THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF ELECTRICAL ENGINEERING

SIMULATING AD HOC NOISE RADAR NETWORKS FOR TARGET LOCATION DETECTION

DIANA ZHANG
SPRING 2016

A thesis
submitted in partial fulfillment
of the requirements
for baccalaureate degrees
in Electrical Engineering and Computer Engineering
with honors in Electrical Engineering

Reviewed and approved* by the following:

Ram Narayanan
Professor of Electrical Engineering
Thesis Supervisor

Jeffrey Mayer
Associate Professor of Electrical Engineering
Honors Adviser

* Signatures are on file in the Schreyer Honors College.

ABSTRACT

This project combines ideas extant in ad hoc radar networking and noise radar to create a target location system that can be easily set up and is difficult to detect. This was simulated in MATLAB by placing a target, *m* transmitters, and *n* receivers at randomly generated X-Y coordinates and comparing the transmitted and received signals for the three receivers to determine target location. This successfully located a target, and despite limitations, shows a mathematically feasible way to create a discreet, quickly arranged radar locating system.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to thank Professor Narayanan for his wisdom, support, and never being too busy to make time for me. In addition, I extend my gratitude to Professor Mayer for his attention and his critical eye. I would also like to recognize Josh Allebach, who was kind enough to provide guidance when I had no idea how to begin this task. Finally, I would like to thank my family, friends, and mentors for always encouraging my mental and academic growth.

Chapter 1

Introduction and Motivation for Project

Ad hoc radar networks and noise radar have existed as disjoint research areas for some time, but our attempt is the first we know of to unify them.

Networks allow radar sensing systems to be more fault-tolerant and less error-prone due to additional sensing nodes, which reduce the impact of each individual node. That is, if a single node fails or provides an incorrect reading, this has less of an impact on the correctness and robustness of the system as a whole. Many areas of radar sensing networks have been explored, such as decision strategies [2], ambiguity [3], orthogonal netting for ad-hoc systems [4], and design with the possibility of ad hoc networks using intelligent clusterheads [1]. These ad hoc networks are of particular interest to this project, since ad hoc ("for this" specific purpose) networks do not depend on pre-made infrastructure, and thus require minimal setup time.

Noise radar has similarly been well investigated, including its challenges associated with the calibration of monostatic nodes [5], imaging with radar networks [6], through the wall imaging [7], and its applications [8]. Noise radar, which uses white noise as the transmitted signal, is difficult to detect, intercept, and jam due to its pseudorandom nature. Noise radar is difficult to detect (not suspicious, as there is noise everywhere), in contrast to fixed-frequency systems or chirps. However, they require a lot of bandwidth and computation to send, receive, and process. They also require a lot of power for their signals to not be lost to the environment.

The goal of this project is to simulate a radar location system that combines the benefits of an ad hoc radar network and noise radar, which allows it to both have minimal setup requirements as well as be difficult to detect. This would be useful for applications such as military target detection, when information is critical but being discovered is dangerous.

1.1 Problem Statement

With terrorism concerns, locating potential enemy targets in a variety of environments is critical. Sensors are becoming a more prevalent complement to create safer and more accurate target detection. However, sensors still include room for improvement in areas such as fault-tolerance and set-up overhead.

1.2 Research Goals

The goal of this research project is to simulate a method for locating a single target using an ad hoc noise radar network in MATLAB. This method should require minimal setup, be fault-tolerant, and be computationally viable to provide location data in real-time. These processing techniques used to locate a target should be useable in applications including military target location.

1.3 Thesis Organization

Chapter 1: Introduction and Motivation – Discussion of the current status of this research area and of the goals of this research project

Chapter 2: Theory – Introduction to relevant mathematics of radar and signal processing used in this project

Chapter 3: Supporting Model for Locating Targets – Discussion of the contextual simulation environment in which the explored processing methods operate

Chapter 4: Three Processing Approaches… – Introduction to implementation details and challenges of the three models for locating a target explored in this project

Chapter 5: Comparison of the Three Approaches – A qualitative comparison of the benefits and downfalls of each approach

Chapter 6: Discussion: Limitations and Assumptions – A discussion of the extent to which the model can hold, due to presuppositions.

Chapter 7: Alternative Approaches – A discussion on potential benefits offered by other approaches to ad hoc noise radar target location

Chapter 8: Results – The major lessons learned from this research project

Chapter 9: Conclusions and Future Work – A discussion of project takeaways and additional areas to explore in this area

Chapter 2

Theory

In this section, mathematical principles from radar and signal processing theory relevant to the project are introduced.

2.1 Radar

Radar (Radio Detection and Ranging), was first demonstrated in the 1880s and has since undergone tremendous development. The guiding principle of radar is that electromagnetic radiation is reflected/absorbed at different ratios for different materials at different frequencies. There are two major approaches to ranging using radio waves; one is based on pulses and the other is based on frequency modulation. While there are benefits to frequency modulation, such as increased resolution between pulses, this approach is more difficult to apply for a noise signal, where the phase changes of a transmitted signal are less predictable.

Thus, we approach the ranging problem from the pulse-delay ranging approach. For this approach, a signal is transmitted and a reflection is received. By comparing the received signal with the transmitted signal, we can determine the propagation delay τ_p.

Furthermore, given that electromagnetic waves travel at the speed of light c, we can determine the distance d between a transmitter/receiver and the target, as given by the equation:

$$d = \frac{c\tau_p}{2} \quad [1]$$

Equation [1] is associated with a monostatic radar, for which the transmitter and the receiver are at the same position. In the case of bistatic radar, where the transmitter and receiver are not at the same position, the following equation applies:

$$d' = c\tau_p \quad [2]$$

Where d' is the distance from the transmitter to the receiver back to the target. This difference is illustrated in Figure 1:

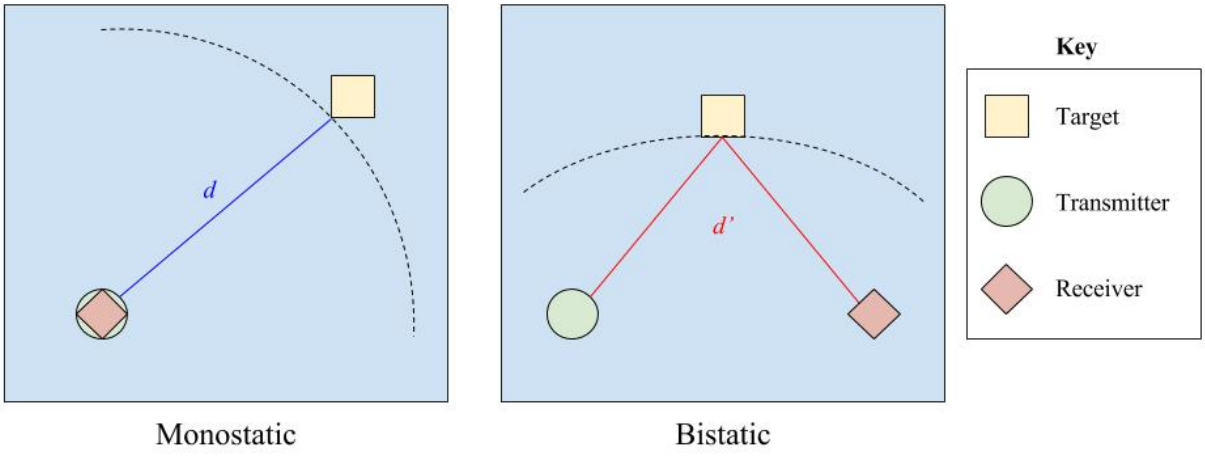

Figure 1. Illustration of Monostatic vs. Bistatic Radar Configurations

Note that given a distance measurement, monostatic radars have a circle with radius d of potential target locations. Bistatic radars, in contrast, have an ellipse of potential target locations with the foci at the transmitter and the receiver.

For this implementation, we will be primarily be working with the bistatic approach. This is because this formula supports increased generalization – by placing transmitters and receivers directly on top of each other, we could thus have a monostatic radar system. A bistatic approach, in contrast, could be useful if there was power to support only transmission or reception of noise radar signals, which can require very large amounts of power to remain meaningful after being scattered.

Pulse-delay ranging typically can have some uncertainty due to pulse repetitions, but since noise signals are pseudorandom, this is not a concern for our application.

For the purposes of this experiment, we are assuming that signals are transmitted from an isotropic antenna. An isotropic antenna transmits its signal power uniformly in a sphere, which gives the following power density equation:

$$Power\ Density\ at\ distance\ R\ from\ the\ transmitter = \frac{P_t}{4\pi R^2} \quad [3]$$

This power density is what the receiver will detect as the received signal. This decay of power density with distance will also be included in the simulation for the received signal.

2.2 Signal Processing

In a noise radar network with multiple transmitters, it is often impractical for all of the transmitters to send the same signal at the same time, as that would require connection by wires. This could cause a combination of setup cost, immobility, or a threat of tangled wires, all which

would ideally be avoided. Thus, each of *m* transmitters will send a different signal. Each receiver, then, must be able to take a signal that is a combination of the *m* signals, which were all received at different times, and determine the *m* distances d' from the receiver to the target to each transmitter.

To calculate this, apply the principle of cross-correlation. Cross-correlation measures how similar a signal is to a lag – or time-shifted version – of another signal. This is commonly used for applications such as echo cancellation. For this application, cross-correlation can be used to associate a time of arrival with each transmitted signal.

The expression for cross-correlation is as follows:

$$\int_{-\infty}^{\infty} f^*(t)g(t+\tau)dt \qquad [4]$$

Where f^*indicates the complex conjugate of some function f and τ indicates the delay. The point of best correlation gives the propagation delay τ_p, which can be plugged into equation [2] to find d'.

Chapter 3

Supporting Model for Locating Targets

In this chapter, the context for which the simulated network analysis algorithms are used will be discussed.

3.1 Overview & Layout

Following sensor network theory, our system has a base station that receives coordinate and noise signal information from each transmitter and receiver. With this information, it can then compute and send location information. For the purposes of this simulation, the computational base system is represented by the computer running the calculations in MATLAB. Transmitters, Receivers, and Targets are placed randomly and used to calculate received signals.

3.2 Processing Algorithms

There were three approaches used to determine target location mathematically: single-transmitter multiple-receiver, multiple-transmitter single-receiver, and multiple-transmitter multiple-receiver. Each approach provided the same functional output – the location of a single point target. The implementation details and challenges are discussed further in Chapter 4. The general shared functionality each approach could provide is discussed in Sections 3.3.5 and 3.3.6.

3.3 Simulation Model

This subsection discusses how the model was created, ie. how physical elements were represented and placed, how the signals were generated and the received signals simulated, and how this approach could then provide distance and location information.

3.3.1 Representation of Physical Elements

There are three types of system elements that will be modeled, all of which will be simulated as points for computational simplicity.

Transmitters – generate and send noise signals. Indicated with a green asterisk.

Target – reflect signals from transmitters towards receivers. Indicated with a yellow solid circle.

Receivers – receive summed noise signals. Indicated with a red hollow circle.

Note that a bistatic system was chosen, where transmitters and receivers do not necessarily have to be at the same location. This was done to support further generalization, and placing receivers at the same position as transmitters would thus support a network of monostatic nodes.

3.3.2 Placing Targets, Transmitters, and Receivers

To set up the network, system elements were assigned randomly generated *x* and *y* coordinates in MATLAB. Vectors were assigned using a random number generator for each object type and each axis: for example, ReceiverX = [0 1 2] and ReceiverY = [3 4 5] would jointly define three receivers, with X, Y

coordinates {(0, 3), (1, 4), (2, 5)}. The same assignments were done with TargetX and Y and TransmitterX and Y to assign positions to these system elements.

To display the system elements, the scatter function was used in MATLAB. An example is shown in Figure 2. The x-coordinate and y-coordinate indicate, in meters, where the elements fall. For an application, the base station would be able to calculate these positions using GPS information.

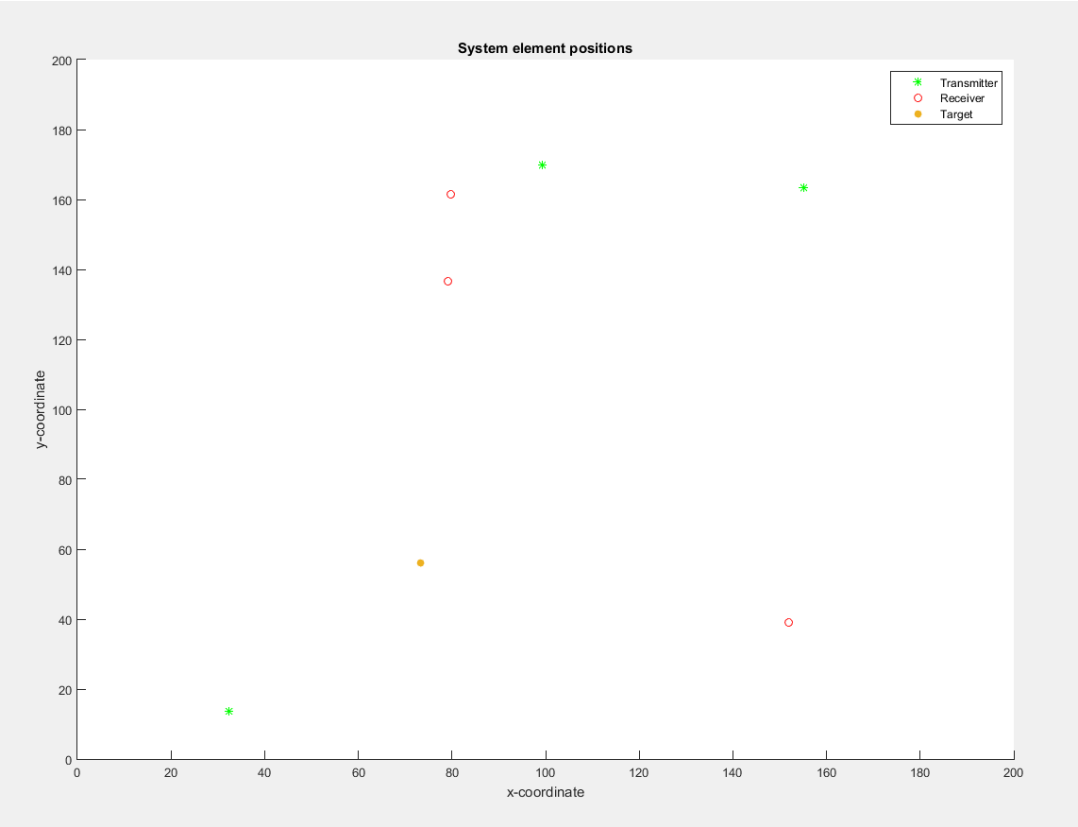

Figure 2. Example Scatter Plot of a Transmitter, Receiver, and Target

This method of setting up a network generalizes to other, larger networks, and our final MIMO setup supports *m* transmitters, *n* receivers, and a single target.

3.3.3 Generation of Transmitted Signal

Our transmitted signal for these simulations was white Gaussian noise filtered from 1MHz to 5MHz, with a power of 0 dBW. Since the goal of this project was to show the plausibility of ad hoc noise radar networks as a general approach, this was arbitrarily chosen. Another limitation of this generated signal is that it has a fixed number of points rather than being a continuous output of transmitted radio waves, as would be the case when transmitting signals in the field. This was chosen to keep signal sizes small and cross-correlation costs low for simulation purposes, without a loss of generality. Since noise signals are pseudorandom, treating them as packets as was done in this project and treating them as continuous signals does not affect the accuracy of cross-correlation.

3.3.4 Simulating Received Signals

To simulate a received signal, the distance from the transmitter to the target to the receiver was calculated. This distance was used to determine the propagation delay between when the signal was transmitted and received as well as how much the signal would have decayed by the time the transmitter received it. Since the distance between the transmitter and receiver is constant, each signal being sent from the transmitter to the receiver decays by a constant amount. There is also an offset in the received signal due to the propagation delay.

Below are the graphs for transmitted and received signals of a single-transmitter, multiple-receiver system. Each received signal has been delayed and has power reduced depending on its distance from the target. This is shown in Figure 3.

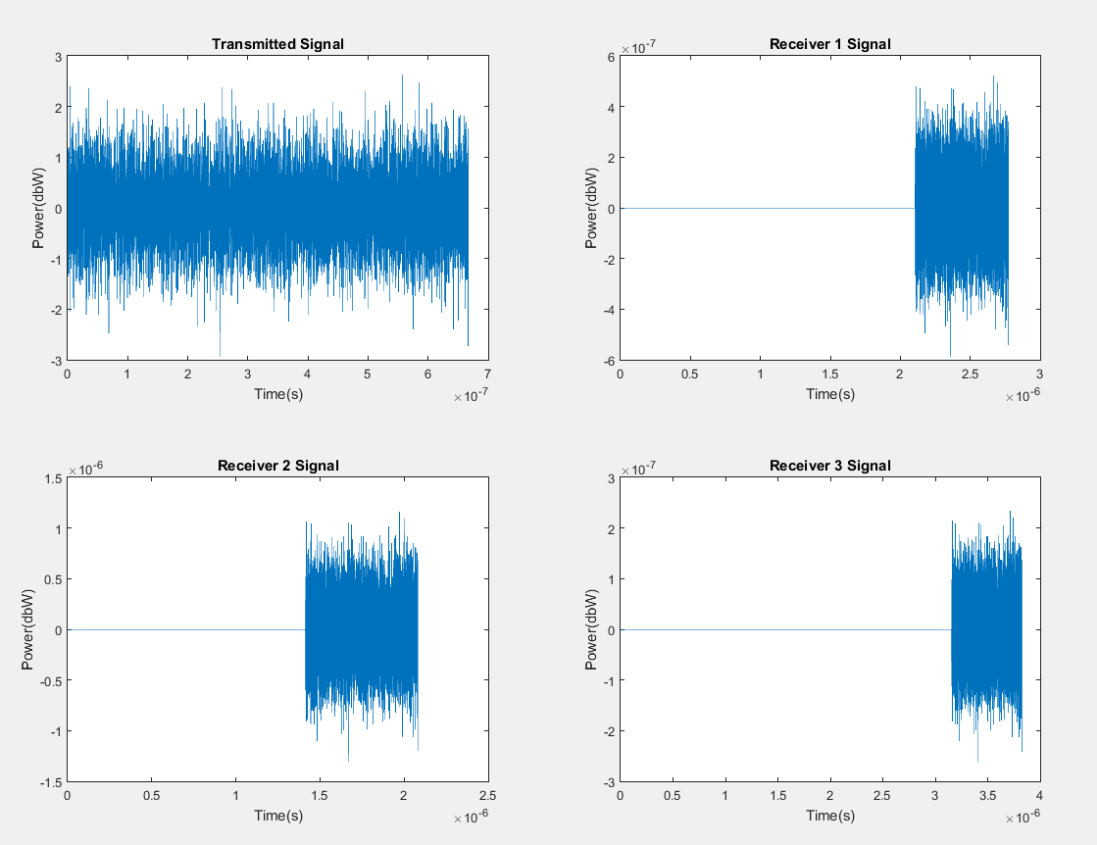

Figure 3. A Transmitted and a Received Signal for a Single-Transmitter System

Observe that the power of each received signal is lower if its propagation delay time is greater, as both of these factors correlate with distance.

For a multiple-transmitter system, each receiver receives what would effectively be the sum of all the signals it received from each transmitter. An example is shown in Figure 4.

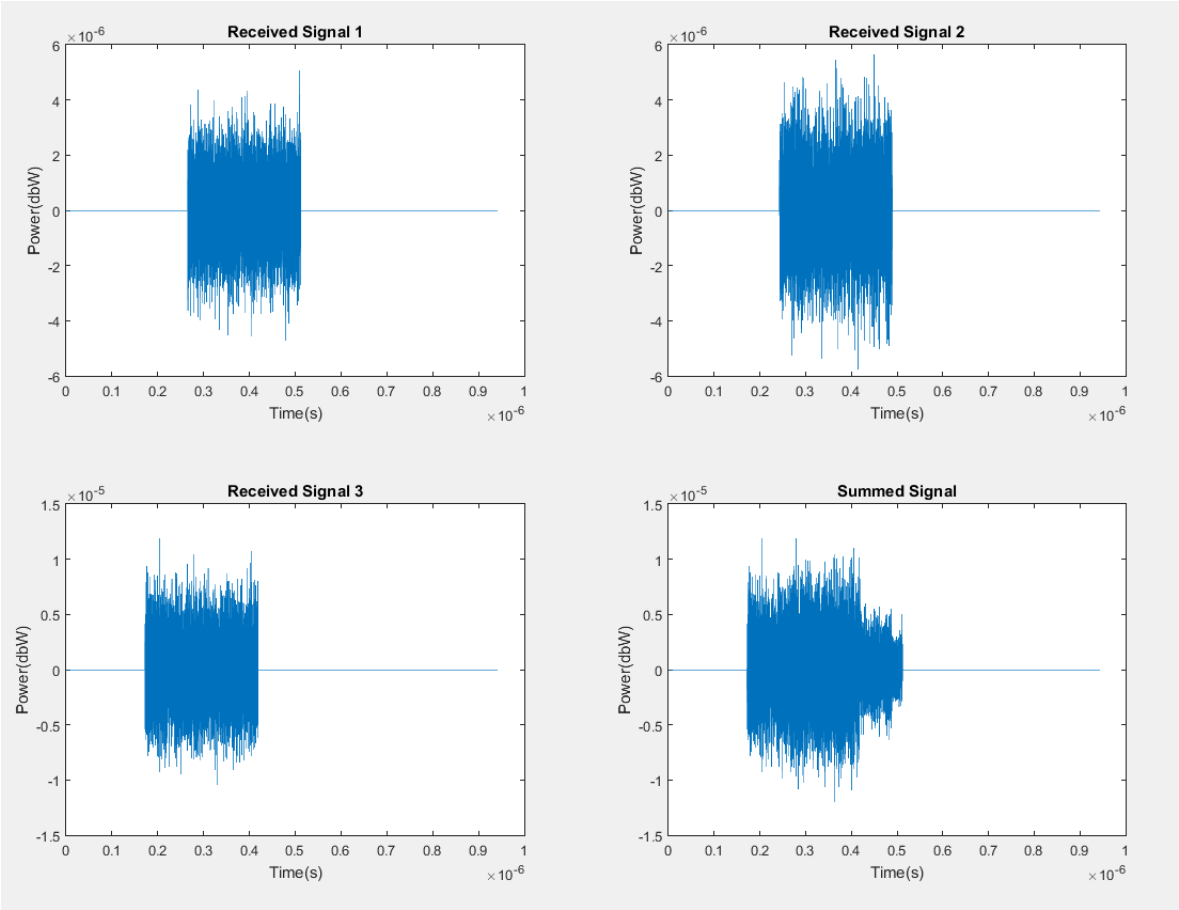

Figure 4. Signal Received by a Receiver in a Multiple-Transmitter System

Observe that signals that arrive later contribute less to the summed signal in power due to their greater distance. For signals that may be further away and be less reflective, an appropriate signal strength must thus be selected to isolate enough received signals to determine location.

3.3.4 Calculating Distance

To calculate distance, cross-correlation, as described in Section 2.2, was used. MATLAB's built in xcorr function was able to determine quite clearly where the point of best correlation. This was especially the case for a single-transmitter multiple-receiver case, as each receiver receives only the exact signal sent by the transmitter, as shown in Figure 5.

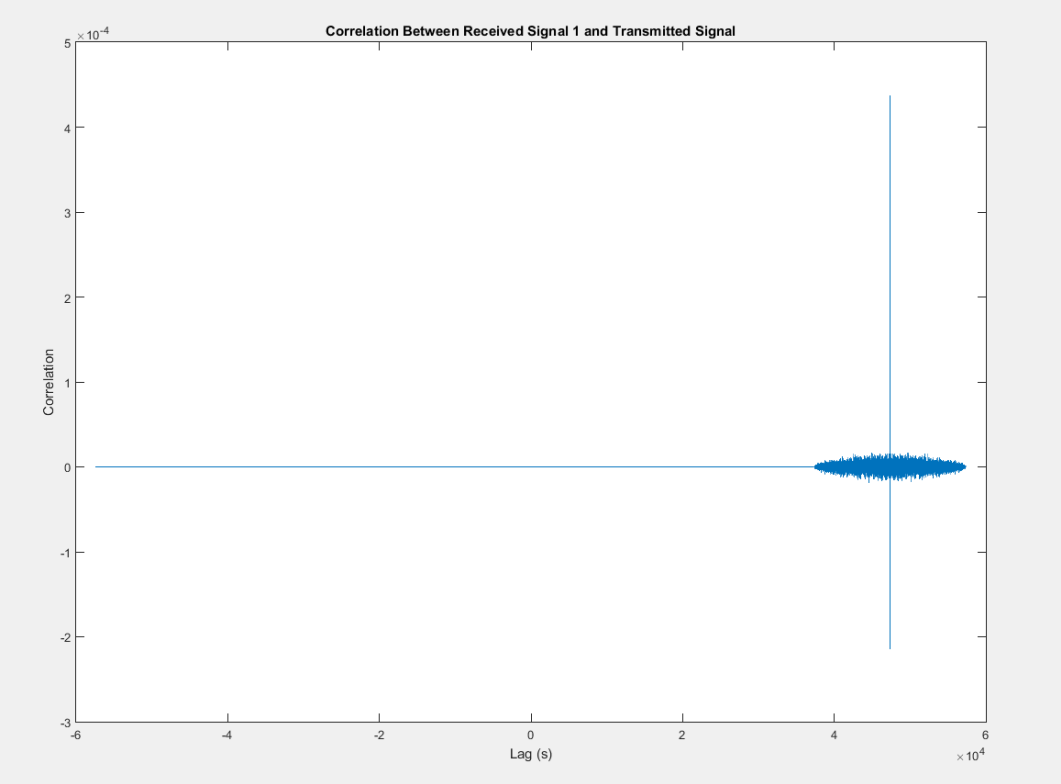

Figure 5. Cross-Correlation of Sent and Received Signals for a Single-Transmitter System.

This point of best correlation indicates a time delay, which can be used to calculate the distance from the transmitter to the receiver to the target as discussed in Section 2.1.

This point of best correlation is also quite evident in multiple transmitter systems, although it does appear qualitatively different – since there are other signals being added in, the graph is no longer perfectly symmetrical. This is shown in Figure 6.

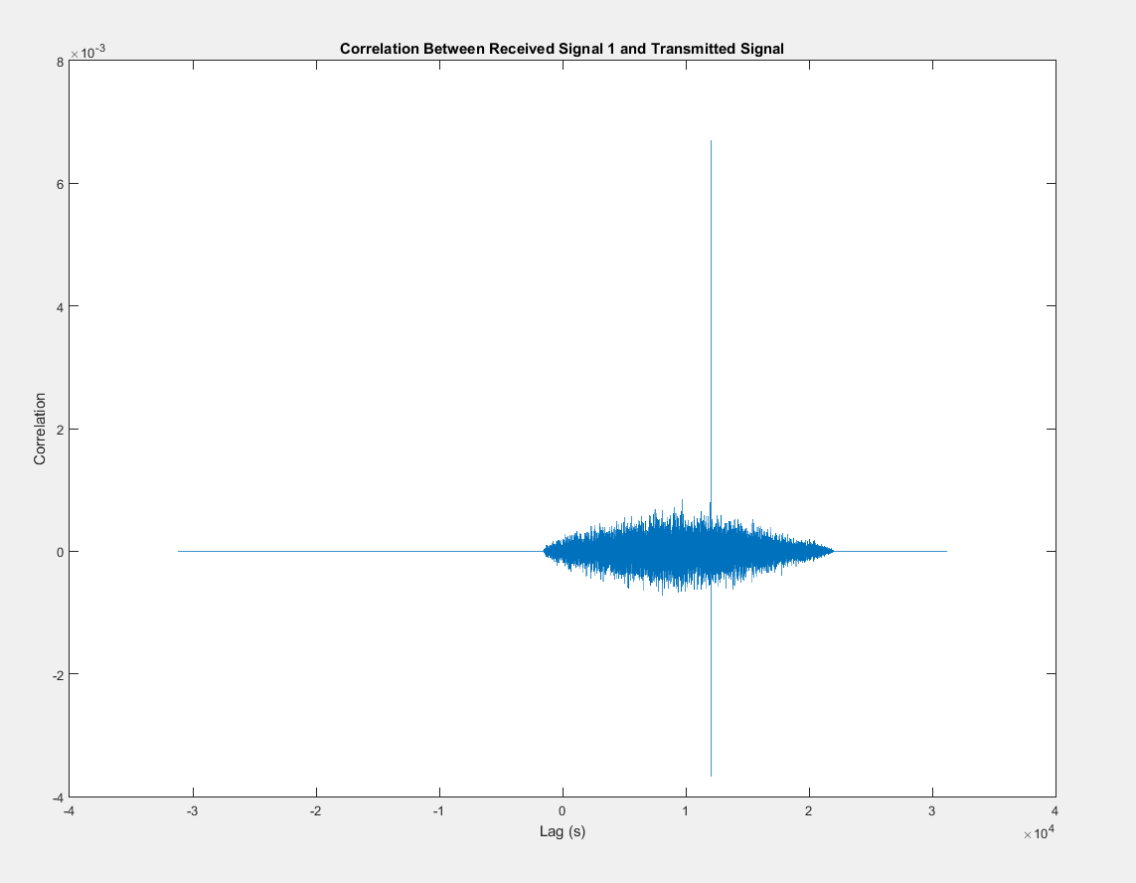

Figure 6. Cross-Correlation between a Received Signal from Multiple Transmitters and a Single Transmitted Signal

The point of best correlation, when observed, was always clearly evident. Thus, in both the single-transmitter and multiple-transmitter cases, cross-correlation was an effective tool to determine d', the distance from the transmitter to the target to the receiver.

3.3.5 Finding Potential Target Locations for a Transmitter-Receiver Pair

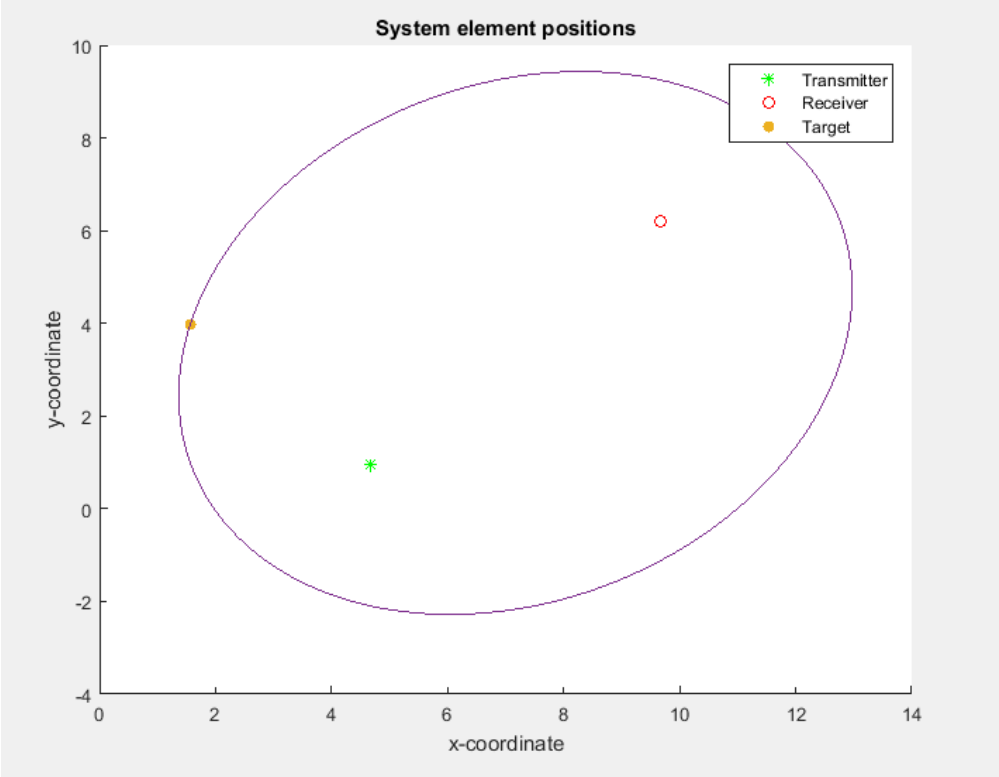

Figure 7. An Ellipse of Possible Target Locations for a Given Transmitter-Receiver Pair

With d' determined and knowing the location of the respective transmitter and receiver, an ellipse of possible target locations can be obtained; one such ellipse is pictured in the figure above. It is defined with the foci at the transmitter and the receiver, and a known vertex-to-vertex distance. Thus, with the aid of an ellipse plotting function developed by Roger Stafford at MathWorks, the ellipse defining possible locations a target could thus be plotted using polar coordinates.

3.3.6 Determining Target Location

Finally, the intersection point of at least three ellipses obtained in the previous step establishes the location of the target. To determine the intersection point, the algorithm developed by Douglas M. Schwarz of Rochester University was used. This algorithm breaks graphs into line segments and uses matrices to calculate whether the line segments intersect. Calculating the intersections of two potential target location ellipses produces two potential location targets; this intersection was calculated with two ellipse pairs to produce two intersection vectors. The element that was repeated in both vectors (within an arbitrarily chosen tolerance value of 0.3) was a coordinate of the target. Figures 8, 9, and 10 show the determination of potential target locations for each of the three approaches.

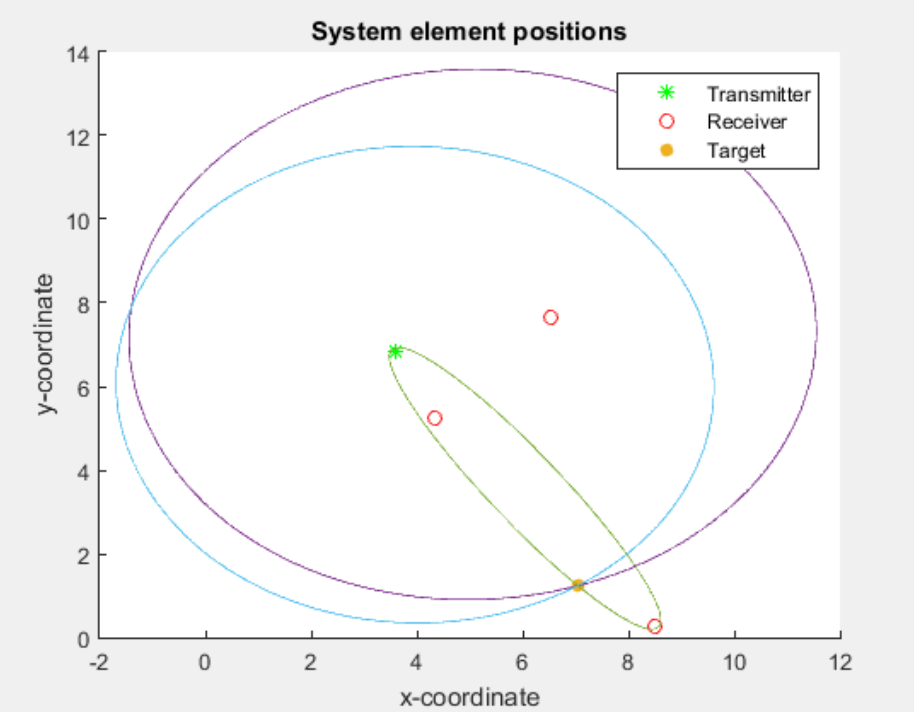

Figure 8. Located Target for Multiple-Receiver System

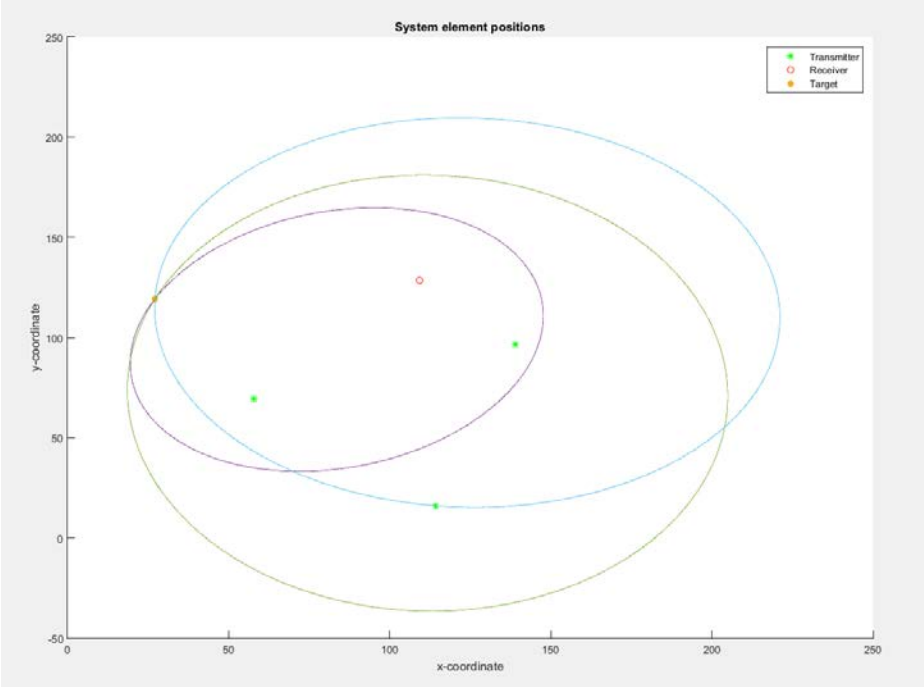

Figure 9. Located Target for a Multiple-Transmitter System

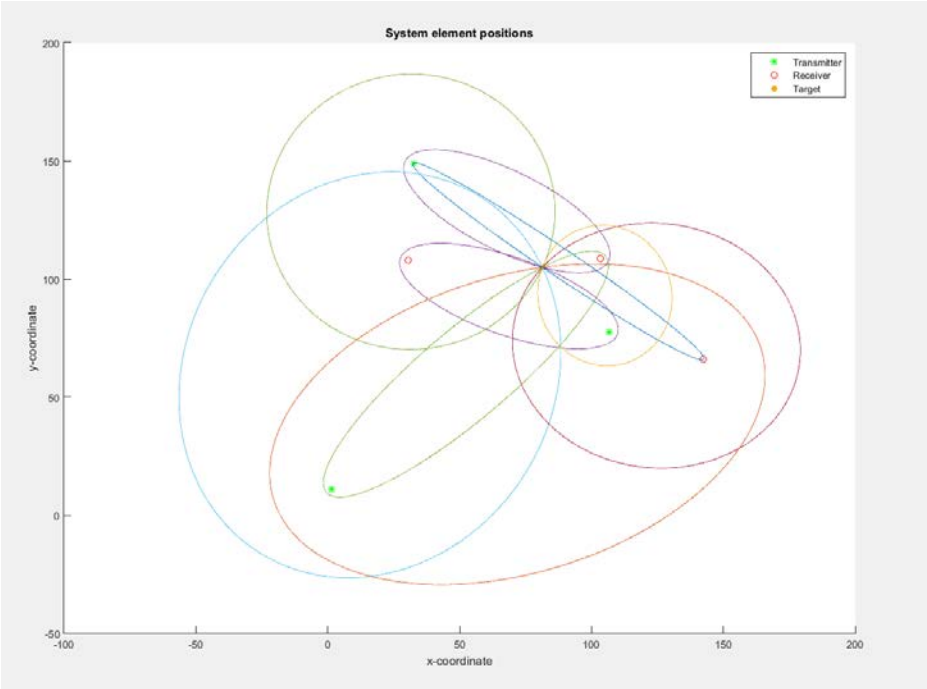

Figure 10. Located Target for a Multiple-Transmitter, Multiple-Receiver System

Chapter 4

Three Processing Approaches for Locating a Point Target using an Ad Hoc Noise Radar Network

In this section, the goals and challenges for implementing each of the three conceptual models for locating a point target using an ad hoc noise radar network are discussed.

4.1 Single-Transmitter, Multiple-Receiver

The Single-Transmitter, Multiple-Receiver approach for target location using ad hoc networks was the first to be implemented. As might be expected, the model involves one transmitter and n receivers. $n = 3$ was selected for minimal computational complexity and typical location success.

The goal of the single-transmitter, multiple-receiver approach was to develop a minimal model to show that this was a viable approach to locating targets. The single-transmitter, multiple-receiver model was fairly simple due to the fact that each received signal was exactly the transmitted signal.

The minimum possible number of transmitter-receiver pairs is three to triangulate the target's position. Thus, this was implemented using a transmitter and three receivers. This approach was not fool-proof, however – for example should the target, transmitter, and receivers all form a straight line, there would not be enough horizontal information to determine where the target is. Furthermore, if a transmitter or receiver wanders out of range, there would not be enough information. This could be ameliorated by adding additional receivers, and would be of less concern if the transmitters and receivers are moving about.

An example of an ambiguous target location is shown in Figure 11, where all three "ellipses" (which in this case, degenerate to lines) overlap in the entire region between the lowermost receiver and the transmitter.

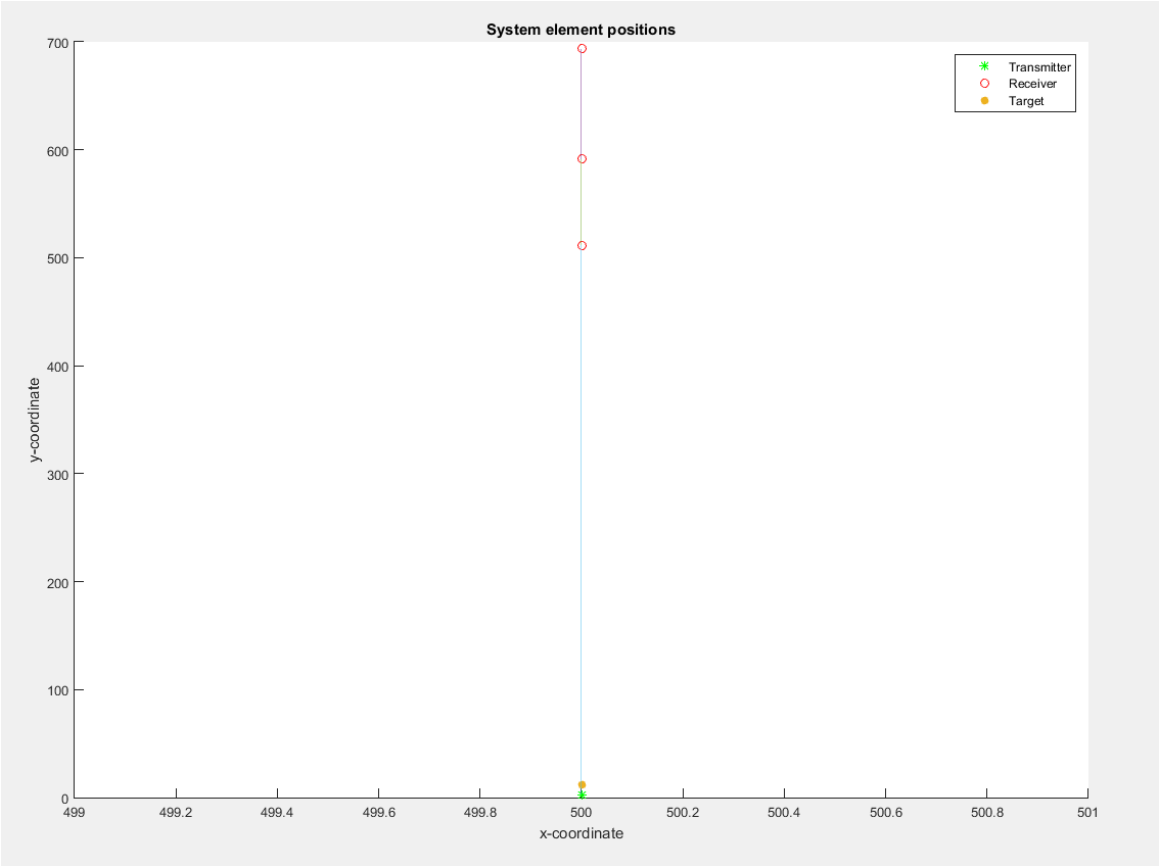

Figure 11. Example where Multiple Receivers would Fail to Locate Target.

Thus, the target could be located at any of those points and would not be successfully located. Further detail on the targeting locating methods used are discussed in Chapter 3.

4.2 Single-Receiver, Multiple-Transmitter

A Single-Receiver, Multiple Transmitter approach was simulated with one receiver and three transmitters. The Single-Receiver, Multiple-Transmitter approach was the next step in simulating a generalized ad hoc network for locating targets. Having a single receiver and multiple transmitters made the received signal in MATLAB more complicated to generate. Since vectors in MATLAB can only be added if they are the same length, calculating the signal as measured by the observer required padding all the arrays to the length of the longest array to ensure no signal was lost, and then subsequently adding them together. The different arrival times of the transmitted signal resulted in a few different ways such that they might overlap in the received signals. A few examples of different types of possible received signals are shows in Figures 12, 13, and 14.

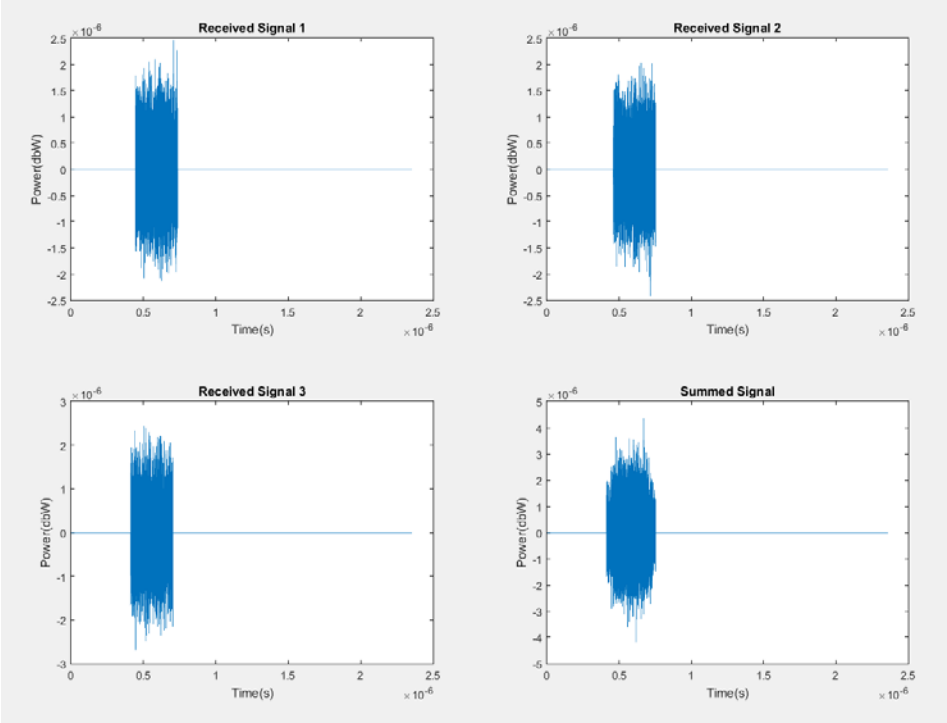

Figure 12. Three Summed Signals with Significant Overlap

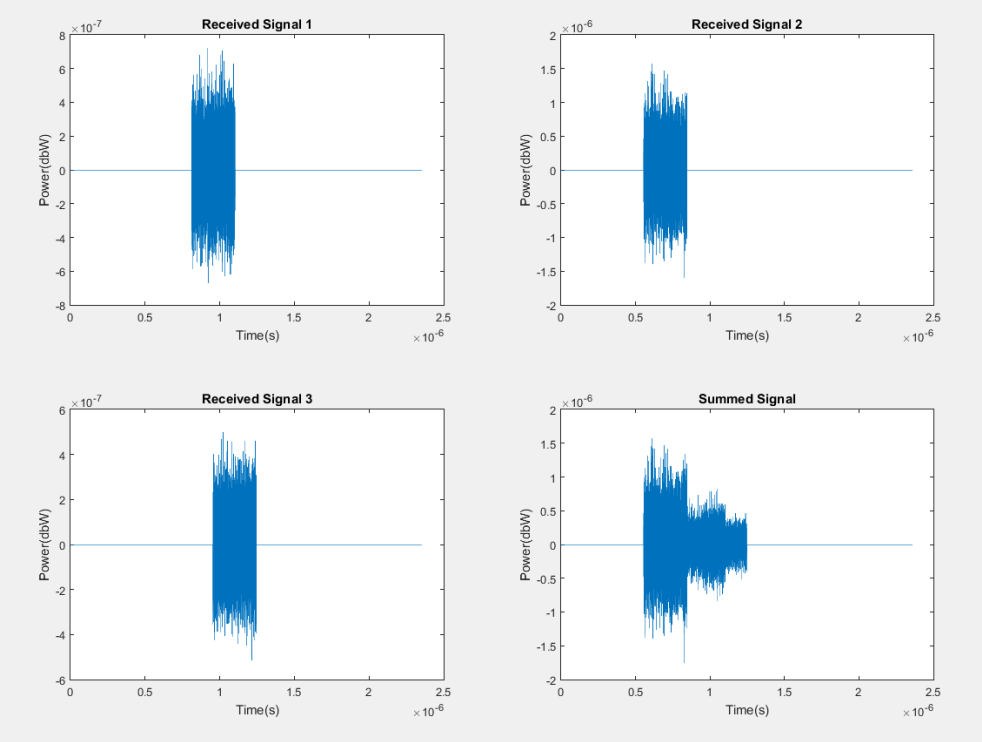

Figure 13. Three Summed signals with Minimal Overlap

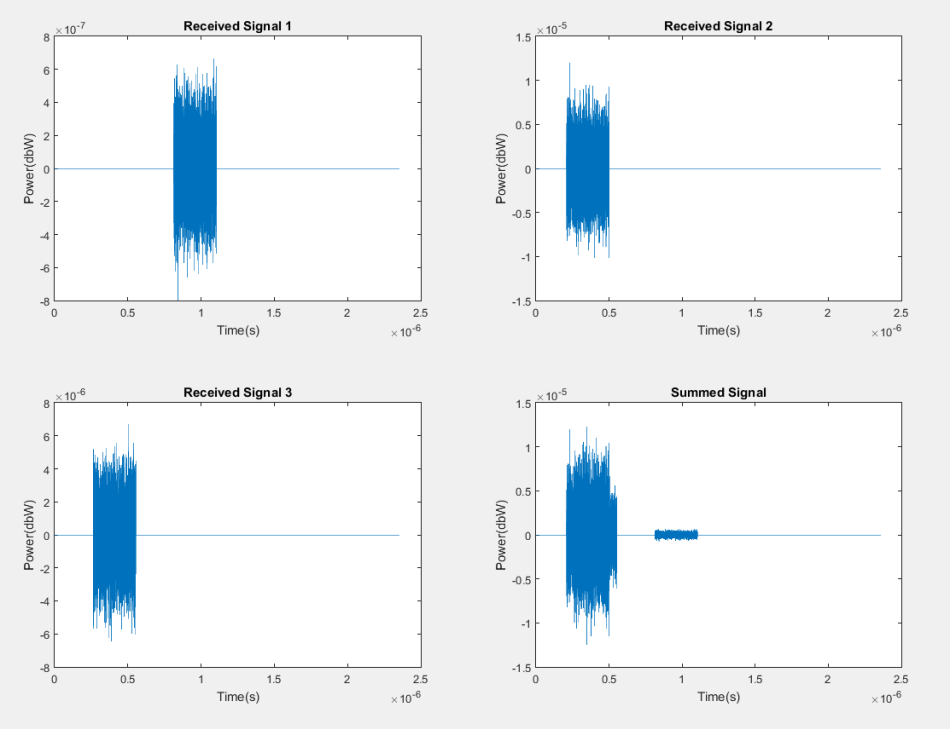

Figure 14. Three Summed Signals with Partial Overlap

Despite the effects of overlap and fairly significant qualitative differences between the received signals for the single-receiver multiple-transmitter and the single-transmitter multiple-receiver cases, cross-correlation still proved an effective way to determine distance, as discussed in Chapter 3.

Otherwise, the simulation of single-receiver, multiple-transmitter systems introduced few new challenges. The three-transmitter, single-receiver model used for demonstration purposes is similarly doomed to failure if the single transmitter fails or wanders out of range, or if the system elements are lined up incorrectly, as three is the bare minimum required to triangulate.

4.3 Multiple-Transmitter, Multiple-Receiver

The goal for the Multiple-Transmitter, Multiple-Receiver model was to combine the target location principles that were implemented for the Single-Transmitter, Multiple-Receiver model with the summed signals present in the Multiple-Transmitter, Single-Receiver model. A further goal was to generalize the model to support n receivers and m transmitters. This required restructuring the code to support an unknown number of signals, and had the additional benefit of being able to act as either of the other two approaches should n or $m = 1$.

This involved transitioning many hardcoded values and calculations to a loop-based approach. One such example, calculating the received signal in the Single-Transmitter, Multiple-Receiver model, is shown in Figures 15 and 16.

```
%% Calculate received signal
[receivedTime1, receivedSignal1] = calcReceivedSignal(Tx, Ty, ...
    Rx(1), Ry(1), Targetx, Targety, fs, L, signal);
[receivedTime2, receivedSignal2] = calcReceivedSignal(Tx, Ty, ...
    Rx(2), Ry(2), Targetx, Targety, fs, L, signal);
[receivedTime3, receivedSignal3] = calcReceivedSignal(Tx, Ty, ...
    Rx(3), Ry(3), Targetx, Targety, fs, L, signal);
```

Figure 15. Calculate Received Signals for a 3-Receiver, Single-Transmitter System

```
%% Calculate total received signal
receivedSignal = zeros(numReceivers * numTransmitters, maxSignalLength);
time = linspace(0, maxDelay, maxSignalLength);

% Calculate each received signal
% Using idivide to truncate. Signal number must match transmitter number!
% Postcondition: R11, R12, R13, R21, R22, R23 ...
for n = 0 : numReceivers * numTransmitters -1
    [~, tempSignal] =  ...
        calcReceivedSignal(Tx(mod(n,numTransmitters)+1),...
        Ty(mod(n,numTransmitters)+1), ...
        Rx(idivide(n,int8(numTransmitters))+1), ...
        Ry(idivide(n,int8(numTransmitters))+1), ...
        Targetx, Targety, fs, L, signal(mod(n,numTransmitters)+1, :));
    receivedSignal(n+1, :) = padarray(tempSignal, ...
        [0 maxSignalLength-length(tempSignal)], 'post');
end
```

Figure 16. Calculating Received Signals for an N-Receiver, M-Transmitter System

This was the technique used to generate calculated signals for each possible receiver-transmitter combination for *n* receivers and *m* transmitters. First, the maximum signal length was calculated and all received signals were stored as a 2-dimensional array to allow iteration in a for loop. Using integer division and the modulo operator, we can store then each received subsignal in the array using a for loop.

In addition to being more scalable, the Multiple-Input Multiple-Output configuration is more fault-tolerant. For example, if there are two transmitters, it makes less of an impact if one of them fails. The number of possible transmitter/receiver combinations increases when both *n* and *m* are greater than 1. The addition of another transmitter to a single-transmitter, three-receiver system would increase the number of transmitter/receiver pairs by 3, unlike the incremental benefit in single-transmitter or single-receiver systems.

Chapter 5

Comparison of Three Approaches

Single-Transmitter and Single-Receiver systems have fewer transmission/receiver pairs than Multiple-Transmitter Multiple-Receiver systems, which is a benefit from a computational perspective. Fewer signals have to be transmitted to the base station for calculation, and fewer cross-correlations need to be computed to determine distance. This saves both power and time, providing target information more quickly at less of an energy cost. Between these two, Single-Receiver, Multiple-Transmitter may be more difficult to implement in practice since the different transmitted signals may interfere with each other.

Multiple-Transmitter Multiple-Receiver systems, in contrast, is more computationally expensive but allows more support for loss of single node elements. A balanced approach for most applications may be two transmitters and multiple receivers, with one transmitter being used primarily and the other one existing as a backup – effectively a single-transmitter, multiple-receiver system with capabilities to expand beyond that.

Chapter 6

Discussion: Limitations & Assumptions

For these location detection systems to work, we assume that there is a centralized computer base system that controls the signals transmitted and receives the received signals and is aware of all transmitter and receiver positions. This base station could also exist within a system element – for example, a multiple-transmitter single-receiver system's receiver could have all the information necessary to locate a target's location as long as it knew what signals were being transmitted.

There are other assumptions made that would require modification should this system be physically implemented. Each receiver would need to account for interference from the other receivers and transmitters, for example – and since these are noise signals, adjustments would have to be made pointwise. The received signal would also have to remove the component of the signal that has directly been transmitted from the transmitter to the receiver without reflecting off the target, which could be calculated given the positions of the receiver and transmitter in question. Perfect reflection from the target was also assumed, so the actual received signal would need to be scaled by some factor $\alpha < 1$.

Furthermore, this model does not account for the Doppler effect. For this model, objects were assumed to be stationary, so the Doppler effect would not apply. This simulation also does not account for the fact that the receiver can receive a signal from the transmitter that has not bounced off the target. The theoretical removal of this signal should be mathematically trivial, as it would simply be subtracting the known signal with a known time delay (since the distance between target and receiver can be calculated).

Finally, this model does not account for different reflection coefficients at different frequencies, which could add complexity to the cross-correlation. However, if the different reflection coefficients are known, they can be divided from the received signal in the frequency domain to retrieve the originally transmitted signal.

Chapter 7

Alternative Approaches & Power Conservation Techniques

The assumed layout was chosen rather arbitrarily – in this chapter, alternative approaches that could also work based on the situation are discussed. The intention of this section is to provide alternative approaches to a problem for which much of the processing would be the same as discussed earlier in this paper.

7.1 An Alternative Network Configuration

The given configuration, with transmitting and receiving nodes and a processing base station, is only one potential approach to addressing this problem. There are other possible approaches, depending on the requirements of the system.

One alternative example would be instead of having only a base station, intermediaries could be set up between the transmitting and receiving nodes if transmitting power was constrained and if a sampled version of a noise signal would work similarly well. This is depicted in Figure 17.

Figure 17. Wireless Sensor Network with an Intermediary

Without the intermediary, assume that it would take the nodes 10 dB to communicate send data to the base at 250 kBaud at 315 MHz. For TI's CC1150 transmitter, that would draw 25.6 mA per node, or the equivalent of draining a AA battery in about 100 hours (thus transmitting 900MB). Meanwhile, transmitting to the intermediary halfway between the nodes and base would take about 3 dB or draw 15.8 mA, thus allowing each node to transmit about 1.5 GB on the same amount of battery life.

For this system, it would come at a slight loss of data to sampling and perhaps loss of complete information. If this comes at the cost of a few centimeters in certainty and conserves a significant amount of energy, though, perhaps it would be worthwhile.

7.2 An Alternative Signal Processing Method

Another way to locate a target besides cross-correlation could possibly be to use power measurements, which could save the energy required to communicate noise signals between system elements. This is shown in Figure 18.

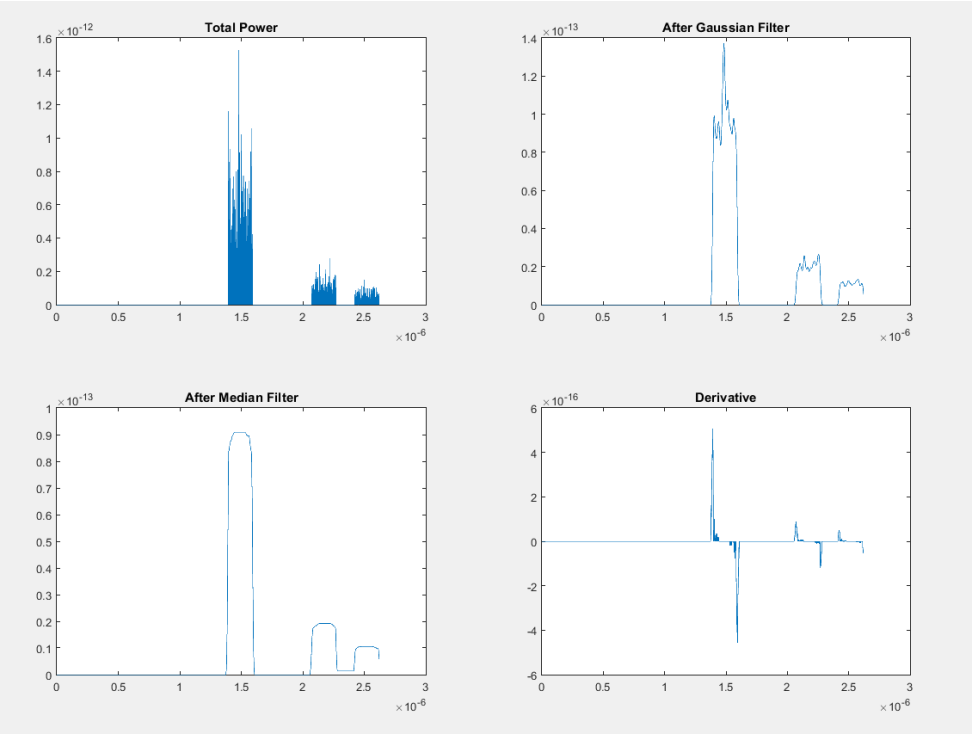

Figure 18. Signal Processing to Find Arrival Times using Power

Given the position and a distinct transmitted power of each transmitter, along with the calculated arrival times of signals (visible as peaks in the derivative graph), the target location could be determined.

7.3 Monostatic Approach over Bistatic Approach

To implement this as a monostatic radar, the transmitter and receiver can be set to the same coordinates in the simulation. This can be beneficial by requiring fewer users to carry transmitters and receivers. Users in the application would carry a dual transmitter/receiver module that can tolerate partial failures of the transmitter or receiver components. For the monostatic approach, the processing algorithms discussed earlier operate as expected, since the monostatic approach is one particular case of bistatic. A monostatic configuration for target location is shown below:

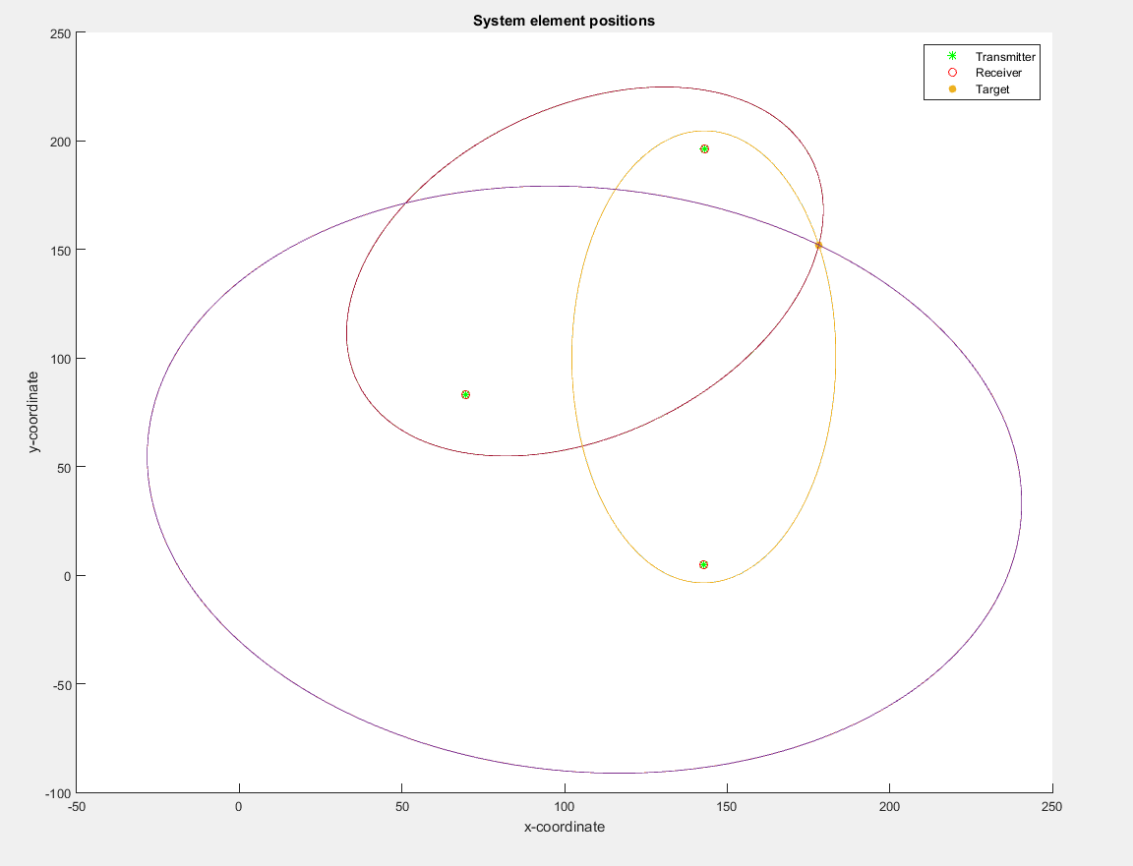

Figure 19. A Monostatic Target Location System

Chapter 8

Results

Simulation of target locating using an ad hoc noise radar network in MATLAB was shown to be implementable. Because of its ad hoc nature, these algorithms can apply to arbitrarily placed transmitters and receivers, thus giving it a low setup cost. The use of Gaussian white noise as the signal avoids detection, and supporting multiple transmitters and multiple receivers provides fault-tolerance via redundancy. With a base computer system with high processing power and virtually unlimited access to power (ie. plugged into an outlet), it is computationally viable for real-time applications.

Chapter 9

Conclusions & Future Work

This is a mathematically feasible approach to a radar network that is both difficult to detect (due to noise radar signals) and requires minimal setup (due to the ad hoc nature).

Moving forward, there are areas that can be explored further to enhance this model. For one, this model could be integrated with more complex models for targets, receivers, and transmitters, beyond simply point charges, as extensive research has been done on more detailed target modeling. This model could also benefit from further integration of non-ideal effects and the development of a certainty metric for target location, such as a % certainty as well as "within x meters" bounds. Physical implementation of the system would be beneficial by providing unwanted noise and non-ideal effects to refine the model, as well as testing the mathematical basis used on a packet-based approach with a continuous noise signal. Approximate computing could allow for faster processing time while still providing the gist of the information, depending on the application.

Computationally, further optimization of runtime would make this more practical as a real-time target locater. For example, if there were some upper limit during which the cross correlation stops running, that could cut out half the time dedicated to cross-correlation. Changing frequency ranges and sampling rates could also be useful depending on the application. Reflection coefficient information with the target could be combined with this to further enhance the model. Doppler effects or multiple targets could also be accounted for.

Ad hoc noise radar networks are a viable way to locate targets, but significant further work is necessary before seeing its use in the field and in other applications.

Appendix A

MATLAB Code for Multiple-Transmitter, Multiple-Receiver System

Below is the MATLAB code for the multiple-transmitter, multiple-receiver system. To achieve functionality of the single-transmitter or single-receiver systems, set n or m to 1.

Main Function: Main.m

```
clear;
close all;

%% Define Parameters
% Constants
c = 3e8;        % speed of light
n = 3;          % number of receivers
m = 3;          % number of transmitters
size = 200;      % size of coordinate grid

% Noise Signal Parameters
L = 10001;      % the number of points used in time vector
fl = 1E9;       % Lower Frequency for filtering in Hz
fm = 5E9;       % Maximum Frequency of Interest in Hz
fs = 15E9;      % sampling frequency (at least 2 * fm)
P = 0;          % power, in units dBW

maxDelay = ((size^2 + size^2)^.5)/c;   % longest possible delay time
maxSignalLength = ceil(maxDelay * fs * 2) + L; % longest possible signal
length

% Locations of system elements, in X-Y coordinates
% 3 Transmitter
Tx = size * rand(1, m);
Ty = size * rand(1, m);
numTransmitters = length(Tx);
% 3 Receiver
Rx = size * rand(1, n);
Ry = size * rand(1, n);
numReceivers = length(Rx);
% Target
Targetx = size * rand;
Targety = size * rand;
numTargets = length(Targetx);

%% Place Receivers, Transmitters, and Targets
figure(1)
hold on
```

```
% colorTransmit(1:numTransmitters) = .3;  % indigo
% colorReceive(1:numReceivers)  = .8;     % yellow
% colorTarget(1:numTargets) = .5;         % teal
scatter(Tx, Ty, 'g*')
scatter(Rx, Ry, 'r')
scatter(Targetx, Targety, 'filled')
legend ('Transmitter', 'Receiver', 'Target')
title ('System element positions')
xlabel ('x-coordinate')
ylabel ('y-coordinate')

%% Generate Transmitted Noise Signals
time = zeros(numTransmitters, L);
signal = zeros(numTransmitters, L);
for n = 1 : numTransmitters
    [time(n, :), signal(n, :)] = GenerateNoise(L, fl, fm, fs, P);
end

%% Calculate total received signal
receivedSignal = zeros(numReceivers * numTransmitters, maxSignalLength);
time = linspace(0, maxDelay, maxSignalLength);

% Calculate each received signal
% Using idivide to truncate. Signal number must match transmitter number!
% Postcondition: R11, R12, R13, R21, R22, R23 ...
for n = 0 : numReceivers * numTransmitters -1
    [~, tempSignal] =  ...
        calcReceivedSignal(Tx(mod(n,numTransmitters)+1),...
        Ty(mod(n,numTransmitters)+1), ...
        Rx(idivide(n,int8(numTransmitters))+1), ...
        Ry(idivide(n,int8(numTransmitters))+1), ...
        Targetx, Targety, fs, L, signal(mod(n,numTransmitters)+1, :));
    receivedSignal(n+1, :) = padarray(tempSignal, ...
        [0 maxSignalLength-length(tempSignal)], 'post');
end

totalSignal = zeros(numReceivers, maxSignalLength);

for n = 0 : numReceivers - 1
    tempSum = 0;
    for m = 0 : numTransmitters - 1
        tempSum = tempSum + receivedSignal(numTransmitters*n + m + 1, :);
    end
    totalSignal(n + 1, :) = tempSum;
end

% Plot components of one receiver
figure(2);
subplot(2, 2, 1)
plot(time, receivedSignal(1, :))
title('Signal 1');
xlabel('Time(s)');
ylabel('Power(dbW)');
subplot(2, 2, 2)
```

```
plot(time, receivedSignal(2, :))
title('Signal 1');
xlabel('Time(s)');
ylabel('Power(dbW)');
subplot(2, 2, 3)
plot(time, receivedSignal(3, :))
title('Signal 1');
xlabel('Time(s)');
ylabel('Power(dbW)');
subplot(2, 2, 4)
plot(time, totalSignal(1, :))
title('Summed Signal');
xlabel('Time(s)');
ylabel('Power(dbW)');

% % Plotting transmitted signals and the received signal
% figure(2);
% subplot(2, 2, 1);
% plot(time, totalSignal(1, :))
% title('Signal 1');
% xlabel('Time(s)');
% ylabel('Power(dbW)');
% subplot(2, 2, 2);
% plot(time, totalSignal(2, :));
% title('Signal 2');
% xlabel('Time(s)');
% ylabel('Power(dbW)');
% subplot(2, 2, 3);
% plot(time, totalSignal(3, :));
% title('Signal 3');
% xlabel('Time(s)');
% ylabel('Power(dbW)');

%% Using Power of Signal to determine distance
% power = totalSignal.^2;
%
% subplot(2, 2, 1);
% plot(time, power);
% title('Total Power');
%
%
% windowWidth = 500;
% halfWindow = windowWidth/2;
%
% gaussFilter = gausswin(windowWidth);
% gaussFilter = gaussFilter/sum(gaussFilter);
%
% smoothedPower = conv(gaussFilter, power);
% subplot(2, 2, 2);
% plot(time, smoothedPower(halfWindow:end-halfWindow));
% title('After Gaussian Filter');
%
% smoothedPower = medfilt1(smoothedPower, 4000);
% subplot(2, 2, 3);
% plot(time, smoothedPower(halfWindow:end-halfWindow));
```

```
% title('After Median Filter');
%
% derivative = diff(smoothedPower(halfWindow:end-halfWindow));
% subplot(2, 2, 4);
% plot(time(1:length(time)-1), derivative);
% title('Derivative')

%% Correlate Signals and find distance
distance = zeros(numTransmitters * numReceivers, L);
for i = 1 : numReceivers
    for j = 1 : numTransmitters
        distance(numTransmitters*(i-1)+ (j -1)+ 1) = ...
            xcorrDistance(signal(j, :), ...
            totalSignal(i,:), fs);
    end
end

%% plot ellipses of possible locations for each receiver
% with foci (Rx, Ry), (Tx, Ty) where [distance] is the
% distance from (Tx, Ty), to the ellipse, to (Rx, Ry)

% figure(1);
%
%         [xtemp, ytemp] = ellipse(Tx(1), Ty(1), Rx(1), Ry(1), distance(3*(n-
1)+3*(m-1)+1));
%         plot(xtemp, ytemp);
figure(1);
for n = 1 : numReceivers
    for m = 1 : numTransmitters
        [xtemp, ytemp] = ellipse(Tx(m), Ty(m), Rx(n), Ry(n),
distance(numTransmitters*(n-1)+ (m -1)+ 1));
        plot(xtemp, ytemp);
    end
end

hold off;
```

Noise Generation: GenerateNoise.m

```
%% This script generates a noise signal
% Modified from code from Josh Allebach
% Parameters:
% Inputs  -     L:      number of points used in the time vector
%               fl:     lower frequency bound
%               fm:     maximum freqency bound
%               fs:     sampling frequency
%               P:      Power (units dBW)
% Outputs -     time: the time scale of the generated noise signal
%               signal: a noise signal
function [time, TxTime] = GenerateNoise(L, fl, fm, fs, P)
```

```
% Define the time period of sampling:
T = 1/fs;

% Define the time vector:
time = linspace(0, T*L, L);

%% Generate the Noise Signal:
%   This is uing the 'wgn' function from MATLAB. It generates a white
%   Gaussian vector of length L with the output in terms of voltage.  The
%   inputs are the length of the vector and the average power of the noise
%   signal.

%   The input power, which is a required input, should be entered in units
%   of decible watts (dBW).  x dBW = 10*log10(X_watts).

%   The 'wgn' assumes a impedance load of 1 ohm.  This can be modified if
%   specific values are known, however it is common to just use 1 ohm since
%   it is essentially only a scaling factor between the power and the
%   voltage.

%   The 'wgn' can also be modified to output complex noise.  In this case,
%   the power is split 50/50 between the real and imaginary components.
%   See "help wgn" for more.  The default is a real valued signal.

% Generate signal with L elements and P dBW:
Signal = wgn(1, L, P);

%% Filter the Signal around the Frequencies of Interest:

% Notes on Generation:
%   The 'butter' function desgins a Butterworth filter.  Can modify so a
%   lowpass, bandpass, or highpass filter is generated.  For this case a
%   bandpass filter, around the frequencies of interest, is being
%   generated.

%   Essentially gives the 'b' and 'a' coeffs for the filter depending on
%   the inputs.  For the bandpass case need to give the low and upper
%   frequencies normalized by the sampling frequencies.

%   See the help for more info.

% Generate a 4th order filter from Flow to Fhigh:
[bFil,aFil] = butter(4, [(fl) (fm)]/(fs/2));

% Apply Filter:
TxTime = filtfilt(bFil, aFil, Signal);
```

Calculating Received Signal: calcReceivedSignal.m

```
function [time, signal] = calcReceivedSignal(Tx, Ty, Rx, Ry, Targetx,
Targety, fs, L, signalTransmit)
%% Calculate received signal
% Calculate Delay (using radar range equation)
% distance from transmitter to target
dtT = sqrt((Tx-Targetx).^2 + (Ty-Targety).^2);
% distance from target to receiver
dTr = sqrt((Rx-Targetx).^2 + (Ry-Targety).^2);
% Total Distance
dTotal = dtT + dTr;
% Calculate dtime
dtime = dTotal/3e8;

% Decay signal strength
subSignal = signalTransmit ./ (4*pi*(dtime*3e8).^2);

% delay signal by time
sampleShift = fix(fs * dtime);        % number of samples delayed
signal = [zeros(1, sampleShift),subSignal(1:end)];
time = linspace(0, L/fs+dtime, L+sampleShift);
```

Determining Distance: xcorrDistance.m

```
%% This script calculates distance using cross-correlation
% Parameters:
% Inputs  -     signal: transmitted signal
%               receivedSignal: received signal, for comparison
%               fs:     sampling freqency
% Outputs -     distance: the distance traveled by received signal,
%                         in comparison to transmitted signal

function distance = xcorrDistance(signal, receivedSignal, fs)

[acor,lag] = xcorr(receivedSignal, signal);
% figure(3);
% plot(lag, acor);
% title('Correlation Between Received Signal 1 and Transmitted Signal');
% xlabel('Lag (s)');
% ylabel('Correlation');
[~, idx] = max(acor);
timedelay = lag(idx)/fs;
distance = timedelay * 3e8;
```

BIBLIOGRAPHY

[1] Jing Liang; Qilian Liang, "Design and Analysis of Distributed Radar Sensor Networks," in Parallel and Distributed Systems, IEEE Transactions on , vol.22, no.11, pp.1926-1933, Nov. 2011

[2] Srinivasan, R., "Distributed radar detection theory," in Communications, Radar and Signal Processing, IEE Proceedings F , vol.133, no.1, pp.55-60, February 1986

[3] Papoutsis, I.; Baker, C.J.; Griffiths, H.D., "Netted radar and the ambiguity function," in Radar Conference, 2005 IEEE International , vol., no., pp.883-888, 9-12 May 2005

[4] Deng, Hai, "Orthogonal netted radar systems," in Aerospace and Electronic Systems Magazine, IEEE , vol.27, no.5, pp.28-35, May 2012

[5] Nelms, M.E.; Collins, P.J., "Development and evaluation of a multistatic ultrawideband random noise radar network," in Radar Conference (RADAR), 2011 IEEE , vol., no., pp.1068-1073, 23-27 May 2011

[6] Ashley Schmitt, “Radar Imaging with a Network of Digital Noise Radar Systems (Master’s Thesis).” Available at http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA498641.

[7] Pin-Heng Chen; Shastry, M.C.; Chieh-Ping Lai; Narayanan, R.M., "A Portable Real-Time Digital Noise Radar System for Through-the-Wall Imaging," inGeoscience and Remote Sensing, IEEE Transactions on , vol.50, no.10, pp.4123-4134, Oct. 2012

[8] Ram M. Narayanan ; Yi Xu ; Paul D. Hoffmeyer and John O. Curtis "Design, performance, and applications of a coherent ultra-wideband random noise radar", Opt. Eng. 37(6), 1855-1869 (Jun 01, 1998).

ACADEMIC VITA

Academic Vita of Diana Zhang
doz5072@psu.edu

Education

B.S. in Electrical Engineering, Computer Engineering **May 2016**
The Pennsylvania State University, Schreyer's Honors College
Thesis: Simulating Ad Hoc Noise Radar Networks for Target Location Detection
Advisor: Professor Ram Narayanan

Teaching Experience

Penn State School of Electrical Engineering and CS, Teaching Intern **Fall 2014 - Present**
Explained concisely and clearly core electrical engineering, computer engineering, and computer science concepts
Enriched student experience with additional conceptual explanations
Connected material to industry and subsequent coursework
Previous: Troubleshooted hands-on work with students as Lab Assistant

Penn State Learning, Math Tutor **June 2013 - Present**
Led exam review sessions with a partner to rooms of 100+ students
Assisted students with Multivariable Calculus, Differential Equations, Matrices
Motivated students using applications to their particular fields

Research Experience

Clemson University, Undergraduate Research Assistant **Summer 2014**
Designed power system for a solar-powered, ultra-low power sensor module
Utilized CAD tools, a repository, and collaboration for effective design
Presented a midterm oral presentation and a final poster presentation

Penn State College of IST, *Research Assistant* **Jan. – Aug. 2013**
Researched mobile learning technologies independently and applied research toward development of a mobile learning application
Contributed to writing grant proposals and scholarly articles
Considered effective learning strategies and student perspectives to help implement online learning tools in IST 110H and IST 413

Honors and Awards

Penn State Schreyer's Gateway Scholars Program Scholarship	Fall 2013 – Spring 2016
Penn State College of Engineering Honors Scholarship	Fall 2012 – Spring 2016
Penn State Dept. of Electrical Engineering Scholarship	Fall 2013 – Spring 2016
College of IST Summer Undergraduate Research Fellowship	Summer 2013
Schreyer's Honors College Travel Grant	Spring 2015
Penn State College of Engineering Langdon Travel Endowment	Spring 2015

University Service & Involvement

President, Penn State Association of Women in Computing **Sept. 2014 – Sept. 2015**

Led efforts to attract and retain women in computing at Penn State
Coordinated events with companies and Penn State student organizations
Previous: Organized Socials, Apparel, and Meetings as Secretary

Music Director, No Strings Attached a cappella ensemble **Mar 2013 – Dec 2014**

Led rehearsals twice a week, focusing on developing musicality
Planned long-term ensemble performance goals, and scheduled accordingly
Motivated members to stay engaged throughout rehearsals
Previous: Tenor Section Leader

Board of Directors, CodePSU Programming Competition **Oct 2013 – May 2015**

Outreach Committee Member, HKN Honors Society **Dec 2013 – Dec 2014**

Technical Skills

Hardware Design Tools: Verilog, Multisim/SPICE, Eagle, ModelSim
Programming: C, C++, Java, Python
Assembly-Level Programming: ARM, MIPS, CPU12
Other Software: MATLAB, Linux, FreeRTOS, Esterel, GitHub, Mercurial

Educational Travel

University of Auckland, New Zealand **Semester 1, 2015**

Studied Embedded Systems Design, NZ History, and NZ Popular Music
Immersed myself in a foreign culture
Logged & reflected on experience at https://thenewzealandzealot.wordpress.com/