Performance Analysis Of Active and Passive Multi-Array Sonar Networks

Brent A. Gold

Thesis submitted to the Faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master of Science

In

Mechanical Engineering

APPROVED:

Dr. Michael J. Roan, Chair

Dr. Alessandro Toso Dr. Marty E. Johnson

December 5, 2007
Blacksburg, Virginia

Keywords: Multi-array networks, active sonar, passive sonar, array fusion

Brent A. Gold

Investigation of Simulated Performances Of Multi-Array Networks using Active and Passive Sonar

Brent A. Gold
Mechanical Engineering Department
Virginia Tech

Abstract

This work investigates the ideal distribution of sensors in networked arrays. MATLAB models these arrays and simulates the results these networks obtain using active and passive sonar. These results determine the ideal sensor placement for optimal parameter detection and estimation of targets.

This work's first part focuses on active sonar networks with a fixed number of sensors in a differing number of arrays. MATLAB simulates the data of these sensors taking into account the geometries and velocities of the arrays and targets, then estimates the parameters of the targets using an elliptical filter, a conventional beamformer, a matched filter and one of three fusion methods. This work compares the performance of each network and fusion method. This work shows that the adding more arrays, regardless of size, enhances the overall performance of the network. It also shows the larger arrays obtain more robust parameter estimation.

The second part of this work investigates the effects of uncertainty of the array position and orientation using passive sonar. Two networks, one with 2 32-channel arrays and one with 8 2-channel arrays, estimate a sound source's location using a conventional beamformer. MATLAB simulates the data taking into account the geometries of the arrays and the sound source. The results of these simulations show that when uncertainty of position and orientation increases, the better the smaller arrays estimate the location of the sound source compared to the larger arrays.

Acknowledgements

There are so many people I want to thank for allowing me to chase my dream of earning an advanced degree from a premier research institution. The first people I want to thank are the people who work to make Virginia Tech an outstanding institution of higher learning. Without their hard work and dedication, this university would not be one of the leaders in research as it is today.

Among those people are the professors that I have had the honor of working with them. They include my adviser, Dr. Roan, whose patience and hard work made this research a pleasant working experience that I have enjoyed. Also included are the other two members of my committee, James Carneal and Marty Johnson, who have aided my research as I have needed direction.

I would also like to thank the other students in the Vibrations and Acoustics Lab, with whom I have become good friends. Their cheerful attitude made coming into the lab a good time, and their willingness to help me out when I needed it was welcomed. In particular, I would like to thank Elizabeth Hoppe, Phillip Gillett, Mark Sumner, Ben Smith, Dan Domme and Caroline Hutcheson for all the laughs, help and insight they have shared with me over the two years I have been here.

One of the main reasons I came to Virginia Tech instead of other schools was the campus ministry that is in place here. I had high expectations from the Ambassadors for Christ, and they have not disappointed me in any way. I feel honored to have been their president over the past three semesters. Without their support outside of research, my spiritual life would have been very much depleted, rendering everything else in my life useless. I give God all the credit for giving me the knowledge that will be displayed in this work.

Last but not least, I would like to thank my parents for supporting me during this time. It has been hard on them seeing their son so far away, and it's been hard on me to have them at this distance as well. Without their understanding and support for education over the years, there is no possible way I could have discovered what is in the pages to follow. Thank you, Mom and Dad.

Table of Contents

List of Figures

List of Tables

Nomenclature

AOA	Angle of Arrival
CRB	Cramer-Rao Bound
FFT	Fast Fourier Transform
MAN	Multi-Array Network
ROC	Receiver Operating Curve
SNR	Signal to Noise Ratio
$A(t)$	Data recorded by sensor
B	Gaussian curve value
b_{opt}	Optimal filter taps
c	Speed of sound in medium
d	Total distance traveled by sound source (m)
d_b	Distance between arrays (m)
f_c	Center frequency of emitted signal (Hz)
f_d	Doppler-shifted frequency (Hz)
f_s	Sampling frequency (Hz)
f_l , f_h	Lower and higher frequency of expected echo (Hz)
M	Maximum sufficient statistic for a target
m(t)	Emitted signal
N_w	Number of emitted waves
$n(t)$	Random noise component of signal
P	Power of signal (V*s)
Q	Sample number of the first simulated reading from the echo
R	Cross-correlation matrix
r	Sufficient statistic
S	Steering vector
s	Distance between sensors (m)
T	Threshold value
T_r	Reference threshold value
t	Time taken by emitted signal to reach target (s)
t_d	Doppler-shifted duration (s)

t_r	Reference time (s)
v	Relative velocity of target (m/s)
x	Distance from array to target
(x_a, y_a)	Assume location of array
(x_e, y_e)	Universal location estimate of target
(x_n, y_n)	Individual array's location estimate of target
(x_s, y_s)	Actual location of array
(x_{ss}, y_{ss})	Location of sound source
β	Bandwidth
$\sigma_{x,}\ \sigma_y$	Variance of array location (m)
σ_θ	Variance of array orientation
θ	Tested bearing angle
θ_a	Assumed array orientation
θ_b	Bearing angle to array
φ	Estimated velocity direction
τ_d	Time delay due to distance (s)
τ_θ	Time delay due to bearing angle (s)
τ_t	Total time delay (s)

1 Introduction

This chapter discusses the background and existing researching of networked arrays and active and passive detection of underwater targets. Key concepts, such as beamforming, matched filtering, and data fusion are introduced and discussed. This chapter also discusses previous works in the fields of networked arrays algorithms, followed by the motivation for this thesis. An outline of this thesis will conclude this chapter.

1.1 Background

1.1.1 Sonar Methods

Sonar can be performed in one of two ways, active and passive. In active sonar, a sound source, such as a transducer, emits a known signal into the surrounding medium. Sensors, such as hydrophones in aquatic environments, receive this signal after it reflects off of objects. By measuring the time delay between the signal's emission and its echo, a distance estimate can be calculated by multiplying the time delay with the speed of sound in the surrounding medium. By measuring the phase difference between the different elements in each array, a bearing estimate can be obtained by beamforming. If the object (and/or sensor) is moving, the received signal will undergo a transformation known as the Doppler Effect, which will change the frequency and duration of the echo due to the compression or expansion of the signal's waves. By comparing the returned echo to different Doppler shifted versions of the sent signal, the relative velocity of the object can be determined.

Passive sonar does not use a sent signal; rather it simply detects sound sources using the sound that they emit. Using beamforming and triangulation, the position of the source can be estimated. However, since there is no sent signal, a time delay estimate cannot be made, and the range of the target from each array cannot be determined by itself by the methods discussed in this work. Instead, multiple arrays must be employed with assumed positions and orientations. The position of the target can then be found using the intersection point of the determined bearing angle from two arrays. In the case of more then two arrays, the method used in this work was to take the median x and y coordinates of all possible intersection points.

1.1.2 Signal processing techniques

The main algorithm presented in this work utilizes several types of signal processing techniques. After MATLAB generates the raw data for each sensor based on the array's geometry and the location of the targets and arrays, the first operation performed is a simple band pass elliptical filter that severally reduced the noise outside the expected frequency range of an echo. Because of the possibility of a Doppler shifted version of the sent signal being recorded by the sensors, traditional white noise filters such as the LMS filter and the Wiener filter were not used since they would have cancelled out any variation of the returned signal as well as any white noise that the sensors recorded.

Next, after all the sensors have their data filtered, a beamformer determines the angle of arrival (AOA) on all arrays. Several types of beamforming have been studied and researched, such as MDVR and null-steering beamforming. In this work, a simple algorithm called delay-and-sum (also called conventional) beamforming determines the bearing estimate on all arrays in both active and passive simulations. The delay-and-sum beamformer uses a fast Fourier transform (FFT) to determine the phase angle for a potential AOA and the signal's center frequency. Once the raw data is multiplied by the steering vector that was created for a potential AOA in the frequency domain, the product undergoes an inverse fast Fourier transform (IFFT) to translate the array's data back into the time domain with one single output at each array for that particular AOA.

The output of the delay-and-sum beamformer is used in a process called matched filtering to determine the distance and perpendicular velocities of the targets when active sonar is used. Cross-correlating the beamformer output with a Doppler shifted version of the sent signal determines the distance and perpendicular velocity of potential targets relative to the arrays. The Doppler shifted version of the signal is both time-dilated and frequency-shifted for a given velocity in a given medium. The output of the matched filter is sufficient statistics for each velocity and time sample, and measures how well the time series data matches a particular Doppler-shifted version of the sent signal across the entire length of the time series. Without a sent reference signal, no matched filtering can be used with passive sonar since there is no way to estimate a time delay.

The sufficient statistics for all velocities and time samples are compared to a predetermined threshold to determine whether or not a target exists at a specific range traveling at a specific velocity. The sufficient statistics of all the arrays is fused together to make a

determination on the parameters of the targets using active sonar. In this work, three different fusion methods for position estimation are presented. The first transforms the matched filter outputs from all arrays into a common matched filter map and multiplies them together. The second uses a 2-D Gaussian curve centered at the point each array determines a target to be located. These curves are then multiplied together to determine the location of the objects. A third fusion method involves a weighted average, based on the output of the matched filter, and the parameters estimated from the matched filter and the AOA determined by the beamformer.

1.2 Previous work

In recent years, researchers have been given increasing attention to networked arrays. Late in the last century, the ability of sensor networks was being discovered, along with the potential applications they could perform[1]. Ideal parameter estimation has also been investigated in terms of the Cramer-Rao bound. In order to successfully fuse the data from all the sensors, new algorithms were being formulated and tested.

1.2.1 Applications of Networked Arrays

One discussed use for networked arrays was for disaster surveillance. In this scenario, several, perhaps thousands, of inexpensive sensors would be dropped into a disaster area in order to gather as much data as possible, thereby eliminating the need for humans to be in potentially dangerous situations. These sensors would then work together in order to record accurate data. Networked arrays were also being sought as solutions to other areas of interest, such as wireless Internet communications[2].

The Department of Defense has also placed importance on networked array research. One application of networked sonar arrays can be found in the AN/BSY-2 Submarine Combat System. The system uses sensor networks to control the sensors as well as track targets and weapons and gather data[3]. Three active US Navy submarines currently use the AN/BSY-2 Submarine Combat System, with the most recently deployed being the Jimmy Carter, which was commissioned in February 2005.

1.2.2 Parameter Estimation using Cramer-Rao Bounds

Parameter estimation from networked arrays has also been researched in the past several

years. Using Cramer-Rao bounds (CRB), the limit of performance of any array network can be found by using the algorithm used in [4]. Previous work has shown that the CRB will form an ellipse [5]. To find the CRB ellipse, the first calculation to be performed is the standard deviation of each direction. This is done by taking the square root of diagonal terms of the CRB matrix. For this 2-D location estimate, there are two terms, one for each direction. Once the standard deviations are known, the ellipse can be drawn, using equation (1) as specified in [6]:

$$\frac{1}{1-\rho^2}\left(\frac{(x-x_s)^2}{\sigma_x^2}-\frac{2\rho(x-x_s)(y-y_s)}{\sigma_x\sigma_y}+\frac{(y-y_s)^2}{\sigma_y^2}\right)=k^2 \qquad (1)$$

Using the fact that the correlation (ρ) between the x-coordinate and the y-coordinate is zero (ρ =0) simplifies equation (1) to

$$\frac{(x-x_s)^2}{\sigma_x^2}+\frac{(y-y_s)^2}{\sigma_y^2}=k^2 \qquad (2)$$

where (x_s, y_s) is the location of the source, σ_x and σ_y

are the determined standard deviations of the x and y coordinates of the target, and *k* is the number of standard deviations to be considered while drawing the ellipse.

If equation (2) is integrated over the area of the ellipse, a percentage of points that would fall inside of the ellipse can be calculated. As seen in [7], this number is:

$$1-e^{-k^2/2} \qquad (3)$$

Since the CRB uses one standard deviation in its calculation, the *x* and *y* terms in equations (2) and (3) can be set equal to one. This yields a *k* value of one, and $1\text{-}e^{-1/2}$, or 39.6%, of any set of data points would fall within the CRB ellipse.

The algorithm to find the CRB ellipse for an active sonar case with one target has been developed in [8]. In that work, the CRB is calculated for several array geometries to find the optimal placement of the arrays. It was found that the area in the CRB ellipse would be the smallest if the targets are located along the principal axes of inertia of an array.

An example of how the CRB can change with how the target is positioned can be seen in [9]. It was shown that target placement significantly changed the CRB while everything else remained constant. For example, if a target is collinear with two arrays, the CRB will be very large whereas a smaller CRB can be expected if the target is between the two arrays with

Doppler shifts having different signs with respect to the two arrays. It was also shown that multiple arrays have smaller CRBs then single-array networks.

The performance bounds could also be analyzed using fuzzy information. Instead of the sensors having definite parameters determining the returned signal, the detected parameters are vague. A minimum least squares method[10] then determines the best estimate for a target's parameters. It was also demonstrated how the fuzzy information can be used to create the Fisher information matrix, which is the inverse of the CRB. However, once a large number of sensors were involved, the method became computationally unfeasible.

Work has been done on the number of sensors required to achieve certain accuracy of an environment of a sensor field. The quality of the sensor network was based on a lower bound, similar to the Cramer-Rao bound that was introduced earlier in this work[11]. Using this bound, it was shown that the network would require fewer sensors if they were to work together instead of working separately while maintaining the same accuracy. It was also shown that adding more targets required more sensors to be added to the network in order to achieve the same performance.

1.2.3 Signal Processing Techniques and Data Fusion

Whenever data from multiple arrays is collected, there must be a way to gather the desired information from the recorded signal, and then fuse the data with the other arrays in the network. In [12], a bistatic towed array is simulated and then tested using active sonar with a space-time adaptive processing algorithm. The results of this method show that using the algorithm improves the output signal by approximately 4dB compared to more common signal processing techniques. This algorithm may be used in future work, as it offers better performance then the algorithm presented here, although the work offers no algorithms on parameter estimation.

Another algorithm using energy detection to track targets has been researched. Using this method, each sensor would make a positive detection decision if the signal power exceeded a threshold value and would send its parameters to a fusion center. Based on the data received from all sensors, an overall detection decision can be made, as well as an attempt to classify the target[13]. However, this method did not use a beamformer or matched filter, and instead used the energy decay of the signal to determine the location of the object. While these methods may

work when the SNR is great, lower SNRs can cause problems because the random component of the signal may exceed thresholds when there is no signal present, and a weak signal may not be enough for the signal to exceed the threshold. Additionally, interference from other objects could create problems for this method.

With the rise in use networked arrays came a need to effectively use all the sensors to make robust parameter estimation using various algorithms. Fusing the data from these sensors should lead to better performances as opposed to a decentralized approach where each sensor makes an estimation separately.

Research has been conducted on a sensor network comprised of 100 individual sensors in no arrays; each one is not grouped with any other sensors. In this network, the sensors are distributed with their positions given a variance as they track objects within the network's range[14]. Sensors are activated only when the object being tracked is within a certain distance due to signal attenuation with increased distance, thus saving power. Once all the sensors have reported their data, it gets fused by a central processor to obtain measurements on its velocity and position. This is similar to the research being done in this work; however the sensors are arranged in arrays here as opposed to single stand-alone sensors.

In [15], a position tracking algorithm using a maximum a posteriori penalty function (MAP-PF) tracked a moving vehicle using four sensor arrays and passive sonar. The research conducted in that work shows how the effective the MAP-PF can be as it tracks a vehicle very well over its traveled path. This can be very useful in tracking targets once they have been detected, however the focus of this work will be on the initial detection of targets.

The main thrust of the research discussed in this work originates from [16]. In that paper, a method for fusing the information of MANs is introduced. The first step of the algorithm discussed is an optimal filter, followed by a beamformer. The beamformed data is then passed through a matched filter for each array in the MAN. Fusing the data from all the arrays together, which is done after a transformation of all the arrays into a single array's output, completes the algorithm. Figure 1 shows the algorithm's flow chart:

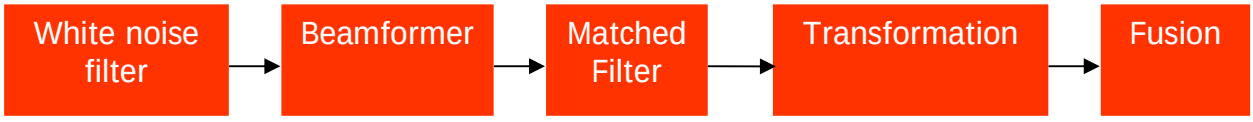

Figure 1: Flow chart of algorithm used in this thesis

If the fused output of this algorithm exceeds a preset threshold, then a positive detection decision is made for the bearing, velocity and distance as determined by the beamformer and matched filter.

1.3 Motivation and Contribution

The motivation for this work stems from the optimization of sensor placement in MANs. If only so many sensors are available for detecting objects, this work will hopefully aid in how those sensors should be arranged. While more sensors with fewer arrays will yield better parameter estimation when the position and orientation are certain, uncertainties in position and orientation could create large errors. Having more arrays with fewer sensors would minimize the errors because some of the smaller sensors would be closer to their assumed position and orientation, thereby correcting errors caused by other array's error in position and orientation.

The first part of this work tests whether or not is it more beneficial to have more arrays with fewer sensors or fewer arrays with more sensors using active sonar with certain array orientation and positions. MATLAB generated simulated signals based on the geometries of the arrays and objects and desired noise level. The simulated data is then filtered, beamformed, matched filtered, and fused as described above. A comparison can then be made regarding each network configuration.

The second part of this work tests when it is more beneficial to have fewer arrays with more sensors or vice versa when the position and orientation of the arrays is uncertain when using passive sonar. Again, MATLAB generated data based on the geometry of the target and arrays. The conventional beamformer determined the AOA for each array and the location of the sound source was determined using triangulation. Uncertainties of varying degree are applied to the orientation and position of the arrays before the data is simulated.

The contribution of this work is threefold. The first objective of this thesis to test the method described in [16]. By simulating data in MATLAB, comparisons can be made on the theoretical performance and the simulated performance of the detection method, and to offer a possible alternative to the data fusion method suggested in [16]. The second objective is to compare different array configurations when the number of sensors remains constant across a varying number of arrays. The third objective of this work is carried out in the second part of

this work, as the comparison is drawn when the position and orientation of the arrays are uncertain.

1.4 Thesis outline

This thesis investigates the performance of active sonar networks with certain position and orientations, and passive sonar location estimation when the orientation and position of the arrays are uncertain. Chapter 2 investigates the processes of filtering, beamforming and matched filtering to determine target location and parameter estimation using a set number of sensors across differing numbers of arrays, as well as a comparison between fusion methods. Chapter 3 investigates the effects of uncertainty of array position and orientation on location estimation of a sound source using passive sonar. A comparison is made to determine if it is more beneficial to have several arrays with fewer sensors or fewer arrays with more sensors. The fourth and final chapter will offer conclusions and possible future work that is related to the ideas presented here.

2 Data Fusion Using Sufficient Statistics in Active System

This chapter discusses the algorithm in chapter 1 as well as the methods of simulation more thoroughly. This chapter investigates the results of these simulations, and some conclusions based on these results conclude this chapter.

2.1 Main Algorithm Used in Active Sonar Simulations

This algorithm was discussed in [16]. The first step of this algorithm is a white noise filter, followed by a conventional beamformer, and concludes with a matched filter for each array. One of three fusion algorithms fuses the output of these arrays together.

2.1.1 Noise Filtering

The first step in this algorithm is a white noise filter. Ideally, the filter would eliminate as much Gaussian noise as possible while still retaining any returned signals. In order to do this, a white noise filter was attempted. The white noise filter is the same that appears in [17]. Figure 2 shows the attempted process:

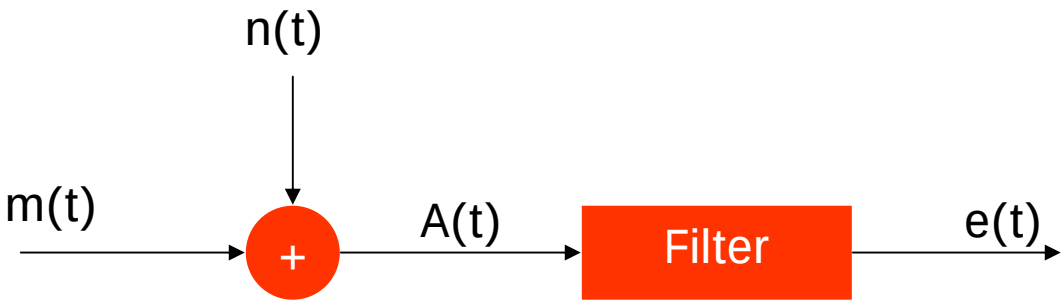

Figure 2: Schematic of White Noise filtering

In Figure 2, m(t) is the returned echo, n(t) is the noise within the system, A(t) is the raw data at each of the sensors and e(t) is the estimated value of m(t). The filter minimizes the error between the estimated signal and the echo, that is, m(t)-e(t) approaches zero. If $(m(t)-e(t))^2$ defines the error, the filter design should minimize this error for a time sample. This can be seen in (1):

$$J = E\left\{\left[m(t) - \sum_{n=0}^{N-1} b_{opt} A(t-\tau) \right]^2 \right\} \qquad (1)$$

In (1), J is the error, and b_{opt} is the optimal filter tap at all time delays (τ). If J is to be minimized, its derivative relative to b_{opt} equals zero, as in (2):

$$\frac{\delta J}{\delta b_{opt}} = 2E\left[e(t) \frac{\delta e(t)}{\delta b_{opt}} \right] = 0 \qquad (2)$$

Substitution and rearranging yields:

$$E\left[e(t) A(t-\tau)\right] = 0 \qquad (3)$$

$$E\left[\left(m(t) - \sum_{i=0}^{N-1} b_i A(t-i) \right) A(t-\tau) \right] = 0 \qquad (4)$$

$$E\left[A(t-\tau) m(t)\right] = E \sum_{i=0}^{N-1} b_{opt} \left[A(t-\tau) A(t-i)\right] \qquad (5)$$

Using the Wiener-Hopf equations on (5) yields:

$$\sum_{i=0}^{N-1} b_{i,opt} R_{AA}(i-\tau) = R_{Am}(-\tau) \qquad (6)$$

Solving (6) for b_{opt} yields:

$$b_{opt} = R_{AA}^{-1} R_{Am} \qquad (7)$$

In (7), R_{AA} is the auto-correlation matrix of the recorded data on a specific sensor. R_{Am} is the cross-correlation matrix between the recorded data and the sent signal. The raw data in a set time-block would be sent through the taps of the filter, thus eliminating most of the Gaussian noise.

This method was investigated as a way to cancel out the white noise of the simulated data each sensor recorded. However, in practice, results were limited since there was no way to account for Doppler shifted version of the sent signal. Using a Doppler shifted version of each possible velocity to filter the raw data would remedy this problem. However, this method would be very computationally expensive to perform. An LMS filter would have behaved very much the same way, as its algorithm would have cancelled out any possible Doppler shifted version of the echo.

In order to reduce the white noise in the sensors, a bandpass elliptical filter attenuated frequencies outside of the echoes range. The raw data at each sensor passed through this filter, reducing the amount of white noise in its signal. Since the expected frequency range of the echo is between 14-16 kHz, the elliptical filter design attenuated any frequencies not in the 12.5 kHz and 17.5 kHz, with the extra range given for stability. Figure 3 shows the filter response of the elliptical filter:

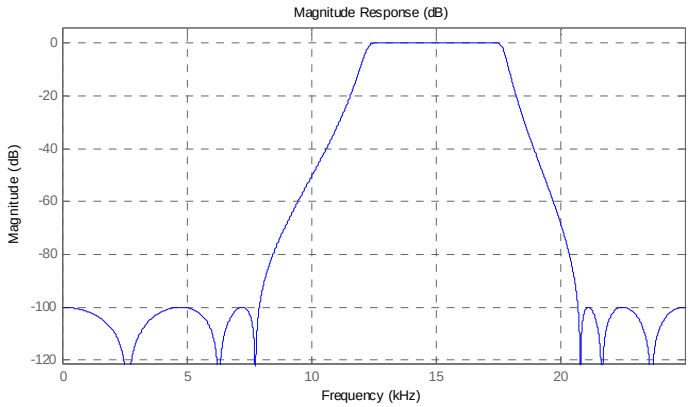

Figure 3: Filter response of elliptical filter used in main algorithm

Figure 4 shows the results of this filtering. Notice how the filtered data shows higher values around t = 0.06-0.08 seconds, which corresponds to a returned echo.

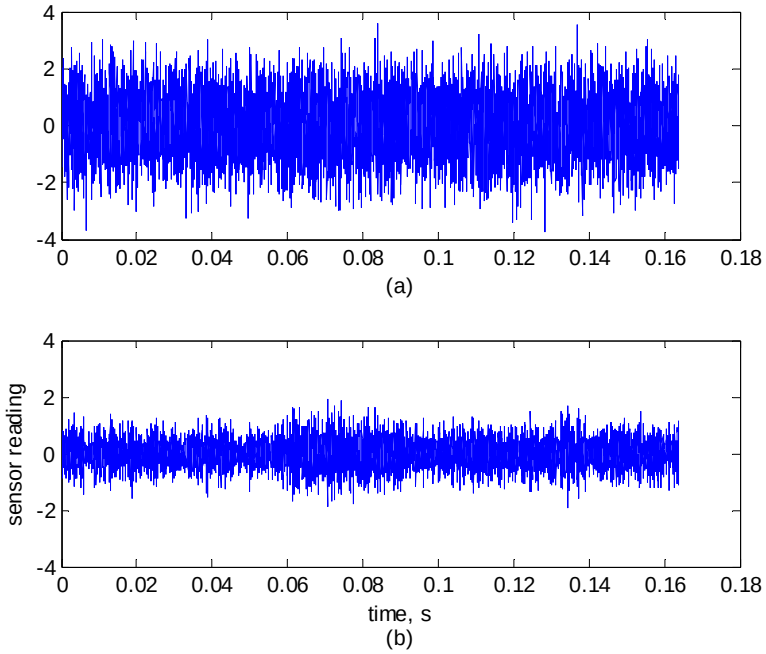

Figure 4: Example of sensor data before (a) and after (b) elliptical filtering

2.1.2 Beamforming data across all arrays

The next step in the main algorithm is to beamform the filtered data across all arrays. This process will lead to an estimate of the AOA of the echoes coming from any targets within the network's range.

2.1.2.1 Delay-and-Sum Theory using steering vectors

In conventional beamforming, the outputs of all sensors in an array are delayed by an appropriate phase difference for a tested bearing angle. The algorithm adds these delayed signals together, thereby creating an enhanced signal if the tested bearing angle contains a signal at the center frequency used in the steering vector.

In order to beamform an array's data using conventional beamformer, all the sensors in the array undergo an FFT. This data is multiplied by a steering vector, based on the tested bearing angle. Using (8) and (9) creates the steering vector based on the tested bearing angle:

$$\tau_n = \frac{(n-1)s\sin(\theta)}{c} \qquad (8)$$

$$S_n = \exp\left(\sqrt{-1} * 2\pi f_c \tau_n\right) \qquad (9)$$

In these equations, s is the space between the sensors in the array, θ is the tested bearing angle, c is the speed of sound in the medium, τ_θ is the time delay between the first element in the array and the n^{th} element, f_c is the center frequency of the expected signal, and S is the steering vector. Multiplying (9) by the FFT of each sensor's filtered data is the equivalent of delaying then summing the outputs of all the sensors using the appropriate time delay as seen in (8).

2.1.2.2 Output of conventional beamformer into time domain

Taking the inverse FFT of that product yields one signal for the entire array for a given bearing angle. For comparison purposes, this output is normalized by dividing it by the number of sensors in each array. The closer the echo's AOA is to the bearing angle being measured, the more power that the signal should contain within the expected frequency range of the echo. The beamformer drastically reduces the amount of noise when comparing its output to one of the sensors in the array, as seen in Figure 5:

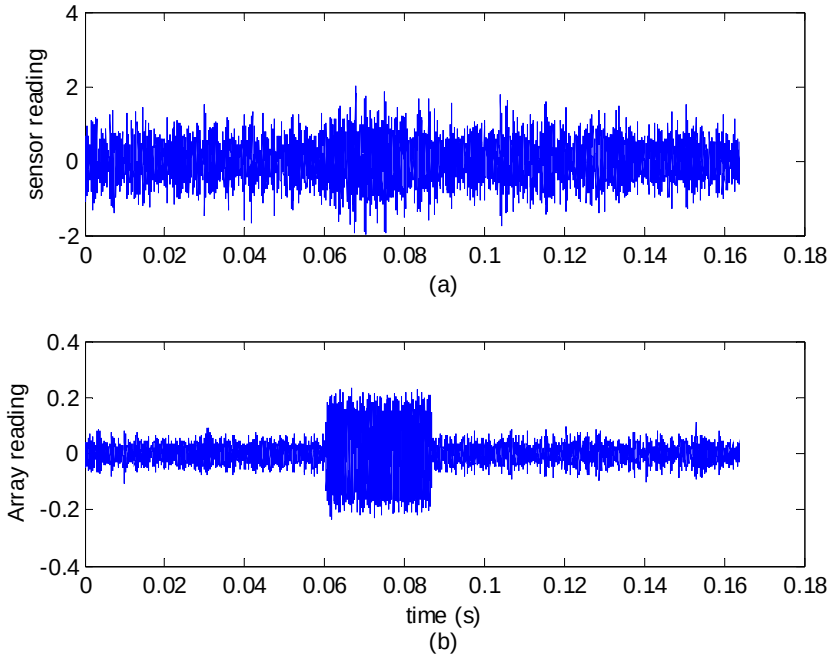

Figure 5: Time series showing one sensor's data (a) and beamformed data (b) in the direction of a target

This signal enhancement makes detection of objects much easier. The more members present in an array, the greater this enhancement becomes, as adding more coherent signals to the overall sum at that bearing angle positively constructs the echoed signal.

2.1.2.3 Determination of AOA

The determination of the AOA is vital to the location of a target using active sonar. The first fusion method, the transformation method, uses the bearing angles of the arrays to accurately transform the output of all secondary arrays to the primary array's range-Doppler map. The other methods use this estimate to determine the location of targets as well, as later sections will discuss.

2.1.2.3.1 Determination of AOA of a single target

The beamformer output's power can be measured across the expected frequencies of the echoes. By calculating the power spectral density (psd) of the beamformer output, each frequency bin at each bearing angle contains a certain amount of power. Adding this power across the expected echo's frequencies, as seen in (10), determines an estimate for the signal strength for a set bearing angle.

$$P(\theta) = \sum_{f_l}^{f_h} psd(f, \theta) \qquad (10)$$

In (10), f_l and f_h represent the lower and higher bounds of the frequencies where the psd is being measured, and $P(\theta)$ is the total power for the bearing angle θ. In the experiments done here, the algorithm uses the signal power between 14 and 16 kHz. Figure 6 shows the psd of the beamformer's output across all possible bearing angles and frequencies. Figure 7 shows the summation across the frequencies across all bearing angles, with a target at approximately 58º, a 16-channel array, and an SNR of -16 dB at each channel.

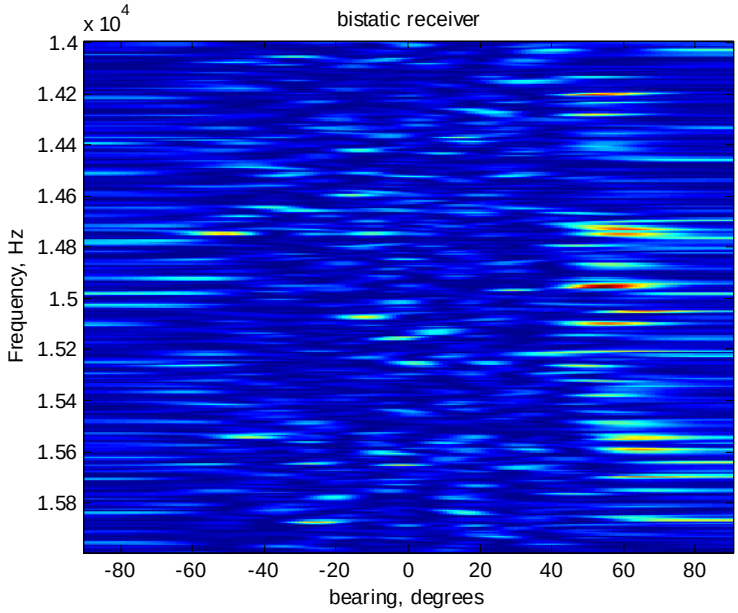

Figure 6: Power Spectral Density measurement as function of frequency and bearing angle

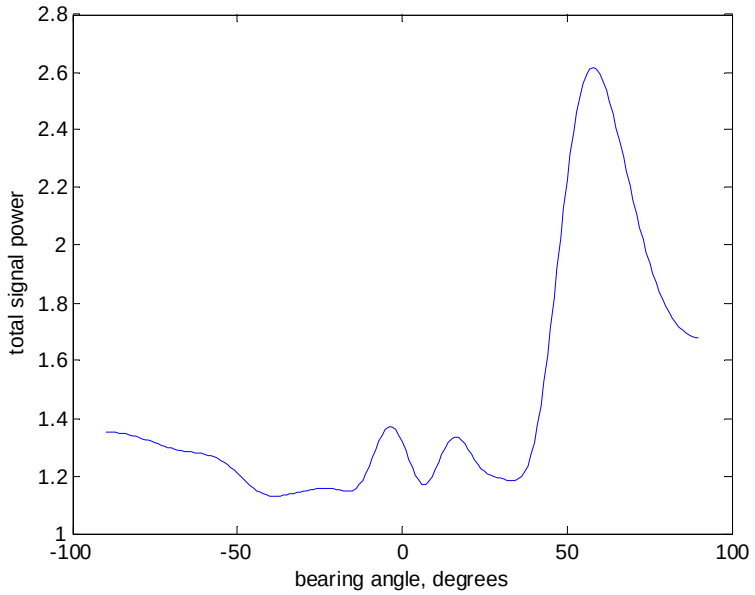

Figure 7: Plot of summed psd as function of bearing angle with target at 58º

In cases where it is known that there is only one target in the network's field, the bearing angle that has the maximum power at that output can be used as the estimate for the echo's AOA. The next section will discuss cases with multiple targets.

2.1.2.3.2 Determination of AOA with multiple targets

In cases of multiple targets, it is intuitive to try using a threshold approach to detect objects from the bearing angle being measured, that is decide a target is present at a certain angle if $P(\theta)$ exceeds a preset value. However, due to sidelobes that occur in the beamforming process as well as inconsistency due to random noise in several trials using this method, simulations with more then one target in the network's range did not use this method.

Instead of that method, the sufficient statistics from the matched filter estimated the bearing angle of multiple targets. Instead of just the peak angle's output getting match filtered, all angle's output undergo match filtering. A separate matrix records the maximum sufficient statistic for each time sample and velocity. For each new tested bearing angle, the resulting sufficient statistics are compared to previous sufficient statistics. After all outputs have been filtered, if the sufficient statistic exceeds the threshold for that time sample, the estimate for the target's bearing angle becomes the bearing angle that yielded the highest sufficient statistic.

2.1.3 Matched filtering of beamformed data

Matched filtering the beamformed data allows for the estimation of distance and relative velocity of targets using active sonar. This is done by cross-correlating a Doppler-shifted version of the emitted signal.

2.1.3.1 Determination of velocity by Doppler-shifting

Whenever sound reflects off a moving object, it undergoes a Doppler shift which effects the duration and frequency of the sound relative to a stationary receiver. The compression or expansion of the sound waves reflecting off the moving object causes this phenomenon. If the object moves toward the observer (or sensor), the sound waves will compress, resulting in a higher observed frequency and a shorter sound duration. The opposite is true for an object moving away from the stationary observer, as the motion will lower the observed frequency and lengthen the sound duration. The important concept here is that the *number of waves does not change*- which is why time dilation occurs as well as a change in frequency. This change in frequency is calculated in (11):

$$f_d = f\left(c/(v+c)\right) \qquad (11)$$

In (11), c is the speed of sound in the medium, and v is the relative velocity of the object to the observer. Since the number of waves must remain the same, a new duration occurs due to the change in frequency. The dilated time can be calculated using (12):

$$t_d = N_w / f_d \qquad (12)$$

In (12), N_w is the number of waves emitted. Combining these effects, a Doppler-shifted version of the emitted signal for any velocity can be constructed. Cross-correlating this shifted version with the output of the beamformer algorithm obtains an estimate for relative velocity.

In practice, a practical range of velocities should be selected to detect high-speed objects in water. In these simulations, a range of -20 to 20 m/s was selected.

2.1.3.2 Determination of distance using cross-correlation

Once a set of desired Doppler-shifted versions of the emitted signal has been created, they can be cross-correlated with the beamformer's output to determine the distance from the array. Cross correlation measures how closely two data sets are related over time. In equation form, it looks like (13):

$$r(k) = \sum_{t=0}^{N-1} m(t)A(t-\tau) \qquad (13)$$

In (13), τ is the lag, and r is the sufficient statistic. The lag in this case represents a time delay. Since there is no data for t >0, all of these times for A are assigned a zero value. After cross correlating all possible Doppler-shifted versions of the emitted signal, a range-Doppler map can be constructed using the sufficient statistics. Figure 8 shows an example output of a range-Doppler map, where there are three targets present with different parameters:

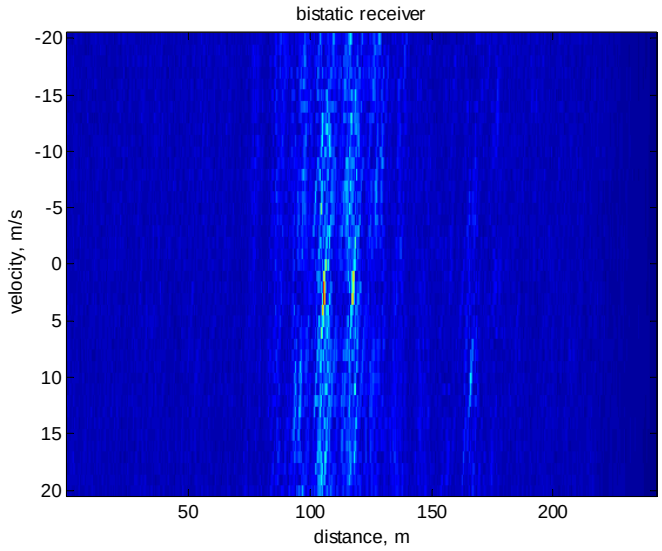

Figure 8: Range-Doppler map showing three targets with different ranges and velocities

Using (14) readily determines the distance from the monostatic array to the target, with the distance divided by two to account for the round trip:

$$x = ct/2 \qquad (14)$$

However, when a bistatic array such as the one in Figure 9 is considered, finding the proper distance from the array must use the law of cosines.

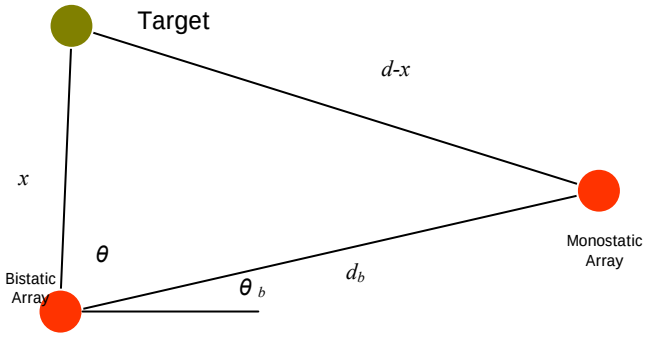

Figure 9: Bistatic array detection scenario

For the scenario seen in Figure 9, using (15) can find x:

$$x = d - \frac{(d_b^{\,2} + d^2) - 2dd_b\cos(\theta - \theta_b)}{2d - 2d_b\cos(\theta - \theta_b)} \qquad (15)$$

In (15), d represents the total distance from the original sound source to the target to the array as determined by the matched filter, and θ is the bistatic array's AOA as determined by the beamformer.

Once the arrays estimate the distances to a target, the determined AOA from the beamformer and the distance determined by the matched filter can make an estimate of their location. The estimated coordinates of this location, x_e and y_e, can be determined using (16):

$$(x_e, y_e) = (x_s + d\sin(\theta), y_s + d\cos(\theta)) \qquad (16)$$

In (16), x_s and y_s are the coordinates of the array. The arrays, depending on the fusion method, can send their estimates to a fusion center to help make a universal estimate of the location of any targets within the sensor field. Future sections will discuss this at length.

In order to find the proper velocity, the network must use at least two arrays that are not collinear with the target. Each array can measure only relative velocities in the direction of the determined AOA; that is the velocity vectors must be parallel to the determined AOA of each array. Figure 10 shows a graphical method to find the true velocity using two arrays that use the AOA from the beamformer and the measured velocity from the matched filtering:

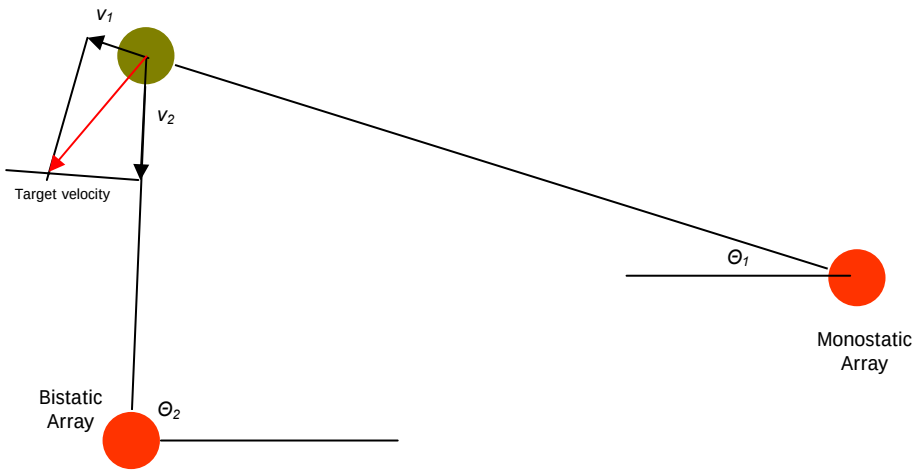

Figure 10: Graphical representation to find target's velocity using estimated AOAs and measured relative velocities

In order to determine the complete velocity numerically, the above situation can be viewed as the intersection of two lines that start at the ends of the velocity vectors, v_1 and v_2, and are perpendicular to the AOAs for each array, θ_1 and θ_2. That is, the starting points for i^{th} line, v_{sxi} and v_{syi}, can be found using (17):

$$(v_{sxi}, v_{syi}) = (V_i \cos(\theta_i), V_i \sin(\theta_i)) \quad (17)$$

The slopes of those two lines are the slopes of the AOAs for either array. Using simple algebra allows for a solution of the velocity, as seen in (18):

$$v_x = \frac{\tan(\theta_2)v_{sx2} - \tan(\theta_1)v_{sx1} + v_{sy1} - v_{sy2}}{\tan(\theta_1) - \tan(\theta_2)} \quad (18a)$$

$$v_y = \tan(\theta_1)(v_x + v_{sx1}) - v_{sy1} \quad (18b)$$

A section later in this work will discuss an algorithm to find the velocity if the MAN uses more then two arrays using Gaussian curves.

2.1.4 Fusing data from all arrays to make universal parameter estimations

This work discusses three different fusion methods as a way to fuse the data from all the arrays in a MAN. The three fusion methods are: transformation of the output of all arrays into one array's output, using 2-D Gaussian curves around the points where targets have been detected, and the use of weighted averages to determine location of the objects.

2.1.4.1 Transformation method using geometries estimated by beamformer and matched filter

In order to transform the sufficient statistics of a secondary array into a reference array, two transformations need to be performed. The first transformation is velocity. This is done by first making a determination of the target's velocity by using the algorithm described above. Next, taking the arctangent of the x and y components of the determined velocity finds the direction of the objects velocity, φ as seen in (19):

$$\phi = \arctan\left(\frac{v_x}{v_y}\right) \quad (19)$$

Once (19) determines ϕ, two dot products as seen in (20) transform the secondary array's velocity into the reference array's velocity:

$$V_t = V_2 \frac{((\sin(\phi),\cos(\phi)) \bullet (\sin(\theta_2),\cos(\theta_2)))}{((\sin(\phi),\cos(\phi)) \bullet (\sin(\theta_1),\cos(\theta_1)))} \qquad (20)$$

In (20), θ_1 is the AOA of the reference array, θ_2 is the AOA of the secondary array, and V_2 is the velocity to be transferred into the reference array's range-Doppler map. The dot product in the numerator projects the secondary array's velocity onto the determined direction of the target's velocity, and the dot product in the denominator projects that velocity into the reference array's AOA. Figure 11 shows a graphical representation of this algorithm:

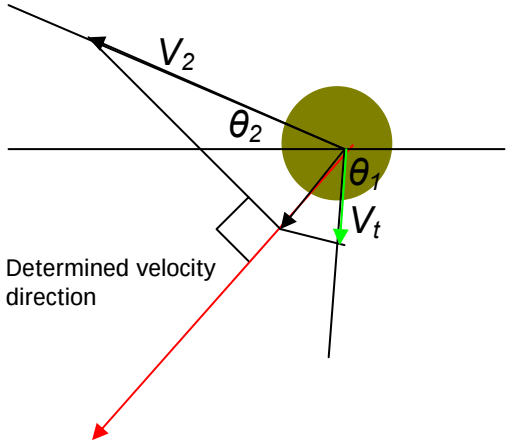

Figure 11: Graphical representation of velocity transformation

Once the dot products find the transformed velocities, new Doppler shifted versions of the sent signal based on these velocities are created and passed through the matched filter algorithm described in the previous section.

The second part of this algorithm transforms the distances from the secondary array into the reference array. This is accomplished using law of cosines, as seen in (21):

$$x = d - \frac{(d_b^{\,2} + d^2) - 2dd_b\cos(\theta_2 - \theta_b)}{2d - 2d_b\cos(\theta_2 - \theta_b)} \qquad (21)$$

In (21), the transformed distance, x, is determined for all time samples that correspond to a distance d. When needed, the algorithm uses an interpolated value for the sufficient statistic between two distances with known sufficient statistics. Figure 12 shows an example of a transformed range-Doppler map:

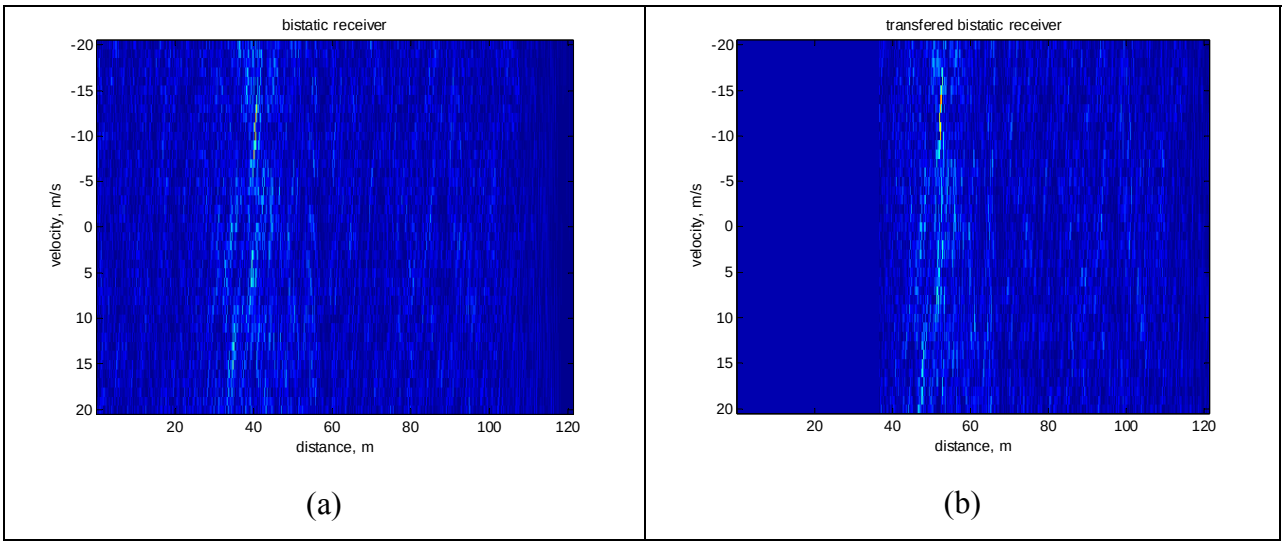

Figure 12: Example of transformed sufficient statistics, with the original range-Doppler map (a) and the transformed map in (b).

The geometry of the determined AOA and the positions of the arrays create the large uniform area on the left hand side of Figure 12b, since it is impossible for the target to come that close to the primary target given the determined AOA for the secondary receiver. These sufficient statistics are therefore given a zero value.

The transferred range-Doppler maps for all the arrays within the networks are then multiplied together. If the parameters estimated by the beamformer and matched filter are correct in all the arrays, the high sufficient statistics caused by the echo should constructively multiply, causing an enhanced sufficient statistic with the correct parameters of the target relative to the reference array. Figure 13 shows this effect, where the target's distance and velocity relative to the primary array are clearly seen:

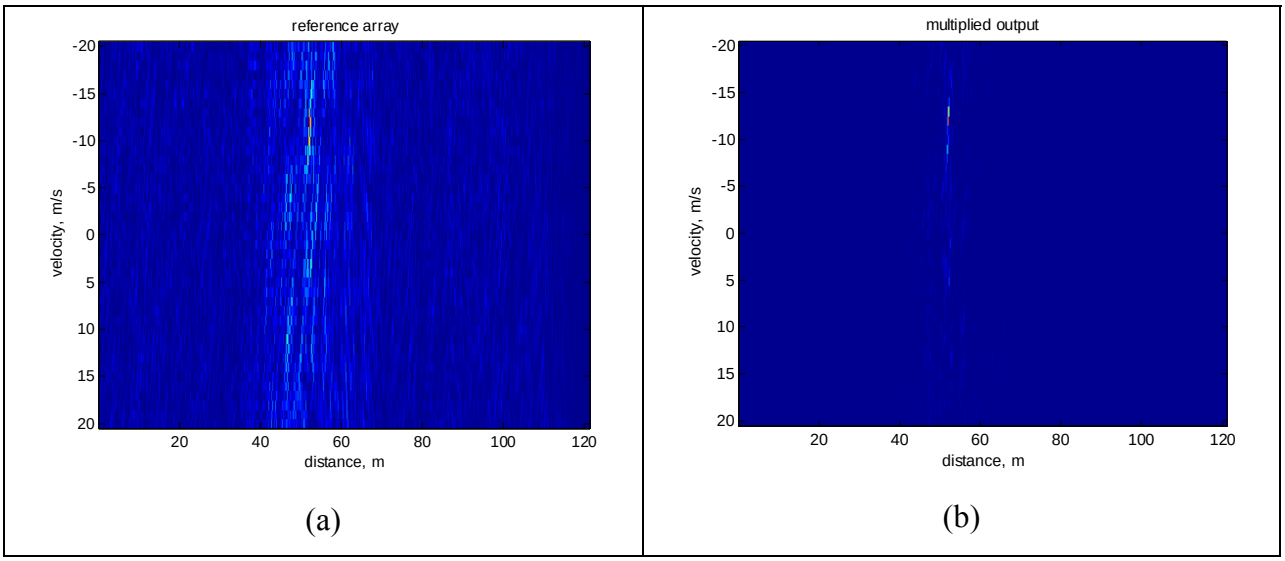

Figure 13: Example of sufficient statistic enhancement using range-Doppler map transformation.

The reference array's range-Doppler map is shown in (a), and is multiplied by a transferred map from another array to create (b). If more arrays can estimate the parameters of the target correctly, the higher the fused sufficient statistics will become. This in turn will lead to better detection of targets within the network's field.

2.1.4.2 Data fusion using Gaussian curves to determine parameters of targets

Another method of data fusion involves fitting a 2-D Gaussian curve around the estimated position of targets. First, a mesh of points throughout the network array's range is created. The maximum distance of this mesh field should equal the distance a sound would travel in the amount of time sampled by the sensors. Since the sound travels about 240m in the amount of sampled time, the grid was set up as a 240 by 240m mesh, with .25m between each point and the emitted chirp originating at the center of this mesh.

In cases where the signal strength was constant across all arrays, the algorithm used a constant threshold to make detection decisions. In cases where the signal strength varied with distance, a threshold that matched the signal's decay was used, as determined by (22):

$$T = \frac{T_r}{\left(\frac{t}{t_r}\right)^2} \qquad (22)$$

In (22), the threshold T is determined by a reference threshold, T_r at a reference time, t_r. This value is computed for each time sample in the generated data. Practice showed that one target

can cause more then one sufficient statistic to exceed the threshold. Because of this, a proximity test was performed on any sample that exceeds the threshold in both range and velocity. For these simulations, the algorithm performed the proximity test on all sufficient statistics within 50 time samples and all velocities.

The maximum sufficient statistic in the area of the proximity test has its location estimated using its AOA and distance as determined by the beamformer and matched filter. For each array, each point gets assigned a value based on (23)-(25):

$$d = \sqrt{\left((x - x_e)^2 + (y - y_e)^2\right)} \tag{23}$$

$$\sigma = \sqrt{\left((x_e - x_s)^2 + (y_e - y_s)^2\right)} \tan(\beta) \tag{24}$$

$$B(x, y) = \frac{\exp\left(-\frac{d^2}{2\sigma^2}\right)}{\sigma\sqrt{2\pi}} \tag{25}$$

In these equations, (x_e, y_e) is the estimated position of the target, (x_s, y_s) is the position of the array, and β is an angle equal to the beamwidth of a noiseless signal. The result of these equations show higher B values when the beamwidth is lower (e.g. there are more sensors in each array) and closer to an array. Each point within the mesh grid has these equations performed for each array. Since the beamwidth depends on the bearing estimate of the signal and the number of sensors in an array, differing values for β were used according to Table 1:

16 channels	
$\lvert\theta\rvert$	β
0º-21º	6º
22º-32º	7º
33º-43º	8º
44º-50º	9º
51º-52º	10º
53º-55º	11º
56º-57º	12º
58º-59º	13º
60º-62º	14º
62º-70º	15º
71º-79º	16º
80º-84º	17º
85º-90º	16º

8 channels	
$\lvert\theta\rvert$	β
0º-24º	10º
25º-35º	11º
36º-42º	12º
43º-47º	13º
48º-52º	14º
53º-55º	15º
56º-59º	16º
60º-61º	17º
62º	18º
62º-64º	19º
65º	20º
65º-67º	21º
68º	22º
69º	23º
70º-71º	24º
72º-73º	25º
74º-76º	26º
77º-78º	27º
79º-80º	28º
81º-82º	29º
83º-90º	θ-53º

4 channels	
$\lvert\theta\rvert$	β
0º-32º	26º
33º-38º	27º
39º-41º	28º
42º-46º	29º
47º-50º	30º
51º-52º	31º
53º-55º	32º
56º-58º	33º
59º-61º	34º
62º-64º	35º
65º-68º	θ-28º
68º-69º	θ-29º
70º-72º	θ-30º
73º-74º	θ-31º
75º-78º	θ-32º
79º-83º	θ-33º
84º-90º	θ-34º

Table 1: List of beamwidths used given array size and bearing estimate

As discussed previously, for multiple targets all of the bearing angles have their output matched filtered, while if it is known that only one target is present, only the bearing angle with the maximum power gets match filtered. Each target detected has its own estimated position, and for each point in the mesh the values determined in (23)-(25) for each target add together. The values on the grid multiply together across all arrays to make a universal location decision. An example of two grids being fused together with one target is seen in Figure 14:

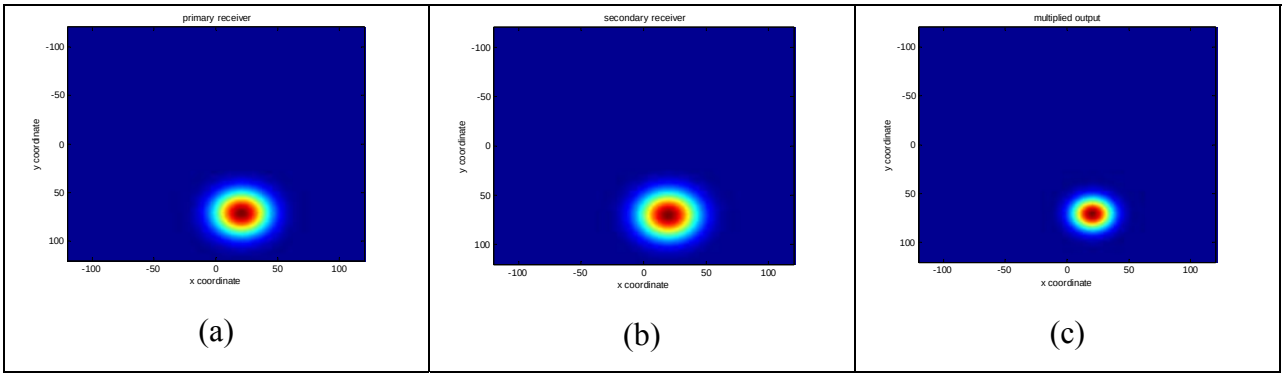

Figure 14: Example of two arrays, (a) and (b), whose outputs are being fused together to make (c).

The reasoning behind this fusion method is that it is somewhat comparable to the transformation fusion method. The outputs will constructively multiply if there are multiple arrays estimating targets to be in the same area. It also allows for uncertainty in estimation, as the points close to the estimated position will still have high values, whereas a simple transformation might have relatively low values, resulting in cancellation when these values are multiplied, or ambiguous estimates due to a lack of precise estimates. As can be seen in the previous figure, the fused output has a smaller circle for the detected range of the target.

In order to find targets, the mesh is checked across all points to see if it is a local maximum. This check declares targets at local maximums.

Velocities can also be found using a Gaussian curve, however this curve is fitted over a line and not a point, since a Doppler shifted echo can come from a target moving in any direction as long as the relative velocity matches the estimate as determined by the matched filter. Again, this method uses a mesh and all points are given a value based on their distance from this line. The estimated AOA from the beamformer and the estimated relative velocity determine the equation for the line for the i^{th} array (26):

$$y = -\tan(\theta_i)x - V_i \qquad (26)$$

The distance to this line is found using (27):

$$d_v = \frac{\tan(\theta_i)x + y + \tan(\theta_i)V_i \sin(\theta_i) + V_i \cos(\theta_1) + 1}{\sqrt{1 + (\tan(\theta_i))^2}} \quad (27)$$

Once (27) finds the distance to this line, (28) assigns a value based on a Gaussian curve to each point on the grid:

$$B(x, y) = \frac{\exp\left(-\frac{d_v^{\,2}}{2\sigma^2}\right)}{\sigma\sqrt{2\pi}} \quad (28)$$

These values are done for each target found by each array. A proximity test determines if the arrays see the same target, and the grids for each individual target are multiplied accordingly. Figure 15 shows an example of this method for a one target case.

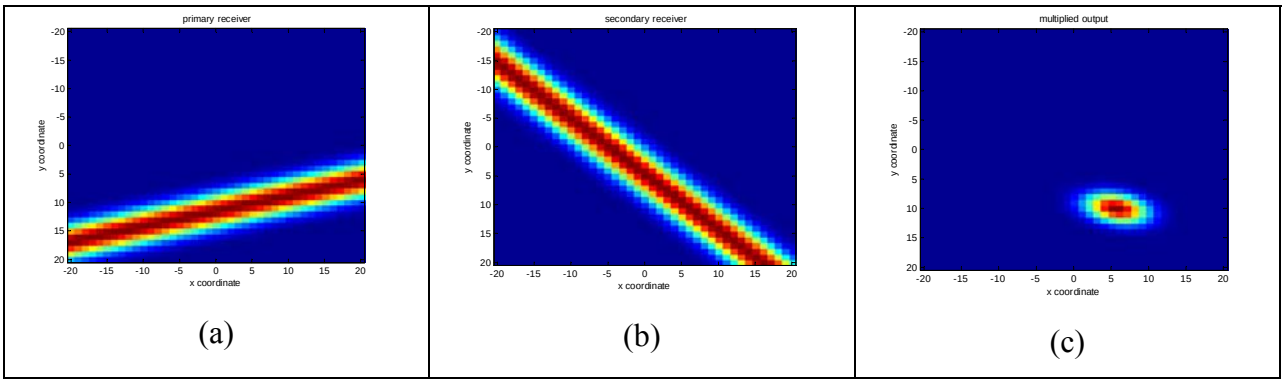

Figure 15: Example of fusing two arrays velocity measurements, (a) and (b), to make a universal decision (c).

The x and y component corresponding to the highest value in this grid for each target represents the estimate of the target's velocity based on the data received from all arrays. The velocity mesh for all simulations was -20m/s to 20 m/s in both the x and y directions in one m/s increments.

2.1.4.3 Data fusion using weighted averages to determine the location of targets

The final data fusion method tested used weighted averages based on the sufficient statistics from each array. If the sufficient statistic exceeds a threshold, a positive detection decision is made. The locations of the targets for each array is then determined using (16). The sufficient statistic determines the weight this point receives. This allows for more certain estimates to influence the overall estimate more then less certain values. In case of multiple targets, other array's estimates compare these points and if they are within a preset distance to each other, (29) computes the universal estimate of the target's position:

$$(x_e, y_e) = \left(\frac{\sum_{n=1}^{N} M_n x_n}{\sum_{n=1}^{N} M_n}, \frac{\sum_{n=1}^{N} M_n y_n}{\sum_{n=1}^{N} M_n} \right) \quad (29)$$

In (29), M is the sufficient statistic that each array assigns to each individual target, and (x_n, y_n) is the estimated position of the target by an individual array. (x_e, y_e) is the universal estimate of the target's position.

2.2 Raw data simulation using MATLAB

The algorithm described in 2.1 was implemented into a MATLAB code, and ran on MATLAB generated data. This section investigates how MATLAB generated the data based on the position and velocities of the arrays and targets.

2.2.1 Model of emitted signal

The emitted signal used in the active sonar simulations was a sequence of eight tones with varying length and frequency. The designed signal intentionally had an average frequency of 15000 Hz, as that matches the central frequency in the beamformer algorithm. The frequencies differed no more then 5% more or less then 15000 Hz, therefore the lowest frequency in the signal was 14250 Hz, and the highest was 15750 Hz. The duration of the total emitted signal was 0.0267 seconds. Table 2 shows the specifics of the emitted signal:

Tone #	1	2	3	4	5	6	7	8
Frequency (Hz)	14250	15750	14800	15250	14650	15350	14700	15550
Duration (s)	.00421	.00254	.00372	.00197	.00375	.00423	.00238	.00386

Table 2: Duration and frequency of the eight tones in emitted signal

This particular signal limits the ambiguities in the matched filter. If the emitted sound used a pure tone, the output of the matched filter would have ambiguity in range. If a linear chirp was used, the sufficient statistics would have ambiguity in velocity, as Figure 16 shows:

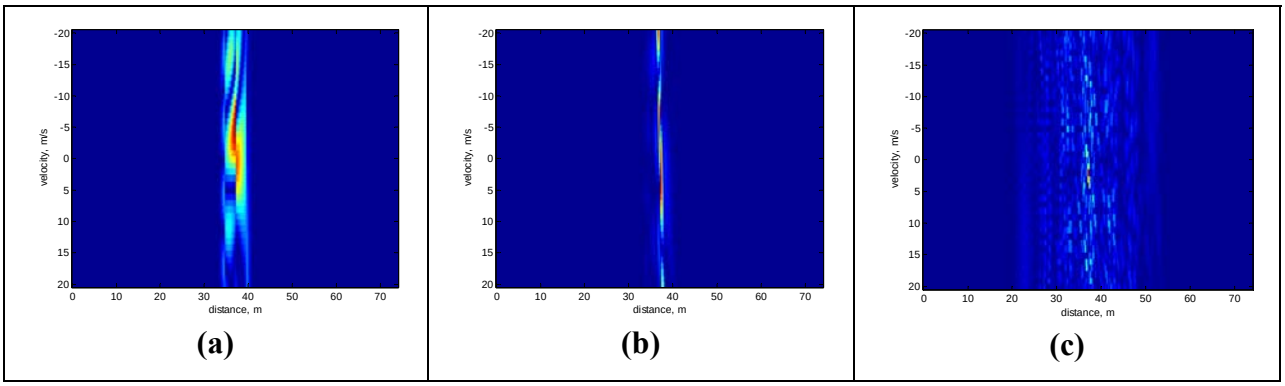

Figure 16: Range-Doppler outputs for a noiseless tone (a), linear chirp (b), and the eight-tone signal (c)

As shown, using this signal defines the peak in (c) much more then in the other two plots, with less ambiguity in range and velocity. However, this chirp produces some ambiguity in both range and velocity, as seen by the several smaller peaks surrounding the main peak in (c).

2.2.2 Position of the arrays and targets

Placing the targets in front of the arrays eliminated any front-back ambiguity. This algorithm could easily be tailored to suit any position of a target, but for computational purposes this work will focus only on targets in front of the arrays. Figure 17 shows the position of the arrays and targets used in the simulations:

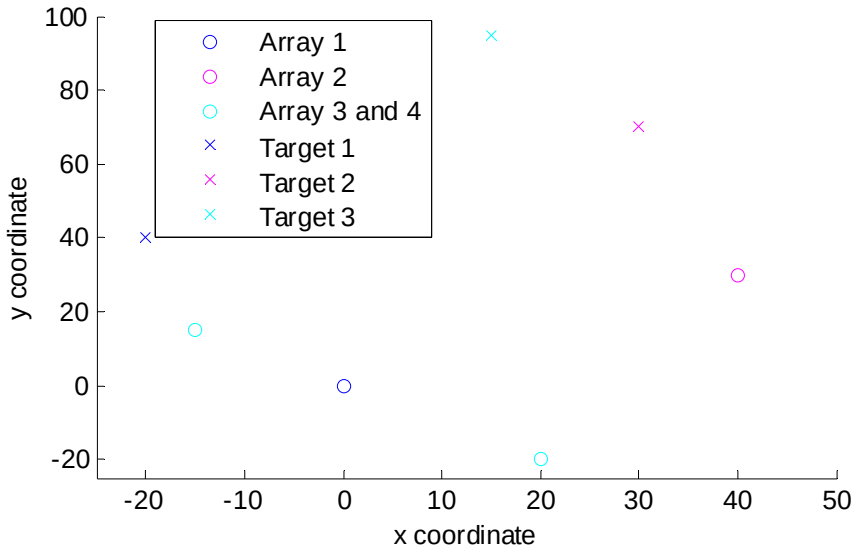

Figure 17: Position of arrays and targets used in active sonar simulations

In Figure 17, the blue circle indicates position of the primary array and the source of the emitted signal. It is used in all simulations. Networks that have at least two arrays have arrays at the

magenta circle, and networks with four arrays also have arrays at the cyan circles. In cases with at least one target, it was placed at the blue cross. In cases with two targets, the magenta cross indicates the position of the second target. In cases with three targets, a third target was placed at the cyan cross.

2.2.3 Generation of simulated data

Based on the geometries as shown in the previous section, MATLAB simulated for all the sensors. This section will detail how these data were simulated, with each sensor recording 8192 samples with a sampling frequency of 50 kHz.

2.2.3.1 Time delay due to distance

The first parameter considered in the signal model is the delay of the returned signal. This delay depends on the distance between the reference signal to the target to the array, and the amount of time taken for a plane wave to travel between the individual elements of an array. The time delay due to the location of the center of the array, τ_d, is found using (30):

$$\tau_d = \frac{\sqrt{(x_a - x_t)^2 + (y_a - y_t)^2} + \sqrt{(x_1 - x_t)^2 + (y_1 - y_t)^2}}{c} \quad (30)$$

In (30), (x_1, y_1) is the location of the emitted signal, which in these simulations is the same as the location of the primary array. In the case where the primary array's data is generated for one of its sensors, (30) reduces to (31):

$$\tau_d = \frac{2\sqrt{(x_1 - x_t)^2 + (y_1 - y_t)^2}}{c} \quad (31)$$

This value for τ_d is the same across all sensors in a given array.

2.2.3.2 Time delay due to AOA of echo

The echoes in this simulation assumes plane wave behavior, therefore an equal amount of time passes for the echoes to reach one sensor to the next across the entire linear array. The amount of time it takes for the echo to reach a given sensor due to the AOA relative to the center of the array can be determined from (32):

$$\tau_\theta = \frac{\sin(AOA)s\left(\frac{N-1}{2}\right)n}{c} \qquad (32)$$

In (32), N is the number of sensors on the array, s is the space between each sensor, and n is the sensor number as counted from the right side of the array. Using (32), a wave from a source directly in front of the array has an AOA of 0º, and any AOA from the left side of the array is considered negative.

The time delays are then added together for each sensor on each array, yielding (33):

$$\tau_t = \tau_d + \tau_\theta \qquad (33)$$

With the time delays known for each sensor, the appropriate time delays for each sensor were programmed into each sensor's data. However, since the sampling frequency was not high enough to simply create the echo, an oversampled signal was created with a time step 100 times lower then the sampling frequency. Based on the time difference from the last sample time to when the echo arrived, the algorithm used one of the first 100 samples of this new signal as the first sample with the echo included, using (34):

$$Q = 100\left[floor\left(\tau_t - t\right)\right]f_s \qquad (34)$$

In (34), t is the time of the last time sample before the echo reaches the sensor, and Q is the sample number of the first sample to be included in the simulated echo. After the first reading, subsequent values of the echo use every 100th sample of the oversampled signal. This created a properly delayed signal at the sampling frequency used in the simulations.

2.2.3.3 Doppler shifting based on relative velocity

The simulated echo also undergoes a Doppler shift consistent with the target's relative velocity. A dot product is performed on the target's velocity vector and a unit vector in the direction of the proper AOA of the echo in order to obtain the relative velocity, v_r, of the target, as seen in (35):

$$v_r = \left(v_x, v_y\right) \bullet \left(\sin\left(AOA\right), \cos\left(AOA\right)\right) \qquad (35)$$

Once the dot product determines the relative velocity, each of the eight tones in the emitted signal undergoes a Doppler shift in frequency as seen in (11) and the length of each tone is dilated using (12). The oversampled version of the echo includes these effects.

2.2.3.4 Addition of Gaussian noise

In these simulations, all samples were given a random number based on a Gaussian distribution with a variance of one. Instead of changing the noise level, the signal strength varied depending on the simulation performed. The signal strength was either held constant across all arrays, or the total distance traveled by the sound from the sound source to the target and to the array determined the strength. In the non-constant case, the SNR of the echo emulates a well-known equation that relates a sound's intensity to the total distance it travels, as seen in (36):

$$SNR = SNR_1\left(\frac{d_1}{d_t}\right)^2 \quad (36)$$

In (36), SNR_1 is the known signal to noise ratio (SNR) at a known distance, d_1 and d_t is the total distance traveled by the emitted signal, as determined in (31). In the simulations that use this equation to determine signal strength, SNR_1 was set to 75 dB and d_1 was set to one meter.

These four components, the delay due to distance, the delay due to AOA, the Doppler shift and the SNR, constitute the simulated data. The complete signal model for one target's echo can now be formulated using (37):

$$A(t) = n(t) \qquad t < \tau_t, \qquad (37a)$$

$$t > \tau_t + t_d$$

$$A(t) = SNR\left[\sin\left(2\pi f_d\left(t - \tau_t\right)\right)\right] + n(t) \qquad \tau_t < t < \tau_t + t_d \qquad (37b)$$

Using (37), MATLAB generates the signal in three stages. The first stage simulates the sensor reading before the echo reaches the target, resulting in a white noise measurement, $n(t)$. The second stage simulates the part of the reading that records the echo along with a white noise component. The last stage, which occurs after the complete Doppler-shifted version of the signal has past, also simulates a white noise measurement. In cases where more then one target is simulated in the sensor field, (37) is applied to each individual target, and the results at each time sample add together. Figure 18 shows an example of a raw signal with three returned echoes with and without noise:

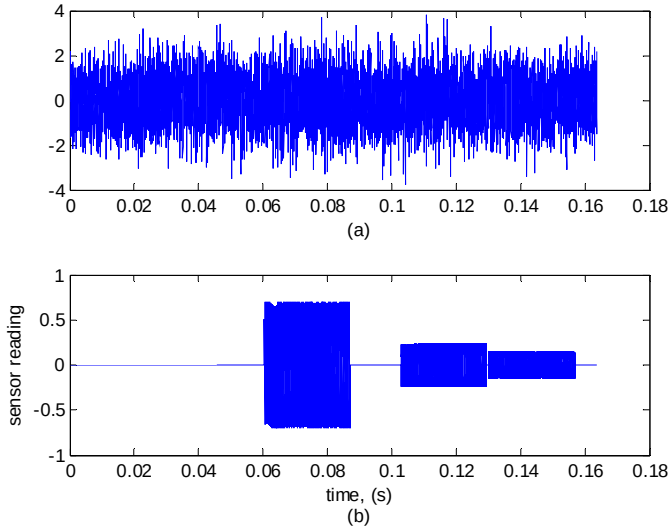

Figure 18: Example of generated data with three echoes, with (a) and without (b) white noise

These signals are generated across all sensors in the network, and algorithm described in 2.1 processes the data.

2.3 Results of MATLAB simulations

This section investigates the performance of the networks when this algorithm is implemented. In these simulations, a total of 32 sensors were used in one, two and four arrays, with positions as seen in Figure 17. The average error across all simulated cases will measure the performance.

2.3.1 Parameter estimation using one array on one target

To gain further insight into the performance of the fused output, this section will investigate the performance of each individual array. Each array estimates four parameters: the bearing angle as determined by the beamformer, and the range, relative velocity and sufficient statistic as determined by the matched filter. These parameters will be compared with different noise levels on each of the sensors. The results in this section show the performance of one array's estimation of one target's parameters. In these simulations, each case was run at least 500 times, with only the output from the highest bearing measurement match filtered.

2.3.1.1 Estimation of AOA

As can be expected, lower signal strength results in less accurate estimates for all parameters, including AOA. Also, more sensors will cause the beamformed signal to be stronger because the signals will add constructively at the proper AOA. Since the beamformer was performed in one degree increments, this discretization may cause some of the small errors. Figure 19 shows the average error of the bearing estimate with different signal strengths and number of sensors in each array.

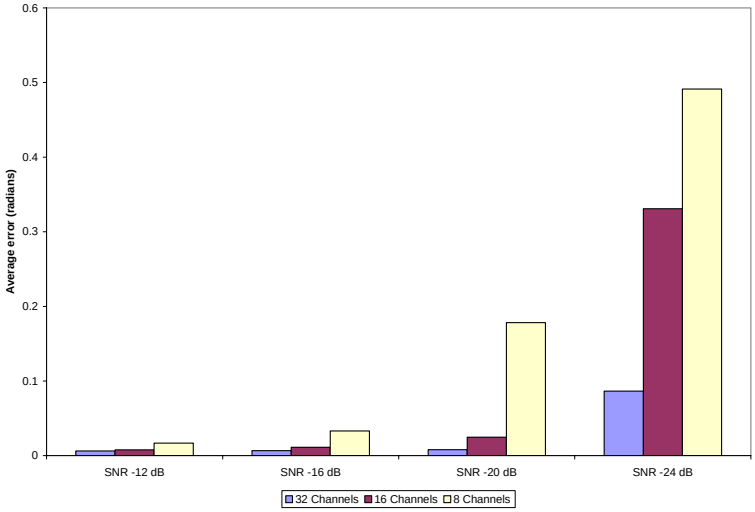

Figure 19: Mean errors of estimated AOA on one array with given SNR and number of channels, in radians

2.3.1.2 Estimation of range

The estimation of range should also display similar behavior to the estimation of AOA, as a weaker signal allow for higher errors in parameter estimation. Figure 20 shows the average error in range for a given number of channels and noise level:

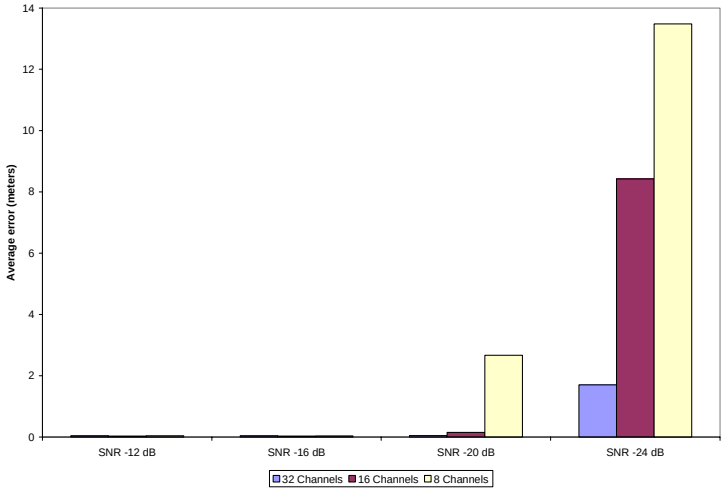

Figure 20: Errors in meters of the estimate of range of the primary array to target 51.95m away

It should be noted that the discretization of distance using the sampling frequency of 50 kHz with a speed of sound of 1480 m/s is 0.0148 m, which means the smaller errors are within three time samples. This may cause the inconsistency in these numbers at higher signal strengths.

Using (16), a position estimate for an object can be obtained using the AOA as determined by the beamformer, and the distance that corresponds to the maximum sufficient statistic. Figure 21 shows the average errors using one array to locate one target:

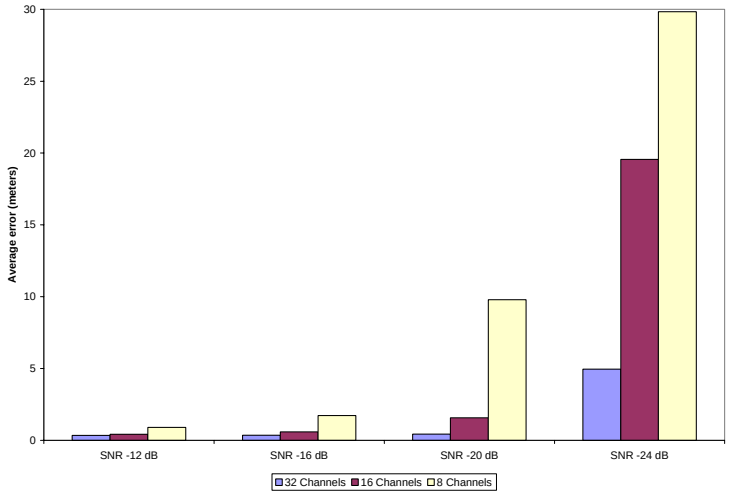

Figure 21: Error of target location in meters using one array given number of sensors and signal strength

2.3.1.3 Estimation of relative velocity

Using the matched filter, the relative velocity estimate should also follow this pattern of higher errors with fewer channels or weaker signal strengths. Again, discretization may also be a factor in these estimates, as the matched filter measured the relative velocity in 1 m/s increments. Figure 22 shows the average errors of the estimated relative velocities given the number of channels and noise levels:

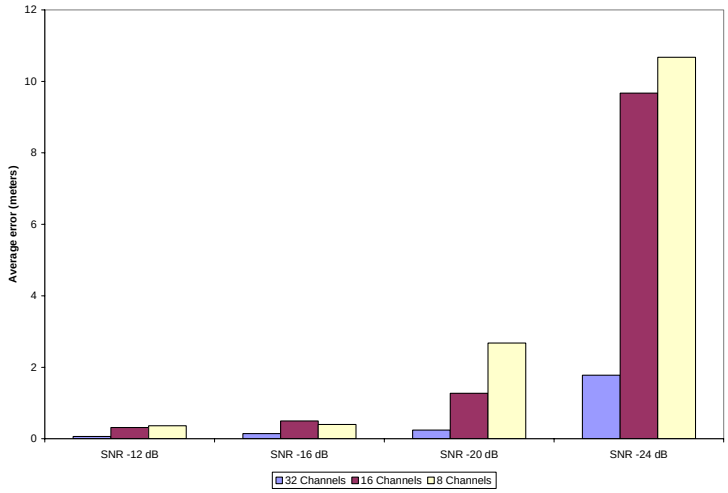

Figure 22: Average errors in m/s of relative velocities on the reference arrays, with v_r equaling 10.9778 m/s

The extremely high errors when the SNR is set to -24 dB for some of the smaller arrays may represent the estimate being totally random, as this approaches the bound in error for a uniform distribution across all possible velocities.

2.3.1.4 Creation of sufficient statistics

The maximum value that the matched filter assigns is important in the detection of the object. Understanding the distribution of the sufficient statistics that result from simulating the main algorithm several times is vital to setting appropriate threshold values for the detection of objects- setting a threshold too high causes no targets to be detected, while setting a threshold too low allows false detections. Since the output of the beamformer is normalized, a comparison can be made with arrays of varying sizes. Figure 23 shows histograms of each array size.

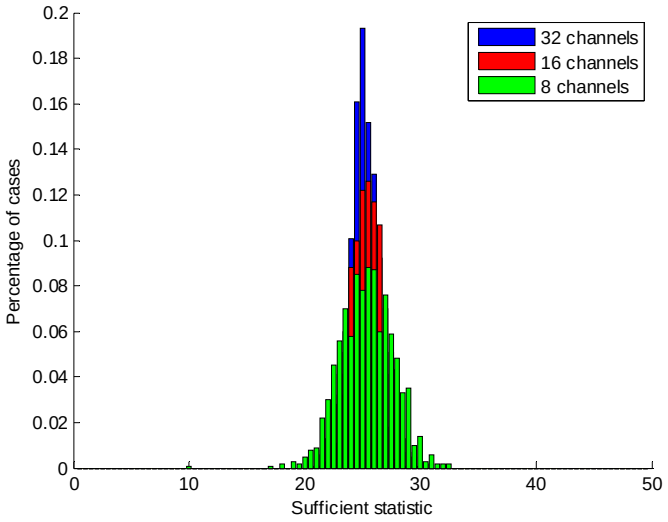

Figure 23: Histogram of sufficient statistics, SNR of -16 dB

The mean for these arrays is very similar when the noise level is held constant, because the beamformer output is normalized. It can be seen that smaller arrays cause a higher standard deviation of the sufficient statistic, since the white noise component of the signal has more weight when using smaller arrays, resulting in higher variances of the output of the matched filter, as seen in Figure 24:

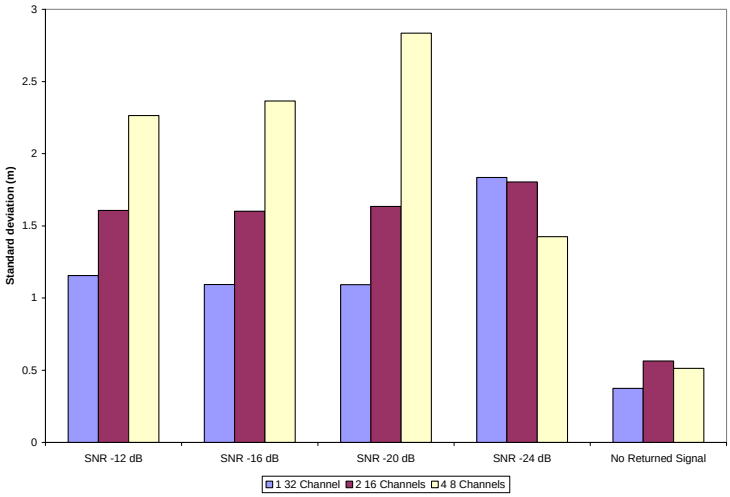

Figure 24: Standard deviation of sufficient statistics given array size and signal strength

This also causes the sufficient statistics for cases with no returned signal to rise, since non-correlated noise does not cancel itself out as much with fewer channels. Figure 25 displays the histograms of simulations with no returned signal with a differing number of channels in each array:

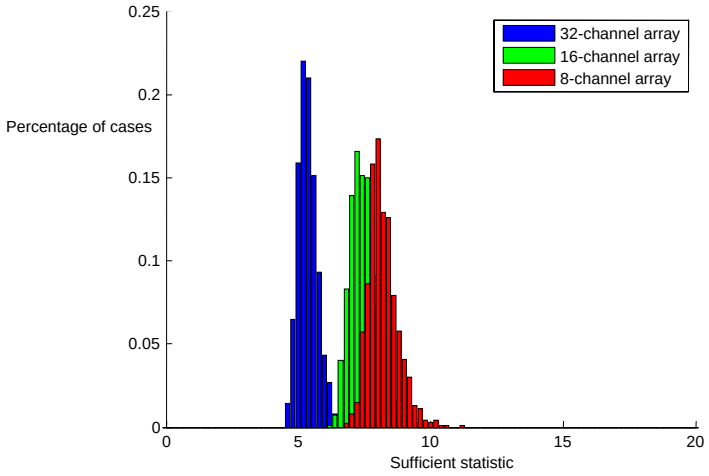

Figure 25: Comparison of noisy sufficient statistics across different sized arrays

However, when the number of sensors is held constant, the mean increases when the signal strength increases. This should be expected as the matched filter multiplies and sums together higher values in the returned echoes, resulting in higher sufficient statistics. Figure 26 shows this, as well as Figure 27:

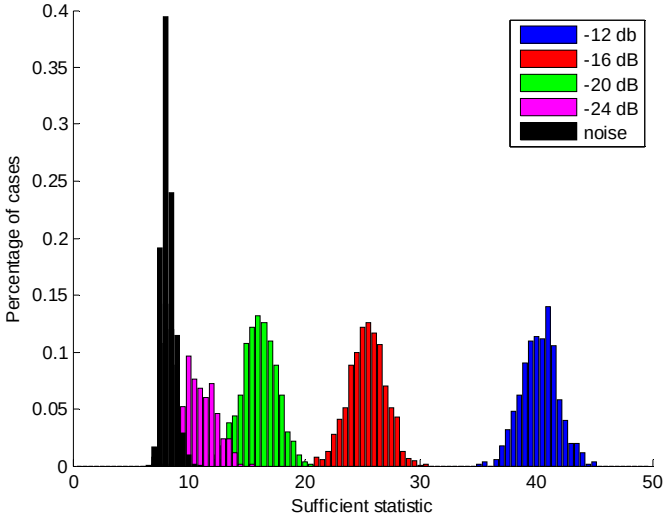

Figure 26: Histograms of 16-channel array sufficient statistic

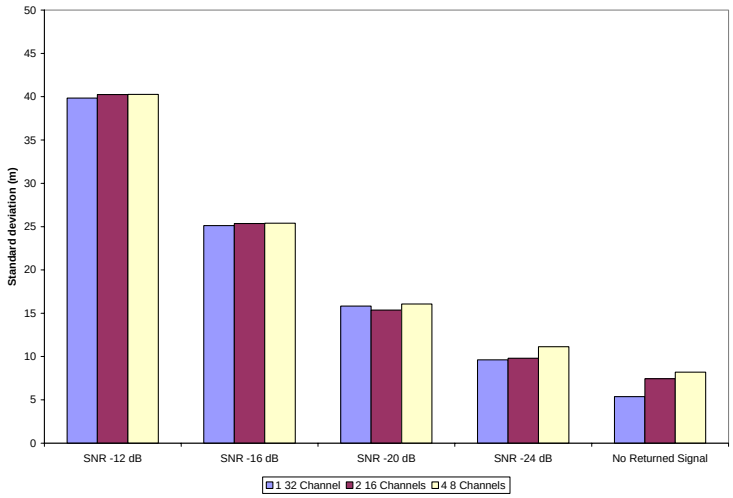

Figure 27: Averages of sufficient statistics with a given signal strength and array size

Once these statistics have been calculated, using a Gaussian distribution with the mean and standard deviation derived from the MATLAB simulations can determine the receiver operating curves (ROCs). The ROC measures of the likelihood of a false detection vs. the likelihood of a true detection. The curves can be drawn in two different ways. One method holds the signal strength constant with varying numbers of sensors, while another method varies the signal strength with the same number of sensors in an array. Figure 28 displays the ROCs for a 32-channel, a 16-channel and an eight-channel array:

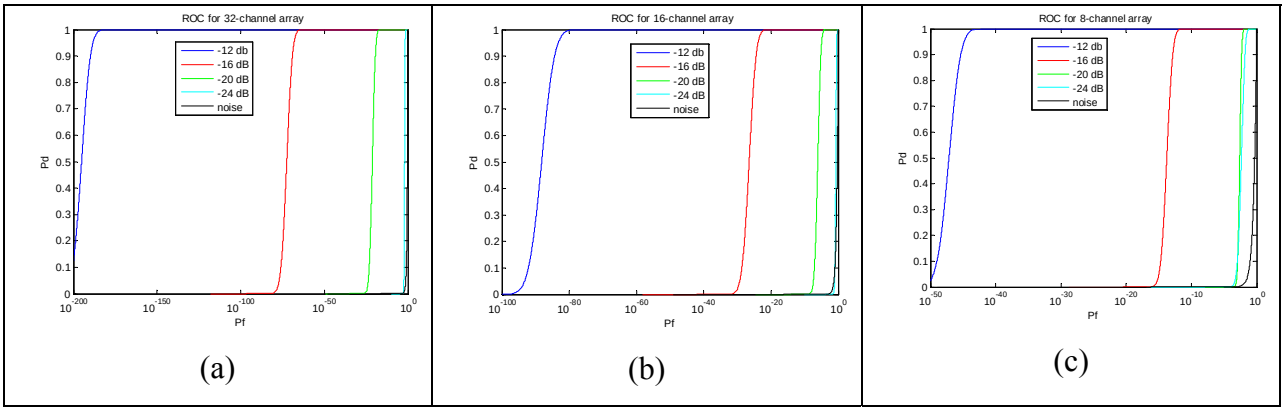

Figure 28: ROCs for one array's detection of one target, with the array size of (a) 32 (b) 16 and (c) 8 channels

As expected, the louder signals created lower false detection rates for the same true detection rate. The larger arrays also outperformed the smaller arrays when the signal strength was held constant.

2.3.2 Results using fused data

In this section, the three different networks are tested to see whether it is better to have fewer large arrays or more smaller arrays while maintaining the same number of total sensors using the algorithm discussed previously. While the previous section showed that the larger arrays work better individually, having more arrays available will enhance the performance of that network since the data from each array can be fused together. This section will show the results of a network consisting of one 32-channel array, two 16-channel arrays and four eight-channel arrays with varying signal strength at each sensor. The three fusion methods discussed previously will be employed and compared to determine the performance of the networks on one target, and then multiple targets.

2.3.2.1 Parameter estimation on one target

The first step in determining the performance of these networks is to use them on one target. This section will compare the performance using one array to determine if using multiple arrays instead of just one array of a particular size enhances the performance. This section will employ all three fusion methods discussed previously, and the best method will be used to determine the performance of arrays when multiple targets are within the sensor field. In these simulations, only the beamformer output with the highest psd in the expected frequencies of the echo was matched filtered.

2.3.2.1.1 Parameter estimation using transformation

As discussed previously, the transformation method uses the estimated AOA from the beamformer and the sufficient statistics from the matched filter of a secondary array to transform that output to get an estimate of the output of a reference array. The key to making this method work is accurate estimations of the parameters being estimated, particularly the beamformer so that the high sufficient statistics from the matched filter multiply constructively, resulting in relatively much higher sufficient statistics that are multiplied together. Figure 29 displays the results of using this method on one target, which states the errors in location encountered when using a specific network:

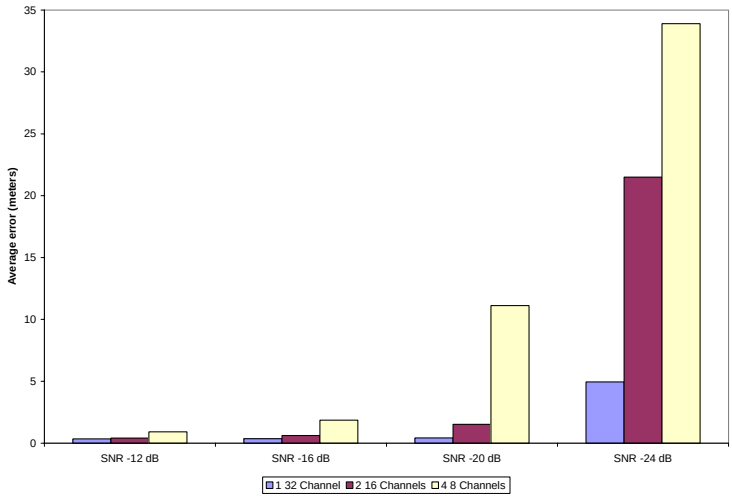

Figure 29: Errors in meters of position estimation using transformed range-Doppler maps

These values are all about the same as those found in Figure 21, which displays the average errors found by the individual arrays. In the one 32-channel array, the values are the same since no fusion is taking place. The reason behind this similarity in the MANs is due to the errors in parameter estimation, which cause the peak in sufficient statistics to be misaligned in the range-Doppler map, and therefore not multiply constructively. This causes a discrepancy in parameter estimation amongst the arrays, which can be seen in Figure 30. There is slight disagreement in the range-Doppler maps, which causes two distinct peaks:

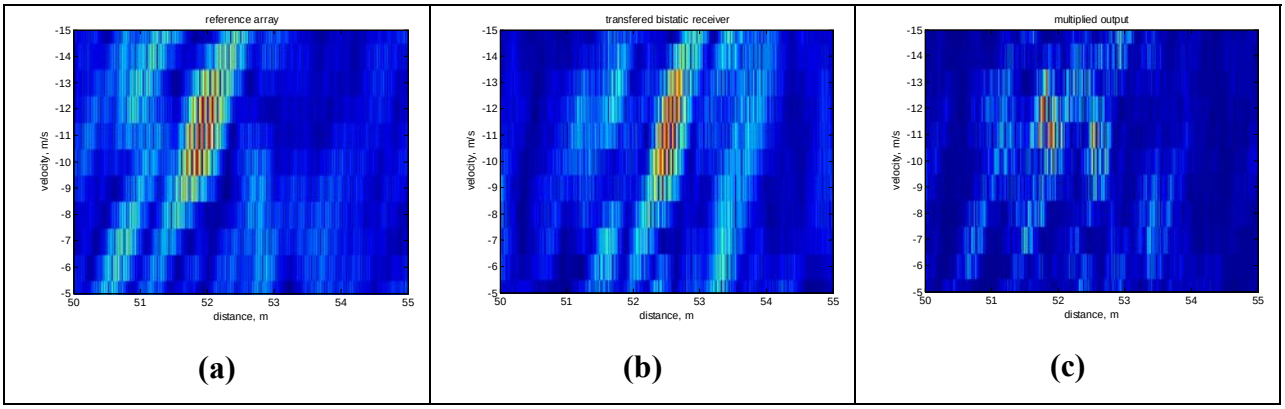

Figure 30: Example of reference arrays output (a) and a secondary array's transformed output (b) being fused together with a smaller error in parameter estimation (c). SNR= -16 dB

This effect is more pronounced in the four eight-channel MAN, since parameter estimations on arrays with fewer channels are less accurate, which leads to less accurate estimations for position and velocity, which causes multiple non-coherent estimates of a target's location.

These multiplied sufficient statistics determine the ROCs for the network. Since the sufficient statistics are multiplied, the multiple array case does better then using just one array because another array might detect an object the other one does not, leading to enhanced detection. Figure 31 shows the ROCs for the different networks:

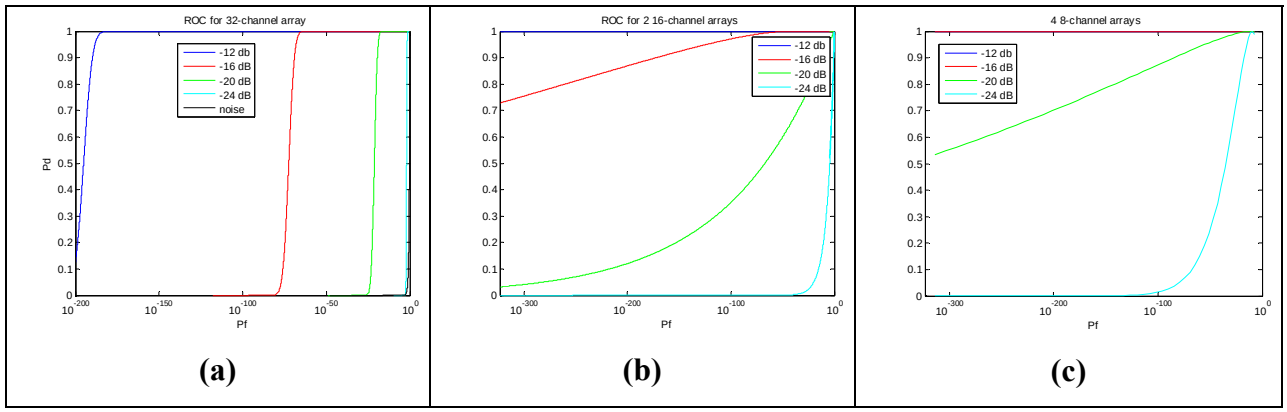

Figure 31: ROC curves for 1 32-channel network (a), 2 16-channel network (b), and 4 8-channel network (c).

A gamma distribution characterized the distribution for the fused output. The mean and standard deviation from the simulated cases were found and a gamma distribution based on this data determined the ROCs. By looking at these plots, it is clear that the four eight-channel network has the best detection rate of any of the networks. However, given the magnitude of a false alarm probability with very little loss in detection probability, it can be assumed that any of these networks are reliable at detecting objects with an SNR of -20 dB.

2.3.2.1.2 Parameter estimation using Gaussian curve method

The second fusion method investigated uses a grid and a Gaussian curve to determine the location of targets. This fusion method uses a threshold on the sufficient statistic to determine if a target is at a certain bearing angle and distance. This method was developed to counter the problem seen in Figure 30, where small errors in parameter estimation cause errors in parameter estimation. Using a Gaussian curve corrects this problem, as it would allow for high values that are close to each other to still be multiplied constructively, as seen in Figure 14. Figure 32 shows the results using this method using one target:

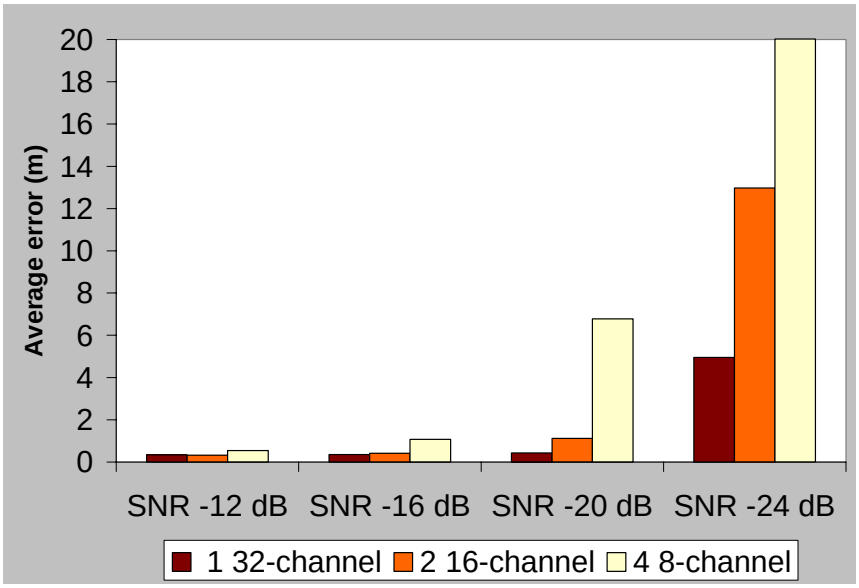

Figure 32: Average errors in meters for position estimate of Gaussian curve method with given network and signal strength

Using this method lowers the average errors seen in Figure 21 for the signal array networks, and lowers the values seen in Figure 29, particularly for the -24 dB SNR case. The method described in 2.1.4.2 estimated the velocities of the target, and Figure 33 shows its errors in velocity measurement:

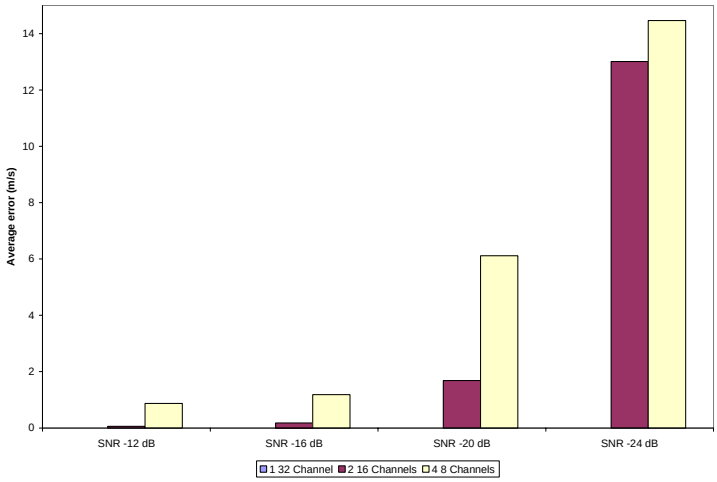

Figure 33: Error in velocity estimation with given network and noise level

Since the velocity measurements are in 2-D space in these arrays, a comparison with the relative velocities measured in Figure 22 cannot be drawn. This is also true for the one 32-channel array network, since it cannot measure a complete 2-D velocity by itself. Once again, the network with larger arrays outperforms the networks with smaller arrays.

2.3.2.1.3 Parameter estimation using weighted average method

The final fusion method investigated in this paper is the weighted average method. This method takes estimates of the target's location and assigns weights on them based on their sufficient statistic. Figure 34 shows the average error of location estimation when this method fuses the estimates:

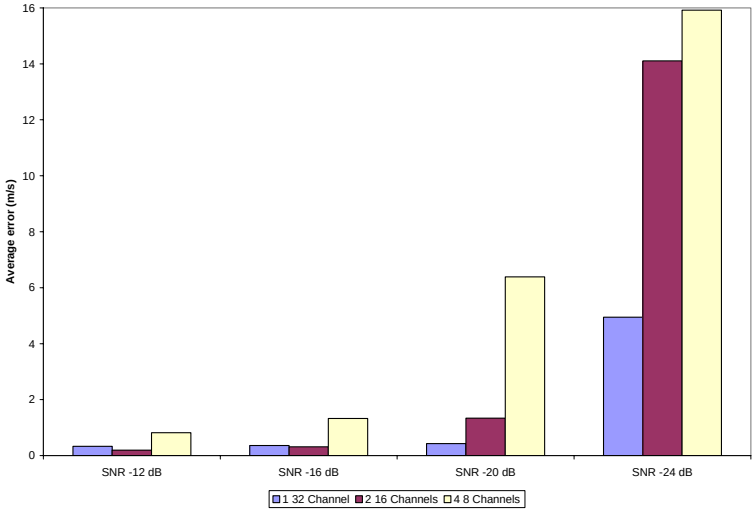

Figure 34: Average errors for position estimate of weighted average with given network and signal strength

In all cases, these values lower the average error when using just one array in the network. These errors were also all lower using this method instead of the transformation method, as seen in Figure 29. When compared to the Gaussian curve method, it has lower errors when the MAN contains two arrays with a stronger signal and when four arrays are used with a weaker signal.

Since the method of finding the velocity was the same using this method and the Gaussian curve method, no comparison can be made on the velocity estimate.

2.3.2.2 Parameter estimation in multi-target scenarios

The previous section investigated the performance of networks using multiple arrays. This was done using three different fusion methods. Since the weighted average and Gaussian curve methods performed better then the transformation method, just these two methods will be used in this section. The algorithm used will change slightly, as now every angle beamformed has its output matched filtered, with the angle that corresponds to the highest sufficient statistic at each time sample and velocity stored.

Using the three networks, MATLAB simulated a scenario with two targets present 250 times, followed by 250 more simulations using three targets. The estimated ranges, bearing angles, relative velocities and sufficient statistics determined by the individual arrays were stored, and both data fusion methods use the same data to make location and velocity estimates of all the targets in the sensor field. In these simulations, instead of a constant signal strength across all arrays, the distance the sound traveled determined the strength of the signal, as seen in (36).

2.3.2.2.1 Location estimation in multi-target scenarios using Gaussian curves

The Gaussian curve method as described earlier was employed to estimate the position of the targets. Figure 35 displays the results of the error in position for the two target case using just one array and Figure 36 shows the average error using the three target case:

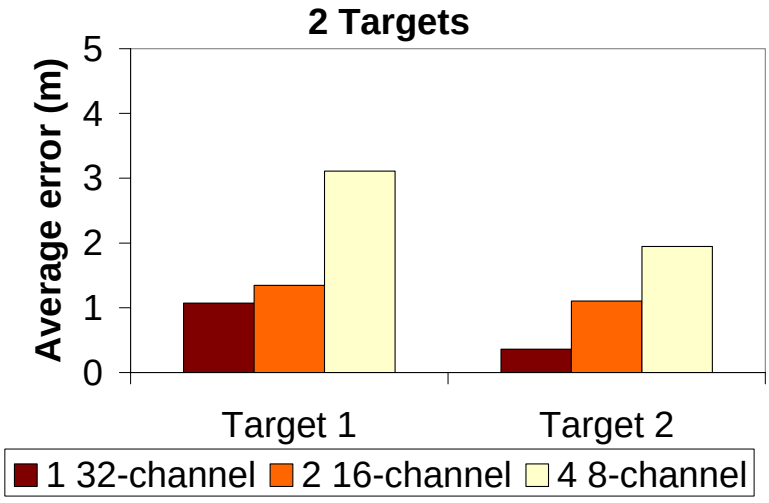

Figure 35: Average error in meters of location using one array on two target scenario

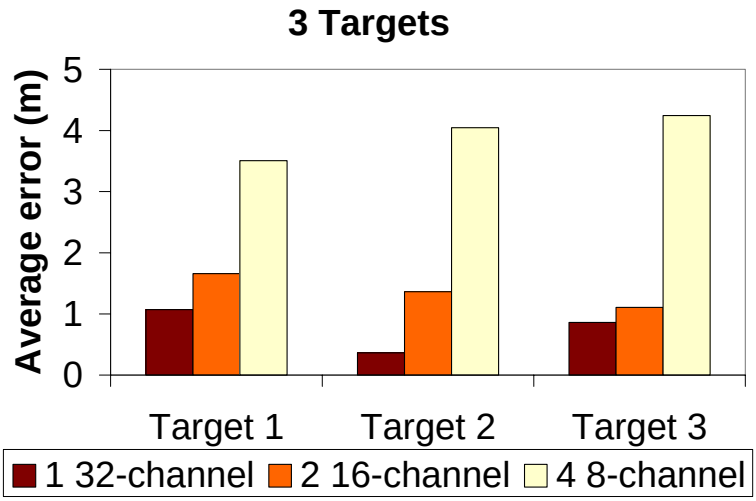

Figure 36: Average error in meters of location using one array on three target scenario

These values will be used to compare the effectiveness of the fusion methods.

After each array determined the number of targets within the sensor field and their approximate location and relative velocities, the arrays send these parameters and the sufficient statistic for each target to a fusion center that makes universal parameter estimates based on these parameters. Figure 37 shows the results for their location estimate using the Gaussian curve method:

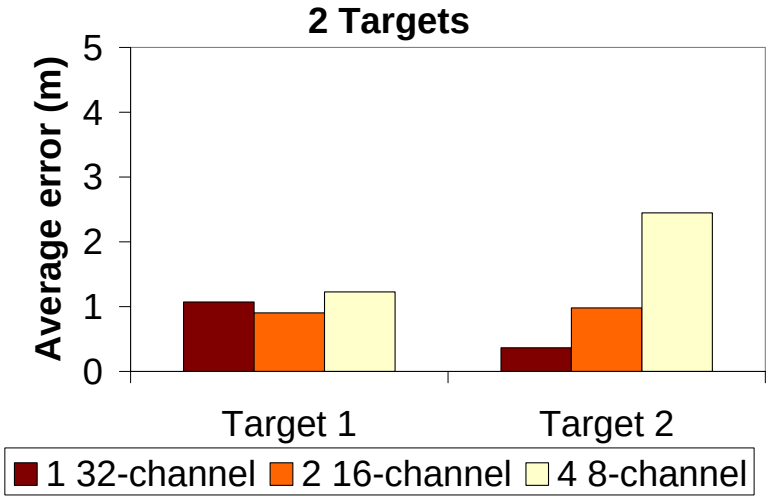

Figure 37: Average errors of fused location estimate using Gaussian curve method

These values are lower then the single array case, except for the estimation of the second target's location. Figure 38 shows the reason for this, since the wider Gaussian curves interfere with each other:

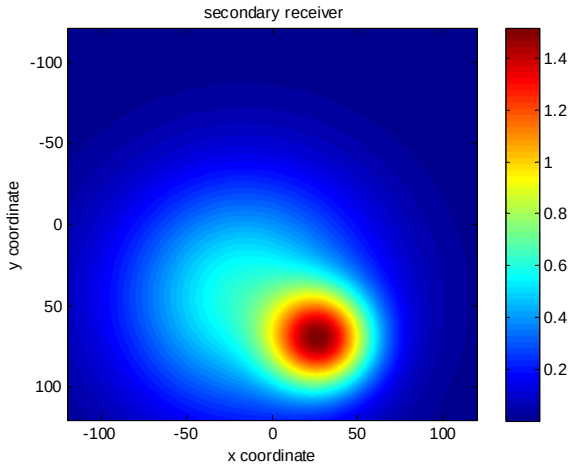

Figure 38: Example of one array's Gaussian curves with two targets

As shown in this figure, the wider Gaussian curves interfere with each other, therefore causing the peaks from one target to potentially dominate the peak from another target. This causes the location estimation of the target to be skewed. When all 250 locations for the second target are plotted with the estimated position derived from four arrays, this effect is clearly seen, as in Figure 39:

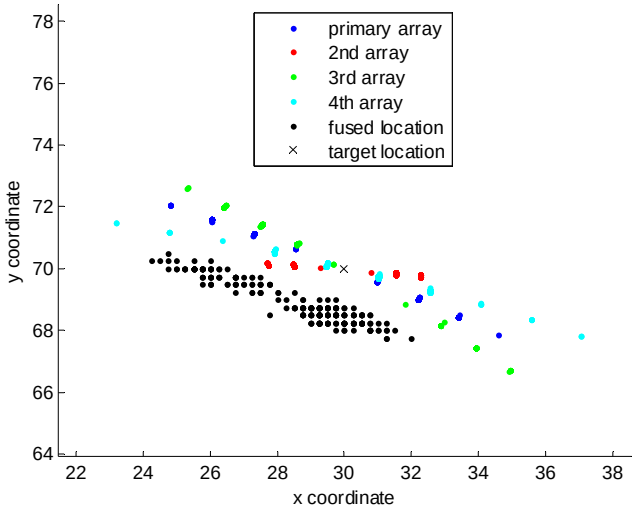

Figure 39: Plot of target location with estimates as determined by Gaussian curve fusion and each individual array

Adding a third target enhances this effect, as the third target in the four eight-channel network is not seen in any of the 250 simulations in the network. The bias seen here is in the direction of

the other target, and is fairly consistent across all the simulations. The estimates on the other two targets also have their average errors increased, as Figure 40 shows:

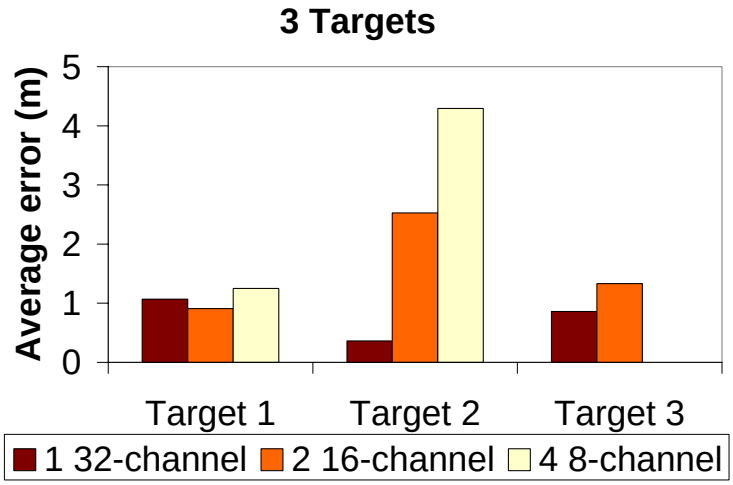

Figure 40: Average error using Gaussian curves on three targets

Since this method increases the errors incurred by individual arrays, this method is not a feasible method to estimate the location of these targets. Figure 41 displays an individual array's Gaussian curve values. Notice how the two weaker peaks are indistinguishable compared to the dominant peak:

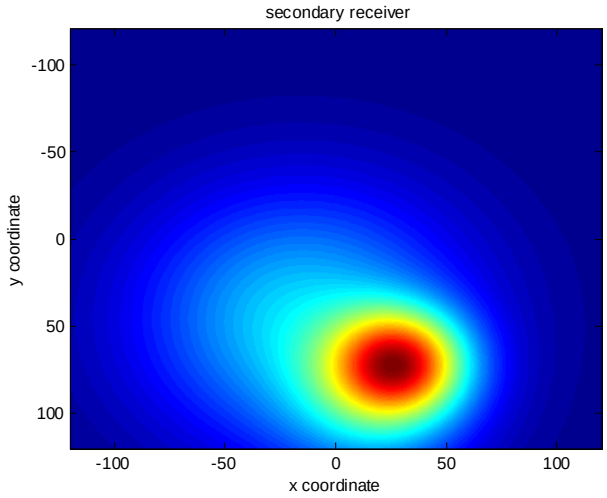

Figure 41: Gaussian curve output for single array with three targets

2.3.2.2.2 Location estimation in multi-target scenarios using weighted averages

The remaining method of data fusion is the weighted average method, where the location of the target as estimated by each individual array is weighted according to its sufficient statistic.

Using the same target locations as found in the previous section, the weighted fusion method determined a universal location estimate, and Figure 42 shows the results with two targets:

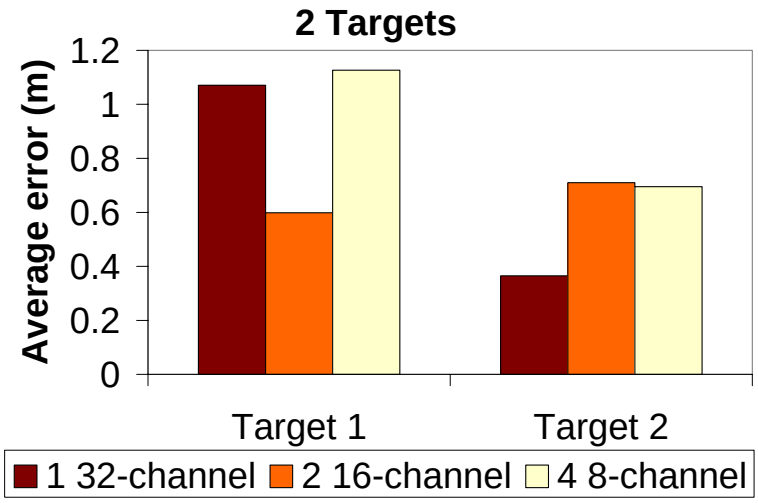

Figure 42: Average errors in meters of fused location estimate using weighted average method

Using this method lowered the average errors found using the Gaussian curve method, and lowered the errors found in the individual arrays. Figure 43 shows the estimates of location for the second target.

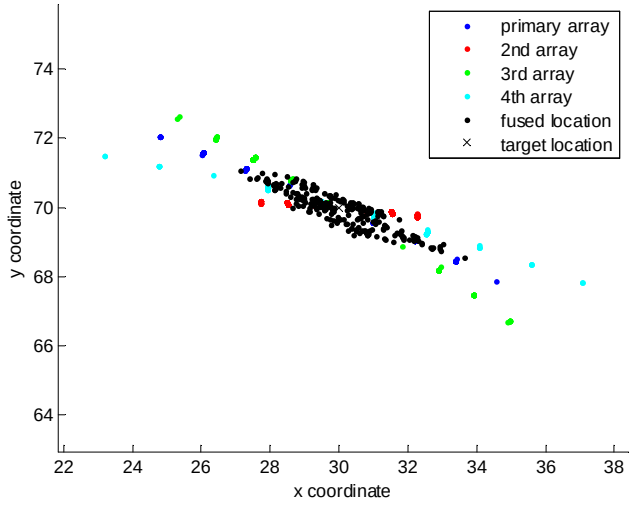

Figure 43: Plot of target location with estimates as determined by weighted average fusion and each individual array

As can be seen, these locations show little or no bias in relation to the target, especially when compared to the estimations as seen in Figure 39. Using this method with four eight-channel

arrays estimating the second of two targets lowered the average error by a factor of about three and a half.

When a third target is introduced, the location estimates can be seen in Figure 44:

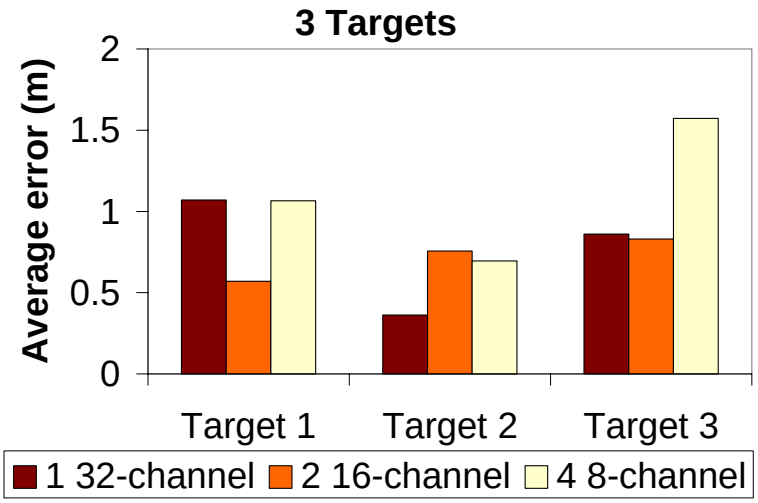

Figure 44: Average error using weighted averages on three targets

Once again, a problem occurs with the third target's location when the four eight-channel network estimates the targets position. In this case, an inconsistency in the detection of the third, and most distant, target causes it not be detected in some simulations. This distance relative to the arrays causes the returned signal to be not as powerful, resulting in 7.6% of the simulations where no target was detected. Also some smaller peaks caused several detections made by the third target, as happened in 7.2% of the simulations. Figure 44 shows the error of the estimate of the cases where that target was identified. After accounting for this potential error, this method lowered the average error of the third target by a factor of about three and a half.

The other networks detected the third target with no false positives in all 250 simulated cases.

2.3.2.2.3 Velocity estimation in multi-target scenarios using Gaussian curves

The velocity of each target was also estimated using Gaussian curves. Figure 45 shows the results of the two target case:

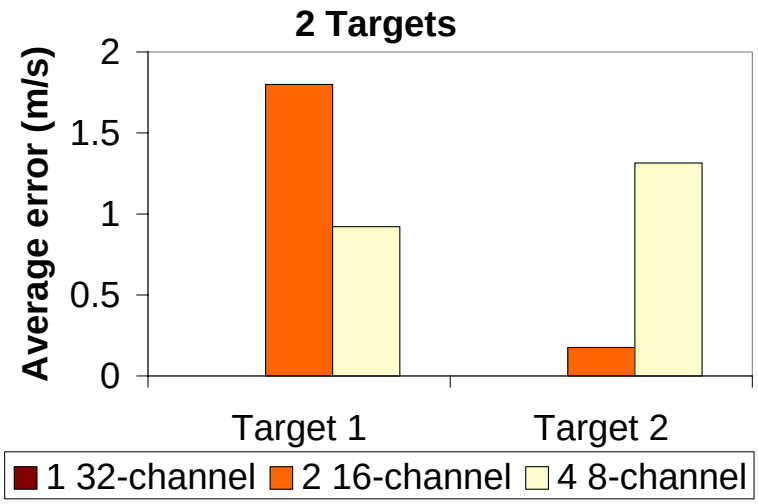

Figure 45: Errors in velocity using Gaussian curves method on two target scenario

The increase in the second target's error when more targets are added can be attributed to the other sensors not multiplying their outputs constructively. The greater distance of those arrays, as well as differing estimates of relative velocity, cause the estimation on one array not to be as robust with fewer channels causes this non-constructive multiplication. However, in the case of the first target, the estimates from the individual arrays were good enough to be multiplied coherently, resulting in an enhanced estimation for the first target's velocity.

This algorithm was also used in the three target scenarios, and Figure 46 shows the average errors in velocity:

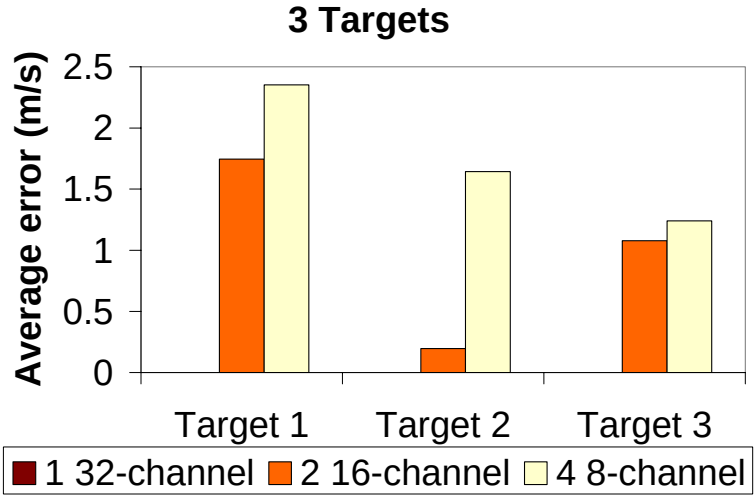

Figure 46: Errors in velocity using Gaussian curve method on three target scenario

The errors for the three target case resemble those of the two target case, with the exception of the first target in the four eight-channel array network. Interference from the other targets could cause this, which would create errors in the individual array's estimation of velocity for that target. The two 16-channel arrays had smaller errors then the four eight-channel array networks in all three targets.

2.4 Conclusions

This chapter investigated three different fusion methods using the outputs from all the arrays in order to make universal parameter estimates as well as universal target detection decisions. The average error over 250 simulations with three different networks tested the quality of these methods. This section will also compare the performance of the three networks, and offer potential application of this algorithm.

2.4.1 Comparison of fusion methods

The first method, the transformation of sufficient statistics, offers lossless data processing; no thresholding takes place and all sufficient statistics found by the matched filter get fused together. This leads to better object detection using this method as opposed to the other methods that use thresholds to make detection decisions, since the thresholding eliminates possible objects as well as rejecting false positives. Finding the peaks of the fused Gaussian curves limited the amount of targets that can be found, as was seen when a third target was introduced. This method could also have problems with targets in close proximity to each other, as the peaks that are formed from each object could be put together in such a way as to create one dominate peak with other targets not able to create peaks, as displayed in Figure 41. Using the weighted average method improved the detection of the objects, as long as all the arrays in the network were able to positively identify the correct number of targets in the network's field.

When estimating the parameters of the objects, the transformation method did not significantly improve the estimation of an object's location when compared to a single array used in that network. This is due to the effect seen in Figure 30, where the transformed outputs from the two arrays did not match exactly, thereby creating an instance where whichever array had the higher sufficient statistic would determine the universal detection decision. In cases where the output from the two arrays would multiply constructively, the estimate of one array would be the

same as the estimate of the fused data, therefore not improving the estimate of the target's parameters.

Using the Gaussian curve method offers some improvement to the individual array's performance, as long as the curves for each target do not interfere with each other. The effects of this interference are shown in Figure 39, where another target has skewed the estimates for that target in its direction. Another danger of using this method arises when one array detects one target, while another array detects a different target. When that happens, it results in an estimation of one target estimated in between where the two targets. This is usually caused by setting the threshold of detection on one array too high, so that there are missed detections of targets in the networks field.

The weighted average case also improved the estimates of objects over using just a single array, as well as the Gaussian curve method. Using this method does have some flaws, however. If two targets were close enough together, they could be considered the same target as they would be close enough according to the proximity test. It also gives arrays with weaker signals a chance to skew the universal decision if the array gives a poor estimate, and could lead to false positives if the point fails the proximity test. However, it exhibits little or no biases compared to other targets, and allows for stronger signals to have more input on the determination of a target's estimated parameters, thereby creating the best estimates of position of the three methods discussed in this work.

2.4.2 Comparison of networks

This chapter used three networks to detect objects and estimate their parameters. These three networks used 32 sensors, placed in one array, two arrays or four arrays. Using the transformation method, Figure 31 shows that using four eight-channel arrays detected objects the best. However, one 32-channel array estimated the parameters the best using that method. This tradeoff is due to the matched filter's output not multiplying constructively in all cases. Using more arrays detects targets better because one array's output for a specific distance and velocity could be high enough to cause a positive detection decision, even when the sufficient statistics at other arrays are not.

Using the weighted average method to compare the networks, the one target instances that used the one 32-channel array network performed the best when the signal strength was

weak, with the lowest average error when the SNR was -20 dB or less. When the SNR exceeded than that, the two 16-channel arrays had the lowest error, as shown in Figure 34. Using multiple targets with non-constant signal strength, Figure 42 and Figure 44 showed that the two 16-channel array had the most consistent accuracy across all targets. While using one 32-channel array can give better estimates of location at times, it cannot estimate the complete velocity of the target, only its relative velocity. Using multiple arrays allows for the estimation of the target's complete velocity, as long as two arrays and the target are not collinear. Using two 16-channel arrays also did not have any false positives or missed detections of the third, more distant target in the three-target simulations, while the four eight-channel array network either did not detect the target or falsely identified targets about 15% of the time. The velocity measurements were also the best using the two 16-channel array network.

2.4.3 Potential application of this algorithm

Any multi-arrayed network could use the algorithm described in this chapter. Each array could perform the beamformer and matched filtering on separate platforms, one for each array. However, the real use of this method stems from the low amount of data each array would send to the fusion center. The arrays would send only the bearing angle, distance, relative velocity and a sufficient statistic for each target. In cases where the position and orientation of an array are known, the arrays could send location estimations to the fusion center. This is of particular importance since as little data as possible needs to be transmitted because of low data transfer rates underwater. If entire range-Doppler maps are to be transmitted, the arrays should employ a compression scheme such as MPEG-4 to effectively transmit this data.

The adjustable threshold concept can also very quickly be implemented by testing the SNR of the chirp at a known distance. A different SNR at a different distance could serve as a benchmark for thresholding, just as this work uses a 75 dB SNR at a distance of one meter. This should also be noted when determining the possible range for a sonar network, as the matched filtering process will not distinguish a signal that has traveled far from Gaussian noise.

3 Uncertainty analysis of MAN using passive sonar

The previous chapter investigated the performance of different MANs with certain array positions and orientations using active sonar. This chapter investigates the performance of two different MANs using passive sonar with variances in array position and orientation. One MAN consists of two 32-channel arrays, while the other consists of eight two-channel arrays. These MANs estimate the position of a sound source placed at different locations, with a variance of each array's position and orientation. MATLAB programming simulated returned data across all sensors, and then beamformed the data using the beamformer algorithm discussed in the previous chapter. These estimates of the AOA at all the arrays determined the universal location estimate of the sound source.

3.1 Experimental set-up and procedure

In order to test the performance of each MAN, MATLAB simulated the data gathered by arrays whose position and orientation were subjected to various uncertainties based on the final location of the array and the sound source. These simulations used only one sound source in all cases. This section describes how MATLAB created the data used in the simulations, and how the MAN made location estimates.

3.1.1 Placement of arrays and sound source

In order to test these MANs, a grid measuring 100m by 100m was created with 10m in between each point, for a total of 121 locations at which the sound source was placed in the simulations. The simulations in this chapter used only one sound source in all cases. Figure 47 shows the grid along with the positions of the arrays used during the simulations:

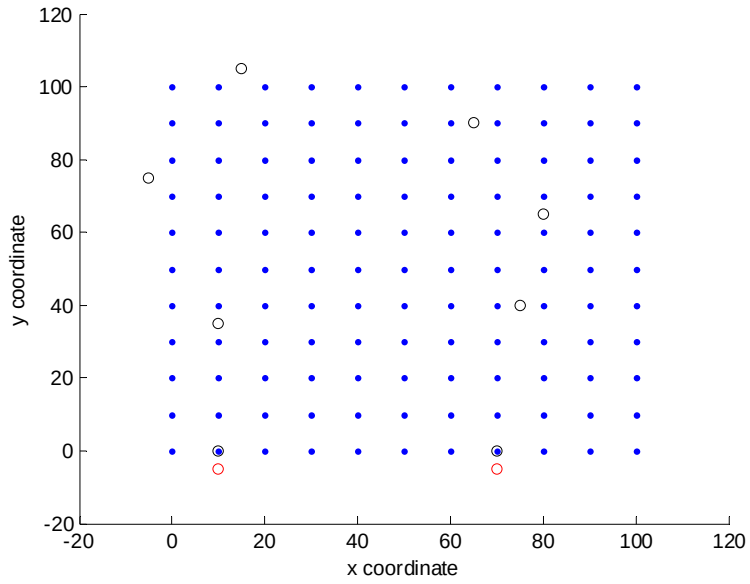

Figure 47: Location of sound source (blue dots) and two 32-channel arrays (in red) and eight two-channel arrays (in black)

The two arrays used in the two 32-channel MAN had an orientation of 0º, and were placed at (-5, 10) and (-5, 70). Table 3shows the position and orientation of the arrays in the eight two-channel array MAN:

Array #	1	2	3	4	5	6	7	8
Position	(10,0)	(70,0)	(75,40)	(80,65)	(65,90)	(15,105)	(-5,75)	(10,35)
Orientation	0^{o}	0^{o}	75^{o}	60^{o}	180^{o}	195^{o}	240^{o}	270^{o}

Table 3: Position and orientation of arrays in the eight two-channel array network

The variance in position was simulated in 3 meter increments in both the x- and y-directions, starting with no variance and going to a maximum of 15 meters. The variance in orientation stated at no variance, and went to a maximum of 25 degrees in 5 degree increments. This made 36 total cases, and the average and median errors were recorded at all points during the 100 simulations.

3.1.2 Data simulation using MATLAB

Similar to the last chapter, MATLAB simulated the data using the positions of the arrays and the sound source. In instances of uncertain position, a random number was multiplied by the appropriate variance and added to the assumed coordinates. A similar procedure added the

variance in orientation, with the appropriate variance multiplied by a random number and added to the assumed orientation of the array. In order to find the location of each sensor after the variances were applied to their location, (38) determined its location:

$$(x_s, y_s) = (x_a + \sigma_x r + s\cos(\theta_a + \sigma_\theta r), y_a + \sigma_y r + s\sin(\theta_a + \sigma_\theta r)) \quad (38)$$

In (38), (x_s, y_s) is the location of the sensor after the uncertainties have been added, (x_a, y_a) is the assumed location of the array, θ_a is the assumed orientation of the array, σ is the variance added to either the position or orientation, and r is a random number with a Gaussian distribution with a mean of zero and a variance of one.

Since the sound emitted by the source was modeled as being continuous, no time delay was programmed into the arrays due to the distance from the source to the center of the array. However, the simulated data included a phase difference relative to the first sensor of the first array. The distance between the sound source and simulated sensor determined this phase difference, using (39):

$$d_s = \sqrt{(x_s - x_{ss})^2 + (y_s - y_{ss})^2} \quad (39)$$

In (39), (x_{ss}, y_{ss}) is the location of the sound source. After (39) calculates the distance from the source to the sensor, (40) calculates the phase angle relative to the first sensor on the first array:

$$\phi_s = \frac{2\pi(d_s - d_1)f}{c} \quad (40)$$

In (40), f is the frequency emitted by the sound source, and d_1 is the distance from the first sensor of the first array to the sound source. With this phase angle now known, a complete equation for the modeled data, as seen in (41), can be written:

$$A(t) = \sin(2\pi f_c t + \phi_s) + n(t) \quad (41)$$

Every sensor has this signal computed for 1000 time steps at each position, with the emitted frequency set to 6600 Hz, with an SNR of 0 dB regardless of distance between the sensor and the sound source, and once again *n(t)* represents a noise term with a Gaussian distribution with a zero mean and a variance of one. The sampling frequency was set to 50 kHz, and MATLAB performed 100 simulations at each location at on the grid for all 36 cases of variance.

3.1.3 Determination of sound source estimation

The same beamformer algorithm as described in the previous chapter determined the AOA at each array. Using triangulation, two arrays estimated the sound source's location by the determined the intersection point of their AOAs, with the arrays in their assumed positions and orientations. For the two 32-channel array MAN, the one intersection of the beams determined the universal location estimate. In the eight two-channel array MAN, there are 28 such intersections. The median x-coordinate and y-coordinate determined the universal location estimate. In the case of two arrays with parallel AOAs, there is no intersection. If there were any such arrays, this "intersection" was removed from the 28 intersections, and the median of the remaining locations determined the universal location estimate.

3.2 Results of MATLAB simulations

This section investigates the results of the simulations performed by MATLAB. As in the previous chapter, the average errors, along with the median errors, measure the performance of the MAN.

Within this grid, 67 of the 121 points are located in front of all eight arrays when no variance in orientation is introduced. These 67 points form a "core" region, as seen in Figure 48:

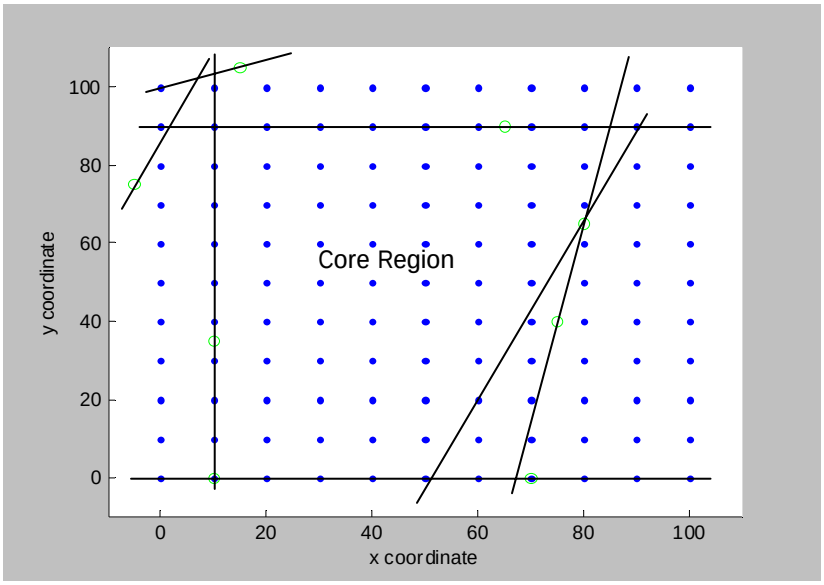

Figure 48: Core region as bounded by orientations of arrays

This is important due to front back-ambiguity, since a linear array has no way of telling if a signal from a given bearing angle is coming from in front of it or behind it. In these simulations, the signal is always assumed to come from in front of the array. This causes location errors in

the area outside the core zone, because some arrays incorrectly assume the target to be in front of them. Because of this effect, this section presents the errors the results four different ways: the overall average error, the overall average of the median error, the core average error and the average of the core median errors. This section applies these four methods to both MANs.

3.2.1 Performance of the eight two-channel array network

When an array makes an estimate of the AOA of a signal, the more channels it has the better its estimate, as seen in Figure 19. Since this MAN uses arrays with fewer channels, the bearing estimates of each array are not as robust as the estimates of the 32-channel arrays. This leads to the 28 intersections being spread out over a large area, as seen in Figure 49:

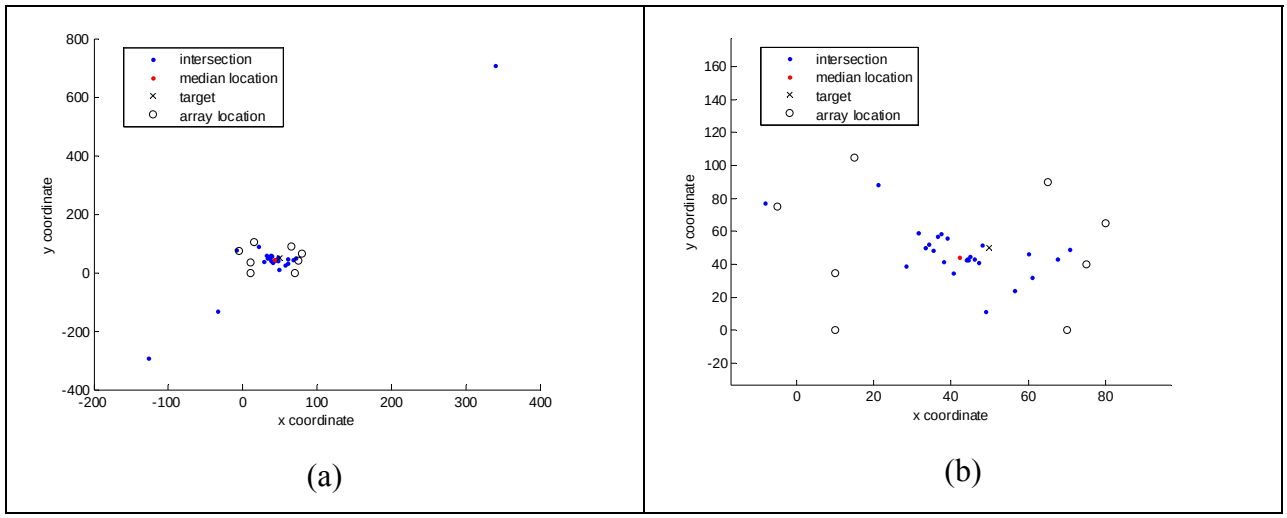

Figure 49: Intersection points between two arrays in eight array MAN with no variances, overall (a) and zoomed in (b)

The median of the 28 intersections was chosen instead of the average because in an average, an extreme outlier out of one of the points significantly skews the overall estimate, whereas an outlier does not affect the median unless a significant number of outliers are present. This leads to lower errors of the position of the sound source.

Once the MAN made an estimate of the sound source using the median x and y coordinates, the error is calculated for all 100 simulations at all 121 grid points. Figure 50 shows the average of the average errors at each point on the grid using the eight two-channel array network with a given variance in position and orientation:

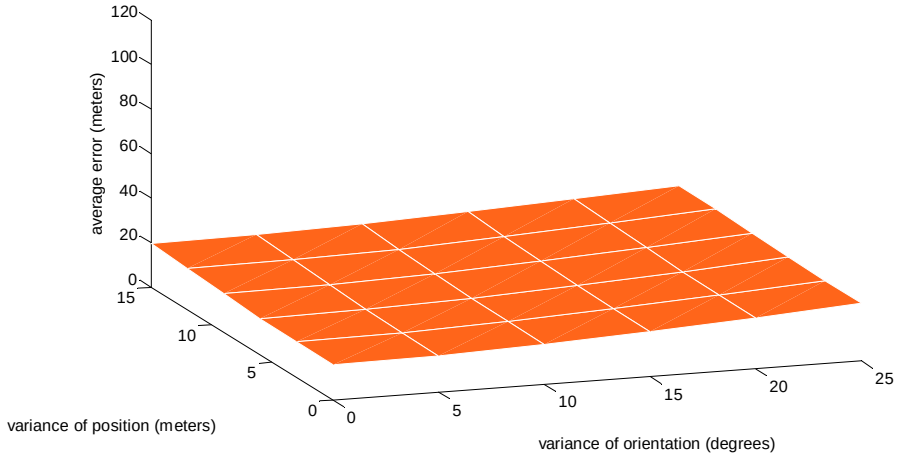

Figure 50: Average of the average errors on all 121 points in the grid using 8 2-channel arrays

Figure 51 shows the average of the median errors of the 100 simulations across all 121 points:

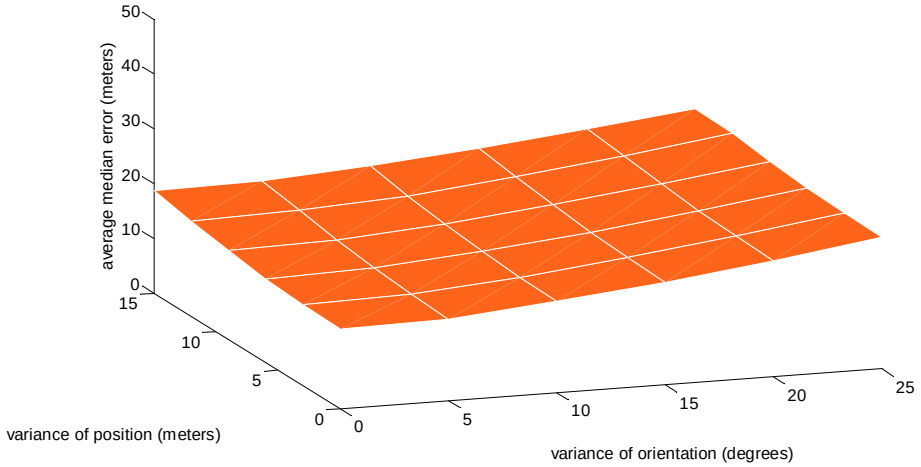

Figure 51: Average of the median errors on all 121 points in the grid using 8 2-channel arrays

Taking this average median error decreases the errors by about 7% in all cases. Extreme outliers cause the decrease, as these high errors skew the results to show higher errors when the average error is used. When the core region is considered, Figure 52 shows the average errors, and Figure 53 shows the median errors:

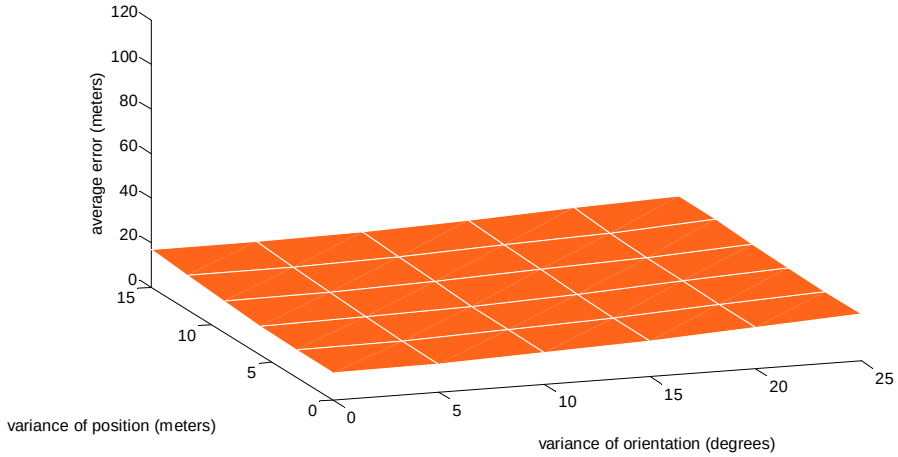

Figure 52: Average of the average errors within core region using 8 2-channel arrays

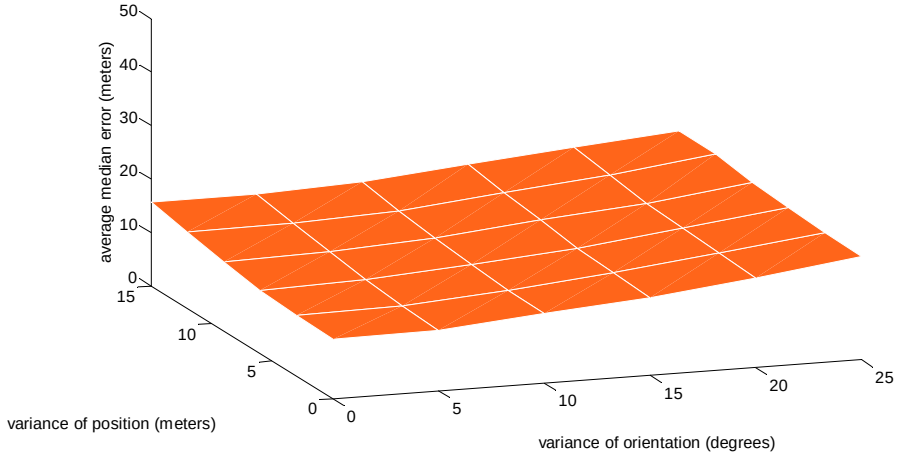

Figure 53: Average of the median errors within core region using 8 2-channel arrays

As expected, these values are lower then the overall average. Once again, taking the average of the median errors slightly lowered all estimates. This MAN appears to be more sensitive to a five degree variance then a three meter variance in both coordinates, although the relationship between variance and average error does not appear to be linear.

3.2.2 Performance of 2 32-channel array MAN

When using two arrays, the arrays from only one intersection which serves as the location estimate of the sound source. Since this MAN uses four times as many total sensors and each array has a much lower error in bearing measurement, this MAN outperforms the eight two-

channel MAN with no uncertainties introduced to the array's position and orientation, as seen in Figure 54, where each MAN locates the target across all 121 points on the grid:

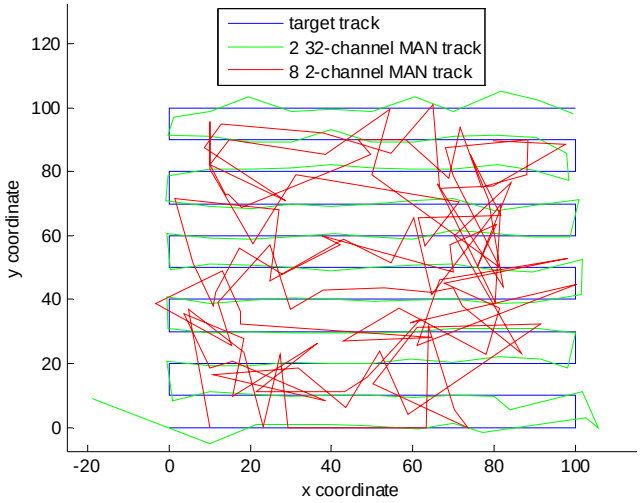

Figure 54: Determined path of sound source by both MANs with no uncertainties in array position or orientation

However, since this MAN uses only two arrays, uncertainties in the position and orientation of those arrays create large errors since this MAN gives only one intersection. The previous MAN has more intersections, thereby enhancing its estimates since outlying intersections have little impact on the overall estimation of the sound source's location. Because of this, the average and median errors increase rapidly when increasing variance is applied to the arrays, as seen in Figure 55 and Figure 56:

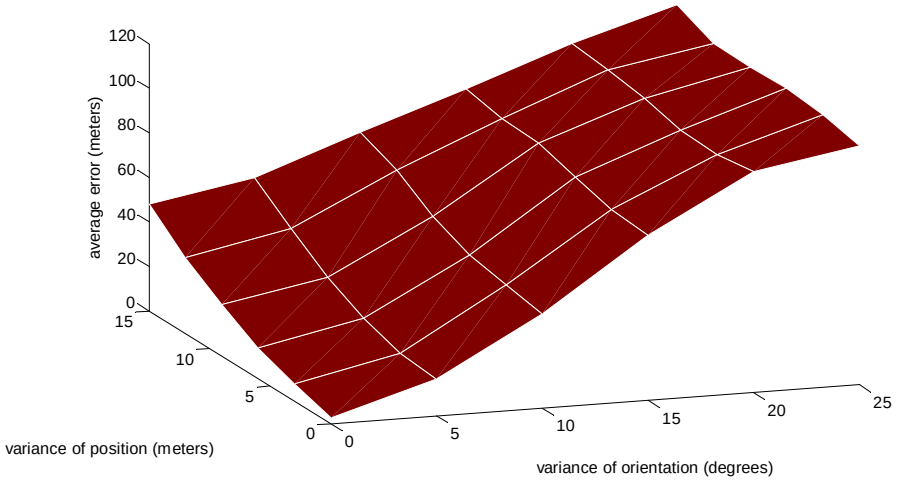

Figure 55: Average of the average errors on all 121 points in the grid using 2 32-channel arrays

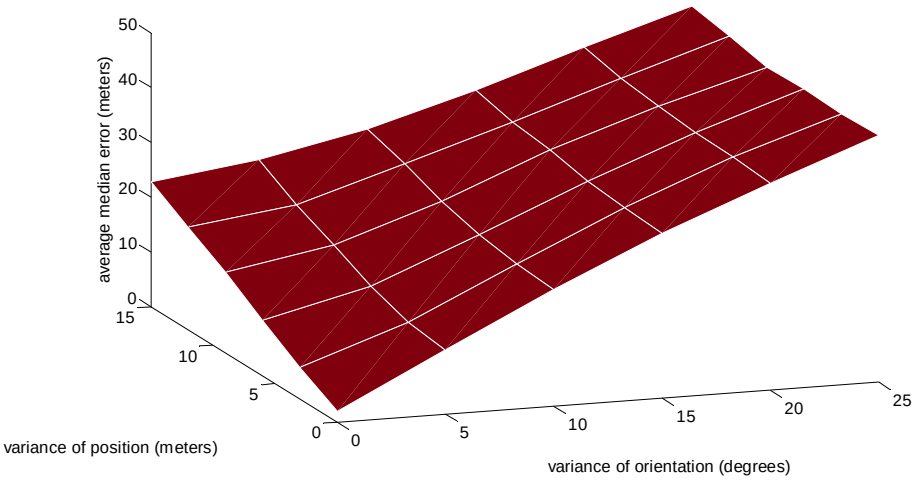

Figure 56: Average of the median errors on all 121 points in the grid using 2 32-channel arrays

The median errors do not increase as rapidly as the average errors because the outliers do not influence this statistic as much. However, these tables show that the errors increase very rapidly when variance is added to the position or the orientation to the arrays in the overall grid.

Even though no front-back ambiguity occurs because of the position of the arrays in this MAN, the performance of this MAN was measured in the core region as described above as well. This region has lower errors because AOAs that are nearly parallel, as some points out the core region would cause, create higher errors in location estimates. This is due to the discretization of the angles- a small change in AOA estimation causes a great change in the location estimate. The beamformer also shows a decrease in performance when the AOA moves toward an endfire case, which also occurs outside the core region. This also allows for a fair comparison between the two MANs, as the same points' errors are averaged in both MANs. Figure 57 and Figure 58 show the average and median errors, respectively:

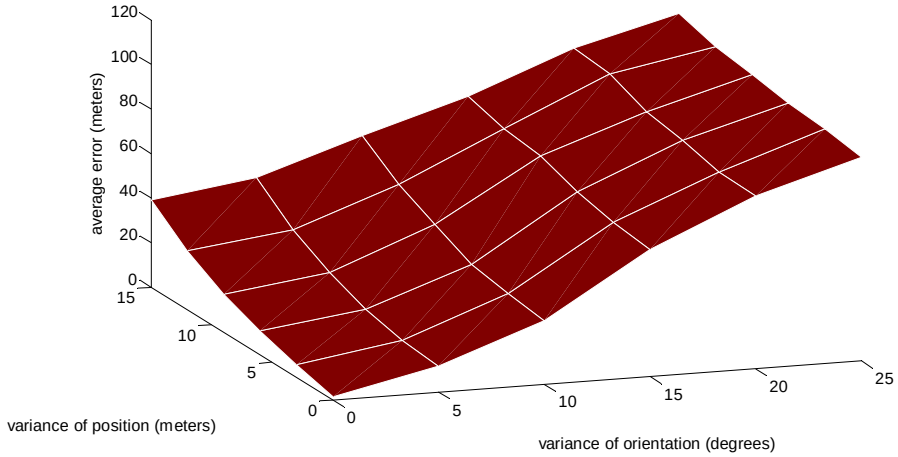

Figure 57: Average of the average errors within core region using 2 32-channel arrays

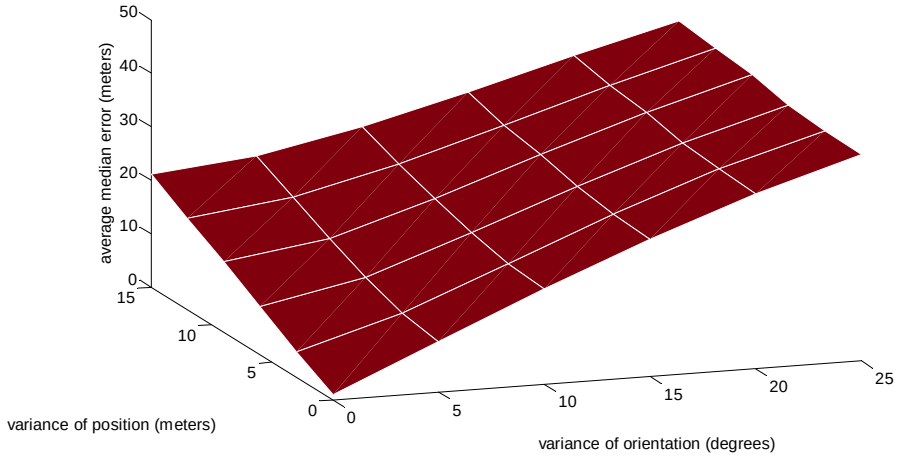

Figure 58: Average of the median errors within core region using 2 32-channel arrays

These errors are lower then the errors in the overall grid by about 12%. The error increases very rapidly, using either the median or average error, when the variance in position or orientation increases. Again, these errors do not seem to have linearity to them, with this MAN being more sensitive to a five degree variance then a three meter variance in position in either direction.

3.2.3 Comparison of the two MANs given same uncertainties

With the known results of the simulations, the two MANs can now be compared against one another. While the two 32-channel MAN offers better performance when the position and orientation of the arrays have little or no variance, the eight two-channel MAN has lower errors when the position and orientation of its arrays have a variance introduced to them. When comparing the average errors, Figure 59 shows which MAN had the smaller average error, and

Figure 60 shows which MAN had smaller median errors. This comparison was the same in either the overall grid or the core region.

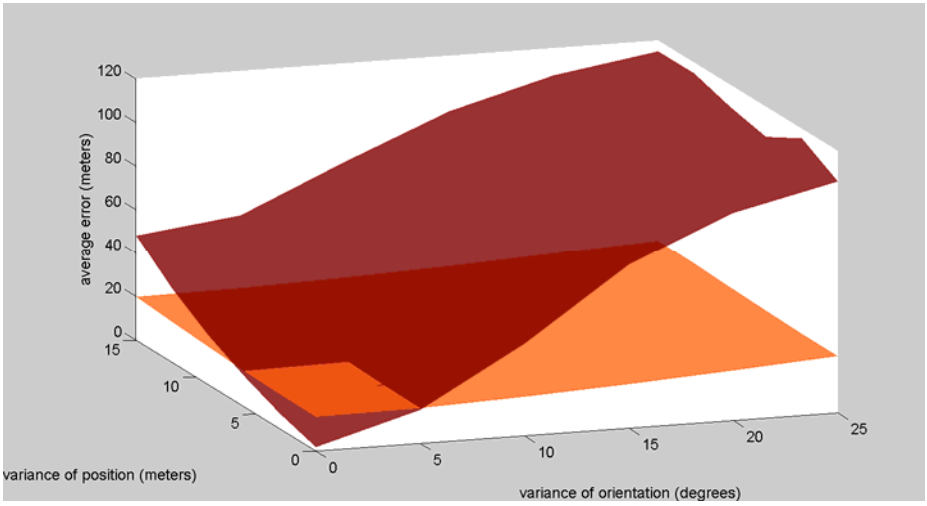

Figure 59: Comparison of the two MANs using average error

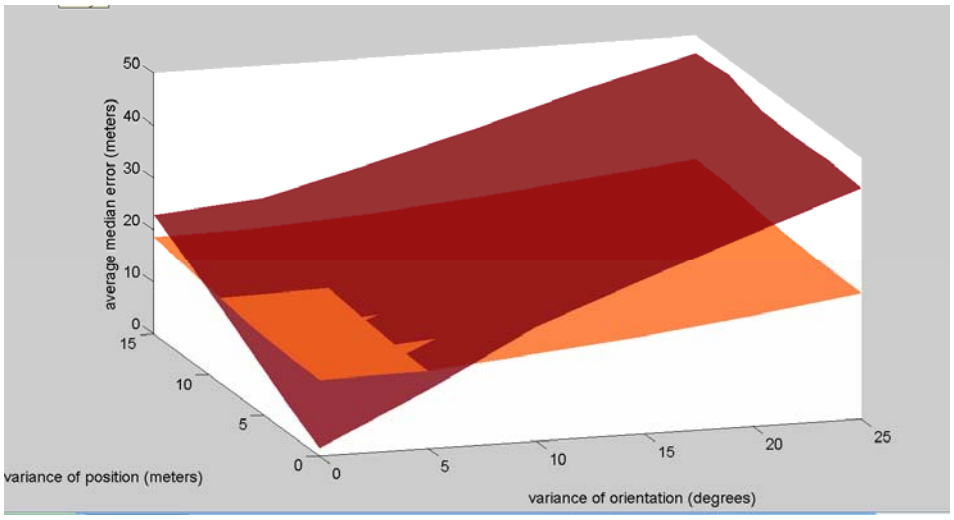

Figure 60: Comparison of the two MANs using median error

These figures show that even though the eight two-channel MAN has one fourth of the sensors, it outperforms the two 32-channel array network in most of the simulated cases using either method. Taking the median errors instead of the average errors improves the performance of the two 32-channel arrays slightly, as it gains two cases using this method.

Adding a variance in the position and orientation of the arrays creates a larger area where the estimate for the sound source is made. Figure 61 shows this intuitive result, as the spread of the points increases when the variance in orientation increases in the two 32-channel MAN:

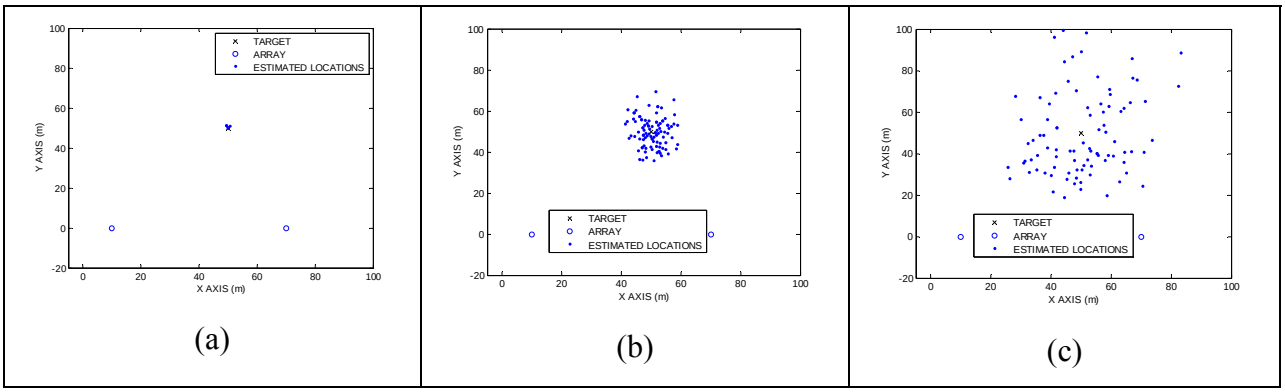

Figure 61: Plots showing estimation locations of the 2 32-channel MAN, with σ_0= (a) 0º, (b) 5º and (c) 15º

However, when the same point with the same variances are used on the eight two-channel MAN, the area starts out greater then the other MAN, and has slower increases as shown in Figure 62:

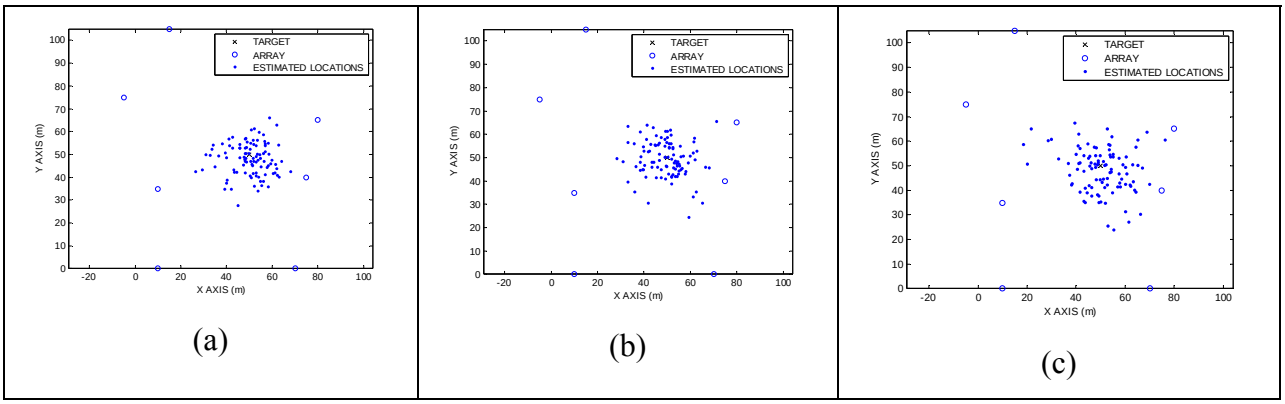

Figure 62: Plots showing estimation locations of the 8 2-channel MAN, with σ_0= (a) 0º, (b) 5º and (c) 15º

More insight is gained into these errors by looking at the individual points on the grid. Figure 63 shows the errors on the grid with no variances:

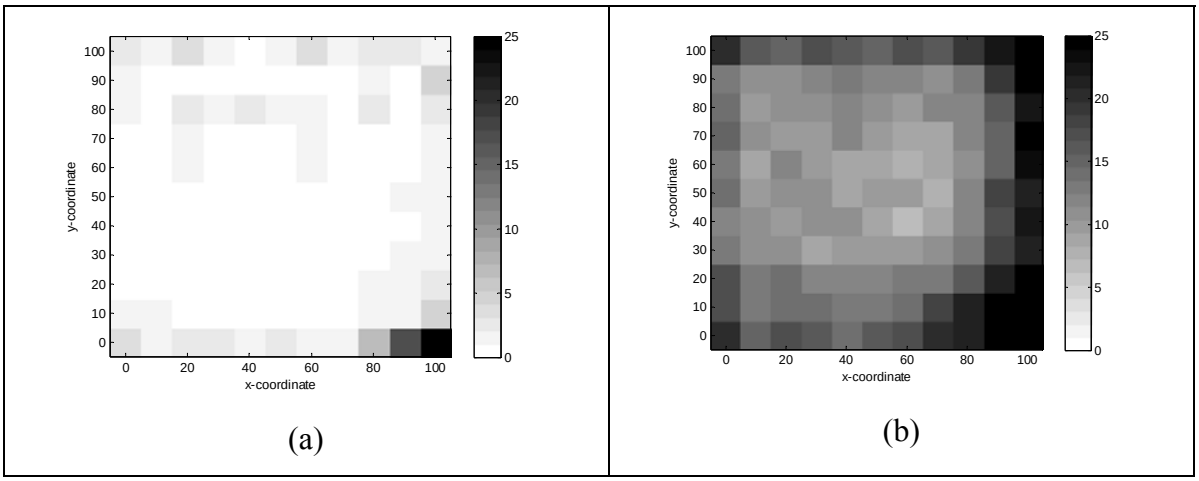

Figure 63: Plots of median errors in meters of the 2 32-channel arrays (a) and 8 2-channel arrays (b) with no introduced variances

This figure shows the superiority of the two 32-channel arrays when the position and orientation have no variance. The plot in the eight two-channel array network also shows the effects of front-back ambiguity outside the core region where the errors are shown to be significantly higher. When variance is introduced into the position and orientation of the arrays the errors increase in either MAN, especially in the two 32-channel MAN. Figure 64 shows the effects of moderate uncertainty parameters of the arrays:

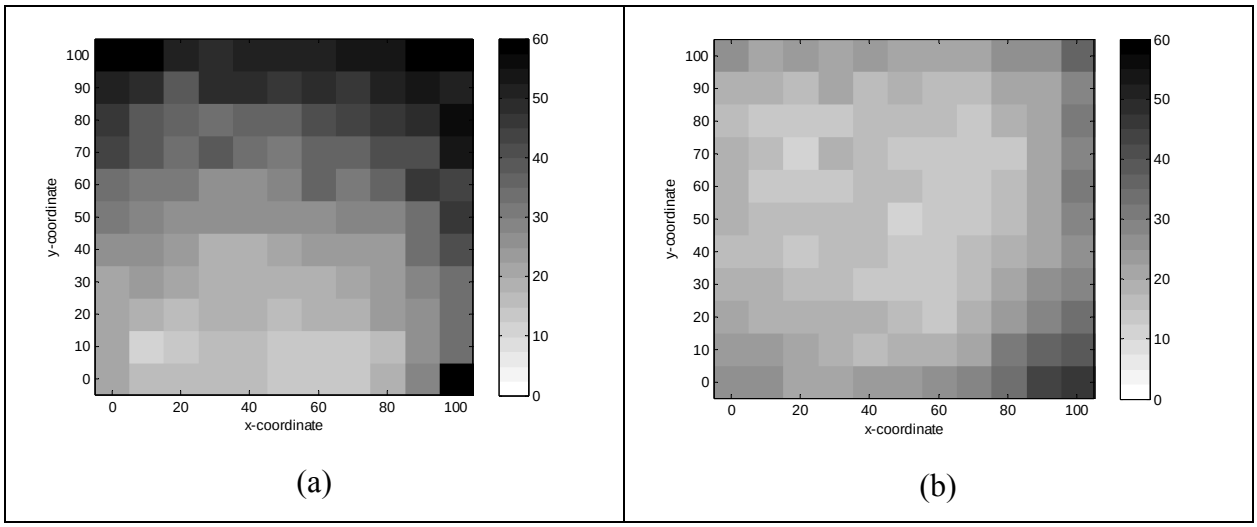

Figure 64: Plots of median errors in meters of the 2 32-channel arrays (a) and 8 2-channel arrays (b) with σ_x and σ_y equaling 9m, and σ_0 equaling 15º

When extreme variances are applied to the arrays, the eight two-channel MAN becomes a much better choice, as shown in Figure 65:

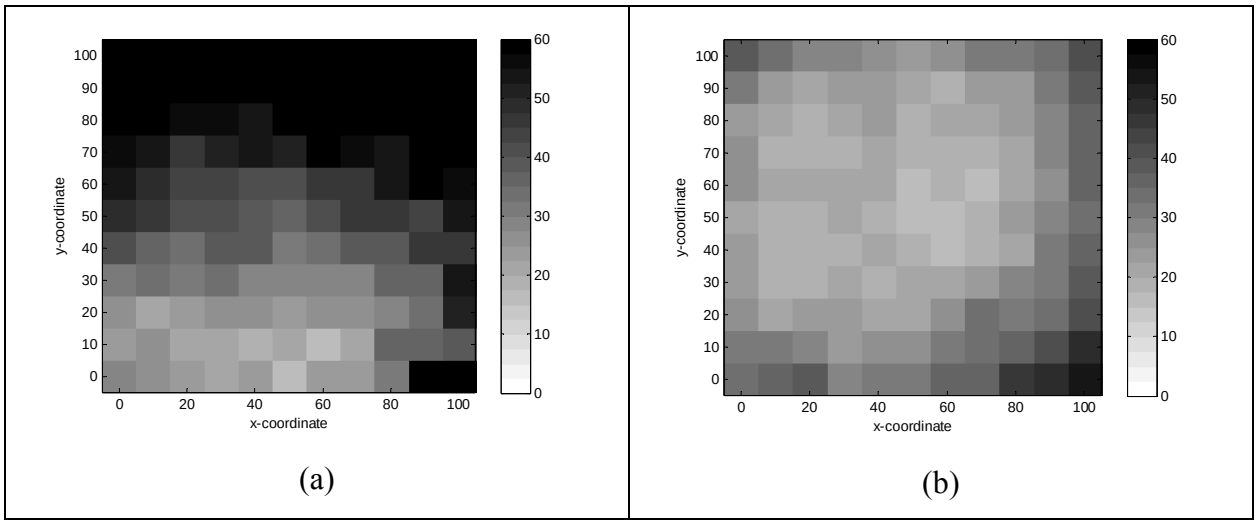

Figure 65: Plots of median errors in meters of the 2 32-channel arrays (a) and 8 2-channel arrays (b) with σ_x and σ_y equaling 15m, and σ_0 equaling 25º

As shown in these figures, the two 32-channel MAN's error increases as the sound sources increases its distance to the arrays. Because of this effect, a better gauge of the performance of the MAN might be normalizing the errors. Figure 66 shows the effects of normalizing this error by the average assumed distance to an array in each MAN:

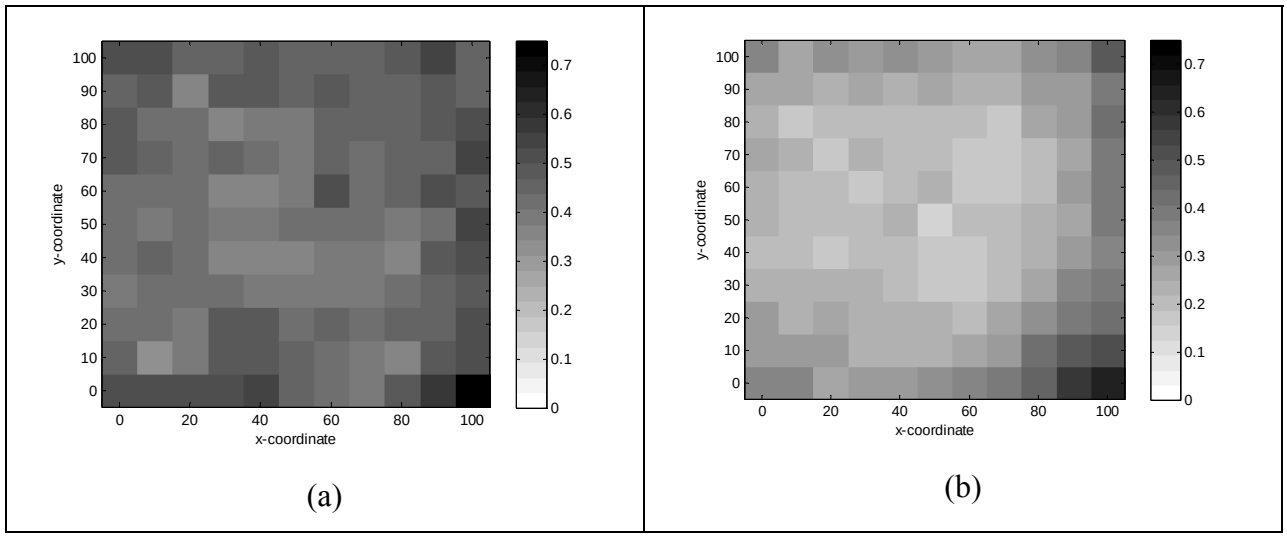

Figure 66: Normalized median errors of the 2 32-channel arrays (a) and 8 2-channel arrays (b) with σ_x and σ_y equaling 9m, and σ_0 equaling 15°

This normalization eliminates the high errors that occur because the sound source is at a farther distance from the arrays used in the two 32-channel MAN, which would give the eight two-channel MAN an advantage since it has arrays closer to the sound source. Instead, the two 32-channel MAN shows almost no bias in the normalized error throughout the grid, while the eight two-channel MAN still maintains lower errors in the core region of the grid. Figure 67 shows the results when the data in Figure 53 and Figure 58 are normalized:

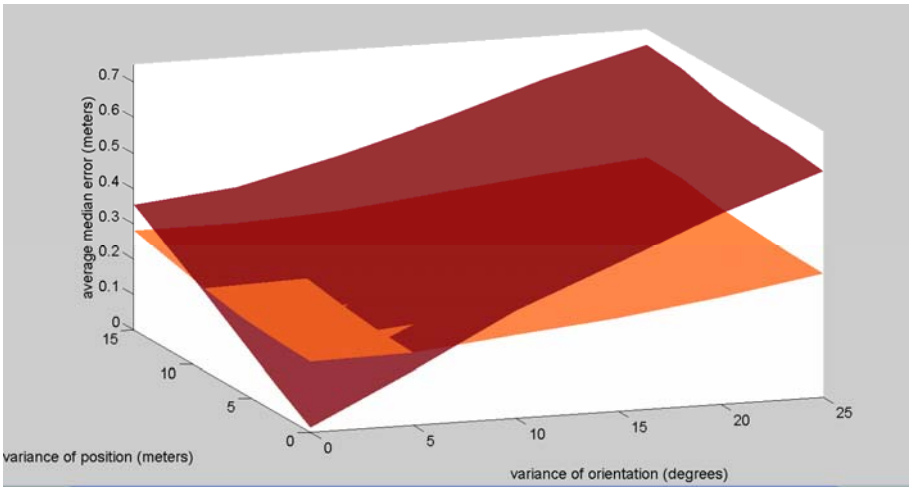

Figure 67: Average normalized median errors of the MANs in core region

Since the average distance to an array was 60.36m for the two 32-channel MAN across the grid, the values in Figure 53 are divided by that distance to yield the values found in Figure 67. Figure 58 is divided by 55.76m, the average distance to an array in the eight two-channel MAN across the grid. The comparison of the median errors remains the almost the same, as the two 32-channel MAN gained one more case when the errors were normalized.

3.3 Conclusions

This chapter investigated the effects of adding variances to the position and orientation of array within a MAN. The results of these simulations show that the more variance is added to arrays within a MAN, the error of the location estimation increases in any MAN, as expected. However, in cases with moderate variance in the array's position and orientation, this chapter showed that having more arrays in the MAN helped limit the amount of error encountered in its location estimates, even when the arrays have very few sensors in them. Also shown was the vulnerability of a MAN with few arrays, as they have only a few (or one) intersection points to base location estimations. These estimates are therefore subject to rapidly increasing errors as the variance increases, as seen in the tables in this chapter.

This chapter shows how errors increase when the source is further away from the arrays of a MAN, even when the SNR of the signal was held constant at all points in the grid. In practice, the signal strength will decay as seen in Chapter 2, creating even higher errors in those locations. Having arrays spread out over a wide area would diminish this effect, as some of the closer arrays would have higher SNRs, which lead to better estimates of the source's parameters. However, just as outliers have little impact on the overall estimate, so too would these better estimates only slightly impact the estimate.

In practice, there will almost always be some uncertainty in position and orientation of an underwater array due to ocean currents, GPS drift, or some other way error is introduced into the estimate of an array's location or orientation. As this chapter has shown, the location estimate of sound sources can become inaccurate with a small amount of variance in the position and orientation of the arrays, especially if the MAN uses only a few arrays, therefore the relatively small errors that may occur because of these factors may cause large errors in location estimation. However, adding more arrays to the MAN is a possible solution to this problem,

even if the arrays themselves have a lower number of sensors on them, and therefore less robust AOA estimates for each individual array. This is especially true if the MAN is placed in an area known to cause high uncertainties in the position or orientation of the arrays, such as an ocean surface or an area with strong currents.

Getting all eight arrays to work properly might also present a challenge. As is the case with any experiment, the more items involved the less probable all items will properly work. Having fewer arrays would limit this problem, and would also take less set-up time and preliminary testing.

4 Future work and final conclusions

This chapter will reiterate the goals stated at the beginning of this work, as well as review the conclusions of the work performed in this research. Future work that would aid in continuing the research discussed in this work will be discussed as well.

4.1 Final conclusions

The first part of this work investigated the performance of different MANs and different fusion methods using active sonar. The thrust of the research was to determine the ideal number of arrays with the same number of total sensors, using data simulated in MATLAB. The algorithm used to make parameter estimates was outlined in the second chapter. The second chapter also tested three different fusion methods to make universal parameter estimations based on the parameter estimations of the individual arrays. MATLAB ran the algorithm and the three fusion methods: transformation of the range-Doppler map, Gaussian curve fitting and a weighted average, and found the average errors on the MANs using the same number of total sensors.

The second part of this work investigated the performance of two MANs with uncertain array location and orientation using passive sonar. The goal of this research was to investigate the performance of a MAN with fewer, more accurate arrays compared with a MAN with more, less accurate arrays with different amounts of variance introduced to the position and orientation of the arrays. Using MATLAB simulated data, the average and median errors were found throughout a grid of points where the sound source was placed.

4.1.1 Active sonar conclusions

The goals investigated using active sonar were to determine the ideal number of arrays and an ideal fusion method that improves the performance of the parameter estimation when all arrays are used instead of one array individually.

The first step in determining the ideal number of arrays was to test the performance of each array individually. This served as a benchmark to determine the robustness of the estimates that were fused together. As seen in the results of the second chapter, the arrays performed much better when they had more sensors; the more sensors in the array the lower the average error became for the parameter estimations. However, when the MAN uses multiple arrays, the data fusion enhances the performance of the parameter estimates.

When the data is fused, it was shown that using two arrays gave the most consistent estimations in parameter estimation. Using more arrays while maintaining the same number of total sensors limited the accuracy of the individual arrays, and left the arrays more vulnerable to making incorrect detection decisions. Using one array gave incomplete velocity measurements, and because the estimates could not be fused with any other array's estimates, left it vulnerable to outlying estimates as well as errors due to discretization.

When comparing fusion methods, it was determined that the transformation method did not enhance parameter estimation. This is due to the estimates of each array not multiplying constructively, which causes ambiguities in the estimation of the target's parameters, as shown in Figure 30. It did, however, offer superior detection decision statistics. However, using the ROC curves seen in the second chapter, it is safe to assume that at sufficient signal strength, the risk of an incorrect detection decision being made is negligible.

Using the 2-D Gaussian curve method offered improvement over the estimates of the individual arrays when one target was introduced into the simulations. However, when more then one target was introduced into the simulations, this method showed a bias toward closer targets because of the higher peaks created by the Gaussian curves with lower variances. When the curves produced by each array are multiplied, these biases still exist, causing biased parameter estimations whenever two or more targets are in a sensor field. Worse, if the curves are wide enough, this method will miss targets that exist in the sonar's range.

The weighted average method proved to be the most reliable method of the three fusion methods investigated in this work. Using two 16-channel arrays, the MAN detected all three targets in the sonar's range without any false or skipped detection. Using this method also decreased the errors in parameter estimation using only one array on all targets.

4.2.2 Passive Sonar conclusions

The second part of this work investigated the effects of uncertainty in position and orientation of the arrays in two different MANs, one using two hi-accuracy arrays, with the other using eight low-accuracy arrays. MATLAB simulated the data at each array, and using the beamformer used in the previous chapter, the MAN made location estimations on a sound source that was placed at one of 121 points within a grid of points.

Using 36 total variations of uncertainty, the results showed that at low uncertainties, the average and median errors were lower using fewer, more accurate arrays. However, as the variance of the array's parameters increased, the MAN using more, less accurate arrays had lower errors, even when the sound source was subjected to front-back ambiguity at some points in the grid. This should be a consideration when selecting the type of MAN being used when detecting targets, as this work showed the MAN with more arrays outperformed the MAN with more accurate arrays, even though the MAN with more arrays had one-fourth the total number of sensors, even when the results were normalized to include the effects of distance on the results.

4.2 Future work

Since this work was theoretical in nature, experimentations should be done on the methods suggested here before accepting it as a proven method for object detection and parameter estimation. This section will make suggestions for future work to be performed on this project, such as experimentation and improvements to the methods described in this work.

4.2.1 Future work using active sonar and data fusion with MANs

The most obvious area future work could be performed on the methods discussed would be to perform experiments on this method using actual gathered data instead of MATLAB simulated data. While MATLAB could still perform the algorithm on this data, these simulations did not include various effects seen underwater, such as thermoclines and sea floors that could potentially create major problems using sonar. Other factors, such as sea noise and the absorption and diffraction of the emitted sound by the target could create errors in the detection and parameter estimation of targets.

Many different methods of beamforming exist to determine the bearing angle of a target. The algorithm presented here uses a conventional beamformer, which is able to determine the bearing angle of a target, but can easily be replaced by a different type of beamformer, such as MVDR or null-steering beamforming. These methods could offer better bearing angle estimates, especially when multiple targets are considered.

The fusion methods discussed here could also be improved. The weighted average function, while offering many positive attributes, requires that all arrays in the MAN detect the target and the estimates be close enough to pass the proximity test. Any number of fusion

methods could be employed to successfully fuse the data from the arrays, including method described in the introduction.

As stated in the second chapter, the targets were placed in front of the arrays to avoid problems with front-back ambiguity in the fusion process. One way to eliminate those problems would be to use a circular array instead of a linear array. The circular array would create beams at all 360 degrees, instead of the 180 degrees as done here, thereby eliminating any front-back ambiguities form the bearing estimates.

Another item worth exploring is the third dimension in these simulations. These simulations acted on a 2-D plane in a constant medium. If the third dimension is to be simulated, some effects of the thermoclines should be included to account for the actual temperature gradient seen underwater.

4.2.2 Future work using passive sonar with uncertainties

Again, the method described in the third chapter needs to be verified with experimentation before its results are accepted. There may be several challenges to this, such as making sure the power of the noise equals the power of the signal across all sensors in use. Uncertainties in position and orientation could be modeled by simply moving the arrays in a random fashion, and then using the assumed array parameters in the location estimation. These tests would have to be done in an anechoic environment to eliminate as much of the interfering noise as possible.

In these simulations, a constant noise of 0 dB was used across all sensors in either MAN. However, since actual noise levels decrease logarithmically as it travels from a sound source, a more realistic version of the noise level with this effect included, similar to what was done in the second chapter, could be performed along with differing noise levels if the noise level is to remain constant.

Another suggestion for this part of this research would be to add more sound sources with different frequencies and have the MANs track both targets. This could create problems with the conventional beamformer, especially if the frequencies do not differ by more then 10%. This would also require the beamformer to operate twice, extending the computing time required to estimate the location of both objects.

Bibliography

[1] Deborah Estrin, Ramesh Govindan, John Heidemann and Staish Kumar, "Next Century Challenges: Scalable Coordination in Sensor Networks," proceedings from MOBICOM, (Seattle), pp. 263-270, (1999).

[2] Philip Leivs, David Gay and David Culler, "Active Sensor Networks," Fuzzy Information Processing Society, 1997. NAFIPS '97, (1997).

[3] Varshney, Lav, "Review of the IEEE Upstate New York Workshop on Sensor Networks," Syracuse Research Corporation, (Syracuse, NY), (29 October 2003).

[4] Josh Erling, Michael J. Roan, and Mark R. Gramann, "Performance Bounds for Multisource Parameter Estimation using a Multiarray Network," to be published.

[5] R. Kozick and B. Sadler, "Source localization with distributed sensor arrays and partial spatial coherence," IEEE Transactions on Signal Processing, vol. 52-3, pp. 601- 616, (Mar. 2004).

[6] H. Cramér, *Mathematical Methods of Statistics*, Princeton University Press, Princeton, 1946, ch. 21.

[7] M. Abramowitz and I. Stegun, *Handbook of Mathematical Functions*, Dover Publications, New York, 1970, ch. 26.

[8] Aleksandar Dogandžić, "Cramér–Rao Bounds for Estimating Range, Velocity, and Direction with an Active Array," IEEE Transactions on Signal Processing, Vol. 49, No. 6, pp. 1122-1137, (June 2001).

[9] John A. Fawcett and Brian H. Maranda, "Cramer-Rao Lower Bounds for Multi-sensor Localization," Proceedings, OCEANS '93: 'Engineering in Harmony with Ocean,' Vol. 2, pp. II/247-II/252, (Victoria, B.C.), (1993).

[10] V.N.S. Samarasooriya and P.K. Varshney, "Decentralized Parameter Estimation with Fuzzy Information,"

[11] Yaron Rachlin, Rohit Negi, and Pradepp Khosla, "Sensing Capacity for Discrete Sensor Network Applications," Proceedings from the Fourth International Symposium on Information Processing in Sensor Networks, pp. 126-132 (2005).

[12] Vijay Varadarajan and Jeffrey Krolik, "Model-based Space-time Adaptive Processing for Active Sonar," Proceedings of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers, (2004).

[13] Dan Li, Kerry D. Wong, Yu H. Hu and Akbar M. Sayeed, "Detection, Classification and Tracking of Targets in Distributed Sensor Networks," IEEE Signal Processing Magazine, Vol. 19, Issue 2, pp. 17-29, (March 2002).

[14] Donal McErlean and Shrikanth Narayanan, "Distributed Detection and Tracking in Sensor Networks," Proceedings, 36th Asilomar Conf. Signals, Systems & Computers, (Pacific Grove, CA), (2002).

[15] Kristen L. Bell, "MAP-PF Tracking with a Network of Sensor Arrays," Proceedings of the Acoustic, Speech and Signal Processing, Vol. 4, pp. iv/849-iv/852, (2005).

[16] Leon H. Sibul, Michael J. Roan, Stuart Schwartz and Christian Coviello, "Lossless Information Fusion for Active Ranging and Detection Systems," IEEE Transactions on Signal Processing, Vol. 54, No. 10, pp. 3980-3990, (Oct. 2006).

[17] Harry L. Van Trees, *Detection, Estimation, and Modulation Theory Part I,* John Wiley & Sons, Inc., New York, (1968).

Vita

Brent Gold was born August 26, 1981 in Lewisburg, PA to the parents of Bill and Chris Gold. He spent his early childhood there before moving to York, PA where he graduated from York Suburban High School in 2000. That fall, he started attending Bucknell University where he graduated with a Bachelor of Science in mechanical engineering in May 2004, having made Dean's list three times. During his time at Bucknell, he served as an intern at United Defense in York, PA in the summer of 2003. Upon completion of his degree, he worked for two companies in the York area before entering Virginia Tech's graduate school and working for Dr. Michael Roan in the Vibration and Acoustics Lab in January 2006. After working for Dr. Roan for two years, Brent defended his thesis in December 2007.