Utilizing I/Q Data to Enhance Radar Detection and Accuracy Metrics

by

Alexandria Velez

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2019

Author . . . .
Department of Electrical Engineering and Computer Science
May 18, 2019

Certified by . . . .
John N. Tsitsiklis
Professor of Electrical Engineering
Thesis Supervisor

Certified by . . . .
Josh Erling
Group Leader, MIT Lincoln Laboratory
Thesis Supervisor

Accepted by . . . .
Katrina LaCurts
Chairman, Department Committee on Graduate Theses

Utilizing I/Q Data to Enhance Radar Detection and Accuracy Metrics

by

Alexandria Velez

Submitted to the Department of Electrical Engineering and Computer Science
on May 18, 2019, in partial fulfillment of the
requirements for the degree of
Master of Engineering

Abstract

The incorporation of advanced digital processing technologies including high-bandwidth networks, low-cost commercial components, and advanced FPGAs into novel radio frequency (RF) sensors has resulted in significantly increased sensor capabilities while at the same time dramatically increasing the size of the data associated with test events. This work focuses on the development of management tools to analyze these large datasets to increase overall situational awareness and as a result, sensor performance which requires the development of advanced algorithms designed to address data decimation, parallelization of processing, and novel detection and filtering techniques among others. These algorithms are developed and optimized through post-processing existing MIT-LL sensor data in MATLAB.

Thesis Supervisor: John N. Tsitsiklis
Title: Professor of Electrical Engineering

Thesis Supervisor: Josh Erling
Title: Group Leader, MIT Lincoln Laboratory

Acknowledgments

I want to thank Andrew Daigle at Lincoln Laboratory and my advisor at MIT, John Tsitsiklis, for guiding me through my project and for always keeping their doors open whenever I had a question or concern.

I would also like to thank my family for their continuous support throughout my years at MIT and for always believing that I could be better than I ever thought I could be. I will forever be grateful to them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Radar uses radio waves to detect the range, Doppler (velocity), and the azimuth of target objects. The earliest radar systems were developed shortly before the start of the Second World War to allow the Royal Air Force to detect Nazi ships crossing the channel before they could attack English towns and cities. At the time, targets appeared as just a blip on a screen. Because radar systems then were not at all as sensitive as the modern radars today, clutter was rarely a problem since larger objects, such as ships, were displayed as these blips, while smaller objects did not appear at all [1]. Advancements in radar technologies have expanded the range of applications for radar systems, and have improved radar performance and efficiency. Consequently, modern radars face the issue of accurately determining and tracking the presence of a target signal in unwanted clutter and noise.

1.1 System Overview

Figure 1-1 presents a diagram of a modern radar system. The current radar system used for this project consists of a series of 8 cross-dipole antennas for a total of 16 receiver channels. A waveform generator generates a low power signal that is amplified by the high power amplifier to some power P_t. The high power signal is sent to the transmitter and is emitted by the antenna with antenna gain G and antenna aperture A_r. The signal travels through space with propagation factor F until it comes across some object with radar cross section, σ_r some distance, R, away. The portion of the signal that has crossed paths with the object is reflected back to the antenna. The power received by the receiver is given by Equation 1.1. This equation is more commonly known as the radar-range equation. The data is then filtered and processed to maximize the signal while minimizing the noise, so that target detection is an easier and simpler process.

$$P_r = \frac{P_t G_t A_r \sigma_r F^4}{(4\pi)^2 R^4} \tag{1.1}$$

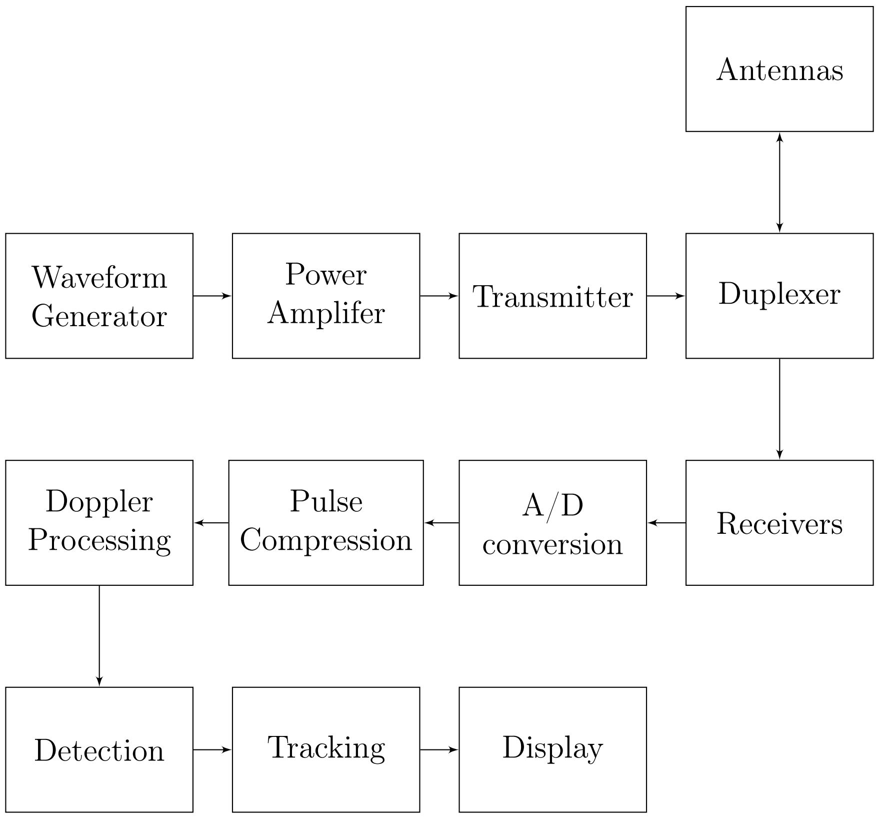


Figure 1-1: A block diagram of a modern radar system

1.1.1 Processing Tool

The radar signal processing tool consists of three major sections: filtering and Doppler processing, target detection, and direction finding. In part 1 of the design, the raw quadrature data (or I/Q data) from the radar is subject to preprocessing and filtering to compress the overall size of the data and reduce total runtime. A matched filtering pulse compression technique is used on the channel data in order to maximize the signal-to-noise ratio (SNR) in the presence of stochastic noise and increase range-resolution. After pulse compression, the reduction of input data size is implemented through one of two methods: data decimation and downsampling. Data decimation consists of two steps: first is the reduction of high-frequency signals through a digital low-pass filter, then the downsampling of the newly filtered data by a factor r by keeping only every r^{th} sample point. Both decimation and downsampling successfully reduce the total sample size by a factor of R. However, decimation has the added benefit of removing aliasing effects from the downsampling of the data.

The runtime of decimation is significantly longer than that of downsampling alone, and it only increases as the order of the filter and size of the data increases. For this project, the data has a high enough sampling frequency compared to its bandwidth that aliasing is avoided. Therefore, decimating the data would add unnecessary computational load, while downsampling would provide the most advantages in the creation of a real-time data-processing tool.

Finally, a range-Doppler map is generated for each time instance by applying a Fourier transform to convert the data to channel space in the order of 16 range-Doppler maps (RDMs). This data is run through a beamformer to generate 11 evenly-spaced beams for each time interval for a total of $B * T$ range-Doppler maps, where B is the number of beams and T is the number of time instances. Again the data size is reduced by having a smaller number of beams than channels, and we get some information about the angle from the data alone.

From here, we move to the second section of the tool: target detection. The original detection algorithm takes the sum of the power across each of the 11 beams and performs a target detection algorithm on the resulting range-Doppler map. The result of performing the detection algorithm on the resulting range-Doppler map is a logical matrix of the same size, where a value of 1 corresponds to a target being present, and a value of 0 corresponds to a target not being present. The logical matrix is then multiplied by the summed range-Doppler matrix to receive a matrix of target powers across range and Doppler.

One goal of the system is to reduce the number of targets to be processed. Finding the regional maximum of the matrix does this by only taking the detection with the highest power level in each connected group of detections as they are more likely to be target detections, therefore speeding up the process while minimizing the range, Doppler, azimuth errors in comparison to the truth data. This method works under the assumption that no two aircraft are operating relatively near each other in both range and Doppler. Any nonzero value resulting from the regional max matrix is considered to be a target.

The final section runs a direction finding algorithm called Root-MUSIC [2] to find the location of targets in azimuth. The current implementation of this algorithm, while efficient, is not as accurate as is desired for the current system, with angle estimation errors as high at seven degrees for a single target.

1.2 Thesis Overview

This thesis consists of five chapters. In Chapter 2 of this thesis, I will describe the original implementation for target finding in the system and introduce the various Constant False Alarm Rate (CFAR) algorithms and their performance relative to the original target finding algorithm. In Chapter 3, I will present the monopulse direction-finding algorithm. I will discuss how this direction-finding algorithm further processes the detection data to find the targets' angle of arrival. In Chapter 4, I will introduce the Extended Kalman FIlter and propose how to use the combined results from Chapter 2 and Chapter 3 for multiple target tracking to improve system accuracy. Lastly, in Chapter 5, I will summarize and analyze how the improved target and direction finding algorithms as well as the Kalman Filter tracker overall improved system accuracy and efficiency.

Chapter 2

Detections

The simplest way to detect objects from a series of range-Doppler maps is by determining a constant threshold that any cell is unlikely to exceed if it consisted solely of background noise. This threshold can be determined by examining the distributions of both the signal and noise. Since the maximization of the probability of detection is desired, a threshold should be chosen that brings the detection probability close to 1.0.

Frequently, there will be some overlap between the signal and noise distributions (see Figure 2-1). In such cases, determining a threshold is more complicated than choosing an arbitrary value that the noise distribution is unlikely to exceed, as it is also desirable to have a high probability of detection while maintaining a low probability of false alarm. Instead, information from the cells immediately surrounding the cell being tested for the presence of a target should be used to determine the threshold for target detection in that cell. Nevertheless, if clutter, interference, or multiple targets are present in the data, there is a higher probability for misclassification of both target and non-target cells.

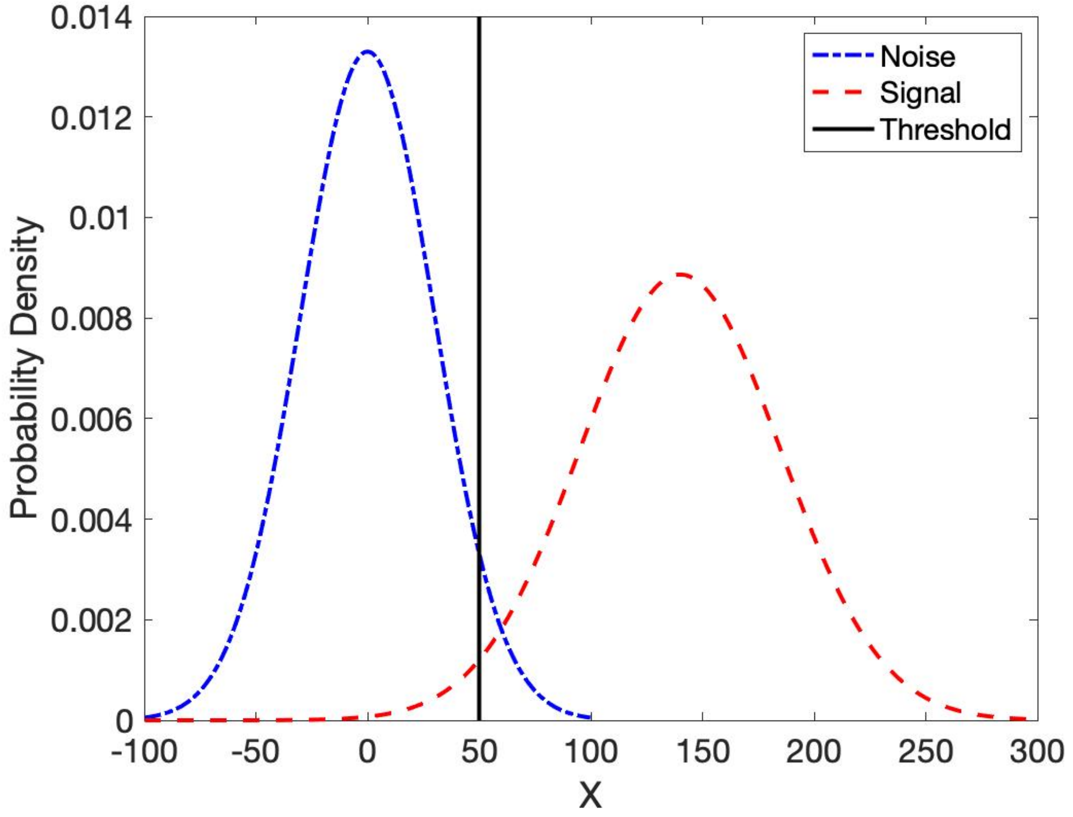


Figure 2-1: Displays an example of overlapping signal and noise distributions with a threshold that attempts to maximize the probability of detection while minimizing the probability of false alarm

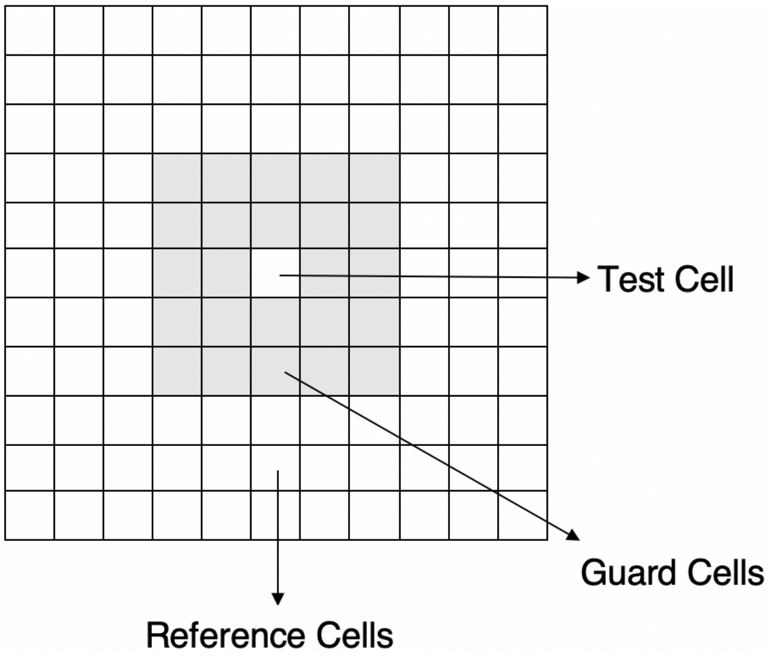
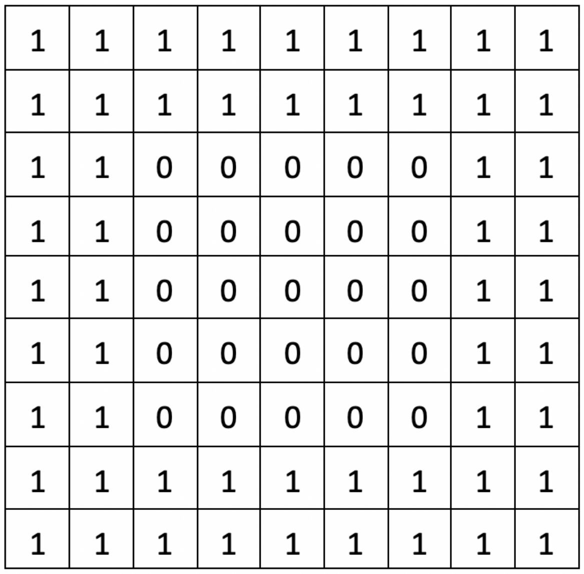


Figure 2-2: CFAR window where the cell under test (CUT) is the cell in question, the reference cells are the cells used to estimate the expected noise at the CUT, and guard cells are ignored.

2.1 CFAR Detectors

The original system addressed the concerns that result from naive thresholding by convolving the data in range-Doppler space using the kernel shown in Figure 2-3. The center cell is the cell under test (CUT), and the surrounding zeros in the kernel mask any information of the target that may have bled into the surrounding range and Doppler cells, they are referred to as guard cells and the remaining cells are the window's reference cells 2-2. The result of this convolution is an estimate of the expected noise present near each cell. The test cell is subsequently divided by this noise value, and the result is used to calculate the signal-to-noise ratio (SNR) using Equation 2.1.

$$SNR = 10 \log_{10} \frac{signal}{noise} \tag{2.1}$$



1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	0	0	0	0	0	1	1
1	1	0	0	0	0	0	1	1
1	1	0	0	0	0	0	1	1
1	1	0	0	0	0	0	1	1
1	1	0	0	0	0	0	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1

Figure 2-3: Detection kernel to find targets within the window

If the SNR exceeds 13dB, then there is a high likelihood that the cell contains a target. For each of the 11 beams at each time step, the previously described detection algorithm is applied, resulting in 11 logical matrices that correspond to where detections lie in range and Doppler for each time step. The 11 logical matrices are then superimposed to obtain a matrix where each cell in the matrix represents whether a target exists there, in at least one of the 11 range-Doppler maps. The logical matrix is then multiplied by a range-Doppler matrix that is the maximum power across the 11 beams, and the regional maximum of the resulting matrix is calculated to find the final detections.

While the original system has done an exceptional job in detecting targets in beamspace, it is vital to achieve a constant false alarm rate (CFAR) in radar systems to prevent too many signals from being falsely labeled as targets and to improve the probability of target detection. In this case, clutter and interfering targets in the data have yet to be addressed in the detection algorithm. The most common approach to address these concerns in radar detection has been through the use of the CFAR cell averaging and censoring algorithms [3].

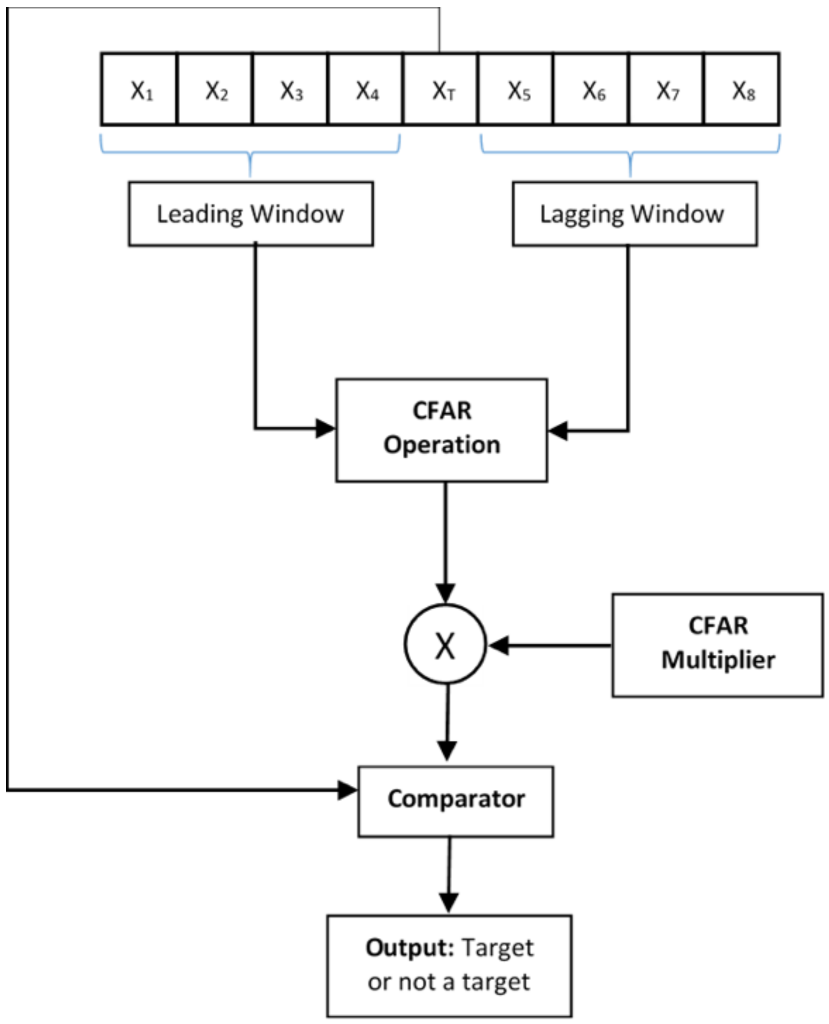


Figure 2-4: CFAR diagram

In this section, we will discuss five different CFAR algorithms that were implemented in the system and analyze their performances. All CFAR detectors follow a simple structure, as shown in Figure 2-4.

2.1.1 Cell-Averaging CFAR (CA-CFAR)

Finn and Johnson introduced the simplest CFAR algorithm in 1968 called Cell-Averaging CFAR (or CA-CFAR) [3]. It finds the average of the reference cells and multiplies it by a constant multiplier, T (2.2), where P_{FA} is the probability of false alarm.

$$T = P_{FA}^{-1/N} - 1 \tag{2.2}$$

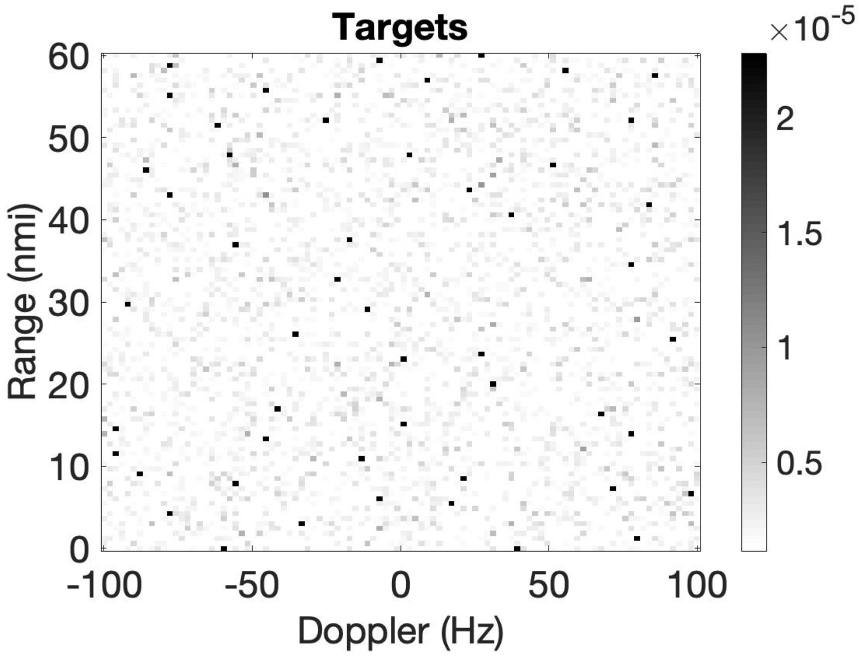
The Cell-Averaging CFAR detector is most effective in detecting targets in range-Doppler space that exists within one range and Doppler cell and homogeneous clutter; it fails in multiple-target situations and detections that exist in non-homogeneous clutter. In multiple-target situations, there may exist another target within the reference cells other than the target in the CUT. Higher than expected values in the CUT will increase the average of the reference cells and therefore, increase the threshold for the CUT to be considered a target. The CUT might be incorrectly classified as noise if the average becomes too high as a result of other targets that exist within the reference cells. Figure 2-5 displays 50 targets in a single-target geometry (Figure 2-5a) and a multiple target geometry (Figure 2-5b). Figure 2-6 shows how targets can be masked by using the Cell-Averaging detector in the system.

2.1.2 GO-CFAR Detector

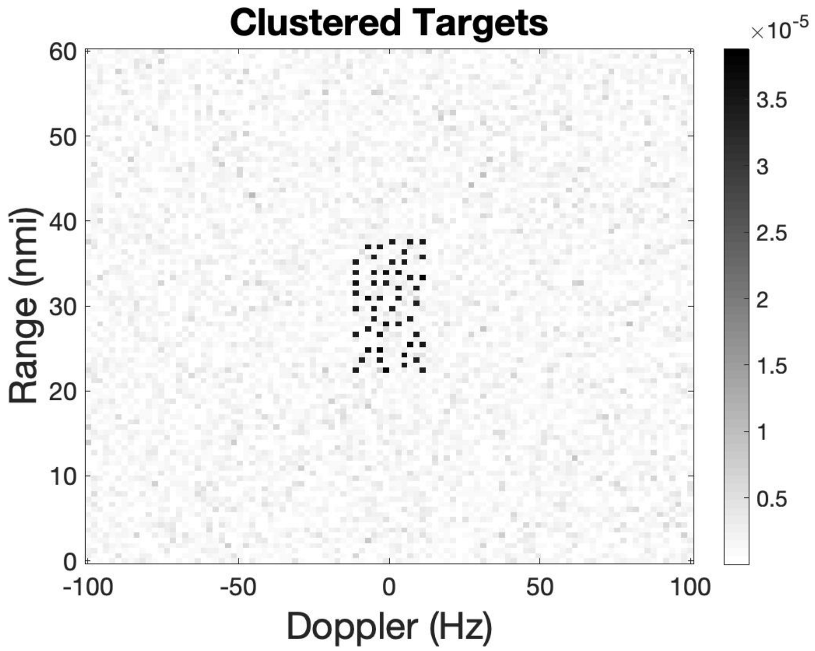
Greatest-Of CFAR, or GO-CFAR, was introduced by V.G. Hansen in 1973 [4][5], and it introduced a new feature in the CFAR windows; the window is subdivided into a leading window and a lagging window (Figure 2-7). The sum of the signals in the leading window is compared to that in the lagging window. Whichever window contains the more significant sum has the CFAR operation applied to it. It then calculates the threshold multiplier, T, using Equation 2.3.

The GO-CFAR detector faces the same issues present in the CA-CFAR detector if both the leading and lagging windows contain multiple targets. Figure 2-8 shows how targets can be masked by using the GO-CFAR detector in the system.

$$P_{FA} = 2(1+T)^{-M} - 2\Sigma_{k=0}^{M-1} \binom{M+k-1}{k} (2+T)^{-(M+k)} \tag{2.3}$$

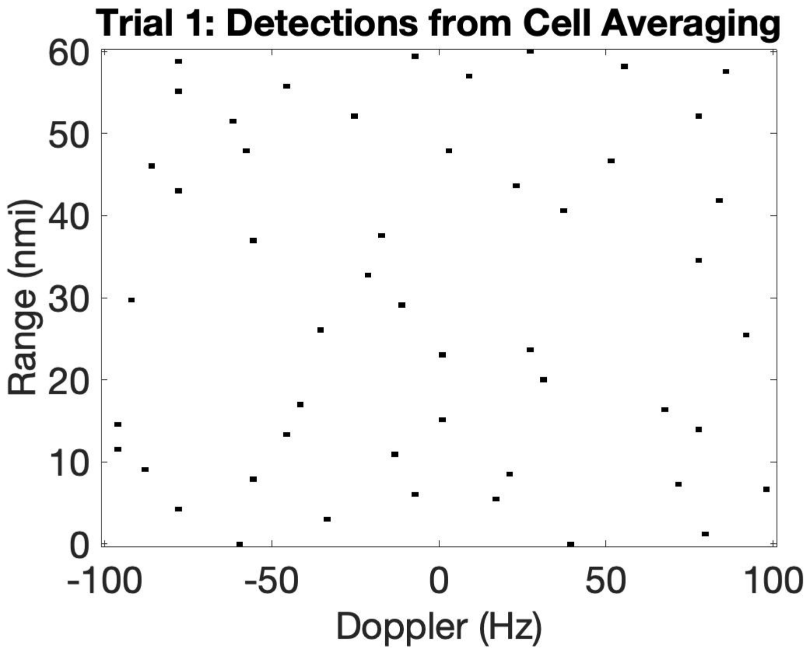


(a)

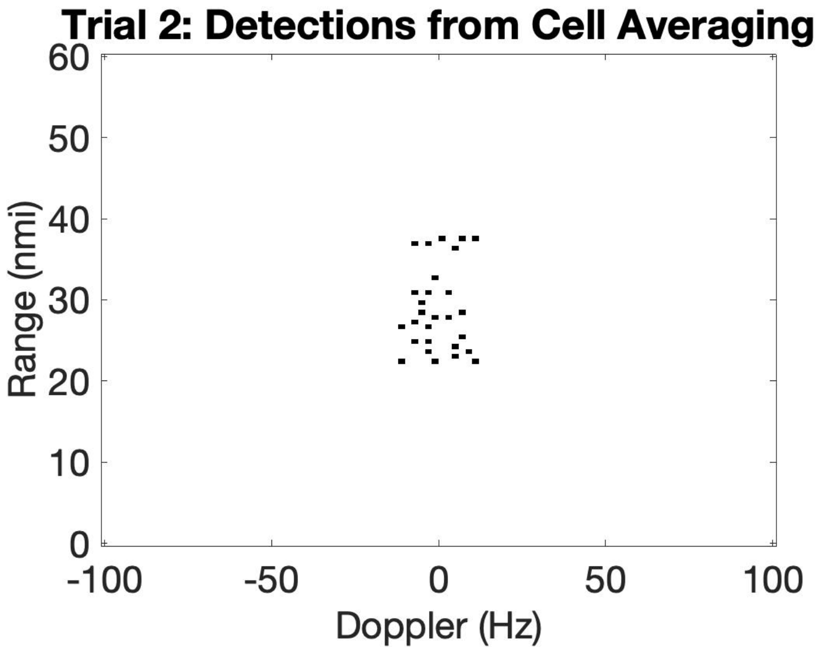


(b)

Figure 2-5: 50 targets in (a) sparsely arranged in range and Doppler so that no more than one target exists in every set of reference cells, and (b) closely clustered together so that multiple targets exist within the reference cells



(a)



(b)

Figure 2-6: Performance of the CA detector in (a) sparse target situation (b) a clustered multi-target situation

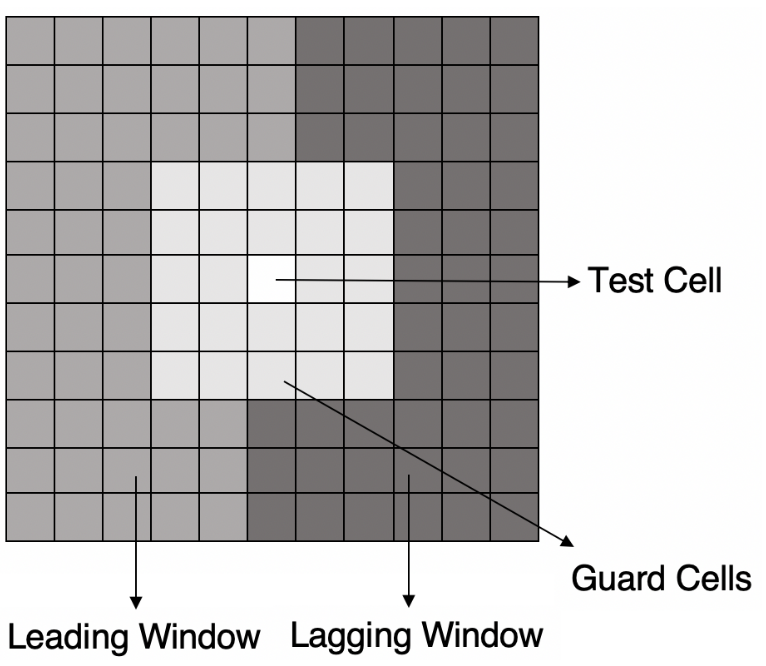


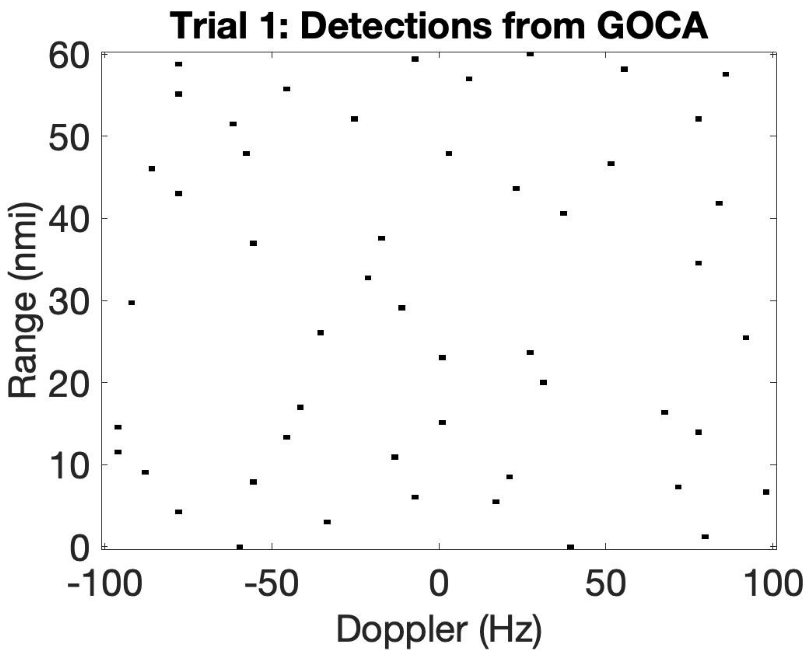
Figure 2-7: CFAR window where the leading window contains the reference cells to the left of the CUT and the lagging cells contain the reference cells to the right of the CUT

2.1.3 SO-CFAR Detector

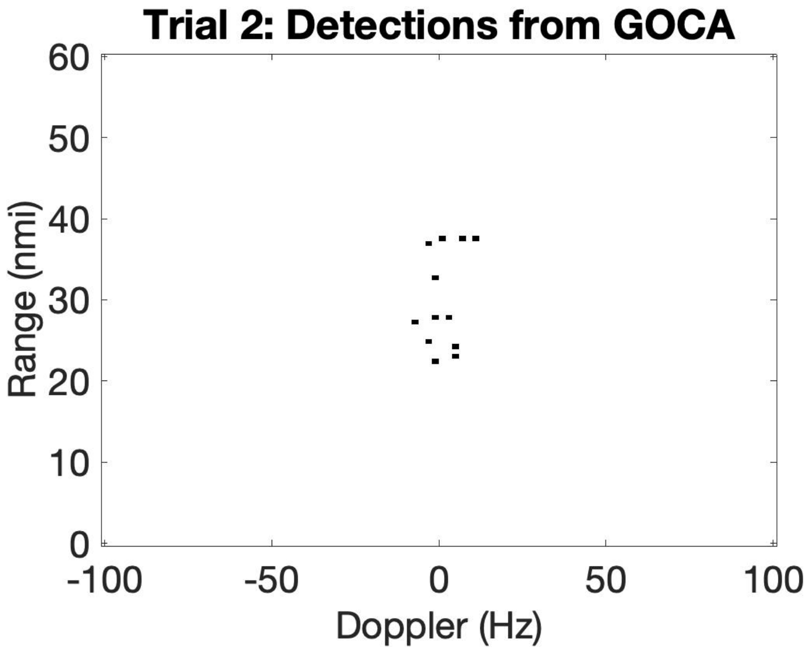
G. V. Trunk presented Smallest-Of CFAR, or (SO-CFAR), in 1978 [6]. It is similar to GO-CFAR except it performs the CFAR operation on whichever window (leading or lagging) contains the smaller sum. It calculates the threshold multiplier, T, by using Equation 2.4. It somewhat solves the issue of masking, as can be seen by 2-9, by choosing the smaller average of the two windows, decreasing the noise floor and increasing the probability that a window with multiple targets in its reference cells will be correctly classified. However, once again, if there were ever a case that both the leading and lagging windows contained multiple targets in its reference cells, there is a possibility for the CUT to be misclassified as a noise.

2-9

$$P_{FA} = 2(2+\frac{T}{M})^{-M}\Sigma_{k=0}^{M-1}\binom{M+k-1}{k}(2+\frac{T}{M})^{-k} \tag{2.4}$$

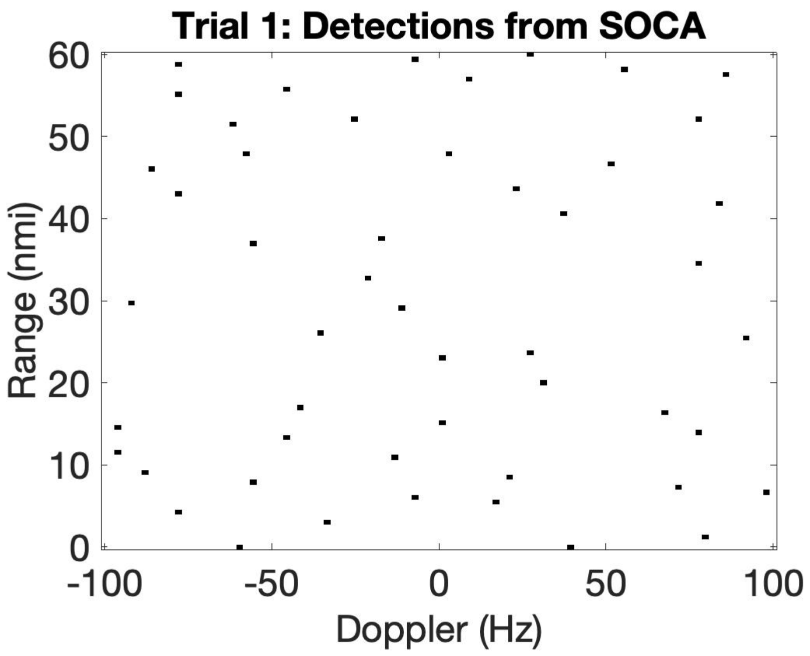


(a)

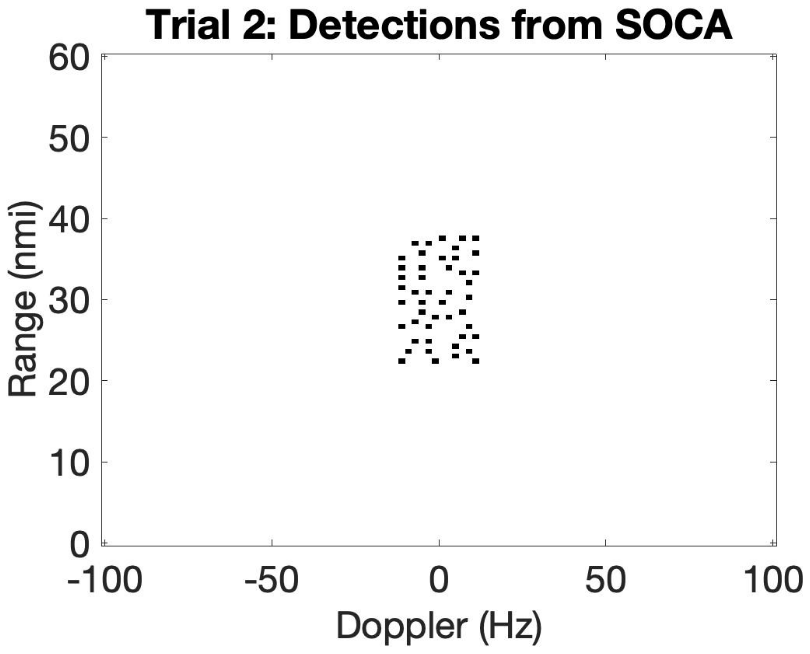


(b)

Figure 2-8: Performance of the GOCA detector in (a) sparse target situation (b) a clustered multi-target situation



(a)



(b)

Figure 2-9: Performance of the SOCA detector in (a) sparse target situation (b) a clustered multi-target situation

2.1.4 OS CFAR

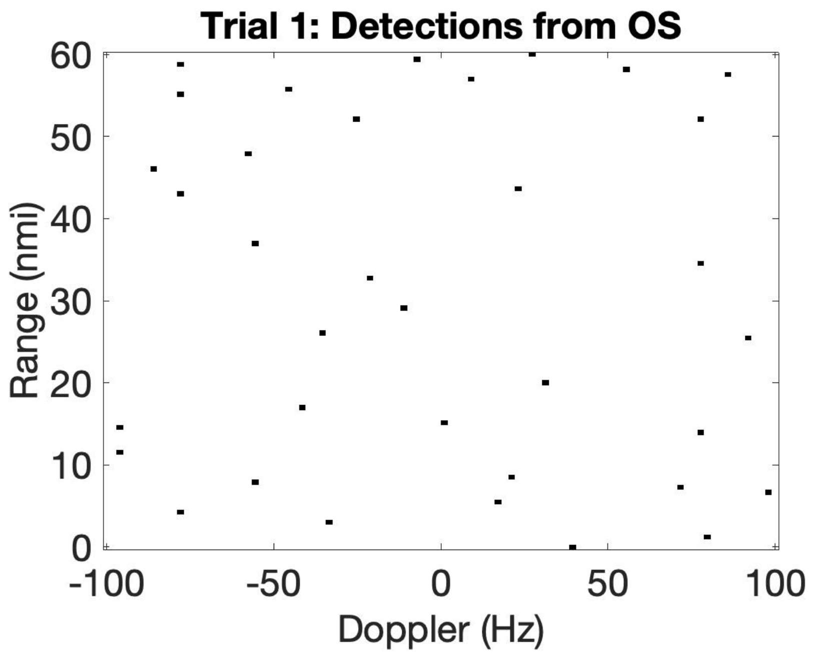
H. Rohling describes the Ordered-Statistic CFAR, or OS-CFAR, detector in 2011 [7]. It removes the k largest cells in the reference window and calculates the average of the remaining cells, where k is some predetermined constant. It calculates the threshold multiplier, T, using Equation 2.5.

$$P_{FA} = \Sigma_{i=0}^{k-1} \frac{N-i}{N-i+T} \tag{2.5}$$

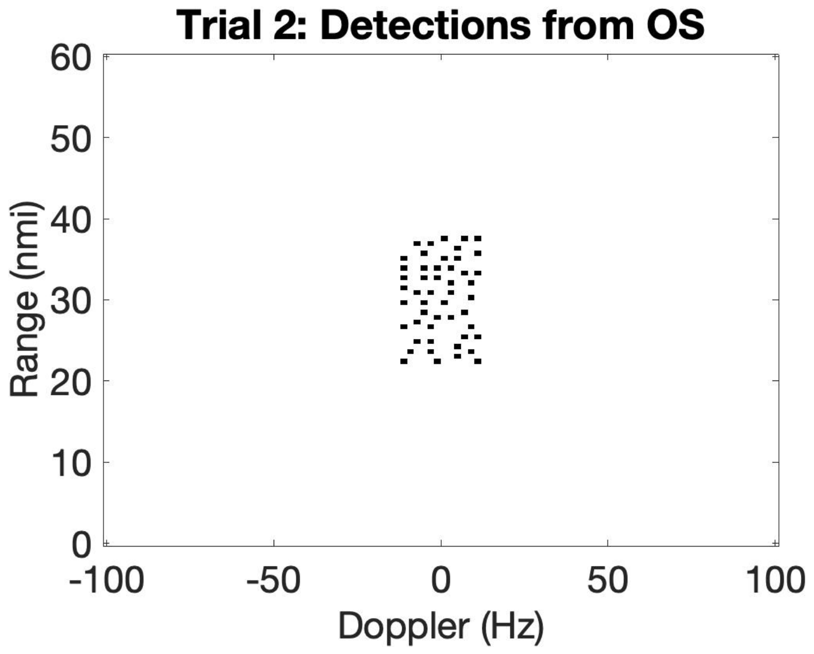
A k of 10 was used to generate Figure 2-10, where k was determined through trial and error. This was the optimal k in order to find all 50 detections in the range-Doppler map. Any choice of k lower than 10 would result in missed targets.

2.1.5 GCMLD-CFAR Detector

The GCMLD was introduced by S.D. Himonas and M. Barkat [8]. The GCMLD algorithm determines the number of interfering targets in the reference cells by iteratively censoring cells without any prior knowledge of the number of interfering targets in the window. It follows a slightly different structure than what is depicted in Figure 2-4. Instead, it sorts all of the reference cells into a list in ascending order. It assumes that the first cell in the list ($k = 1$) is the estimation for the background noise and compares its value to the product of the next cell value and some threshold multiplier, T_1, calculated through Equation 2.6. If the value of the product is larger than the first cell, then all cells after the first cell are assumed to be interfering targets. Otherwise, the new noise estimation is the sum of cells one and two. This new sum is compared to the product of cell three and some multiplier T_2. This process is repeated until the end of the list is reached, or some cell (and consequently all proceeding cells) are declared to be interfering targets and are censored. Given the number of censored cells in the window, m, it is now possible to perform regular CFAR detection in the window for the CUT using Equation 2.7. As can be seen by Figure 2-11, the GCMLD



(a)



(b)

Figure 2-10: Performance of the OS detector in (a) sparse target situation (b) a clustered multi-target situation

algorithm finds all fifty detections in both single and multi-target geometries.

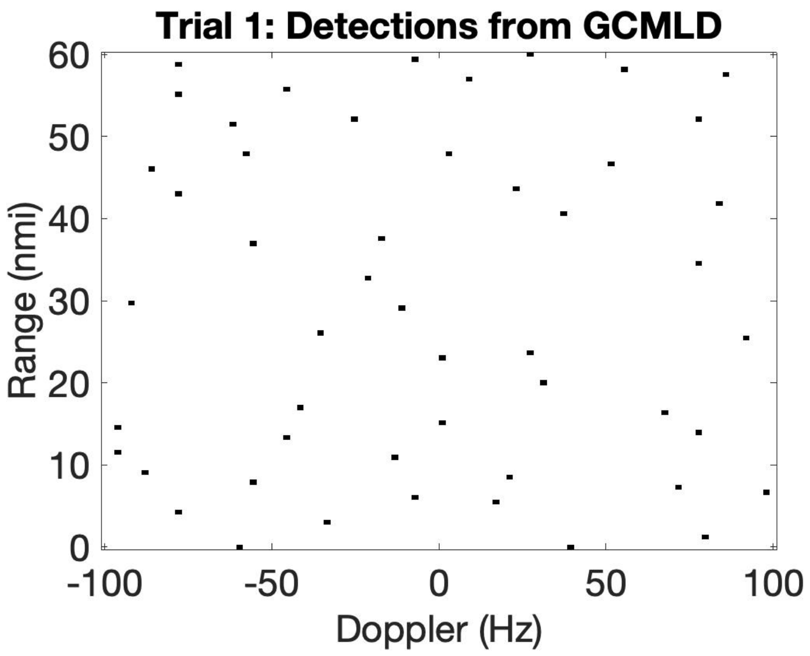
$$P_{FC} = \frac{M-k+1}{(1+T_{M-k})^{M-k}} \tag{2.6}$$

$$P_{FA} = \frac{1}{(1+T)^{N-m}} \tag{2.7}$$

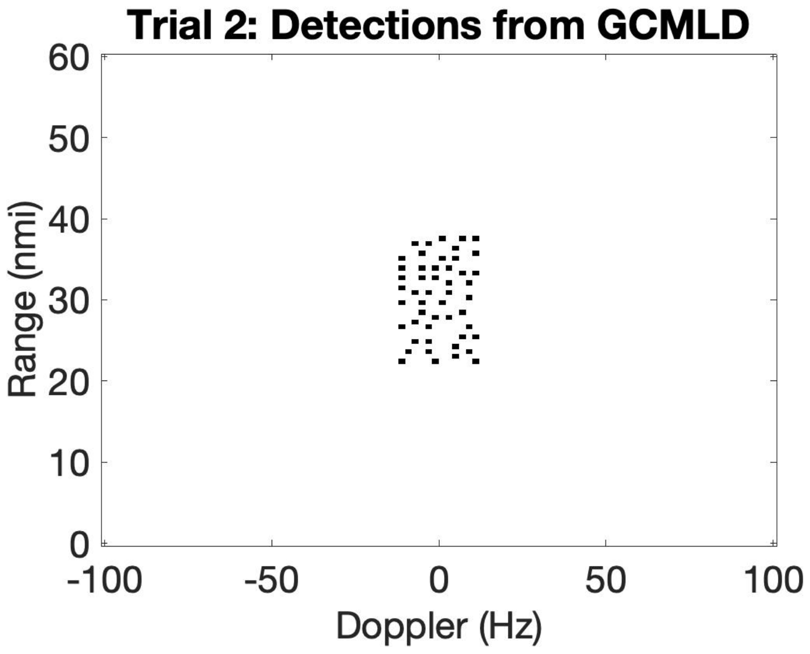
2.2 Results

From Table 2.1 it is apparent that in sparse-target situations or single-target situations, any CFAR detector, including the Cell-Averaging CFAR detector will be sufficient to locate any target that exists within a range-Doppler map. However, according to Table 2.2, none of the detectors, except for OS and GCMLD, found all 50 targets in a multiple-target setting.

OS CFAR is exceptionally efficient as it only requires sorting and calculating the threshold given the probability of false alarm. On the other hand, it also requires some prior knowledge of how the targets are clustered, which may not always be available. Otherwise, arbitrarily choosing some value for k may result in an increased probability of misclassified targets. GCMLD CFAR is significantly slower than OS CFAR but requires no prior knowledge on the targets. Therefore it was chosen to be the default algorithm for the radar signal processing tool.



(a)



(b)

Figure 2-11: Performance of the GCMLD detector in (a) sparse target situation (b) a clustered multi-target situation

CFAR Detection Analysis				
CFAR Algorithms	Targets	Found	Missed	False Alarm
CA	50	50	0	0
GOCA	50	50	0	0
SOCA	50	50	0	0
OS	50	50	0	0
GCMLD	50	50	0	0

Table 2.1: Performance of the 5 CFAR algorithms in non-clustered geometries.

CFAR Detection Analysis				
CFAR Algorithms	Targets	Found	Missed	False Alarm
CA	50	28	22	0
GOCA	50	12	38	0
SOCA	50	47	3	0
OS	50	50	0	0
GCMLD	50	50	0	0

Table 2.2: Performance of the 5 CFAR algorithms in clustered geometries.

Chapter 3

Direction Finding

3.1 Monopulse Direction Finding

Currently, the system has been updated to use a variation of monopulse direction finding rather than Root-MUSIC in the interest of efficiency and accuracy. The application of Root-MUSIC on the system seems to result in a high root-mean-square error which we would like to improve through monopulse direction-finding. In the monopulse direction-finding algorithm, information about the detection amplitude across beams is used to determine the precise direction of a target given an initial guess by the detection's beam position.

The 11 beams are used as an initial estimate of where a detection exists in azimuth. Given 11 beams labeled as integers between -5 and 5, Figure 3-1 shows an example of how beam numbers map to angle values between -60 and 60 degrees. A target is detected by a beam if the signal passes through it. Most of the time, a target is not detected by a single beam. As can be seen in 3-2, the signal first passes through beam 2, then it passes through beam 1, and finally, it passes through beam 3. All three of these beams will contain some amount of target information. Since the signal first passes through beam 2, it is assumed that the detection has been found at an angle of about 24 degrees.

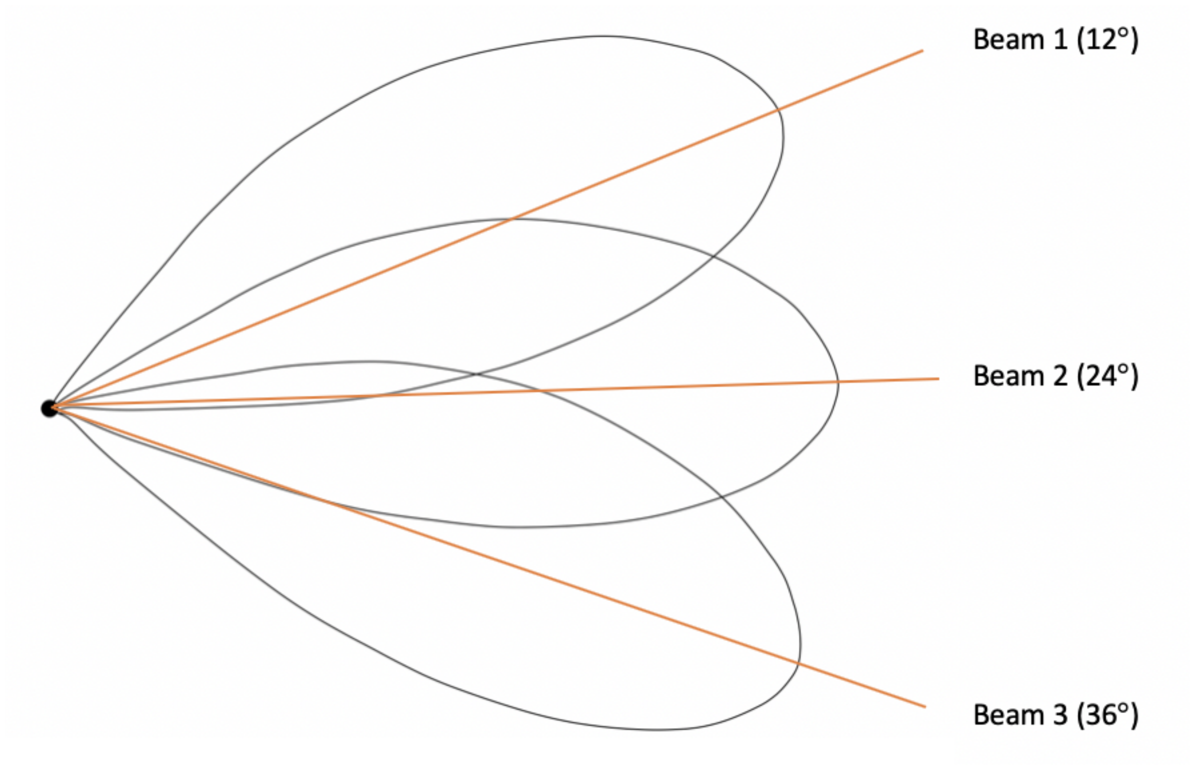


Figure 3-1: Diagram of how beam position corresponds with angle

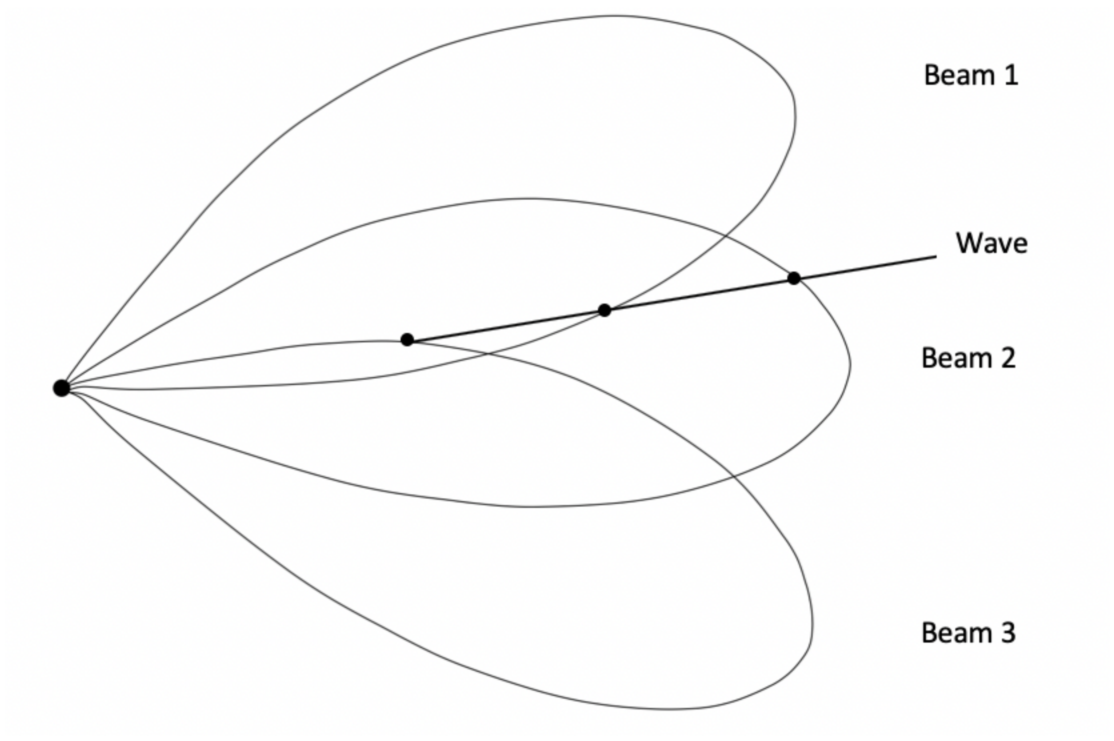


Figure 3-2: Example of propagation wave passing through 3 beams

The point of intersection between a beam and a signal is related to the power level of the detection found in that beam. From Figure 3-2 it can be concluded that the power level of a target (at a specific range and Doppler cell) found by beam 2 is significantly higher than the power level for that same detection found by beam 1, and similarly for beam 3.

An estimate for the actual direction angle can be made by extracting detection power levels from beams at either side of the main beam where the detection is thought to exist in (in this case, beam 2). With these three data values it is possible to make a polynomial fit of the data to map power levels to detection angle.

We then extrapolate data points from the polynomial that results. For an accurate measurement, we extrapolate 100 points from the polynomial. This is similar to placing 100 smaller evenly-spaced points between beams 1 and 3, each representing a .24 degree difference in angle. From the extrapolated points, we find the point with the smallest difference between itself and the highest power level among the three original data points. The location of the point with the minimum difference in the list of 100 points gives an estimate of where the detection angle lies. For example, if point 40 gave the smallest difference, then the new detection angle is 24 - 2.4 degrees, or 21.6 degrees. As will be further discussed in the conclusion, this method reduced root-mean-square error in the detection angle by 1.48deg.

Chapter 4

Multi-Target Tracking With Kalman Filtering

The tracking algorithm helps to eliminate false alarms that the CFAR detector and filtering algorithms introduce into the system. It eliminates the number of false alarms in the system by deleting detections that are more likely to be noise rather than targets. This is done by calculating the expected motion of the detections, as opposed to the power levels of the detections. Running the extended Kalman filter [9] on the detections found at each time step should also reduce the error in parameter estimation by reducing the noise introduced into the system by its sensors. The following sections will discuss the tracking algorithm and discuss the performance of its two subparts: the track association algorithm and the parameter-estimation algorithm.

4.1 Nearest Neighbor Association Groups

In single target situations - assuming that each target exists in one range-Doppler cell - all detections made at each time step should be a result of either a different target detected or a false alarm.

After the first time step and in all subsequent time steps, there is some understanding of where a target should be in the next time step given its previous position in range, Doppler, and azimuth. For example, if a single detection was found at a range of 80km while all previous detections were found at ranges between 10 and 20km, it can be assumed with high probability that the detection found at 80km is either a false alarm - where eliminating it reduces the overall false alarm rate of the radar signal processing tool - or the detection of a new target.

Since the goal of the radar signal processing tool is to detect and track multiple targets, it is essential to place the detections into track groups, where each group is a list that represents a potential target detected by the tool and its detected movement over time.

One challenge in creating these track groups lies in the fact that a single target may exist in more than one range-Doppler cell. If detected by a conventional CFAR detector, it may result in the target being represented by several detections at each time step. This poses the challenge of determining which detections belong to which targets at each time step as to not create a new track for a target that already exists.

As a result of the detection filtering methods described in Chapters 1 and 2, it is safe to assume that each detection made at a specific time step represents the detection of a different target. If the radar processing script finds n detections at time step 1, then it initializes n tracks for the n potential objects. Looping through each subsequent time step, t, the processing script extracts all of the detections at that time step using the methods described in Chapter 2. Then, for each track, it performs a gating test on every detection found at that time step. This gating test finds all detections that may be associated with a track by eliminating any detection that is too distant in range, Doppler, or azimuth values with the last associated detection in the track. After gating, each track is left with a set of possible detections it may be grouped with. All detections that do not exist in any of the sets are placed in new tracks.

Finally, the 1-Nearest Neighbors algorithm is applied on the remaining detections to associate detections to each track [10]. If there are fewer detections than tracks, then those tracks that were not updated are recorded. If there are more detections than tracks, then the remaining detections after 1-Nearest Neighbors are placed into new tracks. If any track exceeds five-time steps where it is not updated, then the track is deleted, and those detections are considered false alarms. The pseudocode for the association algorithm is given in Algorithm 1.

Algorithm 1 Association algorithm for radar tracking

```
1: procedure AssociateGroups($DetectionData, RG, DG, AG$)
2:     Let $Tracks$ be an struct array of length $n$
3:     $DetData\_true \leftarrow DetectionData$
4:     for $t = 2$ to length of $DetectionData$ do
5:         Let $passed\_detections$ be an empty set
6:         for $track$ in $Tracks$ do
7:             for $det$ in $DetectionData(t)$ do
8:                 if $t - track(end).t > 5$ then
9:                     Delete detections in $track$ from $DetData_true$
10:                    Delete $track$
11:                else
12:                    if $track.isUpdated$ is $false$ then
13:                        $passed\_rng \leftarrow |det.range - track.range| < RG$
14:                        $passed\_dop \leftarrow |det.doppler - track.doppler| < DG$
15:                        $passed\_az \leftarrow |det.azimuth - track.azimuth| < AG$
16:                        if $passed\_rng$ and $passed\_dop$ and $passed\_az$ then
17:                            add $det$ to $passed\_detections$
18:            Perform $1 - NearestNeighbors$ on $Tracks$ and $passed\_detections$
19:            Create new tracks for unassociated detections in $passed\_detections$
```

4.2 Parameter Estimation and the Extended Kalman Filter

Given the list of the tracks received from the method described above, the extended Kalman Filter is applied to each track for better parameter-estimation in the form of

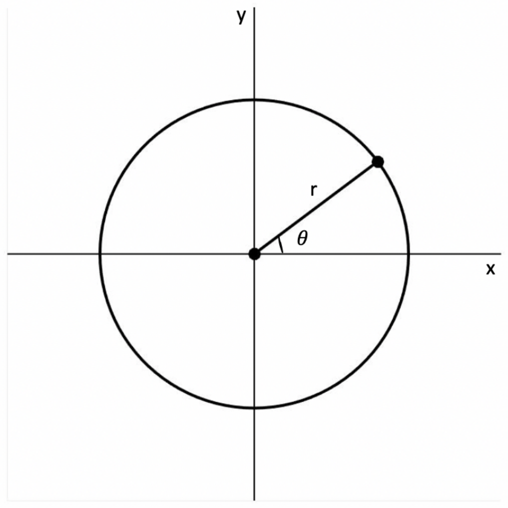


Figure 4-1: A propagation wave extended to range r with azimuth θ represented in an xy-plane

noise reduction. This project used an implementation of the extended Kalman Filter as opposed to the regular Kalman Filter [11] because the system is nonlinear. The Kalman Filter cannot be applied to nonlinear or non-Gaussian systems.

The extended Kalman Filter allows the system to be nonlinear so long as the model is differentiable and it works by transforming the nonlinear model at each time step into a linearized approximation through the use of the Jacobian of the model.

After the system has been linearized, an algorithm very similar to Kalman Filtering is applied to get the estimated state. In this case, the system makes measurements in terms of range (r), azimuth (θ), and Doppler ($\dot{r}$). The Extended Kalman Filter receives these data values and obtains a linear approximation of the state of each detection in terms of x-position, x-velocity, y-position, and y-velocity, where the xy-plane is ground position relative to the antennas.

Figure 4-1 and Equations 4.1 and 4.2 show how x- and y-positions are calculated using the range and azimuth observations. The positions and velocities are stored in a vector, $\hat{x}_i$.

$$x_i = r_i \cos\theta_i \tag{4.1}$$

$$y_i = r_i \sin\theta_i \tag{4.2}$$

In Kalman Filtering, there is the prediction step and the update step. In the prediction step, the prediction for $\hat{x}_i$, as shown in Equation 4.3, is calculated using the state estimation for the previous time step, x_k. For the system current model, the prediction for $\hat{x}_i$ is calculated using simple kinematic equations as shown in Equations 4.4 and 4.5. The kinematic equations are represented as the state-transition matrix, F (Equation 4.6), and which is then used in Equations 4.3 - 4.8 to calculate the state prediction.

Also in the prediction step, the prediction for the error of the state at time step i is represented as the covariance matrix, P, calculated using Equation 4.9, where Q is the covariance matrix of the process noise.

$$\hat{x}_i = \begin{bmatrix} x_i \\ \dot{x}_i \\ y_i \\ \dot{y}_i \end{bmatrix} \tag{4.3}$$

$$x_i = x_{i-1} + \dot{x}_i \Delta t \tag{4.4}$$

$$y_i = y_{i-1} + \dot{y}_{i-1} \Delta t \tag{4.5}$$

$$F = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.6}$$

$$\hat{x}_i = F\hat{x}_{i-1} \tag{4.7}$$

$$\hat{x}_i = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \hat{x}_{i-1} \tag{4.8}$$

$$P = FPF^T + Q \tag{4.9}$$

The next step in Kalman Filtering is the update step where the observed measurement is used to update the prediction made during the prediction step. The observation vector, given by z, is the measurement in range, azimuth, and Doppler received by the system's sensors, and it is of the for 4.10.

The next step requires the calculation of the observation matrix, H, which is the Jacobian of the state-transition model. It maps the state, x_i to its equivalent in polar coordinates as can be seen in Equation 4.11. The values of the Jacobian matrix, H can be calculated using Equations 4.12 - 4.14. The resultant matrix is given in Equation 4.15.

$$\begin{aligned} \hat{z} &= \begin{bmatrix} \hat{r} \\ \hat{\theta} \\ \hat{dop} \end{bmatrix} \\ &= \begin{bmatrix} \hat{r} \\ \hat{\theta} \\ \hat{\dot{r}} \end{bmatrix} \end{aligned} \tag{4.10}$$

$$H_i x_i = \begin{bmatrix} \hat{r} \\ \hat{\theta} \\ \hat{\dot{r}} \end{bmatrix} = \begin{bmatrix} \sqrt{x_i^2 + y_i^2} \\ \arctan \frac{y_i}{x_i} \\ \frac{x\dot{x}+y\dot{y}}{\sqrt{x^2+y^2}} \end{bmatrix} \tag{4.11}$$

$$\hat{r} = \sqrt{\hat{x}^2(1) + \hat{x}^2(3)} \tag{4.12}$$

$$\hat{\theta} = \arctan \frac{\hat{x}(3)}{\hat{x}(1)} \tag{4.13}$$

$$H = \begin{bmatrix} \frac{\partial r}{\partial x} & \frac{\partial r}{\partial \dot{x}} & \frac{\partial r}{\partial y} & \frac{\partial r}{\partial \dot{y}} \\ \frac{\partial \theta}{\partial x} & \frac{\partial \theta}{\partial \dot{x}} & \frac{\partial \theta}{\partial y} & \frac{\partial \theta}{\partial \dot{y}} \\ \frac{\partial \dot{r}}{\partial x} & \frac{\partial \dot{r}}{\partial \dot{x}} & \frac{\partial \dot{r}}{\partial y} & \frac{\partial \dot{r}}{\partial \dot{x}} \end{bmatrix} \tag{4.14}$$

$$= \begin{bmatrix} \frac{x}{\sqrt{x^2+y^2}} & 0 & \frac{x}{\sqrt{x^2+y^2}} & 0 \\ -\frac{x}{x^2+y^2} & 0 & \frac{x}{x^2+y^2} & 0 \\ \frac{y(y\dot{x}-x\dot{y})}{(x^2+y^2)^{3/2}} & \frac{x}{\sqrt{x^2+y^2}} & \frac{x(x\dot{x}-y\dot{y})}{(x^2+y^2)^{3/2}} & \frac{y}{\sqrt{x^2+y^2}} \end{bmatrix} \tag{4.15}$$

Now that the observation matrix has been calculated, the remaining steps in extended Kalman are the same as those in the regular Kalman Filter. Calculate the covariance estimate of the total error (S), the Kalman gain (K), the residual between the observed measurement and the predicted measurement (calculated as Hx), then update the estimate for the state at time step i using the Kalman gain, residual, and the original estimate for the state calculated in the prediction step. Finally, update the error covariance P. These steps are outlined in Equations 4.16 - 4.20.

$$S = H_i P_i H_i^T + R \tag{4.16}$$

$$K = P_i H_i S^{-1} \tag{4.17}$$

$$residual_i = z_i - H_i x_i \tag{4.18}$$

$$\hat{x_i} = \hat{x_i} + K * residuals \tag{4.19}$$

$$P = (I - KH)P \tag{4.20}$$

4.3 Performance

In conclusion, the association algorithm helps to eliminate false alarms by deleting detections whose projected motion over time does not fit with the detections found by the system. The tracking algorithm was expected to reduce the noise in the estimations of range, Doppler, and azimuth through the use of an extended Kalman Filter; which transforms the nonlinear model of the system into a series of linear equations through the use of the Jacobian of the model. However, with the use of the extended Kalman Filter, it was found that the estimations of the parameters tended to diverge and was not consistent with the data found by the detection algorithm.

The weak performance of the tracking algorithm could be a result of one of several failings of the extended Kalman Filter. The first reason lies in the fact that the extended Kalman Filter is not an optimal estimator if the system is nonlinear, as is the case with our system. Secondly, the poor performance of the Extended-Kalman Filter may be due to poor initial estimations of the model. In other words, because the model uses Jacobians to linearize the system, poor model initialization may cause the extended Kalman Filter to diverge. Finally, in extended Kalman Filtering, the estimated covariance matrix is often an underestimate of its real value. Therefore, the system runs the risk of becoming inconsistent without the addition of "stabilising noise" [12].

Chapter 5

Conclusion

In summary, I improved a radar processing tool by implementing and benchmarking several CFAR algorithms and settling on GCMLD as the detector with the best performance. I implemented detection filtering techniques to reduce the overall size of the detection data. This reduced the size of the data that needs to undergo further processing by the system while maintaining enough target information that the overall accuracy of the system remains unaffected. Then, from the set of filtered detections, I used a simplified direction finding technique to calculate detection angles and improve the efficiency of the processing script processing tool. Finally, I implemented a detection tracker using 1-Nearest Neighbors to group the detections into target tracks based on the range, Doppler, and azimuth information received from the previous section. Then, I applied the Extended Kalman Filter to these tracks in an attempt to reduce noise in parameter estimation.

5.1 System Performance

In an effort to test the accuracy of the processing script, I relied on another MATLAB script designed to test the accuracy of detection data given the truth data. Essentially, this script plots the truth data and calculates the mean error and the root-mean-square error between the truth data and the data collected from the processing script in range, Doppler, and azimuth. This then creates a short movie of the progression of the detections on a grid in range and azimuth. I improved the script in two major ways. One, the script only had truth data from a single object. In order to prevent over-fitting, I included truth data from an additional 49 objects - all of which varied in both shape and size for a total of 50 objects. The addition of truth data helped to create a more accurate tool to measure the performance of the radar signal processing tool. Two, the truth data for Doppler velocity for all 50 objects was nonexistent because of an error present during data collection. Doppler velocity can accurately be measured by the difference in range over time given by Equation 5.1. Therefore, we can use both the range truth data and the sampling times for the truth data to estimate Doppler velocity.

$$v_{i+1} = \frac{r_{i+1} - r_i}{t_{t+1} - t_i} \tag{5.1}$$

One problem with this method is that the truth data for detections were not sampling consistently. Gaps between sampling times could range anywhere between 0.03 seconds to 15 seconds. This is a problem because significant gaps in sampling times can cause large leaps in Doppler frequency from one-time instance to another and a visibly non-smooth curve which causes detection to be misclassified as a false alarm. Unfortunately, this leads to less accurate measurements for Doppler frequency.

In order to address this problem, it is essential to make the sampling times small and consistent. The issue was solved by using a Gaussian Process to estimate the range at a range of 10 per second and then using these new sampling times and estimated range values to calculate the truth doppler velocity. With these improvements to the accuracy tool, we can now analyze the performance of the new processing tool in comparison to the original.

From Table 5.1, it is apparent that the error in azimuth experienced the most significant improvement in the new radar signal processing code. There was an overall

Error Analysis		
	Original RMS Error	Updated RMS (Mean) Error
Range (km)	0.39	0.31
Doppler (m/s)	4.11	3.46
Azimuth (deg)	3.63	2.15

Table 5.1: The the rms error analysis of the new radar signal processing tool in comparison to the original in range, doppler, and azimuth.

reduction of root-mean-square error by nearly 41% (1.48deg). There was an improvement in the Doppler velocity by 16% (0.65 m/s) as well as a reduction in the range root-mean-square error by 20% (0.08 km).

The root-mean-square error in Doppler velocity also improved by 16% (0.65 m/s) and the root-mean-square error in the range reduced by 20% (0.08 km).

After making the previously described changes, it was clear that there was a significant improvement in the overall runtime of the script. Initially, the script took an average of 3782 seconds (over an hour) to process 5 minutes of radar data. However, now the script runs in approximately 1949 seconds (around half an hour). This change cut the runtime by nearly half. This project was able to cut the overall runtime of the script in half by parallelizing the code and downsampling the data as opposed to decimating it.

The parallelization of the code produced the most substantial speedup, particularly in the target detection portion of the code. Nearly 80% of the runtime was used in detection finding alone. Parallelizing this section of the code initially improved the runtime of the code by a factor 9 yet, since the detector has been changed to GCMLD, a much less efficient algorithm than the detector in the original system, the overall runtime of the script saw a less impressive improvement in speed.

Downsampling the data rather than decimating it produced the next largest speedup. As explained in Chapter 1, decimation passes the data through a high pass filter. This is computationally intensive. It removes the effect of antialiasing.

On the other hand, downsampling with a rate of k takes every k^{th} data point. Since we can assume that the data has a high enough sampling frequency compared to its bandwidth that aliasing is avoided, downsampling is much more efficient than decimation.

5.2 Future Work

There are several different routes that this project may take in the future. One such route would be to improve the efficiency of the script. Currently, it takes over 30 minutes to process 5 minutes of data in the updated MATLAB script on a machine with 16 cores. That is over 6 times the amount of input data collected, 60% of which is spent beamforming and performing the detection algorithm at each time instance and each. This is because the detection algorithm has to iterate through each beam at each time instance, so for 300 time instances with 11 beams each, for example, that is 3300 calls to the GCMLD algorithm.

If the project should continue to be developed in MATLAB, there may still be some valuable improvements in runtime through MATLAB's Parallel Computing Toolbox. Currently, the code is parallelized through simple parallel for loops when performing data extraction, downsampling, and detection finding. However, there may be other areas in the code that may benefit from parallelization. Another such way to improve runtime is to develop in C++. Ultimately the goal would be to develop a radar signal processing tool that can detect and track objects in real-time using C++.

Another route this project may take in the future is using machine learning to improve detection techniques. This can be done through two different methods. The first method would train the detector against many different sets of truth data to develop a detector that would be able to recognize target features much more efficiently and just as accurately as a CFAR detector. The second method would be to create a continuously learning detector. This detector would learn while processing the data, in order to keep introducing new data sets and possible target features to the detector. However, this method would severely degrade the runtime of the detector, possibly making it impossible to design a real-time version using it.

Another route this project may take in the future is in the use of machine learning to improve the detection techniques. This can be done through two different methods. The first method would train the detector against many different sets of truth data to develop a detector that would be able to recognize target features much more efficiently and just as accurately as a CFAR detector. The second method would be to create a continuously learning detector. This detector would primarily learn while processing the data, to keep introducing new data sets and possible target features to the detector. This, however, would severely degrade the runtime of the detector, possibly making it impossible to design a real-time version using it.

Bibliography

[1] Russel Rzemien. Coherent radar: Guest editor's introduction. *JOHNS HOPKINS APL TECHNICAL DIGEST*, 18(3):344–347, 1997.

[2] X. Jing and Z. C. Du. An improved fast root-music algorithm for doa estimation. In *2012 International Conference on Image Analysis and Signal Processing*, pages 1–3, Nov 2012.

[3] H.M. Finn and R.S. Johnson. Adaptive detection mode with threshold control as a function of spacially sampled clutter level estimates. *RCA Review*, 29(3):414–464, 1968.

[4] V. G. Hansen. Constant false alarm rate processing in search radars. 1973.

[5] V. G. Hansen and J. H. Sawyers. Detectability loss due to "greatest of" selection in a cell-averaging cfar. *IEEE Transactions on Aerospace and Electronic Systems*, AES-16(1):115–118, Jan 1980.

[6] G. V. Trunk. Range resolution of targets using automatic detectors. *IEEE Transactions on Aerospace and Electronic Systems*, AES-14(5):750–755, Sep. 1978.

[7] H. Rohling. Ordered statistic cfar technique - an overview. In *2011 12th International Radar Symposium (IRS)*, pages 631–638, Sep. 2011.

[8] M. Barkat, S. D. Himonas, and P. K. Varshney. Cfar detection for multiple target situations. *IEE Proceedings F - Radar and Signal Processing*, 136(5):193–209, Oct 1989.

[9] J. Shen, Y. Liu, S. Wang, and Z. Sun. Evaluation of unscented kalman filter and extended kalman filter for radar tracking data filtering. In *2014 European Modelling Symposium*, pages 190–194, Oct 2014.

[10] Behnam Banitalebi and Hamid Amiri. An improved nearest neighbor data association method for underwater multi-target tracking. 2008.

[11] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[12] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis. Analysis and improvement of the consistency of extended kalman filter based slam. In *2008 IEEE International Conference on Robotics and Automation*, pages 473–479, May 2008.