Major Qualifying Project

RSS-Based 3D Drone Localization and Performance Evaluation

Zilu Tian
ztian@wpi.edu

Submitted to:

Professor Kaveh Pahlavan(ECE)
Professor Suzanne Weekes (MA)

Aug. 25, 2016 - May 2, 2017

Acknowledgment

The four-term project *RSS-Based 3D Drone Localization and Performance Evaluation* has three team members: Chen Li, Yu Li, and Zilu Tian. Chen Li and Yu Li are graduate students who undertook this topic as their direct research project in A and B terms. We would like to express our sincere gratitude to our advisors, Professor Kaveh Pahlavan from ECE department and Professor Suzanne Weekes from MA department, for their thoughtful insight and patient guidance. We would also like to extend our gratitude to the ECE department at Worcester Polytechnic Institute (WPI) for funding our project.

Abstract

This project is motivated by identifying the location of a trespassing drone. A drone is often equipped with a wireless camera which makes it vulnerable to the misuse of harassment, stalking, and other illegal activities. To mitigate this issue, we are interested in determining the location of a drone based on channel modeling, which studies the characteristics of signal propagation between transmitters (e.g. APs) and receivers (base stations). We implemented three localization algorithms: maximum likelihood estimation (MLE), weighted centroid, and recursive least square method (RLS). We also compared these algorithms with Cramer-Rao lower bound (CRLB), a theoretical lower bound for an unbiased estimator.

Contents

1 Introduction **6**

1.1 Project Description 6

1.2 Report Outline 6

2 Background **8**

2.1 Introduction to Path-loss Model 8

2.2 Design of Algorithms 8

2.2.1 Maximum Likelihood Estimation 8

2.2.2 Weighted Centroid 8

2.2.3 Recursive Least Square Method 9

2.3 Introduction to Cramer-Rao Lower Bound 9

3 Methodology **10**

3.1 Scenario Analysis and Path-loss Model 10

3.2 Implementation of Algorithms 14

3.2.1 Maximum Likelihood Estimation 15

3.2.2 Weighted Centroid 16

3.2.3 Recursive Least Square Method 19

3.3 Calculation of Cramer-Rao Lower Bound 23

4 Results and Discussions **27**

4.1 Presence of Significant Measurement Error 27

4.2 Performance Comparison in 3D 28

5 Conclusions and Future Work **30**

A Federal Regulations of Unmanned Aircraft Systems **31**

B Data Collection **32**

C Text Parsing in Python **33**

D MATLAB Implementation of 2D MLE **35**

E MATLAB Implementation of 2D Weighted Centroid **37**

F MATLAB Implementation of 2D RLS **38**

G MATLAB Implementation of 3D RLS **40**

H MATLAB Implementation of 2D CRLB **43**

I MATLAB Implementation of 3D CRLB **44**

J GPS Conversion to NE coordinate **45**

K Motion Detection **46**

L MATLAB Implementation of Fourier Transform of Received Signals **48**

List of Figures

3.1 Experiment for LoS measurements . . . 10
3.2 Experiment for NLoS measurements . . . 11
3.3 Measurement locations on campus . . . 11
3.4 Design of Example for LoS Measurement . . . 12
3.5 Path-loss Model (Track) . . . 12
3.6 Path-loss Model (Quad) . . . 13
3.7 Path-loss Model (Stratton Hall) . . . 13
3.8 Path-loss Model (3D,LoS) . . . 14
3.9 Path-loss Model (3D,NLoS) . . . 14
3.10 Maximum Likelihood Estimation - 4 Base Stations . . . 15
3.11 Maximum Likelihood Estimation in 3D . . . 16
3.12 Centroid Method Illustrated . . . 17
3.13 Circles with Radius d_i Centered at Each Base Station . . . 18
3.14 Weighted Centroid Estimation with Weight Factor $\frac{1}{d}$ . . . 18
3.15 Weighted Centroid Estimation with Weight Factor $\frac{1}{d^2}$ . . . 19
3.16 Cramer-Rao Lower Bound of 8 Base Stations . . . 25
3.17 Cramer-Rao Lower Bound in 3D, H=12 . . . 26
4.1 Significant Measurement Error . . . 27
4.2 Significant Measurement Error - Weighted Centroid Method . . . 28
4.3 Significant Measurement Error - Recursive Least Square . . . 28
4.4 Evaluate MLE and RLS using CRLB . . . 29
B.1 A screen copy of WirelessMon in action . . . 32
J.1 Outdoor Scenario for GPS Measurement . . . 45
K.1 FFT of RSS with upwards motion . . . 46
K.2 FFT of RSS with downwards motion . . . 46
K.3 FFT of RSS with no motion . . . 47

1 Introduction

Our project is motivated by localizing a trespassing drone. A commercial drone is often equipped with a wireless camera which makes it vulnerable to the misuse of harassment, stalking, wiretapping, and other illegal activities [3]. Multiple incidents had happened in the recent years involving regulations of private drones [2]. To alleviate this issue, we are interested in determining the location of an undesirable drone based on our knowledge of channel modeling, which studies the characteristics of signal propagation between transmitters and receivers.

To transmit back images, most drones have a wireless module which serves as an access point (AP) to transmit Wi-Fi signal that can be measured using software such as WirelessMon. The design of experiment for data collection includes four or more base stations. At each base station, more than 100 readings are gathered. Three localization algorithms are used to process data: maximum likelihood estimation, weighted centroid method, and recursive least square algorithm. For 2D measurement, a phone in the hot-spot mode served as an AP was put on the ground. Four base stations were connected to the AP and measured the signal strength of the AP from predefined distances. In 3D, phone is bundled with a drone hovering steadily at various height. The actual location of the AP is recorded in both GPS reading and north east coordinate. Path-loss model is constructed by finding the least square fit of the scatter plot of received signal strength (RSS) vs distances in the log scale. The distance between the AP and each base station is calculated based on RSS readings using path loss model.

1.1 Project Description

The goal of our project is to localize an undesirable drone based on RSS readings at different base stations. We purchased a DJI Phantom 3 Basic for our experiment but realized that the AP was built in the remote controller rather than the drone. Though there was still signal transmitting back to the controller from the wireless camera, we were unable to detect such signals using WirelessMon, which could only read the RSS of an AP. An alternative to WirelessMon was Savvius Omnipeek, a commercial network analysis software that could detect signals from such transmitters, which was available for $1194 at the time of writing. Due to the budget limitation, we did not pursue further and bundled a phone with the drone to study the localization problem when signals were transmitted from the drone. Please note that drones are subject to federal regulations, as detailed in Appendix A.

The first part of the project is largely consisted of experiment design and data collection, which is detailed in Appendix B and C. After analyzing RSS readings, we apply three localization algorithms to approximate the position of the AP. The estimation error is defined as the distance between the estimated location of the AP and the actual location. The algorithms are evaluated based on the estimation error and comparison with Cramer-Rao lower bound.

1.2 Report Outline

This report is organized as following. Chapter 2 Background provides an overview of concepts such as path-loss model, localization algorithms, and Cramer-Rao lower bound. The implementations of these concepts

are detailed in Chapter 3 Methodology. In Chapter 4 Results and Discussions, we will show the performance evaluation among the algorithms. Finally, we will summarize the conclusion of this project and future works in Chapter 5.

The appendices include all the codes implemented for this project. Apart from localization, we also tested for motion detection using RSS readings, which is documented in Appendix K and L.

2 Background

In this chapter, we first explain the concept of path-loss model. Next, we present an introduction to three algorithms that we considered for localization: maximum likelihood estimation, weighted centroid, and recursive least square method. Finally, we describe what Cramer-Rao lower bound is and how to calculate it.

2.1 Introduction to Path-loss Model

A path loss model is a statistical propagation model that describes received signal strength P_r of a transmitter at distance d, as defined in [5].

$$P_r = P_0 - 10\alpha \log d + X \tag{1}$$

where P_r is the received signal power, or received signal strength (RSS), expressed in the unit dBm. Power x[mW] can be converted to dBm by

$$x[\text{mW}] = 10 \log_{10} \frac{x}{1\text{mW}}[\text{dBm}].$$

α is the distance power gradient of the environment characterizing how fast a signal decays as the distance increases. $X \sim N(0, \sigma^2)$ is a random variable describing the effects of shadow fading.

In our project, we measured P_r and the standard deviation of the noise σ at each base station and calculated α using least square best fit line of the scatter plot of P_r and $\log d$.

2.2 Design of Algorithms

We considered three localization algorithms: maximum likelihood estimation, weighted centroid, and recursive least square method.

2.2.1 Maximum Likelihood Estimation

Maximum likelihood estimation of a parameter θ is the value of $\hat{\theta}$ that maximizes a given likelihood function. From the path loss model, we have

$$d(p_r) = 10^{\frac{p_r - (p_0 + X)}{10\alpha}}.$$

Based on this equation, we can draw an annulus centered at each base station with inner radius

$$r_{\text{inner}} = d(p_{\text{avg}} + 2\sigma)$$

and outer radius is defined as

$$r_{\text{outer}} = d(p_{\text{avg}} - 2\sigma).$$

The location of the AP is the center of the most densely overlapped region.

2.2.2 Weighted Centroid

Weighted centroid method is a modification of centroid method, which can be summarized mathematically in the equation below for the estimated AP location (x, y):

$$x = \sum_{i=1}^{n} w_i x_i \quad y = \sum_{i=1}^{n} w_i y_i$$

where (x_i, y_i) is the location of the ith base station. w_i is the weight factor which is not unique. We explored two choices, as described in the section Methodology:

$$w_{1i} = \frac{\frac{1}{d_i^2}}{\sum_{j=1}^{n} \frac{1}{d_j^2}} \quad w_{2i} = \frac{\frac{1}{d_i}}{\sum_{j=1}^{n} \frac{1}{d_j}}.$$

2.2.3 Recursive Least Square Method

Recursive least square method estimates the position of AP (x, y) as the minimizer of the sum of error functions

$$E = \sum_{i=1}^{n} f_i^2$$

where (x_i, y_i) is the location of the ith base station and f_i is the error function for each base station. The choice of f_i can also be vary. We considered 2 options, as described in the section Methodology:

$$f_{1i}(x, y) = \sqrt{(x_i - x)^2 + (y_i - y)^2} - d_i^2$$

$$f_{2i}(x, y) = (x_i - x)^2 + (y_i - y)^2 - d_i^2.$$

Solving the minimization problem requires an iterative approach, and we compared both Newton's method and Gauss-Newton's method.

2.3 Introduction to Cramer-Rao Lower Bound

The Cramer-Rao Lower Bound (CRLB) is a theoretical lower bound with the property that

$$\text{var}(\hat{\theta}(Y)) \geq \text{CRLB} = \frac{1}{I(\theta)}$$

for any unbiased estimator $\hat{\theta}(Y)$ of parameter θ with observation variable Y. An estimator $\hat{\theta}(Y)$ of Y is unbiased if

$$E[\hat{\theta}(Y)] = \theta(Y).$$

$I(\theta)$ is the Fisher matrix that measures the amount of information an observable random variable Y carries about parameter θ, which is defined as the expectation of the square of relative steepness of the likelihood function

$$I(\theta) = E\left[\left(\frac{\frac{\partial}{\partial\theta} p(y;\theta)}{p(y;\theta)}\right)^2\right] = E\left[\left(\frac{\partial \ln p(y;\theta)}{\partial\theta}\right)^2\right] = -E\left[\frac{\partial^2 \ln p(y;\theta)}{\partial\theta^2}\right].$$

In our case, observable random variable $Y = P_r$, $\theta = P_0 - 10\alpha \log \text{d}$. Since $X \sim \mathcal{N}(0, \sigma)$, the likelihood function is

$$p(y;\theta) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y-\theta)^2}{2\sigma^2}}.$$

3 Methodology

In this chapter, we first define two scenarios tested in our project, line-of-sight (LoS) and non-line-of-sight (NLoS). Measurements were taken at various locations on campus. The path-loss model calculated at each location is also shown. Then we describe the implementation of localization algorithms as well as CRLB.

3.1 Scenario Analysis and Path-loss Model

LoS refers to the situation when an AP and a base station align in a line without any obstacle in between them. When a transmitter is separated from the receiver by an object, we call the setting NLoS. The following two drawings illustrate the experiment design for both scenarios.

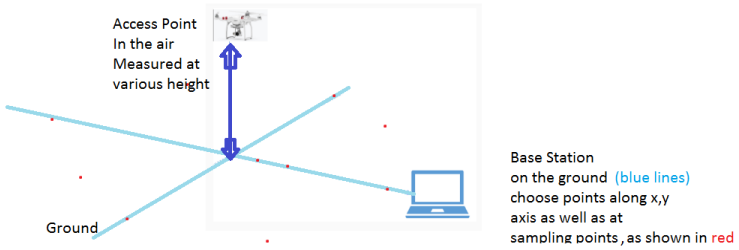


Figure 3.1: Experiment for LoS measurements

Figure 3.1 depicts the experiment design for LoS measurement. Each base station is a laptop and the AP is the drone. The ground is shown as x-y plane in light blue, and the base station(s) can be placed at various locations on the ground, as indicated in the red dots.

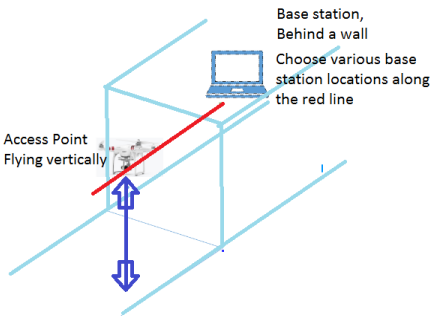


Figure 3.2: Experiment for NLoS measurements

Figure 3.2 illustrates the design for NLoS measurement. The obstacle (wall) is drawn in light blue with the base station and AP placed on different side of the wall. The drone is controlled at different heights, as indicated in dark blue line, and a base station is placed along designated red line.

For 2D experiment when the AP is on the ground, we measured RSS readings in LoS scenario at different locations on campus, including outside Stratton Hall, on the quad, and on the track, as shown in Figure 3.3. Measurement locations are highlighted in black rectangle.

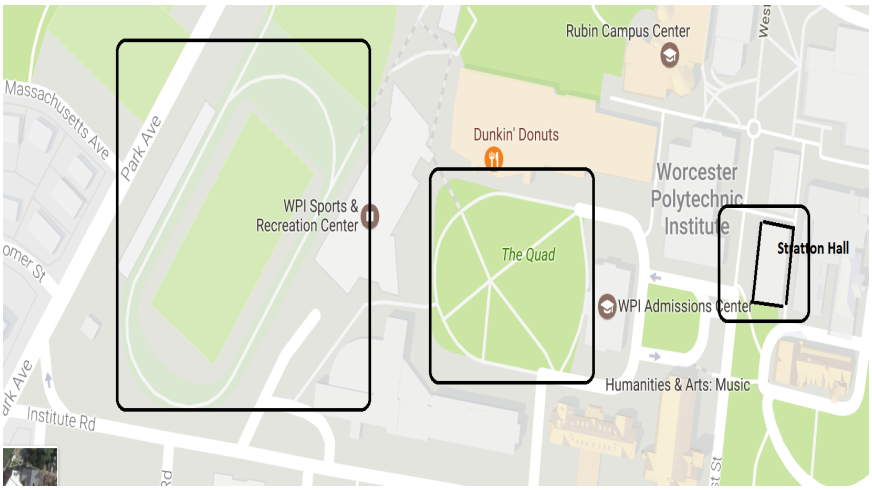


Figure 3.3: Measurement locations on campus

The experiment for LoS measurement is shown in Figure 3.4. Base stations are marked in yellow pins, labeled as reference points. The position of the AP is marked as H on this map. To collect data for channel modeling, the AP is fixed and the base station is placed along marked distances. For localization experiment, base stations are fixed and AP is placed at different positions.

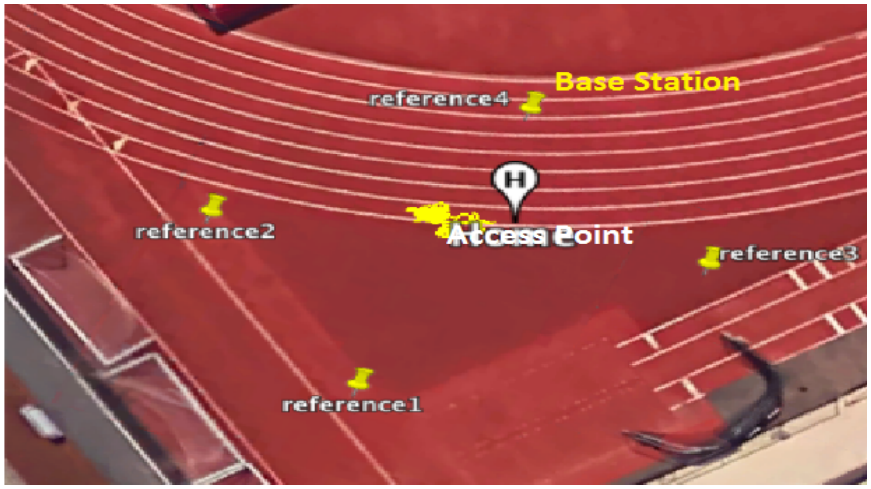


Figure 3.4: Design of Example for LoS Measurement

After acquiring data from different locations on campus, namely on the track, on the quad, and outside Stratton Hall, we noticed that distance power gradient α calculated using measurements from different locations differ significantly, as illustrate in following examples.

The measurements shown in Figure 3.5 were taken on the track. The path-loss model was

$$P_r = -63 - 26 \log_{10} d \quad \alpha = 2.6.$$

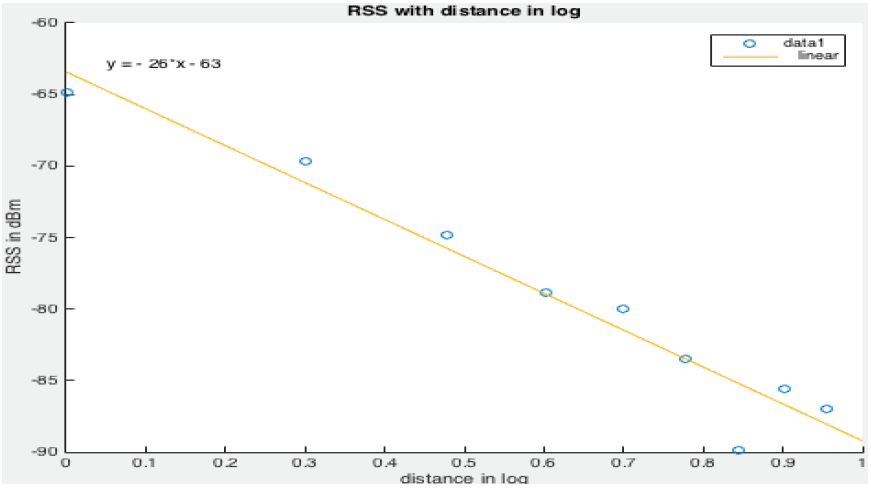


Figure 3.5: Path-loss Model (Track)

The measurements shown in Figure 3.6 were taken on the quad. The path-loss model was

$$P_r = -47 - 36 \log_{10} d \quad \alpha = 3.6.$$

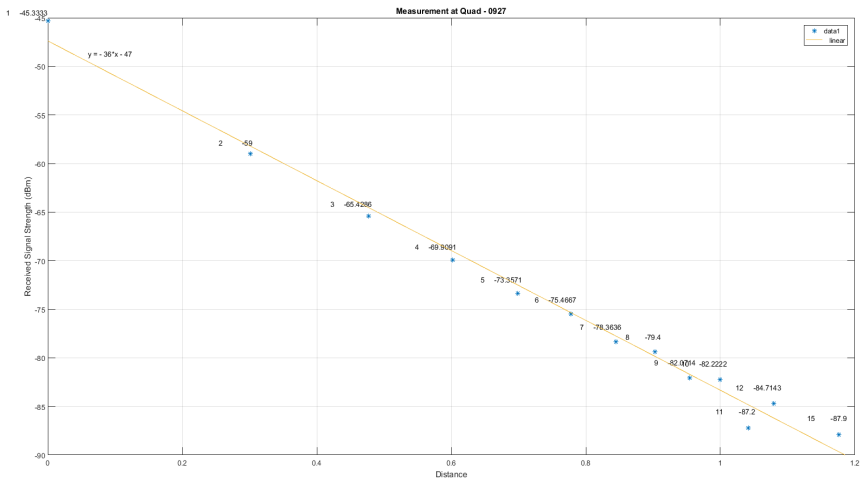


Figure 3.6: Path-loss Model (Quad)

The measurements shown in Figure 3.7 were taken in front of the Stratton Hall. The path-loss model was

$$P_r = -48 - 29 \log_{10} d \quad \alpha = 2.9.$$

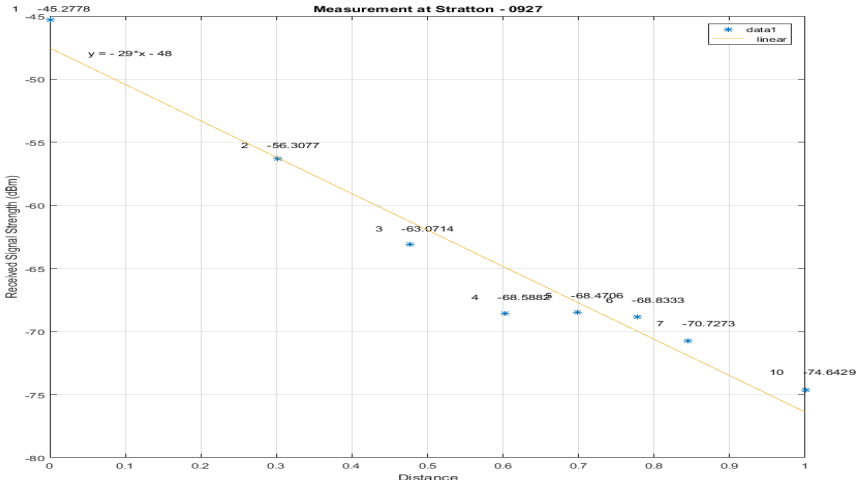


Figure 3.7: Path-loss Model (Stratton Hall)

Note that hardware equipment can also introduce measurement errors. Readings using different network interface card may not be the same.

Measurements in 3D LoS scenario exhibited less variations comparing with measurements in 2D LoS. From Figure 3.8, the path-loss model was

$$P_r = -38 - 20 \log_{10} d \quad \alpha = 2$$

which matches the standard for free space specified in 802.11b. The distance d is calculated using Euclidean norm for 3D.

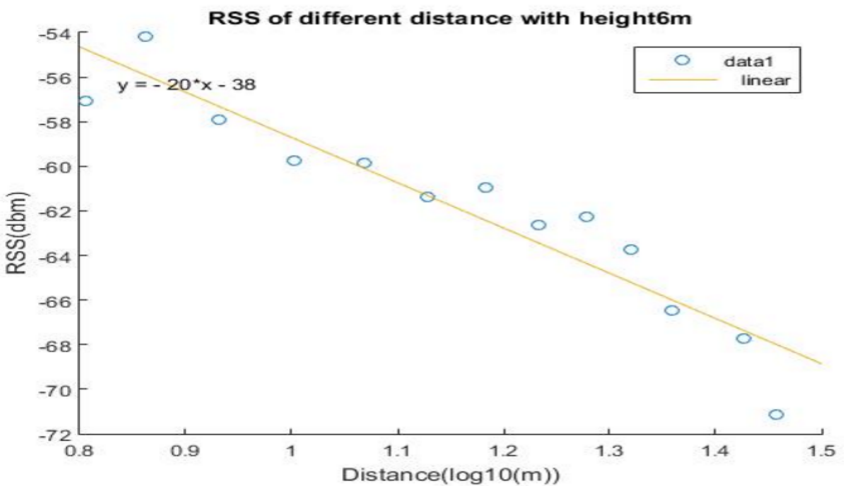


Figure 3.8: Path-loss Model (3D,LoS)

For 3D NLoS scenario, measurements in Figure 3.9 were taken when the AP was 3.5m above the ground. The path-loss model was

$$P_r = -27 - 32 \log_{10} d \quad \alpha = 3.2$$

which was close to the standard $\alpha = 3.5$ as in 802.11b for urban area. d is calculated using Euclidean norm for 3D.

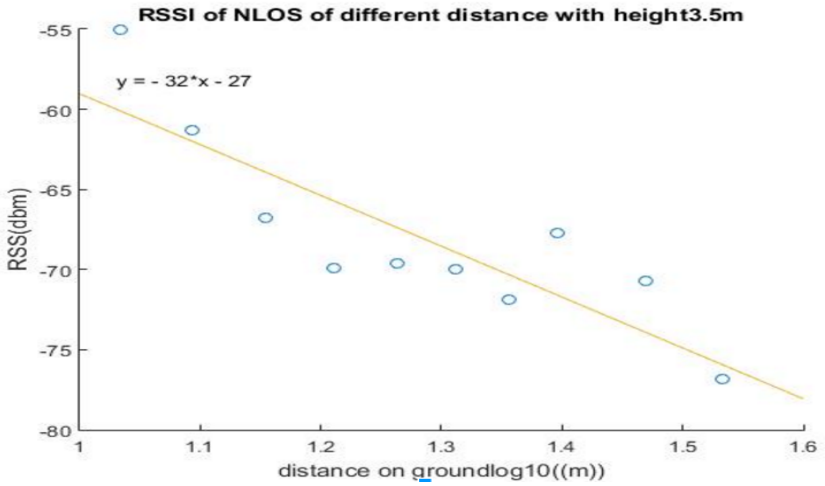


Figure 3.9: Path-loss Model (3D,NLoS)

3.2 Implementation of Algorithms

Once we have path-loss model and RSS readings, we applied three localization algorithms to estimate the location of AP: Maximum Likelihood Estimation, Weighted Centroid, and Recursive Least Square Estimation. All algorithms were implemented in MATLAB as attached in Appendix D through Appendix G.

3.2.1 Maximum Likelihood Estimation

Maximum likelihood estimation of a parameter θ is the value of $\hat{\theta}$ that maximizes a given likelihood function. From the path loss model, we have

$$d(p_r) = 10^{\frac{p_r - (p_0 + X)}{10\alpha}}.$$

Based on this equation, we can draw an annulus centered at each base station with inner radius

$$r_{\text{inner}} = d(p_{\text{avg}} + 2\sigma)$$

and outer radius

$$r_{\text{outer}} = d(p_{\text{avg}} - 2\sigma).$$

Based on different situations, one can adjust the multiples of σ to other constant values to achieve a reasonable overlapping region (e.g. search space). The location of the AP is the center of the most densely overlapped region, as illustrated in Figure 3.10.

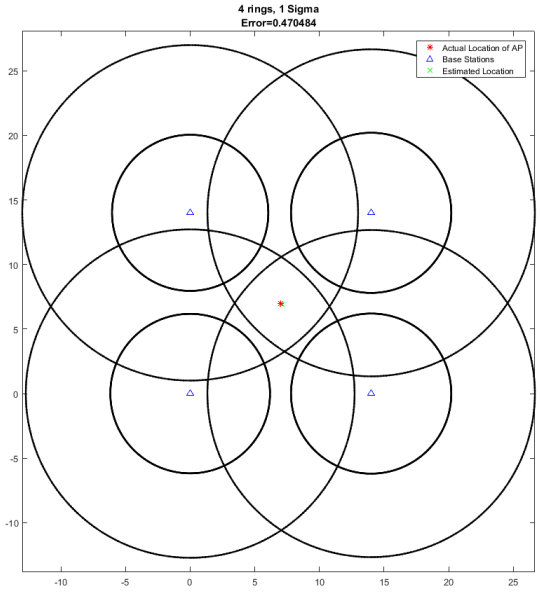


Figure 3.10: Maximum Likelihood Estimation - 4 Base Stations

In Figure 3.10, the AP location is highlighted in red star and the estimated AP position is labeled in green cross. The estimation error is 0.47m.

The computation time for MLE in 3D increases dramatically comparing to 2D. Figure 3.11 illustrates the application of MLE in 3D with 4 base stations. The algorithm for MLE 3D is an expansion based on 2D MLE and is not provided in Appendix due to inefficient computation speed.

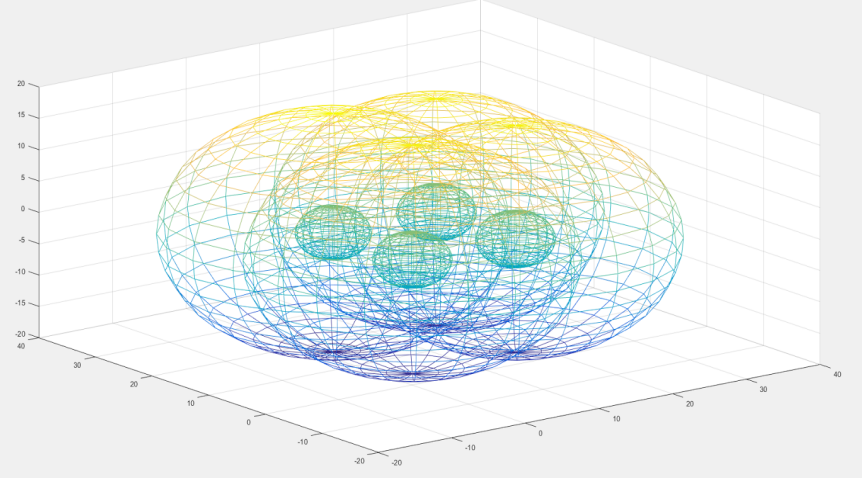


Figure 3.11: Maximum Likelihood Estimation in 3D

3.2.2 Weighted Centroid

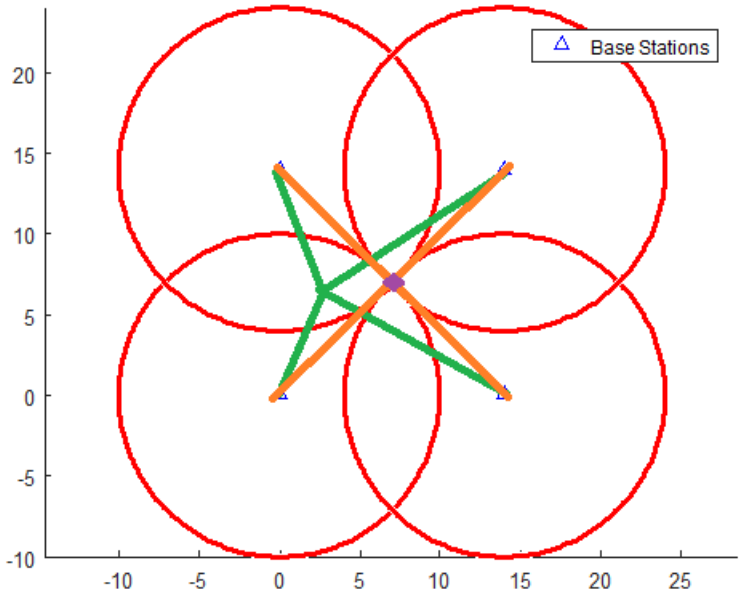
Weighted Centroid is another commonly used algorithm in localization. We first consider a simpler case, the centroid estimation, which can be summarized as

$$(x, y) = \frac{1}{n} \sum_{i=1}^{n} (x_i, y_i)$$

where (x, y) is the estimated AP position, n is the number of base stations, and (x_i, y_i) is the location of the ith base station. The equation can also be derived from minimizing a cost function J defined as the sum of distances from the AP to the center of each circle,

$$J = \sum_{i=1}^{n} \sqrt{(x_i - x)^2 + (y_i - y)^2}$$

as illustrated in Figure 3.12:



From Triangle Inequality, it is clear that the center of the base stations, highlighted in purple, is the optimal point that minimizes the distance from any point to the center of each circle.

Figure 3.12: Centroid Method Illustrated

To minimize J, set $\frac{\partial J}{\partial x} = 0$, $\frac{\partial J}{\partial y} = 0$. We have

$$\frac{\partial J}{\partial x} = 0 \implies \sum_{i=1}^{n}((x_i - x)^2 + (y_i - y)^2)^{-\frac{1}{2}}(x - x_i) = 0$$
$$\frac{\partial J}{\partial y} = 0 \implies \sum_{i=1}^{n}((x_i - x)^2 + (y_i - y)^2)^{-\frac{1}{2}}(y - y_i) = 0$$
$$((x_i - x)^2 + (y_i - y)^2)^{-\frac{1}{2}} \geq 0 \implies x = \frac{1}{n}\sum_{i=1}^{n} x_i, y = \frac{1}{n}\sum_{i=1}^{n} y_i$$

Centroid method is easy to implement but less accurate. One improvement is to assign a weight factor to each base station representing the inverse relationship between the RSS and distance. The modified centroid method including the weight factor w_i can be expressed as:

$$x = \sum_{i=1}^{n} w_i x_i \quad y = \sum_{i=1}^{n} w_i y_i$$

where weight factor w_i can take on different values

$$w_{1i} = \frac{\frac{1}{d_i}}{\sum_{j=1}^{n}\frac{1}{d_j}} \quad w_{2i} = \frac{\frac{1}{d_i^2}}{\sum_{j=1}^{n}\frac{1}{d_j^2}}.$$

As shown in the following example, w_{2i} has a slightly lower estimation error than w_{1i} and is implemented for weighted centroid algorithm.

In this experiment, base stations were located at(0,0), (0,14), (14, 0), and (14, 14) respectively. RSS readings collected at each base station in the order as presented were [-55, -54, -59, -62][dBm]. Figure 3.13 shows the location of each base station and the radius of each circle calculated using path-loss model described before in (3.2.1) with P_r=P_{avg}. Figure 3.14 contains the estimation result using w_{1i} and Figure 3.15 contains the estimation result using w_{2i}.

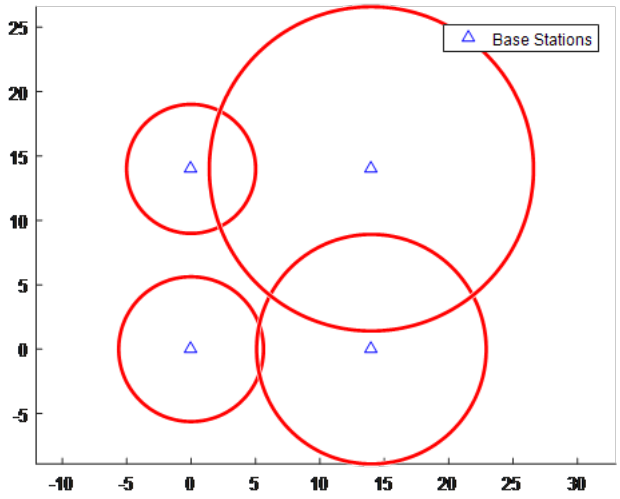


Figure 3.13: Circles with Radius d_i Centered at Each Base Station

Figure 3.14 shows the estimation result using w_{1i}. The error is 4.2273m.

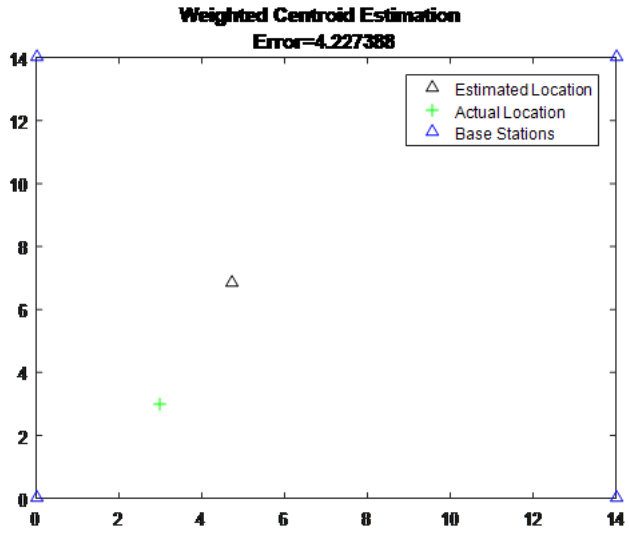


Figure 3.14: Weighted Centroid Estimation with Weight Factor $\frac{1}{d}$

Figure 3.15 shows the estimation result using w_{2i}. The error is 4.1485m. w_{2i} results in a slight better approximation comparing with w_{1i} and is adopted for this project.

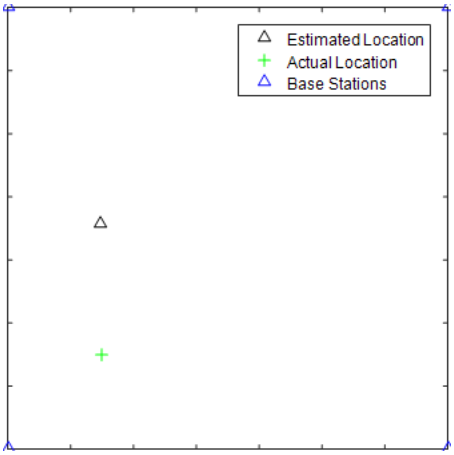


Figure 3.15: Weighted Centroid Estimation with Weight Factor $\frac{1}{d^2}$

3.2.3 Recursive Least Square Method

We also applied an iterative method, Recursive Least Square (RLS), to find (x, y) that is closest to all n base stations. This technique requires us to first define an error function f_i for each base station that describes the distance between the estimated AP (AP) location and circles centered at each base station, which is explained below. The total error which is the sum of error functions for all base stations is of form $E = \sum_{i=1}^{n} f_i^2$. To find minimizer (x, y), we need to solve $\nabla E = 0$. In addition to Gauss-Newton's method, we also applied Newton's method for comparison. Newton's method has a higher computational cost comparing with Gauss-Newton's method, but with similar estimation errors and the number of iterations for convergence. To increase the speed of the algorithm, we also tested different stopping criteria for the iterative algorithm: the difference between successive steps and the scaled difference between successive steps. For the same tolerance 10^{-4}, condition 2 the scaled difference takes less steps to converge, with identical estimation error as condition 1. Therefore we used Gauss-Newton's method with scaled difference error as stopping criteria for our RLS method.

We considered two options for f_i. The first error function f_{1i} is defined as the Euclidean distance from the AP to each circle,

$$f_{1i}(x, y) = |\sqrt{(x_i - x)^2 + (y_i - y)^2} - d_i|$$

where (x, y) is the estimated location of the AP, (x_i, y_i) are coordinates of the center of the ith circle and d_i is the radius of each circle calculated using path loss equation described in the previous section Scenarios and Channel Models. The total error function for n base stations is $\sum_{i=1}^{n} f_{1i}$. However, absolute value function is not differentiable and we can not applying standard technique of taking derivative to optimize. Thus we use an alternative error function that simplifies the calculation

$$E_1(x, y) = \sum_{i=1}^{n} f_{1i}^2 = \sum_{i=1}^{n} (\sqrt{(x_i - x)^2 + (y_i - y)^2} - d_i)^2. \tag{2}$$

The second error function f_{2i} is defined as the absolute value of difference between the square of the

distances, which is commonly used in localization

$$f_{2i}(x,y) = |(x_i - x)^2 + (y_i - y)^2 - d_i^2|.$$

The total error for n base stations is $\sum_{i=1}^{n} f_{2i}$ which suffers from similar indifferentiable problem due to the absolute value involved and is replaced with following

$$E_2(x,y) = \sum_{i=1}^{n} f_{2i}^2 = \sum_{i=1}^{n} ((x_i - x)^2 + (y_i - y)^2 - d_i^2)^2. \tag{3}$$

To minimize (2) and (3), we want to solve for $\nabla E_1 = \begin{bmatrix} \frac{\partial E_1}{\partial x} & \frac{\partial E_1}{\partial y} \end{bmatrix} = \vec{0}$ and $\nabla E_1 = \begin{bmatrix} \frac{\partial E_2}{\partial x} & \frac{\partial E_2}{\partial y} \end{bmatrix} = \vec{0}$. The partial derivatives of E_1 and E_2 are calculated as

$$\begin{aligned}
\frac{\partial E_1}{\partial x} &= \sum_{i=1}^{n} 2\left(((x_i - x)^2 + (y_i - y)^2)^{1/2} - d_i\right)((x_i - x)^2 + (y_i - y)^2)^{-1/2}(x - x_i) \\
&= \sum_{i=1}^{n} 2\left(1 - \frac{d_i}{\sqrt{(x_i - x)^2 + (y_i - y)^2}}\right)(x - x_i) \\
\frac{\partial E_1}{\partial y} &= \sum_{i=1}^{n} 2\left(((x_i - x)^2 + (y_i - y)^2)^{1/2} - d_i\right)((x_i - x)^2 + (y_i - y)^2)^{-1/2}(y - y_i) \\
&= \sum_{i=1}^{n} 2\left(1 - \frac{d_i}{\sqrt{(x_i - x)^2 + (y_i - y)^2}}\right)(y - y_i) \\
\frac{\partial E_2}{\partial x} &= \sum_{i=1}^{n} 4\left((x_i - x)^2 + (y_i - y)^2 - d_i^2\right)(x - x_i) \\
\frac{\partial E_2}{\partial y} &= \sum_{i=1}^{n} 4\left((x_i - x)^2 + (y_i - y)^2 - d_i^2\right)(y - y_i).
\end{aligned}$$

Both $\nabla E_1 = \vec{0}$ and $\nabla E_2 = \vec{0}$ are nonlinear systems of equations that can not be solved analytically. Thus we turn to an iterative approach to solve the system, such as Newton's method.

Newton's root-finding method starts with an approximation of a general function using Taylor's expansion. Taylor's first order expansion of a univariate function $f : \mathbb{R} \to \mathbb{R}$ at point $x^{[k]}$ is

$$f(x) \approx f(x^{[k]}) + f'(x^{[k]})(x - x^{[k]}) \tag{4}$$

Let $f(x) = 0$, we get

$$f(x^{[k]}) + f'(x^{[k]})(x - x^{[k]}) = 0 \implies x = x^{[k]} - \frac{f(x^{[k]})}{f'(x^{[k]})}.$$

Given a guess $x^{[k]}$, we hope to get a better solution to $f(x) = 0$ by solving the approximated linear equation (4) to get $x^{[k+1]}$ that solves $f(x^{[k+1]}) = 0$. Starting with an initial guess $x^{[0]}$, Newton's method gives successive iterates $x^{[1]}, x^{[2]} \ldots x^{[n]}$ which hopefully converges to a solution x for $f(x) = 0$:

$$x^{[k+1]} = x^{[k]} - \frac{f(x^{[k]})}{f'(x^{[k]})}. \tag{5}$$

The iteration is well defined if $f'(x^{[k]}) \neq 0$ at each step.

For $F : \mathbf{R}^n \to \mathbf{R}^m$, $F(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \dots f_m(\mathbf{x})]^T$, Taylor's first order expansion of f at the point $x^{[k]}$ is

$$F(x) \approx F(x^{[k]}) + \nabla F(x^{[k]})(x - x^{[k]}) \tag{6}$$

where the Jacobian matrix ∇F is

$$\nabla F = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}$$

In the same manner as before for $f(x) = 0$, we get Newton's method for solving a system of equations, which is well defined if the Jacobian matrix is invertible

$$x^{[k+1]} = x^{[k]} - (\nabla F(x^{[k]}))^{-1} F(x^{[k]}). \tag{7}$$

Since equation (7) requires Jacobian matrix to be invertible, $n = m$ i.e $F : \mathbf{R}^n \to \mathbf{R}^n$,

$$F(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) & f_2(\mathbf{x}) & \dots & f_n(\mathbf{x}) \end{bmatrix}^T$$

with $\vec{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}$.

For our work, we aim to minimize error function $E : \mathbf{R}^n \to \mathbf{R}$ defined in (2) and (3). As stated earlier, this requires to solve $\nabla E = \vec{0}$ using an iterative approach. So we use an iterative method and replace F with ∇E in (7):

$$x^{[k+1]} = x^{[k]} - (\nabla^2 E(x^{[k]}))^{-1} \nabla E(x^{[k]}) \tag{8}$$

which is well-defined if $\nabla^2 f$ is invertible. In particular,

$$E(\vec{x}) = \sum_{i=1}^{n} f_i(\vec{x})^2 = \vec{F}^T \vec{F}$$

where $f_i : \mathbf{R}^n \to \mathbf{R}, \vec{F} = \begin{bmatrix} f_1 & f_2 & \dots & f_n \end{bmatrix}^T$. The Jacobian matrix of E is calculated as

$$\nabla E = \left(\frac{\partial E}{\partial x_j} \right)_j = \left(2 \nabla (\vec{F})^T \vec{F} \right)$$

and the Hessian matrix of E is calculated as

$$\begin{aligned} \nabla^2 E = \left(\frac{\partial^2 E}{\partial x_j \partial x_i} \right)_{j,k} &= \left(\frac{\partial}{\partial x_j} \left(\sum_{i=1}^{m} f_i(\vec{x}) \frac{\partial f_i}{\partial x_k} \right) \right) \\ &= \sum_{i=1}^{m} \left(\frac{\partial f_i}{\partial x_j} \frac{\partial f_i}{\partial x_k} + f_i \frac{\partial^2 f_i}{\partial x_k \partial x_j} \right). \end{aligned}$$

When the value of f_i is small or f_i is close to linear, the second order derivative term is close to 0 and we get the following approximation for Hessian matrix:

$$\nabla^2 E = (\frac{\partial^2 E}{\partial x_j \partial x_i})_{j,k} \approx \sum_{i=1}^{m} \frac{\partial f_i}{\partial x_j} \frac{\partial f_i}{\partial x_k} = \nabla \vec{F}^T \nabla \vec{F}.$$

Substitute the approximation of Hessian matrix (3.2.3) and (3.2.3) into the Newton's method stated in (8) and we get a variation of Newton's method, Gauss-Newton's method:

$$x^{[k+1]} = x^{[k]} - (\nabla \vec{F}^T \nabla \vec{F})^{-1} (\nabla (\vec{F})^T \vec{F}).$$

To choose between Newton's method and Gauss-Newton's method, we compared the estimation error as well as the number of iterations required to converge for both methods. Estimation error is defined as the Euclidean distance between the estimated AP location and the actual AP location. The estimation error of applying both algorithms is identical, as can be seen from Table 1. Though the number of iterations until convergence is similar for both techniques, Newton's method has a slightly higher computational cost comparing to Gauss-Newton's method, which does not require the calculation of Hessian matrices. In the rest of the discussion, we consider only Gauss-Newton's algorithm.

The base stations are at $(0,0), (0,14), (14,0), (14,14)$. The actual location of AP is $(3,3)$. The initial estimation of the AP location is the output of the Weighted Centroid algorithm, $(2.929, 4.3)$. In the following two tables, we recorded 10 sets of experiments with RSS readings at each base station randomly generated between -70dBm and -50dBm, as shown in Table 2. In Table 1, columns 2 to 5 contain the distance d_i between the estimated AP location and each base station calculated using RSS in Table 1 and path-loss model described in section Scenarios and Channel Models. Columns 6 to 9 in Table 1 are the estimation error using Gauss-Newton's method, the number of iterations to converge using Gauss-Newton's method, estimation error using Newton's method, and the number of iterations to converge using Newton's method.

Table 1: Distance Between Estimated Location to Each Base Station and Iteration Numbers and Error

Distance	(0,0)	(0,14)	(14,0)	(14,14)	error_gn	iter_gn	error_n	iter_n
set 1	10.42	4.74	8.05	6.37	6.91	11	6.91	11
set 2	23.95	3.29	30.03	6.31	21.56	16	21.56	10
set 3	8.14	5.89	11.88	7.27	6.27	9	6.27	9
set 4	18.79	14.33	15.39	29.27	11.40	9	11.40	11
set 5	13.04	8.24	21.80	26.99	12.29	6	12.29	10
set 6	8.26	24.67	20.95	15.15	10.48	18	10.48	9
set 7	17.71	3.92	11.94	9.31	11.78	6	11.78	6
set 8	16.20	4.17	25.45	7.01	15.91	11	15.91	65
set 9	7.64	4.81	7.97	12.37	3.87	8	3.87	9
set 10	17.17	17.35	10.69	4.79	13.93	7	13.93	7

Table 2: Random RSS Readings Within (-70, -50) at Each Base Station

RSS Reading	(0,0)	(0,14)	(14,0)	(14,14)
set 1	-60.36	-53.51	-58.11	-56.08
set 2	-67.59	-50.35	-69.55	-56.00
set 3	-58.21	-55.40	-61.49	-57.23
set 4	-65.48	-63.12	-63.75	-69.33
set 5	-62.31	-58.32	-66.77	-68.62
set 6	-58.34	-67.84	-66.42	-63.61
set 7	-64.96	-51.87	-61.54	-59.38
set 8	-64.19	-52.41	-68.12	-56.91
set 9	-57.66	-53.64	-58.03	-61.85
set 10	-64.69	-64.79	-60.58	-53.60

For the stopping criteria, we considered two alternative conditions for terminating the iterative algorithm. The tolerance is 10^{-4}. Condition 1 is $|x^n - x^{n+1}| < \text{tol}$ and condition 2 is $|\frac{x^n - x^{n+1}}{x^{n+1}}| < \text{tol}$. We tested the two stopping conditions on 10 sets of randomly generated RSS readings and compared the estimation error as well as the number of iterations until convergence for both conditions, using Gauss-Newton's method, for each set. The results are presented in the following Table 3. Column 1 and 2 contain the number of iterations until convergence for condition 1 and condition 2 respectively. As can be seen, condition 2 requires less iterations comparing to condition 1. Therefore we used condition 2 as stopping criteria for RLS in our project.

Table 3: The Number of Iterations Required for Two Stopping Conditions

cond1 iter	cond2 iter
31	25
20	17
45	37
23	19
21	17
31	26
26	22
17	14
26	22
21	18

3.3 Calculation of Cramer-Rao Lower Bound

The general form of an observation function is $y = \theta + \eta$, where η is Gaussian noise with zero mean and variance σ^2. The observation function for our project is:

$$P_r = P_0 - 10\alpha \log d + X$$

So we have

$$\theta(d) = P_0 - 10\alpha \log d$$

and Gaussian noise η corresponds with shadow fading $X \sim \mathcal{N}(0, \sigma^2)$. The distribution function of an observation function y given θ is

$$p(y;\theta) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{[y-\theta(d)]^2}{2\sigma^2}}$$

From [4], Fisher information matrix is defined as

$$F = E\left[\frac{\partial \ln p(y;\theta)}{\partial \theta}\right]^2 = -E\left[\frac{\partial^2 \ln p(y;\theta)}{\partial \theta^2}\right].$$

After calculation, we get

$$F = \frac{(\theta'(d))^2}{\sigma^2}$$

and

$$\text{CRLB} = F^{-1} = \frac{(\ln 10)^2 \sigma^2}{100\alpha^2} d^2$$

The square root of the CRLB is the ranging error σ_p [5]. For example, in one of our path-loss models, distance power gradient $\alpha = 2.1$ from the best fit line, and $\sigma = 2.5$ calculated from the measurement result one location. The ranging error in this case is $\sigma_p \geq \frac{\ln 10\sigma}{10\alpha} d = 0.184d$. So our estimation result using RSS for this model at the location has an uncertainty of 0.184 times the distance.

For multiple observations, $y_i = \theta_i(x, y) + \eta_i, i = 1 \ldots K$ [5], where (x, y) is the coordinate of a given point. For this project,

$$y_i = P(r_i) = P_0 - 10\alpha_i \log d_i + X \quad i = 1, \ldots, K$$

$$\theta_i(x, y) = P_0 - 10\alpha_i \log d_i$$

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

We can express them in a vector form:

$$G(x, y) = [\theta_1(x, y) \quad \theta_2(x, y) \quad \ldots \quad \theta_N(x, y)]^T$$

$$N = [\eta_1 \quad \eta_2 \quad \ldots \quad \eta_K]^T$$

Then $y = G(x, y) + N$. $\Sigma = E[\eta^T \eta]$ is a diagonal covariance matrix for variance of the noise σ_i, the distribution function of the observation is

$$p(\theta; x, y) = \frac{1}{\sqrt{(2\pi)^K |\Sigma|}} e^{-\frac{1}{2}|y-G(x,y)|^T \Sigma^{-1} |y-G(x,y)|}$$

Define $H = \nabla_x G(x, y)$. Fisher information matrix becomes [5]

$$F = E[\{\nabla_{x,y}[y - G(x, y)]^T\}\Sigma^{-1}\{\nabla_{x,y}[y - G(x, y)]\}] = H^T \Sigma^{-1} H$$

where

$$H = \nabla_{x,y}[y - G(x, y)] = -\nabla x, y[G(x, y)] = -\frac{10}{\ln 10}[\alpha_1 \alpha_2 \ldots \alpha_K] I_N \begin{bmatrix} \frac{x-x_1}{r_1^2} & \frac{y-y_1}{d_1^2} \\ \frac{x-x_2}{r_2^2} & \frac{y-y_2}{d_2^2} \\ \cdots & \cdots \\ \frac{x-x_K}{r_K^2} & \frac{y-y_K}{d_K^2} \end{bmatrix}$$

The CRLB is then given by

$$\text{CRLB} = \text{Trace}||F^{-1}|| = \text{Trace}\{H^T\Sigma^{-1}H\}.$$

If we assume zero-mean white Gaussian noise,

$$\Sigma = \begin{cases} \sigma_p^2, i = j \\ 0, i \neq j \end{cases} \quad i, j = 1, 2, \ldots, K$$

CRLB is calculated as

$$\text{CRLB} = \text{Trace}[\sigma_p^2(H^T H)^{-1}] = \text{Trace}\begin{bmatrix} \sigma_x^2 & \sigma_{xy}^2 \\ \sigma_{xy}^2 & \sigma_y^2 \end{bmatrix} \implies \sigma_p = \sqrt{\sigma_x^2 + \sigma_y^2}.$$

In the following example, we have 8 base stations, $K = 8$. The relationship between location of points and ranging error σ_p is shown by plotting the contour of (x, y, CRLB).

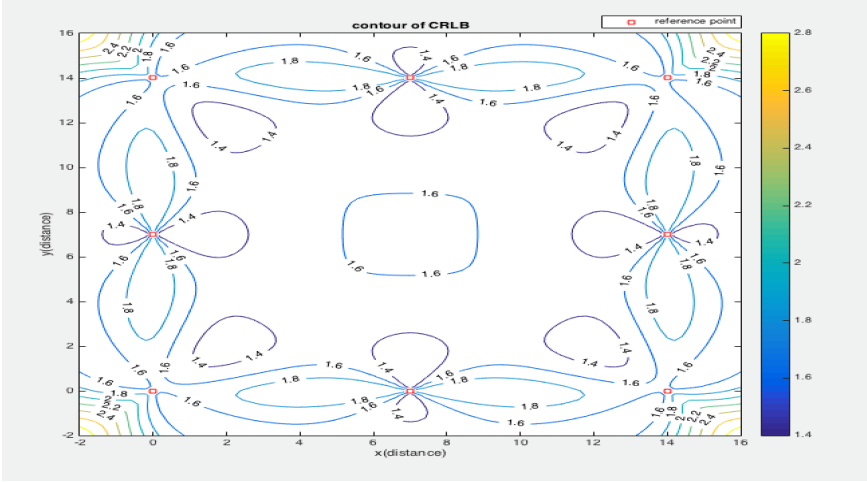


Figure 3.16: Cramer-Rao Lower Bound of 8 Base Stations

Each base station in Figure 3.16 is marked as a red dot. We can see the stability of estimated result; larger variance leads to lower stability. If AP is close to any base station, the contour plot is denser in that region. The MATLAB implementation for computing CRLB in 2D is attached in Appendix H.

The CRLB for 3D localization when AP is 12m above the ground is shown below in Figure 3.17. Base stations are indicated by red rectangles in the figure.The MATLAB implementation for computing CRLB in 2D is attached in Appendix I.

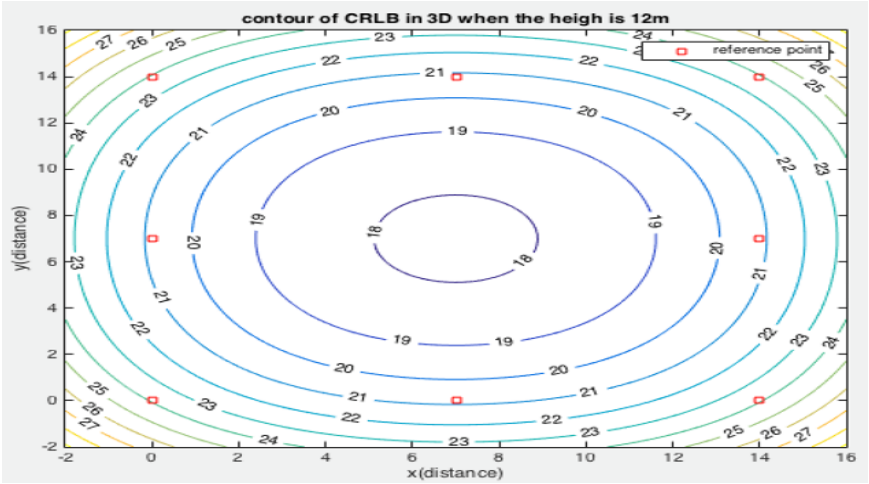


Figure 3.17: Cramer-Rao Lower Bound in 3D, H=12

4 Results and Discussions

In this chapter, we cross examine the performance of three algorithms under different assumptions. Weighted centroid is relative robust in the presence of significant measurement error when the ratios among distances are preserved. In 3D measurement, MLE has better performance than RLS.

4.1 Presence of Significant Measurement Error

As shown in the following figure, the measurement error can be quite significant due to multi-path interference. AP is located at (3,3) and the location of base stations are (0,0), (0,14), and (14,0). The RSS reading at each base station is -73.3dBm, -78.52dBm and -78.88dBm respectively. Upon converting these measurements into distances, we get the following figure. The radius of each circle is much larger than expected. When the ratio among the actual distances is relatively the same as the ratio among the RSS readings, despite the variance caused by noise, the weighted centroid algorithm is a very robust method to estimate the location of AP in 2D.

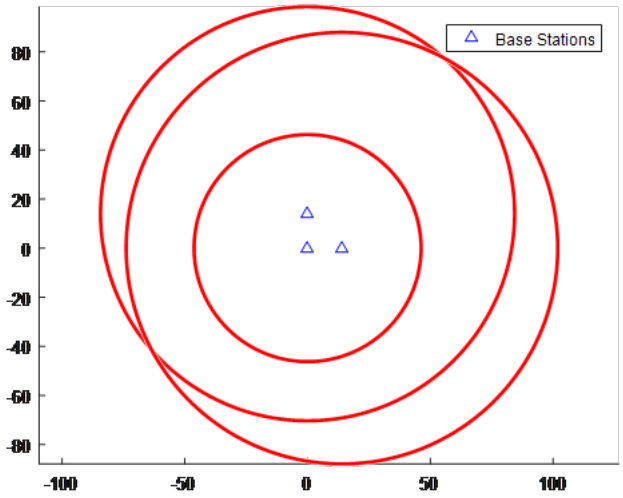


Figure 4.1: Significant Measurement Error

Weighted centroid offers a good estimation in this case, as shown in the following graph. The estimation error is 0.89m.

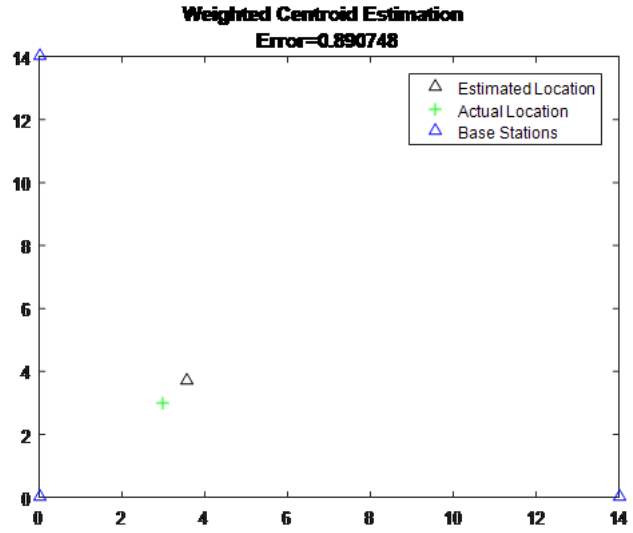


Figure 4.2: Significant Measurement Error - Weighted Centroid Method

Other two methods both failed in this case. As shown below, the estimation error using RLS is 73m. The MLE has similar poor performance.

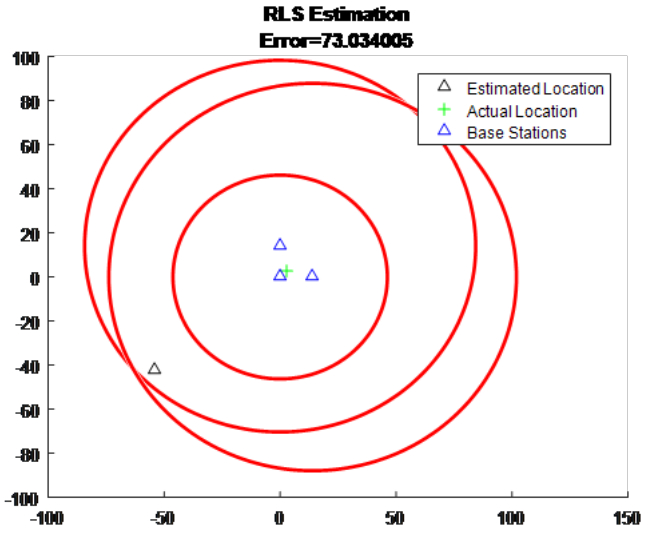


Figure 4.3: Significant Measurement Error - Recursive Least Square

4.2 Performance Comparison in 3D

Since all base stations are placed on the ground, weighted centroid method does not generalize to 3D with the current measurement data. In 3D, MLE has lower estimation error than RLS in 3D, as can be seen in the following table.

Method	No.Rx	Error(m)	ErrorX(m)	ErrorY(m)	ErrorZ(m)
MLE	4	6.8573	4.1729	4.3572	3.2594
	6	5.2	3.2339	3.3185	2.3818
	8	4.9312	3.0885	3.1011	2.2716
RLS	4	8.1162	4.6312	5.1023	4.2886
	6	7.3471	4.2809	4.2536	4.1907
	8	6.3955	3.7510	4.0034	3.2872

Table 4: Maximum Likelihood Estimation vs Recursive Least Square: estimation error

This observation is also supported by computing CRLB of the two algorithms in 3D. MLE shown in green in figure below has lower variance comparing to RLS in blue and is closer to CRLB, as shown in red.

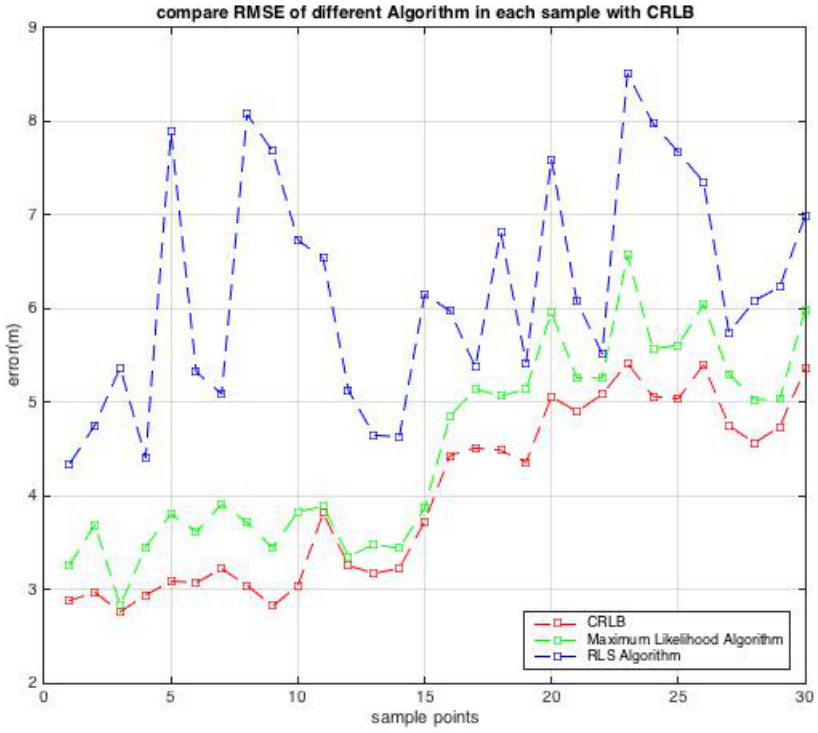


Figure 4.4: Evaluate MLE and RLS using CRLB

5 Conclusions and Future Work

In this project, we designed and implemented three localization algorithms based on RSS readings from different base stations. Our results showed that 3D RSS-based drone localization is achievable with reasonable error under certain condition. In 2D, weighted centroid is relatively robust to the fluctuation of measurement errors. In 3D, MLE has lower estimation error and is closer to CRLB comparing with RLS.

For future projects, here are some suggestions that can be improved:

- Intercept the actual signal transmitted from the drone instead of using a phone.
- WiFi password of the drone is currently assumed to be known, which is not quite realistic.
- We used laptops as base stations. Can explore the possibility of using Raspberry Pi, Arduino, or other embedded system.
- Improve the algorithm of MLE in 3D to reduce run time.
- Weighted centroid method is limited to 2D due to the location of base stations. Can experiment with different positions of base stations and to include information of height.
- Localization techniques that do not based on RSS can also be explored: angle of arrival, time of arrival, and etc.
- Apply machine learning to train the data and get better prediction.

A Federal Regulations of Unmanned Aircraft Systems

A drone as a type of Unmanned Aircraft Systems(UAS) is subject to federal regulations. At the beginning of our project, we went through the rules and followed them closely. Below is a summary of the requirements for a non-commercial light-weight UAS, according to the Federal Aviation Association at the time of writing [1].

- Register your UAS if it weighs more than 0.55 pounds and less than 55 pounds
- Label your UAS with your registration number
- Fly at or below 400 feet
- Keep your UAS within sight
- Never fly near other aircraft, especially near airports
- Never fly over groups of people
- Never fly over stadiums or sports events
- Never fly near emergency response efforts such as fires
- Never fly under the influence
- Be aware of airspace requirements

B Data Collection

To measure the received signal strength (RSS) of a Wi-Fi signal, we used software WirelessMon, which is available for Windows and Android devices.

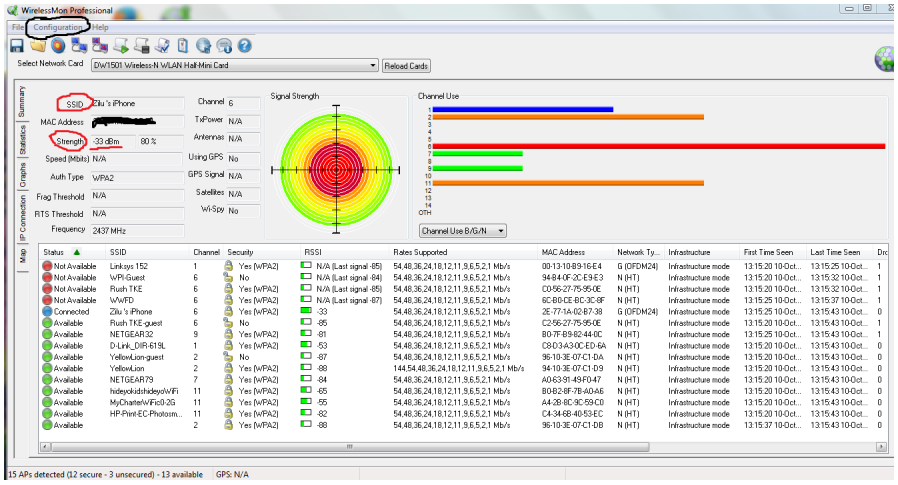


Figure B.1: A screen copy of WirelessMon in action

Image B.1 is the user interface of WirelessMon. **Service Set Identifier** (SSID) shown in the image is a case sensitive, unique identifier attached to the header of packets sent over a wireless local-area network, which can be considered as the name for an access point. **Signal** shows the received signal strength in unit dbm, which is the power ratio in decibels of the received power per one milliWatt. The measurement of **Signal** is updated periodically, and the time interval can be specified in **Configurations/General**. For our project, we need to log continuous readings and compute the average and variance of the signal strength measurements. This can be achieved in **Configurations/Logging Options** which specifies output path for **Connected Node Log** that contains received signal strength at different time instances at the same location.

Constructing a path-loss model requires 8 to 15 measurements, which will result in 8 to 15 files each time. To automate the process of calculating the average and variance from each file repeatedly, we wrote a Python script to loop through all the text files in a designated data folder with a naming convention xxyy.txt, where xx is the access point coordinate and yy is the measurement location. Please refer to Appendix C for data parsing script.

C Text Parsing in Python

```
# A script to parse data input from WirelessMon
# Sample input (0044.txt):
#
# Index, Time, SSID, MAC, Channel, Percentage(%), Strength(dBm)
# 0, 16:44:58 7-Oct-2016, "Lee "de iphone (3), 22-3c-AE-48-B6-C0,6,63,-54
#
# The length of the file is undetermined.
# Want to keep the last element of the line, and calculate the average and std


# the length of the file
# num_lines = sum(1 for line in f)

def meanList(numbers):
    return float(sum(numbers)/len(numbers))


def stdList(numbers, avg):
    sumSquare = sum((x - avg)**2 for x in numbers)
    stdev = (sumSquare/len(numbers))**0.5
    return stdev

import os
# directory that contains all the data file
dataFiles = os.listdir('C:/Users/Zilu/Desktop/tmpData');
# a summary file that all the results can write to
#summaryFile = open('C:\Users\Zilu\Google Drive\MQPLocalization\Test1007\1007Zilu\summary.txt');

for fileNames in dataFiles:
    # extract only data files that end with .txt
    if fileNames.endswith('.txt'):
        fname = fileNames

    # received signal strength reading
    rss = [];

    with open(fname,'r') as input_file:
        # skip the header of the file
        lines = input_file.readlines()[2:]
        for line in lines:
            # text parsing by comma
            line = line.split(',')
            # check the mac address to make sure the measurement is LEE IPHONE
            if line[-4]=='22-3C-AE-48-B6-C0':
                rss.append(line[-1][:-1])
            else:
                print('ERROR: Wrong MAC Address!')
```

```
    # convert list of strings to int
    rss = map(int, rss)
    # compute the mean of the list
    meanRSS = meanList(rss)

    AP_center = [int(fname[-6:-5]),int(fname[-5:-4])]
    obs_point = fname[0:-6]

    print ('AP', AP_center,'at',obs_point,'Average', meanRSS, 'Std', stdList(rss, meanRSS))
```

D MATLAB Implementation of 2D MLE

```
function [pointInRing] = ring(center, inner_radius, outer_radius)
    xmin = center(1)-outer_radius;
    xmax = center(1)+outer_radius;
    ymin = center(2)-outer_radius;
    ymax = center(2)+outer_radius;

    % create a grid of points
    X = [xmin:0.1:xmax];
    Y = [ymin:0.1:ymax];

    pointInRing = [];

    for i=1:length(X)
        x = X(i);
        for j = 1:length(Y)
            y = Y(j);
            if ((x-center(1))^2 + (y-center(2))^2 ≤ (outer_radius^2)) && ((x-center(1))^2 + ...
                (y-center(2))^2 ≥ (inner_radius^2))
                pointInRing = [pointInRing; [x,y]];
            end
        end
    end

    return
end

function [est_loc, error] = Maximum_Likelihood_2D(refLoc, apLoc, rss, rssSig, alpha, c)

    xval = refLoc(:,1);
    yval = refLoc(:,2);

    nodesNum = length(rss);
    sigRange = 1;

    rssUpper = rss + sigRange*rssSig;
    rssLower = rss - sigRange*rssSig;

    du = zeros(nodesNum,1);
    dl = zeros(nodesNum,1);
    for i=1:length(rss)
        du(i) = 10^((1/alpha)*(rssUpper(i) - c));
        dl(i) = 10^((1/alpha)*(rssLower(i) - c));
    end

    figure;
    for i = 1:length(rss)
        viscircles(refLoc(i,:),du(i));
```

```
        hold on;
        viscircles(refLoc(i,:),dl(i));
    end

    % interior regions, exterior regions, ring regions
    Rings = cell(length(xval));
    for i = 1:length(rss)
        Rings{i} = round(ring([xval(i),yval(i)],du(i),dl(i))*10)/10;
    end
    C1 = intersect(Rings{1}, Rings{2},'rows');
    C2 = intersect(C1, Rings{3},'rows');
    C3 = intersect(C2, Rings{4},'rows');

    edx = sum(C3(:,1))/length(C3);
    edy = sum(C3(:,2))/length(C3);

    error = ((edx-apLoc(1))^2+(edy-apLoc(2))^2)^.2;

    h1 = plot(edx, edy, 'k^');
    hold on;
    h2 = plot(apLoc(1), apLoc(2),'g+');
    hold on;
    h3 = plot(refLoc(:,1),refLoc(:,2),'^b');
    titleStr = sprintf('ML Estimation\nError=%2f m', error);
    title(titleStr);
    legend([h1, h2, h3], {'Estimated Location', 'Actual Location', 'Base Stations'});
    axis('square');
    est_loc = [edx, edy];
end
```

E MATLAB Implementation of 2D Weighted Centroid

```
function [est_loc, error] = Weighted_Centroid_2D(refLoc, apLoc, rss, alpha, c)
    rss = rss';
    % path loss model used for calculation
    d=10.^((rss-c)/alpha);

    for j=1:1:size(rss,2)
        % estimate the location based on different weighting factors
        for i=1:1:size(rss,1)
            w(i)=(1/d(i,j).^2)/sum(1./d(:,j).^2);
        end
        x=w*refLoc(:,1);
        y=w*refLoc(:,2);
        edx(j)=x;
        edy(j)=y;
    end
    for y=1:1:size(rss,2)
        error2(y)=(sum(([edx(y),edy(y)]-apLoc(y,:)).^2))^0.5;
    end

    figure;

    h1 = plot(edx, edy, 'k^');
    hold on;
    h2 = plot(apLoc(1), apLoc(2),'g+');
    hold on;
    h3 = plot(refLoc(:,1),refLoc(:,2),'^b');
    titleStr = sprintf('Weighted Centroid Estimation \nError=%2f', error2);
    title(titleStr);
    legend([h1, h2, h3], {'Estimated Location', 'Actual Location', 'Base Stations'});
    axis('square');

    error = error2;
    est_loc = [edx, edy];
end
```

F MATLAB Implementation of 2D RLS

```
function [est_loc, error] = RLS_2D(refLoc, apLoc, rss, alpha, c, init_loc)

    distance = 10.^((rss-c)/alpha);

    figure;
    for i = 1: length(rss)
        viscircles(refLoc(i,:), distance(i));
        hold on;
    end

    % calculate the RLS
    base_station_num = size(refLoc,1);
    temp_location = init_loc ;
    estimated_error = norm(sum((refLoc - (ones(base_station_num,1)*temp_location)).^2,2)-...
    (distance.^2)',1).^2;
%     estimated_error =  (sum((refLoc - ...
    (ones(base_station_num,1)*temp_location)).^2,2).^0.5-(distance)').^2;

    while norm(estimated_error) > 1e-4 %iterative process
        jacobian_matrix =  -2*(refLoc - ones(base_station_num,1)*temp_location);
        f = sum((refLoc - (ones(base_station_num,1)*temp_location)).^2,2)-(distance.^2)';
%         jacobian_matrix = sum((refLoc - ...
    (ones(base_station_num,1)*temp_location)).^2,2).^(-0.5).*(...
ones(base_station_num,1)*temp_location(1)-refLoc(:,1));
%         f = sum((refLoc - (ones(base_statiThe distance between the estimated location and ...
    the actual position of access point is 5.096m, which corresponds to a relative error of ...
    25.74\%.
on_num,1)*temp_location)).^2,2).^0.5-(distance)';
        estimated_error = -inv(jacobian_matrix' * jacobian_matrix) *(jacobian_matrix') * f ;
        temp_location = temp_location + estimated_error' ;
    end

    edx = temp_location(1);
    edy = temp_location(2);

    error = ((edx-apLoc(1))^2+(edy-apLoc(2))^2)^0.5;

    h1 = plot(edx, edy, 'k^');
    hold on;
    h2 = plot(apLoc(1), apLoc(2),'g+');
    hold on;
    h3 = plot(refLoc(:,1),refLoc(:,2),'^b');
    titleStr = sprintf('RLS Estimation\nError=%2f m', error);
    title(titleStr);
    legend([h1, h2, h3], {'Estimated Location', 'Actual Location', 'Base Stations'});
    axis('equal');

    figure;
```

```
    for i = 1: length(rss)
        viscircles(refLoc(i,:), distance(i));
        hold on;
    end
    hold on;
    h4 = plot(refLoc(:,1),refLoc(:,2),'^b');
    legend([h4],'Base Stations');
    axis('equal');

    est_loc = [edx, edy];
end
```

G MATLAB Implementation of 3D RLS

```
% helper function
function [final_x, final_y,final_z,estimated_error,h3] = ...
    RLS(known_references,initial_guess,distances)
    if size(known_references,2) ~= 3
        error('location of known reference points should be entered as Nx3 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+(known_references(j,3) - ...
            temp_location(i,3))^2 -distances(j)^2) ;
    end

    estimated_error = temp_error ;
    % while norm(estimated_error) > 2*(1e-2)%iterative process for LS algorithm

    for j=1:1:1000
        for j = 1 : size(known_references,1) %Jacobian has been calculated inadvance
            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+(known_references(j,3) - 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' ;
        i = i + 1;
    end

    final_x = temp_location(i,1) ;
    final_y = temp_location(i,2) ;
    final_z = temp_location(i,3) ;

    h3=plot3(final_x,final_y,final_z,'rx');% plot

    % text(final_x,final_y,'  estimated point using RLS-RSS')
    title('coordinate')
    xlabel('x(m)');
    ylabel('y(m)');
end
```

```
reference1=[42.274768,-71.811306];
reference3=[42.274642,-71.811421];
reference2=[42.274835,-71.811453];
point1=[42.274748,-71.811417];
point2=[42.274715,-71.811361];
point3=[42.274807,-71.811378];
point4=[42.274733,-71.811551];

[reference3codx,reference3cody]=coordinateTrans(reference3(1),...
reference3(2),reference1(1),reference1(2));
[reference2codx,reference2cody]=coordinateTrans(reference2(1),...
reference2(2),reference1(1),reference1(2));
[point1codx,point1cody]=coordinateTrans(point1(1),point1(2),...
reference1(1),reference1(2));
[point2codx,point2cody]=coordinateTrans(point2(1),point2(2),...
reference1(1),reference1(2));
[point3codx,point3cody]=coordinateTrans(point3(1),point3(2),...
reference1(1),reference1(2));
[point4codx,point4cody]=coordinateTrans(point4(1),point4(2),...
reference1(1),reference1(2));

figure(1)
x=[0,reference3codx,reference2codx];
y=[0,reference3cody,reference2cody];
z=[0,0,0];
b=scatter3(x,y,z,'ok');
px=[point1codx,point2codx,point3codx,point4codx];
py=[point1cody,point2cody,point3cody,point4cody];
pz=[4.5,4.5,4.5,4.5];
hold on
d=scatter3(px,py,pz,'rd');
legend([b,d],'BS','actual point of drone')
xlabel('x(m)')
ylabel('y(m)')
zlabel('z(m)')
title('comparison of NE coordiante transformed by GPS and actual coordinate we set ')
text(x(1)+0.5,y(1),0,'r1')
text(x(2)+0.5,y(2),0,'r3')
text(x(3)+0.5,y(3),0,'r2')
text(px(1)+0.5,py(1),pz(1),'p1')
text(px(2)+0.5,py(2),pz(2),'p2')
text(px(3)+0.5,py(3),pz(3),'p3')
text(px(4)+0.5,py(4),pz(4),'p4')

rss=[-58.17,-58.13,-60;-55.84,-55.08,-62.82;-57.3,-62.88,-54.96;
-64.82,-62.97,-61.44];
for i=1:1:3
   references(i,:)=[x(i),y(i),z(i)];
 end
[xx, yy, zz]=sphere(50);
```

```
j=4;
r1=10.^((rss(j,1)+38)/(-20));
r2=10.^((rss(j,2)+38)/(-20));
r3=10.^((rss(j,3)+38)/(-20));
figure(2)
mesh(r1*xx+x(1),r1*yy+y(1),r1*zz);
hold on
mesh(r2*xx+x(2),r2*yy+y(2),r2*zz);
mesh(r3*xx+x(3),r3*yy+y(3),r3*zz);
axis equal
alpha(0.3)

distances=[r1,r2,r3];
initial_guess=[px(j), py(j) ,pz(j)];
[final_x, final_y,final_z,estimatederror,h3] = RLS(references,initial_guess,distances);
%
error(j)=((final_x-px(j))^2+(final_y-py(j))^2+(final_z-pz(j))^2)^0.5;
```

H MATLAB Implementation of 2D CRLB

```
x = 0:0.2:16;
y = 0:0.2:16;
AP=[0,0;14,0;0,14];
e=2;
[X,Y] = meshgrid(x,y);
for j=1:1:81
    for k=1:1:81
 for i=1:1:3
     r(i)=sqrt((X(j,k)-AP(i,1)).^2+(Y(j,k)-AP(i,2)).^2);
     P(i)=-21/log(10)*((X(j,k)-AP(i,1))/(r(i).^2));
     q(i)=-21/log(10)*((Y(j,k)-AP(i,2))/(r(i).^2));
     H(i,1)=P(i);
     H(i,2)=q(i);

 end
 C=e^2*inv(H'*H);
 Z(j,k)=sqrt(C(1,1)+C(2,2));
end

figure
contour(X,Y,Z,'ShowText','on')
ylabel('y')
xlabel('x')
title('CRLB')
```

I MATLAB Implementation of 3D CRLB

```
x = -2:0.2:16;
y = -2:0.2:16;
AP=[0,0;7,0;14,0;0,7;14,7;0,14;7,14;14,14];
e=8;
[X,Y] = meshgrid(x,y);
for j=1:1:91
    for k=1:1:91
 for i=1:1:8
     r(i)=sqrt((X(j,k)-AP(i,1)).^2+(Y(j,k)-AP(i,2)).^2+36);
     P(i)=-20/log(10)*((X(j,k)-AP(i,1))/(r(i).^2));
     q(i)=-20/log(10)*((Y(j,k)-AP(i,2))/(r(i).^2));
     z(i)=-20/log(10)*(36)/(r(i).^2);
     H(i,1)=P(i);
     H(i,2)=q(i);
     H(i,3)=z(i);

 end
 C=e^2*inv(H'*H);
 Z(j,k)=sqrt(C(1,1)+C(2,2)+C(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=[0,0;7,0;14,0;0,7;14,7;0,14;7,14;14,14];
referencex=known_references(:,1);
referencey=known_references(:,2);
h1=plot(referencex,referencey,'rs');
legend(h1,'reference point')
```

J GPS Conversion to NE coordinate

The location of the drone at different time stamps can be retrieved from flight record. The following algorithm converts the GPS readings to NE coordinate. The idea is similar to the conversion of a point from spherical coordinate to Cartesian coordinate, as illustrated in following code snippet. The return values deltx and delty are length in x and y directions respectively.

```
% lon: longitude, lat: latitude, both in radians; R is the radius of the earth: 6371000m
R = 6371000;
deltx=R*delt(lon)
detly=R*delt(lat)
```

The theoretical error of GPS measurement is about 10m. We decided to test for GPS errors in our experiment. We started by defining the following experimental setup in Google Map, with reference points and drone's locations marked in yellow pins.

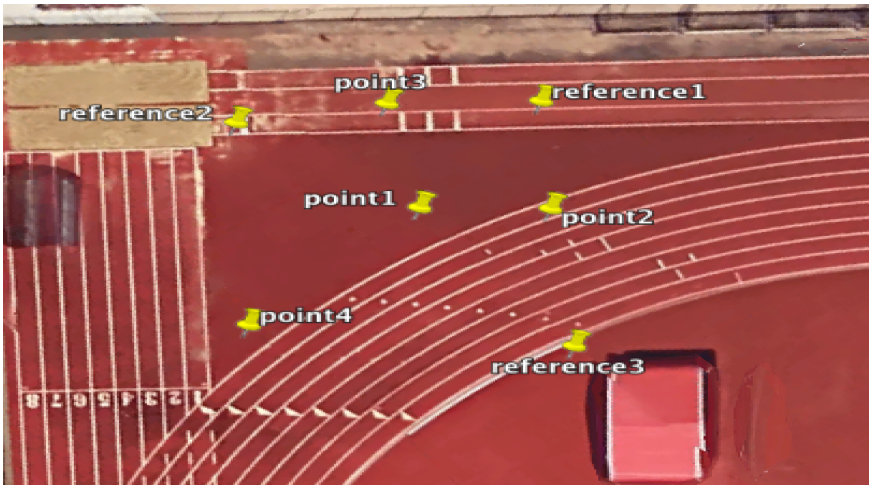


Figure J.1: Outdoor Scenario for GPS Measurement

We then measured the distance between the two reference points in Google Map and compared it with the computed result from GPS to NE algorithm. The distance between reference1 and reference3 measured in Google Map is 16.58m, and the calculated result of our algorithm is 16.91m, which is fairly accurate. We repeated the calculation for different reference points and all results are within reasonable accuracy.

K Motion Detection

To abstract information about motion from the received signals, we applied Fourier Transform to convert signals from time domain to frequency domain. In MATLAB, this can be done using command fft(). Following are figures of the FFT of signals with motions. Oscillations and noise make it hard to reach any conclusion about motion detection.

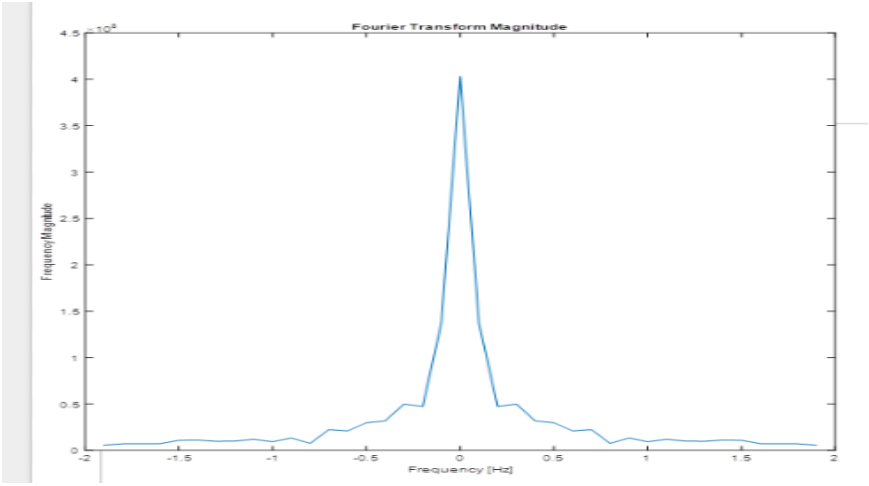


Figure K.1: FFT of RSS with upwards motion

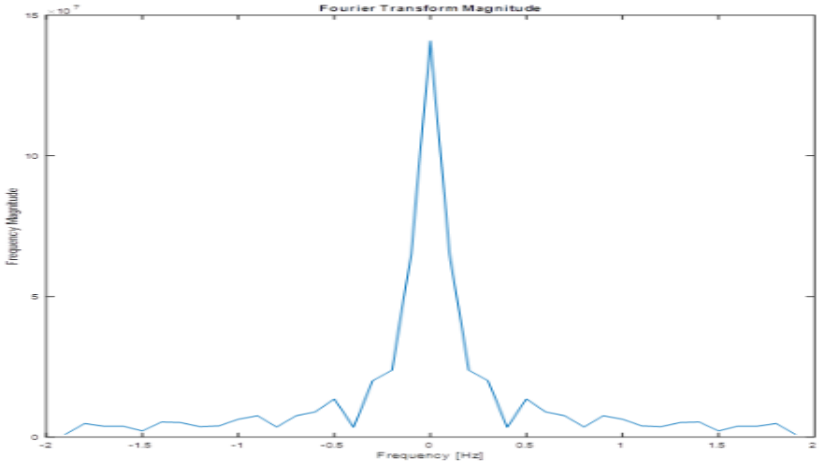


Figure K.2: FFT of RSS with downwards motion

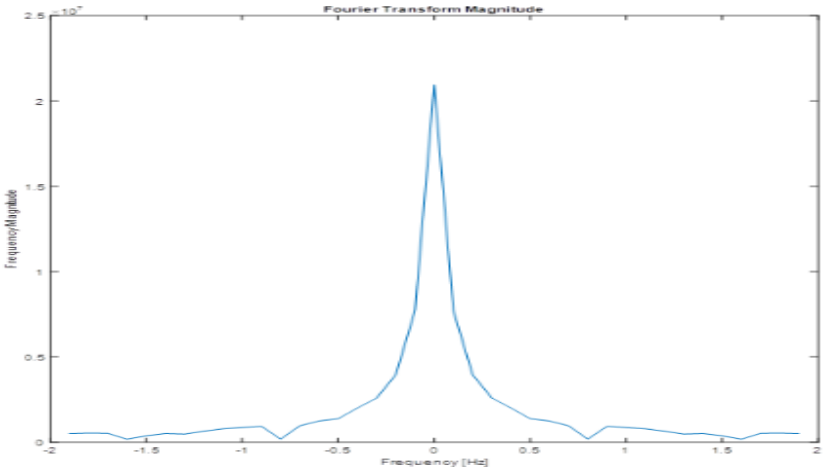


Figure K.3: FFT of RSS with no motion

L MATLAB Implementation of Fourier Transform of Received Signals

```
% fourier transform of the signals for motion detection
close all;
clear all;
clc;

% rss readings at 7,7 when AP is motionless
rss = [-52  -52 -52 -52 -52 -52 -52 -52 -52 -52 -52 -52 -52 -57 -57 -57 -57 -57 -57 -57 ...
    -57 -57 -57 -57 -57 -57 -57 -57 -57 -57 -57 -57 -57 -57 -57 -57 -57 -57 -57 -60 -60 -60 ...
    -60 -60 -60 -60 -60 -60 -60 -60 -60 -59 -59 -59 -59 -59 -59 -59 -59 -59 -59 -59 -59 -59 ...
    -58 -58 -58 -58 -58 -58 -58 -58 -58 -58 -58 -54 -54 -54 -54 -54 -54 -54 -54 -54 -54 -54 ...
    -54 -54 -54 -54 -56 -56 -56 -56 -56 -56 -56 -56 -56 -56 -56 -56 -56 -56 -56 -56 -56 -56 ...
    -56 -56 -56 -56 -56 -56 -56 -56 -56 -56 -49 -49 -49 -49 -49 -49 -49 -49 -49 -49 -49 -49];
sigStr = 10.^(-.1*rss);

rss_cpy = rss';

time_interval = 0.1;

len = 20;

f_vec = (-len+1:len-1)*time_interval;

f_rss = fft(sigStr);
f_rss = f_rss(1:len);
% flip all the points to the right of the origin
f_rss_reverse = fliplr(f_rss(1,2:end));

f_rss = [f_rss_reverse,f_rss];
% rss_filtered = firpm(17, f_rss, );
f_rss_m = abs(f_rss);
f_rss_a = unwrap(angle(f_rss));

figure;
plot(f_vec, f_rss_m,'-');
xlabel('Frequency [Hz]');
ylabel('Frequency Magnitude');
title('Fourier Transform Magnitude');
grid on;
```

References

[1] Federal Aviation Administration. Unmanned aircraft systems. `https://www.faa.gov/uas/`. Accessed: Sept. 1, 2016.

[2] Nick Bilton. When your neighbor's drone pays an unwelcomed visit. `https://www.nytimes.com/2016/01/28/style/neighbors-drones-invade-privacy.html?_r=0`, 2016. Accessed: Feb. 26, 2017.

[3] Alissa M. Dolan and Richard M. Thompson II. Integration of drones into domestic airspace: Selected legal issues. *Congressional Research Service*, April 2013.

[4] Steven M Kay. *Fundamentals of Statistical Signal Processing, Volume 1: Estimation Theory.* Prentice Hall Signal Processeing Series, 1993.

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