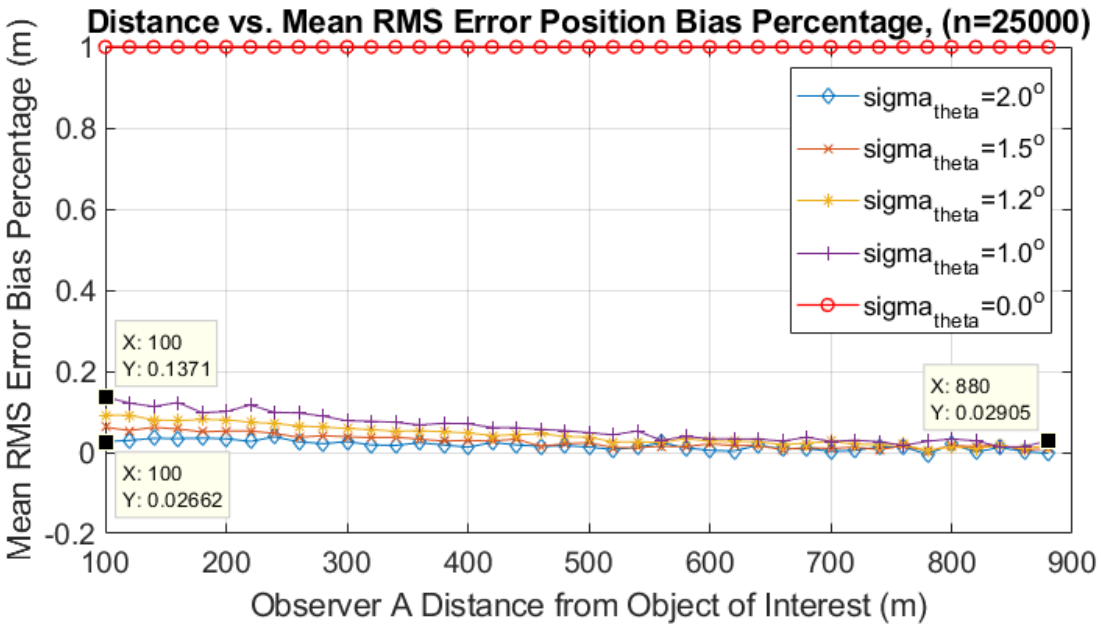

Figure 5-16: Mean RMS Error Position Bias Percentage

One of the lines in Figure 5-16 had no orientation error and it was clear that 100% of the bias was due to the position error. With orientation error ranging from 1.0° to 2.0°, it was similarly observed as in Figure 5-15 that the higher the standard deviation of the orientation error distribution, the lower the percentage the mean RMS error position bias was compared to the overall mean RMS error. Interestingly, the difference between each different orientation error becomes smaller as the distance from observer A to the object of interest increases.

This leads to two main findings. One, the effect that position errors have over longer distances diminishes due to the orientation error essentially overpowering them. Two, as the orientation error increases, the position error begins to have less of an overall effect on the intersection mean RMS error.

Discussion

The results of the varying distance Monte Carlo showed that when an observer is moved away from the object of interest, while keeping the angle between observers constant, the intersection mean RMS error increases. This finding was, however, dependent on there being an orientation error for the observer. As shown in Figure 5-14, the effect of error on position was characterized as a bias, mostly independent of how far away an observer was. When examined further, however, this bias actually decreased as distance increased due to the orientation error

essentially overpowering the effect of the position error. The effect of the orientation error was dependent on how far away the observer was. As an observer moved farther and farther away, the magnitude of the intersection mean RMS error increased due to the orientation error.

5.1.3 Varying Distance and Orientation Monte Carlo

Methods

In the previous series of Monte Carlo simulations, the observers' orientation was kept constant and only the x-coordinate difference between observer A and the object of interest was varied. In a new series of simulations, the distance and angle between observers was varied across each Monte Carlo run. The angle between observers is also referred to as the angle on the object in this report. Figure 5-17 shows the distance and angle between observer variables.

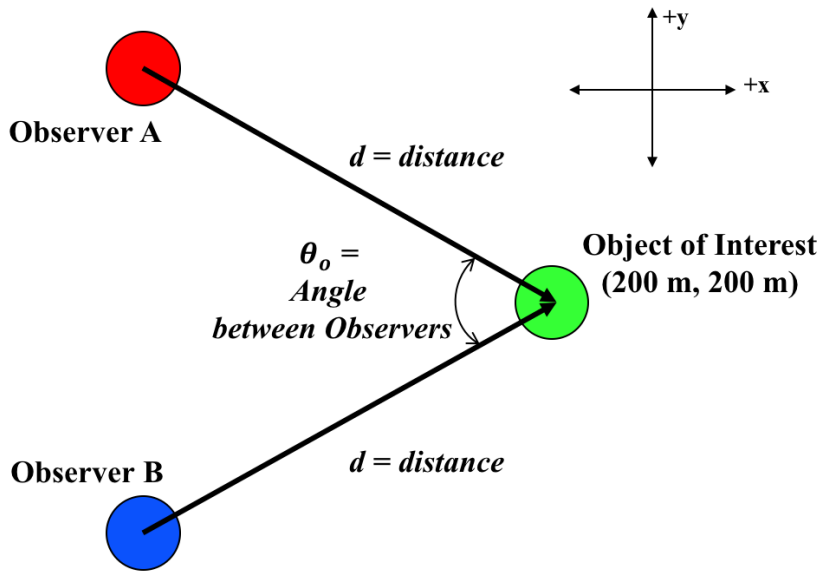

Figure 5-17: Distance and Orientation Monte Carlo Variables

By iterating over the distance and angle between observers as shown in Figure 5-17 a series of angles between observers, θ_o, at varying distances, d, from the object could be characterized. The simulation swept through θ_o from 160º to 20º in 5º increments beginning at distance of 25 m. Then, distance was increased by 25 m and the angles were swept through again. The object of interest was kept constant at (200 m, 200 m). This process was repeated until the distance was equal to 350 m. Shown in Appendix C is the table of distance and orientations used and Figure 5-18 shows all truth positions for both observers.

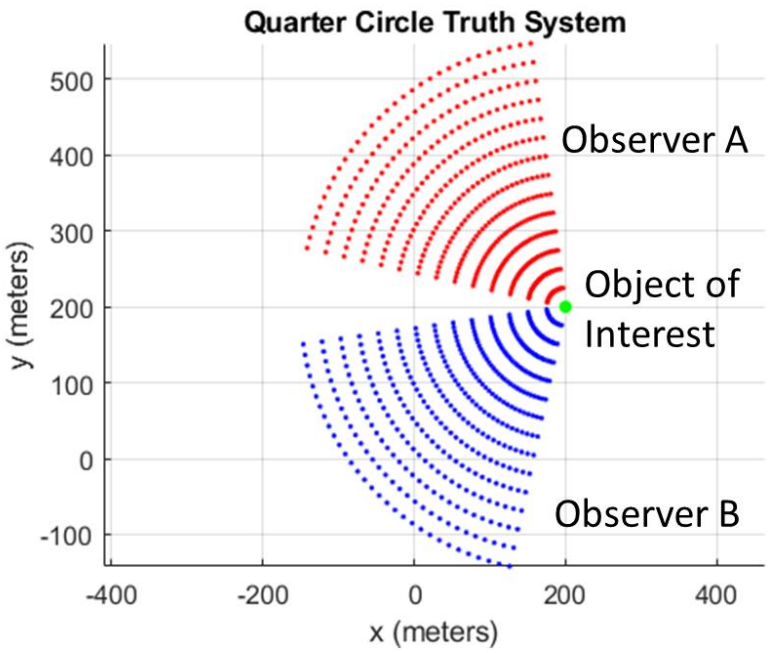

Figure 5-18: Distance and Orientation Monte Carlo Truth Locations

Each Monte Carlo simulation applied the same error distribution to both observers, so that the only change between Monte Carlo simulations was in the observer's position and the angle between the observers, not the errors applied to them. The position error distribution's standard deviation was 2.5 meters and the orientation error distributions standard deviation was 1°.

Results

From the intersection cloud, the mean RMS error from the truth intersection coordinate (200 m, 200 m) was found for each Monte Carlo run. The mean RMS error was then plotted against the angle between observers which is shown in Figure 5-19 to explore if an optimal angle existed where the mean RMS error was minimized. Each line in Figure 5-19 represents a sweep through each angle between observers at a different distance from the object.

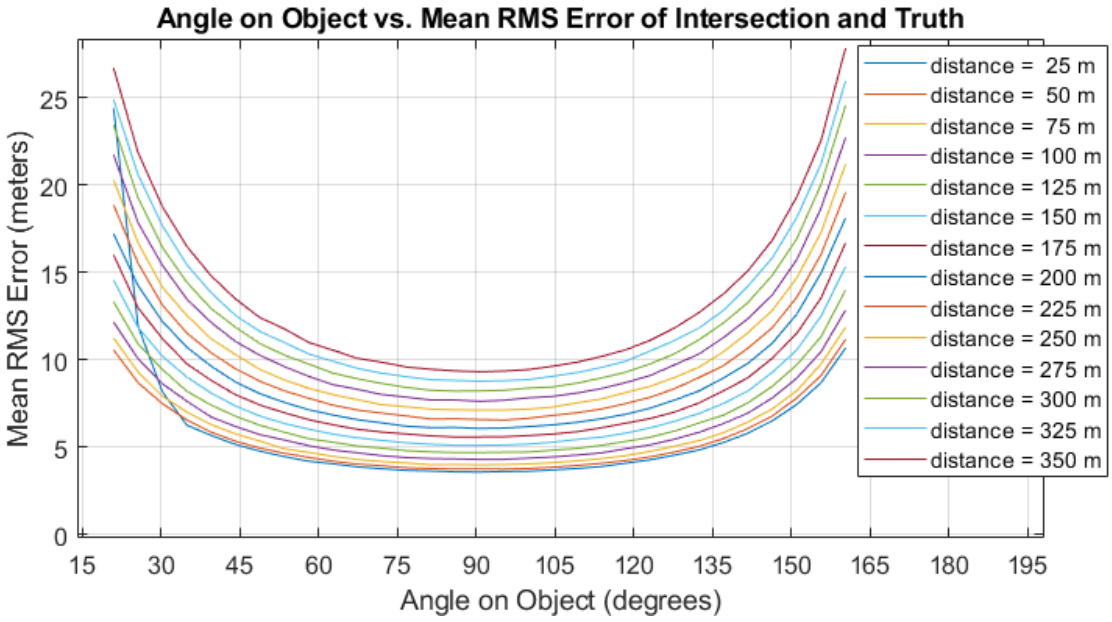

Figure 5-19: Mean RMS Error with Orientation and Position Error. The bottom-most line corresponds to a distance, *d*, of 25 m. Each line higher is 25 m more in distance, with the top-most line showing 350 m.

The data followed a U-formation with a valley that is centered at 90^{o}, which showed that for this series of simulations, when a 90^{o} angle was formed on the object by the observers, there was in general the least amount of error on the intersection. The mean RMS error exponentially increased when the angle between observers became more acute and more obtuse. The increase from the minimum mean RMS error value for all lines at 90^{o} to the mean RMS error value at 75^{o} and 105^{o} for all lines only ranged from 2% - 3%. At the angles of 60^{o} and 120^{o}, there was a 17% - 18% increase from the minimum mean RMS error for all lines. Overall, as the distance of the observers from the object increased, the magnitude of the mean RMS error increased.

To compare the individual effects of position and orientation, the same series of simulations was repeated twice with the same conditions, but one had only position error and the other had only orientation error. Figure 5-20 shows the intersection mean RMS error results with only position error and Figure 5-21 shows the intersection mean RMS error results with only orientation error.

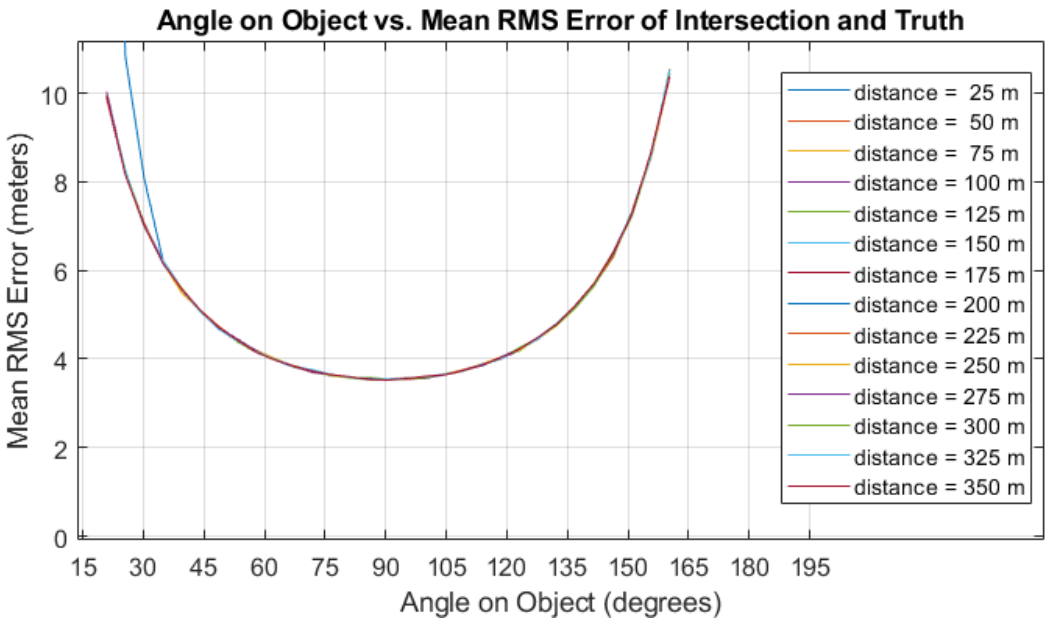

Figure 5-20: Mean RMS Error with Only Position Error

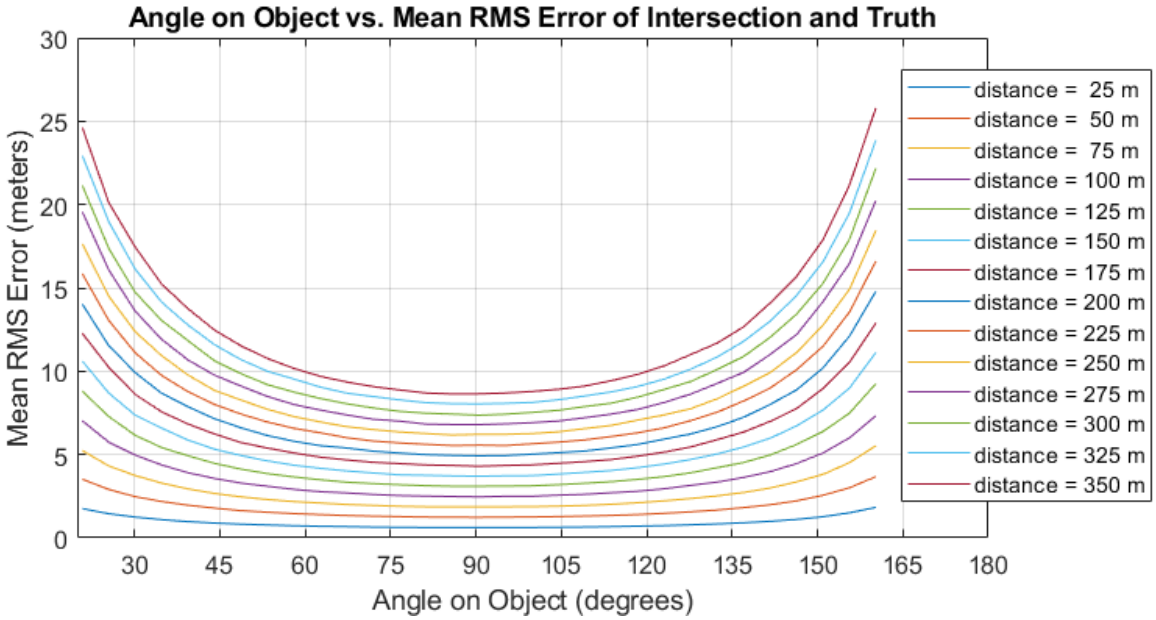

Figure 5-21: Mean RMS Error with Only Orientation Error

The results in Figures 5-20 and 5-21 both follow the same trend that an angle between observers of 90^{o} produces the least mean RMS error. With only position error, however, the same mean RMS error was seen at all distances. With only position error, however, the same mean RMS error was seen at all distances and is essentially a lower bound. With only orientation error, the

overall magnitude of the mean RMS error increased as the distance increased. There was, however, an exception at acute angles at the least distance of 25 m, where the mean RMS error spiked. To compare the effects the position lower bound on the intersection, a single only position error mean RMS error line from Figure 5-20 was plotted with the only orientation error lines in Figure 5-19, which is shown in Figure 5-22.

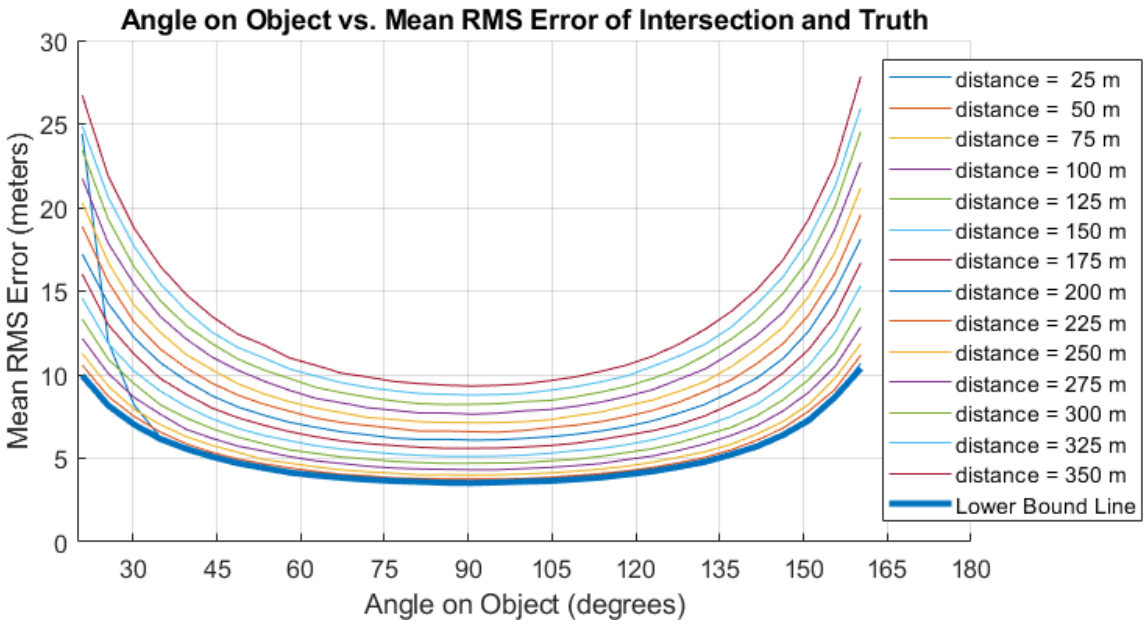

Figure 5-22: Mean RMS Error of Only Position Error and Only Orientation Error

The only position error line was very close the distance of 25 m line. It was also found that at a distance of 250 m, the mean RMS error with position and orientation errors is double that of the lower bound shown in Figure 5-22. This finding shows that at a distance of 250 m, the additional effect of orientation error begins to have more of an effect on the intersection mean RMS error than position error.

Discussion

The results of the distance and orientation Monte Carlo simulation showed that the optimal angle between observers to minimize intersection mean RMS error is 90º. As the angle between observers became more acute or obtuse the mean RMS error increased exponentially. The results in Figure 5-19 show that even with an angle between observers within 75º to 105º, there is only an increase in 2% - 3% of mean RMS error. Therefore, in a real-world application where verifying or obtaining a 90º angle between observers might be difficult, there is only a

small increase in mean RMS error at slightly larger or smaller angles. The position error was essentially a bias that kept the same trend for all distances. The orientation error, however, affected the intersection mean RMS error based on the distance of the observers from the object. At a distance of 250 m the intersection mean RMS error due to only orientation error was approximately equal to the intersection mean RMS error due to only position error at all distances.

5.1.4 Covariance Mapping, Determining a Closed-Form Solution

Methods

One of the objectives of this report was the ability to map the uncertainty of the indirect geolocation result. Therefore, a closed-form solution was found to determine the covariance of the intersection. The following is the derivation of the closed-form solution for mapping the covariance of the location and orientation to the covariance of the intersection solution.

In order to map the uncertainty, the covariance of each device's location and orientation must be known. This covariance can be represented by the covariance matrix seen in Equation 5-2, which shows each variance ($\sigma_{x,x}^2, \sigma_{y,y}^2, \sigma_{z,z}^2$) and covariance ($\sigma_{y,x}^2, \sigma_{z,x}^2, \sigma_{x,y}^2, \sigma_{z,y}^2, \sigma_{x,z}^2, \sigma_{y,z}^2$). The subscripts x, y, and z represent the location coordinates in three-dimensional space.

$$Location\ Uncertainty = \begin{bmatrix} \sigma_{x,x}^2 & \sigma_{y,x}^2 & \sigma_{z,x}^2 \\ \sigma_{x,y}^2 & \sigma_{y,y}^2 & \sigma_{z,y}^2 \\ \sigma_{x,z}^2 & \sigma_{y,z}^2 & \sigma_{z,z}^2 \end{bmatrix} \quad \text{(Eq. 5-2)}$$

Conveniently, the orientation covariance is modeled during each cycle of the extended Kalman Filter (EKF). Thus, this orientation uncertainty can be represented by the covariance matrix seen in Equation 5-3**,** where *r* represents roll, *p* represents pitch, and ψ represents yaw.

$$Orientation\ Uncertainty = \begin{bmatrix} \sigma_{r,r}^2 & \sigma_{p,r}^2 & \sigma_{\psi,r}^2 \\ \sigma_{r,p}^2 & \sigma_{p,p}^2 & \sigma_{\psi,p}^2 \\ \sigma_{r,\psi}^2 & \sigma_{p,\psi}^2 & \sigma_{\psi,\psi}^2 \end{bmatrix} \quad \text{(Eq. 5-3)}$$

The main diagonal of the covariance's in Equation 5-3 represents either the variance of each location or variance of each orientation angle with respect to itself. In the two-dimensional intersection method proposed in Section 2.1.2, the 2-D location variances represent the x and y

uncertainties while the orientation variance $\sigma_{z,z}^2$ represents the yaw uncertainty. When combined, these variances represent the overall 2-D system uncertainty for one observer. In the intersection method, there must be two observers, so the overall system covariance matrix contains the location and orientation uncertainties of two observers. The combined uncertainty matrix can be seen in Equation 5-4, where A represents observer one and B represents observer two. The zero components of the matrix occur when the two variables are independent from each other.

$$System\ Covariance = \begin{bmatrix} \sigma_{A\,x,x}^2 & \sigma_{A\,y,x}^2 & 0 & 0 & 0 & 0 \\ \sigma_{A\,x,y}^2 & \sigma_{A\,y,y}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{B\,x,x}^2 & \sigma_{B\,y,x}^2 & 0 & 0 \\ 0 & 0 & \sigma_{B\,x,y}^2 & \sigma_{A\,y,y}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{A\,\psi,\psi}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{B\,\psi,\psi}^2 \end{bmatrix} \quad \text{(Eq. 5-4)}$$

Once the system covariance is found, it must be transformed from each observer's location to the intersection point. The Jacobian of the intersection function can be used to transform these two points to the intersecting point.

A Jacobian matrix is useful to linearly transform and approximate a function to a new point. The Jacobian of this intersection function was found by taking the partial derivative of the intersection function with respect to the variables in the 2-D location and orientation. The configuration of the Jacobian matrix of this function can be seen in Equation 5-5, where A is observer one, B is observer two, X is the x-axis location component, Y is the y-axis location component, ψ is the yaw orientation.

$$Jacobian = \begin{bmatrix} \frac{\partial_{Xi}}{\partial_{AX}} & \frac{\partial_{Yi}}{\partial_{AX}} \\ \frac{\partial_{Xi}}{\partial_{AY}} & \frac{\partial_{Yi}}{\partial_{AY}} \\ \frac{\partial_{Xi}}{\partial_{BX}} & \frac{\partial_{Yi}}{\partial_{BX}} \\ \frac{\partial_{Xi}}{\partial_{BY}} & \frac{\partial_{Yi}}{\partial_{BY}} \\ \frac{\partial_{Xi}}{\partial_{A\psi}} & \frac{\partial_{Yi}}{\partial_{A\psi}} \\ \frac{\partial_{Xi}}{\partial_{B\psi}} & \frac{\partial_{Yi}}{\partial_{B\psi}} \end{bmatrix} \quad \text{(Eq. 5-5)}$$

In order to transform the system covariance into the intersection covariance, the system covariance must be multiplied by the Jacobian. The matrix multiplication property can be used to

perform this transformation, which is shown in Equation 5-6. A is the original matrix and B is the transform matrix.

$$(A \cdot B) = B \cdot A \cdot B^T \quad \text{(Eq. 5-6)}$$

This property can be implemented into the intersection covariance transformation. The final intersection covariance equation can be seen in Equation 5-7, where the transpose of the Jacobian is matrix B for the matrix multiplication property.

$$Intersection\ Covariance = Jacobian^T \times System\ Covariance \times Jacobian \quad \text{(Eq. 5-7)}$$

The resulting intersection covariance will be in the form of a 2x2 matrix that represents an ellipse of location uncertainty. The ellipse can then be mapped directly to the intersection point to represent the uncertainty of the indirect geolocation intersection point. This closed-form solution was then compared to an actual system result.

Results

To characterize the performance of the closed-form solution, the solution was compared to the first actual result of the distance and orientation Monte Carlo, shown in Section 5.1.3. The covariance of the intersection was obtained using the closed-form solution and converted to standard deviation, so that this expected standard deviation from the closed-form solution could be compared to the standard deviation of the actual intersection result. Figure 5-23, shows the standard deviation from the closed-form solution.

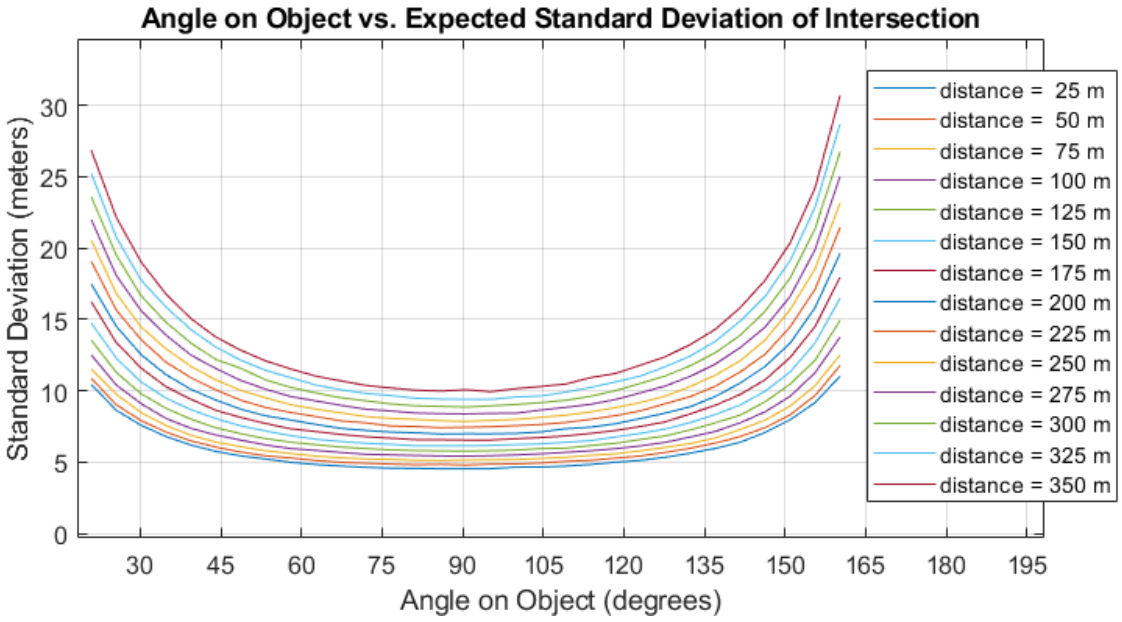

Figure 5-23: Expected Standard Deviation for Distance and Orientation Simulation

Then, these data were subtracted from the standard deviation of the actual result to obtain Figure 5-24.

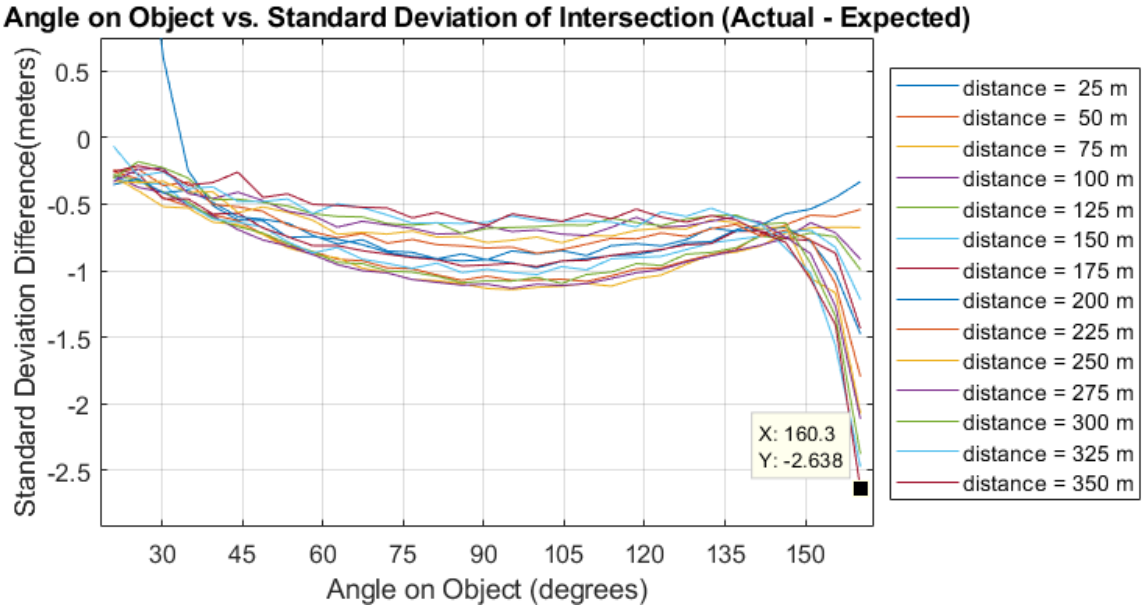

Figure 5-24: Expected Standard Deviation for Distance and Orientation Simulation

The difference between the expected and actual standard deviation error ranges from -0.06388 to -2.638 m, excluding the distance of 25 m simulation.

Discussion

When the expected standard deviation error shown in Figure 5-24 was subtracted from the actual standard deviation error, there was a range of difference between the two. Overall, the closed-form solution was fairly accurate, but as the angle became very obtuse, there were significant standard deviation differences up to -2.638 m. There was also a significant difference when the distance was 25 m and the angle was very acute. The ability to determine the two-dimensional intersection uncertainty from the observer's uncertainty was very useful because the Kalman Filter estimates the orientation and location uncertainty on each iteration. Therefore, each iteration could then estimate the uncertainty of each solution found through the two-dimensional intersection algorithm. A real-time system would be able to take advantage of this capability by knowing its estimated accuracy in real-time.

5.2 Characterization of the Three-Dimensional Intersection Algorithm

The three-dimensional (3-D) intersection model created in MATLAB was shown to work correctly in the zero error model seen in Section 4.1.1. In a real world implementation, there are errors within sensor measurements, as described in Section 2.1.1, and these errors produce inaccuracies within the final intersection location estimate. To observe and quantify how these errors affect the intersection location estimate Monte-Carlo simulations, as described in Section 2.4.1, were performed with the 3-D intersection algorithm. The three-dimensional intersection simulations were produced using MATLAB and the results were characterized graphically as well as quantitatively.

5.2.1 Static Scenario Monte Carlo

Methods

To begin, a Monte Carlo simulation was ran with 25,000 iterations using a few of the scenario configurations from the validation and zero-error model in Section 4.3 and errors were applied to the position and orientation of both observers. The X and Y position had a standard deviation of 2.5 m, the Z position had a standard deviation of 5 m, and the yaw and pitch both had a standard deviation of 2.5 degrees. These standard deviation were for zero-mean Gaussian

distributions. These standard deviation values were modeled after typical static sensor noise and uncertainties that were observed during a real world test. This simulation produced a "cloud" of points in the form of an ellipsoid that showed the observers' location uncertainty and a cloud of points that occurred at the intersection of the vectors. This cloud of points increased in area as the noise inputs became larger. The specific scenarios that were be analyzed in this section are similar to the zero-error model scenarios. The simulation scenarios are listed in Figure 5-25 below.

	Observer A			Observer B		
Scenario	X,Y,Z Position	Yaw Angle	Pitch Angle	X,Y,Z Position	Yaw Angle	Pitch Angle
1	(0,0,0)	45	0	(100,0,0)	-45	0
2	(10,10,10)	30	0	(100,10,10)	0	0
3	(10,10,10)	60	0	(100,10,10)	0	0
4	(0,0,0)	45	0	(100,0,20)	-45	0

Figure 5-25: Static Scenario Simulations

These scenarios were chosen specifically in order to observe the distribution and shape of the location uncertainty. The intersection distribution was represented by the standard deviation from the true intersection point along each axis, known as the RMS error value. As the RMS error became larger, the larger the distribution along the axis became. Ultimately, this section showed the performance of the system with real world noise on the measurements.

Results

A constant noise was applied to the position and orientation of both observer's measurements to observe the effect of real-world noise to the intersection location shape. There were four separate scenario configurations to observe the intersection shape. The first simulation configuration consisted of truth values with noise added afterwards. The configuration of this simulation is: starting point of (0 m, 0 m, 0 m), 45 degree yaw angle, and 0 degree pitch angle for observer A, and a starting point of (100, 0, 0), -45 degree yaw angle, and 0 degree pitch angle for observer B. The resulting intersection cloud can be seen in Figure 5-26 and 5-27 below.

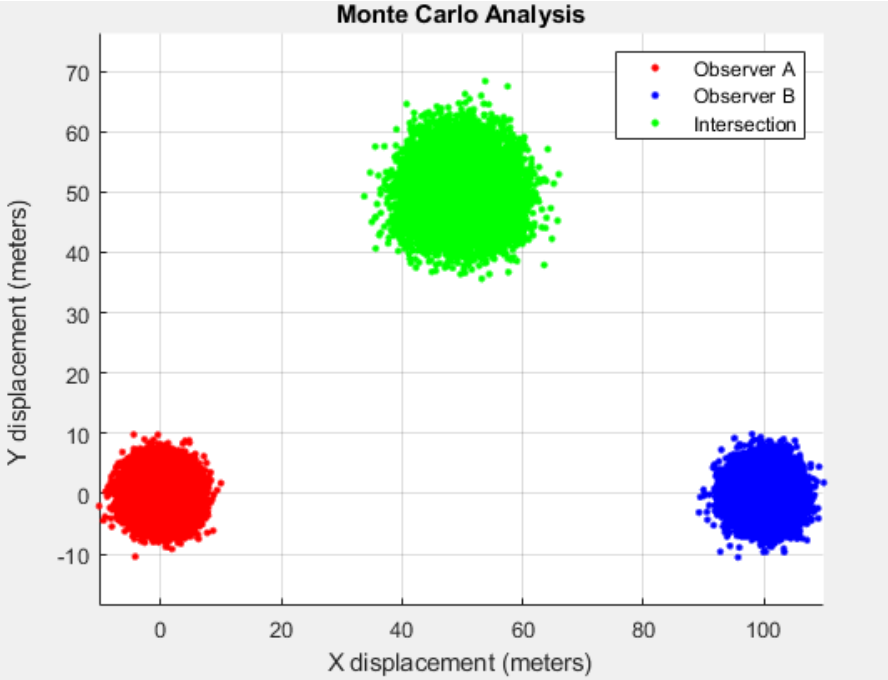

Figure 5-26: Scenario 1 Monte Carlo View A

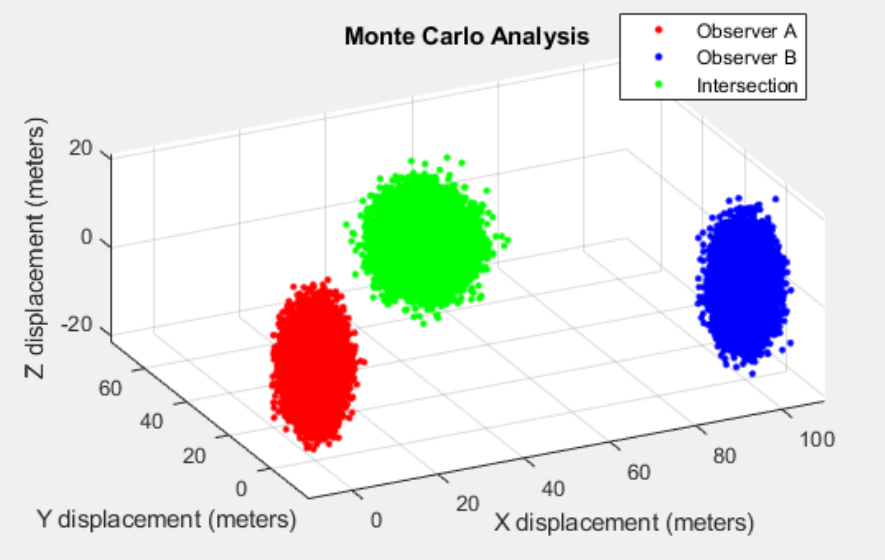

Figure 5-27: Scenario 1 Monte Carlo View B

The cloud of this intersection can be described using the RMS distribution along each X, Y, and Z plane. Figure 5-28 shows the RMS error values along each axis.

Axis	RMS Error
X	3.9618
Y	3.9858
Z	4.1472

Figure 5-28: Monte Carlo Scenario 1 RMS Error

A probability density function (PDF) was used to represent this RMS error distribution. Figure 5-29 shows the PDF representation of the intersection uncertainty.

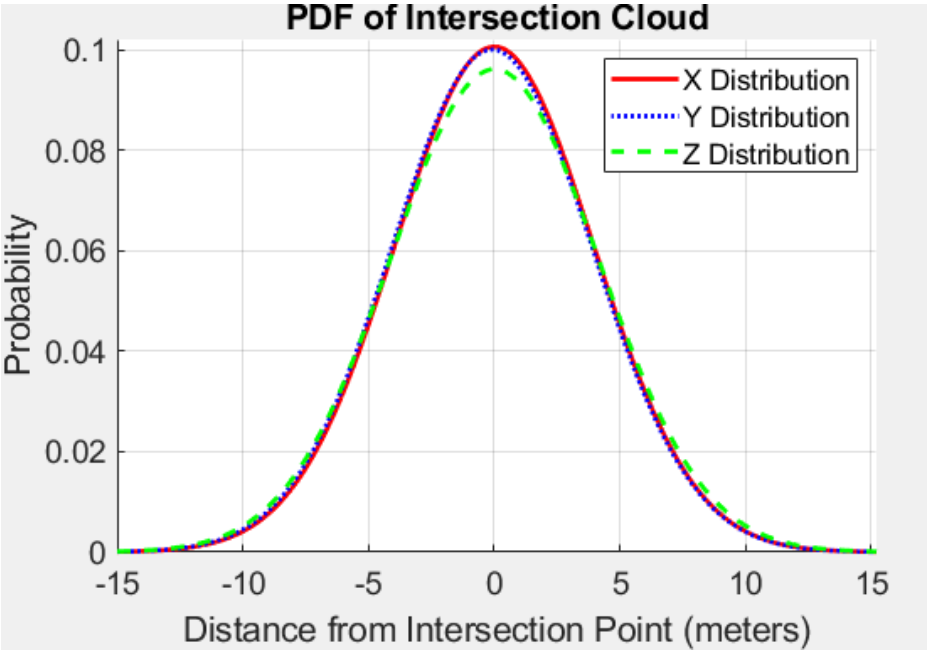

Figure 5-29: Monte Carlo Scenario 1 PDF

Note: if each axis distribution line cannot be seen, they are overlapping.

Next, these PDF distributions for each X, Y, and Z were combined to represent a single absolute PDF distribution for the intersection point cloud.

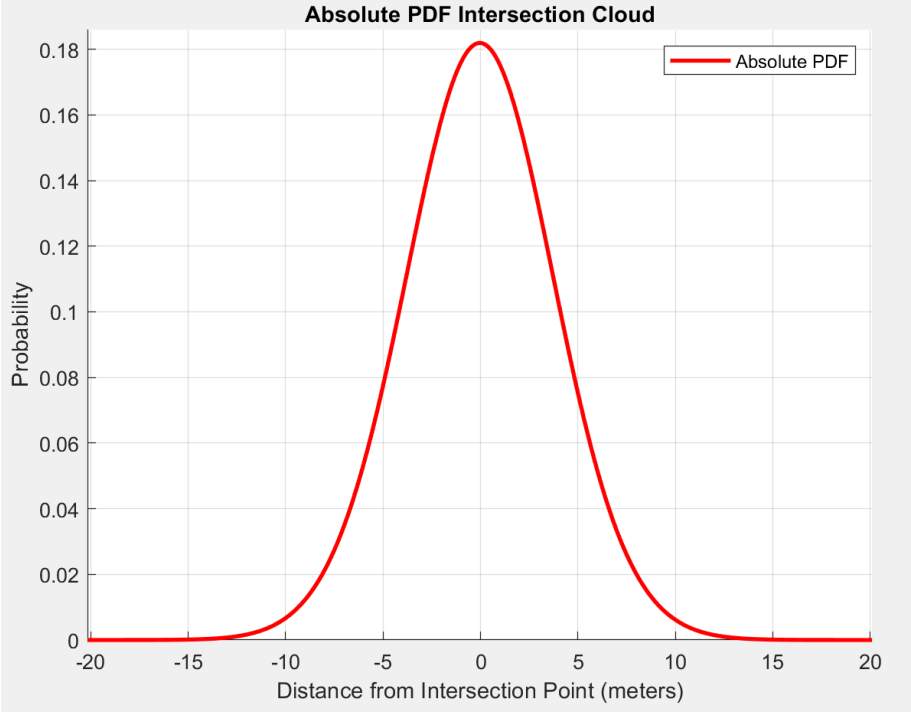

Figure 5-30: Monte Carlo Scenario 1 Absolute PDF

These RMS error values were seen to have similar distribution along each axis, since the intersection occurs directly in between each observer. This circular intersection distribution can be seen within the Figure 5-26 and 5-27 above.

The second simulation configuration consisted of truth values with noise added afterwards. The configuration of this simulation is: starting point of (10, 10, 10), 30 degree yaw angle, and 0 degree pitch angle for observer A, and a starting point of (100, 10, 10), 0 degree yaw angle, and 0 degree pitch angle for observer B. The resulting intersection cloud can be seen in Figures 5-31 and 5-32 below.

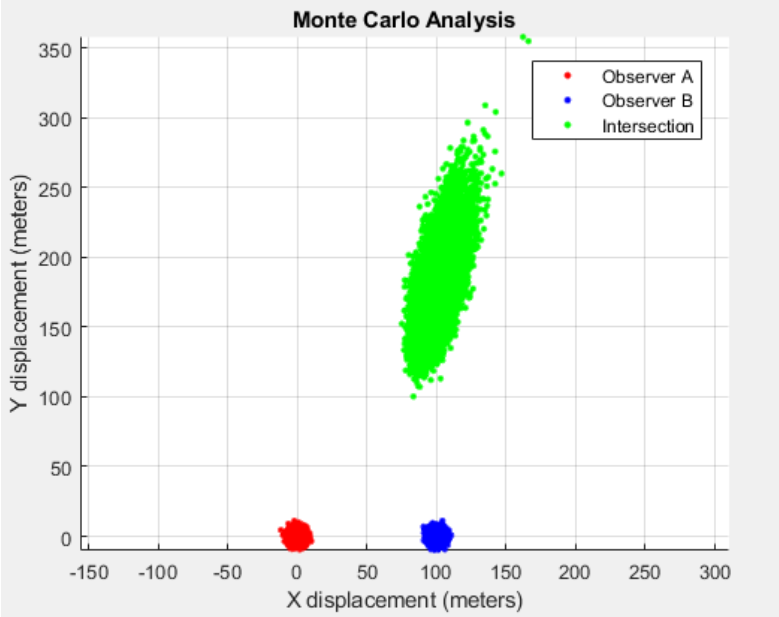

Figure 5-31: Scenario 2 Monte Carlo View A

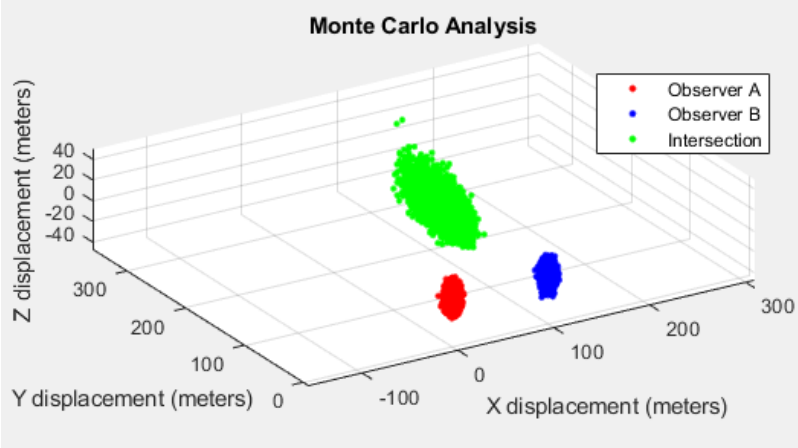

Figure 5-32: Scenario 2 Monte Carlo View B

The cloud of this intersection can be described using the RMS distribution along each X, Y, and Z plane. Figure 5-33 shows the RMS error values along each axis.

Axis	RMS Error
X	8.1516
Y	23.489
Z	6.77

Figure 5-33: Monte Carlo Scenario 2 RMS Error

A PDF was used to represent this RMS error distribution. Figure 5-34 shows the PDF representation of the intersection uncertainty.

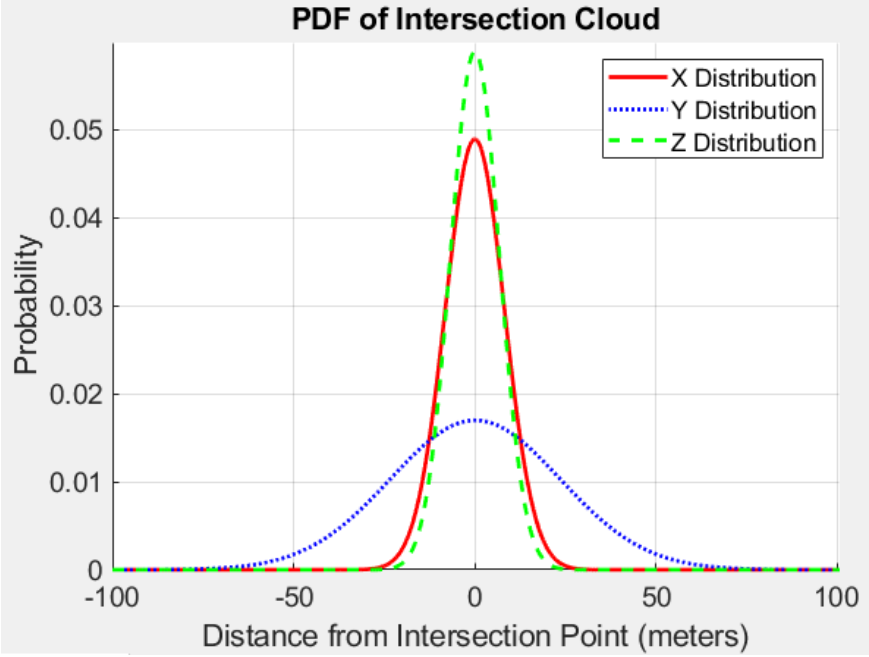

Figure 5-34: Monte Carlo Scenario 2 PDF

These RMS errors were seen to have larger distribution than the previous scenario. There are similar distributions along the X and Z axis, but there is a large distribution along the Y axis. This larger distribution is due to the intersection occurring further away from both observers and a smaller intersection angle. This intersection distribution can be seen within the Figures 5-31 and 5-32 above.

The first simulation configuration consisted of truth values with noise added afterwards. The configuration of this simulation is: starting point of (10, 10, 10), 60 degree yaw angle, and 0 degree pitch angle for observer A, and a starting point of (100, 10, 10), 0 degree yaw angle, and 0 degree pitch angle for observer B. The resulting intersection cloud can be seen in Figures 5-35 and Figure 5-36 below.

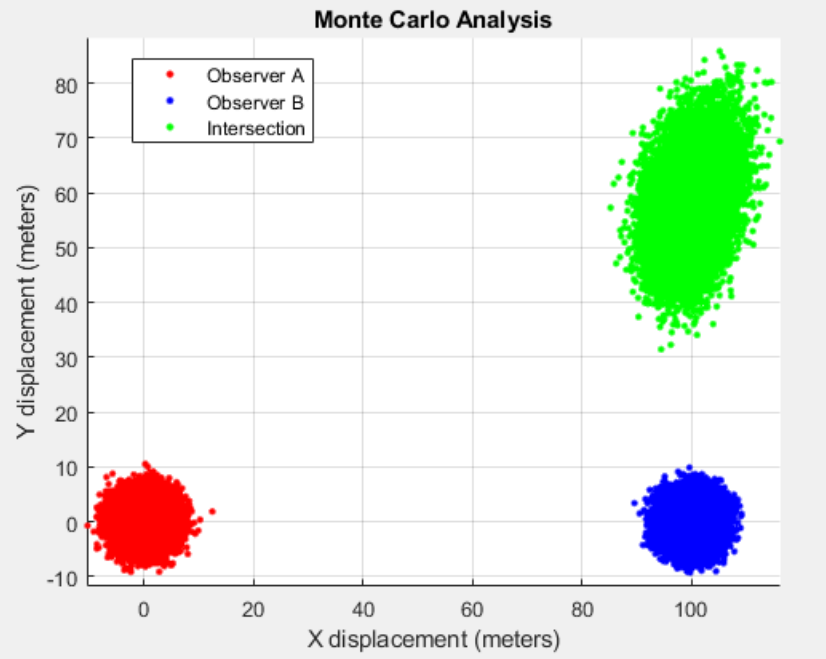

Figure 5-35: Scenario 3 Monte Carlo View A

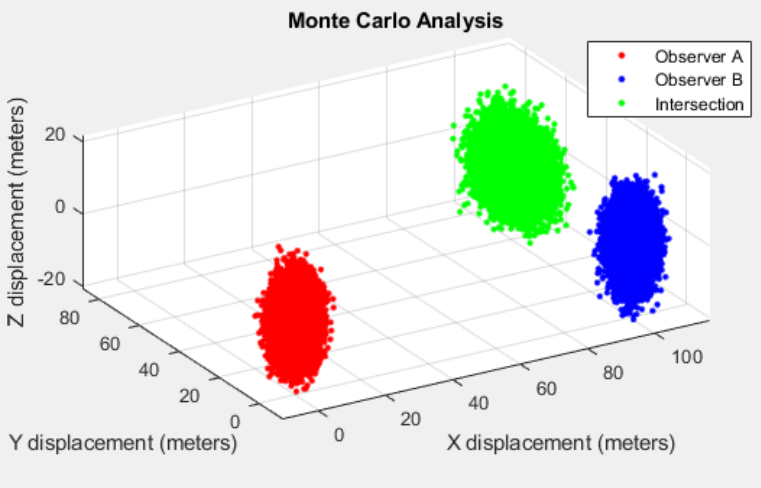

Figure 5-36: Scenario 3 Monte Carlo View B

The cloud of this intersection can be described using the RMS distribution along each X, Y, and Z plane. Figure 5-37 shows the RMS error values along each axis.

Axis	RMS Error
X	3.5969
Y	6.8426
Z	4.556

Figure 5-37: Monte Carlo Scenario 3 RMS Error

A PDF was used to represent this RMS error distribution. Figure 5-38 shows the PDF representation of the intersection uncertainty. Note: if each axis distribution cannot be seen, they are overlapping.

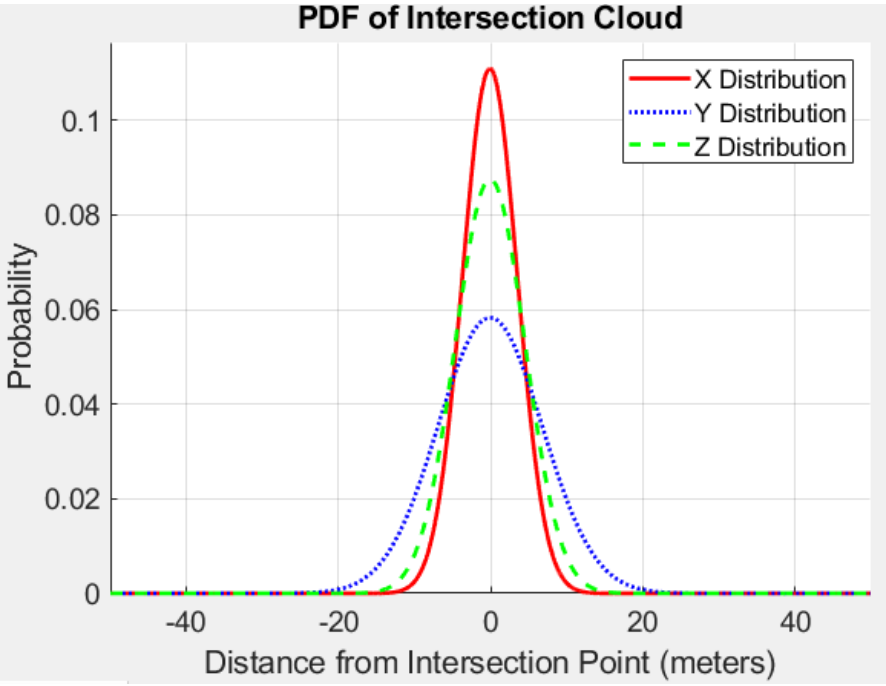

Figure 5-38: Monte Carlo Scenario 3 PDF

These RMS error values were seen to have the most distribution along the Y axis, with Z and X distributions being more precise. This distribution can be seen in Figures 5-35 and 5-36 above.

The fourth simulation configuration consisted of truth values with noise added afterwards. The configuration of this simulation is: starting point of (0, 0, 0), 45 degree yaw angle, and 0 degree pitch angle for observer A, and a starting point of (100, 0, 20), -45 degree

yaw angle, and 0 degree pitch angle for observer B. The resulting intersection cloud can be seen in Figures 5-39 and 5-40 below.

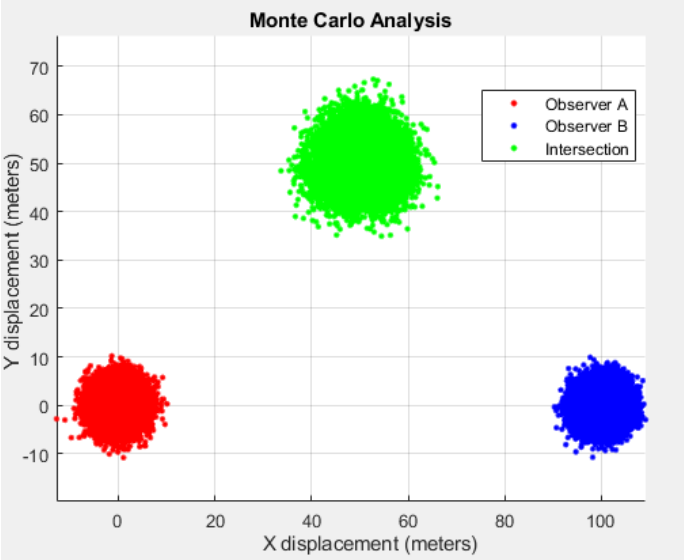

Figure 5-39: Scenario 4 Monte Carlo View A

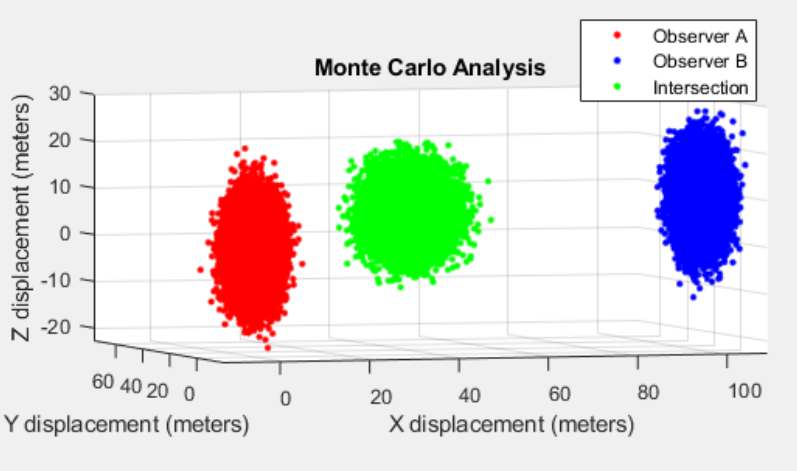

Figure 5-40: Scenario 4 Monte Carlo View B

The cloud of this intersection can be described using the RMS distribution along each X, Y, and Z plane. Figure 5-41 shows the RMS error values along each axis.

Axis	RMS Error
X	3.9825
Y	3.997
Z	4.1364

Figure 5-41: Monte Carlo Scenario 4 RMS Error

A PDF was used to represent this RMS error distribution. Figure 5-42 shows the PDF representation of the intersection uncertainty. Note: if each axis distribution cannot be seen, they are overlapping.

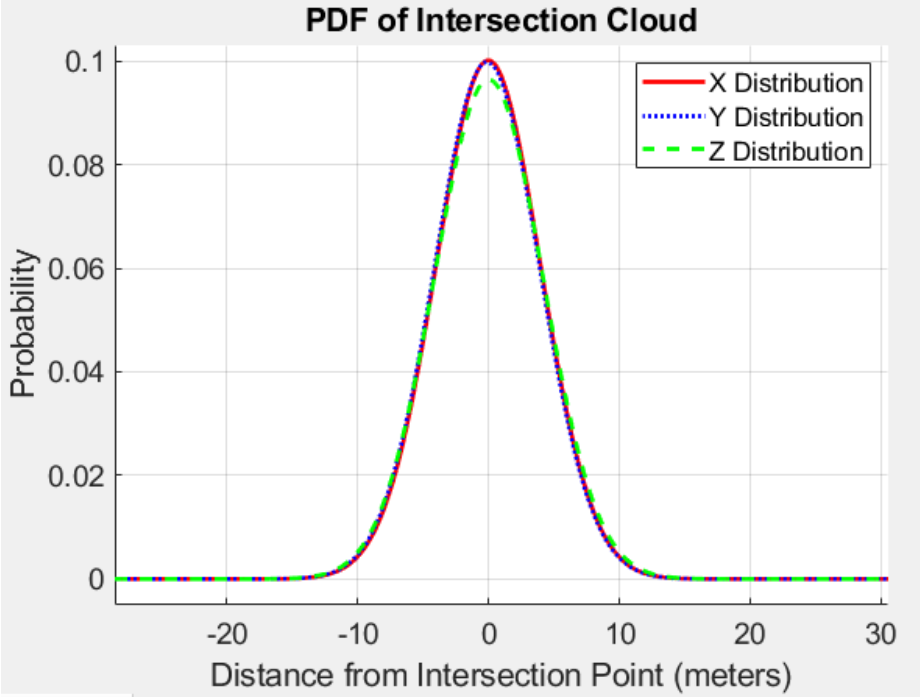

Figure 5-42: Monte Carlo Scenario 4 PDF

These RMS error values were seen to have similar distribution along each axis, since the intersection occurs directly in between each observer. This circular intersection distribution can be seen within the Figures 5-39 and 5-40 above.

These RMS values and Monte Carlo figures were interpreted to determine the effect of typical sensor noise to observe the intersection uncertainty during a real world test. Further investigation and explanation is detailed in the following discussion section.

Discussion

The cloud of intersection points was seen to change shape as the position and angle of each observer changes. The intersection seen in the first scenario of this section consists of an intersection angle of 90 degrees in the X, Y plane. With this angle of intersection, the cloud was seen to have equal X and Y distribution, with the Z distribution being independent from these measurements. As the pitch angle error was the same as the yaw error, the distribution of Z was similar to the X and Y. The intersection seen in the second scenario of this section shows a greater uncertainty in the Y direction than X and Z. This uncertainty was due to the intersection angle being smaller.

When the intersection angle becomes smaller, the more effect the yaw angle error will have on the resulting intersection position between the two vectors. Since these two vectors start on the same Y position, the greater the distribution in the Y direction, as it is between the two vectors. The intersection seen in the third scenario shows a large Y distribution and a similar distribution for X and Z. This distribution in the Y direction was less than the previous scenario due to the intersection angle being 60 degrees instead of 30. The yaw uncertainty still has an effect on the Y distribution, but is not as significant of an effect because the intersection angle is larger than the previous.

The intersection seen in the fourth scenario of this section was seen to be almost identical to the first scenario. This similarity was expected because the least squares method takes the average value between both observers' vectors. These two vectors, on average, were parallel in the Z axis, therefore the intersection points occurred at a Z coordinate directly between each observer in the Z axis. This intersection produced an intersection distribution in the Z axis that was half of the difference in the Z coordinate between observer A and B. This intersection cloud resulted in a similar distribution to scenario 1 but shifted up in the Z direction. In 2-D, the X Y intersection angle was 90 degrees therefore the X and Y distribution was very similar as well. These intersection distributions were modeled after real world errors and gave the ability to determine what errors and uncertainty might be seen in a field test.

5.2.2 Noise Input Varying Monte Carlo

Methods

In the previous Monte Carlo simulation, the observers' location and orientation values were analyzed with a constant noise distribution. In the real-world, however, these variables will encounter changing noise distributions, and these differences were tested with a Monte Carlo simulation. The simulation tests consisted of: varying the X and Y location error of a single observer, varying the Z location error of an observer, varying the yaw error of an observer, and varying the pitch error of an observer. The specific configuration of each input in the tested scenarios can be seen in Figure 5-43 to 5-46 below. In each scenario, there is no error for observer B. Observer B acted as a truth location and orientation vector, so the effects of each error could be observed from only observer A. The X Y noise test seen in Figure 5-43 started with no error on any of the measurements and the X Y position error was incremented. After each increment, a Monte Carlo simulation was run. The X Y position error was incremented by 0.1 m until it was equal to 10m of error.

X Y Noise Test										
	Observer A					Observer B				
Configuraiton	**X**	**Y**	**Z**	**Yaw**	**Pitch**	**X**	**Y**	**Z**	**Yaw**	**Pitch**
Gaussian Noise	0 -> 10	0 -> 10	0	0	0	0	0	0	0	0
Starting	0	0	0	45	0	100	0	0	-45	0

Figure 5-43: X Y Simulation Noise Test

The Z noise test seen in Figure 5-44 started with no error on any of the measurements, and the Z position error was incremented. After each increment, a Monte Carlo simulation was ran. The Z position error was incremented by 0.1 m until it was equal to 10m of error.

Z Noise Test										
	Observer A					Observer B				
Configuraiton	**X**	**Y**	**Z**	**Yaw**	**Pitch**	**X**	**Y**	**Z**	**Yaw**	**Pitch**
Noise	0	0	0 -> 10	0	0	0	0	0	0	0
Starting	0	0	0	45	0	100	0	0	-45	0

Figure 5-44: Z Simulation Noise Test

The yaw noise test seen in Figure 5-45 started with no error on any of the angle and location measurements, and the yaw angle error incremented. After each increment, a Monte

Carlo simulation was ran. The yaw angle error was incremented by 0.1 degree until it was equal to 10 degrees of error.

Yaw Noise Test										
	Observer A					Observer B				
Configuraiton	**X**	**Y**	**Z**	**Yaw**	**Pitch**	**X**	**Y**	**Z**	**Yaw**	**Pitch**
Noise	0	0	0	0 -> 10°	0	0	0	0	0	0
Starting	0	0	0	45	0	100	0	0	-45	0

Figure 5-45: Yaw Simulation Noise Test

The pitch noise test seen in Figure 5-46 started with no error on any of the angle measurements, and the pitch angle error was incremented. After each increment, a Monte Carlo simulation was ran. The pitch angle error was incremented by 0.1 degree it was equal to 10 degrees of error.

Pitch Noise Test										
	Observer A					Observer B				
Configuraiton	**X**	**Y**	**Z**	**Yaw**	**Pitch**	**X**	**Y**	**Z**	**Yaw**	**Pitch**
Noise	0	0	0	0	0 -> 10°	0	0	0	0	0
Starting	0	0	0	45	0	100	0	0	-45	0

Figure 5-46: Pitch Simulation Noise Test

A resulting intersection location sensitivity, or variable-dependent area of uncertainty, was determined by varying each input one at a time. This test determined the specific variables that produced the highest intersection-location variance and was used to place the most emphasis to filter or ensure the specific variables were accurate. Ultimately, this simulation provided the correlation between each input variable's uncertainty to the accuracy and precision of the intersection location estimate.

Results

Noise was applied to the position and orientation of one observer's measurements individually and incrementally increased to observe the variable's effect on the intersection shape. Refer to Section 5.2.1 specific noise values during each test.

The first scenario was analyzed with increasing noise applied to the X and Y locations. Refer to Figure 5-43 for the progression of the noise being tested. The resulting intersection and position uncertainty can be seen in Figure 5-47 below.

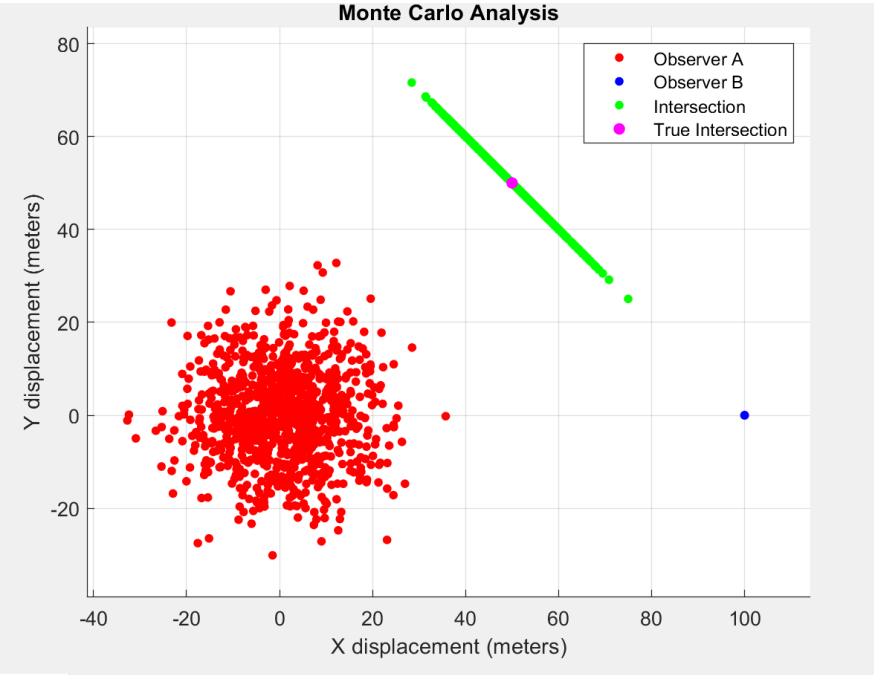

Figure 5-47: Observer A X Y Uncertainty Ending with 10 m Noise

The resulting RMS error value of the intersection uncertainty can be seen in Figure 5-48 below.

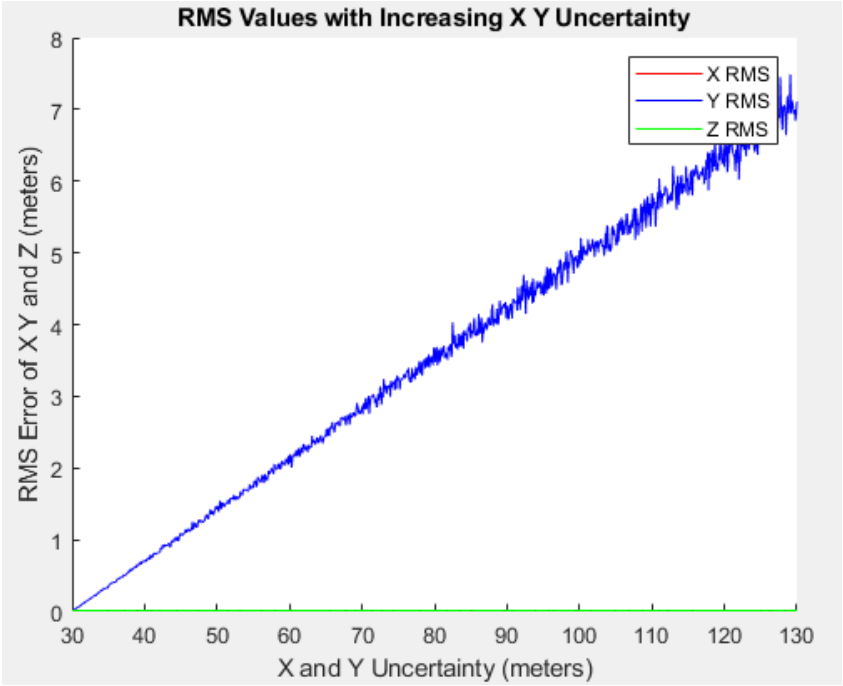

Figure 5-48: Monte Carlo RMS Values with X Y Uncertainty. The X RMS line cannot be seen because it is identical to the Y RMS line.

In Figure 5-48 above, The X and Y RMS error value of intersection distribution was seen to increase linearly as the X and Y location uncertainty increases.

The first scenario was next analyzed with noise applied to the Z coordinate, to model the location uncertainty in the Z direction. Refer to Figure 5-44 for the progression of the noise being tested. The resulting intersection uncertainty shape can be seen in Figure 5-49 below.

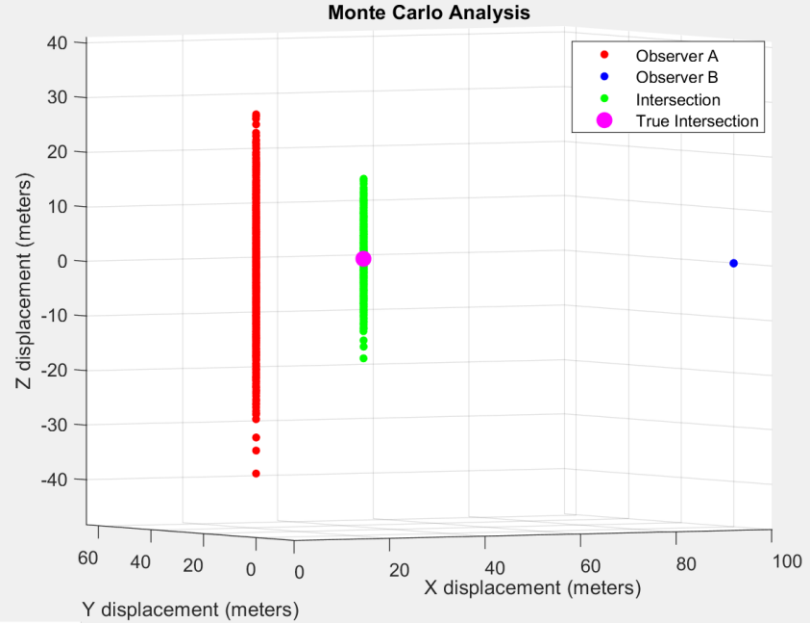

Figure 5-49: Observer A Z Uncertainty Ending with 10m Noise

The resulting RMS value of the intersection uncertainty can be seen in Figure 5-50 below.

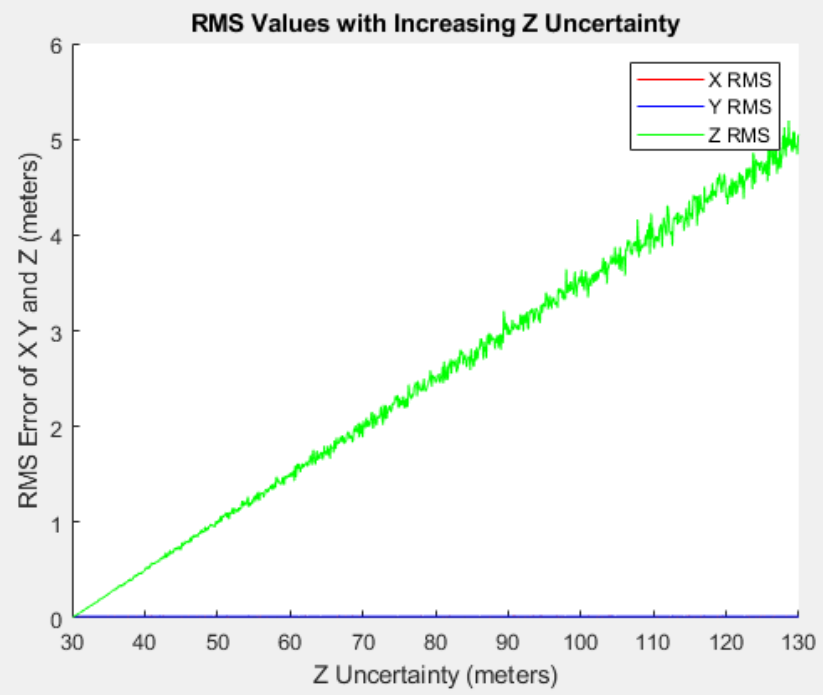

Figure 5-50: Monte Carlo RMS Values with Z Uncertainty The X RMS line cannot be seen because it is identical to the Y RMS line.

The Z RMS value of the intersection distribution was seen to increase linearly as the Z location uncertainty increases.

The first scenario was next analyzed with noise applied to the yaw angle, to model the 2-D angle uncertainty. Refer to Figure 5-45 for the progression of the noise being tested. The resulting intersection uncertainty shape can be seen in Figure 5-51 below.

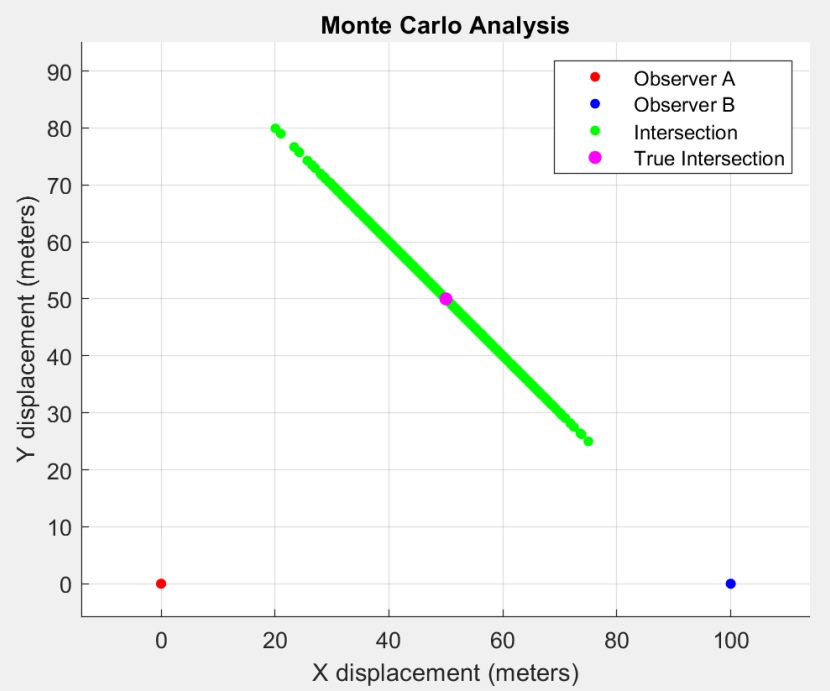

Figure 5-51: Observer A Yaw Ending with 10 degree Uncertainty

The resulting RMS value of the intersection uncertainty can be seen in Figure 5-52 below.

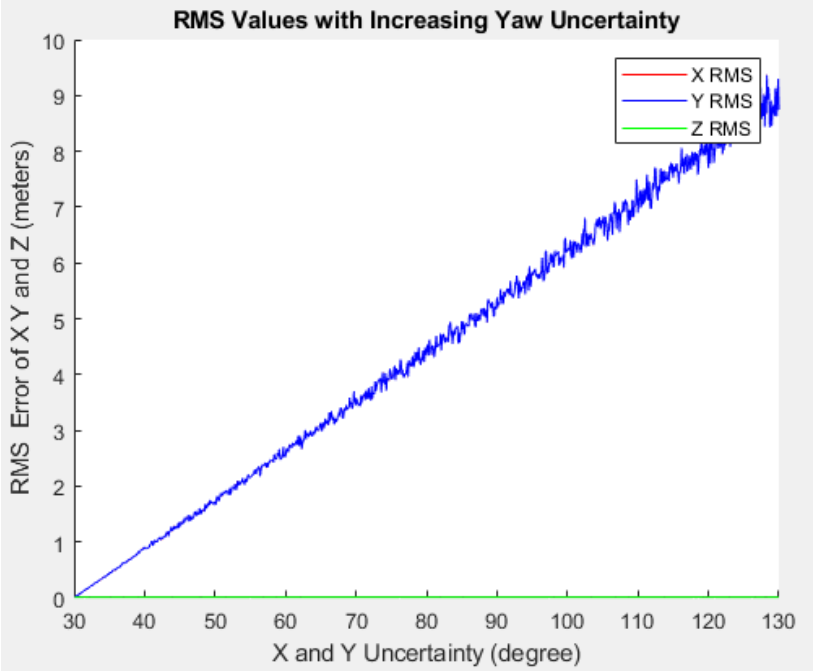

Figure 5-52: Monte Carlo RMS Values with Yaw Uncertainty. The X RMS line cannot be seen because it is identical to the Y RMS line.

The X and Y RMS value of intersection distribution was seen to increase linearly as the yaw angle uncertainty increases.

The first scenario was next analyzed with noise applied to the pitch angle, to model the 3-D angle uncertainty. Refer to Figure 5-46 for the progression of the noise being tested. The resulting intersection uncertainty shape can be seen in Figure 5-53 below.

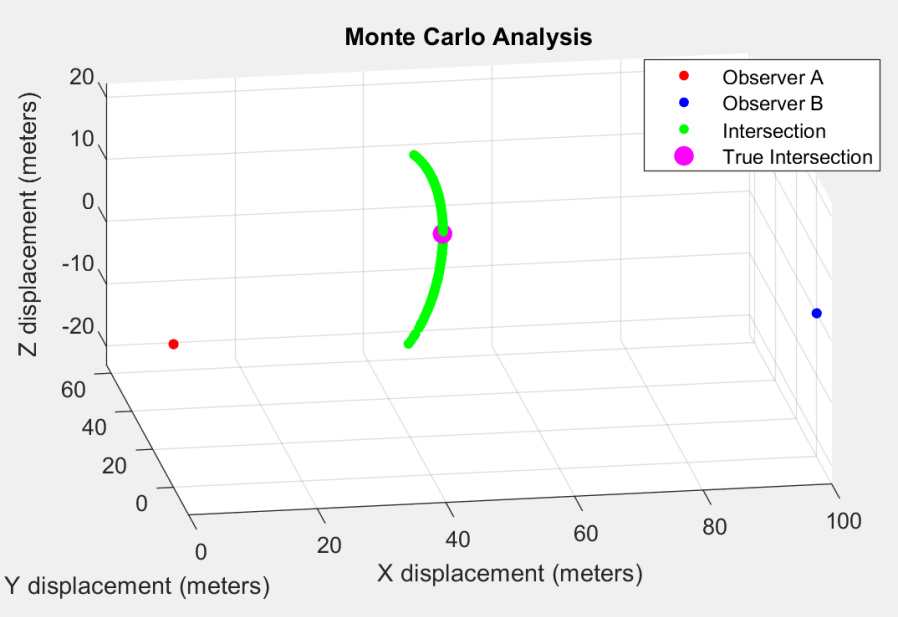

Figure 5-53: Observer A Pitch Ending with 10 degree Uncertainty

The resulting RMS value of the intersection uncertainty can be seen in Figure 5-54 below.

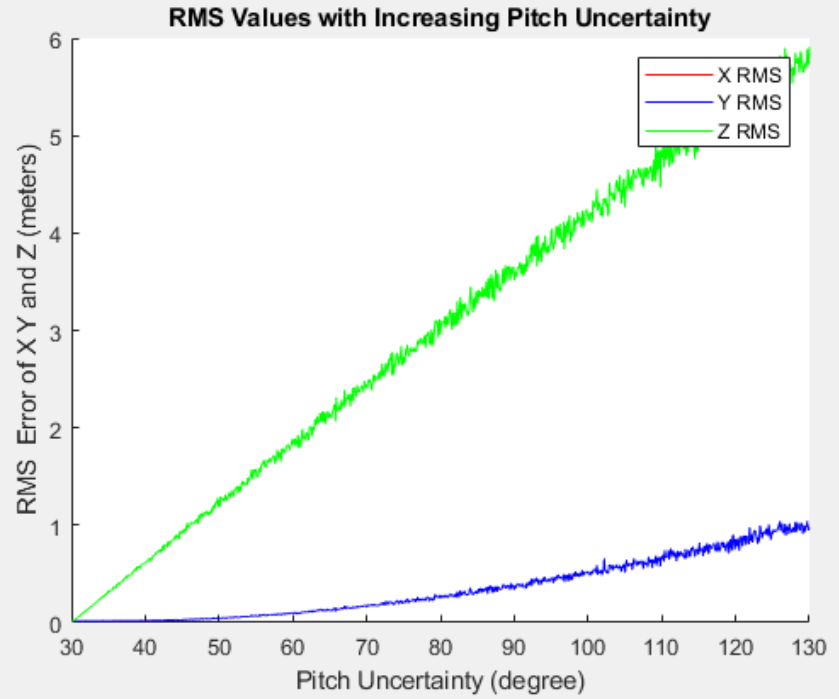

Figure 5-54: Monte Carlo RMS Values with Pitch Uncertainty. The X RMS line cannot be seen because it is identical to the Y RMS line.

The Z RMS value of intersection distribution was seen to increase linearly and the X and Y RMS was seen to increase at a slower rate than the Z RMS as the pitch angle uncertainty increases. These results are further analyzed within the discussion section of this chapter.

Discussion

Simulations were run while independently varying location or orientation noise in order to determine the influences of each sensor error to the resulting intersection uncertainty cloud using the initial configurations from scenario 1. Each scenario involved increasing noise on one variable for one observer, while the other observer had no location or orientation error. This simulation was conducted in order to characterize each variable noise and compare it to a truth vector. Seen in Figure 5-48, the X and Y RMS error values of the intersection uncertainty were seen to linearly increase as the X and Y location uncertainty for observer A increased, and Z RMS error remained constant. This X and Y position uncertainty simulation shows a relation to an X and Y intersection uncertainty. Next, the Z position noise was varied, with no noise on the X and Y position or angles. As the Z position uncertainty became larger, the Z RMS error of the intersection uncertainty became larger, and the X and Y intersection uncertainty stayed constant.

The Z RMS error was seen to increase linearly as the Z uncertainty increased, but at a smaller rate than the previous scenario with the X and Y RMS error, due to the averaging in the least-squares method when the vectors did not directly intersect, even when the target was stationary at Z = 0. This Z position uncertainty simulation shows a relation to a Z intersection uncertainty.

In the next simulation, the yaw angle noise was varied with no noise on the position or pitch angle. As the noise became larger, the X and Y RMS error of the intersection uncertainty became larger, and the Z RMS error remained constant. The X and Y RMS error was seen to follow a linear trend as the yaw error increased. This yaw angle uncertainty simulation shows a relation to X and Y intersection uncertainty. Lastly, the pitch angle noise was varied with no noise on the position or yaw angle. As the noise became larger, the Z RMS error of the intersection uncertainty followed a linear trend, and the X and Y RMS error increased with a smaller slope, but still a linear trend, after the pitch uncertainty was greater than 4 degrees. The pitch angle uncertainty simulation shows a relation to the Z intersection uncertainty and a relation with the X and Y intersection uncertainty after a certain error threshold.

This additional component was due to the least-squares approximation; when the pitch difference between each observer was large enough, the minimum difference between each vector will occur in a location with a different X and Y position than the 2-D intersection. This occurrence can be seen in simulations 3-4, 6-8, and 10-16 in the validation section 4.3. These simulations provided the relationships between each observer measurement uncertainty and the resulting intersection uncertainty. An important finding was determining that pitch uncertainty can cause the X and Y intersection uncertainty to become greater and to occur at a different X and Y coordinate than the 2-D intersection. These findings were further used to determine the root cause of the intersection uncertainty distributions.

5.2.3 Moving Observers: Location and Angle Varying Monte Carlo

Methods

The next simulation consisted of modeling moving observers to analyze the effect of intersection angle and distance on the intersection uncertainty shape and distribution. The first simulation was used to assess the optimal vector intersection angle on the object of interest that produced the most precise intersection uncertainty distribution, and the next simulation observed

the effect of location and orientation error on the resulting intersection uncertainty, as distance increased. In the first simulation, observer A started at position (193.96, 249.634, 0) with a yaw angle of 160°, and observer B started at the position (193.65, 150.4, 0) with an yaw angle of 20°, and the intersection point was static at (200, 200, 0). The starting configuration can be seen in Figure 5-55 below.

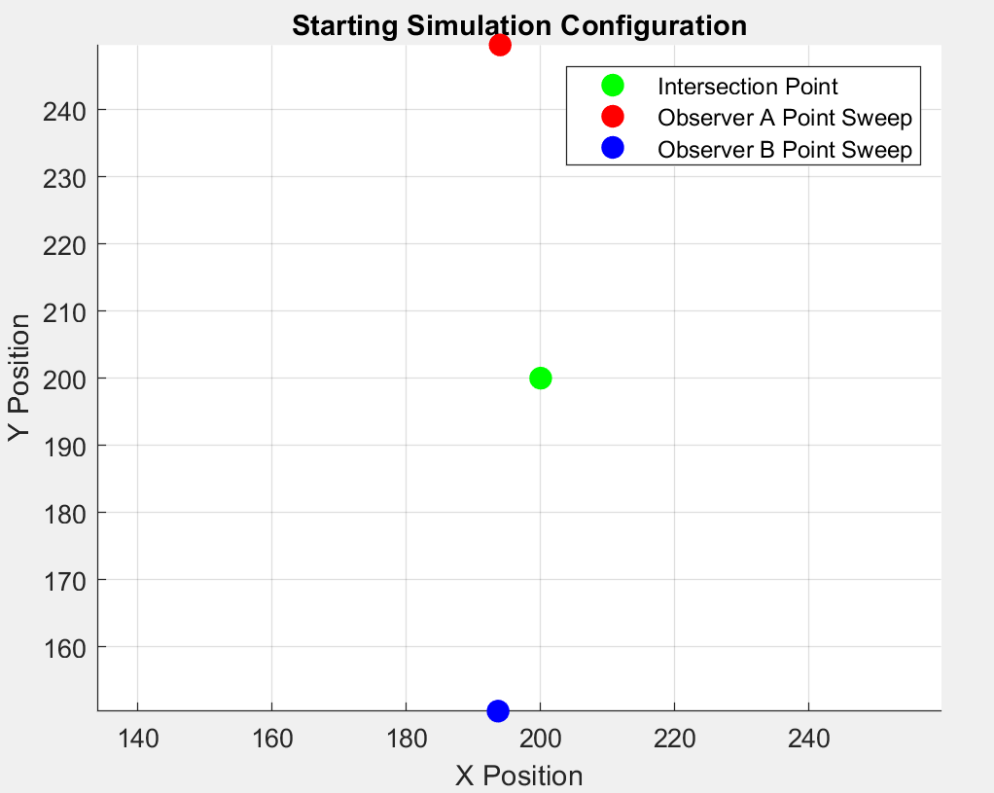

Figure 5-55: Simulation Starting Point

The error values for each variable can be seen in Figure 5-56 below. Notably, the location error in the Z direction is larger than the error in the X and Y to reflect the larger GPS uncertainty when determining altitude.

	Location			Orientation	
	X	Y	Z	Yaw	Pitch
Error	2.5 m	2.5 m	5 m	1°	1°

Figure 5-56: Optimal Angle Simulation Error Values

These two observers were swept following a circular path, with a 1° increment and a radius of 50 m. The resulting intersection RMS error was observed as the intersection angle changed. The distance was then incremented by 50 m and the resulting error was observed, until the distance was equal to 350 m. The position sweep for each iteration can be seen in the Figure 5-57 below.

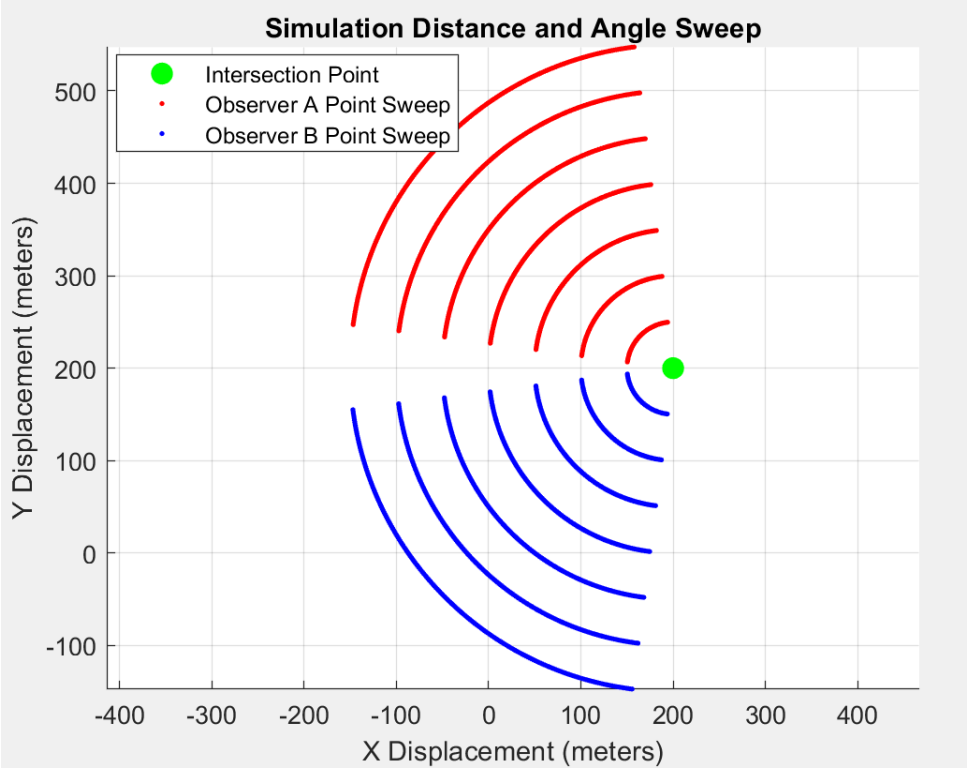

Figure 5-57: Simulation Distance and Angle Sweep

The second simulation had the same test parameters seen in Figure 5-56, but the orientation and location errors were changed in order to observe their effects on the resulting intersection uncertainty. The test parameters for this simulation can be seen in Figure 5-58 below.

	Location			Orientation	
	X	Y	Z	Yaw	Pitch
Location Test	2.5 m	2.5 m	5 m	0°	0°
Orientation Test	0 m	0 m	0 m	1°	1°

Figure 5-58: Location and Orientation Error Test

The location test consisted of only location error, and the effect of the resulting intersection uncertainty as distance increased was observed. Next, the orientation test consisted of only orientation error, and the effect was observed as well. Lastly, the location test was used to determine the lower bound, or best case intersection uncertainty. It was imposed on the figure resulting from the test seen in Figure 5-56 and was used to also determine the distance at which the orientation error effects the intersection uncertainty more than the location error.

Results

The RMS error from the intersection cloud to the truth intersection coordinate (200 m, 200 m, 0 m) was found for each Monte Carlo run. In the first simulation, the RMS error was found with relation to the intersection angle for each distance increment. The configuration for the test can be seen in Figures 5-56 and 5-58, and the resulting plot can be seen in Figure 5-59 below. The minimum line is at a distance of 50 m, and each increasing line is from an additional 50 m away from the intersection point.

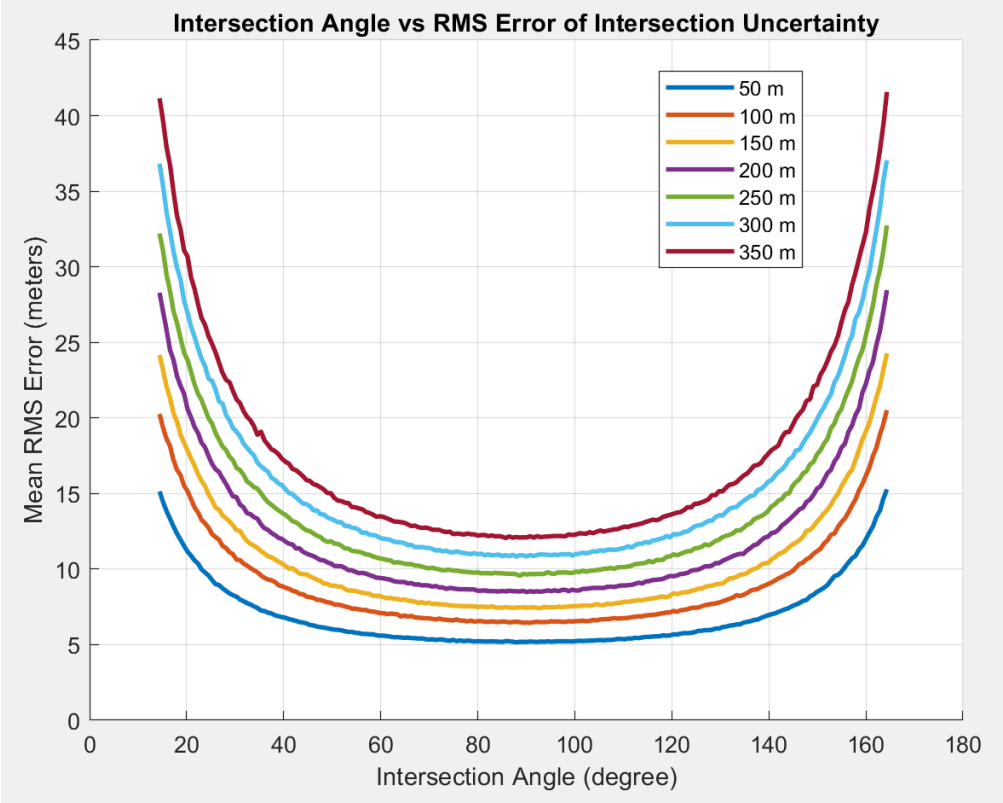

Figure 5-59: Monte Carlo Mean RMS Error with Changing Intersection Angle

The data followed a U-formation with a valley that is centered at 90°, so in general, the mean RMS error minimum will occur around 90 °. It can also be observed that as distance increases, so does the RMS error.

In the second simulation, two separate tests were performed with the same varying parameters as the previous simulation. First, the location error test was performed with only location error, no orientation error. The resulting plot can be seen in Figure 5-60 below, with the distances incremented by 50 m until 350 m is reached.

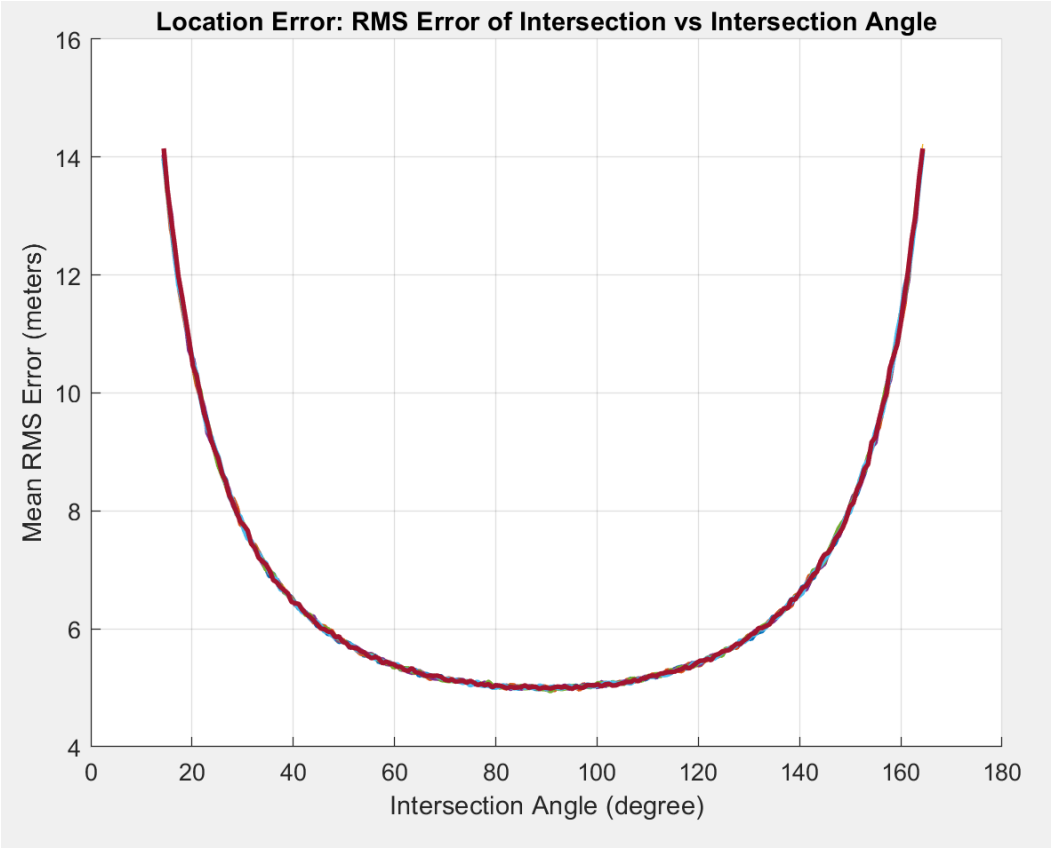

Figure 5-60: Monte Carlo Mean RMS Error with Changing Intersection Angle for Location Error

Note: The lines, for every distance, are overlapping each other

The data followed a similar trend as Figure 5-59, with a U-formation with a valley that is centered at 90°. But, the mean RMS error did not increase as the distance increased. Instead, the error was a direct translation from each observer to the resulting intersection, not dependent upon the distance from the intersection point.

Second, the orientation error test was performed with only orientation error, no location error. The resulting plot can be seen in Figure 5-61 below, with the starting distance 50 m being the minimum line and each increasing RMS error line is from an additional 50 m away from the intersection point.

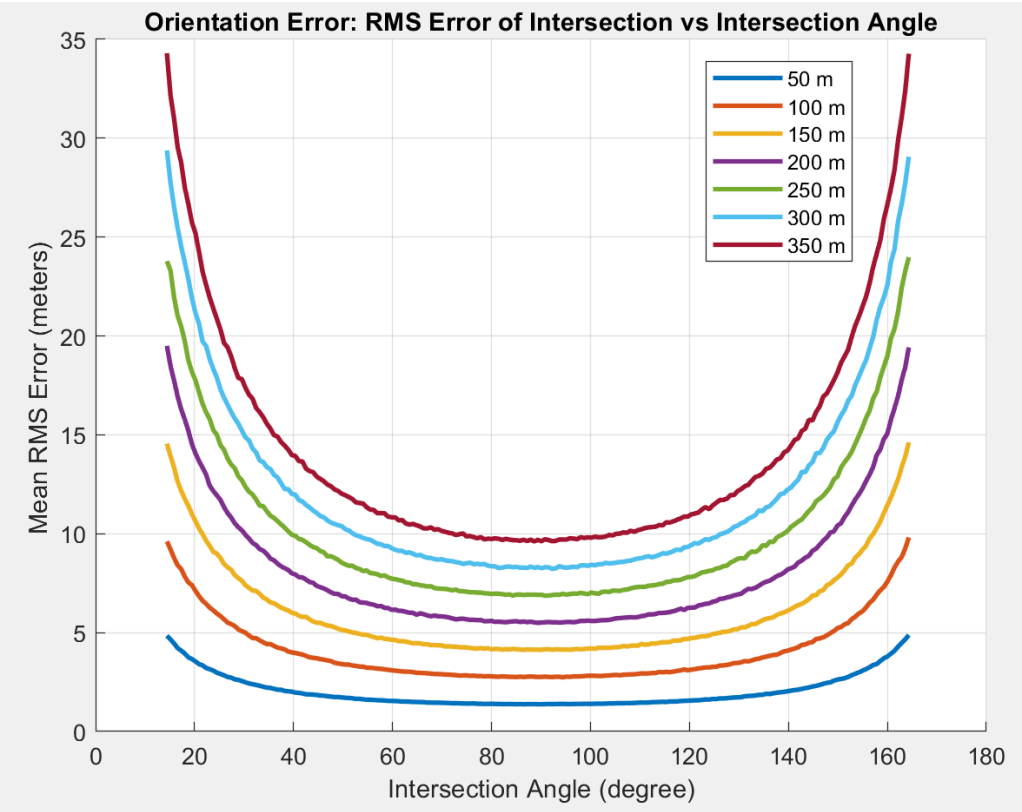

Figure 5-61: Monte Carlo RMS Error with Changing Intersection Angle for Orientation Error

The data followed a similar trend as Figure 5-59, with a U-formation with a valley that is centered at 90^{o} and the RMS error increased as the distance increased. Except, the mean RMS error is shifted down compared to Figure 5-59 due to the absence of the location error.

Lastly, the location error in Figure 5-60 was imposed on the graph in Figure 5-61 to exemplify the lower bound, or limiting factor, to the indirect geolocation solution with errors added. The resulting plot can be seen in Figure 5-62 below, with the lower bound being the bold black line.

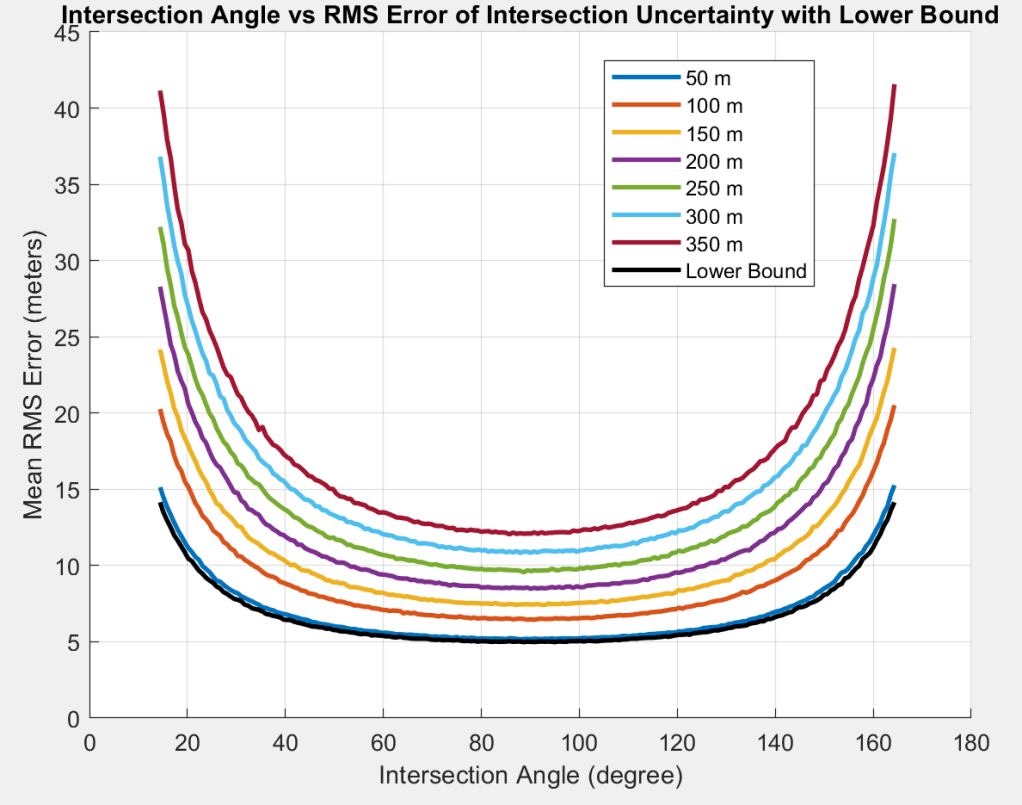

Figure 5-62: Monte Carlo RMS Error with Changing Intersection Angle with Lower Bound

The lower bound was observed to be the minimum line seen in the plot in Figure 5-62 above. This lower bound is the best case indirect geolocation mean RMS error solution. At this line, the system is considered GPS limited, where the lowest intersection RMS error can be found. This system is considered GPS limited because the location is solely found using the GPS, so the best intersection solution the system can calculate is limited by the location error in the GPS. The point where orientation error has more effect on the location solution can be determined from the data produced in Figure 5-62 as well. The lower bound was divided by each distance to determine its contribution to each distance uncertainty. The 50% point was determined to occur close to 250 m, and any distance after this point was considered to be more orientation dependent error. These results are further analyzed and interpreted within the discussion section of this chapter.

Discussion

In this section, simulations were run with observer A and observer B being moved in a circular position with different radii to observe the relationship between the mean RMS error and

intersection angle as distance increased. And next, simulations were run with the same test parameters, but the location and orientation errors were changed to observe their effect on the intersection uncertainty as the distance increased. Figure 5-59 shows that as the intersection angle changed, the minimum mean RMS error minimum centered on an intersection angle of 90^o, and remained true as distance from the intersection point increased. This minimum is due to the factors that acute or obtuse angles have on the resulting intersection uncertainty, with input errors added. When these angles are acute or obtuse, the yaw and location errors are amplified and each error has a larger effect on the point of the resulting intersection. From this figure, it was determined that the most precise intersection uncertainty distribution occurs at an intersection angle 90^o with typical real world error values added. The second scenario consisted of similar test parameters, but the difference in location error and orientation error was tested as distance increased. Seen in Figure 5-60, it was observed that distance is not a factor when there is only location error in the system. Due to the location being found directly from the GPS, it is the limiting factor for obtaining a flawless system. Therefore, this location error can be determined as the lower-bound error when comparing to the first simulation in this section. The resulting comparison can be seen in Figure 5-62. The orientation error test showed that distance is a factor when there is only orientation error in the system. The results from Figure 5-62 were used to determine that at 250 m, the orientation effects the intersection uncertainty cloud more than the location error. This point can be used to determine which error to take into account more when wanting to perform indirect geolocation at particular distances. These simulations provided valuable information regarding the optimal intersection angle to produce the most precise intersection uncertainty, and the effect of location and orientation error over distance to the intersection uncertainty.

5.3 Discussion

In this chapter, the characterization of the two-dimensional intersection algorithm, the three-dimensional algorithm were detailed. First, the two-dimensional algorithm was characterized using Monte Carlo simulations. The specific simulations consisted of: static scenario, varying distance, and varying distance and orientation. The static scenario simulations showed the two main characteristics of the intersection point cloud that can be characterized:

RMS error and standard deviation. These simulations showed that when the degrees of freedom are limited by removing orientation error and having an angle on the observer of 90^{o} the RMS error follows a Rayleigh distribution. The varying distance simulations showed the different effects that position and orientation errors have. In general, position error acted as a bias in the intersection mean RMS error. Orientation error was the main source of mean RMS error for all runs. The varying distance and orientation simulations showed mainly that there exists an optimal angle on the observer that minimizes the intersection mean RMS error. That angle over a series of simulations was shown to be 90^{o}. All these simulations characterized the two-dimensional intersection algorithm throughout a variety of scenarios. Future work in characterizing the 2-D intersection could examine observer with different error distributions, i.e. if one observer has a bad device, what is the effect on the intersection if the other observer has a good device.

Second, the three-dimensional algorithm was characterized using Monte Carlo simulations. The specific simulations consisted of: static scenario, noise input varying, and location and angle varying simulations. The static scenario simulations were conducted with similar measurement errors that would be seen in real world simulations. The simulations provided insight about the intersection distributions with various configurations and gave the ability to determine what errors and uncertainty might be seen in a field test. The noise input varying simulations were conducted with each input being independently varied in order to observe the effects of each variable on the resulting intersection distribution. It was found that when the X and Y position errors become greater, the intersection X and Y RMS error values increased linearly, and the Z RMS error remained constant. When the Z position error became greater, the intersection Z RMS error value increased linearly, and the X and Y RMS error remained constant. When the yaw error was increased, the intersection X and Y RMS error values increased linearly, and the Z RMS error remained constant. Lastly, when the pitch error increased, the intersection Z RMS error increased linearly, and the X and Y RMS error value was observed to increase at a slower rate than the Z RMS error, but still increase linearly with the pitch error. The finding from these simulations provided information to determine the root cause of the intersection uncertainty distributions that might be seen in noisy environments. Lastly, the location and angle varying simulations were conducted to observe the relation between RMS absolute error and intersection angle as distance increased, and the effect of location and

orientation error on the resulting intersection uncertainty. It was found that the most precise intersection uncertainty distribution occurs at an intersection angle 90^{o} with typical real world error values added, and remained true as distance increased. It was also seen that as distance increased, the location error had the same effect on the intersection uncertainty and acts as a lower bound limitation to the indirect geolocation system. For orientation error, however, the RMS absolute error increased as distance increased. It was found that at 250 m distance away from the intersection, the orientation error has more of an effect on the intersection uncertainty than the location error. These simulations provided valuable information regarding the characteristics of the 3-D intersection and the resulting uncertainties that would be experienced in field tests.

Future work in the characterization of the 3-D intersection could include a model that introduces user error, e.g. inadvertent shaking of the device, into the simulations, rather than only sensor errors. The additional user error would create a more realistic real world simulation, but the user error would have to be estimated, or quantified within field tests. Overall, these simulations are a representation of a few scenarios that were encountered during field tests and relationships between observer position and intersection angle that were able to be observed and concluded.

Chapter 6: Field Tests

This chapter begins with a summary of the process used to conduct field tests. Following, a more detailed description of the methods and results for static field tests are presented. After discussing the static tests, the methods and results of a dynamic test are presented. Finally, a detailed discussion of the results and some ideas for future work are presented.

6.1 Two-Dimensional Intersection Real World Performance

In order to supplement the results obtained using simulations, a series of field tests was also conducted. Specifically, these field tests were classified into two groups: static and dynamic. The static field tests were executed using a very strict, controlled protocol. Changing one variable at a time, these tests sought to model the theory provided by the 2-D and 3-D simulations of Sections 5.1 and 5.2. Reflective of highly idealized case scenarios, these tests attempted to mitigate the many random variables present within real-world testing.

On the contrary, the dynamic tests were more indicative of a real-world application. A means of testing the true rigidity, precision, and accuracy of the system, the dynamic tests helped indicate the feasibility of using smartphones for indirect geolocation. In order to make the data analysis of the dynamic tests simpler, the tests still followed relatively strict protocols. Despite these strict protocols, however, the increased degrees of freedom made the dynamic tests difficult to replicate with simulation.

In order to establish as much certainty into the testing procedure as possible, the tests were performed at Wachusett Mountain in Princeton, Massachusetts. At the summit of the mountain is a United States Geological Survey (USGS) marker with very accurate latitude and longitude coordinates. Leveraging the truth reference provided by these coordinates, a 5-foot-tall tripod was placed on top of the USGS marker as the object of interest. As stated per the USGS datasheet, this marker was located at 42 29 20.59612 (N) and 071 53 12.26962 (W) (NGS Data Sheet).

For data collection, the smartphones used the SensorFusion application available on Google Play. Using just one Samsung Galaxy J7 smartphone, the application's data logging feature logged raw data from the accelerometer, magnetometer, gyroscope, and GPS to the

phone's internal storage. The data from each experiment, arranged by timestamp, was then transferred to a micro SD card for post-processing.

6.2 Static Tests: Methods and Results

As seen in Figure 6-1, six locations, as represented by the blue dots, were chosen for testing. Furthermore, at each location a series of different tests was performed. For the sake of thoroughness and simplicity, however, this section focuses on just two different tests performed at three specific angles: 90 degrees, 60 degrees and 135 degrees.

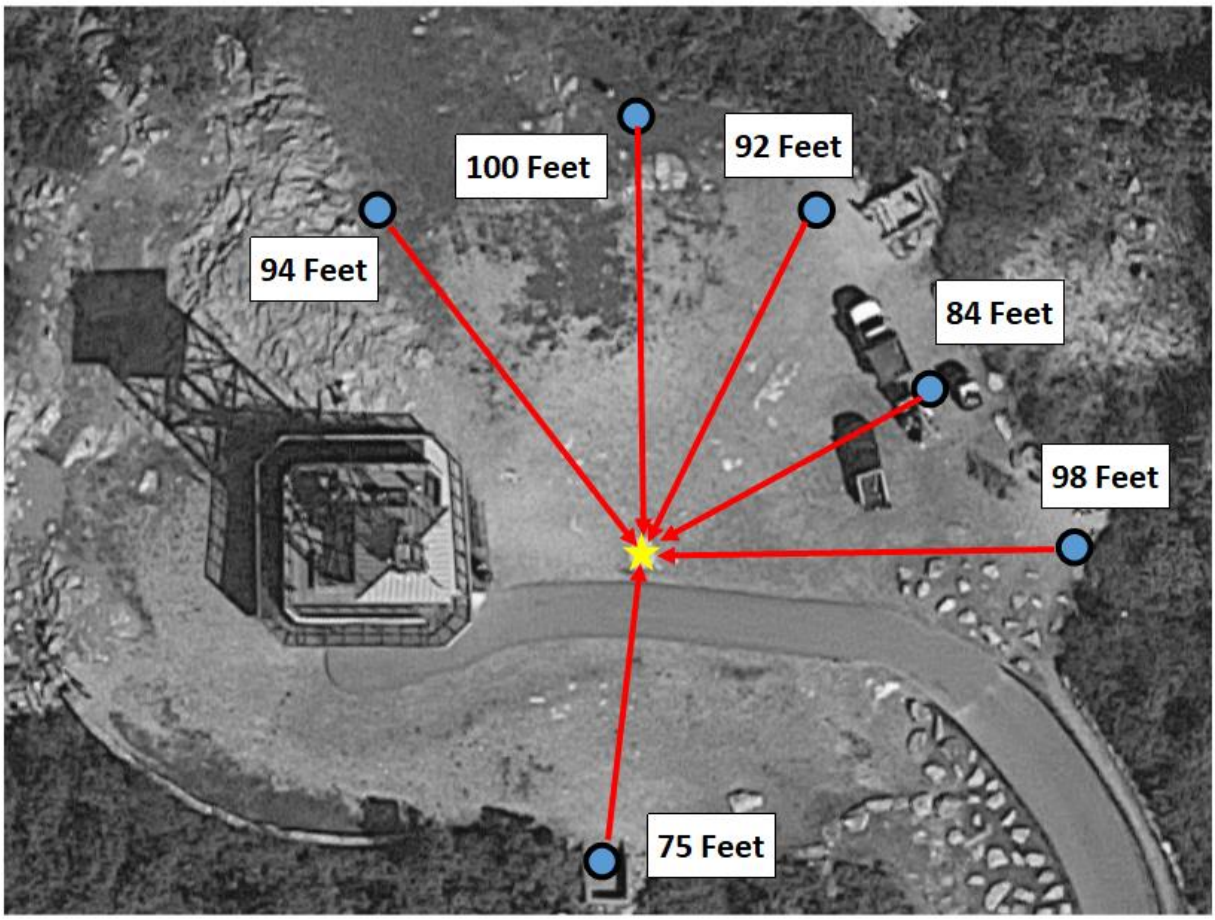

Figure 6-1: User Testing Locations and Orientations for Field Tests at Wachusett Mountain

In order to ensure that testing was performed as accurately as possible, two major precautions were taken. First, all observer locations, as represented by the blue dots, were measured using a tape measure with respect to the USGS marker (i.e. the object of interest). Consequently, this precaution allowed each observer location to be measured comparatively to

the known truth. Second, in order to obtain accurate orientation data, all observer pointing vectors, as represented by the red lines, were measured using a laboratory field-testing compass. Together, these two precautions helped create a much more consistent testing experience based on known and estimated truths.

GPS Error:

In order to better understand the performance of the indirect geolocation system under static tests, the performance of the GPS had to be well characterized. Unlike the accelerometer, gyroscope, and magnetometer values, which underwent pre-filtering, calibration, and Kalman Filtering, the GPS data had thus far been unaltered.

In order to determine the accuracy and precision of the GPS, the phone was placed with its back lying down on the USGS marker. As each of the two phones was likely to have different GPS errors, only one phone was used throughout the duration of the static testing. Furthermore, the GPS error test was performed after all other experiments. As such, the GPS had already established frequent communication with the nearest constellation of satellites, thus eliminating the possibility of a cold start by the receiver.

After conducting two one-minute static tests with the phone positioned on the USGS marker, the location distributions in Figure 6-2 and Figure 6-3, respectively were obtained. Although these distributions referred to specific static tests, the two figures represented the general trends of the GPS data.

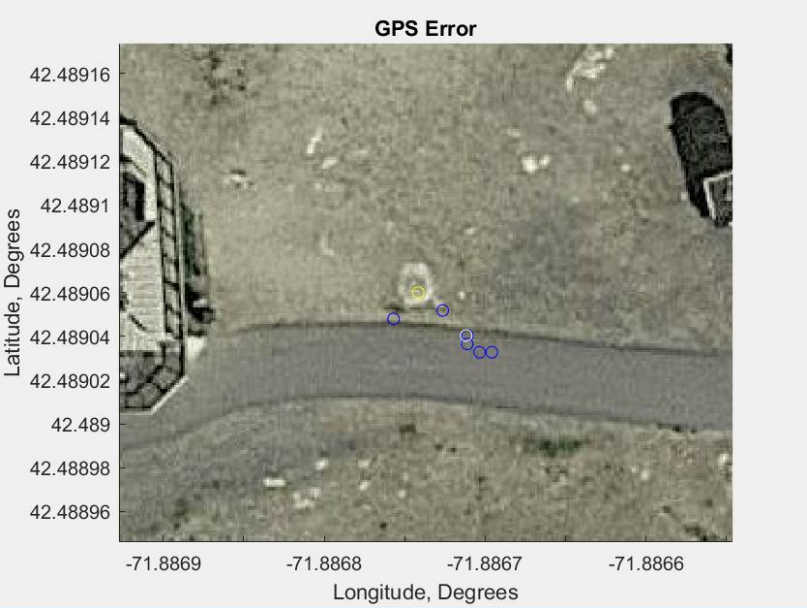

Figure 6-2: Location Distribution from GPS #1

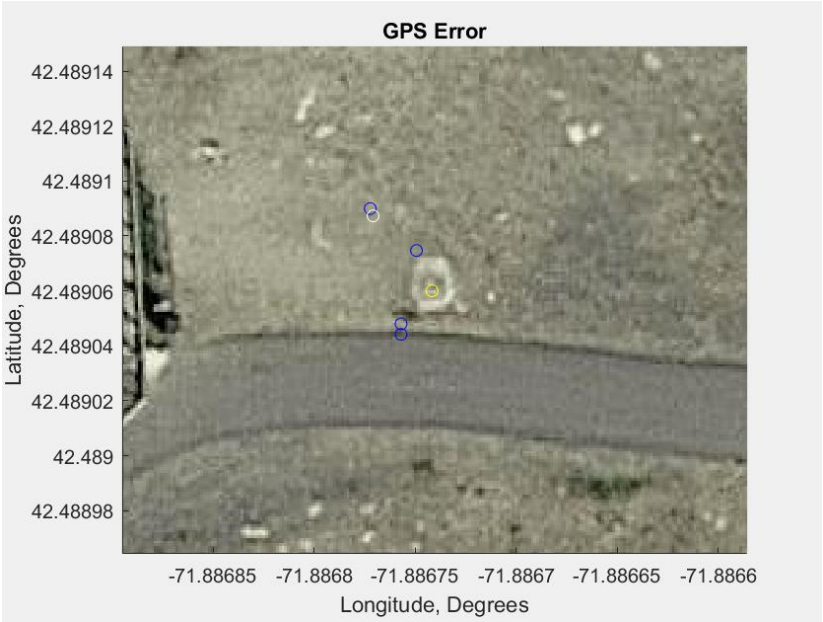

Figure 6-3: Location Distribution from GPS #2

As seen in Figure 6-2, the GPS data represented by the blue markers tended to congregate southeast of the truth location as represented by the yellow marker. With a standard deviation of 0.258 meters in latitude and 0.735 meters in longitude, the GPS achieved fairly consistent and precise results. In regard to the accuracy of the system, however, the mean GPS location was 2.9

meters away from the USGS marker. As such, the GPS possessed a clear offset and thus was moderately inaccurate.

As the bias of the GPS was assumed to be consistent, it was expected that Figure 6-3 would produce similar results to those of Figure 6-2. Unexpectedly, however, the performance of the GPS changed. Although the standard deviation values differed, with 1.107 meters in latitude and 0.544 meters in longitude, these parameters still indicated a highly precise system. Additionally, the accuracy of the system achieved comparable results as in Figure 6-2, as the mean GPS location was 4.4 meters away from the USGS marker. The major difference between the two tests, however, was the direction of the bias.

As seen in Figure 6-2, the bias translated the GPS location southeast of the actual USGS marker. Meanwhile, in Figure 6-3, the bias translated the GPS location northwest of the actual USGS marker. From these data collected, the GPS appeared to undergo a Rician distribution. Due to the presence of a drift value, however, the behavior of the receiver regularly changed; thus making the randomness of the GPS difficult to model. As such, no action was taken in order to attempt to offset the translational error caused by the GPS.

90 Degree Tests:

As illustrated in Figures 6-4 and 6-5, all 90 degree tests consisted of two observers whose pointing vectors formed a 90 degree angle about the blue marker (the object of interest). Observer A, as represented by the red marker, was located 100 feet away from the object of interest oriented 180 degrees (due South). Meanwhile, observer B, as represented by the green marker, was located 98 feet away from the object of interest oriented 270 degrees (due West). Among the many variations of the test performed, those of most importance were: user versus tripod and flat versus portrait.

Figure 6-4: Illustration of 90 Degree Test

90 Degree Test			
	Observer A	Observer B	Target
Location	100 Feet From Target	98 Feet From Target	42.4890544777778 Deg., -71.8867415611111 Deg.
Orientation (including magnetic declination)	180 Degrees	270 Degrees	N/A

Figure 6-5: Metrics of 90 Degree Test

Results: User versus Tripod

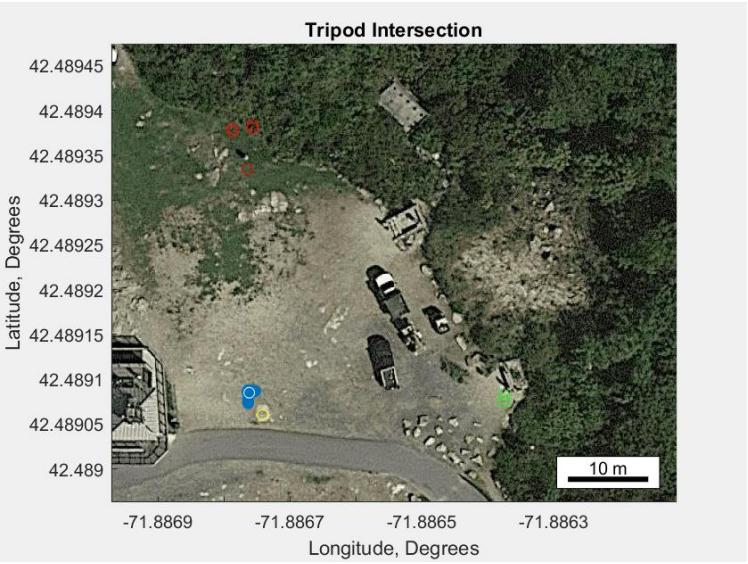

Figure 6-6: Estimated Intersection from Tripod

Figure 6-6 illustrates the tripod test performed with a 90 degree rotation between the pointing vectors of observer A and observer B. Observer A, as represented by the red markers, was located via the smartphone's GPS receiver. Additionally, observer B, as represented by the green markers, was located via the same smartphone GPS receiver. At each observer location, a 5-foot-tall tripod was stationed with the smartphone held upright in the portrait orientation. Using both the compass and the camera of the smartphone to achieve accurate alignment with the object of interest, the indirect geolocation system ultimately achieved a spread of intersection estimates. Represented by the cloud of blue markers, this distribution was then averaged in order to help determine system accuracy. As seen by the white marker, this average was then compared to the yellow marker, the location of the object of interest. For further understanding of the metrics corresponding to Figure 6-6, refer to Figure 6-7.

Mean Latitude of Intersections (Degrees)	42.489084950459770
Mean Longitude of Intersections (Degrees)	-71.886761720937670
Standard Deviation of Intersection: Latitude, Longitude (meters)	0.389, 0.401
Standard Deviation of Observer A: Latitude, Longitude (meters)	0.942, 0.630
Standard Deviation of Observer B: Latitude, Longitude (meters)	0.083, 0
Mean Observer A GPS Location Distance To Truth Versus Actual Distance (meters)	36.3 versus 30.48
Mean Observer B GPS Location Distance To Truth Versus Actual Distance (meters)	30.2 versus 29.8704
Haversine Distance From Mean Intersection To Truth (meters)	3.8
Observer A Angle Error Using Mean GPS Location (Degrees)	3.661
Observer B Angle Error Using Mean GPS Location (Degrees)	4.733

Figure 6-7: Metrics of 90-Degree Test: Tripod

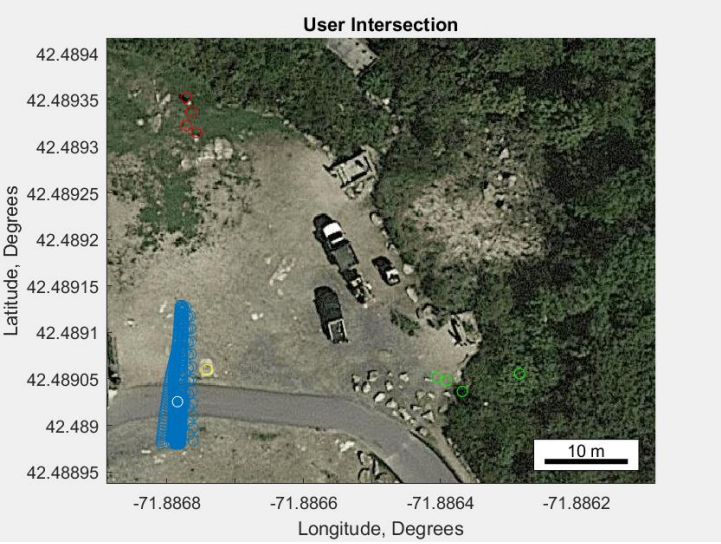

Figure 6-8: Estimated Intersection from User

Figure 6-8 illustrates the user test performed with a 90 degree rotation between the pointing vectors of observer A and observer B. Following the same variable representation as seen in Figure 6-6, observer A and observer B were represented by the red and green markers, respectively. Their estimated intersection points, based on heading and observer location, were then represented as the blue cloud of markers. The mean of the intersection cloud was

represented by the white marker, which could then be compared to the yellow marker, the USGS marker, to determine system accuracy.

Unlike the tripod test illustrated in Figures 6-6 and 6-7, however, the user test had less control. Without the stability of the tripod, the phone was held in portrait mode by the user. Leveraging the phone's camera, the user attempted to increase precision and accuracy by aligning the object of interest to the center of the screen. For further understanding of the metrics corresponding to Figure 6-8, refer to Figure 6-9.

Mean Latitude of Intersections (Degrees)	42.489025447656740
Mean Longitude of Intersections (Degrees)	-71.886784587850170
Standard Deviation of Intersection: Latitude, Longitude (meters)	3.995, 0.363
Standard Deviation of Observer A: Latitude, Longitude (meters)	0.635, 0.265
Standard Deviation of Observer B: Latitude, Longitude (meters)	0.226, 2.178
Mean Observer A GPS Location Distance To Truth Versus Actual Distance (meters)	30.1 vs 30.48
Mean Observer B GPS Location Distance To Truth Versus Actual Distance (meters)	27.9 vs 29.8704
Haversine Distance From Mean Intersection To Truth (meters)	4.8
Observer A Angle Error Using Mean GPS Location (Degrees)	8.766
Observer B Angle Error Using Mean GPS Location (Degrees)	2.653

Figure 6-9: Metrics of 90-Degree Test: User

Analysis: User versus Tripod

The user versus tripod tests provided a means of quantifying human error. In any real-world scenario, there is a large number of random variables that can affect the outcome of an experiment. Of the many, one of the most influential is the impact of human error. When performing a geolocation test, the user, among many flaws, is prone to temporary distraction and shaking. As such, both the accuracy and precision of the system can be reduced, when integrating human error into testing. When using the tripod, however, many of the undesired negative effects of human error are mitigated. Possessing greater stability and not prone to the

many variables of human life, the tripod provides an element of consistency and accuracy unique to the real world.

The performance of the tripod test can be seen from the illustration in Figure 6-6 and the corresponding data in Figure 6-7. Meanwhile, the performance of the user test can be seen from the illustration in Figure 6-8 and the corresponding data in Figure 6-9. Among the many metrics that categorized the system's overall performance in the two tests, those of most importance were precision and accuracy.

In terms of precision, the tripod test performed drastically better. Reflected by the tight cluster of blue markers in Figure 6-6, the indirect geolocation system produced highly precise location estimates. Experiencing one sigma standard deviations of 0.389 meters in latitude and 0.401 meters in longitude, the several thousand intersection points were highly reproducible.

On the contrary, the user test performed with significantly less precision. As seen in Figure 6-8, the intersection estimates created a significantly larger cluster. Supported by the metrics shown in Figure 6-9, the user test experienced one sigma standard deviation bounds of 3.995 meters in latitude and 0.363 meters in longitude.

Due to the implementation of user error, the decrease in precision was expected. The unique distribution of the intersection estimates, however, was not expected. The intersection estimates from the tripod case, as shown in Figure 6-6, followed a relatively circular distribution. In support of theory, the circular distribution of the data points indicated the lack of correlation between the latitude and longitude intersection estimates when the two observers were perpendicular to one another.

Despite the increased horizontal heading error within the user test, the system was expected to also generate a circular distribution of data. As the implemented human error was assumed to be zero-mean Gaussian, the intersection cluster was supposed to be less compact, but still circular. Unexpectedly, however, the user test created high levels of imprecision in latitude. Reflected by the comparatively large standard deviation value, 3.995 meters, the intersection estimates formed a long and narrow data distribution.

Among the many potential reasons as to why the standard deviation value of the latitude was so large, one of the most likely was the inaccuracy of the magnetometer. Given the static

nature of the test (angular rate equaled zero), the EKF was heavily dependent on the accuracy of the magnetometer. As such, any perturbations or noise values corrupting the magnetometer also corrupted the estimated orientation accuracy. Consequently, this increased error extended the one sigma bounds of the intersection distribution.

Due to the severity of the increase in latitude uncertainty, there were likely additional faults in the system. With a more refined and robust magnetometer, however, the distribution would become much more circular in nature.

In terms of accuracy, the difference in performance between the two systems was less clear. Using the Haversine formula to calculate the distance between a pair of latitude, longitude coordinates, the absolute distance from the mean intersection value to the truth location was used to determine system accuracy. Not accounting for GPS drift, the tripod test produced a mean intersection estimation 3.8 meters away from the truth value. Meanwhile, for the user test, the mean intersection estimation was located 4.8 meters away from the truth value. Unfortunately, such offsets were larger than desired. By taking the GPS locational bias into account though, the errors can decrease substantially.

Ultimately, from the 90 degree static tests, two main limitations were found. First in terms of precision, the gyroscope provided little to no insight regarding the orientation of the smartphone. As such, the EKF became largely dependent on one sensor, the magnetometer, to produce heading estimations. Although frequently accurate, the magnetometer was subject to random perturbations disturbing the system. As this was the only sensor that provided heading data, however, the precision of the system was limited by the precision of the magnetometer. Second in terms of accuracy, the occurrence of GPS drift created a translational offset error for the estimated intersection points. Furthermore, as the nature of the GPS drift was difficult to characterize, this offset subsequently induced absolute error randomly.

Results: Flat versus Portrait

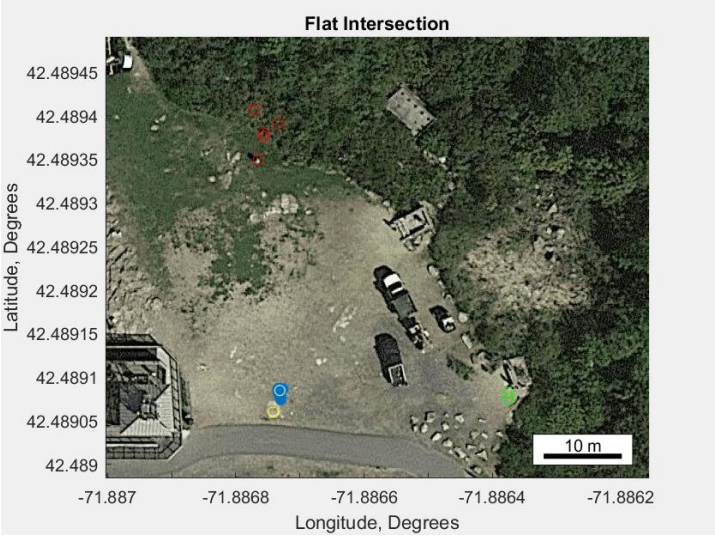

Figure 6-10: Estimated Intersection from Flat Orientation

Figure 6-10 illustrates the flat orientation test performed with a 90 degree rotation between the pointing vectors of observer A and observer B. Following the same variable representation as seen in Figures 6-6 and 6-8, this test helped quantify the accuracy and precision of the system depending on the orientation of the phone. In order to perform the test, the phone was placed in the tripod on its back. As zero pitch and roll were desired, a level was used to help ensure that the plane of the back of the phone was parallel to the two-dimensional plane of the ground. Finally for accurate heading, the high accuracy compass was used in order to ensure that the top of the phone was aligned to the object of interest. For further understanding of the metrics corresponding to Figure 6-10, refer to Figure 6-11.

Mean Latitude of Intersections (Degrees)	42.489084754891020
Mean Longitude of Intersections (Degrees)	-71.886731271597920
Standard Deviation of Intersection: Latitude, Longitude (meters)	0.358, 0.119
Standard Deviation of Observer A: Latitude, Longitude (meters)	0.741, 0.472
Standard Deviation of Observer B: Latitude, Longitude (meters)	0.083, 0
Mean Observer A GPS Location Distance To Truth Versus Actual Distance (meters)	35.8 vs 30.48
Mean Observer B GPS Location Distance To Truth Versus Actual Distance (meters)	30.2 vs 29.8704
Haversine Distance From Mean Intersection To Truth (meters)	3.5
Observer A Angle Error Using Mean GPS Location (Degrees)	7.764
Observer B Angle Error Using Mean GPS Location (Degrees)	4.739

Figure 6-11: Metrics of 90 Degree Test: Flat

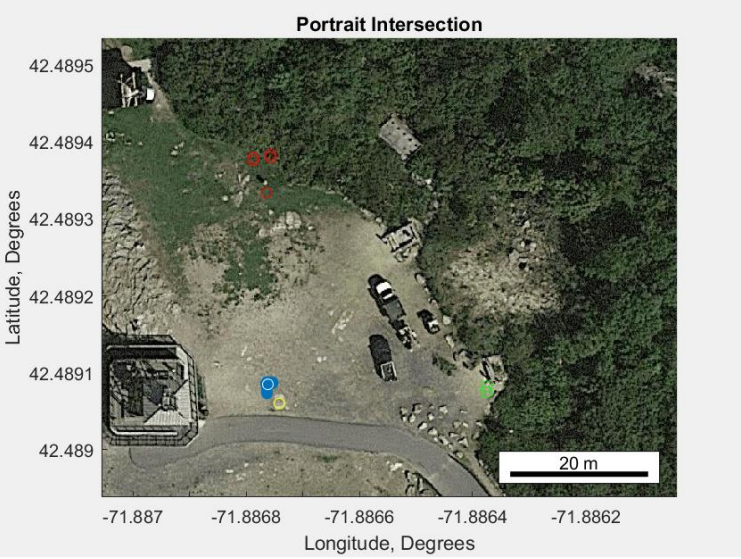

Figure 6-12: Estimated Intersection from Portrait Orientation

Figure 6-12 illustrates the portrait orientation test performed with a 90 degree rotation between the pointing vectors of observer A and observer B. Identical to the data collected in the 90 degree tripod test from Figure 6-5, the metrics corresponding to Figure 6-12 are found in Figure 6-13.

Mean Latitude of Intersections (Degrees)	42.489084950459770
Mean Longitude of Intersections (Degrees)	-71.886761720937670
Standard Deviation of Intersection: Latitude, Longitude (meters)	0.389, 0.401
Standard Deviation of Observer A: Latitude, Longitude (meters)	0.942, 0.630
Standard Deviation of Observer B: Latitude, Longitude (meters)	0.083, 0
Mean Observer A GPS Location Distance To Truth Versus Actual Distance (meters)	36.3 vs 30.48
Mean Observer B GPS Location Distance To Truth Versus Actual Distance (meters)	30.2 vs 29.8704
Haversine Distance From Mean Intersection To Truth (meters)	3.8
Observer A Angle Error Using Mean GPS Location (Degrees)	3.661
Observer B Angle Error Using Mean GPS Location (Degrees)	4.733

Figure 6-13: Metrics of 90 Degree Test: Portrait

Analysis: Flat versus Portrait

As can be seen in Figures 6-10 to 6-13, the indirect geolocation system achieved comparable results in both the flat and portrait tests. In terms of precision, comparison of each test's intersection sigma bounds indicated the similarity in performance. Meanwhile, in terms of accuracy, both systems reported comparable results (3.8 meters versus 3.5 meters). A theme seen in all of the analyzed flat versus portrait tests, the similarity in performance between the two phone orientations indicated that the phone orientation was a parameter that no longer had to be monitored.

Prior to testing, however, there were concerns with location estimation using portrait mode. Specifically, there was uncertainty as to the ability of the system to track multiple degrees of freedom. When the phone is positioned flat, both its roll and pitch values are set to approximately zero. Thus, the system relies almost entirely on the magnetometer. When the phone is positioned in a portrait orientation, however, more sensor fusion must occur. Using the accelerometer as a tilt sensor, the rotation matrix between the phone's body frame and that of the inertial frame must be determined. Thus, the increased data manipulation creates more opportunity for error.

60 Degree Tests:

As illustrated in Figure 6-14 and Figure 6-15, the 60 degree test consisted of two observers whose pointing vectors formed a 60 degree angle about the object of interest. Observer A, as represented by the red marker, was located 92 feet away from the object of interest at an angle of 210 degrees. Meanwhile, observer B, as represented by the green marker, was located 98 feet away from the object of interest at an angle of 270 degrees.

Additionally, all 60 degree tests utilized the same variable representation as seen in the 90 degree tests. As a result, red markers represented observer A, green markers represented observer B, blue markers represented the estimated geolocation points, the white marker represented the mean of the estimated geolocation points, and the yellow marker represented the truth location.

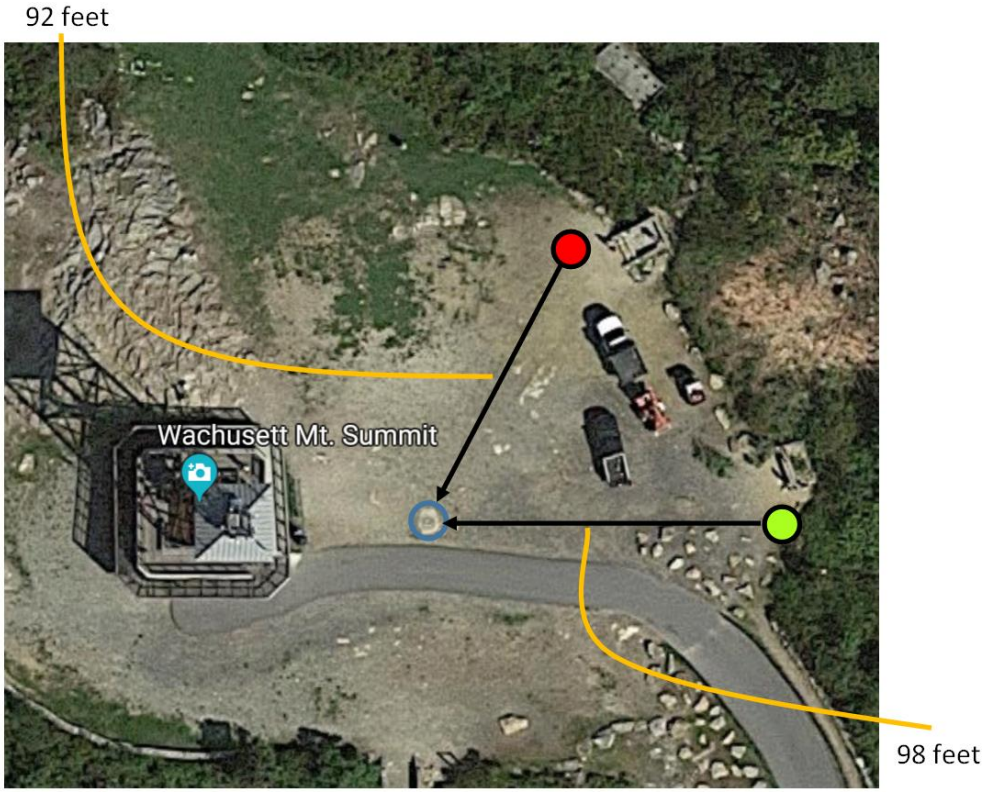

Figure 6-14: Illustration of 60 Degree Test

60 Degree Test			
	Observer A	Observer B	Target
Location	92 Feet From Target	98 Feet From Target	42.4890544777778 Deg., -71.8867415611111 Deg.
Orientation (including magnetic declination)	210 Degrees	270 Degrees	N/A

Figure 6-15: Metrics of 60 Degree Test

Results: User versus Tripod

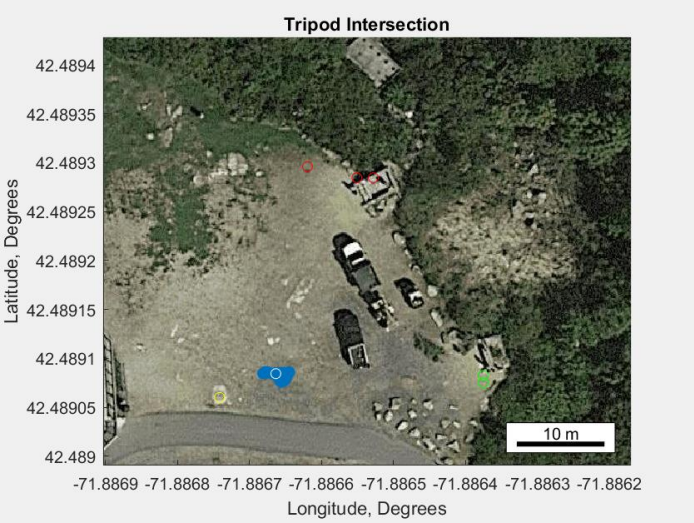

Figure 6-16: Estimated Intersection from Tripod

Following an identical procedure to the tripod test for the 90 degree case, the phone was placed upright onto the tripod. Additionally, in order to increase system accuracy and precision, the tripod was aligned with the object of interest by utilizing both the high accuracy compass and the smartphone camera. For further understanding of the metrics corresponding to Figure 6-16, refer to Figure 6-17.

Mean Latitude of Intersections (Degrees)	42.489084316553100
Mean Longitude of Intersections (Degrees)	-71.886664341803380
Standard Deviation of Intersection: Latitude, Longitude (meters)	0.287, 1.320
Standard Deviation of Observer A: Latitude, Longitude (meters)	0.165, 1.079
Standard Deviation of Observer B: Latitude, Longitude (meters)	0.083, 0
Mean Observer A GPS Location Distance To Truth Versus Actual Distance (meters)	30.0 vs. 28.0416
Mean Observer B GPS Location Distance To Truth Versus Actual Distance (meters)	30.2 vs 29.8704
Haversine Distance From Mean Intersection To Truth (meters)	7.1
Observer A Angle Error Using Mean GPS Location (Degrees)	10.119
Observer B Angle Error Using Mean GPS Location (Degrees)	4.744

Figure 6-17: Metrics of 60 Degree Test: Tripod

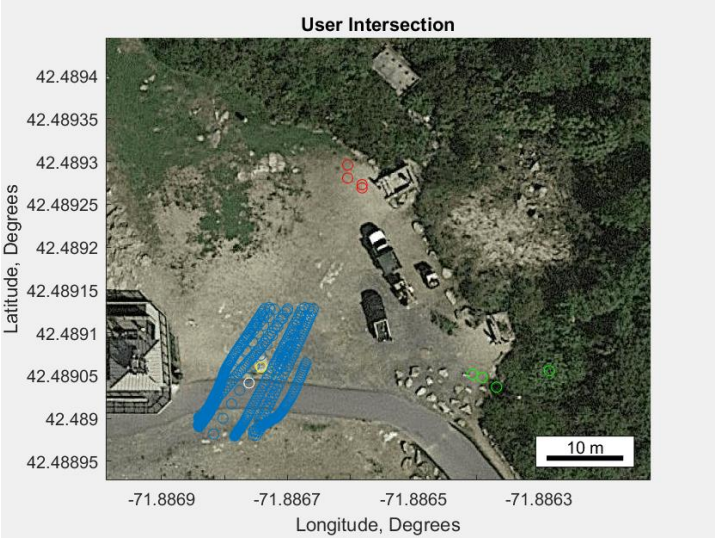

Figure 6-18: Estimated Intersection from User

Figure 6-18 illustrates the user test performed with a 60 degree rotation between the pointing vectors of observer A and observer B. Following the same procedure as outlined in the 90 degree user test, the observer leveraged both the high accuracy compass and the phone's camera to achieve as accurate and precise of intersection estimates as possible. For further understanding of the metrics corresponding to Figure 6-18, refer to Figure 6-19.

Mean Latitude of Intersections (Degrees)	42.489041442320630
Mean Longitude of Intersections (Degrees)	-71.886760869365990
Standard Deviation of Intersection: Latitude, Longitude (meters)	4.236, 4.412
Standard Deviation of Observer A: Latitude, Longitude (meters)	0.358, 0.528
Standard Deviation of Observer B: Latitude, Longitude (meters)	0.226, 2.178
Mean Observer A GPS Location Distance To Truth Versus Actual Distance (meters)	27.6 vs. 28.0416
Mean Observer B GPS Location Distance To Truth Versus Actual Distance (meters)	27.9 vs. 29.8704
Haversine Distance From Mean Intersection To Truth (meters)	2.1
Observer A Angle Error Using Mean GPS Location (Degrees)	2.664
Observer B Angle Error Using Mean GPS Location (Degrees)	1.965

Figure 6-19: Metrics of 60-Degree Test: User

Analysis: User versus Tripod

Based on the same reasoning as in the 90 degree user versus tripod test, the 60 degree tripod test, as illustrated in Figures 6-16 and 6-17, experienced significantly better precision than the user test. Illustrated in Figure 6-17, the tripod test experienced an intersection estimate standard deviation of 0.287 meters in latitude and 1.320 meters in longitude. Meanwhile, the user test, as seen in Figure 6-19, experienced an intersection estimate standard deviation of 4.236 meters in latitude and 4.412 meters in longitude.

While the difference in precision between the two tests was expected, the difference in accuracy of the two tests was unique. As seen in Figures 6-16 and 6-17 of the tripod test, the system achieved a mean location estimation 7.1 meters away from the truth location. For the user test, however, as seen in Figures 6-18 and 6-19, the system achieved a mean location estimation 2.1 meters away from the truth location. Drastically better than that of the tripod test, the system achieved greater accuracy in the test subject to human error.

As both tests experienced many random variables, it is difficult to fully explain the reasoning as to why the user test achieved more accurate results than the tripod. Most likely,

however, the comparative success of the user test was due to the random nature of the GPS drift that corrupted observer locations.

As prior stated, the static and user tests performed at the 60 degree orientation were very static. Consequently, the orientation estimation of the phone became largely dependent on the accuracy of the magnetometer. Despite the heavy pre-calibration and filtering undergone by the magnetometer, the device was prone to error and inconsistency. As such, slightly inaccurate heading estimations from the magnetometer were possible.

As any potentially random GPS bias or small angle offset could alter the accuracy of the system, it was therefore not unlikely for an alternative test to perform more accurately, despite the presence of human error.

135 Degree Tests:

As illustrated in Figure 6-20 and Figure 6-21, the 135 degree test consisted of two observers whose pointing vectors formed a 135 degree angle about the object of interest. Observer A, as represented by the red marker, was located 94 feet away from the object of interest at an angle of 135 degrees. Meanwhile, observer B, as represented by the green marker, was located 98 feet away from the object of interest at an angle of 270 degrees.

Figure 6-20: Illustration of 135 Degree Test

135 Degree Test			
	Observer A	Observer B	Target
Location	94 Feet From Target	98 Feet From Target	42.4890544777778 Deg., -71.8867415611111 Deg.
Orientation (including magnetic declination)	135 Degrees	270 Degrees	N/A

Figure 6-21: Metrics of 135 Degree Test

Results of User versus Tripod:

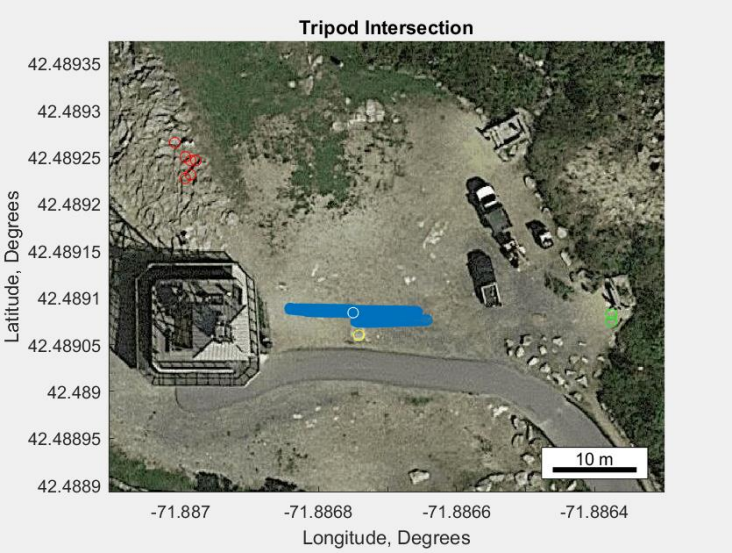

Figure 6-22: Estimated Intersection from Tripod

Figure 6-22 illustrates the tripod test performed with a 135 degree rotation between the pointing vectors of observer A and observer B. For further understanding of the metrics corresponding to Figure 6-22, refer to Figure 6-23.

Mean Latitude of Intersections (Degrees)	42.489083949368750
Mean Longitude of Intersections (Degrees)	-71.886749956984730
Standard Deviation of Intersection: Latitude, Longitude (meters)	0.412, 5.978
Standard Deviation of Observer A: Latitude, Longitude (meters)	0.708, 0.384
Standard Deviation of Observer B: Latitude, Longitude (meters)	0.083, 0
Mean Observer A GPS Location Distance To Truth Versus Actual Distance (meters)	28.2 vs. 29.2608
Mean Observer B GPS Location Distance To Truth Versus Actual Distance (meters)	30.2 vs. 29.8704
Haversine Distance From Mean Intersection To Truth (meters)	3.3
Observer A Angle Error Using Mean GPS Location (Degrees)	4.716
Observer B Angle Error Using Mean GPS Location (Degrees)	3.604

Figure 6-23: Metrics of 135 Degree Test: Tripod

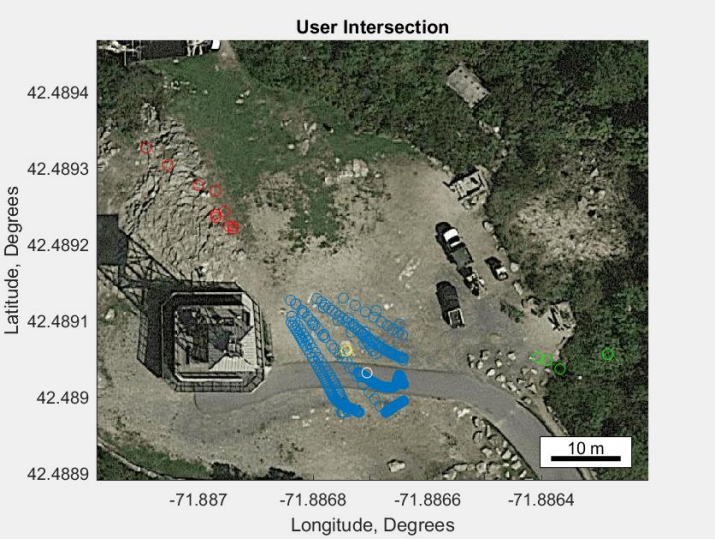

Figure 6-24: Estimated Intersection from User

Figure 6-24 illustrates the user test performed with a 135 degree rotation between the pointing vectors of observer A and observer B. For further understanding of the metrics corresponding to Figure 6-24, refer to Figure 6-25.

Mean Latitude of Intersections (Degrees)	42.489031525133880
Mean Longitude of Intersections (Degrees)	-71.886705749412030
Standard Deviation of Intersection: Latitude, Longitude (meters)	3.56, 6.343
Standard Deviation of Observer A: Latitude, Longitude (meters)	2.774, 3.346
Standard Deviation of Observer B: Latitude, Longitude (meters)	0.226, 2.178
Mean Observer A GPS Location Distance To Truth Versus Actual Distance (meters)	28.0 vs. 29.2608
Mean Observer B GPS Location Distance To Truth Versus Actual Distance (meters)	27.9 vs. 29.8704
Haversine Distance From Mean Intersection To Truth (meters)	3.9
Observer A Angle Error Using Mean GPS Location (Degrees)	1.942
Observer B Angle Error Using Mean GPS Location (Degrees)	0.762

Figure 6-25: Metrics of 135 Degree Test: User

Analysis of User versus Tripod

As seen in Figures 6-22 and 6-23 the tripod test achieved modest precision and high accuracy. Similar to the user test in the 90 degree scenario, the 135 degree tripod test generated large standard deviation values in one direction. Represented by the flat and wide distribution of blue markers, the intersection estimate had significant longitudinal uncertainty. Among the many potential reasons for explaining the system's longitudinal imprecision, the most likely explanation was the magnetometer. As prior mentioned, the magnetometer was prone to small angle errors. A trend seen in many other tests, the performance of the magnetometer appeared to take on a random distribution. Therefore, for some orientation estimations, such as the one for observer B in Figure 6-22, the magnetometer produced a relatively constant angular output. While in others, the magnetometer produced less accurate and consistent data.

Meanwhile, in the user test, the system behaved much more normally. As expected, the user test produced a much more imprecise intersection distribution. Specifically, by superimposing the uncertainty correlated with human error onto the already existing data distribution map, the data formed the distribution as modeled by the blue markers in Figure 6-24. Ideally, the distribution would have been more angular and ellipsoidal than circular due to the correlation between axes at 135 degrees.

Lastly in terms of accuracy, both tests experienced comparable results. As seen in Figure 6-23, the tripod test achieved a mean intersection location 3.3 meters away from the USGS marker. Meanwhile for the user test, the addition of user error did little to affect the accuracy of the system. As such, the system achieved a mean intersection location 3.9 meters away from the USGS marker.

6.3 Dynamic Tests: Methods and Results

The static tests performed and analyzed in Section 6.2 provided valuable insight into the effectiveness of the system under ideal scenarios. In order to test the real world feasibility of an indirect geolocation system using a smartphone, however, dynamic tests also had to be performed. Among the many dynamic tests performed, the test from Figure 6-26 well represents the system's results.

As seen in Figure 6-26, observer A, the red marker, underwent significant motion. Much more comparable to a real-world scenario, observer A first began at the left-most red marker. Moving slowly from each red marker to the next, observer A attempted to keep the hand-held phone aligned with the object of interest. Once observer A reached the location of observer B, the green marker, observer A then turned around and followed the same path back to the starting point. As the user attempted to aim the smartphone while also walking in a different direction, a significant decrease in both accuracy and precision was expected. Consequently, observer A tried to best mitigate these increased errors by operating the phone in portrait mode, so as to use the camera to maintain relatively constant positioning on the target.

Results: Dynamic Test

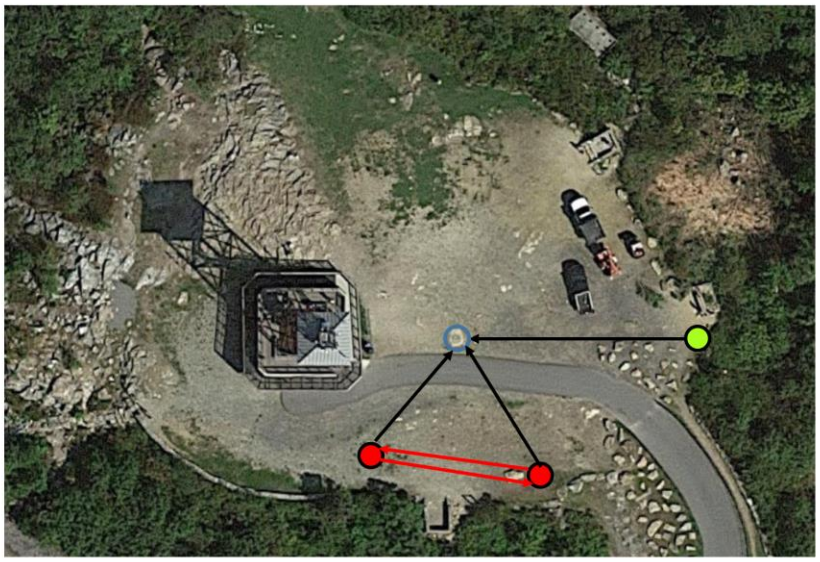

Figure 6-26: Illustration of Dynamic User Test

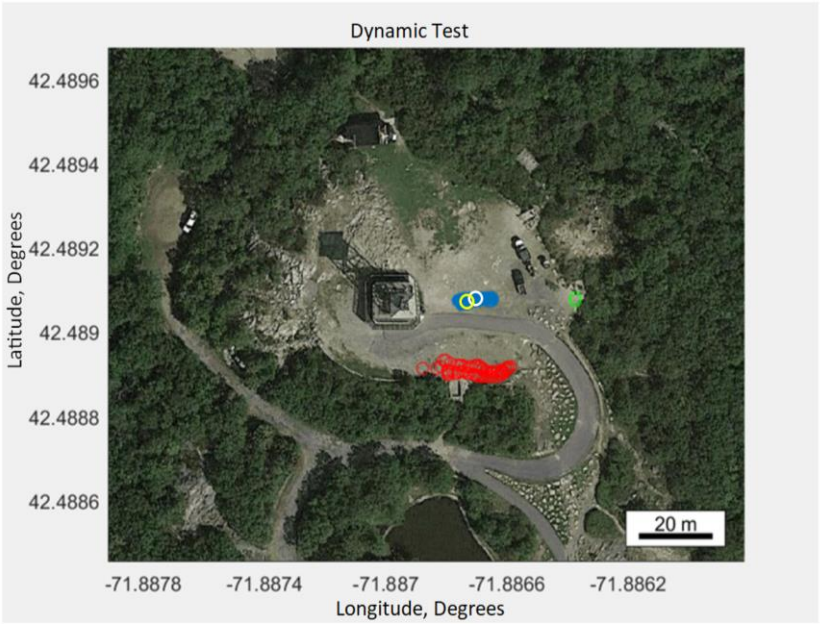

Figure 6-27: Estimated Intersection from Dynamic User Test

Figure 6-27 illustrates the dynamic user test. Observer A, as represented by the red markers, followed the path as illustrated in Figure 6-26. Meanwhile, observer B, as represented by the green markers, remained stationary with the phone placed in the tripod in portrait mode.

Ultimately, the resulting intersection estimations created a spread of data as represented by the blue markers. With a mean estimation location represented by the white marker, this location was then compared to the yellow marker, the truth location, in order to determine system accuracy. For further metrics describing Figure 6-27, refer to Figure 6-28.

Mean Latitude of Intersections (Degrees)	42.489061038924648
Mean Longitude of Intersections (Degrees)	-71.886723034926578
Standard Deviation of Intersection: Latitude, Longitude (meters)	0.73, 3.12
Haversine Distance From Mean Intersection To Truth (meters)	1.6

Figure 6-28: Metrics of Dynamic Test

Analysis: Dynamic Test

As seen in Figures 6-27 and 6-28, the dynamic test performed successfully. In terms of precision, the one sigma standard deviation bounds were larger than several of the static experiments. Despite the decrease in precision, however, this decrease in performance was expected. Unlike the static tests from Section 6.2, the dynamic test underwent a much more complicated data collection process. As the observer was both walking and turning, while also attempting to keep the phone aligned with the object of interest, there was a lot of new error introduced into the system. The ability of the EKF to leverage all sensor data, however, kept the system relatively precise. In terms of accuracy, the dynamic test uniquely achieved more accurate results than the static tests. Most likely due to the more frequent updates of the GPS receiver, the influence of GPS drift seemed to be somewhat mitigated.

Summary:

		$\sigma_{latitude}$ (meters)	$\sigma_{longitude}$(meters)	Absolute Error (meters)
90 Degrees	Tripod	0.389	0.401	3.8
	User	3.995	0.363	4.8
	Flat	0.358	0.119	3.5
	Portrait	0.389	0.401	3.8
60 Degrees	Tripod	0.287	1.320	7.1
	User	4.236	4.412	2.1
135 Degrees	Tripod	0.412	5.978	3.3
	User	3.56	6.343	3.9
Dynamic	User	0.73	3.12	1.6

Figure 6-29: Summary of Calculated Test Values

To summarize, Figure 6-29 restates many of the calculated test values from each of the prior mentioned experiments. Specifically, Figure 6-29 presents the standard deviation latitude, standard deviation longitude, and absolute error of the estimated intersection points. An interesting trend in the data was the difference in performance between the static and dynamic tests. For the static tests, the absolute errors were frequently 10 to 20% of the observer distance from the object of interest (about 30 to 35 meters). Meanwhile, for the dynamic test, the absolute error was approximately 6%. In terms of precision, both systems performed reasonably well. Under most circumstances, the static tests achieved tight data distributions representing the appropriate degree of correlation between the observers. Furthermore, the intersection estimates often accurately modeled the addition of human error, which most likely embodied a zero-mean normal distribution. Meanwhile, for the dynamic test, the precision well represented the nature of the test. Most uncertain in terms of longitude, the distribution of the intersection data helped indicate that the observer was walking at approximately the same latitude coordinates.

Chapter 7: Discussion

This chapter discusses the results and findings for the two-dimensional intersection algorithm, three-dimensional intersection algorithm, field tests, and future work.

Two-Dimensional Intersection Algorithm

The two-dimensional (2-D) intersection algorithm was validated using a zero-error model, where analytical results were compared to actual results obtained from the algorithm. A range of errors from the zero-error intersection solution was observed to be from zero to 1.404e-11 meters. The only scenario where zero error in the intersection solution was calculated was when there was no decimal approximation of the input location and orientation for both observers. When a precision to four decimal places was used, errors up to 1.404e-11 were observed. With such small errors observed due to precision, the algorithm was considered validated.

To characterize the performance of the two-dimensional algorithm, Monte Carlo simulations were used. When errors were introduced to both observers' locations and orientations, an intersection point cloud was formed; two characteristics of this point cloud were analyzed. First, was the standard deviation of the point cloud along the x-axis and y-axis, which described the precision of the intersection solution. Second, was the distribution of the root-mean-square (RMS) difference between each intersection point and the zero-error value. This error was commonly referred to as RMS error in this report. For each Monte Carlo simulation there were four total parameters: location error, orientation error, distance, and angle between observers. The two error parameters were a distribution of zero-mean Gaussian noise with a specified standard deviation. The distance and angle between observers was varied according to each test and were essentially test parameters.

In Section 5.1.2, a series of simulations was used to examine the performance of the intersection algorithm while an observer moved away from the object of interest. These simulations were repeated for different error parameters, including a position error of 2.5 m and 0.001 m for a range of orientation errors from 0.0° to 2.0°. These values corresponded to the standard deviation of the zero-mean Gaussian distribution used to introduce the error. Two main observations were made after examining these scenarios.

First, as an observer's distance increased and there was orientation, the intersection mean RMS error also increased. When there was no orientation error, the mean RMS error remained the same at all distances. The position error acted as a bias in the intersection mean RMS error. For the test where the position error was 2.5 m and the orientation error was 1.0°, the orientation error was observed to have a larger effect on the overall RMS error as the distance increased. From Figure 5-16, the position error at 100 m consisted of 13.7% of the total RMS error, but at 880 m, the position error only accounted for 2.9% of the total RMS error.

In Section 5.1.3, another series of simulations was used to examine not only the distance of the observers, but also the angle between both observers. During the series the observers were placed at a range of distances from 25 m to 350 m in 25 m increments. At each distance increment, the angle between observers was swept through from 20° to 160° in 5° increments. These simulations showed that the optimal angle to obtain the most accurate intersection point was an angle between observers of 90°. These simulations also supported the finding that as the distance from both observers to the object increases, the orientation error plays a greater role in the total RMS error than the position error. For the simulation shown in Figure 5-22, the error due to orientation was found to be greater than that of position at a distance greater than 250 m. This simulation consisted of the standard GPS error of 2.5m and orientation error of 1.0°.

Future work in characterizing the 2-D intersection could examine observers with different error distributions, i.e. if one observer has a malfunctioning device, what is the effect on the intersection if the other observer has a working device? Additional errors could also be characterized, in this report only zero-mean Gaussian distributions were used, but another error distribution that was found consistent with location errors found in the field tests for GPS location was the Rician distribution.

Three-Dimensional Intersection Algorithm

The three-dimensional (3-D) intersection algorithm was validated using a zero-error model in MATLAB consisting of simple geometry and vector distance graphs. The algorithm was observed to work correctly and match up with the simple geometry and vector distance graphs. When the observers' vectors had a direct intersection, geometry analysis was used to validate the MATLAB algorithm. The geometry analysis's intersections were observed to match the MATLAB algorithm's intersections. When the observers' vectors did not have a direct

intersection, vector distance graphs were used. Each vector distance graph was created after the MATLAB algorithm computed the intersection point and showed if the MATLAB algorithm found the minimum and equidistant point from both vectors, the two properties of the least-squares pseudo inverse calculation. The resulting intersection approximation from the MATLAB algorithm was found to satisfy the least-squares pseudo inverse calculation traits in every scenario that was tested. After the validation was complete, the intersection algorithm was judged to be working correctly.

To characterize the performance of the 3-D intersection performance with input errors typically observed in the real world, Monte Carlo simulations were used. For each simulation, errors were added and the resulting intersection point cloud was characterized using X, Y, and Z RMS error and total RMS error values. Three overall simulations were conducted. First, the observers' positions were held constant with location and orientation errors and the resulting intersection point cloud was evaluated. Second, location and orientation variables were independently varied and the influences of each error to the resulting intersection point cloud was observed. Third, the observers' angles and distances to the object were varied to observe the optimal angle that produced the most accurate intersection point cloud, and determined the influence between location and orientation error.

In Section 5.2.1, the simulations examined typical real world errors for each observer and the relationship between the shape of the intersection point cloud as position and angle configurations changed for each observer. These simulations were used to observe that when the two observers' vectors intersect at a 90 degree angle, the intersection point cloud had equal X and Y distributions, with the Z distribution being dependent on the pitch angle and difference in Z position between observers. As the angle between observers became increasingly acute or obtuse, yaw angle error increasingly affected the distribution of the intersection cloud. As the difference in Z position changed for the observers, the intersection point cloud occurred halfway between the two vectors in the Z position due to the vector approximation. These intersection scenarios and distribution observations gave the ability determine what errors and uncertainty might be seen in a field test.

In Section 5.2.2, the simulations examined the influences of each error to the resulting intersection point cloud. Each simulation involved increasing noise on one variable for one observer, while the other observer had no location or orientation error, in order to individually

characterize the effect of each variable to the resulting intersection. As the X and Y position errors become greater, the intersection X and Y RMS error values increased linearly, and the Z RMS error remained constant. When the Z position error became greater, the intersection Z RMS error value increased linearly, and the X and Y RMS error remained constant. When the yaw error was increased, the intersection X and Y RMS error values increased linearly, and the Z RMS error remained constant. Lastly, when the pitch error increased, the intersection Z RMS error increased linearly, and the X and Y RMS error value was observed to increase at a slower rate than the Z RMS error, but still increase linearly with the pitch error. These findings provided the ability to determine the root cause of abnormal intersection uncertainty distributions.

In Section 5.2.3, the simulations examined the optimal intersection angle which produced the most precise intersection point cloud, and determined the influence between location and orientation error as distance increased. Each simulation consisted of a starting point 50 m from the intersection point and sweeping the angles between observers from 20° to 160°. The resulting mean RMS error versus angle between observers was observed and the distance was incremented by 50 m until 350 m was reached. The minimum RMS error was found to occur around 90°, and the minimum error remained at 90° as distance from the object increased. Next, the individual effect of location and orientation error on the intersection point cloud was compared as distance increased. First, the location error was tested without orientation error, with the same test conditions as the optimal angle simulation. From the simulation, it was observed that as distance increased, the RMS absolute error remained the same. As the distance increased with orientation error, however, the RMS absolute error increased as well. Using this observation, it was found that at a distance smaller than 250 m, the location error had more of an effect on the resulting intersection point cloud distribution. When the distance is greater than 250 m, the orientation error had a larger effect on the resulting point cloud distribution. This distance can be used to determine major factor for intersection uncertainty cloud for particular configurations.

Overall, the 3-D intersection was validated using geometry, vector distance graphs, and MATLAB intersections, and the simulations were used to observe the characteristics of the algorithm with real world errors added. This information was used to ensure the algorithm would work correctly during field tests and the simulations were used to compare field test outcomes.

Field Tests Discussion:

One of the fundamental goals of this project was to show the feasibility of a smartphone-based indirect geolocation system. In order to deem the smartphone implementation as "feasible," no real metrics were set. In general the goal was to have the system achieve accuracy levels within approximately 10% of the average observer distance from the truth marker. If most importance, however, were two additional sub-goals. First, an algorithm that could generate an orientation estimate had to be created. Among its many tasks, this algorithm had to take potentially poor sensor data and create a coherent, optimized result. Second, a connection had to be established between simulations and field tests. By showing that the field tests followed many of the same trends as seen in the simulations, the performance of the system would become more legitimate and trustworthy. Ultimately, completion of these two sub-goals would indicate that smartphones possess the qualities necessary to theoretically perform indirect geolocation.

Sub-goal One: Creation of an Orientation Algorithm

To satisfy the first sub-goal, an extended Kalman Filter (EKF) was created. Designed to be a sensor fusion algorithm, the EKF created a coherent orientation estimate from individual sensors. In real-world static tests, the EKF regularly generated orientation estimates within a degree of the estimated truth (refer to Section 6.2). Furthermore, in real-world dynamic tests, the EKF seemingly produced accurate results although there was no estimated truth comparison (refer to Section 6.3). Despite the success of the EKF, however, there were several problems with its design.

For one, the magnitude of the individual sensor errors produced by the phone was unexpected. Consequently, it took longer than expected to design a system that could accurately fuse all sensor data together. Although the EKF naturally eliminated a lot of the uncertainty and error within individual sensor data, there was a tremendous need for pre-filtering and pre-calibration. As seen in Section 4.1, the data underwent NaN correction, high-pass filters, low-pass filters, and calibrations prior to entering the EKF. Thus, it took longer than expected to customize each filtering and calibration stage to optimize the sensor data.

In addition to the unexpected length of time it took to accurately fuse the data, the EKF also required extensive customization for fusion in each individual test. Using a tuning parameter which altered the process noise, the behavior of the EKF would change. Although customization

of this parameter is helpful in optimizing the system based on the dynamics of the test, the tuning parameter required more frequent calibration than desired to obtain an optimal result. Ultimately, the creation of a sensor fusion algorithm that generated an accurate orientation estimation was a success. The system took longer to create than expected and did not act as autonomously as desired, however, it produced accurate orientation results given the scenarios it experienced. In order to further optimize the system, however, it should undergo more real-world testing, particularly tests exploiting edge cases. Such testing would help determine the rigidity of all stages of the filter and further characterize the phone sensors.

Sub-goal Two: Connection Between Simulations and Field Testing

As seen in Sections 5.1 and 5.2, many simulations were performed to reflect both 2-D and 3-D indirect geolocation scenarios. Naturally, the simulations were unable to reflect the distribution of random variables in real-world testing. Despite this limitation, however, the simulations did provide theoretical justification for ideal real-world scenarios. Among the many results generated from the simulations in Sections 5.1 and 5.2, two of the most important were the 90 degree optimal angle analysis and the location versus orientation error analysis.

The 90 degree optimal angle analysis was a result generated in both 2-D and 3-D cases. Specifically, it stated that the absolute error, measured by the root-mean-square (RMS) of the distance between the estimated intersection point and the truth, was a minimum when the angle between the two observers was 90 degrees. When comparing the simulated result with the result obtained in field testing, there were both similarities and differences. In terms of precision, the field test modeled the behavior seen in simulation. Specifically, the field test produced a very small, nearly circular intersection distribution which indicated the lack of correlation between errors from the two observers. In terms of accuracy, however, there was an inconsistency in results. As can be seen by the various testing results shown in Section 6.2, the 90 degree test did not produce the most accurate orientation estimate. This inconsistency with the simulated results most likely was a bi-product of the many random variables that plague real-world tests. Among the many random variables unmodeled in simulation, however, the most significant influence on the inaccuracy of the real-world system was most likely the GPS location error.

Undergoing some form of drift, the GPS uncertainty, as seen in Section 6.2, did not model the zero-mean Gaussian distribution used in simulations. Consequently, the simulations never accounted for a data set in which the mean observer location value experienced some form

of offset. An error prevalent in many of the real-world tests performed, this GPS bias ultimately created inaccuracies in the intersection estimations.

Using the location versus orientation error analysis generated in both the 2-D and 3-D simulations, the influence of the inaccurate GPS readings could be measured. Specifically, the analysis stated that observer location inaccuracy created more intersection estimate error than did orientation inaccuracy when both observers were relatively close to the object of interest. Therefore, as restrictions in space made the field tests relatively small in size (approximately 30 to 35 meters), the inaccurate GPS locations negatively affected the intersection accuracy more than the orientation error. As such, reduction of the GPS error would most likely produce field test results that better modeled the accuracy generated by the simulations.

Main Goal:

Ultimately, as seen in Sections 6.2 and 6.3, the main accuracy goals were met, however, not with the consistency desired. As seen in the 90 degree tripod test, there were cases in which static tests achieved approximately 10% error. Due to the significant GPS bias, however, further optimization of such scenarios was often difficult. Meanwhile, for dynamic tests, the system uniquely achieved greater accuracy. Helping mitigate GPS drift by forcing the GPS to more regularly update, the system was no longer as severely limited by the performance of the GPS. As such, the ability of the EKF to generate accurate orientation estimations produced more accurate intersection results.

Having acknowledged such limitations in experimental success, future work should increase the distance used in field tests. As a result, the influence of the location error should decrease and the system will become much more subject to the accuracy of the EKF. Furthermore, future work should acquire a more refined GPS receiver. Such an acquisition should not be difficult though, as smartphones will most likely soon possess more accurate GPS receivers as technology improves. Future work could also look for a method in correcting the orientation error of the smartphones using the smartphone camera.

Chapter 8: Conclusion

In summary, this report examined the effectiveness of using smartphones as an indirect geolocation system. With multiple known observation vectors representing individual smartphone's "lines of sight," two-dimensional (2-D) and three-dimensional (3-D) intersection algorithms were created, validated, and characterized for indirect geolocation. Leveraging the existing hardware and software found within ordinary smartphones, an Extended Kalman Filter (EKF) was designed to accurately calculate these smartphone poses'. The algorithms and EKF were then subjected to field tests to introduce real-world error into the system and characterize performance.

Several observations were made after characterizing the 2-D and 3-D intersection algorithms. First, with orientation error present in the input, it was found that the intersection mean RMS error increases with distance. Second, it was found that the effect of the position error on the mean RMS error is independent of distance, and acts as a bias. Third, at longer distances the mean RMS error due to orientation error dominates over that due to position error. Fourth, it was found that the intersection angle of 90° minimizes the mean RMS error of the intersection cloud. For a standard location error of 2.5 m and orientation error of 1.0°, an angle from 75° to 105° only produces 2-3% additional mean RMS error in the intersection compared to minimum mean RMS error value at 90°.

The field tests introduced real-world errors into the EKF and 2-D intersection algorithm and assessed the performance of the accuracy of the system. The EKF was designed to be a sensor fusion algorithm, it created a coherent orientation estimate from individual inertial sensors found in smartphones: gyroscope, accelerometer, and magnetometer. In real-world static tests, the EKF regularly generated orientation estimates within a degree of the estimated truth. Although the EKF naturally eliminated a lot of the uncertainty and error within individual sensor data, there was a tremendous need for pre-filtering and pre-calibration. In order to further optimize the system, however, it should undergo more real-world testing, particularly tests exploiting edge cases and tests at longer distances. Such testing would help determine the rigidity of all stages of the filter, further characterize the phone sensors, and further quantify the performance of the Extended Kalman Filter.

Next, a connection was established between simulations and field tests. When comparing the simulated result with the result obtained in field testing, there were both similarities and differences. In terms of precision, the field test modeled the behavior seen in simulation. Specifically, the field test produced a very small, nearly circular intersection distribution which indicated the lack of correlation between errors from the two observers. In terms of accuracy, however, there was an inconsistency in results. This inconsistency with the simulated results most likely was a bi-product of the many random variables that plague real-world tests.

Ultimately through field tests, the main accuracy goals were met, however, not with the consistency desired. As seen in the 90 degree tripod test, there were cases in which static tests achieved approximately 10% error. Due to the significant GPS bias, however, further optimization of such scenarios was often difficult. Meanwhile, for dynamic tests, the system uniquely achieved greater accuracy. Helping mitigate GPS drift by forcing the GPS to more regularly update, the system was no longer as severely limited by the performance of the GPS. As such, the ability of the EKF to generate accurate orientation estimations produced more accurate intersection results.

References

AsahiKASEI. (2013). AK8963 3-axis Electronic Compass. *Asahi Kasei Microdevices Corporation Datasheets*. https://www.akm.com/akm/en/file/datasheet/AK8963C.pdf

Al-Hamad, A., & El-Sheimy, N. (2014). Smartphones based mobile mapping systems. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, *40*(5), 29.

Apache. (2004). Apache License Version 2.0, January 2004. *The Apache Software Foundation.*

Barrett, J. M., (2014). Analyzing and Modeling Low-Cost MEMS IMUs for use in an Inertial Navigation System. *Masters Theses (All Theses, All Years) submitted to Worcester Polytechnic Institute*. 581. https://digitalcommons.wpi.edu/etd-theses/581

Civera, J., Grasa, O., Davison, A., Montiel, J., Stachniss, C., Williams, S., & Neira, J. (2010). 1-Point RANSAC for extended Kalman filtering: Application to real-time structure from motion and visual odometry. *Journal of Field Robotics*, 27(5), 609–631.

Courrieu, P. (2008). Fast computation of Moore-Penrose inverse matrices. *Neural Information Processing-Letters and Reviews*. 8.

"Demographics of Mobile Device Ownership and Adoption in the United States." Pew Research Center, Washington, D.C. (2018, February 5) http://www.pewinternet.org/fact-sheet/mobile/.

Ding, W., & Wang, J. (2011). Precise Velocity Estimation with a Stand-Alone GPS Receiver. *Journal of Navigation*, 64(2), 311-325.

Djuknic, G. M., & Richton, R. E. (2001). Geolocation and assisted GPS. *Computer*, (2), 123-125.

Dowdy, S., Wearden, S., and Chilko, D. (2004) *Statistics for Research*. Hoboken, New Jersey: John Wiley & Sons. pp. 354-355.

Euler Angles, Quaternions, and Transformation Matrices (pp. 1-42, Rep.). (1977). Houston, TX: NASA.

Fakharian, A., Gustafsson, T., and Mehrfam, M. (2011) "Adaptive Kalman filtering based navigation: An IMU/GPS integration approach," *2011 International Conference on Networking, Sensing and Control*, Delft, pp. 181-185.

Feng, K., Li, J., Zhang, X., Shen, C., Bi, Y., Zheng, T., and Liu, J. (2017) A New Quaternion-Based Kalman Filter for Real-Time Attitude Estimation Using the Two-Step Geometrically-Intuitive Correction Algorithm. *Sensors (Basel, Switzerland). 2017;17(9):2146. doi:10.3390/s17092146.*

FitzGerald, D. and Perrin, J. (2015). Magnetic Compensation of Survey Aircraft; a poor man's approach and some re-imagination. *Conference: Extended Abstracts of 14th SAGA Biennial Technical.* 10.1190/sbgf2015-145.

Gikas, V., & Perakis, H. (2016). Rigorous performance evaluation of smartphone GNSS/IMU sensors for ITS applications. *Sensors*, *16*(8), 1240.

GISGeography. (2018, February 23). *Trilateration vs Triangulation – How GPS Receivers Work.* Image Retrieved from https://gisgeography.com/trilateration-triangulation-gps/

Groves, P. D. (2013). *Principles of GNSS, inertial, and multisensor integrated navigation systems*. Boston, MA: Artech house. pp. 13, 23-136, 137-162.

Guo, S., Wu, J., Wang, Z., & Qian, J. (2017). *Novel MARG-Sensor Orientation Estimation Algorithm Using Fast Kalman Filter* (Vol. 2017, Journal of Sensors, Rep. No. 8542153). Hindawi.

Jin, Y., Toh, H. S., Soh, W. S., & Wong, W. C. (2011, March). A robust dead-reckoning pedestrian tracking system with low cost sensors. In *Pervasive Computing and Communications (PerCom), 2011 IEEE International Conference on* (pp. 222-230). IEEE.

Kazusuke M., "MEMS inertial sensors and their applications," *2008 5th International Conference on Networked Sensing Systems, Kanazawa*, 2008, pp. 71-73.

Kos, A., Tomazic, S., Umek, A., & Kos, A. (2016). Evaluation of Smartphone Inertial Sensor Performance for Cross-Platform Mobile Applications. Sensors, 16(4), 477–477.

Li, C., Zhang, S., Cao, Y. (2013). One new onboard calibration scheme for gimballed IMU, *Measurement, Volume 46, Issue 8,* Pp. 2359-2375, ISSN 0263-2241.

Lorenz, A. (2013). Sensorstream IMU+GPS. *Google Play Applications.*

Madgwick, S. (2010). An efficient orientation filter for inertial and inertial/magnetic sensor arrays. *Report x-io and University of Bristol (UK)*, 25, 113-118.

Magers, M. (2016). Geolocation of RF Emitters Using a Low-Cost UAV-Based Approach. *Thesis presented to the Faculty of the Department of Aeronautics and Astronautics, Air Force Institute of Technology.* Write-Patterson Air Force Base, Ohio. United States of America.

Manon Kok, Jeroen D. Hol and Thomas B. Sch¨on (2017), ”Using Inertial Sensors for Position and Orientation Estimation”, *Foundations and Trends in Signal Processing*: Vol. 11: No. 1-2, pp 1-153. http://dx.doi.org/10.1561/2000000094

The MathWorks (2018). MATLAB Mobile. *Google Play Applications*.

Mendes, E., Lacroix, S., & Sola, J. (2016). Parallax angle parametrization in incremental SLAM. Control, Automation, Robotics and Vision (ICARCV), 2016 14th International Conference on (pp. 1–7). *IEEE.*

Musoff, H., and Zarchan, P. (2009). *Fundamentals of Kalman filtering: a practical approach*. Reston, Virginia: American Institute of Aeronautics and Astronautics. pp. 257-292.

Paromik Chakraborty. (2017). In The News: Global Smartphone Sales up by 9.1 per cent, with Maximum Impact from Chinese Firms. *Electronics Bazaar*. Retrieved from Nexis Uni.

Roetenberg, D., Luinge, H. J., Baten, C. T., & Veltink, P. H. (2005). Compensation of magnetic disturbances improves inertial and magnetic sensing of human body segment orientation. IEEE Transactions on neural systems and rehabilitation engineering, 13(3), 395-405.

Sabatini, A. M. (2006). Quaternion-based extended Kalman filter for determining orientation by inertial and magnetic sensing. IEEE Transactions on Biomedical Engineering, 53(7), 1346-1356.

Sabatini, A. M. (2011). Kalman-filter-based orientation determination using inertial/magnetic sensors: Observability analysis and performance evaluation. *Sensors*, *11*(10), 9182-9206.

Solin, A., Cortes, S., Rahtu, E., and Kannala, J. (2018). Inertial Odometry on Handheld Smartphones. *International Conference on Information Fusion (FUSION 2018)*, Cambridge, UK.

Tan, Q. J. (2015). Passive coherent detection and target location with multiple non-cooperative transmitters. *Thesis presented to the Naval Postgraduate School*. Monterey, CA.

Titterton, D., and Weston, J. L.(2004). *Strapdown inertial navigation technology.* Reston, Virginia: *The American Institute of Aeronautics*. pp. 56.

US Department of Commerce, NOAA, & National Geodetic Survey. (2009, May 27). The NGS Data Sheet. Retrieved from https://www.ngs.noaa.gov/

van Diggelen, F., Enge, P., (2015, September) "The World's first GPS MOOC and Worldwide Laboratory using Smartphones," *Proceedings of the 28th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2015)*, Tampa, Florida, pp. 361-369.

VectorNav. (2016). VN-310 DUAL GNSS/INS. *Tactical Series Product Brochure.*

Vu, H., Palacios, A., In, V., Longhini, P., & Neff, J. D. (2011). A drive-free vibratory gyroscope. *Chaos (Woodbury, N.Y.).* 21. 013103. doi:10.1063/1.3532802.

Woodman, O. J. (2007). An Introduction to Inertial Navigation. *University of Cambridge Computer Library Technical Report,* (ISSN) 1476-2986. 696, 1-37.

Appendix A – 2-D Zero Error Model Scenarios

Scenario	x_A (m)	y_A (m)	theta_A (deg)	x_B (m)	y_B (m)	theta_B (deg)
1	-146.4102	0	30	-100	-100	45
2	100	200	0	200	0	90
3	0	200	0	546.4102	0	150
4	100	373.2051	-60	100	100	45
5	300	373.2051	-120	300	200	0
6	200	500	-90	400	546.4102	-120
7	373.2051	100	150	300	100	135
8	400	-100	123.6901	300	0	116.5651
9	150	-100	80.5376	250	-100	99.4624
10	100	500	-71.5651	0	300	-26.5651

Figure A-1: 2-D Error Model Scenarios

*All scenarios intersect at (200 m, 200 m)

Appendix B - Validation Results of Three-Dimensional Intersection

Below, the validation results from scenarios 5 to 11 and 15 to 16 are detailed.

Scenario 5:

Figure B-1 shows the configuration for scenario 5:

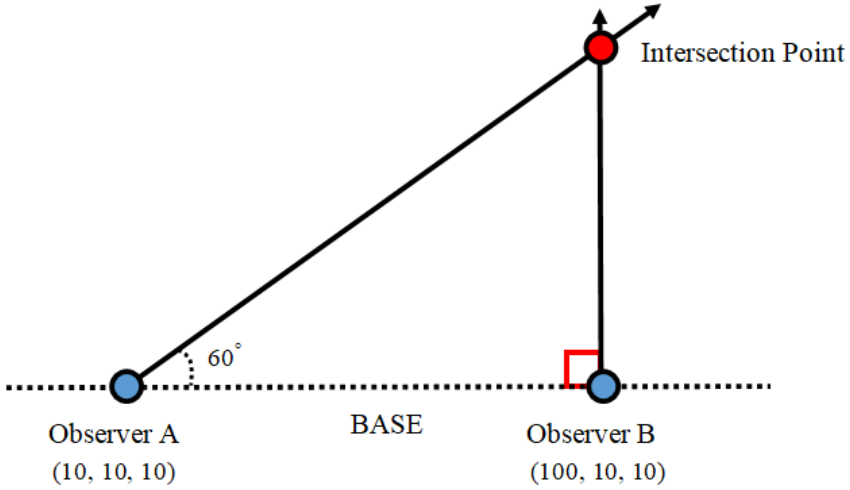

Figure B-1: Scenario 5 Configuration

Using Equation 4-27, the final angle in the triangle can be found. This angle Θ_C can be seen in Figure B-2 below.

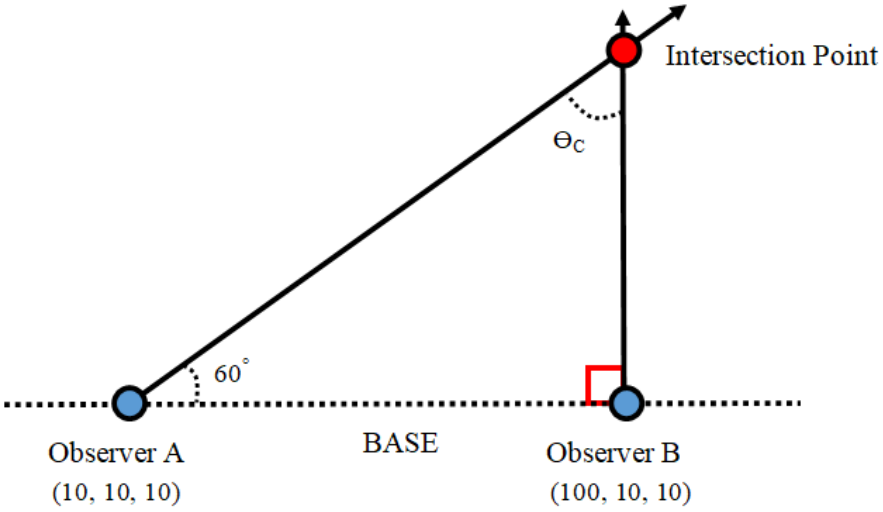

Figure B-2: Scenario 5 with Θ_C Angle

The angle was found to be 30 degrees. Using equation 4-25, the base distance can be found. This distance is found to be 100. With the BASE distance and Θ_C, the distance d_A and d_B can be found using the law of sines. The distances can be seen in the Figure B-3 below.

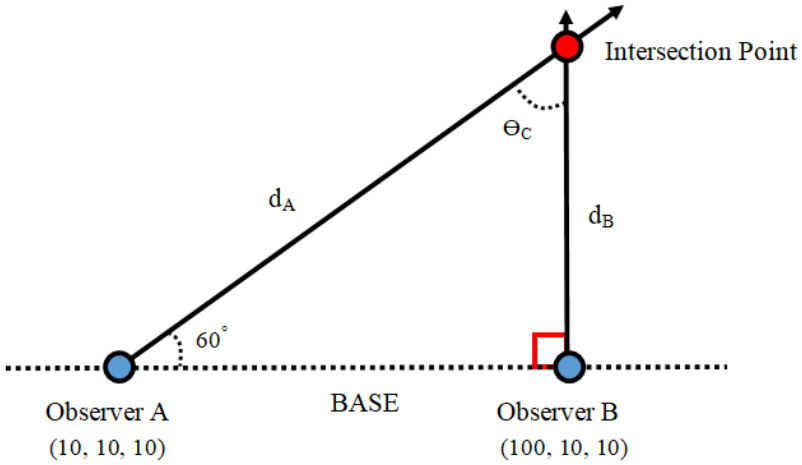

Figure B-3: Scenario 5 with d_A and d_B

Using d_B, the intersection point can be found. This point can be found because there is a right angle formed at observer B, so the distance d_B only contributes to the Y-axis change. The resulting intersection was found to occur at (100, 165.88, 10). Next, the MATLAB zero-error model was analyzed, seen in Figure B-4 and B-5, to observe the resulting intersection.

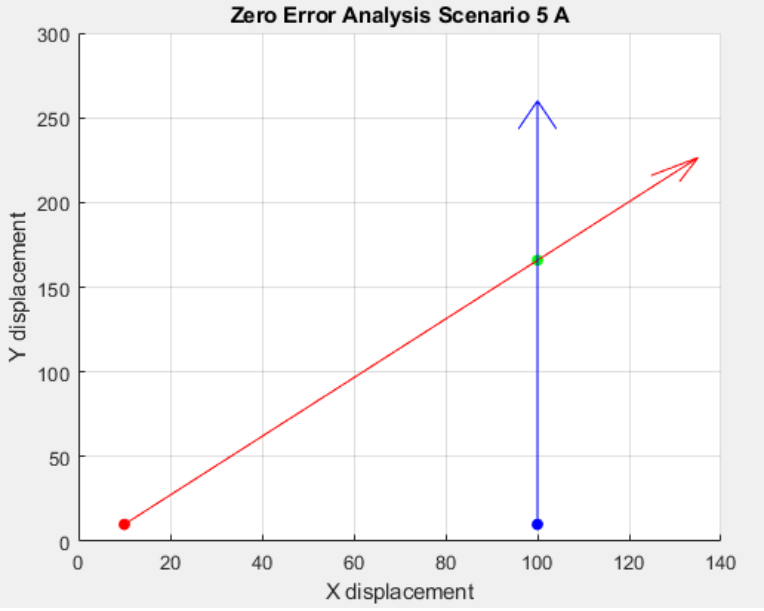

Figure B-4: Scenario 5 MATLAB Zero Error Model View A

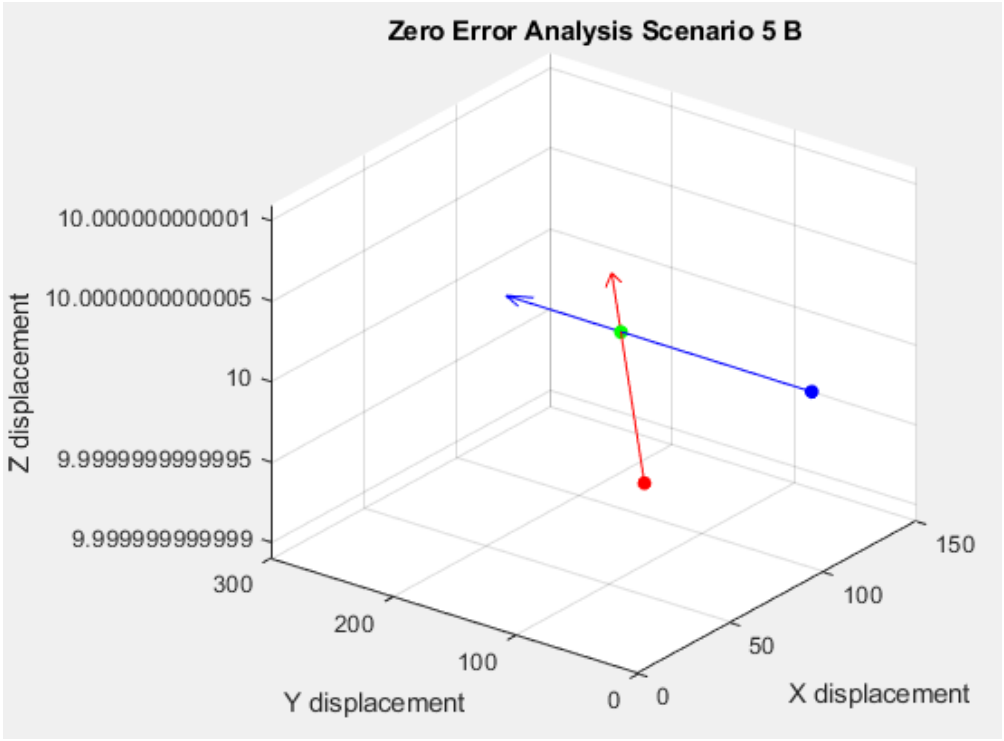

Figure B-5: Scenario 5 MATLAB Zero Error Model View B

The resulting intersection, determined through calculations from the MATLAB zero error model, occurs at (100.0000, 165.8846, 10.0000) and matches up with the geometry analysis.

Scenario 6:

This scenario has a different intersection than scenario 5 due to the two vectors not truly intersecting. Therefore an analysis graph can be seen in Figure B-6 below, with distance from each vector being on the y-axis and the index, or position, along each vector being on the x axis, which allows for the position along each line to be found.

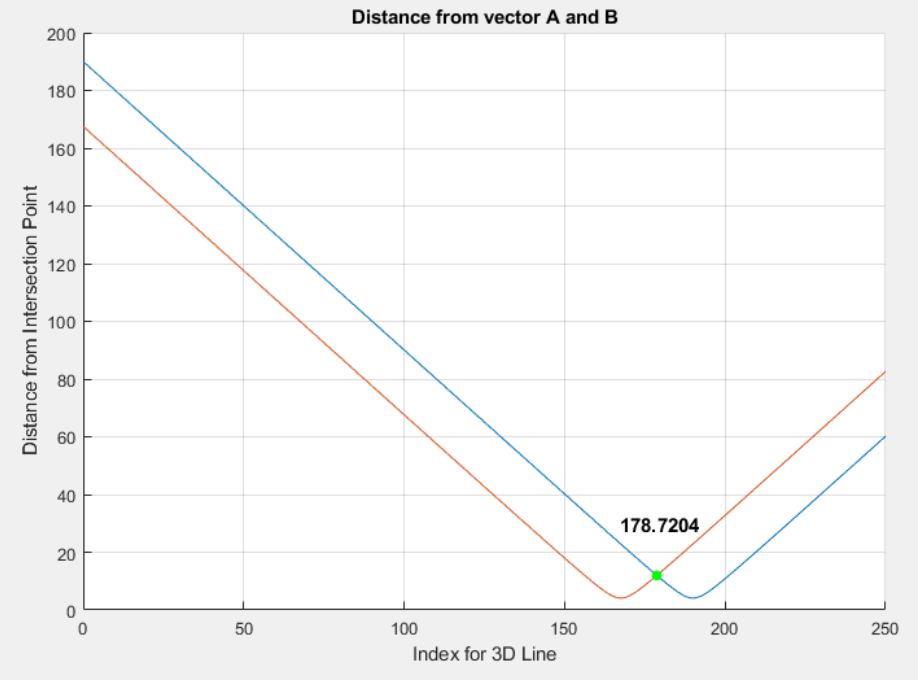

Figure B-6: Scenario 6 Distance to Least Squares Intersection

The position for Vector A at this intersection is (93.9711, 155.4423, 71.1260) and the position for Vector B at this intersection is (100.0000, 177.9423, 71.1260). Next, the scenario was analyzed using MATLAB. The resulting plot can be seen in Figure B-7 and B-8 below.

Figure B-7: Scenario 6 MATLAB Zero Error Model View A

Figure B-8: Scenario 6 MATLAB Zero Error Model View B

The resulting intersection, determined through calculations from the MATLAB zero error model, occurs at (99.6252, 165.9850, 71.1260).

Scenario 7:

This scenario has two vectors not truly intersecting. Therefore an analysis graph can be seen in Figure B-9 below, with distance from each vector being on the y-axis and the index, or position, along each vector being on the x axis, which allows for the position along each line to be found.

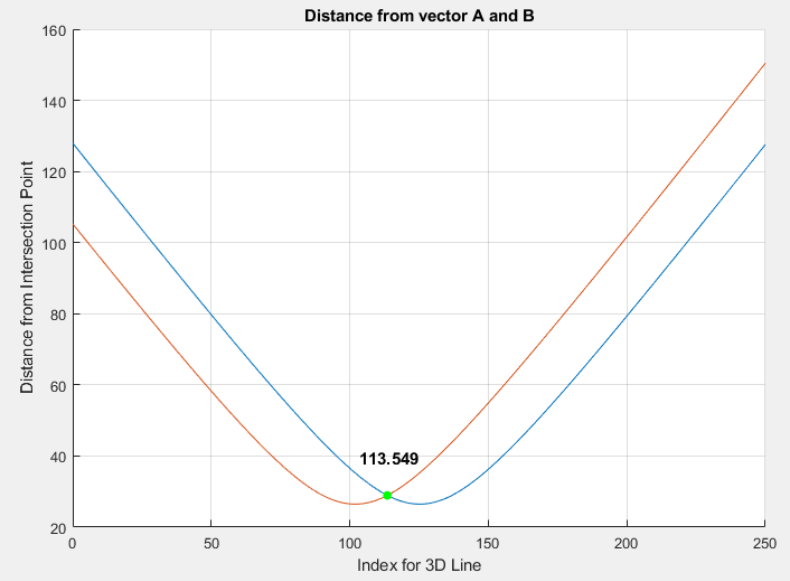

Figure B-9: Scenario 7 Distance to Least Squares Intersection

The position for Vector A at this intersection is (63.3506, 102.4059, 48.8360) and the position for Vector B at this intersection is (100.0000, 123.5490, 10.0000). Next, the scenario was analyzed using MATLAB. The resulting plot can be seen in Figure B-10 and B-11 below.

Figure B-10: Scenario 7 MATLAB Zero Error Model View A

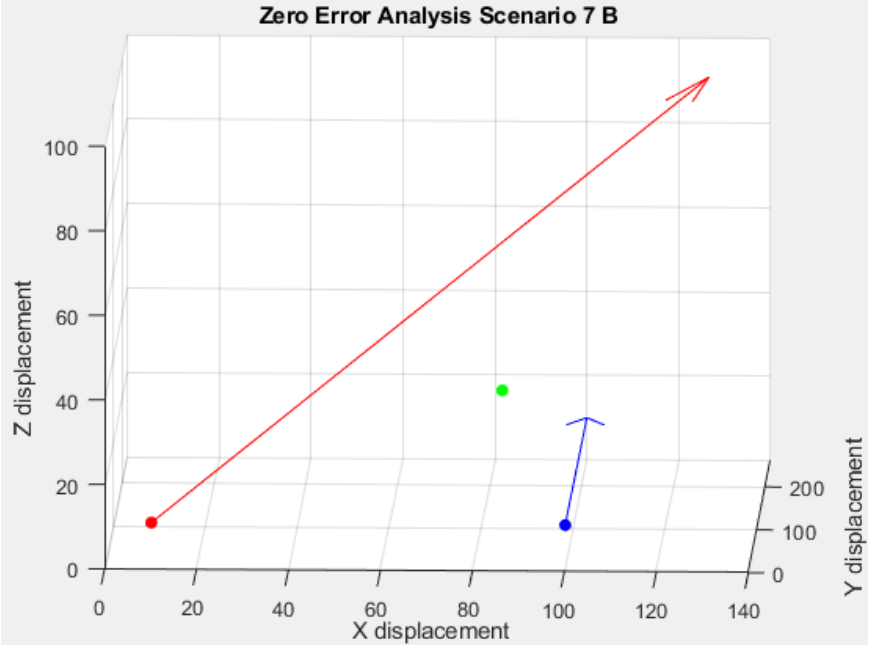

Figure B-11: Scenario 7 MATLAB Zero Error Model View B

The resulting intersection, determined through calculations from the MATLAB zero error model, occurs at (84.4137, 111.8922, 31.4115).

Scenario 8:

This scenario has two vectors not truly intersecting. Therefore an analysis graph can be seen in Figure B-12 below, with distance from each vector being on the y-axis and the index, or position, along each vector being on the x axis, which allows for the position along each line to be found.

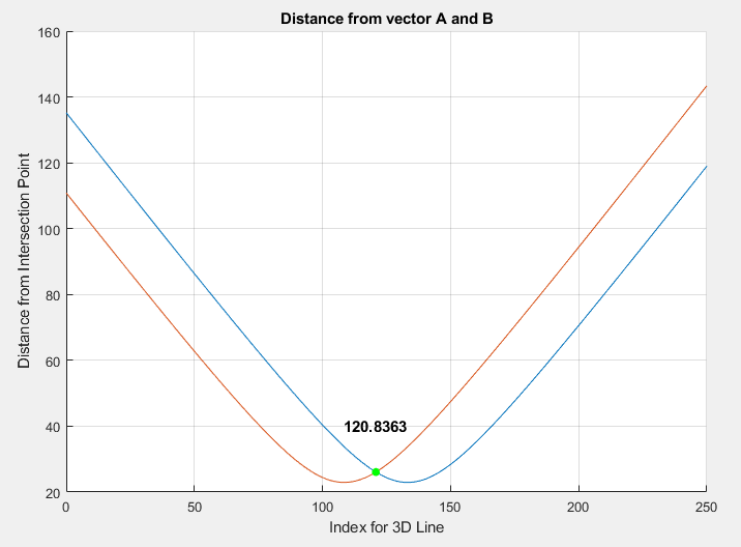

Figure B-12: Scenario 8 Distance to Least Squares Intersection

The position for Vector A at this intersection is (70.4182, 114.6473, 10.0000) and the position for Vector B at this intersection is (100.0000, 123.5490 , 51.3285). Next, the scenario was analyzed using MATLAB. The resulting plot can be seen in Figure B-13 and B-14 below.

Figure B-13: Scenario 8 MATLAB Zero Error Model View A

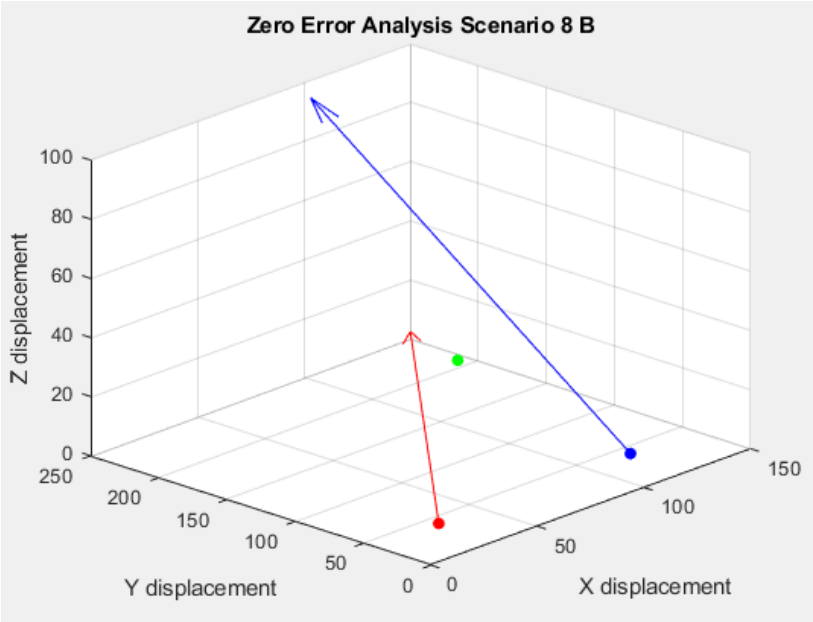

Figure B-14: Scenario 8 MATLAB Zero Error Model View B

The resulting intersection, determined through calculations from the MATLAB zero error model, occurs at (88.3103, 118.6412, 28.5429).

Scenario 9:

The Figure B-15 shows the configuration for scenario 9:

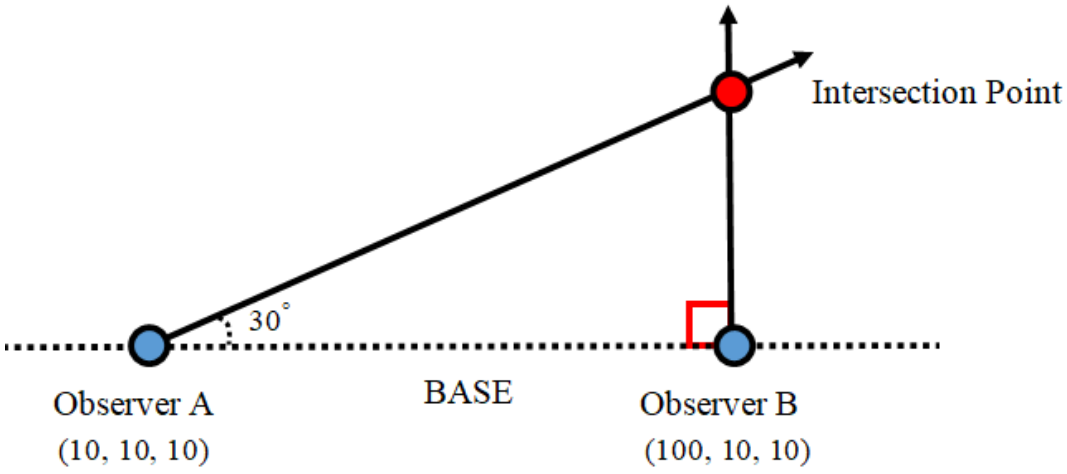

Figure B-15: Scenario 9 Configuration

Using Equation 4-23, the final angle in the triangle can be found. This angle Θ_C can be seen in Figure B-16 below.

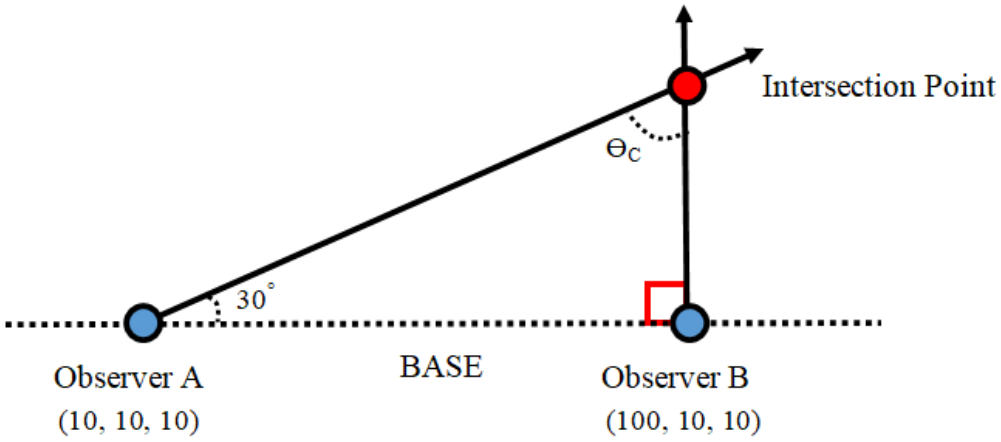

Figure B-16: Scenario 9 with Θ_C Angle

This angle was found to be 60 degrees. Using equation 4-22, the base distance can be found. This distance is found to be 100. With the BASE distance and Θ_C, the distance d_A and d_B can be found using the law of sines. The distances can be seen in the Figure B-17 below.

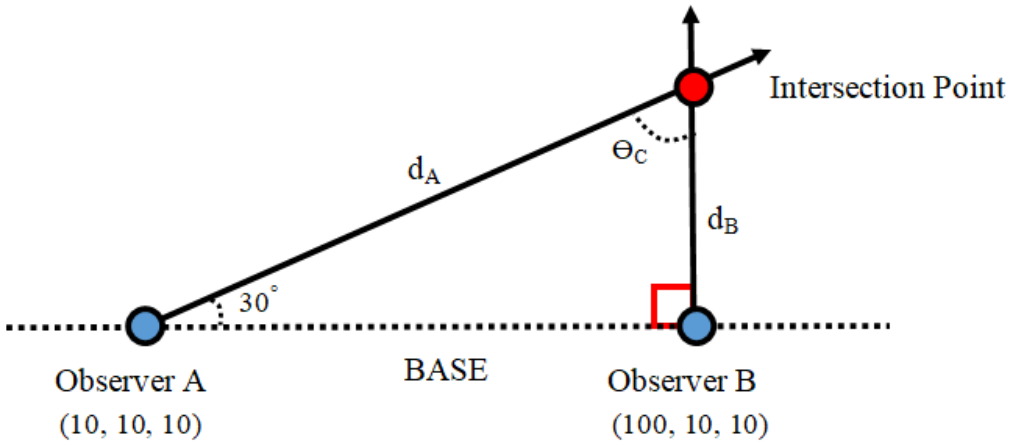

Figure B-17: Scenario 9 with d_A and d_B

Using d_B, the intersection point can be found. This point can be found because there is a right angle formed at observer B, so the distance d_B only contributes to the Y-axis change. The resulting intersection was found to occur at (100, 61.96, 10). Next, the MATLAB zero-error model was analyzed, seen in Figure B-18 and B-19, to observe the resulting intersection.

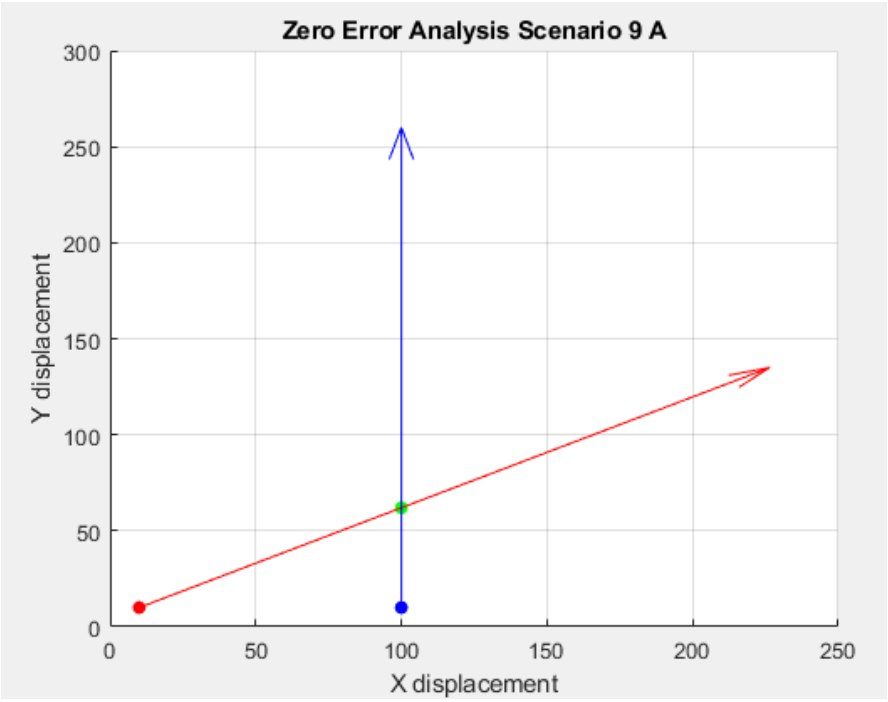

Figure B-18: Scenario 9 MATLAB Zero Error Model View B

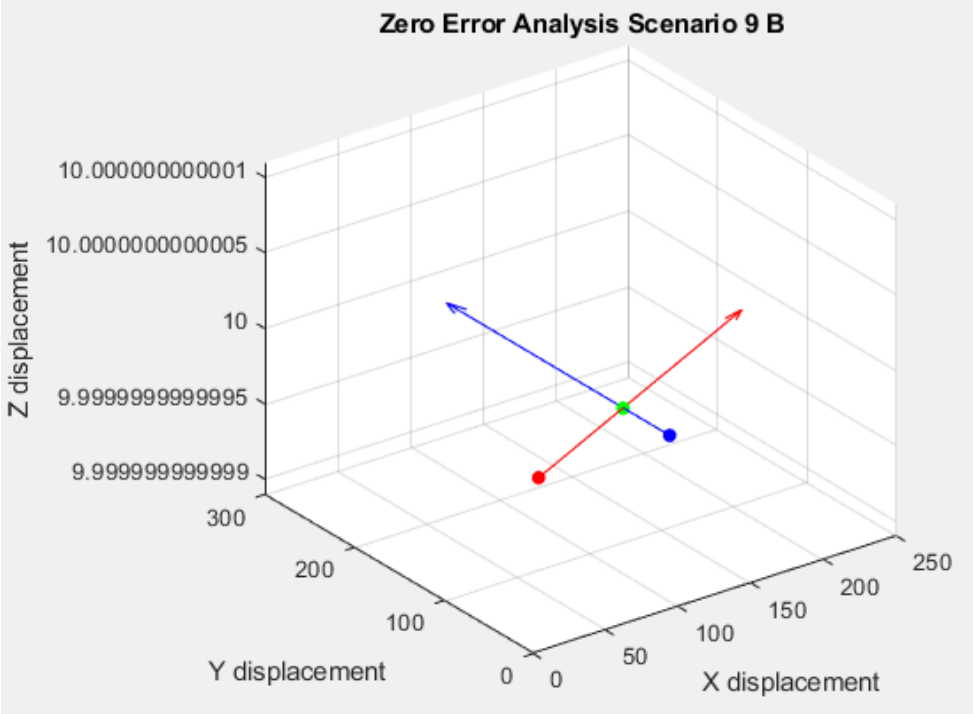

Figure B-19: Scenario 9 MATLAB Zero Error Model View B

The resulting intersection, determined through calculations from the MATLAB zero error model, occurs at (100.0000, 61.9615, 10.0000) and matches up with the geometry analysis.

Scenario 10:

This scenario has two vectors not truly intersecting. Therefore an analysis graph can be seen in Figure B-20 below, with distance from each vector being on the y-axis and the index, or position, along each vector being on the x axis, which allows for the position along each line to be found.

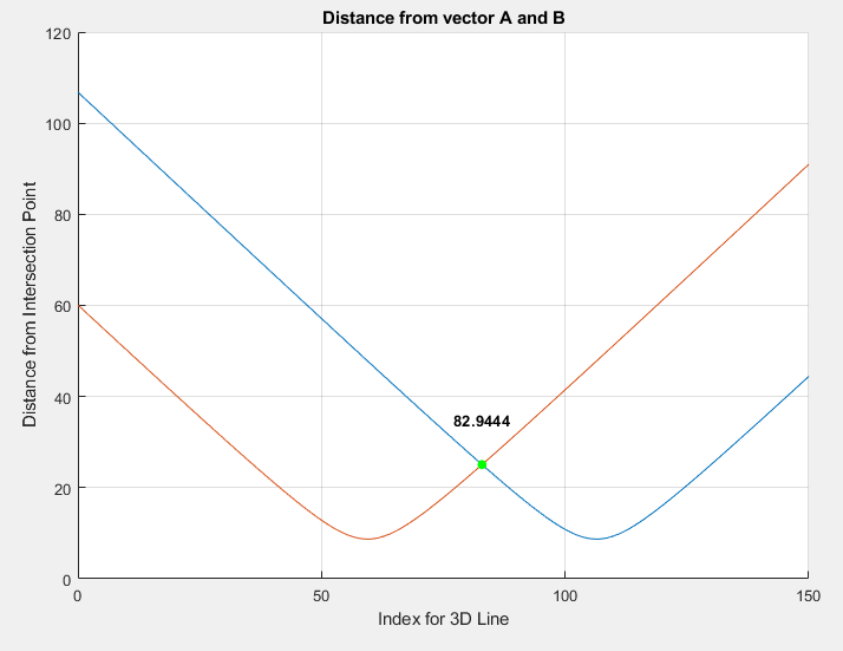

Figure B-20: Scenario 10 Distance to Least Squares Intersection

The position for Vector A at this intersection is (77.5000, 48.9711, 38.3687) and the position for Vector B at this intersection is (100.0000, 87.9423, 38.3687). Next, the scenario was analyzed using MATLAB. The resulting plot can be seen in Figure B-21 and B-22 below.

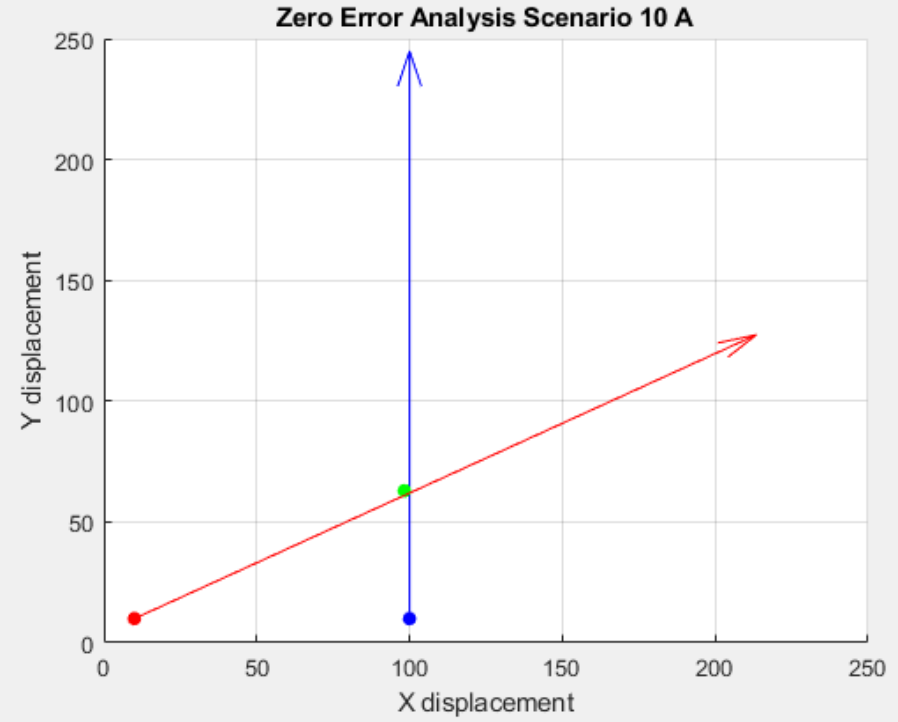

Figure B-21: Scenario 10 MATLAB Zero Error Model View A

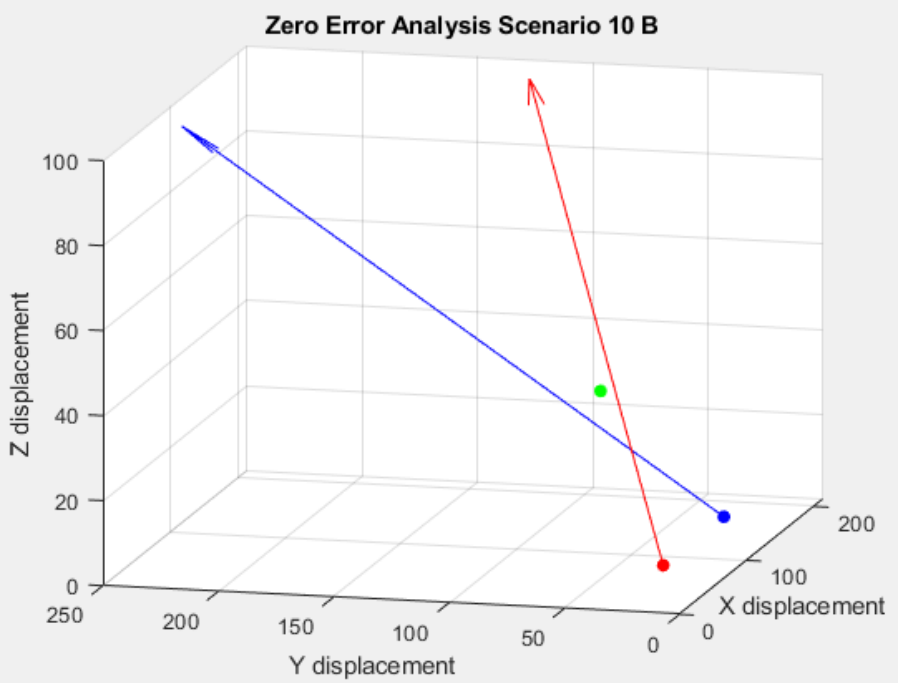

Figure B-22: Scenario 10 MATLAB Zero Error Model View B

The resulting intersection, determined through calculations from the MATLAB zero error model, occurs at (98.3112, 62.9366, 38.3687).

Scenario 11:

This scenario has two vectors not truly intersecting. Therefore an analysis graph can be seen in Figure B-23 below, with distance from each vector being on the y-axis and the index, or position, along each vector being on the x axis, which allows for the position along each line to be found.

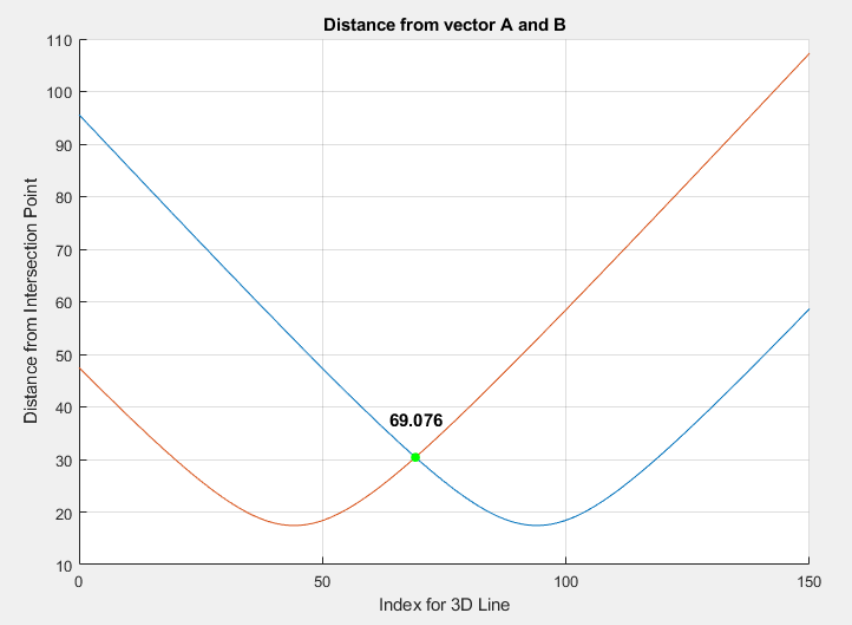

Figure B-23: Scenario 11 Distance to Least Squares Intersection

The position for Vector A at this intersection is (66.2139, 42.4551, 33.6254) and the position for Vector B at this intersection is (100.0000, 79.0760, 10.0000). Next, the scenario was analyzed using MATLAB. The resulting plot can be seen in Figure B-24 and B-25 below.

Figure B-24: Scenario 11 MATLAB Zero Error Model View A

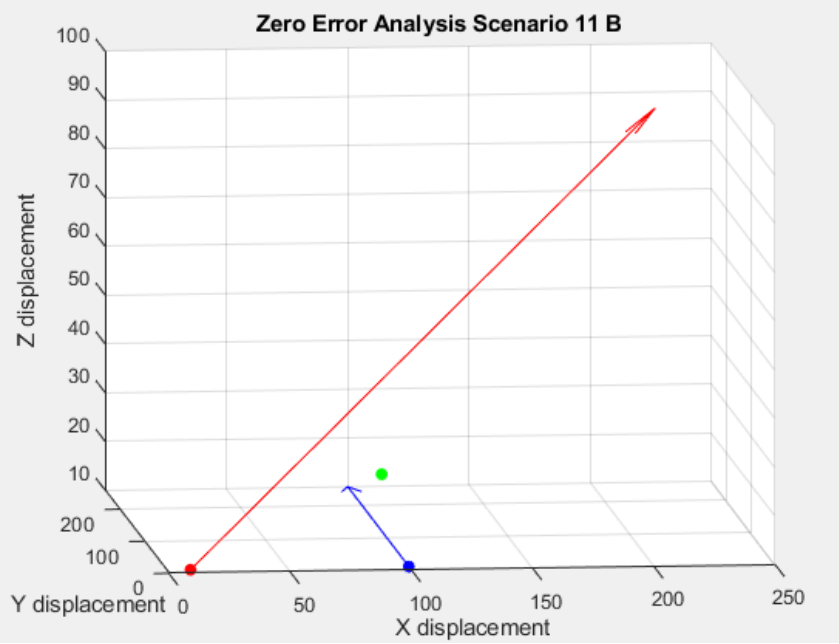

Figure B-25: Scenario 11 MATLAB Zero Error Model View B

The resulting intersection, determined through calculations from the MATLAB zero error model, occurs at (93.2447, 54.1612, 26.0734).

Scenario 12:

This scenario has two vectors not truly intersecting. Therefore an analysis graph can be seen in Figure B-26 below, with distance from each vector being on the y-axis and the index, or position, along each vector being on the x-axis, which allows for the position along each line to be found.

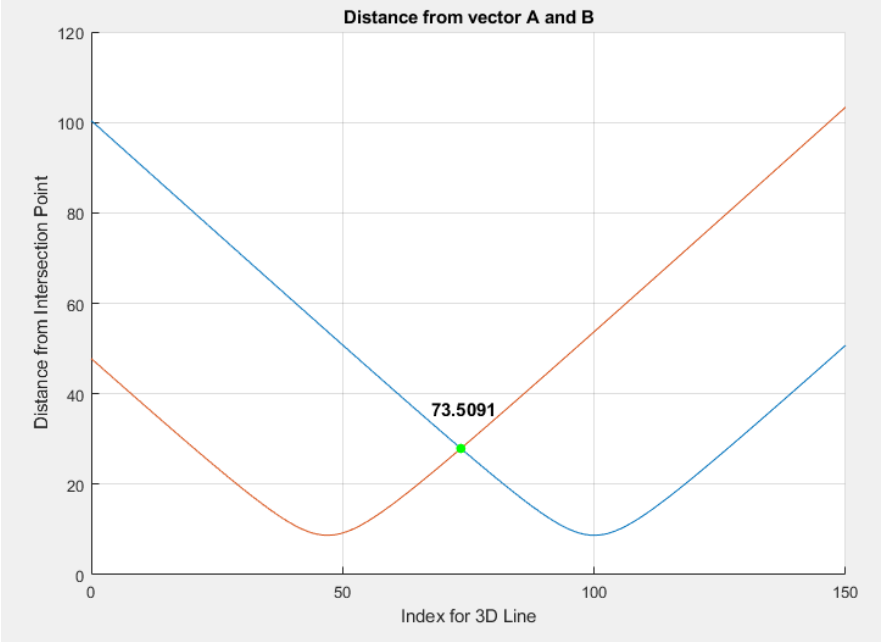

Figure B-26: Scenario 12 Distance to Least Squares Intersection

The position for Vector A at this intersection is (73.6608, 46.7546, 10.0000) and the position for Vector B at this intersection is (100.0000, 79.0760, 35.1416). Next, the scenario was analyzed using MATLAB. The resulting plot can be seen in Figure B-27 and B-28 below.

Figure B-27: Scenario 12 MATLAB Zero Error Model View A

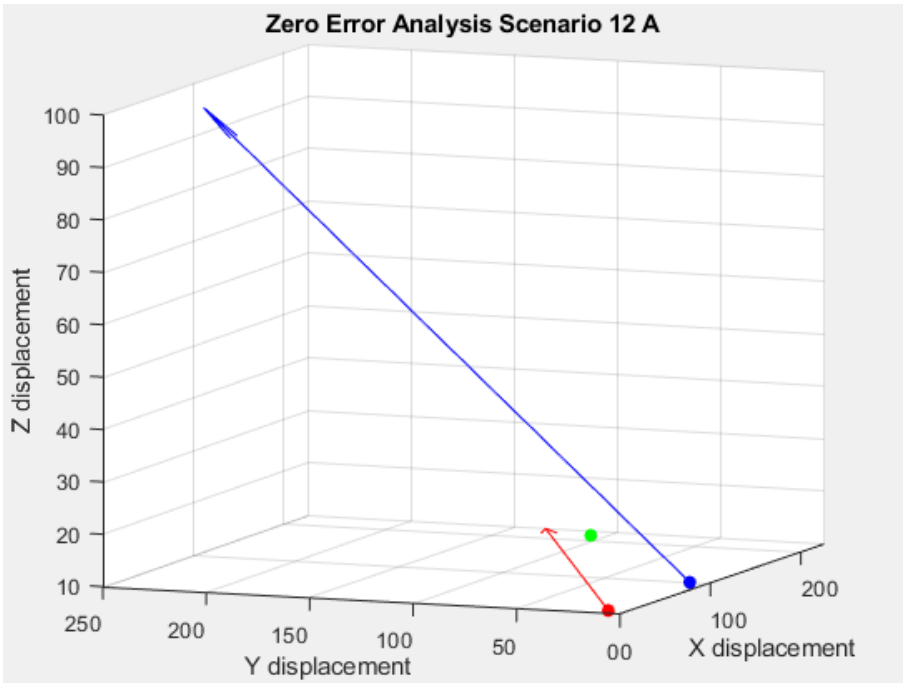

Figure B-28: Scenario 12 MATLAB Zero Error Model View B

The resulting intersection, determined through calculations from the MATLAB zero error model, occurs at (98.3112, 57.0863, 18.0367).

Scenario 15:

This scenario has two vectors not truly intersecting. Therefore an analysis graph can be seen in Figure B-29 below, with distance from each vector being on the y-axis and the index, or position, along each vector being on the x-axis, which allows for the position along each line to be found.

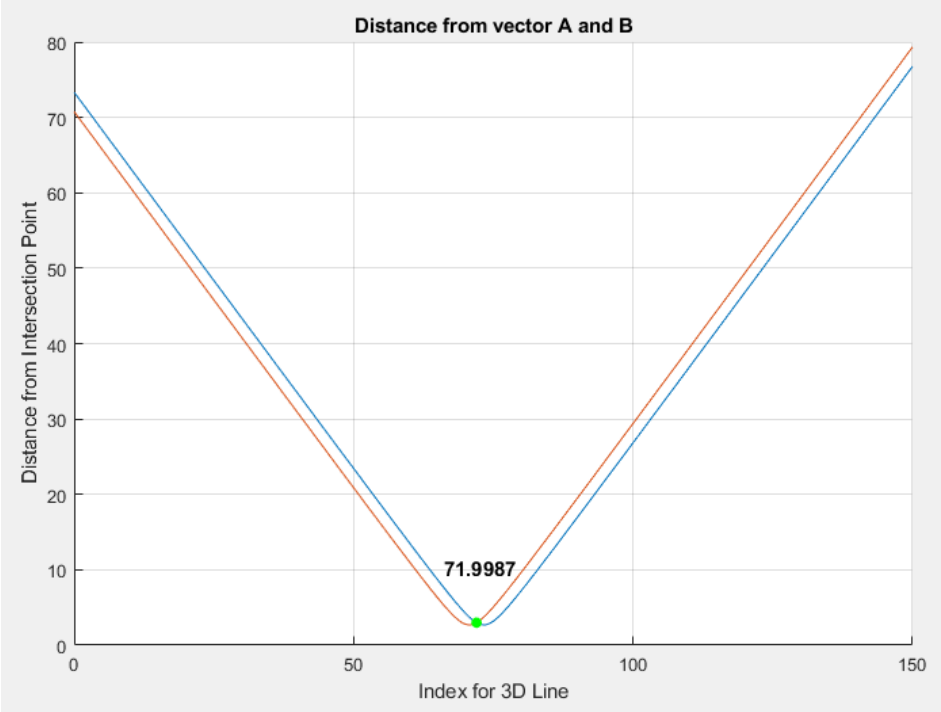

Figure B-29: Scenario 15 Distance to Least Squares Intersection

The position for Vector A at this intersection is (47.8405, 47.8405, 24.6250) and the position for Vector B at this intersection is (49.0892, 50.9108, 20.0000). Next, the scenario was analyzed using MATLAB. The resulting plot can be seen in Figure B-30 and B-31 below.

Figure B-30: Scenario 15 MATLAB Zero Error Model View A

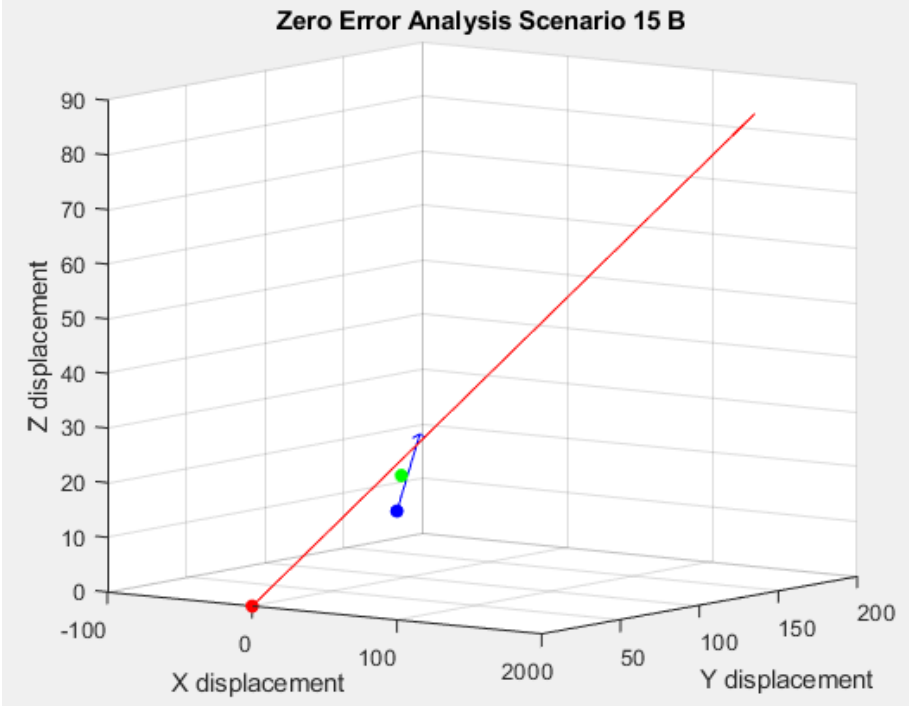

Figure B-31: Scenario 15 MATLAB Zero Error Model View B

The resulting intersection, determined through calculations from the MATLAB zero error model, occurs at (49.3482, 49.3482, 22.5328).

Scenario 16:

This scenario has two vectors not truly intersecting. Therefore an analysis graph can be seen in Figure B-32 below, with distance from each vector being on the y-axis and the index, or position, along each vector being on the x-axis, which allows for the position along each line to be found.

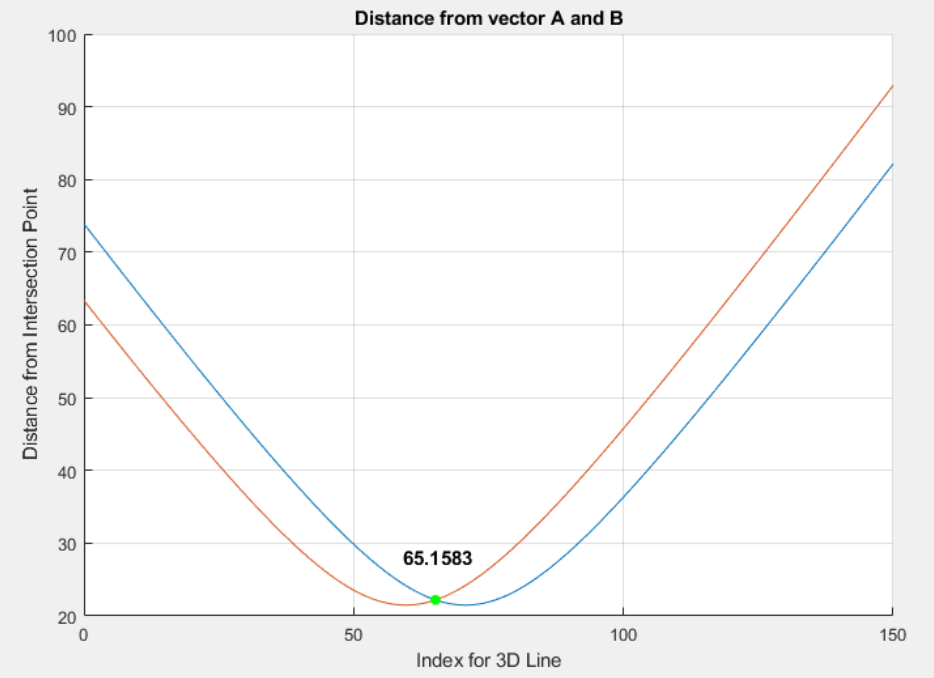

Figure B-32: Scenario 16 Distance to Least Squares Intersection

The position for Vector A at this intersection is (46.0739, 46.0739, 0) and the position for Vector B at this intersection is (56.7047, 43.2953, 42.2854). Next, the scenario was analyzed using MATLAB. The resulting plot can be seen in Figure B-33 and B-34 below.

Figure B-33: Scenario 16 MATLAB Zero Error Model View A

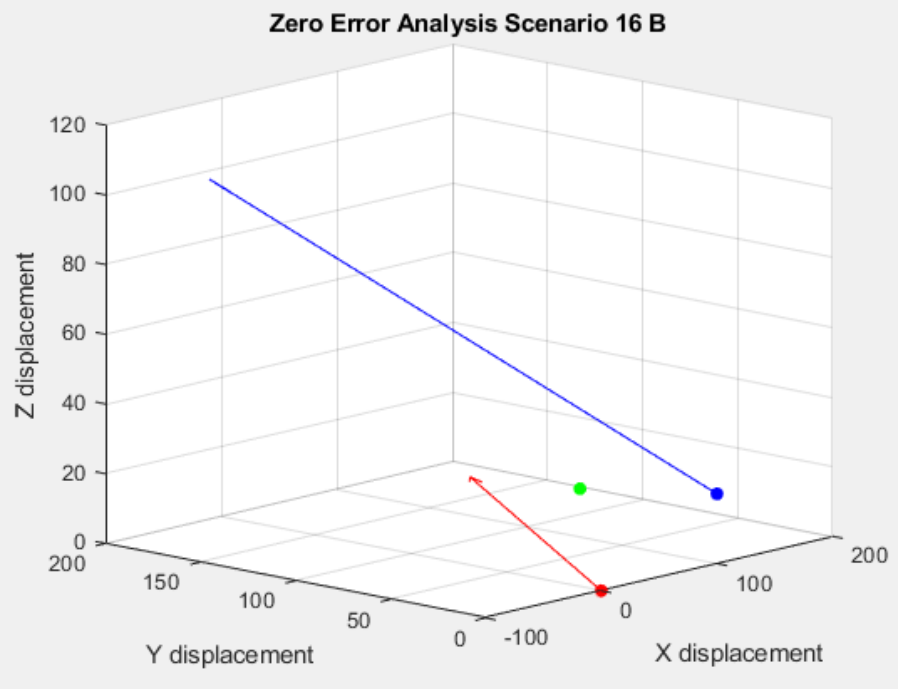

Figure B-34: Scenario 16 MATLAB Zero Error Model View B

The resulting intersection, determined through calculations from the MATLAB zero error model, occurs at (55.1970, 44.8030, 20.1932).

Appendix C – 2-D Distance and Orientation Combinations

(θ_o, *d*)		distance *d* (meters)					
		25	**50**	**…**	**…**	**325**	**350**
Angle between Observers θ_o (degrees)	**20**	(20, 25)	(20, 50)	(20, …)	(20, …)	(20, 325)	(20, 350)
	25	(25, 25)	(25, 50)	(25, …)	(25, …)	(25, 325)	(25, 350)
	…	(…, 25)	(…, 50)	(…, …)	(…, …)	(…, 325)	(…, 350)
	…	(…, 25)	(…, 50)	(…, …)	(…, …)	(…, 325)	(…, 350)
	155	(155, 25)	(155, 50)	(155, …)	(155, …)	(155, 325)	(155, 350)
	160	(160, 25)	(160, 50)	(160, …)	(160, …)	(160, 325)	(160, 350)

Figure C-1: 2-D Distance and Angle Between Observers Simulation Combinations

*Refer to Figure 5-7 in Section 5.1.3 for classification of distance and Angle Between Observers variables.