UAV Localization Performance

A Major Qualifying Project submitted to the Faculty Of Worcester Polytechnic Institute in partial fulfilment of the requirements for the Degree of Bachelor of Science in Electrical and Computer Engineering by:

Brian Mahan

with

Ian Gelman

Abdul Hassan

John Loftus

Submitted to:

Kaveh Pahlavan, Center for Wireless Information Network Studies

Worcester Polytechnic Institute

Acknowledgement

This Project group consists of four team members, Abdul Hassan, Ian Gelman, John Loftus being the other three team members. This projects timeline is three terms long but I am finishing a term early, because of this I am submitted an additional paper to satisfy the additional work needed for finishing a term early. We would like to thank our MQP Professor, Professor Pahlavan and the ECE department for their insight and assistance in the project.

Abstract

This project is motivated to improve the security of the public against Unmanned Aircraft Vehicles (UAV) by creating a system that is able to identify the location of an unwanted drone. Drones have to emit a wireless signal to the drone's controller, which we will be utilizing to intercept and calculate its position. The position is calculated using the localization algorithm Recursive Least Square Method (RLS). This real-time localization system will be able to constantly update the user on the location of the drone, increasing security and safety against unmanned drones. This individual report will focus on the performance and efficiency of this system through the comparison with Cramer Rao Lower Bound.

Table of Contents

Table of Figures

Introduction

In today's world, technology is constantly advancing and has a large influence on everybody. Some of these advancements include new methods of military combat, such as Unmanned Aerial Vehicles (UAV) in military missions. While these technologies are very beneficial, they also can be cause for many security concerns. Drones, for example, can be easily manipulated by others to harm industries or people. Attackers are able to remotely fly drones, and there have been cases of explosives and guns being attached to drones. With the threat of drones rising, our group focused on creating a drone localization system that would help combat these potential threats. Our motivation came from building off of the work coming from the previous MQP localization project "RSS Based 3D Localization and Performance Evaluation" by Zilu Tian [5]. The work done in the project focused on the implementing different RSS localization algorithms based off of RSS readings and evaluating the performance of each. These RSS readings were taken from a phone which was attached to the drone and the actual signal from the drone was not being taken. To continue forward with this project, we aimed to develop a RSS localization system utilizing the information on the different localization algorithms from the previous project. This system will be used for the detection of an unwanted drone which may pose a threat to the system's user. The detection of the drone would come from intercepting the actual signal from the drone and using it to find the position of the drone.

1.1 Project Description

This project is focused on using software defined radios to measure the signals coming from the drones camera, this wireless signal is measured and its received signal strength is used to calculate the position of the drone. The drone used in the project is the DJI Phantom 3 Pro, which camera feed transmits between 2.4 to 2.48GHz frequency range depending on the channel used for camera streaming. For radios, the Ettus USRP2s are utilized, which are designed to measure frequencies in the 2.4 GHz as well as the 5.9 GHz range. These radios, all connected physically by a switch, connect to a host computer, which runs a software called GNU Radio for the digital signal processing. The path loss model of our drones signal, which is the characteristic of a wireless signal used to predict signal strength loss and can be used to calculate the distance from the drone to the radio, is utilized in our project to find the unwanted drone location. From the drones wireless signal, we are able to estimate the distance from our radios to the drone with the use of our path loss model. From there the distances are used in a localization algorithm which takes in the known location of different radios and the distance from each radio to the drone to estimate the drone's position. This system is calculated in real time to provide the user with up to date locations of the drone. The overall effectiveness of the system is analyzed by comparing the accuracy of the system with its Cramer Rao Lower Bound, which is what this report is focusing on.

1.1.1 Group Contributions

As I am finishing my portion of the MQP a term earlier than the rest of my team, I have created a separate report to discuss the additional work that I was responsible for. Throughout this project, my main focus was on the analysis of our localization systems' performance. I was responsible for the derivation of the Cramer Rao Lower Bound algorithms and the resulting data that it produced, as well as comparing it with the experimental data from our project. Along with this I was responsible for the construction of the 2D and 3D simulation functions that assisted in evaluating the performance of our system. The other members of the group, Ian, John and Abdul, were all responsible in the configuration and setup of the system, along with the testing procedure. For this, they were responsible for developing code in python to make the real time localization of the drones possible, as well as the configuration of GNU radio to process the drone signal with minimal noise and error. They were also responsible for processing the data recorded from different tests. This report will go over the background and motivation for exploring Cramer Rao Lower Bound, as well as going over the actual localization system created. For the System, the efficiency and accuracy in each dimension is evaluated and discussed.

1.2 Report Outline

The reports outline is as follows, Chapter 2 will provide the background of the radios and localization technique used as well as the background on Cramer Rao Lower Bound (CRLB). In Chapter 3, the setup and data collection of the project is illustrated along with the expansion of our localization algorithm and Cramer Rao Lower Bound, and in Chapter 4 the experimental results of the system are compared with the Cramer Rao Lower bound and are evaluated. Chapter 5 includes the conclusion as well as future work for the final term of the MQP group. In the Appendix, the Cramer Rao Lower Bound and simulation code used in the project can be found, as well as important figures and information.

Background

In this chapter, general information on our localization system will be provided. For our system, we needed to utilize Software Defined Radios (SDR) to obtain the signal from the given drone. With this data, Path Loss Models and RSS Localization Algorithms were able to be derived as well as Cramer Rao Lower Bound. This chapter will provide the background necessary for the understanding of this project. These topics will be discussed and important equations regarding the topics will also be covered in this chapter. The information provided for the software defined radios section was provided by the Ettus Knowledge Base [2]. For our background on the RSS localization systems and algorithms we utilized the previous MQP [5] along with Professor Pahlavans Textbook ‘Principles of Wireless Access and Localization” [4].

2.1 Software Defined Radios

Software Defined Radios (SDR) are radio communication systems where instead of utilizing hardware components such as amplifiers, modulators, etc. the software defined radio uses software to implements these means from a computer or other embedded system. This gives the SDR a lot of flexibility, as it is easily able to change and modify the configuration of the radio to suit the user and how they would like the signal to be received and processed.

In our project, we chose to utilize the Ettus USRP2s. We chose to use these radios because we saw from [2] each radio contains a 100 mega-sample per second ADC, and a 400 mega-sample per second DAC. These radios also have ethernet connectivity with the host computer, making it easy to communicate with the radios. The USRPs we have chosen have been configured to measure signals in the 2.4 GHz range, with MIMO support to increase the speed of the radios. With these radios we are given the flexibility to configure them to best fit our desired localization system, and easily communicate with the radios to get the received data in a timely and efficient manner. The detailed specifications of our Software Defined Radio is provided below.

USRP
Dual 100 MS/s, 14-bit ADC
Dual 400 MS/s, 16-bit DAC
Gigabit Ethernet Connectivity
MIMO Capabilities
Spartan 3 XC3S2000 FPGA
1 MB SRAM

Figure 1: USRP2 Specifications

2.2 Path Loss Model

Path Loss Model is a linear regression model that illustrates the relation of the received signal strength (RSS) with the distance between the transmitter and receiver based off of the equation defined in [4] :

$$P_r = P_0 - 10\alpha logd + X$$

The Power Received (P_r) is equivalent to Power Loss in the First Meter (P_0) deducted by the Distance Power Gradient (α) multiplied by the 10 times the logarithmic distance. P_r is expressed in dBm and the distance is in meters. Alpha (α) is the derived slope of a linear regression model. Alpha (α) represents the decay of the RF signal strength over distance while X represents the standard deviation of shadow fading.

Shadow Fading is the main cause of fluctuation of received signal strength at certain locations. These variations can be due to the signal being affected by walls and other objects. In this equation the variable X is representing a random value to illustrate the effects of Shadow Fading. This variable, like Alpha, can be derived from the linear regression model of the signal.

2.3 RLS Algorithm for RSS-Based Localization

From this Path Loss Model, distances are able to be derived based off of the received signal strength. When given multiple reference points to receive the signal from the drone, multiple distances at different locations can be derived and the location of the drone can be calculated using localization algorithms.

When looking at [5], three different localization algorithms were explored: Weighted Centroid, Maximum Likelihood Estimation (MLE), and Recursive Least Squares (RLS). The performance and efficiency of these popular localization algorithms were evaluated, through their accuracy based off of Cramer Rao Lower Bound, as well as the time complexity of each algorithm. Through comparison with the CRLB, and total computation times for each algorithm, a table summarizing Tian's findings is shown below:

Algorithm	Accuracy	Time Complexity
Maximum Likelihood Estimation	Best	High
Recursive Least Squares	Better	Medium
Weighted Centroid	Good	Low

Figure 2: Feature Matrix for RSS-Based Localization Algorithms

For our project, we decided to utilize the Recursive Least Square, as its accuracy was seen to be efficient while also not being overly complex.

In the 2D Recursive Least Square Algorithm, the function reflecting ranging error from a device and a radio as defined in [4] is:

$$f_i(x,y) = (x_i - x)^2 + (y_i - y)^2 - d_i^2$$

Where (x,y) is the location of the device to be localized , (x_i, y_i) is the location of the i-th radio and d_i is the calculated distance from the radio and the device. When combining the functions into the quadratic vector function F , which is defined as:

$$F = [f_1(x,y), f_2(x,y), f_3(x,y), ... , f_N(x,y)]^T$$

We are able to convert into a Jacobian Matrix J:

$$J = \begin{bmatrix} \frac{\delta f_1(x,y)}{\delta_x} & \frac{\delta f_1(x,y)}{\delta_y} \\ \frac{\delta f_2(x,y)}{\delta_x} & \frac{\delta f_2(x,y)}{\delta_y} \\ \vdots & \vdots \\ \frac{\delta f_i(x,y)}{\delta_x} & \frac{\delta f_i(x,y)}{\delta_y} \end{bmatrix}$$

With this Jacobian Matrix, we are able to then estimate the location. If we start with a location:

$$l(n) = [x(n), y(n)]$$

We can then update this location through the equation:

$$l(n + 1) = l(n) + E_n$$

Where:

$$E_n = -(J^T J)^{-1} J^T F$$

2.4 CRLB of RSS-Base Positioning for Performance Analysis

In Localization Systems, the performance of the ranging and localization can be compared with the Cramer Rao Lower Bound. This bound is compared with the standard deviation of a localization system, which comes from the spread of error against the estimated distance or location. A lower variance indicates a lower chance of high error from the location estimate. From [4], CRLB gives the smallest variance of a probability distribution function $f(O|\,\alpha)$ such that:

$$Var[\hat{}\ \alpha\,(O) - \alpha] \geq CRLB$$

The CRLB is given by the calculating the inverse of the Fisher Information Matrix

$$F = E\left[\frac{\delta \ln f\,(O|\alpha)}{\delta\alpha}\right]^2 = -E\left[\frac{\delta^2 \ln f\,(O|\alpha)}{\delta\alpha^2}\right]$$

Making the overall CRLB equation

$$CRLB = Var[\widehat{\alpha}\,(O) - \alpha] \geq F^{-1}$$

When looking at observations that are corrupted by zero mean Gaussian noise, the observations O can be seen as

$$O = \alpha + \eta$$

Where η is the Gaussian noise with variance σ^2. The conditional probability density function for O is given by the equation:

$$f(O|\alpha) = \frac{1}{\sqrt{2\pi\sigma}} exp(-\frac{(O-\alpha)^2}{2\sigma^2})$$

When put through the fisher matrix , the function simplifies to

$$\frac{1}{\sigma^2}$$

Therefore:

$$CRLB = F^{-1} = \sigma^2$$

2.4.1 CRLB for Ranging

In our RSS localization system, as mentioned, the signal strength of the drone is the observed power that is used to estimate distance. In this case, our path loss model is our observation function.

$$P_r = P_0 - 10\alpha logd + X$$

From this function, we are able to then convert it into our probability distribution function from [4]

$$\frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(P_r - P_0 + 10\alpha logd)^2}{2\sigma^2}}$$

Put it through the fisher matrix

$$\frac{(10)^2\alpha^2}{(ln\ 10)^2\alpha^2 d^2}$$

And take the inverse to obtain our CRLB

$$CRLB = \frac{(ln\ 10)^2}{100}\frac{\sigma^2}{\alpha^2}d^2$$

To obtain the standard deviation of error for comparison σ_p , we need to take the square root of this equation since the CRLB is the variance. This will give us the needed CRLB for ranging comparison

$$\sigma_p \geq \frac{(ln\ 10)}{10}\frac{\sigma}{\alpha}d$$

2.4.2 CRLB for Positioning in 2D

In positioning across a 2D plane, the path loss model is similar to in ranging where

$$P_r = P_0 - 10\alpha log r_i + X, \quad i = 1, 2, ... N$$

And the distance d from the ranging is replaced with the distance in relation to multiple axes r_i

$$r_i = \sqrt{(x - x_i)^2 + (y - y_i)^2}$$

Where (x, y) is the location of the device and (x_i, y_i) is the location of the reference points for N given reference points. From shadow fading, positioning algorithms experience fluctuations in received power, defined by [4] where:

$$dP_i(x, y) = -\frac{10\alpha_i}{ln\ 10}\left(\frac{x - x_i}{r_i^2}dx + \frac{y - y_i}{r_i^2}dy\right)$$

Which leads to variations in ranging estimates dr . when looking at these parameters in vector form, their relationship can be seen as

$$dP = Hdr$$

And

$$dr = (H^T H)^{-1} H^T dP$$

Where

$$dP = \begin{bmatrix} dP_1 \\ dP_2 \\ \vdots \\ dP_N \end{bmatrix};\ dr = \begin{bmatrix} d_x \\ d_y \end{bmatrix};\ H = -\frac{10}{\ln 10}\,[\sigma_1 \sigma_2 \ldots\ldots\ldots \sigma_N] \begin{bmatrix} \frac{x - x_1}{r_1^2} & \frac{y - y_1}{r_1^2} \\ \frac{x - x_2}{r_2^2} & \frac{y - y_2}{r_2^2} \\ \vdots & \vdots \\ \frac{x - x_N}{r_N^2} & \frac{y - y_N}{r_N^2} \end{bmatrix}$$

From here, we are able to find the covariance of the location estimate dr from the equation

$$cov(dr) = \sigma^2 (H^T H)^{-1} = \begin{bmatrix} \sigma_x^2 & \sigma_{xy}^2 \\ \sigma_{xy}^2 & \sigma_y^2 \end{bmatrix}$$

And then take the standard deviation of the location error σ_r , off of the resulting matrix

$$\sigma_r = \sqrt{\sigma_x^2 + \sigma_y^2}$$

This standard deviation, similar to that of ranging, gives us the minimum accuracy our localization system can have in 2D at a certain (x,y) point.

Methodology

In this chapter, the overall design of our localization system will be discussed. We will then discuss the expansion of the RLS and CRLB algorithms to 3D, as well as the setup of the data acquisition system.

3.1 System Design

For our localization system, we utilized four Ettus USRP2 software defined radios which are connected to a network switch. The network also contains a host user who will receive transmission, and any other users to which data can be sent. The configuration of the radios as well as accessibility to data from multiple users allows for a simpler implementation of localization algorithms.

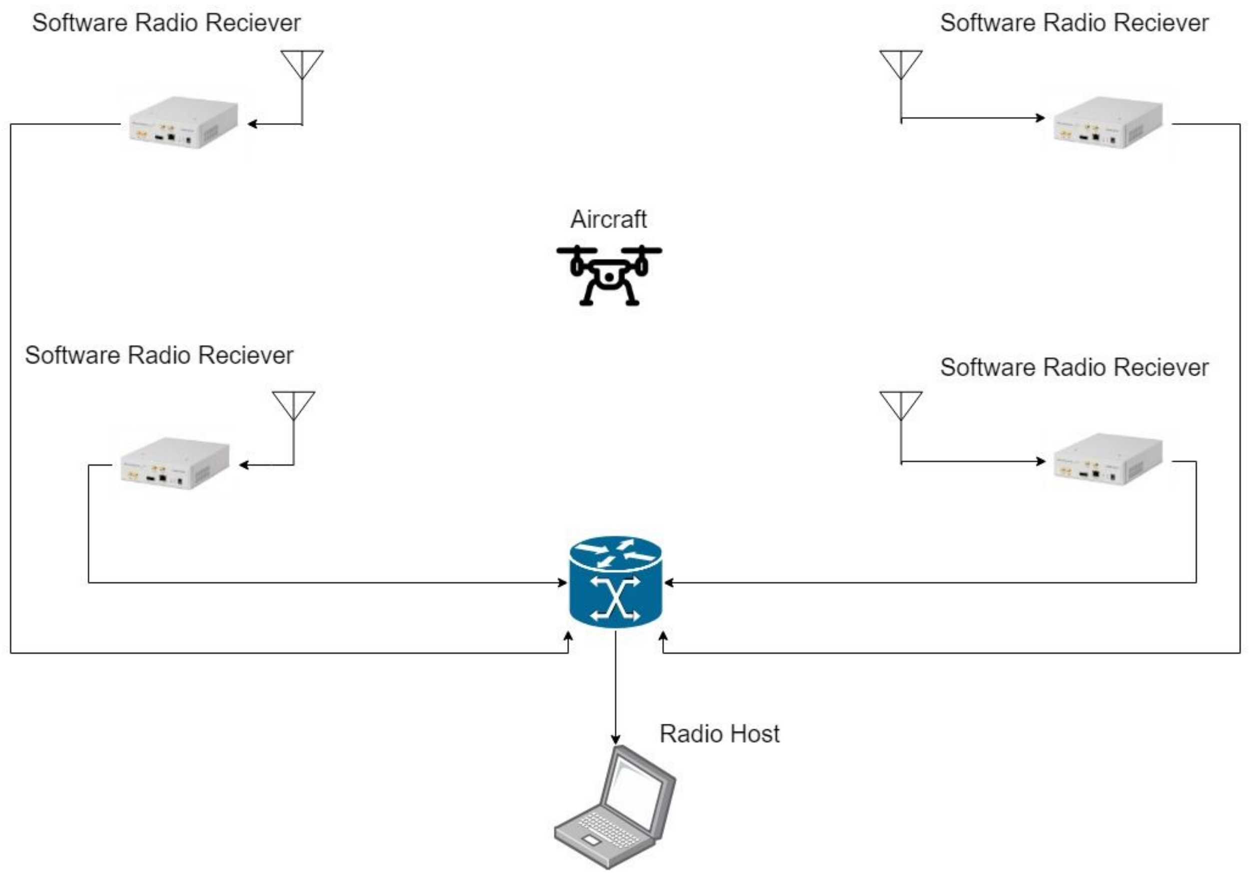


Figure 3: Design Functional Block Diagram of Localization System

As shown in Figure 3 above, the four radios are placed at various spots on the ground to create a large area in which the drone can fly in between the radios. These radios are then connected to the Radio host.

Our system is designed to work for any UAV, but in this project we used a DJI Phantom 3 Professional. This drone, which operates within 2.400 GHz to 2.483 [1], has 8 ISM channels that the operator is able to use for communication between the drones camera and the remote controller. The Mobile device of the operator, which is connected via USB On The Go(USB-OTG) receives the camera feed for display. The communication protocols for the drone, camera and controller can all be seen in the figure below.

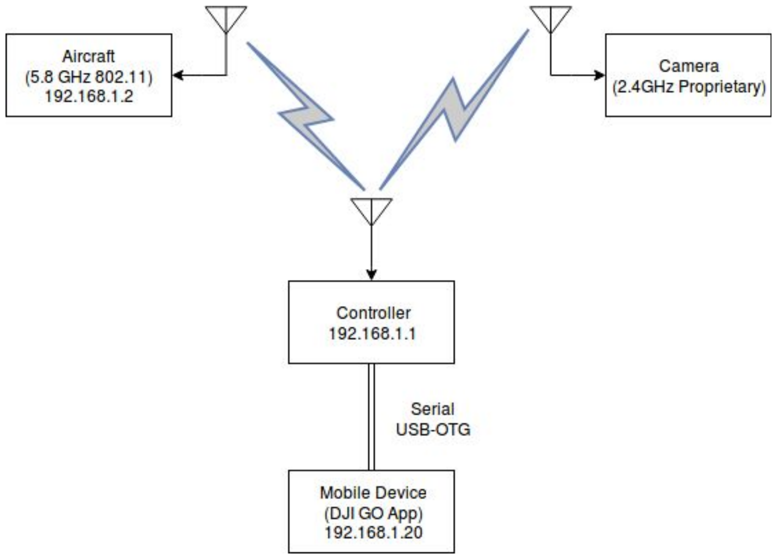


Figure 4: DJI Phantom 3 Professional Communication Protocols

With using the DJI Phantom 3 Pro, we needed to define the 8 different ISM channel characteristics so we would be able to correctly measure the power that the drone is emitting at each signal. To do this we used the software defined radio to create an FFT plot to view the characteristics of each ISM band which can be seen in the figure below.

Channel Name	Center Frequency	Lower Band Limit	Upper Band Limit	Total Bandwidth (MHz)
13	2.40664	2.40179	2.41148	9.69
14	2.41647	2.41172	2.42122	9.5
15	2.42649	2.42172	2.43125	9.53
16	2.43651	2.43184	2.44117	9.33
17	2.44656	2.44185	2.45127	9.42
18	2.45653	2.45181	2.46125	9.44
19	2.46656	2.46184	2.47128	9.44
20	2.47652	2.47176	2.48127	9.51

Figure 5: Measured Frequency Allocation Table

When looking at the characteristics of the camera signals, it can be seen in the table above that the average Bandwidth of the Channels are close to 10 MHz, with the center frequencies ranging from 2.40664 to 2.47652. By knowing the characteristics of the possible bands of the drone's camera signal, we were now able to set up the radios to the correct frequency to measure the received power coming off of the drone.

3.2 3D Expansion for RSS-Based Positioning

One challenge that we faced in our project was taking the 2D algorithms and converting them to 3D which was needed for our localization system analysis. To convert these into 3D, another dimension, accounting for height, had to be put into consideration for these algorithms.

3.2.1 3D Expansion of RLS for RSS-Based Positioning

Similar to the 2D Recursive Least Square Algorithm, the 3D RLS algorithm follows the same steps, except an extra dimension z is added. In the 3D RLS algorithm, the function reflecting ranging error from a device and a radio is defined as:

$$f_i(x,y,z) = (x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2 - d_i^2$$

Where (x,y,z) is the location of the device to be localized , (x_i,y_i,z_i) is the location of the i-th radio and d_i is the calculated distance from the radio and the device.

$$d_i = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2}$$

Combining the functions into the quadratic vector function F , produces a similar outcome as in 2D RLS where:

$$F = [f_1(x,y,z), f_2(x,y,z), f_3(x,y,z), ... , f_N(x,y,z)]^T$$

From here, we convert to a Jacobian Matrix J:

$$J = \begin{bmatrix} \frac{\delta f_1(x,y,z)}{\delta_x} & \frac{\delta f_1(x,y,z)}{\delta_y} & \frac{\delta f_1(x,y,z)}{\delta_z} \\ \frac{\delta f_2(x,y,z)}{\delta_x} & \frac{\delta f_2(x,y,z)}{\delta_y} & \frac{\delta f_2(x,y,z)}{\delta_z} \\ \vdots & \vdots & \vdots \\ \frac{\delta f_i(x,y,z)}{\delta_x} & \frac{\delta f_i(x,y,z)}{\delta_y} & \frac{\delta f_i(x,y,z)}{\delta_z} \end{bmatrix}$$

Similar to 2D RLS, we are able to then estimate the location where if we start with a location:

$$l(n) = [x(n), y(n), z(n)]$$

We can then update a location:

$$l(n+1) = l(n) + E_n$$

3.2.2 3D Expansion of CRLB for RSS-Based Positioning

In 3D CRLB the definitions are similar to that of 2D, but like with the expansion of RLS there now is the extra height dimension z . Now considering an X,Y,Z axis the path loss model is:

$$P_r = P_0 - 10\alpha log r_i + X, \quad i = 1, 2, \ldots N$$

Where

$$r_i = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2}$$

Variations in received power are now defined by:

$$dP_i(x,y,z) = -\frac{10\alpha_i}{ln\ 10}\left(\frac{x-x_i}{r_i^2}dx + \frac{y-y_i}{r_i^2}dy + \frac{z-z_i}{r_i^2}dz\right)$$

And dP, dr and H are all defined as:

$$dP = \begin{bmatrix} dP_1 \\ dP_2 \\ \vdots \\ dP_N \end{bmatrix};\ dr = \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix};\ H = -\frac{10}{\ln 10}[\sigma_1\sigma_2 \dots\dots\dots \sigma_N]\begin{bmatrix} \frac{x-x_1}{r_1^2} & \frac{y-y_1}{r_1^2} & \frac{z-z_1}{r_1^2} \\ \frac{x-x_2}{r_2^2} & \frac{y-y_2}{r_2^2} & \frac{z-z_2}{r_2^2} \\ \vdots & \vdots & \vdots \\ \frac{x-x_N}{r_N^2} & \frac{y-y_N}{r_N^2} & \frac{z-z_N}{r_N^2} \end{bmatrix}$$

This makes the covariance function:

$$cov(dr) = \sigma^2(H^T H)^{-1} = \begin{bmatrix} \sigma_x^2 & \sigma_{xy}^2 & \sigma_{xz}^2 \\ \sigma_{xy}^2 & \sigma_y^2 & \sigma_{yz}^2 \\ \sigma_{xz}^2 & \sigma_{yz}^2 & \sigma_z^2 \end{bmatrix}$$

From here, similar to 2D we are able to take the standard deviation of the location error σ_r, off of the resulting matrix

$$\sigma_r = \sqrt{\sigma_x{}^2 + \sigma_y{}^2 + \sigma_z{}^2}$$

3.4 Data Acquisition System for Empirical Analysis

In our project, data collection was needed for the construction of our localization system. We needed to build a path loss model based on data collected from the drone to the radios. We also needed to collect data from 1D, 2D and 3D which we would be able to compare with CRLB to measure the performance. For this we obtained data from simulations as well as field testings.

3.4.1 Data Acquisition Scenario for Path Loss Modeling

To create our path loss model, we first had to retrieve the necessary parameters, α, P_0 and X, from the path loss equation. To find the first meter path loss P_0, we measured the signal strength of the drone inside of an anechoic chamber exactly one meter away. This resulted in an accurate first meter path loss due to the lack of network interference inside of the chamber.

To find the remaining alpha and shadow fading values, a linear regression model was created. The data for the regression model was measured by recording received signal strengths of the drone at different locations on the WPI football field. This environment provided us with a data set that we were able to use for our linear regression model. Using the yard markers given on the football field we recorded data in increments from 5 yards to 85 yards, along with a longshot measurement at near 100 yards. Each measurement span was 10 Seconds and consisted of 10e8 samples, and the distance was converted to meters after the measurements were completed. The figure below shows the area of the field utilized for our data collection, the red diamonds are the spots where we put the drone for field testing, with the blue circle represents the radio recording the drones signal placed on the right 20 yard line.

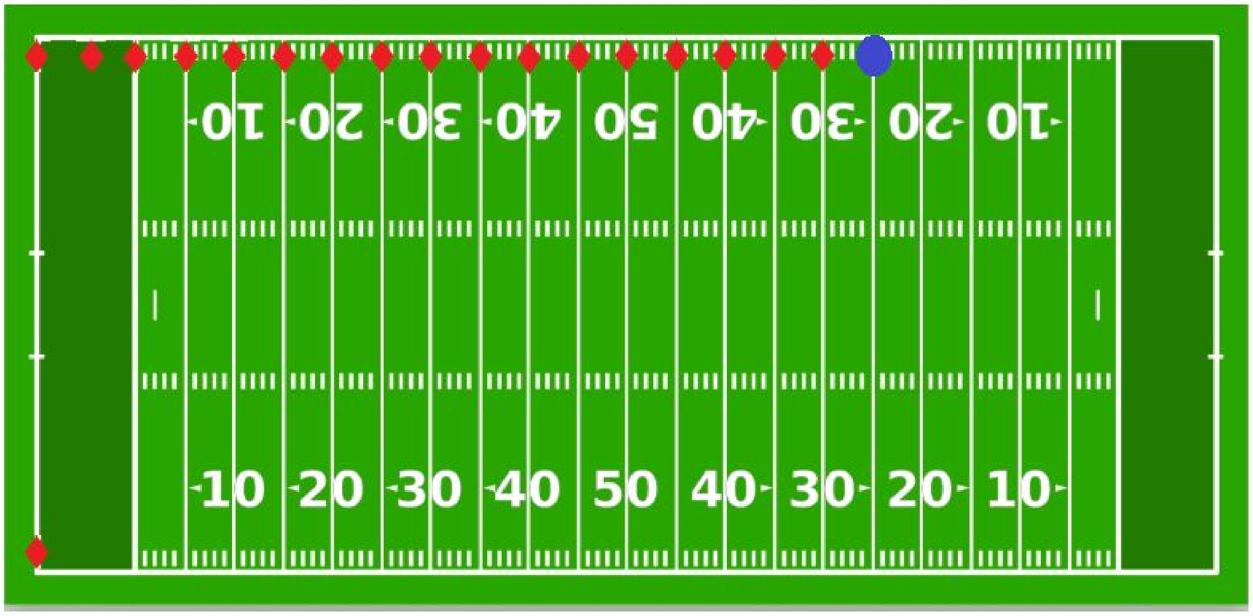


Figure 6: Football Field Measurement Locations for Path Loss Modeling Denoted by ◆

When initial tests from the ground level showed poor results and inconsistent path loss models, we realized the ground was interfering with the signal. To avoid added interference from the ground, the drone and radio were placed at a height of .712 meters during data acquisition, this gave a clearer signal with less path loss. The figure below shows our modified test setup, where we used trash bins to keep the drone and the radio at a constant height above the ground.

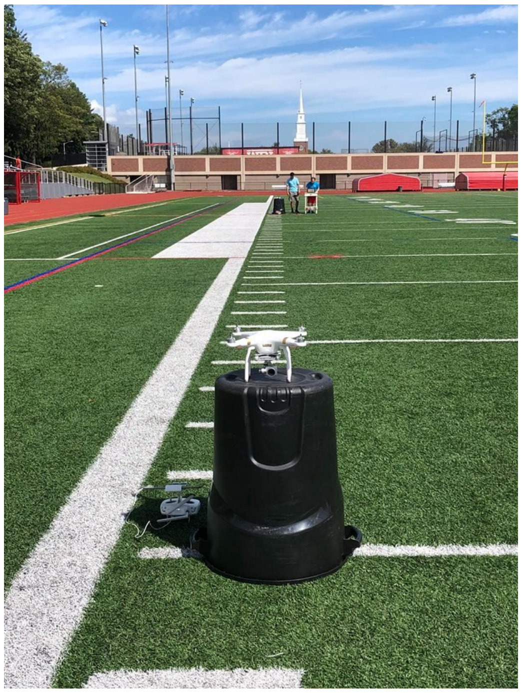


Figure 7: Data Acquisition Test Setup for Path Loss Modeling

3.4.2 Data Acquisition for Analysis of Ranging Error

For our ranging data collection, our first test examined the accuracy of our system in 1D. To do this we set up a similar test environment as the path loss model tests. We set up our radio at one spot and calculated distances derived from the signal of the drone at different locations of the field. For our ranging tests, we tested at distances from 1 meter to 80 meters away from the drone.

With these calculated distances from our path loss model, we compared the actual distance of the radio to the drone with the calculated distances by finding the standard deviation of error between the two. When looking at the standard deviation between the calculated distances d and the actual distance D the calculation can be seen as:

$$\sqrt{\frac{\sum_{i=1}^{n}(d_i - D)^2}{n}}$$

As previously mentioned, the main cause or error in localization is due to shadow fading. To look closer at the effects of shadow fading we set up our test to highlight these effects. For this test, to simulate the conditions of shadow fading we had members of the team hold the drone during data collection and rotate in a circle, so the signal would be periodically blocked by the member holding the drone and create shadow fading.

3.4.3 Data Acquisition for Analysis of 2D Positioning Error

For our 2D positioning tests, we utilized two different methods. Our first method was the construction of a 2D positioning simulator, which will take in desired positions of the drone for testing and return calculated positions based off of the 2D RLS algorithm.

In this simulator the program will take in the desired position (x, y) and calculate the distances from each reference point, from there the distances will be put into the path loss model along with noise due to shadow fading, to derive a theoretical received power:

$$P_r = P_0 - 10 * \alpha * log(d) + (+/-) X(\sigma)$$

From here, we then use the received power with noise to calculate our new distances:

$$\widehat{d} = 10^{\frac{P_r - P_0}{10 * \alpha}}$$

With these distances from our reference points $(\widehat{d_1}, \widehat{d_2}, \widehat{d_3}, \widehat{d_4})$ we are able to run them through our 2D RLS algorithm and obtain our simulated position. From running this simulator multiple times we are able to get variating positions and calculate a simulated standard deviation of error which we are able to compare with our 2D CRLB error.

Our second test was to use real time data, which was the ranging data used on the football field, and run it through our 2D positioning system. In our test system, we had different RSS readings ranging from 5 to 75 meters in increments of five, from these distance measurements we were able to construct a simulated test setting with our data.

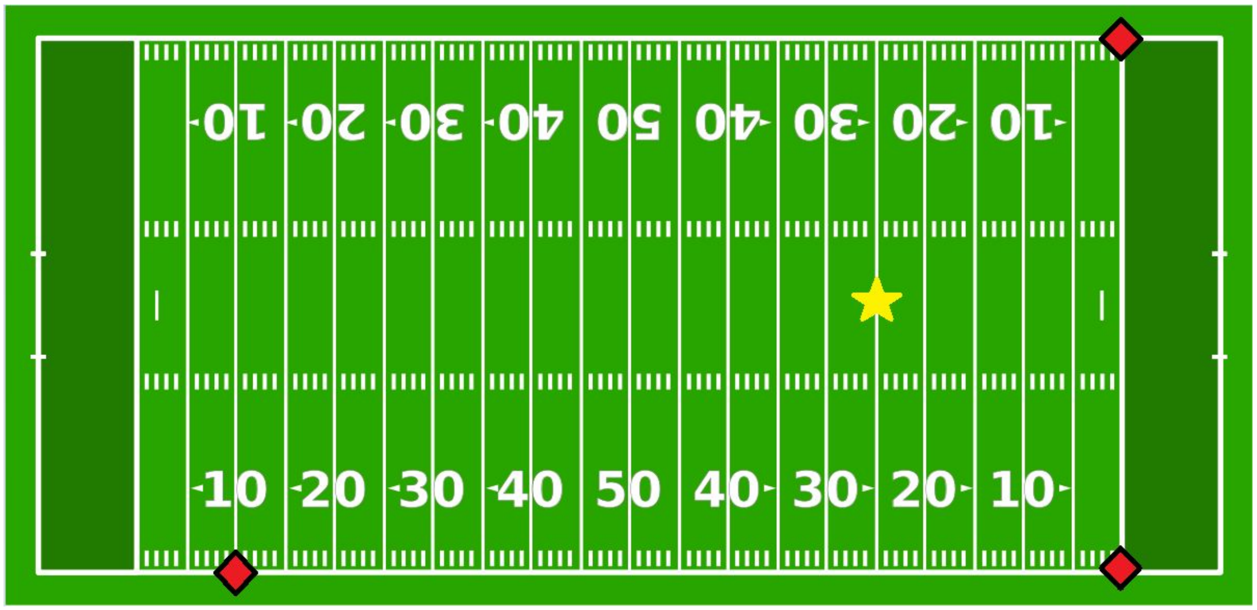


Figure 8: 2D Preliminary Testing Setup Locations of Radios and Drone Position

In our test setup as seen in the figure above we have 3 different radios taking in the recorded data. Radio 1 located at the left side 10 yard line, the radio 2 at the bottom right goalline and radio 3 at the top right goalline. From these radios we fed in our recorded field data: radio 1 took in data measurements from 70 meters, and radios 2 and 3 took in measurements at 35. From these distances it makes our drone location at the middle of the 25 yard line, denoted from the gold star in the figure above. From reading in these data recordings at the known locations we were able to run our system to produce our calculated drone positions, and calculate the standard deviation of error to analyze the 2D performance.

3.4.3 Data Acquisition for Analysis of 3D Positioning Error

For 3D positioning testing, due to the time constraints, I will be finishing my portion of the project before the physical 3D positioning testing will be completed which is scheduled to take place in C term. Because of this, I have created the 3D simulator program, similar to the 2D simulator, so that when the group gets to the 3D real time testing it can be compared with the simulator.

Like the 2D simulator, this 3D simulator will take a given desired position (x, y, z) and calculate the distances from each reference point, which will then be put through a path loss model with noise and generate new distances.

With these distances from our reference points $(\widehat{d}_1, \widehat{d}_2, \widehat{d}_3, \widehat{d}_4)$ we run them through our 3D RLS algorithm and obtain our simulated position. Like in 2D we are able to gather our simulated standard deviation of error and compare them with our CRLB values.

Results and Analysis

In this section, the test procedure and results of ranging and positioning will be discussed. We will show the path loss model generated along with the derived CRLBs for ranging and positioning from the equations provided in the background and methodology. With these CRLBs, we will compare them with the ranging and positioning results to evaluate the performance of our localization system.

4.1 Path Loss Model for Characteristics of RSS

For our path loss model, as mentioned we needed to obtain α, P_0 and X. In the anechoic chamber, we recorded our measured first meter path loss to be -35.476 dBm

To find the remaining parameters, we utilized the initial field data that we took. From the data collected, the RSS values were plotted against distance, which ranged from 1 meter to 91.7 meters. Figure 10 shows the plot of RSS(dBm) vs 10*log10(d)[dB]. With this plot, the linear regression of the data was taken, providing an α and X of 1.809 and 2.4153. With this information we were able to construct our path loss model for our system:

$$P_r = -35.476 - 10(1.809)logd + (+/-)\,2.4153$$

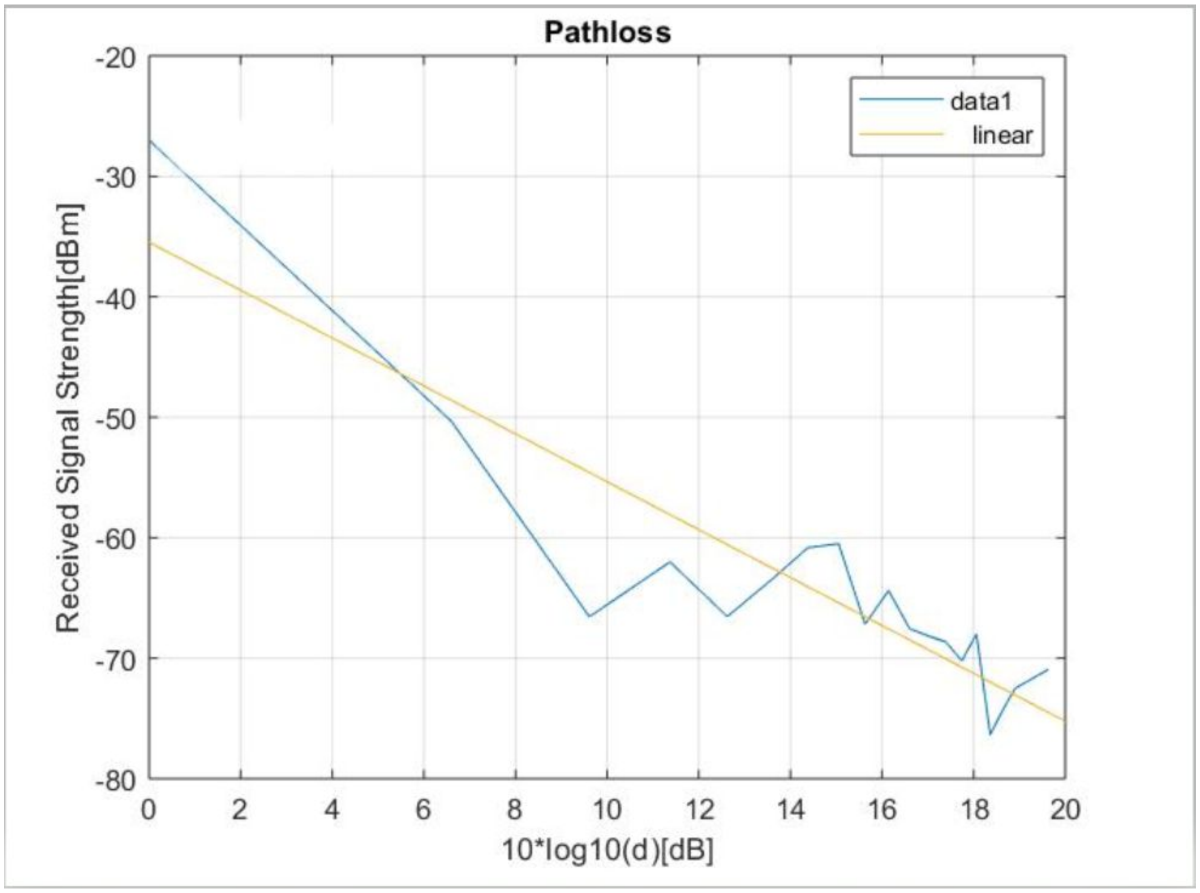


Figure 9: RSS Plot with Linear Regression Model for Path Loss Modeling

4.2 Empirical Analysis of Ranging Error

In ranging, our system is able to be analyzed through the calculation of our theoretical Cramer Rao Lower Bound at different distances, and the comparison of this bound through experimental tests. When comparing the CRLB with the test data, as mentioned earlier, the CRLB value is the theoretical bound where the system cannot have an accuracy below this value. The efficiency is determined by how close the accuracy of the system is to the theoretical lower bound.

4.2.1 Empirical Analysis of Ranging CRLB

With the path loss equation for our system derived, we were able to calculate for the CRLB based on distance. From the ranging CRLB equation we obtained our accuracy bound where at any distance d our RSS localization algorithm should not exceed an accuracy of the given value at that distance. Since we have our path loss model with the necessary parameters, we took our alpha and standard deviation values, plugged them into the CRLB equation and plotted it vs distance to get our ranging CRLB bound.

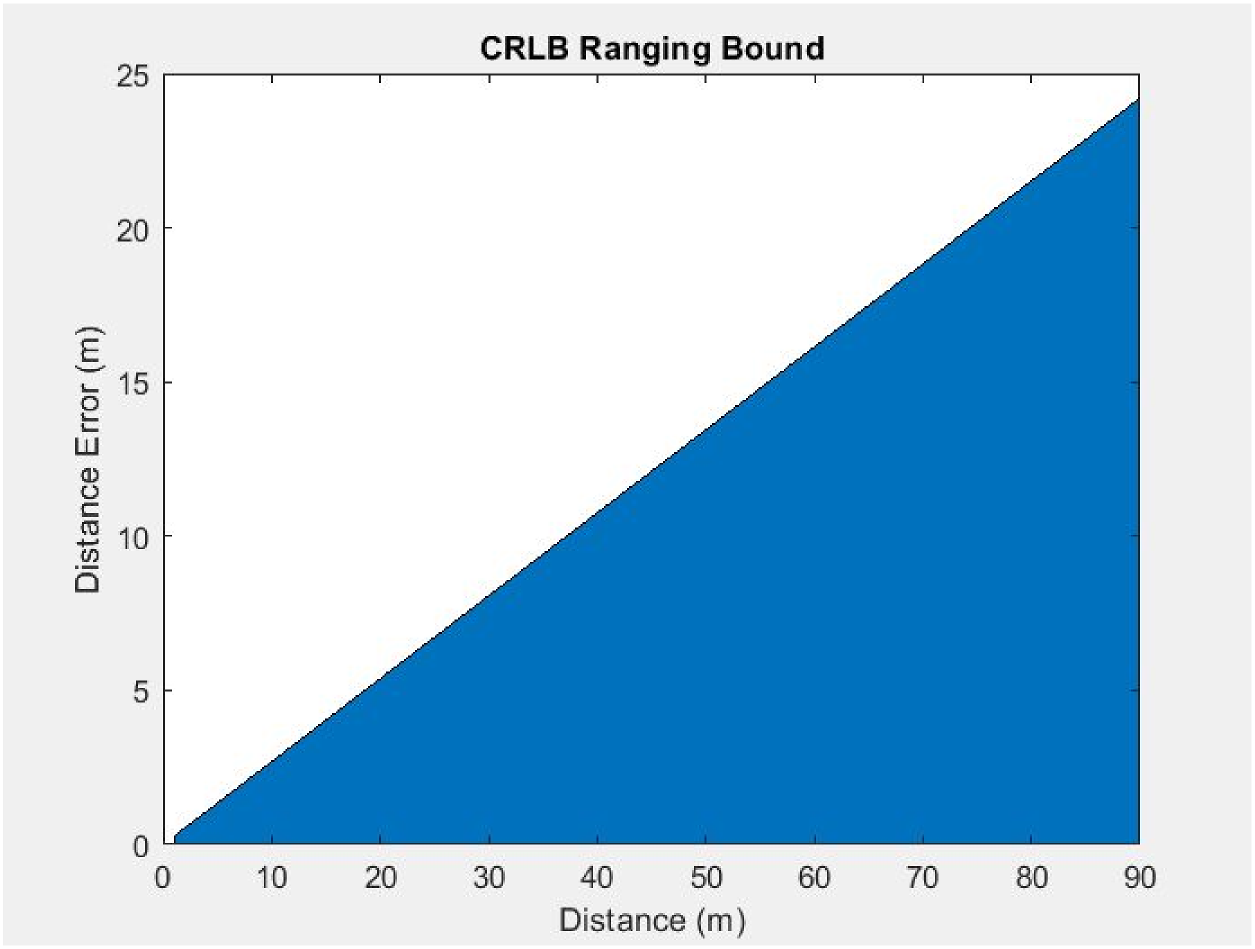


Figure 10: Ranging Error Derived from Cramer Rao Lower Bound

Figure 10 above shows the theoretical bound of our RSS localization system, meaning our system cannot exceed an accuracy at values below the blue area compared to the distance. The X axis is the distance from the drone to the radio, and the Y axis is the ranging error which was derived using the CRLB equation. These errors in this plot range from 1.345m at 5m distance all the way up to 24.21 meters at 90m. With this CRLB line we are able to compare with our test results to see how effective our ranging is.

4.2.2 Empirical Analysis of Ranging Testing and CRLB Comparison

From our ranging test, we were able to clearly see the difference that shadow fading makes. When looking at the signal strength vs time in the figure below, it can clearly be seen how the shadow fading demonstrated in the test effects the signal.

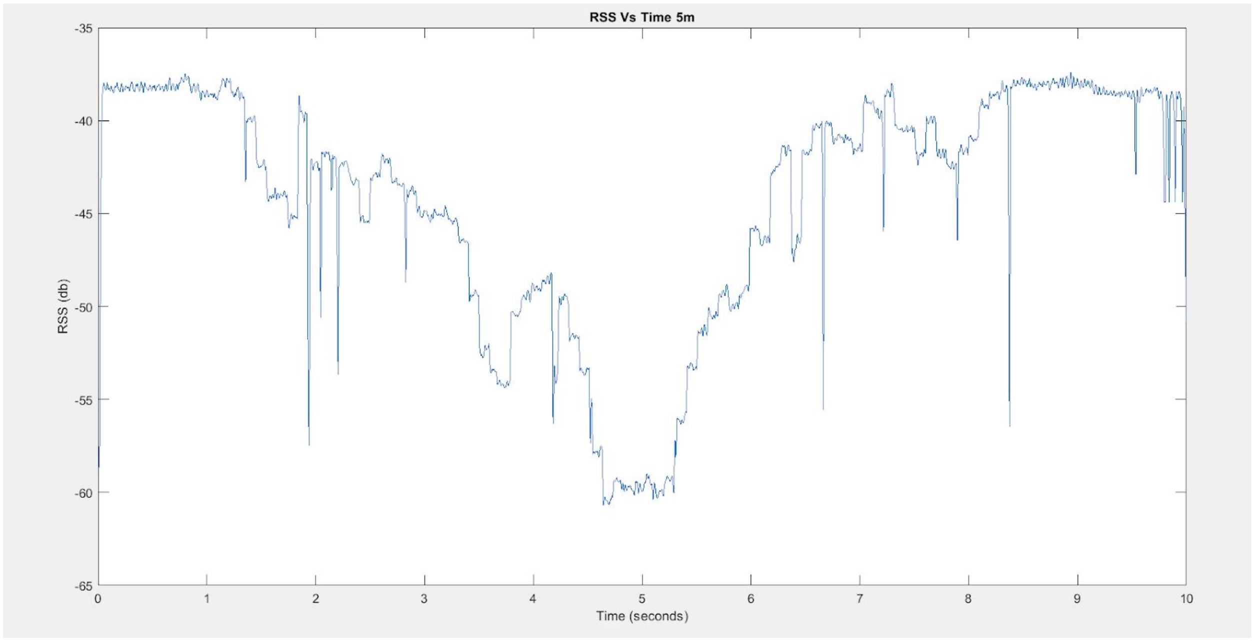


Figure 11: RSS vs Time Plot in Shadow Fading Conditions

As the drone rotates and gets fully blocked by the holders body (at 5 seconds) the signal can be seen to drop up to 20 dB. The additional spikes in signal strength represented in the Y axis are also believed to come from shadow fading. This deviation in signal can lead to a lot of error in the calculated distance in the path loss model and therefore increased error in accuracy as well.

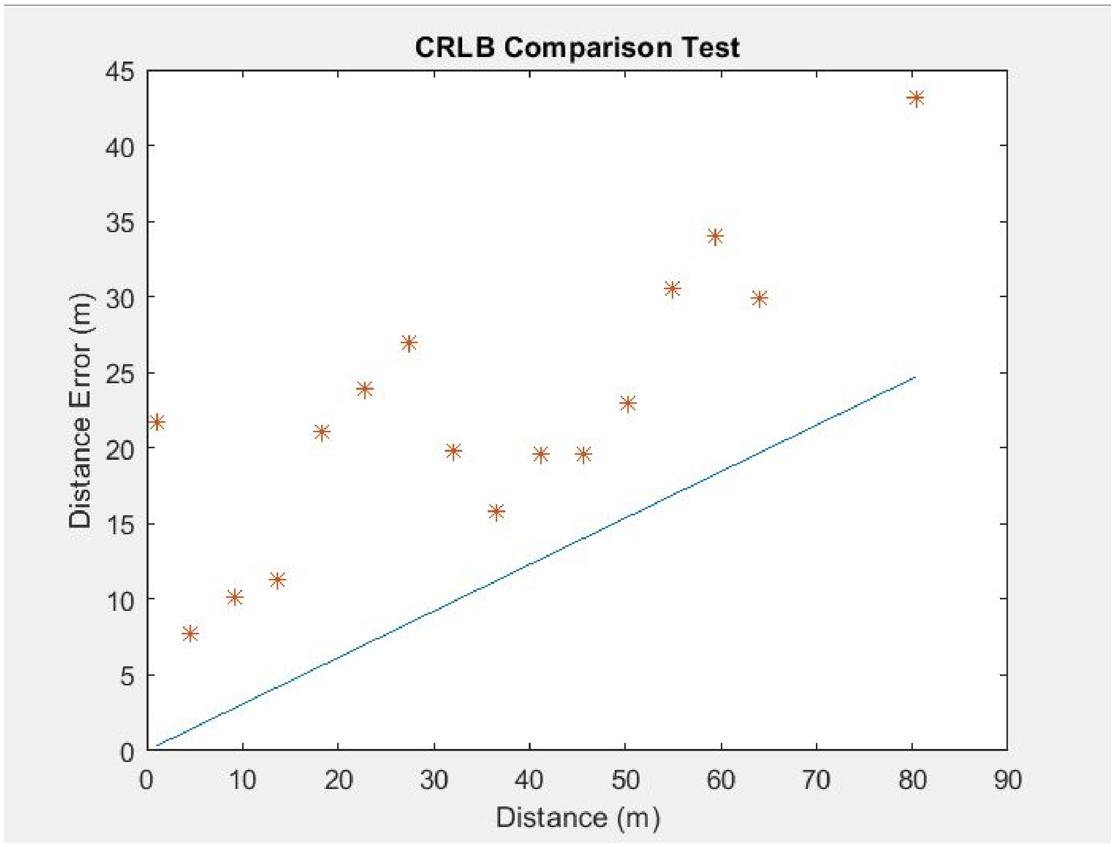


Figure 12: Ranging Error Derived from CRLB Compared with Standard Deviation Test Results

When looking at the standard deviation of error from our test with the CRLB line, we are able to analyze the efficiency of our system. The figure above is similar to that of figure 10 where the X axis is the distance from the drone to the radio and the Y axis is the ranging error from the CRLB, the orange stars in the graph are denoting the standard deviations calculated from the test measurements, which can be compared with CRLB. When comparing the CRLB line with the test it can be seen that the measured values match up well with the Cramer Rao Lower Bound as all are hovering over the Lower Bound.

4.3 Empirical Analysis of 2D Positioning Error

In 2D positioning, similar to ranging, the efficiency of the system is compared through the use of constructing a CRLB. This CRLB, which has different values at the (x, y) positions gives us our lower bound of accuracy to compare our 2D positioning system with.

4.3.1 Empirical Analysis of 2D Positioning CRLB

With our path loss model, we are able to use the 2D CRLB equation provided in the background section to calculate our standard deviation at any given (x, y) point. This standard deviation, similar to that of ranging, gives us the minimum accuracy our localization system can have in 2D at a certain (x,y) point. To visualize this accuracy, we created our theoretical 2D CRLB by calculating our CRLB error bound at every point in a 2D plane and plotting it. The resulting CRLB for our system is shown below:

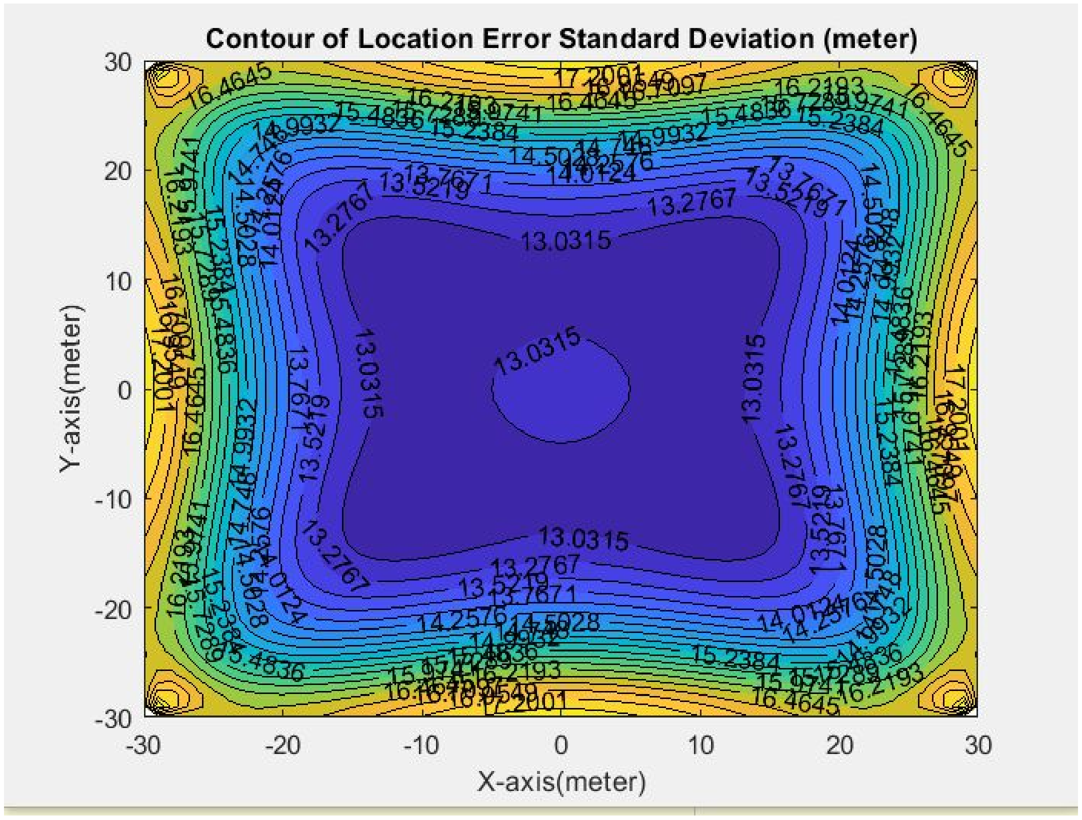


Figure 13: Theoretical 2D Error Derived from CRLB

In this contour plot, the standard deviation values are calculated given four radios, each placed at the corner of a 60 x 60 meter area where the X and Y axis denotes the distance between the radios. The plot shows higher deviation values at the edges and of the area, where it is at its longest distance for some of the radios, and at its lowest in the middle where all of the radios are generally close to the drone. The highest deviation from this plot was recorded at 17.9m and the lowest deviation was recorded at 12.7m.

4.3.2 Empirical Analysis of 2D Positioning Testing and CRLB Comparison

Through our 2D positioning simulations, we were able to simulate different positions that our system calculated hundreds of times at different locations. From these calculated positions, we calculated the standard deviation to compare with our 2D CRLB contour plot. Our setup for this simulation had each reference point in a corner of a 30 x 30 meter square, our simulated points were calculated at every point on the plane along with the CRLB.

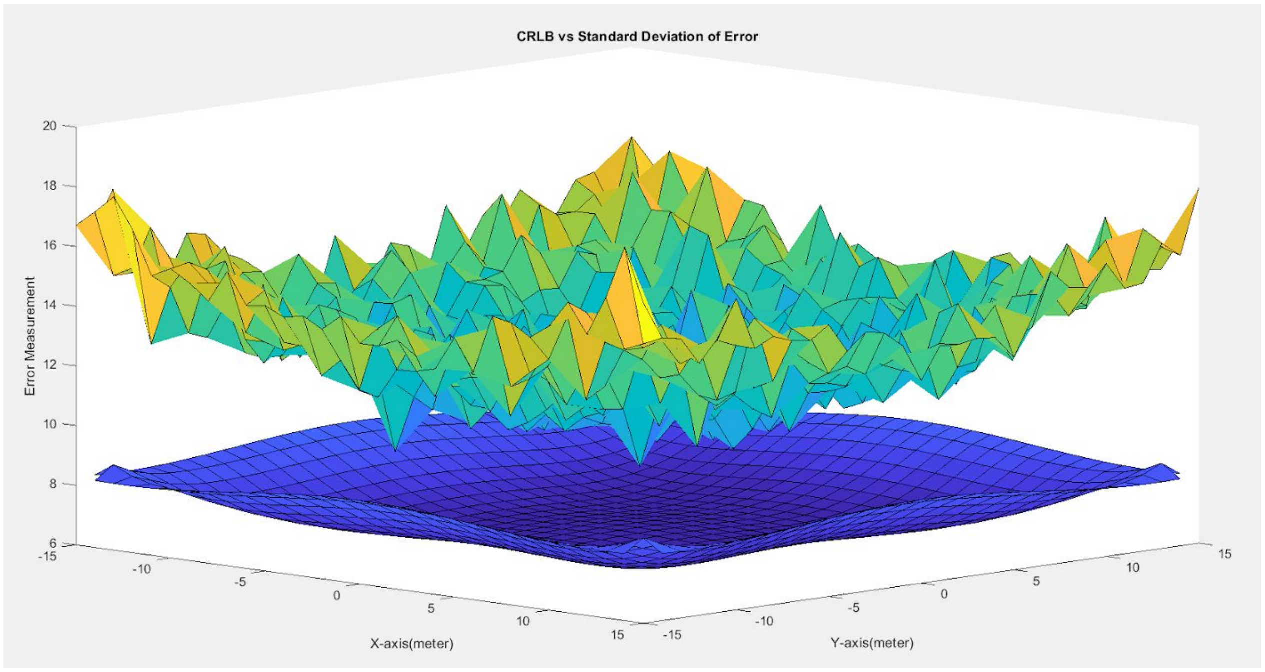


Figure 14: 2D Error Derived From CRLB Compared with Standard Deviation Results from 2D Simulator

The figure above compares the CRLB error with the standard deviation of error from the simulator, where the X and Y axis represent the location of the drone and the Z axis represents the resulting CRLB error and standard deviation from the simulator at that particular spot. In this figure, it can be seen that the simulated points illustrated by the yellow surface plot are all above the CRLB lower bound illustrated by the blue surface plot. As the simulation standard deviations are calculated, the standard deviations vary due to the random noise introduced and account for the spikes in the simulated data compared to the smooth data provided by the CRLB.When further examining the delta between the standard deviation calculated through the simulator and the CRLB values the deltas lie within the range of 2 to 11 meters.

Through this simulation, the theoretical effectiveness of our system is shown. With the CRLB being the minimum bound of error and our simulated data being close we are confident that our system has the potential to be an accurate system.

In the figure below, the positions recorded from our 2D field test are illustrated. This graph shows a plot of the drone positions that our 2D Positioning system calculated where the X and Y axis represent the location of the drone estimate. The actual location of (25,26) is surrounded by a circle, denoting the CRLB error value that is calculated at that specific position, and the blue dots are the calculated drone locations. The shadow fading present in the received signal leads to the majority of error in the location estimate, as the different signal strengths and distances will lead to different positions calculated. We believe the majority of our error from this specific test is coming from the radio 70 meters out, as the greater the distance between the radio and the drone the higher the error is in our system.

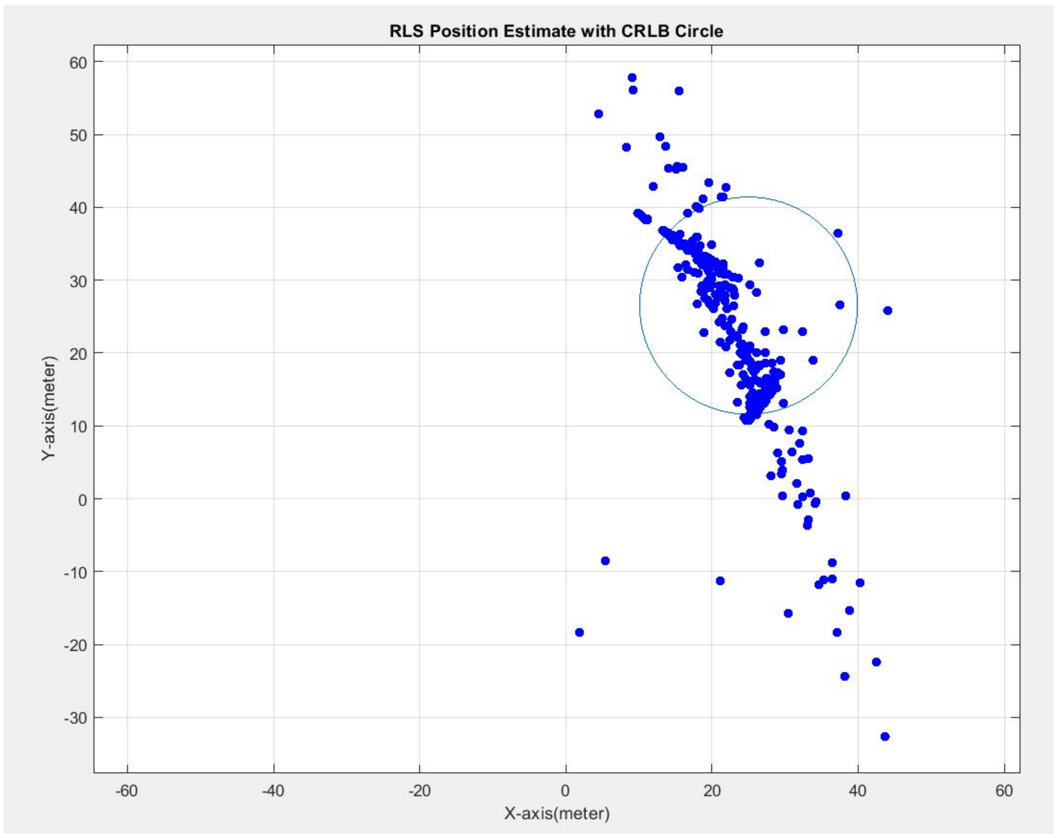


Figure 15: 2D Preliminary Testing Results of Position Calculated from Localization System

When looking at the CRLB comparison of the data, we took the standard deviation of error calculated from the 2D positioning system excluding overly inaccurate locations and compared them with the CRLB plot. In the figure below, similar to Figure 14, the X and Y axis represent the location of the drone and the Z axis represents the positioning error for the CRLB and experimental measurements. The particular red dot in the figure shows a standard deviation of testing error at 24.692 , on top of the full CRLB bound for the testing scenario. From looking at the figure, the left side denotes the garage side of the field where two of the three radio are, and the right side is the scoreboard side of the field where there is only one radio. The CRLB spike on the right is seen to be higher due to the increased distance from the other two radios and the depression on the left side can be seen to be the most accurate spot in our test scenario.

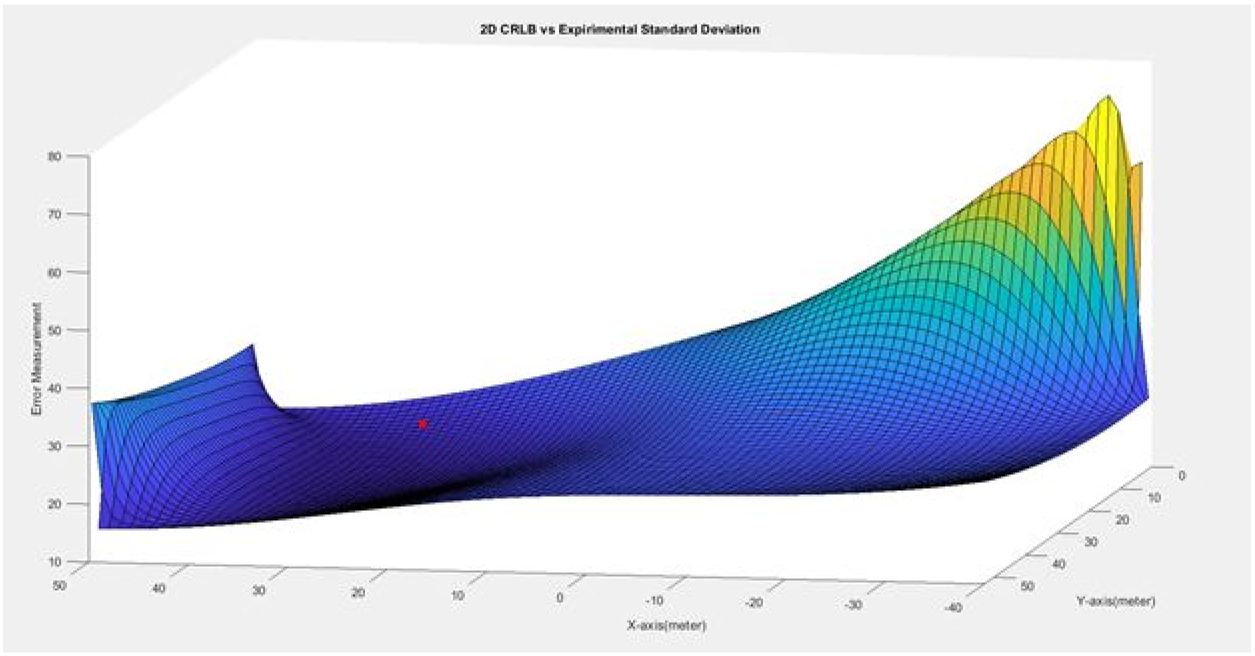


Figure 16: 2D Error Derived From CRLB Compared with Standard Deviation from Positioning Field Test

Looking ahead to the future turn, we hope to reduce this value further through testing different sampling and filtering techniques to create a cleaner signal. We also plan on getting a larger sample size of locations as our preliminary test was at one specific location.

4.4 Empirical Analysis of 3D Positioning Error

In 3D positioning, analysis on the efficiency of our localization system is the same as 2D, except now with the added dimension of height. For this analysis, calculated CRLB values at points (x, y, z) are used to examine the efficiency of our system by comparing test data at those points.

4.4.1 Empirical Analysis of 3D Positioning CRLB

Like 1D and 2D, the path loss model was used calculate the CRLB values at given (x, y, z) points. To visualize this 3D CRLB, we created a code that would calculate the CRLB at different height levels and stack them on top of each other, with different colors depicting the different levels of accuracy.

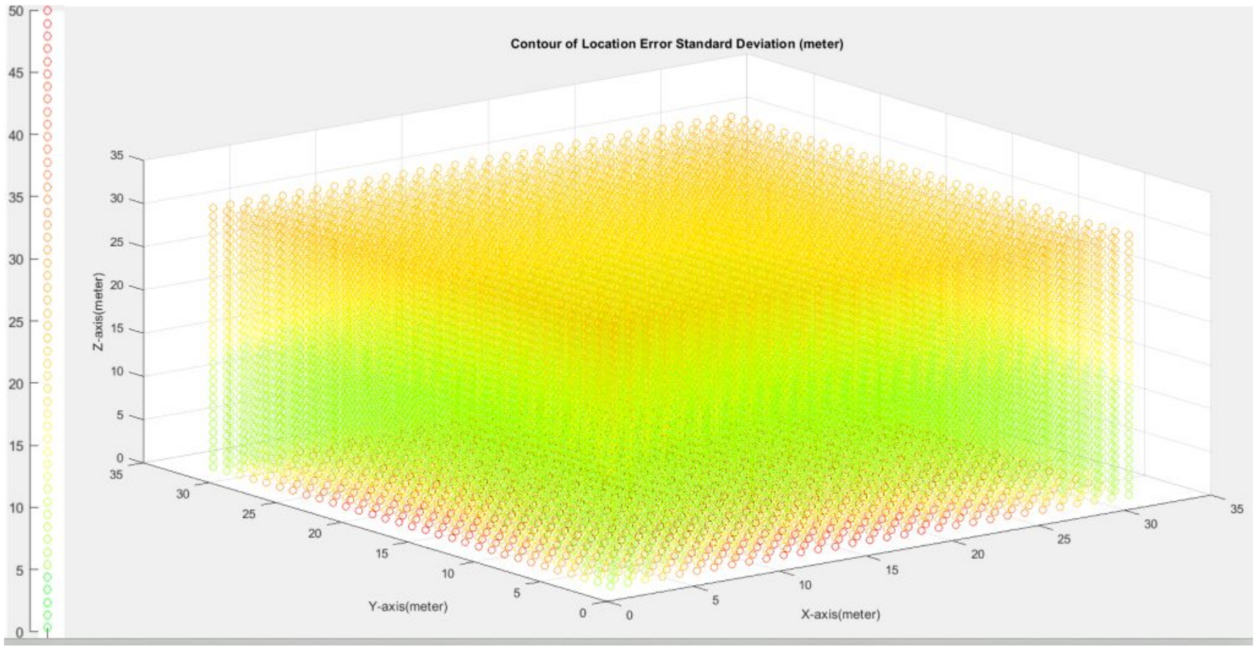


Figure 17: Full 3D Error Derived from CRLB from Heights 0 Meters to 30 Meters

Similar to the 2D plot, the orientation of the four radios are all the same, with each being at the corner of a 30m X 30m plane. In the figure above, the X, Y and Z axis all represent the location of the drone and the different colors represent the different levels of error derived from CRLB. It can be seen that higher error can be found in the lower and upper height regions of the 3D space, while the middle heights have greater accuracy. To further examine the accuracy of the localization system we also created a more detailed plot that only look at specific heights to be able to see the CRLB characteristics closer.

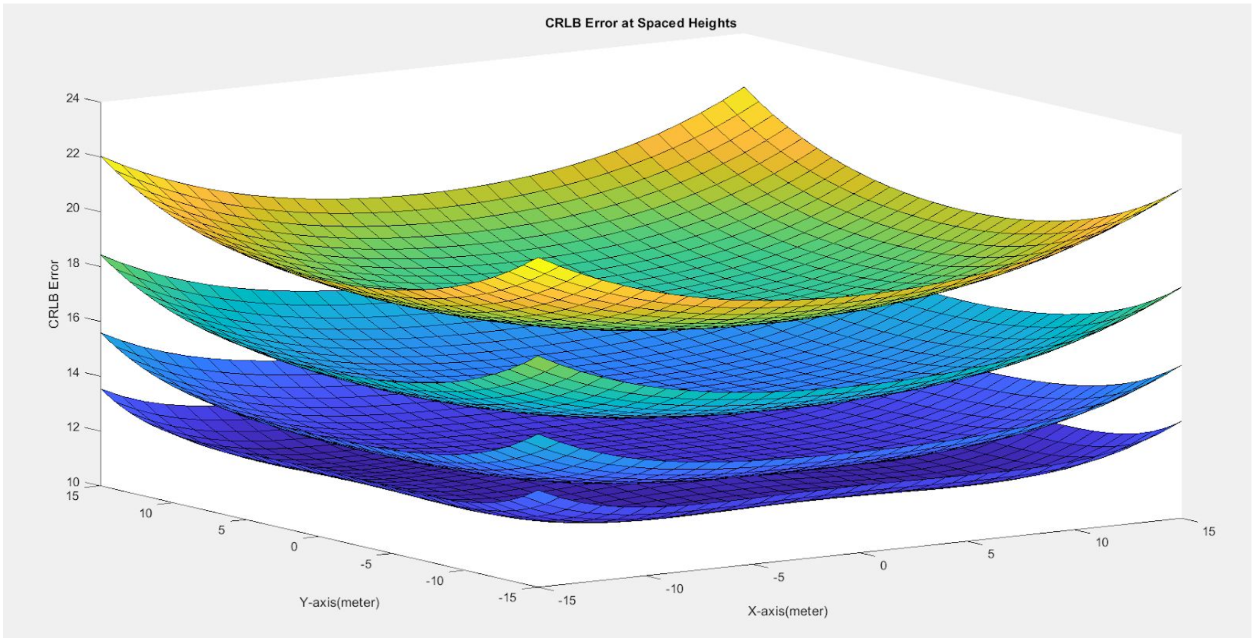


Figure 18: Multi Layered 3D Error Derived from CRLB at Heights 10, 15, 20, 25 Meters

In these figure above, it can be seen how the different heights affect the CRLB of the localization system, with the X and Y axis representing the position of the drone and the Z axis representing the error derived from CRLB. As the height goes up, so does the CRLB value of location error. At the different heights on the graph: 10, 15, 20 and 25 meters they all can be seen to have the similar characteristic of the 2D CRLB plot. At the edges near the individual radios, the CRLB is seen at its highest due to the distance away from the other 3 radios, where in the center the CRLB drops creating a smooth depression in the middle.

From the full 3D graph along, we saw the lower levels of height to return suspicious values of CRLB as we expected the CRLB values to start low and increase along with the height, through further investigation we discovered that the inaccuracy of this value comes from the calculation of the CRLB function, specifically in the inverse portion of the calculations. As explained in the methodology, in the 3D CRLB calculations the parameter H in the location estimate is calculated as:

$$H = -\frac{10}{\ln 10}\left[\sigma_1\sigma_2 \dots\dots\dots \sigma_N\right]\begin{bmatrix} \frac{x - x_1}{r_1^2} & \frac{y - y_1}{r_1^2} & \frac{z - z_1}{r_1^2} \\ \frac{x - x_2}{r_2^2} & \frac{y - y_2}{r_2^2} & \frac{z - z_2}{r_2^2} \\ \vdots & \vdots & \vdots \\ \frac{x - x_N}{r_N^2} & \frac{y - y_N}{r_N^2} & \frac{z - z_N}{r_N^2} \end{bmatrix} = \begin{bmatrix} X_1 & Y_1 & Z_1 \\ X_2 & Y_2 & Z_2 \\ \vdots & \vdots & \vdots \\ X_N & Y_N & Z_N \end{bmatrix}$$

Where (x, y, z) is the position of the drone and (x_i, y_i, z_i) are positions of the different reference points. In instances of low height z, especially when the distance between the reference points and drone is large, the value in the z column highlighted above was found to be very low. When continuing the calculation of the location estimate, to find the covariance of the location estimate the inverse of $(H^T H)$ needs to be calculated which is the root of the cause. When calculating $(H^T H)$ is turns into the multiplication of a 3xN to NX3 matrix where

$$(H^T H) = \begin{bmatrix} X_1 & Y_1 & Z_1 \\ X_2 & Y_2 & Z_2 \\ \vdots & \vdots & \vdots \\ X_N & Y_N & Z_N \end{bmatrix} * \begin{bmatrix} X_1 & X_2 & . & . & X_N \\ Y_1 & Y_2 & . & . & Y_N \\ Z_1 & Z_2 & . & . & Z_N \end{bmatrix}$$

And when multiplied through becomes

$$(H^T H) = \begin{bmatrix} X_1^2 + X_2^2 + \cdots + X_N^2 & X_1Y_1 + X_2Y_2 + \cdots + X_NY_N & X_1Z_1 + X_2Z_2 + \cdots + X_NZ_N \\ Y_1X_1 + Y_2X_2 + \cdots + Y_NX_N & Y_1^2 + Y_2^2 + \cdots + Y_N^2 & Y_1Z_1 + Y_2Z_2 + \cdots + Y_NZ_N \\ Z_1X_1 + Z_2X_2 + \cdots + Z_NX_N & Z_1Y_1 + Z_2Y_2 + \cdots + Z_NY_N & Z_1^2 + Z_2^2 + \cdots + Z_N^2 \end{bmatrix}$$

As seen in the final matrix above, the Z values which were found to be very small under low height conditions are multiplied and added together with the other X and Y values, which creates a near zero values. When calculating the inverse of a matrix, it must not be a square matrix meaning that the determinant of the matrix must not be zero, although the determinant of the matrix in these low height cases are not zero they are very close which we believe is the cause of the error. From looking at the process of finding the inverse of a matrix it can be seen that for matrix M

$$M = \begin{bmatrix} A & B & C \\ D & E & F \\ G & H & I \end{bmatrix}$$

Taking the inverse of this matrix comes out to be

$$M^{-1} = \frac{1}{\det(M)} \begin{bmatrix} (EI - FH) & -(BI - CH) & (BF - CE) \\ -(DI - FG) & (AI - CG) & -(AF - CD) \\ (DH - EG) & -(AH - BG) & (AE - BD) \end{bmatrix}$$

When this inverse of this matrix is found, it can be seen that it turns into a very large number resulting from the inverse of the determinant ends up skewing the values of the covariance function and ultimately results in a higher CRLB value.

When looking at how the position of the drone effects the CRLB in lower height conditions, the figure below illustrates the CRLB error at different height levels , with the X and Y axis being the position of the drone and the Z axis being the error measurement. Like figure 18 above, the orientation of the plot is the same, but unlike the previous figure showing the CRLB characteristics at high height temperatures, the CRLB at low height levels can be seen to be higher towards the center and lower towards the edges. In addition the lower height levels show higher CRLB values and as the height rises, the error calms down and flattens out until we get the normal CRLB of the higher height conditions. In the graph the top plot is the low level height at 2 meters and as the plots go down the height rises up to 10 meters.

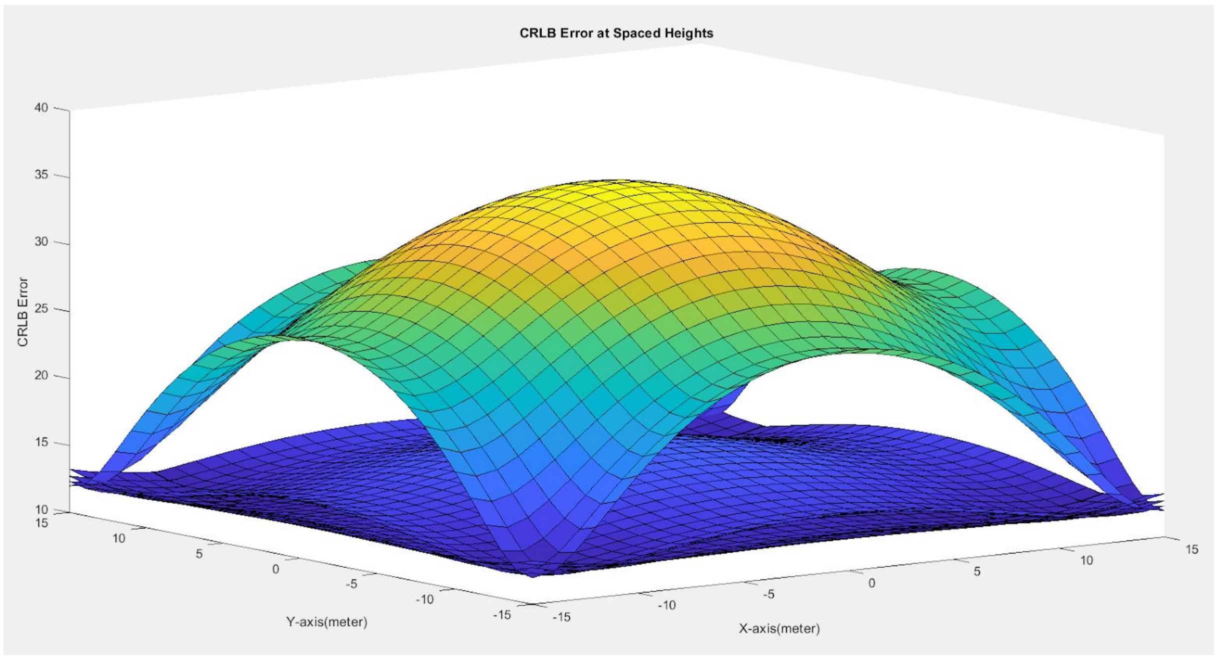


Figure 19: Multi Layered 3D Error Derived From CRLB at Heights 2, 5, 8, 10 Meters

From our investigation, we have concluded that the current CRLB function under low height parameters is unreliable and needs further investigation. For analysis we choice to test and compare under higher height conditions to obtain more reliable CRLB values as well as less interference from the ground in real experimental testing.

4.4.2 Empirical Analysis of 3D Positioning Testing and CRLB Comparison

For the data results from the 3D simulator, we ran this simulator and calculated the standard deviation of error at different points at a height of 25 meters to compare with the CRLB at 25 meters, similar to the 2D simulation. From the figure below, it can be seen that the simulated errors are all higher than the CRLB. In the figure, the X and Y axis represent the position of the drone at a height of 25 meters and the Z axis represents the error coming from CRLB along with the standard deviation of error from the simulator. The error in the 3D simulator can be seen to be higher and more volatile than that of the 2D simulation due to the increase in distance from the radios.

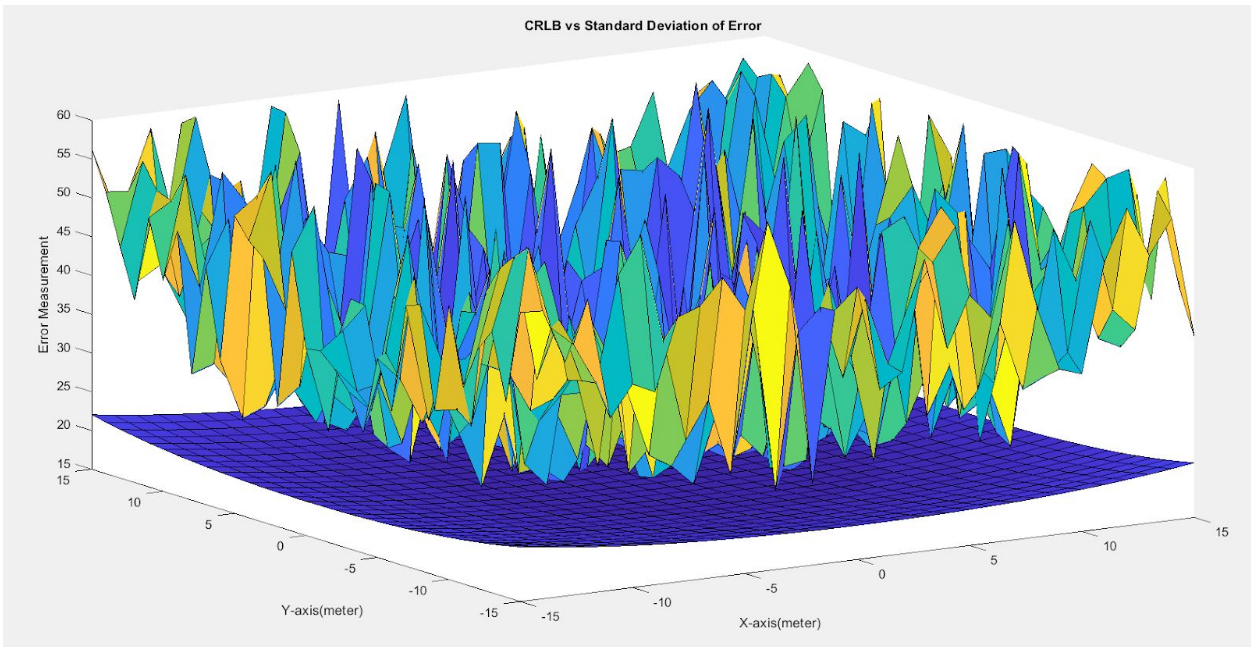


Figure 20: 3D Error Derived from CRLB Compared with Standard Deviation Results from 3D Simulator

When calculating the 3D delta between the simulation and the CRLB values, the higher spikes in error coming from the simulation lead to a greater delta in the 3D comparison. It can be seen that the lowest error delta was 4.2 meters and the highest was 41 meters.

To further analyze the accuracy of our localization system, we also ran the simulator to calculate positions and the resulting standard deviations of the simulator at different heights of the same (x, y) position. For this test, we chose the (x, y) position (0,0) and simulated the standard deviation at multiple heights. Looking at the the figure below, where the X axis represents the different heights at position (0,0) and the Y axis represents the error from CRLB along with error from simulated standard deviation, the CRLB values denoted by the blue line increase as the height increases. The simulated standard deviations denoted by the star all hover above the CRLB line, and the same curving characteristic can be seen through the simulated data.

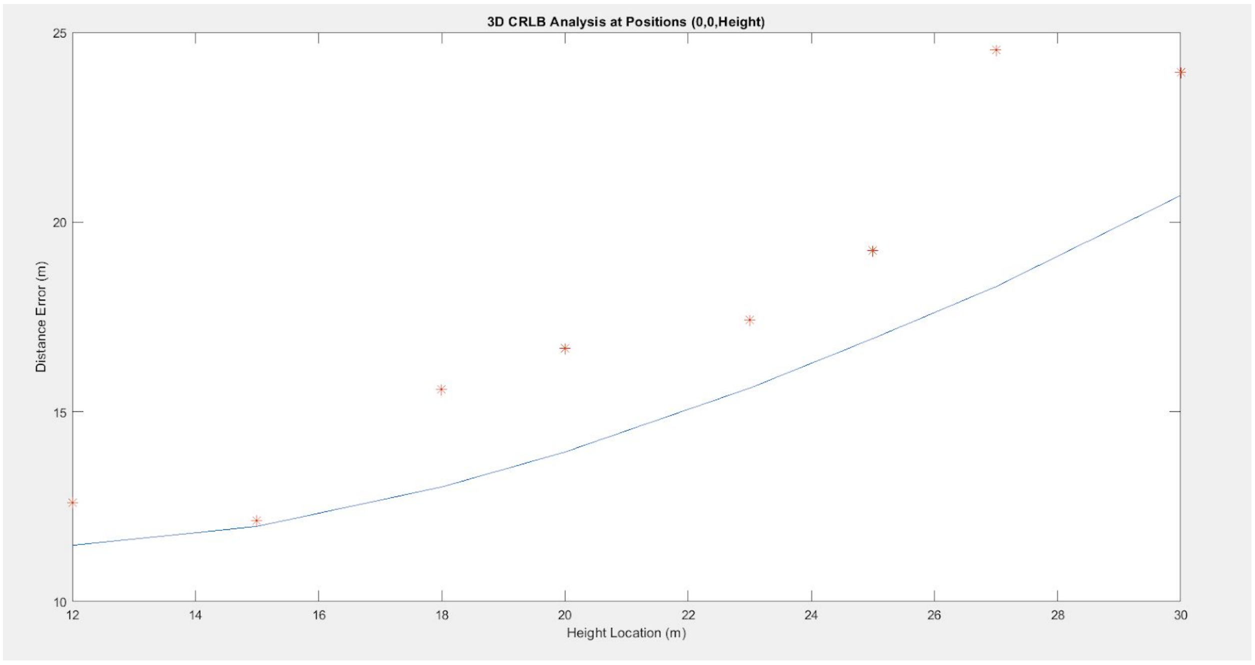


Figure 21: 3D Error Derived from CRLB Compared with Standard Deviation from 3D Simulator at Different Heights of (0,0)

In the upcoming term, once physical 3D tests outside have taken place and data has been recorded. Our team will then be able to compare actual data against the CRLB in similar fashion. One challenge ahead is the derivation of the actual distance of the drone in (x, y, z) position. Different methods on finding the position such as GPS, or using physical measurements have been discussed, and further research and consideration will take place in the upcoming term. Through the simulation data, we aim to reduce the error in our 3D localization system through different methods such as cleaning up the received signal, creating individual path loss models for radios, etc.

Conclusion and Future Work

In this project, we examined the efficiency of our localization system through the use of Cramer Rao Lower Bound. We were able to create theoretical CRLB values and compare them with experimental data from our system to test the efficiency of our localization. In the term coming ahead, the goals for the finishing project are as follows.

- Finalize real time 2D Positioning System
    - Test 2D system and Analyze
- Research and finalize best way to secure 3D position for testing (GPS, Physical Measurements, etc.)
- Finalize real time 3D Positioning System
    - Test 3D system and analyze

Suggestions for future groups working on project include:

- Developing way to authenticate drone signal vs other signal (MAC address)
- Explore other means of Localization : Time of Arrival, Angle of Arrival, etc.
- Exploring ways to make lower height CRLB values more reliable, such as using psuedo inverse for low height conditions

Appendix

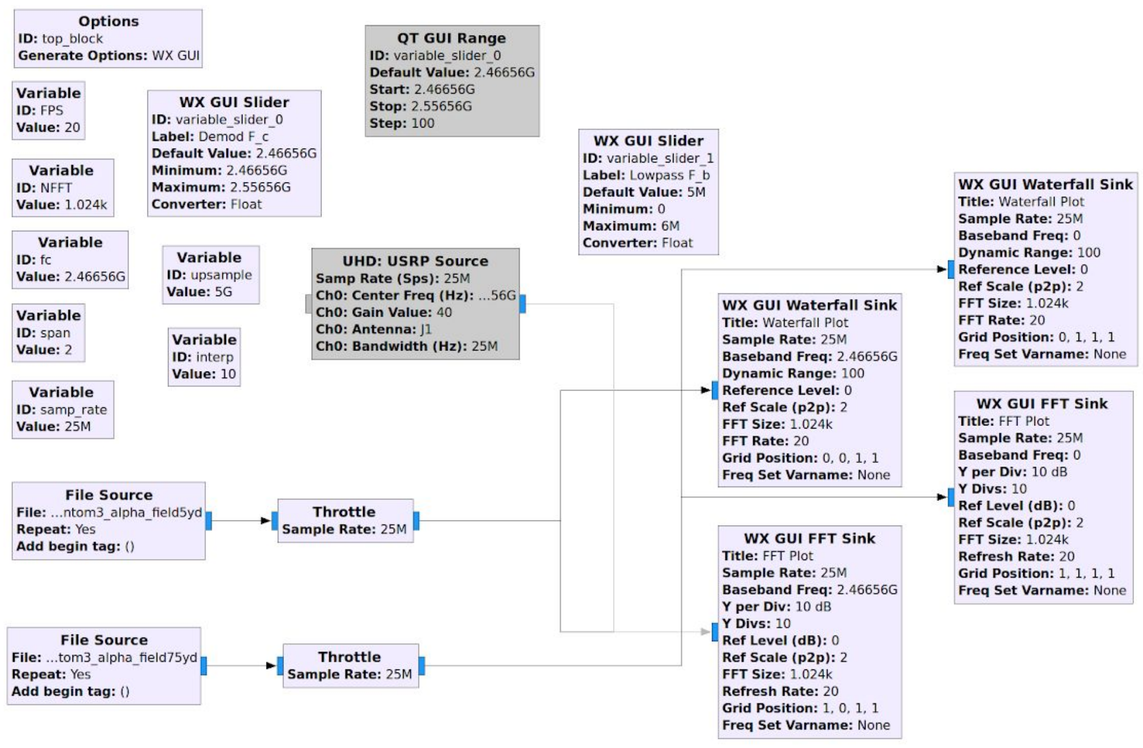
A: Federal Regulations of Drones

Our Project was based on the localization of drones, according to the Federal Aviation Association (FAA) [1] , the summary of regulations for small unmanned aircrafts weighing below 55 pounds are as follows:

- Avoid manned aircrafts and never operate in a reckless manner
- Must be within unaided sight when operating
- Must operate within daylight
- Maximum height is 400 feet from ground, must keep 400 feet away from buildings
- Maximum speed is 100mph
- Cannot fly in covered structure or covered vehicle
- Cannot fly over groups of people

B: Data Acquisition

In our project, we used software defined radios to collect data. Through these radios, GNU radio was utilized to process and relay this data to the computers. Below is an example of a GNU radio block used for data acquisition.



Options
ID: top_block
Generate Options: WX GUI

Variable
ID: FPS
Value: 20

Variable
ID: NFFT
Value: 1.024k

Variable
ID: fc
Value: 2.46656G

Variable
ID: span
Value: 2

Variable
ID: samp_rate
Value: 25M

WX GUI Slider
ID: variable_slider_0
Label: Demod F_c
Default Value: 2.46656G
Minimum: 2.46656G
Maximum: 2.55656G
Converter: Float

Variable
ID: upsample
Value: 5G

Variable
ID: interp
Value: 10

QT GUI Range
ID: variable_slider_0
Default Value: 2.46656G
Start: 2.46656G
Stop: 2.55656G
Step: 100

UHD: USRP Source
Samp Rate (Sps): 25M
Ch0: Center Freq (Hz): ...56G
Ch0: Gain Value: 40
Ch0: Antenna: J1
Ch0: Bandwidth (Hz): 25M

WX GUI Slider
ID: variable_slider_1
Label: Lowpass F_b
Default Value: 5M
Minimum: 0
Maximum: 6M
Converter: Float

File Source
File: ...ntom3_alpha_field5yd
Repeat: Yes
Add begin tag: ()

Throttle
Sample Rate: 25M

File Source
File: ...tom3_alpha_field75yd
Repeat: Yes
Add begin tag: ()

Throttle
Sample Rate: 25M

WX GUI Waterfall Sink
Title: Waterfall Plot
Sample Rate: 25M
Baseband Freq: 2.46656G
Dynamic Range: 100
Reference Level: 0
Ref Scale (p2p): 2
FFT Size: 1.024k
FFT Rate: 20
Grid Position: 0, 0, 1, 1
Freq Set Varname: None

WX GUI FFT Sink
Title: FFT Plot
Sample Rate: 25M
Baseband Freq: 2.46656G
Y per Div: 10 dB
Y Divs: 10
Ref Level (dB): 0
Ref Scale (p2p): 2
FFT Size: 1.024k
Refresh Rate: 20
Grid Position: 1, 0, 1, 1
Freq Set Varname: None

WX GUI Waterfall Sink
Title: Waterfall Plot
Sample Rate: 25M
Baseband Freq: 0
Dynamic Range: 100
Reference Level: 0
Ref Scale (p2p): 2
FFT Size: 1.024k
FFT Rate: 20
Grid Position: 0, 1, 1, 1
Freq Set Varname: None

WX GUI FFT Sink
Title: FFT Plot
Sample Rate: 25M
Baseband Freq: 0
Y per Div: 10 dB
Y Divs: 10
Ref Level (dB): 0
Ref Scale (p2p): 2
FFT Size: 1.024k
Refresh Rate: 20
Grid Position: 1, 1, 1, 1
Freq Set Varname: None

Figure B: GNU Radio Data Acquisition

C: 1D MATLAB Code

CRLB:

```
%% Set Distances and Convert to Meters
d = [5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90];
dm = .9144 * d;
%standdevRSS = [3.30 13.38 11.37 12.39 9.96 6.31 5.73 8.55 2.53 8.04 4.44 5.37 7.63 ↙
2.35 7.15 7.05 5.33];

%% Uncoment to Use Eye Distance Measurements for Calculation

%dist5 = [4.139 5.9 4.12 3.69 3.56 7.55 3.95];
%dist10 = [26.76 8.86 9.07 9.15 9.8 8.77];
% dist15 = [15.86718919 16.99 16.27 15.53];
% dist20 = [26.73 21.88 20 18.01 20.58];
% dist25 = [18.77 26.87 24.67 25.21 24.51];
% dist30 = [13.83 32.832 3.033 38.568 39.92];
% dist35 = [13.31 36.36 35.96 36.05 36.11];
% dist40 = [28.7 43.37 38.73 38.49 36.079 43.53];
% dist45 = [20.77 40.087 49.95 45.44 43.306 40.334];
% dist50 = [29.99 53.83 52.37 47.00 46.94 46.80 46.54];
% dist55 = [32.2 59.206 57.223 55.301 53.960];
% dist60 = [34 55.21 64.35 63.601 62.97 60.03];
% dist65 = [40.78 67.67 62.63 61.99 64.88 62.53];
% dist70 = [31.58 71.88 70.36 69.352 74.552];
% dist75 = [82.8 70.074 78.648 71.411 72.352 ];
% dist80 = [65 82.12 81.652 84.54 80.88 80.765];
% dist85 = [53.02 81.36 89.99 80.12 87.909 86.675];
% dist90 = [55.84];

%% Uncomment to Use CSV distance Files for Caclulation

dist5 = csvread(('csv_distance_5yd.csv'))';
dist10 = csvread(('csv_distance_10yd.csv'))';
dist15 = csvread(('csv_distance_15yd.csv'))';
dist20 = csvread(('csv_distance_20yd.csv'))';
dist25 = csvread(('csv_distance_25yd.csv'))';
dist30 = csvread(('csv_distance_30yd.csv'))';
dist35 = csvread(('csv_distance_35yd.csv'))';
dist40 = csvread(('csv_distance_40yd.csv'))';
dist45 = csvread(('csv_distance_45yd.csv'))';
dist50 = csvread(('csv_distance_50yd.csv'))';
dist55 = csvread(('csv_distance_55yd.csv'))';
dist60 = csvread(('csv_distance_60yd.csv'))';
dist65 = csvread(('csv_distance_65yd.csv'))';
dist70 = csvread(('csv_distance_70yd.csv'))';
dist75 = csvread(('csv_distance_75yd.csv'))';
dist80 = csvread(('csv_distance_80yd.csv'))';
dist85 = csvread(('csv_distance_85yd.csv'))';
dist90 = csvread(('csv_distance_90yd.csv'))';
```

```
%% Calculate Standard Deviation From Actual Distance and Plot Vs CRLB Theoretical

%standdevdist = [2.93 3.53 1.71 2.79 4.11 1.77 22.65 2.27 10.105 2.02 18.64 1.92 2.85 ↙
3.82 2.52 2.55 2.68];
standdevdist = [(CRLBstd(dist5, 5)) (CRLBstd(dist10, 10)) (CRLBstd(dist15, 15)) ↙
(CRLBstd(dist20, 20)) (CRLBstd(dist25, 25)) (CRLBstd(dist30, 30)) (CRLBstd(dist35, ↙
35)) (CRLBstd(dist40, 40)) (CRLBstd(dist45, 45)) (CRLBstd(dist50, 50)) (CRLBstd ↙
(dist55, 55)) (CRLBstd(dist60, 60)) (CRLBstd(dist65, 65)) (CRLBstd(dist70, 70)) ↙
(CRLBstd(dist75, 75)) (CRLBstd(dist80, 80)) (CRLBstd(dist85, 85)) (CRLBstd(dist90, ↙
90))];

CRLB = ( 2.302 / 10) * (2.4153/1.89) * dm;
figure
plot(dm, CRLB);hold on
plot (dm, standdevdist, '*');
title('CRLB Comparison Test')
xlabel('Distance (m)')
ylabel('Distance Error (m)')
hold off
```

Standard Deviation 1D:

```
function y = CRLBstd(distm, distance)
sum = 0;
sumlength = 0;
n = 1;
if length(distm) > 100
    distm = distm(1201:end);
end

distancem = distance * .9144;
while (n <= length(distm))
    if (distm(n) < 100)             %setting noise floor to 80 db
        sum = sum + ((distm(n) - distancem)^2);
        sumlength = sumlength + 1;
    end
    n = n + 1;
end

y = sqrt(sum / sumlength);

end
```

D: 2D MATLAB Code:

CRLB:

```
close all;clear all;clc;warning off;
%% Initialization
% Locations of Access Points
APx(1)=-40;APy(1)=0;
APx(2)=50;APy(2)=0;
APx(3)=50;APy(3)=53;
% APx(4)=-15;APy(4)=15;
%APx(5)=0;APy(5)=0;
SD=2.4153; % Standard Deviation of Shadow Fading
NUM=3; % Number of Access Points
% Locations of Receivers
pace=1;
mx=0:pace:50;
my=0:pace:50;
nxy=length(mx);
for yi=1:nxy
    for xi=1:nxy
        for i1=1:NUM
            alpha=1.8;
            r(i1,xi,yi)=sqrt((mx(xi)-APx(i1))^2+(my(yi)-APy(i1))^2); % Distance↙
Between Transmitter and Receiver
            H1(i1,xi,yi)=-10*alpha/log(10)*(mx(xi)-APx(i1))/(r(i1,xi,yi))^2; % First↙
Column of H Matrix
            H2(i1,xi,yi)=-10*alpha/log(10)*(my(yi)-APy(i1))/(r(i1,xi,yi))^2; % Second↙
Column of H Matrix
        end
        H(:,:,xi,yi)=[H1(:,xi,yi),H2(:,xi,yi)];
        Covv(:,:,xi,yi)=SD^2*((H(:,:,xi,yi))'*H(:,:,xi,yi))^(-1); % Covariance Matrix↙
of Error Estimate
        SDr(xi,yi)=sqrt(Covv(1,1,xi,yi)+Covv(2,2,xi,yi)); % Standard Deviation of↙
Location Error
    end
end
SDr=SDr';

%% Define Expirimental Results


% % Position 1 Measured Values
Position1 = [25,26]; % Actual Location
% PositionM1 = ... % Measured Locations
% [ -.1, 0;
% .1, 0;
% 0, -.1;
% 0, 0;
% -.1, 0;] ;

P1std = CRLB2Dvar(25,26); % Expirimental Measured Std Error
t1 = num2str(P1std);
c1 = cellstr(t1);

% Position 2 Measured Values
Position2 = [-12,-10]; % Actual Location
% PositionM2 = ... % Measured Locations
% [ -12.1, -10;
% -11.1, -11;
% -12, -11.1;
% -10, -10;
% -13.1, -10;] ;

P2std = CRLB2Dvar(-12,-10); % Expirimental Measured Std Error
t2 = num2str(P2std);
c2 = cellstr(t2);
%
% % Position 3 Measured Values
Position3 = [10,10]; % Actual Location
% PositionM3 = ... % Measured Locations
% [ 10.1, 11;
% 11, 10;
% 9, 9.1;
% 10, 13;
% 11.1, 10;] ;
%
P3std = CRLB2Dvar(-10,-10); % Expirimental Measured Std Error
t3 = num2str(P3std);
c3 = cellstr(t3);

Positionsx = [Position1(1) ];
Positionsy = [Position1(2) ];

Positionsxd = Positionsx + .5;
Positionsyd = Positionsy + .1;
Positionstext = [c1 ];



%% Plot Figure
figure(1)
[C h]=contourf(mx,my,SDr,20);
clabel(C,h);
hold on
scatter(Positionsx, Positionsy, 'red', 'filled');
```

```
text(Positionsxd, Positionsyd, Positionstext,'Color', 'red', 'FontSize' , 12);
xlabel('X-axis(meter)');
ylabel('Y-axis(meter)');
title('Contour of Location Error Standard Deviation (meter)');
hold off
```

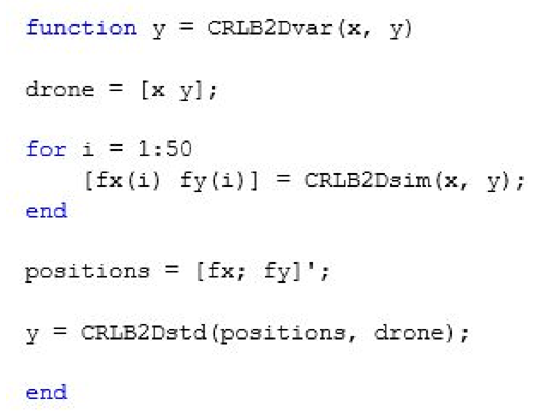
Standard Deviation:

```
function y = CRLB2Dstd(PositionM, Position)

count = 1;
lengthP = length(PositionM);
sumr = 0;

while count <= lengthP
    r = sqrt((PositionM(count)-Position(1))^2+(PositionM(count + lengthP)-Position ↙
(2))^2); % Distance Between Transmitter and Receiver
    sumr = sumr + (r^2);
    count = count + 1;
end


y = sqrt(sumr / lengthP);
end
```

2D Simulation:

```
function [fx fy] = CRLB2Dsim(x, y)

drone = [x y];

x_o = (30*rand) - 15;
y_o = (30*rand) - 15;

drone_guess = [x_o y_o];

AP=[-15,-15;15,-15;-15,15;15,15];
Po = -37;
SD = 5;
alpha = 1.809;


for i = 1:4
    d(i)=sqrt((drone(1)-AP(i,1)).^2+(drone(2)-AP(i,2)).^2);
end

for j = 1:4
    X = (2 * rand) - 1;
    P(j) = Po - 10*alpha*log10(d(j)) + X*SD;
end

for n = 1:4
    dh(n) = 10^(-1*((P(n) - Po)/alpha)/10);
end


[fx, fy] = RLS_2D(AP,drone_guess, dh);



end
```

2D Standard Deviation Stimulation:



```
function y = CRLB2Dvar(x, y)

drone = [x y];

for i = 1:50
    [fx(i) fy(i)] = CRLB2Dsim(x, y);
end

positions = [fx; fy]';

y = CRLB2Dstd(positions, drone);

end
```

2D RLS:

```
function [final_x,final_y] = RLS_2D(known_references,initial_guess,distances)
% known_references = [10,10;0,15;-5,5];
% initial_guess = [5,5];
% distances = [15,10,5];
 if size(known_references,2) ~= 2

 error('location of known reference points should be entered as Nx2 matrix');

 end

% figure(1);
% hold on
% grid on
i=1;
temp_location(i,:) = initial_guess ;
temp_error = 0 ;

for j = 1 : size(known_references,1)
    temp_error = temp_error + abs((known_references(j,1) - temp_location(i,1))^2 + ↙
(known_references(j,2) - temp_location(i,2))^2 - distances(j)^2) ;
end

estimated_error = temp_error ;
% new_matrix = [ ];
% while norm(estimated_error) > 1e-2 %iterative process for LS algorithm
for i = 1:100

    for j = 1 : size(known_references,1)  %Jacobian has been calculated in advance
        jacobian_matrix(j,:) = -2*(known_references(j,:) - temp_location(i,:)) ;   %↙
partial derivative is i.e. -2(x_1-x)
        f(j) = (known_references(j,1) - temp_location(i,1))^2 + (known_references(j, ↙
2) - temp_location(i,2))^2 - distances(j)^2 ;
    end

    estimated_error = -inv(jacobian_matrix' * jacobian_matrix) * (jacobian_matrix') * ↙
f' ; %update the U and E

    temp_location(i+1,:) = temp_location(i,:) + estimated_error' ;
%     current_point = [temp_location(i+1,1),temp_location(i+1,2)];
%     new_matrix = [ new_matrix; current_point];


%     plot(temp_location(i+1,1),temp_location(i+1,2),'rx') ; % plot
%
%     text(temp_location(i+1,1), temp_location(i+1,2)*(1 + 0.005) , num2str(i));

    i = i + 1;

end
```

```
final_x = temp_location(i,1) ;
% disp(final_x);

final_y = temp_location(i,2) ;
% disp(final_y);
```

CRLB Surface Plot Comparison & Delta:

```
function y = CRLB2D_3Dplot_2(SDr)

a = -15:15;
[X Y] = meshgrid(a,a);

for i = 1:31
    for j = 1:31
        SDex(i,j) = CRLB2Dvar((i-16),(j - 16));
        j = j + 1;
    end
    i = i + 1;
end

figure
hold on
surf(X,Y,SDr)
surf(X,Y,SDex)
xlabel('X-axis(meter)');
ylabel('Y-axis(meter)');
zlabel('Error Measurement(meter)');
title('CRLB vs Standard Deviation of Error');
hold off


delta = SDr - SDex;

figure
hold on
surf(X,Y,delta)
xlabel('X-axis(meter)');
ylabel('Y-axis(meter)');
zlabel('Error Measurement Delta');
title('Error Delta of CRLB and Simulation Standard Deviation');
hold off
```

E: 3D MATLAB Code:

Full CRLB:

```
%% Full 3D CRLB and Expirimental Layer CRLB Check Program
% To Graph Full 3D CRLB uncomment Part 1 and 2 and comment Part 3 and 4

% To Graph Single Layer Expirimental CRLB Checks uncomment Part 3 and 4 and
% comment Part 1 and 2


%% Append layers of CRLB Plots Together (Part 1)
for z=1:31
    SDr(:,:,z) = layer_3D(z);
end

%% Call FULL 3D Plotting Function (Part 2)

SDr_Plot(SDr);
 %% Define Expirimental/ Simulated Points (Part 3)

%Expirimental

% measuredv = 4;
%
% % Position 1 Measured Values
% Position1 = [0,0,5]; % Actual Location
% PositionM1 = ... % Measured Locations
% [ -.1, 0, 4.7;
% .1, 0, 5.2;
% 0, -.1 6.3;
% 0, 0, 4;
% -.1, 0 5;] ;
%
% P1std = CRLB3Dstd(PositionM1, Position1); % Expirimental Measured Std Error
% t1 = num2str(P1std);
% c1 = cellstr(t1);
%
% % Position 2 Measured Values
% Position2 = [-12,-10, 15]; % Actual Location
% PositionM2 = ... % Measured Locations
% [ -12.1, -10,14;
% -11.1, -11, 20;
% -12, -11.1, 16;
% -10, -10, 15;
% -13.1, -10, 14.4;] ;
%
% P2std = CRLB3Dstd(PositionM2, Position2); % Expirimental Measured Std Error
% t2 = num2str(P2std);
% c2 = cellstr(t2);
%
% % Position 3 Measured Values
% Position3 = [10,10,10]; % Actual Location
% PositionM3 = ... % Measured Locations
% [ 10.1, 11, 10;
% 11, 10, 10.5;
% 9, 9.1, 14;
% 10, 13, 13;
% 11.1, 10, 8;] ;
%
% P3std = CRLB3Dstd(PositionM3, Position3); % Expirimental Measured Std Error
% t3 = num2str(P3std);
% c3 = cellstr(t3);
%
% % Position 4 Measured Values
% Position4 = [8,-10,10]; % Actual Location
% PositionM4 = ... % Measured Locations
% [ 8.1, -11, 10;
% 9, -10, 10.5;
% 9, -9.1, 14;
% 10, -13, 13;
% 11.1, -10, 8;] ;
%
% P4std = CRLB3Dstd(PositionM4, Position4); % Expirimental Measured Std Error
% t4 = num2str(P4std);
% c4 = cellstr(t4);
%
%
% Positionsx = [Position1(1), Position2(1), Position3(1),Position4(1)];
% Positionsy = [Position1(2), Position2(2), Position3(2),Position4(2)];
% Positionsz = [Position1(3), Position2(3), Position3(3),Position4(3)];
% Positionstext = [c1,c2,c3,c4];
%

%Simulated

measuredv = 4;

% Position 1 Simulated Values
Position1 = [0,0,25]; % Actual Location
P1std = CRLB3Dvar(0,0,25); % Simulated Std Error
t1 = num2str(P1std);
c1 = cellstr(t1);

% Position 2 Measured Values
Position2 = [10,10, 25]; % Actual Location
P2std = CRLB3Dvar(10,10,25); % Expirimental Measured Std Error
t2 = num2str(P2std);
c2 = cellstr(t2);

% Position 3 Measured Values
Position3 = [-12,5,25]; % Actual Location
```

```
P3std = CRLB3Dvar(-12,5,25) ; % Expirimental Measured Std Error
t3 = num2str(P3std);
c3 = cellstr(t3);

% Position 4 Measured Values
Position4 = [8,-10,25]; % Actual Location
P4std = CRLB3Dvar(8,-10,25); % Expirimental Measured Std Error
t4 = num2str(P4std);
c4 = cellstr(t4);


Positionsx = [Position1(1), Position2(1), Position3(1),Position4(1)];
Positionsy = [Position1(2), Position2(2), Position3(2),Position4(2)];
Positionsz = [Position1(3), Position2(3), Position3(3),Position4(3)];
Positionstext = [c1,c2,c3,c4];


% for loop to find expirimental positions at same height
for count = 1: measuredv
    z_v{count} = find(Positionsz == Positionsz(count));
end



%% Call Layered Expirimental Check (Part 4)
plotted = zeros(1,measuredv);
plotted = plotted - 1;
for i = 1:measuredv
    if (0 == ismember(Positionsz(i), plotted))
        CRLB_3D_layer_plot(Positionsx(z_v{i}), Positionsy(z_v{i}), Positionsz(i), ↙
Positionstext(z_v{i}));
    end
    plotted(i) = Positionsz(i);
end
```

Single Layer CRLB:

```
function yf =CRLB_3D_layer_plot(Positionsx, Positionsy, Positionz, Positionstext)

x = -15:0.2:15;
y = -15:0.2:15;
height = Positionz;
AP=[-15,-15;15,-15;-15,15;15,15];
SD = 2.4153;
alpha = 1.809;
[X,Y] = meshgrid(x,y);
for j=1:1:151
    for k=1:1:151
        for i=1:1:4
            r(i)=sqrt((X(j,k)-AP(i,1)).^2+(Y(j,k)-AP(i,2)).^2+(height).^2);
            P(i)=10*alpha/log(10)*((X(j,k)-AP(i,1))/(r(i).^2));
            q(i)=10*alpha/log(10)*((Y(j,k)-AP(i,2))/(r(i).^2));
            zp(i)=10*alpha/log(10)*(height)/(r(i).^2);
            H(i,1)=P(i);
            H(i,2)=q(i);
            H(i,3)=zp(i);
        end
        Cov=SD^2*inv(H'*H);
        Z(j,k)=sqrt(Cov(1,1)+Cov(2,2)+Cov(3,3));
    end
end

%% Plot Figure with Expirimental Points
figure
contourf(X,Y,Z,'ShowText','on')
ylabel('y(distance)')
xlabel('x(distance)')
title(sprintf('Contour CRLB Plot at Height %d', Positionz))
hold on

%Plot References
known_references=AP;
referencex=known_references(:,1);
referencey=known_references(:,2);
h1=plot(referencex,referencey,'rs');
% legend([h1,h2],'Reference point', 'Simulated Error')

% Plot CRLB Values
Positionsxd = Positionsx + .5;
Positionsyd = Positionsy + .1;


h2 = scatter(Positionsx, Positionsy, 'red', 'filled');
legend([h1,h2],'Reference point', 'Simulated Error')
text(Positionsxd, Positionsyd, Positionstext,'Color', 'red');
hold off


end
```

Single Layer:

```
function yf =layer_3D(height)
%% Set up X and Y Limits, AP Points
x = -15:1:15;
y = -15:1:15;

AP=[-15,-15;15,-15;-15,15;15,15];

%% Define SD and alpha
SD = 2.4153;
alpha = 1.809;
[X,Y] = meshgrid(x,y);

%% Loop through X,Y points to Calculate CRLB variance at each point
for j=1:1:31
    for k=1:1:31
        for i=1:1:4
            r(i)=sqrt((X(j,k)-AP(i,1)).^2+(Y(j,k)-AP(i,2)).^2+(height).^2);
            P(i)=10*alpha/log(10)*((X(j,k)-AP(i,1))/(r(i).^2));
            q(i)=10*alpha/log(10)*((Y(j,k)-AP(i,2))/(r(i).^2));
            zp(i)=10*alpha/log(10)*(height)/(r(i).^2);
            H(i,1)=P(i);
            H(i,2)=q(i);
            H(i,3)=zp(i);
        end
        Cov=SD^2*inv(H'*H);
        Z(j,k)=sqrt(Cov(1,1)+Cov(2,2)+Cov(3,3));
    end
end
% figure
% contour(X,Y,Z,'ShowText','on')
% ylabel('y(distance)')
% xlabel('x(distance)')
% title('contour of CRLB in 3D ')
% hold on
% known_references=[-15,-15;15,-15;-15,15;15,15];
% referencex=known_references(:,1);
% referencey=known_references(:,2);
% h1=plot(referencex,referencey,'rs');
% legend(h1,'reference point')
%% Return CRLB of 2D Layer
yf = Z;
end
```

Standard Deviation:

```
function y = CRLB3Dstd(PositionM, Position)

count = 1;
lengthP = length(PositionM);
sumr = 0;

while count <= lengthP
        r = sqrt((PositionM(count)-Position(1))^2+(PositionM(count + lengthP)- ↙
Position(2))^2 + +(PositionM(count + (lengthP *2))-Position(3))^2); % Distance ↙
Between Transmitter and Receiver
        sumr = sumr + (r^2);
        count = count + 1;
end


y = sqrt(sumr / lengthP);
end
```

CRLB Plot:

```
function y = SDr_Plot(SDr)

pace=1;
mx=0:pace:30;
my=0:pace:30;
mz=0:pace:30;
nxyz=length(mx);

figure(1)
SDPlot(1, 1, 1, SDr(1, 1, 1));
xlabel('Distance X m');
ylabel('Distance Y m');
zlabel('Height m');
rotate3d on;
hold on

for zi=1:nxyz
    for yi=1:nxyz
        for xi=1:nxyz
            SDPlot(xi, yi, zi, SDr(xi, yi, zi));
        end
    end
end

xlabel('X-axis(meter)');
ylabel('Y-axis(meter)');
zlabel('Z-axis(meter)');
title('Contour of Location Error Standard Deviation (meter)');

hold off
end
```

```
function y = SDPlot(x, y, z, SDR)

factor35 = 1/35;

factor20 = 1/20;

factor50 = 1/50;

% colorv10 = .5 + (factor20 * SDR);


if (SDR <= 5)
    scatter3(x, y, z,  'MarkerEdgeColor', [0 1 0]);
end

if (SDR <= 15 && SDR > 5)
    colorv10 = .5 + (factor20 * (SDR - 10));
    scatter3(x, y, z, 'MarkerEdgeColor', [colorv10 1 0]);
end

if (SDR <= 50 && SDR > 15)
    colorv35 = (factor35 * (SDR - 15));
    scatter3(x, y, z, 'MarkerEdgeColor', [1 (1-colorv35) 0]);
end

if (SDR <= 100 && SDR > 50)
    colorv40 = (factor50 * (SDR - 50));
    scatter3(x, y, z, 'MarkerEdgeColor', [(1-colorv40) 0 0]);
end


if (SDR > 100)
    scatter3(x, y, z, 'MarkerEdgeColor', [0 0 0]);
end
```

3D Simulation:

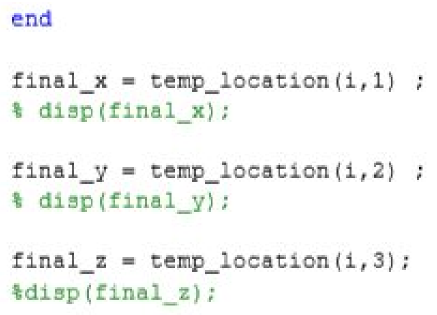
```
function [fx fy fz] = CRLB3Dsim(x, y, z)

drone = [x y z];

x_o = (30*rand) - 15;
y_o = (30*rand) - 15;
z_o = (30*rand);
drone_guess = [x_o y_o z_o];

AP=[-15,-15,0;15,-15,0;-15,15,0;15,15,0];
Po = -37;
SD = 5;
alpha = 1.809;


for i = 1:4
    d(i)=sqrt((drone(1)-AP(i,1)).^2+(drone(2)-AP(i,2)).^2+(drone(3).^2));
end

for j = 1:4
    X = (2 * rand) - 1;
    P(j) = Po - 10*alpha*log10(d(j)) + X*SD;
end

for n = 1:4
    dh(n) = 10^(-1*((P(n) - Po)/alpha)/10);
end


[fx, fy, fz] = RLS_3D(AP,drone_guess, dh);



end
```

3D Standard Deviation Simulation:

```
function y = CRLB3Dvar(x, y, z)

drone = [x y z];

for i = 1:100
    [fx(i) fy(i) fz(i)] = CRLB3Dsim(x, y, z);
end

positions = [fx; fy; fz]';

y = CRLB3Dstd(positions, drone);

end
```

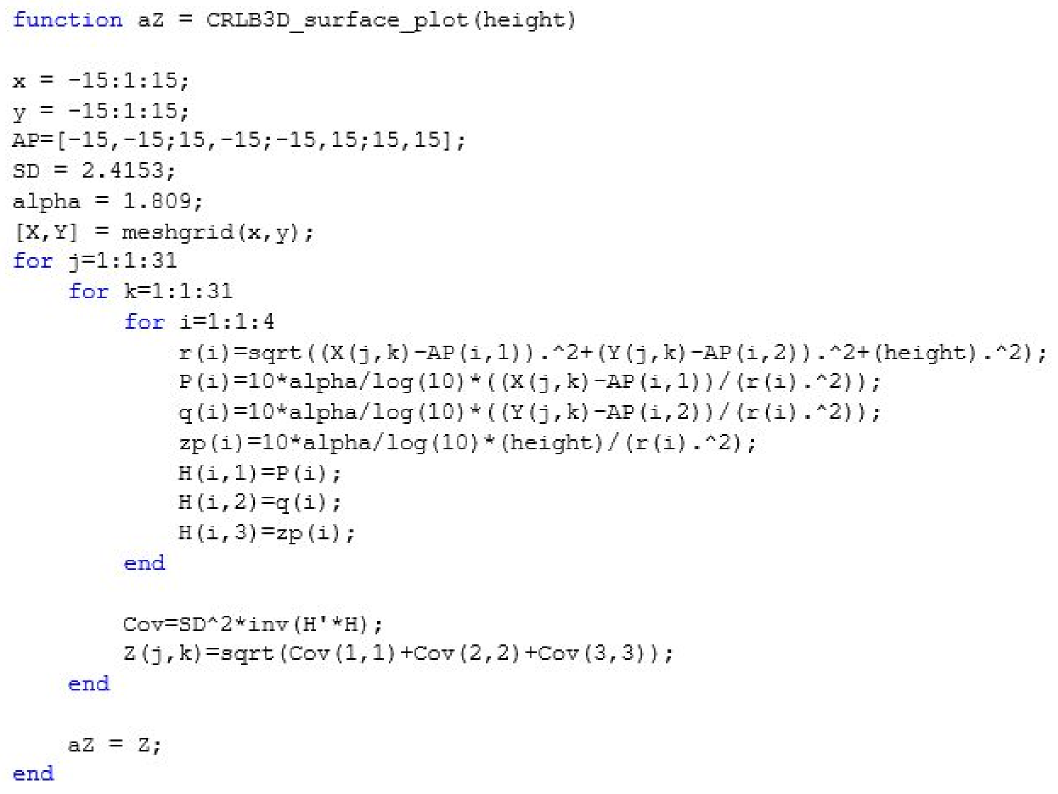
3D RLS:

```
function [final_x,final_y, final_z] = RLS_3D(known_references,initial_guess, ↙
distances)
% known_references = [-15,-15, 0;15,-15, 0;-15,15, 0;15,15, 0];
% initial_guess = [0,0,10];
% distances = [24,24,24,24];
 if size(known_references,2) ~= 3

 error('location of known reference points should be entered as Nx2 matrix');

 end

% figure(1);
% hold on
% grid on
i=1;
temp_location(i,:) = initial_guess ;
temp_error = 0 ;

for j = 1 : size(known_references,1)
    temp_error = temp_error + abs((known_references(j,1) - temp_location(i,1))^2 + ↙
(known_references(j,2) - temp_location(i,2))^2 + (temp_location(i,3))^2 - distances ↙
(j)^2) ;
end

estimated_error = temp_error ;
% new_matrix = [ ];
% while norm(estimated_error) > 1e-2 %iterative process for LS algorithm
for i = 1:100

    for j = 1 : size(known_references,1)  %Jacobian has been calculated in advance
        jacobian_matrix(j,:) = -2*(known_references(j,:) - temp_location(i,:)) ;   %↙
partial derivative is i.e. -2(x_1-x)
        f(j) = (known_references(j,1) - temp_location(i,1))^2 + (known_references(j, ↙
2) - temp_location(i,2))^2 + (temp_location(i,3))^2 - distances(j)^2 ;
    end

    estimated_error = -inv(jacobian_matrix' * jacobian_matrix) * (jacobian_matrix') * ↙
f' ; %update the U and E

    temp_location(i+1,:) = temp_location(i,:) + estimated_error' ;
%     current_point = [temp_location(i+1,1),temp_location(i+1,2)];
%     new_matrix = [ new_matrix; current_point];


%     plot(temp_location(i+1,1),temp_location(i+1,2),'rx') ; % plot
%
%     text(temp_location(i+1,1), temp_location(i+1,2)*(1 + 0.005) , num2str(i));

    i = i + 1;
```



```
end

final_x = temp_location(i,1) ;
% disp(final_x);

final_y = temp_location(i,2) ;
% disp(final_y);

final_z = temp_location(i,3);
%disp(final_z);
```

Single Layer Return:



```
function aZ = CRLB3D_surface_plot(height)

x = -15:1:15;
y = -15:1:15;
AP=[-15,-15;15,-15;-15,15;15,15];
SD = 2.4153;
alpha = 1.809;
[X,Y] = meshgrid(x,y);
for j=1:1:31
    for k=1:1:31
        for i=1:1:4
            r(i)=sqrt((X(j,k)-AP(i,1)).^2+(Y(j,k)-AP(i,2)).^2+(height).^2);
            P(i)=10*alpha/log(10)*((X(j,k)-AP(i,1))/(r(i).^2));
            q(i)=10*alpha/log(10)*((Y(j,k)-AP(i,2))/(r(i).^2));
            zp(i)=10*alpha/log(10)*(height)/(r(i).^2);
            H(i,1)=P(i);
            H(i,2)=q(i);
            H(i,3)=zp(i);
        end

        Cov=SD^2*inv(H'*H);
        Z(j,k)=sqrt(Cov(1,1)+Cov(2,2)+Cov(3,3));
    end

    aZ = Z;
end
```

Multi Layer Surface Plot:

```
heights = [2 5 7 9 12];

a = -15:15;

[X Y] = meshgrid(a,a);

for j = 1:4
    SDr(:,:,j) = CRLB3D_surface_plot(heights(j));
    j = j+1;
end

figure
hold on

for i = 1:4
    surf(X,Y,SDr(:,:,i))
    i = i+1;
end
xlabel('X-axis(meter)');
ylabel('Y-axis(meter)');
zlabel('CRLB Error');
title('CRLB Error at Spaced Heights');
hold off
```

References

[1]DJI. “Phantom 3 Professional - Specs, FAQ, Tutorials, Downloads and DJI GO - DJI.” *DJI Official*, www.dji.com/phantom-3-pro/info.

[2]Ettus Research. “USRP2.” *Ettus Knowledge Base*, Aug. 2016, kb.ettus.com/USRP2.

[3] Federal Aviation Administration. “Fact Sheet – Small Unmanned Aircraft Regulations (Part 107).” *FAA Seal*, 19 Sept. 2014, www.faa.gov/news/fact_sheets/news_story.cfm?newsId=22615.

[4] Pahlavan, Kaveh, and Prashant Krishnamurthy. *Principles of Wireless Access and Localization*. Wiley, 2013.

[5] Tian, Zilu. *RSS-Based 3D Drone Localization and Performance Evaluation*. *RSS-Based 3D Drone Localization and Performance Evaluation*.