REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* 31-12-2015	**2. REPORT TYPE** Final	**3. DATES COVERED** *(From - To)* 28 May 2013 – 31 Dec 2015

4. TITLE AND SUBTITLE Continuous Active Sonar for Undersea Vehicles Final Report: Input of Factor Graphs into the Detection, Classification, and Localization Chain and Continuous Active SONAR in Undersea Vehicles	**5a. CONTRACT NUMBER** **5b. GRANT NUMBER** N000141310642 **5c. PROGRAM ELEMENT NUMBER**
6. AUTHOR(S) Brandi N. Gross John R. Sacha (PI)	**5d. PROJECT NUMBER** 20759 **5e. TASK NUMBER** 01 **5f. WORK UNIT NUMBER** 09300 - Guidance & Control Technologies
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Applied Research Laboratory The Pennsylvania State University P. O. Box 30 State College, PA 16804-0030	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of Naval Research 875 North Randolph Street Arlington, VA 22203-1995	**10. SPONSOR/MONITOR'S ACRONYM(S)** ONR **11. SPONSOR/MONITOR'S REPORT NUMBER(S)**

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A. Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report examines the application of factor graphs (graph-based architectures utilizing message passing) to the areas of continuous active sonar (CAS) and unmanned underwater vehicle signal processing. Closed-form factor graph formulations for detection, classification, and tracking are developed, and the effects on bistatic sonar performance of various categories of inter-vehicle messages are quantified.

[This report is excerpted from the thesis submitted by Brandi N. Gross in partial fulfillment of the requirements for the degree of Master of Science in Mechanical Engineering at Virginia Polytechnic Institute and State University. This research was conducted under ONR's Naval Undersea Research Program (formerly the University Laboratory Initiative) in 2014-2015 at Virginia Tech (Prof. M. Roan, faculty advisor) and the Applied Research Laboratory of The Pennsylvania State University (Dr. J. Sacha, summer internship supervisor).]

15. SUBJECT TERMS

Factor graph, continuous active SONAR, unmanned underwater vehicle, Kalman filter, particle filter

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** John R. Sacha
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U		63	**19b. TELEPHONE NUMBER** *(include area code)* 814-863-4162

FINAL REPORT

CONTINUOUS ACTIVE SONAR FOR UNDERSEA VEHICLES:

INPUT OF FACTOR GRAPHS INTO THE DETECTION, CLASSIFICATION, AND LOCALIZATION CHAIN AND CONTINUOUS ACTIVE SONAR IN UNDERSEA VEHICLES

Award Number	N000141310642
Title of Project	University Laboratory Initiative: Continuous Active Sonar for Undersea Vehicles
Principal Investigator	Dr. John R. Sacha
Organization	Applied Research Laboratory The Pennsylvania State University

Abstract: This report examines the application of factor graphs (graph-based architectures utilizing message passing) to the areas of continuous active sonar (CAS) and unmanned underwater vehicle signal processing. Closed-form factor graph formulations for detection, classification, and tracking are developed, and the effects on bistatic sonar performance of various categories of inter-vehicle messages are quantified.

This report is excerpted from the thesis submitted by Brandi N. Gross in partial fulfillment of the requirements for the degree of Master of Science in Mechanical Engineering at Virginia Polytechnic Institute and State University, titled "Input of Factor Graphs into the Detection, Classification, and Localization Chain and Continuous Active SONAR in Undersea Vehicles." This research was conducted under ONR's Naval Undersea Research Program (formerly the University Laboratory Initiative) in 2014-2015 at Virginia Tech (Prof. M. Roan, faculty advisor) and the Applied Research Laboratory of The Pennsylvania State University (Dr. J. Sacha, summer internship supervisor).

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Office of Naval Research.

THESIS ABSTRACT

The focus of this thesis is to implement factor graphs into the problem of detection, classification, and localization (DCL) of underwater objects using active SOund Navigation And Ranging (SONAR). A factor graph is a bipartite graphical representation of the decomposition of a particular function. Messages are passed along the edges connecting factor and variable nodes, on which, a message passing algorithm is applied to compute the *posterior* probabilities at a particular node.

This thesis addresses two issues. In the first section, the formulation of factor graphs for each section of the DCL chain required followed by their closed-form solutions. For the detector, the factor graph determines if the signal is a detection or simply noise. In the classifier, it outputs the probability for the elements in the class. Last, when using a factor graph for the tracker, it gives the estimated state of the object being tracked.

The second part concentrates on the application to Continuous Active SONAR (CAS). When using CAS, a bistatic configuration is used allowing for a more rapid update rate where two unmanned underwater vehicles (UUVs) are used as the receiver and transmitter. The goal is to evaluate CAS's effectiveness to determine if the tracking accuracy improves as the transmit interval decreases. If CAS proves to be more efficient in target tracking, the next objective is to determine which messages sent between the two UUVs are most beneficial. To test this, a particle filter simulation is used.

Chapter 1: INTRODUCTION

1.1 Motivation

SOund Navigation And Ranging (SONAR) is a technology used in naval applications to detect undersea objects. A signal is transmitted from a source and is then reflected from different undersea entities. This includes rocks, sea creatures, the ocean floor, the ocean surface, and targets of interest. The received echoes are analyzed to determine the identity of, and track the objects that created them. It is important to extract as much information as possible from the reflection of the signal while using minimal resources. This information is vital for knowing whether to stop or continue tracking an entity, or determine if other action is required to be taken. For this process to be optimal, the detector, classifier, and tracker need to utilize models and parameters that accurately match the environment in order to minimize false or missed detections.

In pulsed active SONAR (PAS), a source transmits a short pulse of sound, or ping, and then listens for the echo from the signal. A configuration in which the transmitter and the receiver are co-located is known as a monostatic situation (Yakubovskiy, 2009). If an echo arrives at the same time a transmission is taking place, the data from the reflection is not received, leading to a loss of data. To avoid this loss, long listen intervals between transmissions are used, leading to lower update rates, which can cause difficulties for the tracker (Grimmett & Wakayama, 2013). As opposed to PAS, continuous active SONAR (CAS) transmits a continuous signal. This requires a bistatic configuration, i.e., the use of a separate, non-co-located, transmitter and receiver (Yakubovskiy, 2009). This configuration permits a higher update rate, and in practice will allow the use of a lower transmit source level to achieve the same total echo energy.

1.2 Problem Statement

1.2.1 Detection Classification and Localization

The Detection, Classification, and Localization (DCL) process begins by an unmanned underwater vehicle (UUV) receiving the reflection of the signal previously transmitted. If the received signal level is above a set threshold, set by the detector, the signal is deemed as a possible object to identify and possibly track. The detector sends the data to the classifier to be analyzed by identifying specific features characterized within the signal. Once the echo has been identified as an object of interest, the data is sent to the tracker. The tracker attempts to determine if the data is from an object that is currently being tracked, in which case the data is used to update the track, or if it is from a new object and a new track needs to be created. The tracker can perform further classification, based on the object's movement.

The objective of this thesis is to develop graph-based algorithms for improved detection, classification, localization, and tracking of underwater objects using continuous active sources. Next, is to apply these newly developed factor graph approaches to the detection, classification, and localization chain and if possible, use the output information to feed back into the process for more accurate data.

1.2.2 Continuous Active SONAR

Continuous active SONAR uses a bistatic configuration. In such a configuration, the transmitter and receiver are separate vehicles, allowing the transmitter to send a continuous signal without the threat of information loss. Without this limit of time between signal transmissions, the detector, classifier, and tracker can receive a constant flow of data. A goal of this thesis is to evaluate the performance of CAS to determine if the accuracy of the location determined by the tracker improves as the update rate is increased.

In order to localize the object in relation to the receiver using bistatic SONAR, the relative location of the transmitter to the receiver needs to be known along with the time the signal is transmitted. Position information is easy to obtain if both the transmitter and receiver are immobile, or if Global Positioning System (GPS) is available. Likewise, time synchronization is easy if radio communications are available for use. However, in an unmanned underwater vehicle (UUV) scenario, both the transmitter and receiver are in motion, GPS is generally not available, and there is typically no radio link. However, different pieces of information exist which can be used to help deduce the relative geometry. For example, the direct blast from the transmitter, and the bearing measurements in relation to the receiver. Also, acoustic communications (of low bandwidth) can be used via an acoustic modem, or by embedding messages within the source transmit signal. Messages could also be received by measurements taken from a SONAR sensing mechanism. The goal of the thesis is to determine which messages will be the most beneficial to better localize the object being tracked.

1.3 Original Contributions

Even though the implementation of the detector, classifier, and tracker into factor graphs had been found previously, one of the original contributions of this thesis is the computation of closed form solutions for the factor graph tracker. The closed form solutions were also found to be equivalent to the equations of the standard Kalman filter. MATLAB codes of the two trackers were constructed and ran against each other to track a noisy object, and both were found to identically track the object. A key difference found is the factor graph tracker has the capability to track objects with biased measurements.

Another contribution was proving the effectiveness of CAS versus PAS since it was proven that the smaller the transmit interval, the lower the tracking error. Because of this result, further research was conducted to determine the value of different messages that could be known when tracking an object. The utility of each message combination given different scenarios were found and are valuable contributions to target tracking.

Chapter 2: BACKGROUND

2.1 Factor Graph

A factor graph is the graphical representation of the decomposition of a particular function. It consists of two types of nodes, variable and factor nodes, connected to each other through edges, classifying a factor graph as a bipartite graph. For every variable x_i, there is a variable node, and there is a factor node for every function f_j. *N*odes, x_i and f_j, are connected via an edge, if and only if x_i is an argument of f_j. A function $g(x_1,\dots,x_n)$ that is equal to the product of multiple "local functions", all having arguments from a subset of $\{x_1,\dots,x_n\}$ is shown below in Equation 1.

$$g(x_1, \dots, x_n) = \prod_{j \in J} f_j(X_j) \tag{1}$$

where J is a discrete index set. X_j is a subset of $\{x_1,\dots,x_n\}$, whose elements are arguments of the sub-function $f_j(X_j)$. Equation 1 can easily be represented as a factor graph relating the arguments and their local functions.

As an example, presented in several sources, suppose a function, g, consisting of five variables, x_1, ..., x_5, is represented as the product of five functions, f_A, f_B, f_C, f_D, and f_E, such that $X_A = \{x_1\}$, $X_B = \{x_2\}$, $X_C = \{x_1,x_2,x_3\}$, $X_D = \{x_3,x_4\}$, and $X_E = \{x_3,x_5\}$, where $\{A, B, C, D, E\}$ are elements of the discrete set J. The factorization of the function g, shown in Equation 2,

$$g(x_1, x_2, x_3, x_4, x_5) \;=\; f_A(x_1) f_B(x_2) f_C(x_1, x_2, x_3) f_D(x_3, x_4) f_E(x_3, x_5) \tag{2}$$

can be expressed via the factor graph in Figure 1. (Frank R. Kschischang, 2001).

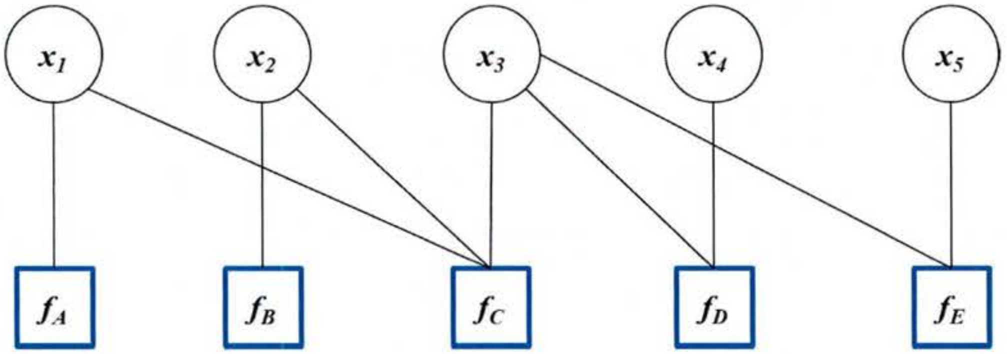


Figure 1: The factorization of $g(x_1,x_2,x_3,x_4,x_5)$ expressed as a factor graph. The squares represent the factor nodes, f_j, and the circles are the variable nodes, x_n. Image based on (Frank R. Kschischang, 2001). This image has been used in several papers as a general representation of a factor graph.

Another form of a factor graph is described by Forney in (Forney, 2005) in which Equation *2* would be represented by Figure 2.

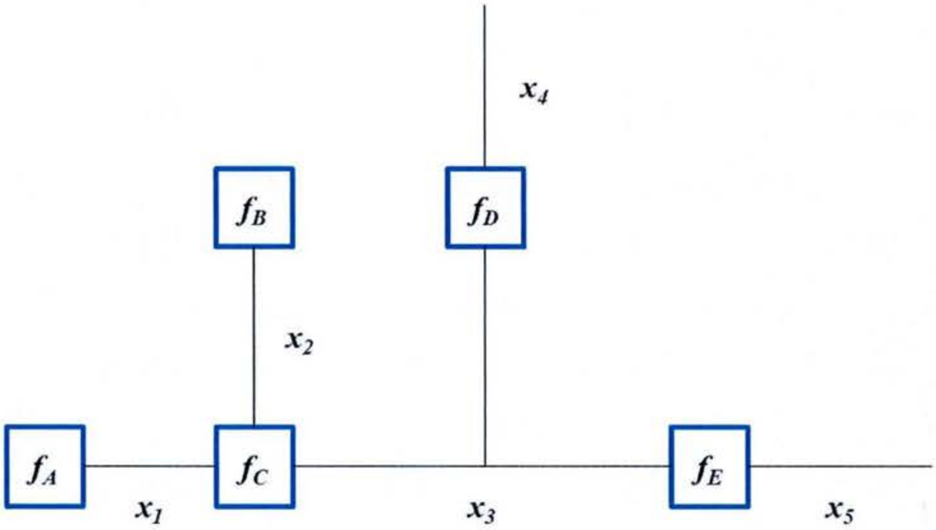


Figure 2: Forney's factor graph of Equation 2.

Forney's version of a factor graph will be used later in the paper, but for further description of how a factor graph may be marginalized; the first factor graph will be used as reference.

Marginal computations at $g_i(x_i)$ may be desired when using factor graphs (i.e. if $g(x_1,x_2,x_3,x_4,x_5)$ is a joint probability mass function) and can be done by using the distributive property. This will be demonstrated by solving for the marginalization of $g_3(x_3)$. To get a better visual of how this property is used, the factor graph in Figure 1 can be rearranged to more clearly identify the edges connecting to $g_3(x_3)$ and is shown in Figure 3. This is known as a rooted tree with x_3 as the root. However, note that only the visual display is changing, while the connections will remain the same as in Figure 1.

The marginalization of $g_3(x_3)$ can be written as

$$g_3(x_3) = \left(\sum_{\sim\{x_3\}} f_A(x_1) f_B(x_2) f_C(x_1, x_2, x_3)\right) \times \left(\sum_{\sim\{x_3\}} f_D(x_3, x_4)\right) \times \left(\sum_{\sim\{x_3\}} f_E(x_3, x_5)\right).$$

Since typically, more than one marginalization will need to be computed, it is more efficient to use the sum-product algorithm, which is explained in section 2.1.1.

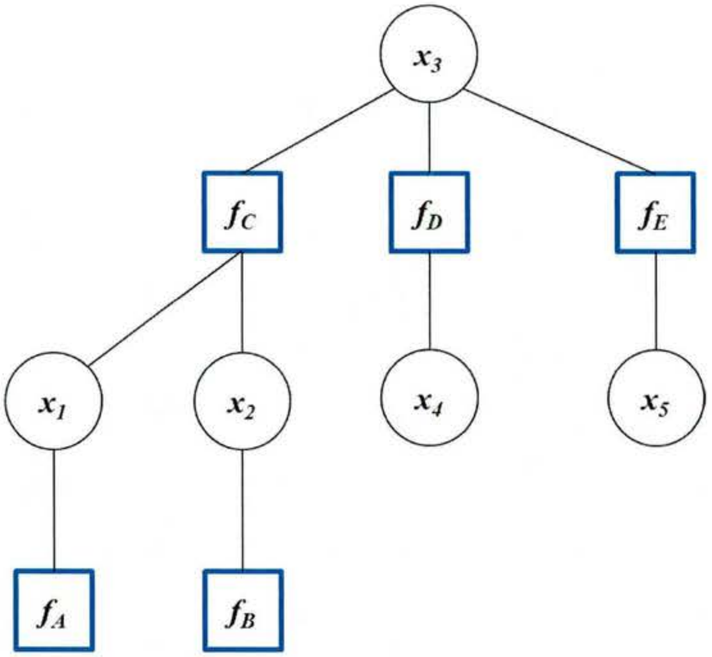


Figure 3: The function $g(x_1,x_2,x_3,x_4,x_5)$ rearranged to focus on the marginalization at $g_3(x_3)$. Image based on (Frank R. Kschischang, 2001). This image has been used in several papers as a general representation of a factor graph.

2.1.1 Belief Propagation

Belief Propagation (BP) can be used for problems in artificial intelligence, digital communications, statistical physics, and error-correcting coding theory, among other areas (Frank R. Kschischang, 2001), (Jonathan S. Yedidia, 2001). Belief Propagation, also known as the sum-product algorithm, is used when trying to calculate the marginals for an inference problem that is using message passing. If a graph doesn't contain any loops, then the BP converges to an exact solution, otherwise, it provides an estimation of the solution (Coughlan, 2009).

Messages are passed and updated between the nodes and once the messages converge, the *posterior* probabilities, also called marginal probabilities or beliefs, are calculated (Coughlan, 2009). Given a variable node x_i and a factor node f_j, the message passed from f_j to x_i can be denoted $\mu_{f_j \to x_i}(x_i)$. The message determines the appropriate state that x_i should be in (Jonathan S. Yedidia, 2001). Messages are passed along the edges between nodes and can travel in both directions. Messages $\mu_{f_j \to x_i}(x_i)$ are vectors with the same dimensions as x_i (Jonathan S. Yedidia, 2001). Nodes connected to a node x_i, through an edge are called neighbors and can be denoted as $n(x_i)$. A section of a factor graph, along with its messages, is illustrated in Figure 4. The messages between nodes are calculated using the sum-product update rule. This rule states that the message sent from a node v on an edge e to another node q, is the product of the local function at v and all messages received at v on different edges than e. The product is then summarized for the variable associated with e. If v is a variable node, then the unit function is multiplied (Frank R. Kschischang, 2001). Referring to Figure 4 for notation, the message calculated from a variable node to a function node is given by Equation 3.

$$\mu_{x\to f}(x) = \prod_{h\in n(x)\{f\}} \mu_{h\to x}(x) \tag{3}$$

$$\mu_{f\to x}(x) = \left(\sum_{\sim\{x\}} f(X) \prod_{y\in n(f)\backslash\{x\}} \mu_{y\to f}(y) \right) \tag{4}$$

Equation 4 is used if a message is passed from a local function to a variable, where $X = n(f)$ is the vector of neighboring variable nodes to the function *f* and the expression $n(f)\backslash\{x\}$ indicates all neighboring variable nodes to the function node *f* excluding the variable node *x*. The dashed ovals in Figure 4 indicate the neighbors of *x* and *f* excluding *f* and *x*, respectively. If a node is only connected to one other node, it is considered a leaf node. If a leaf node is a variable node, such as y_1 in Figure 4, then the message is computed using Equation 5 and if a leaf node is a factor node, like h_2 in Figure 4, Equation 6 is used.

$$\mu_{x\to f}(x) = 1 \tag{5}$$

$$\mu_{f\to x}(x) = f(x) \tag{6}$$

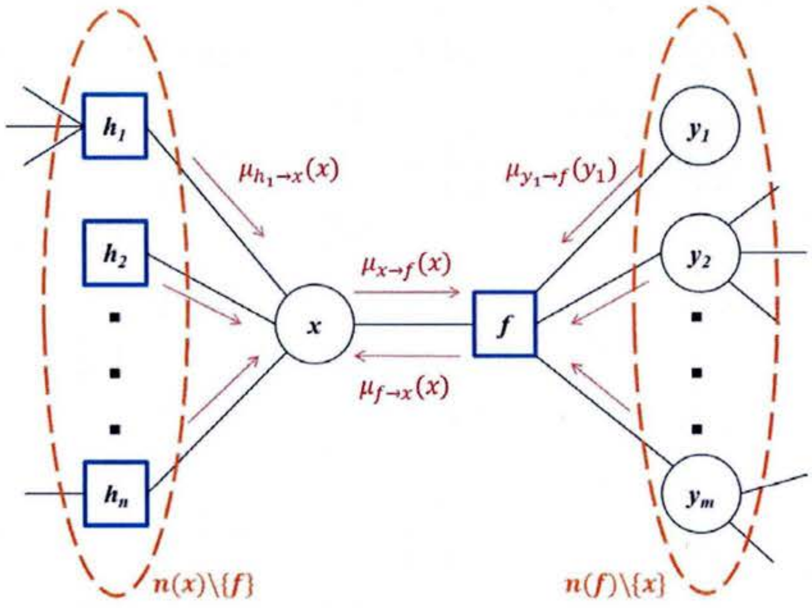


Figure 4: A section of a factor graph illustrating the sum-product update rule. Image based on (Frank R. Kschischang, 2001). This image has been used in several papers as a general representation of a factor graph.

In order to begin using the sum-product algorithm, message passing begins at the leaves of the factor graph. An example from (Frank R. Kschischang, 2001) of the sum-product algorithm will be broken down for Equation 2. The factor graph for Equation 2 is re-arranged in Figure 5 to clearly visualize the messages being passed.

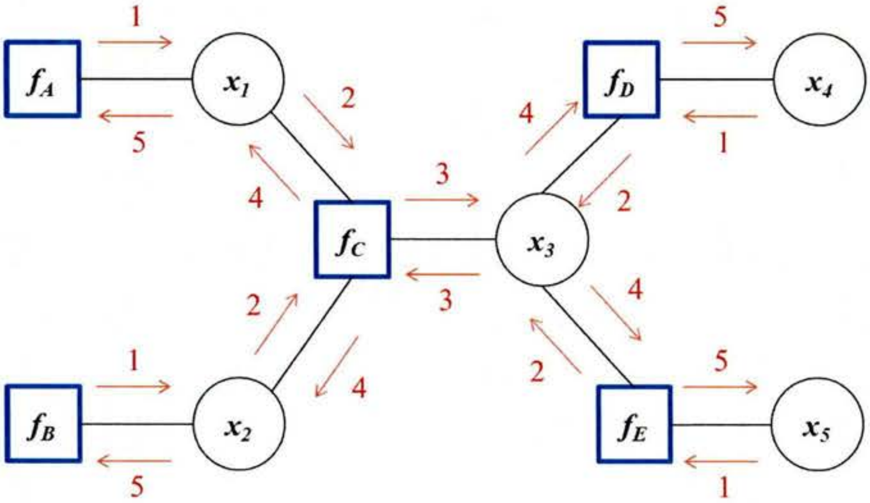


Figure 5: Re-arranged factor graph of Equation 2. Each step of the sum-product algorithm is labeled in red. Image based on (Frank R. Kschischang, 2001). This image has been used in several papers as a general representation of a factor graph.

For this factor graph, only 5 steps are needed to compute the messages, which are labeled in red above in Figure 5.

Step 1:

$$\mu_{f_A \to x_1}(x_1) = \sum_{\sim\{x_1\}} f_A(x_1) = f_A(x_1)$$

$$\mu_{f_B \to x_2}(x_2) = \sum_{\sim\{x_2\}} f_B(x_2) = f_B(x_2)$$

$$\mu_{x_4 \to f_D}(x_4) = 1$$

$$\mu_{x_5 \to f_E}(x_5) = 1$$

Step 2:

$$\mu_{x_1 \to f_C}(x_1) = \mu_{f_A \to x_1}(x_1)$$

$$\mu_{x_2 \to f_C}(x_2) = \mu_{f_B \to x_2}(x_2)$$

$$\mu_{f_D \to x_3}(x_3) = \sum_{\sim\{x_3\}} \mu_{x_4 \to f_D}(x_4) f_D(x_3, x_4)$$

$$\mu_{f_E \to x_3}(x_3) = \sum_{\sim\{x_3\}} \mu_{x_5 \to f_E}(x_5) f_E(x_3, x_5)$$

Step 3:

$$\mu_{f_C \to x_3}(x_3) = \sum_{\sim\{x_3\}} \mu_{x_1 \to f_C}(x_1) \mu_{x_2 \to f_C}(x_2) f_C(x_1, x_2, x_3)$$

$$\mu_{x_3 \to f_C}(x_3) = \mu_{f_D \to x_3}(x_3) \mu_{f_E \to x_3}(x_3)$$

Step 4:

$$\mu_{f_C \to x_1}(x_1) = \sum_{\sim\{x_1\}} \mu_{x_3 \to f_C}(x_3) \mu_{x_2 \to f_C}(x_2) f_C(x_1, x_2, x_3)$$

$$\mu_{f_C \to x_2}(x_2) = \sum_{\sim\{x_2\}} \mu_{x_3 \to f_C}(x_3) \mu_{x_1 \to f_C}(x_1) f_C(x_1, x_2, x_3)$$

$$\mu_{x_3 \to f_D}(x_3) = \mu_{f_C \to x_3}(x_3) \mu_{f_E \to x_3}(x_3)$$

$$\mu_{x_3 \to f_E}(x_3) = \mu_{f_C \to x_3}(x_3) \mu_{f_D \to x_3}(x_3)$$

Step 5:

$$\mu_{x_1 \to f_A}(x_1) = \mu_{f_C \to x_1}(x_1)$$

$$\mu_{x_2 \to f_B}(x_2) = \mu_{f_C \to x_2}(x_2)$$

$$\mu_{f_D \to x_4}(x_4) = \sum_{\sim\{x_4\}} \mu_{x_3 \to f_D}(x_3) f_D(x_3, x_4)$$

$$\mu_{f_E \to x_5}(x_5) = \sum_{\sim\{x_5\}} \mu_{x_3 \to f_E}(x_3) f_E(x_3, x_5)$$

The function can be calculated at any one of the variable nodes by multiplying all messages heading toward the specific variable node. The functions at each variable are written below.

$$g_1(x_1) = \mu_{f_A \to x_1}(x_1)\, \mu_{f_C \to x_1}(x_1)$$

$$g_2(x_2) = \mu_{f_B \to x_2}(x_2)\, \mu_{f_C \to x_2}(x_2)$$

$$g_3(x_3) = \mu_{f_C \to x_3}(x_3)\, \mu_{f_D \to x_3}(x_3) \mu_{f_E \to x_3}(x_3)$$

$$g_4(x_4) = \mu_{f_D \to x_4}(x_4)$$

$$g_5(x_5) = \mu_{f_E \to x_5}(x_5)$$

At this stage, the marginal distribution of each variable node is proportional to the product of all the messages from neighboring nodes directed toward the variable node after being normalized. Equation 7 shows the belief of a variable node before normalization. The belief converging at a factor node is proportional to the product of the function and the messages from the neighboring variable nodes after normalization as shown in Equation 8 (before normalization) (Coughlan, 2009).

$$b_i(x_i) \propto \prod_{f_j \in n(x_i)} \mu_{f_j \to x_i}(x_i)$$

(7)

$$b_{X_j}(X_j) \propto f_j(X_j) \prod_{x_i \in n(f_j)} \mu_{x_i \to f_j}(x_j)$$

(8)

If the factor graph is open and does not contain a closed chain or loop, i.e. a tree, then the exact marginals can be established.

2.1.2 Gaussian Belief Propagation

2.1.2.1 Undirected Graphs

The Gaussian Belief Propagation (GaBP) algorithm is used when the joint probability distributions are Gaussian. When using GaBP, the messages and beliefs are also all Gaussian and the means and inverse covariance matrices can be used for the update equations. Weiss and Freeman (Weiss & Freeman, Oct., 2001) describe the different equations used to determine the beliefs for the GaBP algorithm in an undirected graph, which, when compared to regular belief propagation, are not as difficult.

The information matrix J_{ij} is given by the equation

$$J_{ij} = \begin{bmatrix} a & b \\ b^T & c \end{bmatrix}.$$

(9)

In GaBP, the update equations can be the means and inverse covariance matrices due to the messages between neighboring nodes also being a mean and inverse covariance matrix. The message update equations are given by

$$V_{ij} = c - b(a + V_0)^{-1} b^T$$

(10)

$$m_{ij} = -V_{ij}^{-1} b(a + V_0)^{-1} V_0 m_0$$

(11)

where V_{ij} and m_{ij} is the precision or inverse covariance matrix and mean vector, respectfully, sent from the node x_i to the node x_j. The initials are described in the equations below.

$$V_0 = V_{ii} + \sum_{x_k \in N(x_i) \setminus x_j} V_{ki}$$

(12)

$$m_0 = V_0^{-1}\left(V_{ii}m_{ii} + \sum_{x_k \in N(x_i)} V_{ki}m_{ki} \right) \tag{13}$$

In the above equations, V_{ii} and m_{ii} are the inverse covariance matrix and mean vector, respectively of the function $f_{ii}(x_i, y_i)$. The mean of the *posterior* probability at node x_i is represented by m_i. Likewise, V_i is the inverse covariance matrix of the belief at node x_i. Lastly, the expression $N(x_i)\backslash x_j$ denotes all neighboring nodes of x_i, with the exception of x_j. Finally, the beliefs, or *posterior* probabilities in an undirected graph are declared as Equations 14 and 15.

$$V_i = V_{ii} + \sum_{x_k \in N(x_i)} V_{ki} \tag{14}$$

$$m_i = V_i^{-1}\left(V_{ii}m_{ii} + \sum_{x_k \in N(x_i)} V_{ki}m_{ki} \right). \tag{15}$$

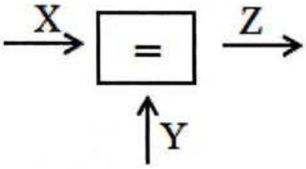
If the information matrix J_{ij} is diagonally dominant, then the Gaussian belief propagation will converge, resulting in precise means (Loeliger, et al., 2007). (Weiss & Freeman, Oct., 2001).

2.1.2.2 Directed Graphs

Unfortunately, a factor graph is a directed graph, which causes the Gaussian messages to become a little more complex to compute. Linear factor graph models contain nodes of multiplication, addition, and equality, resulting in different forks to be formed in the graph. Because of this characteristic, the messages for a directed Gaussian linear factor graph are also Gaussian and remain so throughout computation using the sum-product algorithm. The technique about to be described can be used in many signal processing situations, as referred to by Loeliger. (Loeliger, et al., 2007).

The mean and covariance will still be represented by m and V, however, since this is a directed graph, the notation needs to reflect direction. If x is a variable signified by a directed edge, then the message containing x following the direction of the edge is represented by $\overrightarrow{\mu_x}$, whereas $\overleftarrow{\mu_x}$ is the notation of the message passing in the opposite direction of the edge. Likewise, the mean and covariance of x in the direction of the edge is denoted as $\overrightarrow{m}_x$ and $\overrightarrow{V_x}$, respectively and by $\overleftarrow{m}_x$ and $\overleftarrow{V_x}$ when traveling in the opposite direction. The marginals of the messages are denoted as m_x and V_x. The following update rules defined by Loeliger (Loeliger, et al., 2007) will be described using Forney's factor graph with arrows. The arrows are not necessary but make the direction of the factor graph more evident.

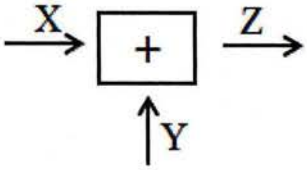
When two branches, X and Y, are directed toward an equality node, the resulting message of edge Z has an equivalent marginal as the two individual incoming messages. This is demonstrated below in Figure 6. The same setup is shown in Figure 7 but with an addition node instead of the equality node. The equations are basic addition and mean subtraction for computation of the distributions at edge Z and the edge X in the opposing direction.



$$m_X = m_Y = m_Z$$

$$V_X = V_Y = V_Z$$

Figure 6: Gaussian messages for basic equality node. Image based on (Loeliger, et al., 2007).



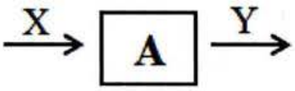
$$\overrightarrow{V_Z} = \overrightarrow{V_X} + \overrightarrow{V_Y} \qquad \overleftarrow{V_X} = \overleftarrow{V_Z} + \overrightarrow{V_Y}$$

$$\overrightarrow{m}_Z = \overrightarrow{m}_X + \overrightarrow{m}_Y \qquad \overleftarrow{m}_X = \overleftarrow{m}_Z - \overrightarrow{m}_Y$$

$$m_X + m_Y = m_Z$$

Figure 7: Gaussian messages for basic addition node. Image based on (Loeliger, et al., 2007).

Figure 8 illustrates the computation of the covariance, mean, and marginals for forward matrix multiplication in a factor graph where *A* is a matrix.



$$\overrightarrow{V_Y} = A\overrightarrow{V_X}A^H$$

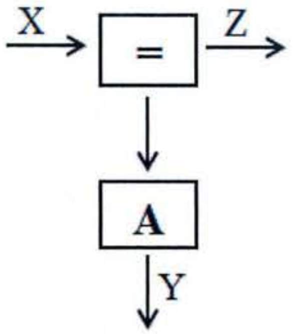
$$\overrightarrow{m}_Y = A\overrightarrow{m}_X$$

$$V_Y = AV_XA^H$$

$$m_Y = Am_X$$

Figure 8: Gaussian messages for matrix multiplication. Image based on (Loeliger, et al., 2007).

Last, when combining a branch of matrix multiplication with the equality node, a more complex calculation is formed and shown in Figure 9.



$$\overrightarrow{V_Z} = \overrightarrow{V_X} - \overrightarrow{V_X} A^H G A \overrightarrow{V_X}$$

$$\overrightarrow{m}_Z = \overrightarrow{m}_X + \overrightarrow{V_X} A^H G (\overleftarrow{m}_Y - A \overrightarrow{m}_X)$$

$$\text{where } G \triangleq (\overleftarrow{V_Y} + A \overrightarrow{V_X} A^H)^{-1}$$

Figure 9: Gaussian messages for combined matrix multiplication and an equality node. Image based on (Loeliger, et al., 2007).

For backwards Gaussian distribution involving matrix multiplication, different steps are required, but are not utilized in this thesis and therefore, will not be covered. The reader may refer to (Loeliger, et al., 2007) for more information regarding this topic. (Loeliger, et al., 2007).

2.2 Signal Processing for Undersea Systems

2.2.1 Signal Detection Theory

Signal Detection Theory is used in various areas of study, such as medical, psychology, legal, digital communication systems, RADAR and SONAR systems, and speaker classification just to name a few (Van Trees, 2001), (Heeger, 2003-2007). In a general signal detection problem, a source produces an output of all possible scenarios, called hypotheses. These are labeled as H_0, $H_1, \ldots, H_N$ for N possible scenarios. In the case of digital communication, these could be H_1 and H_0 to represent a "one" or "zero" being transmitted, respectively. For the medical scenario if a radiologist is attempting to locate a tumor, H_1 would represent a tumor is present and H_0 would represent no tumor. In a SONAR or RADAR scenario, H_1 would represent the reflected signal is present and H_0 would mean that just noise is present. However, the observer does not know which one is the correct hypothesis. The probabilistic transition mechanism knows which of the hypotheses are correct and using the known conditional probabilities densities for each hypothesis, it projects a value into the observation space. The observation space is made up of a set of M observed variables, $r_1, r_2, \ldots, r_M$, which are elements in the vector $\boldsymbol{R}$. These elements are the assigned values from the source output added to random noise n. A diagram demonstrating the basic structure of signal detection theory is given in Figure 10. (Van Trees, 2001).

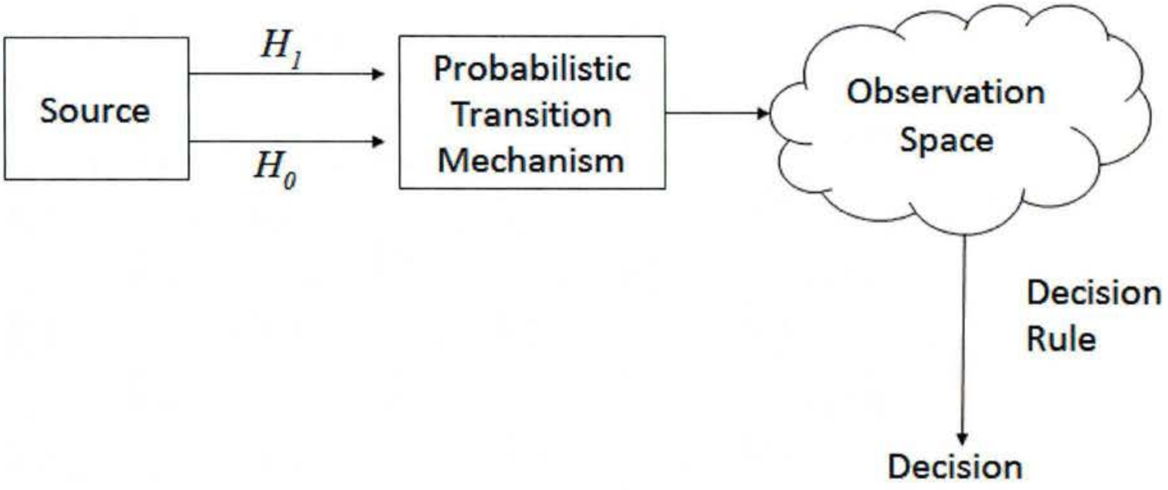


Figure 10: Basic operation of a signal detection theory example. Image based on (Van Trees, 2001).

When assuming a simple binary scenario, such as in SONAR, where H_1 and H_0 are the two hypotheses, the two known conditional probability densities sent by the transition mechanism would be $p(\mathbf{R}|H_1)$ and $p(\mathbf{R}|H_0)$, respectively. Given that either H_1 or H_0 must be true, there are four possible decisions that can be made. Two of which are the correct decisions: if H_0 is true and the decision were to select H_0, and if H_1 is true and selected. The first would be considered a *rejection*, and the later would be termed a *detection*. A *miss* would be made if H_1 were true but H_0 was selected. Last, if H_1 was selected when H_0 were true, then this would be considered a *false alarm*. A decision rule is required for the observer to determine which to choose. This decision can be made by setting a threshold λ, in which everything above the threshold, the observer chooses H_1 and for everything below, H_0 is chosen (Macmillan, 2002). Figure 11 illustrates the use of this threshold on the distribution curves for the noise and the signal plus noise.

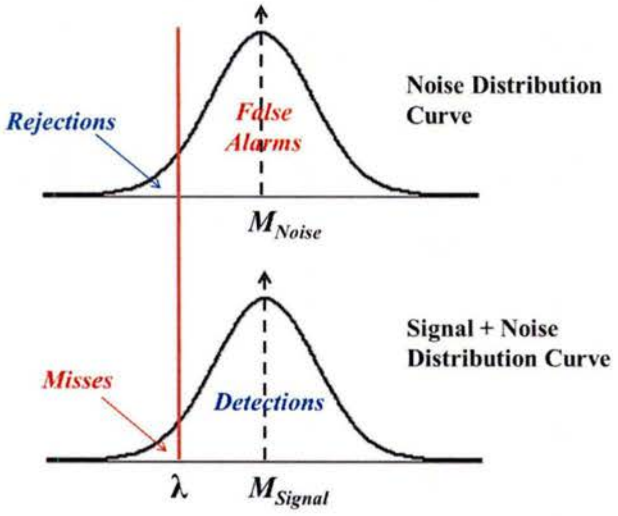


Figure 11: The noise and signal distribution curves with means M_{Noise} and M_{Signal} respectively, with a threshold, λ. Image based on (Macmillan, 2002).

Since every value above the threshold is chosen to be the signal, then that area under the noise curve would all be false alarms, where that same area under the signal curve would be correct detections. The area below the threshold for the noise distribution would be correct rejections, and missed detections when under the signal probability distribution curve.

The goal is to maximize the probability of detection, P_D, while minimizing the probability of a false alarm, P_F. By either using Bayes Criterion or Neyman-Pearson Criterion, both lead to the likelihood ratio, $\Lambda(\mathbf{R})$, which is given in the equation below.

$$\Lambda(\mathbf{R}) \triangleq \frac{p(\boldsymbol{R}\,|\mathrm{H}_1)}{p(\boldsymbol{R}\,|\mathrm{H}_0)} \tag{16}$$

In order to perform a likelihood ratio test, the threshold, λ, needs to be calculated. If using Bayes Criterion, the threshold is formed based on the *a priori* probabilities of the original source outputs for H_1 and H_0, given by P_1 and P_0, respectively. It is also determined by the cost, C_{ij} of each decision, where *i* denotes the hypothesis that was selected, and *j* is the hypothesis that is true. The assumption is made that the cost of a wrong decision is greater than the cost of a correct decision, as is shown below.

$$C_{10} > C_{00}$$
$$C_{01} > C_{11}$$

Using the costs and the *a priori* probabilities, the threshold can be computed as the given equation.

$$\lambda \triangleq \frac{P_0(C_{10} - C_{00})}{P_1(C_{01} - C_{11})} \tag{17}$$

When using the Neyman-Pearson criterion, a desired P_F is selected and set equal to the equation below. In order to find the threshold, the equation is solved for λ.

$$P_F = \int_{\lambda}^{\infty} p\left(\Lambda\,|H_0\right) d\Lambda \tag{18}$$

For both criterions, the likelihood ratio test is given as

$$\Lambda(\mathbf{R}) \underset{H_0}{\overset{H_1}{\gtrless}} \lambda \tag{19}$$

The likelihood ratio test explains that if the likelihood ratio is greater than the threshold, the selection of H_1 should be made. If the ratio is less than the threshold, the selection should be H_0. A final illustration of the signal detection theory using the threshold is given in the following figure. (Van Trees, 2001).

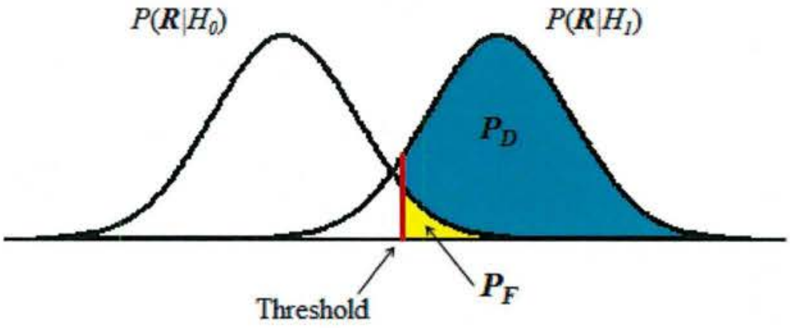


Figure 12: The possible results using signal detection theory. Image based on (Van Trees, 2001).

In the above figure, the two overlapping curves are the conditional probability densities for H_1 and H_0. The threshold is marked in red in the center. The cyan shaded area represents the P_D, while the yellow shaded area is the P_F. The figure displays how the use of the threshold is reducing the probability of false alarms while increasing the probability of detection.

2.2.2 Classification

A classifier is a model predicting a class label for an object given the vector of features supplied by a feature extractor. They are used heavily in the medical field, for instance when classifying a disease, and also in text and language, such as detecting spam within emails. In naval applications, classification is used to identify the class of the undersea object being detected and/or tracked. Within a classifier, the probabilities for the different possible classes are calculated since it is highly improbable to always achieve perfect classification (Duda, Hart, & Stork, 2001). One of the most common forms of a classifier is a Bayesian network.

2.2.2.1 Bayesian Network

A Bayesian network, or Bayesian belief network, is a directed graphical model of a directed graph $G = (V, E)$ where V and E are vertices and edges of the graph, respectively. The probability distributions of these directed graphs factorize as the product of the local functions, given by

$$p(\boldsymbol{y}, \boldsymbol{x}) = \prod_{v \in V} p(v|v^{\pi}) \tag{20}$$

where y is an element of the output variables, x is an element of the input variables, and v^{π} are the parents of v. The set of nodes that come before node v are known as the parents of v, likewise, the nodes that come after v are the children of v. To understand the notation of a Bayesian belief net, Figure 13 will be used as a source of referral. In the belief net, the bold capital letters are nodes with their associated variables, or states, given by the respective lowercase letters. For example, node **A** has variables a_1, a_2, ..., labeled as the feature **a**. The edges between each node are directional and represent the conditional probabilities. For instance, the edge connecting **A** and **C** signifies the conditional probabilities $P(c_i|a_j)$ given by the matrix $P(\mathbf{c}|\boldsymbol{a})$. The nodes that are not connected by an edge are considered to be conditionally independent.

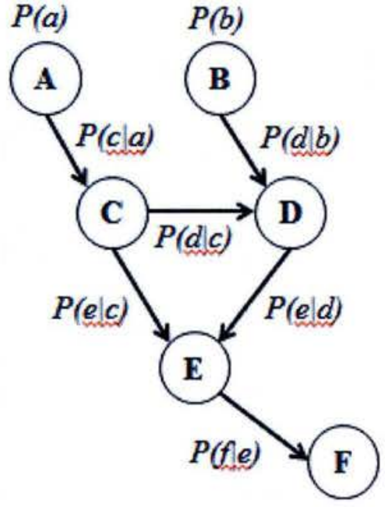


Figure 13: A Bayesian belief net where the bold capital letters are nodes with their associated variables, or states, given by the respective lowercase letters. The edges between each node are directional and represent the conditional probabilities. Image based on (Duda, Hart, & Stork, 2001).

A belief net operates by calculating the state of a node based on the states of the connected nodes. Figure 14 illustrates a section of a Bayesian belief net having nodes $\boldsymbol{\pi}_1$, $\boldsymbol{\pi}_2$, ..., $\boldsymbol{\pi}_n$, **X**, and **C**$_1$, **C**$_2$, …, **C**$_m$ where $\boldsymbol{\pi}_1$, $\boldsymbol{\pi}_2$, ..., $\boldsymbol{\pi}_n$ are **X**'s parents, $\boldsymbol{\pi}$, and nodes **C**$_1$, **C**$_2$, …, **C**$_m$ are **X**'s children, ***C***. If the variable values of all nodes besides **X** are known, a.k.a. the evidence, then the relative probabilities of the set of variables in **x** on node **X**, given the collected evidence **e**, at all other nodes, is known as the belief of **x**, denoted as P(**x**|**e**). Equation 21 shows how P(**x**|**e**) is reliant on both, the parents and children.

$$P(\boldsymbol{x}|\boldsymbol{e}) \propto P(\boldsymbol{e}^{C}|\boldsymbol{x})P(\boldsymbol{x}|\boldsymbol{e}^{\pi}) \tag{21}$$

The evidence of the parent and children nodes is given by $\boldsymbol{e}^{\pi}$ and $\boldsymbol{e}^{C}$, respectively. To find the final belief of **x**, Equation 21 will be normalized over the states at **X**. (Duda, Hart, & Stork, 2001).

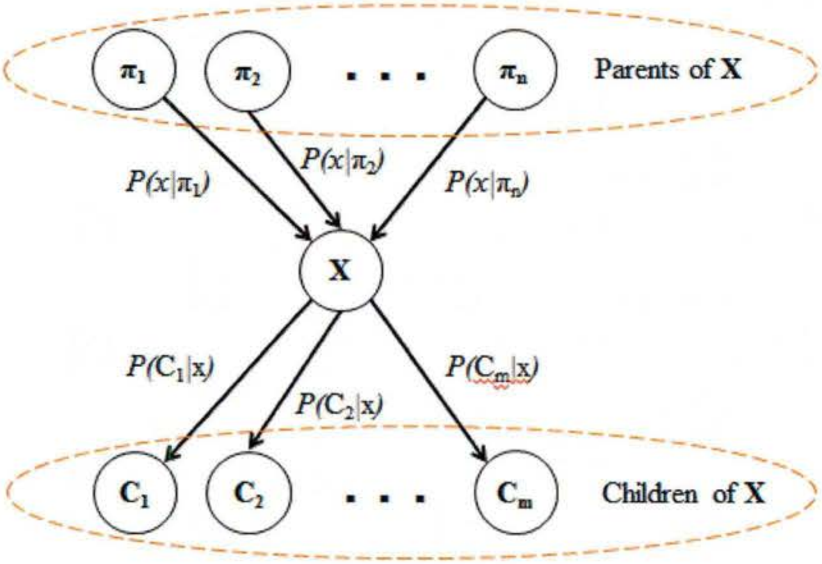


Figure 14: A section of a Bayesian belief net having nodes $\pi_1, \pi_2, \ldots, \pi_n$, X, and $C_1, C_2, \ldots, C_m$ where $\pi_1, \pi_2, \ldots, \pi_n$ are X's parents, and nodes $C_1, C_2, \ldots, C_m$ are X's children. Image based on (Duda, Hart, & Stork, 2001).

The first factor of Equation 21 is the product of the children's independent likelihoods. Equation 22 gives the expansion of the first factor from Equation 21.

$$P(\boldsymbol{e}^{C}|\boldsymbol{x}) = \prod_{j=1}^{|C|} P(\boldsymbol{e}_{C_j}|\boldsymbol{x}) \tag{22}$$

In the equation above, C_j denotes the *j*th child node, while the values of the probabilities of its variables is given by $\boldsymbol{e}_{C_j}$. The number of elements in set *C* is given by $|C|$, and $|\pi|$ is likewise, the number of elements in set π, which will be seen in the equation below for the second factor of Equation 21.

$$P(\boldsymbol{x}|\boldsymbol{e}^{\pi}) = \sum_{all\,\pi_{mn}} P(\boldsymbol{x}|\pi_{mn}) \prod_{i=1}^{|\pi|} P(\pi_i|\boldsymbol{e}_{\pi_i}) \tag{23}$$

Equation 23 shows the reliance the belief has on the parent nodes where π_{mn} represents the specific value for variable *n* on parent node π_m. Comparable to Equation 22, π_i denotes the *i*th parent node, while the values of the probabilities of its variables is given by $\mathbf{e}_{\pi_i}$. Equation 24 shows the proportionality of the belief of **x** to the combination of the two previously defined equations.

$$P(\boldsymbol{x}|\boldsymbol{e}) \propto \prod_{j=1}^{|C|} P(\boldsymbol{e}_{C_j}|\boldsymbol{x}) \sum_{all\,\pi_{mn}} P(\boldsymbol{x}|\pi_{mn}) \prod_{i=1}^{|\pi|} P(\pi_i|\boldsymbol{e}_{\pi_i})$$

(24)

The final belief is found when Equation 24 is normalized over the variables at node **X**. (Duda, Hart, & Stork, 2001).

2.2.2.2 Naïve Bayes Classifier

The naïve Bayes classifier can be illustrated as a simple Bayesian belief net, where the features are conditionally independent. It has shown to be a very efficient, and in certain cases, an optimal classifier, even though the assumption of complete conditional independence of features is very unlikely to be true in real-world scenarios (Zhang, 2004). Equation 25 gives the joint probability formula of which the naïve Bayes classifier is founded.

$$p(y, \boldsymbol{x}) = p(y) \prod_{k=1}^{K} p(x_k|y)$$

(25)

In the above equation, y is the class variable and $\mathbf{x} = (x_1, x_2, ..., x_K)$ are features for the object (Sutton & McCallum). Figure 15 illustrates an example of a naïve Bayes classifier.

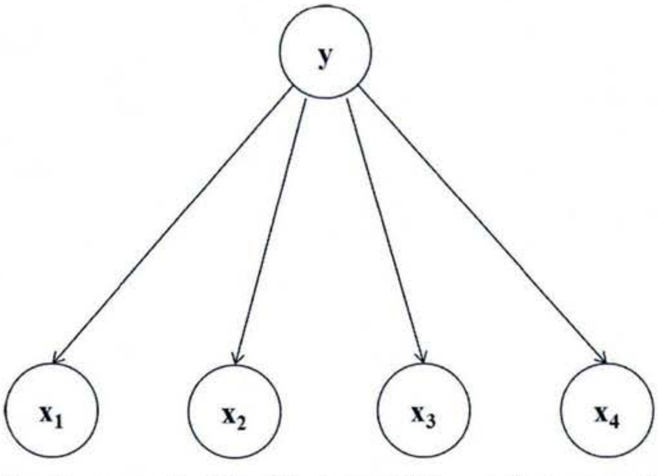


Figure 15: An example of a naïve Bayes classifier with class variable y and features $\mathbf{x} = (x_1, x_2, \ldots, x_K)$.

2.2.3 Tracking

2.2.3.1 Kalman Filtering

The Kalman Filter was first introduced by Rudolf E. Kalman in 1960 when he published his paper describing a recursive solution to the linear filtering and prediction problems (Kalman, March 1960). It is a data fusion algorithm most commonly used for data smoothing and providing the best mathematical estimate for the state parameters. The Kalman Filter is applied to many devices used today such as satellite navigation and control devices in vehicles, smart phones, and certain computer games. (Faragher, 2012). According to (Maybeck, 1979), the job of a filter is to find the best possible estimate with the least amount of error of the state given the noisy data collected. To reach this goal, the filter uses the measurements being collected in the data to portray the conditional probability density of the state. The word conditional is used due to the measurements being the deterministic factor of the shape and location along the x-axis. The shape of the conditional probability density graph determines the confidence in the information at the specific x-value. From this function the best estimate can be determined using, most commonly, the mean estimate, which is the "center of probability mass", the mode estimate, which is the x-value at the highest density point, or the median estimate, which is the x-value dividing the density in half. The Kalman filter is determined to be the best filter under the circumstances where the system model is linear and the process and measurement noises are both white and Gaussian which causes the mean, mode, and median of the estimate to all lie on the same x-value. (Maybeck, 1979).

Suppose a signal s_n, needs to be projected, given real measurement data ($z_0, z_1, \ldots, z_n$), and the signal and measurement data are categorized by the autocorrelation and cross-correlation functions (Sorenson, 1970). Using the reference (Welch & Bishop, 2006), the procedure for estimating the real state x, of a discrete-time controlled process, will be discussed. The state x at time step k, is defined by the linear stochastic differential equation

$$x_k = Ax_{k-1} + Bu_{k-1} + w_k \tag{26}$$

where w is zero-mean, white, Gaussian process noise, and A is an n x n matrix relating the state at the prior time step $k - 1$, to its current time step k. For this demonstration A will remain constant, however, it is possible that A may change with each iteration in real life applications. The variable u, is a real control input that is optional and related to the state x, by the n x l matrix B. The measurement for time step k, is given by

$$z_k = Hx_k + v_k \tag{27}$$

where v is zero-mean, white, Gaussian measurement noise independent from the process noise w, and H is an m x n matrix that describes the relationship between the measurement and state, which will remain constant for this demonstration but is subject to change in a real life application. The covariances of the process noise and measurement noise, Q and R respectively, are assumed to also be constant here, but could change with each time step or measurement.

Let $\hat{x}_k^-$ be real and the *a priori* estimate of the state at time k given x_{k-1} and let $\hat{x}_k$ be the *a posteriori* real state estimate at time k given the measurement, z_k. The errors of both state estimates, *a priori* and *a posteriori*, can now respectively be given as

$$e_k^- = x_k - \hat{x}_k^-$$

$$\text{and} \quad e_k = x_k - \hat{x}_k.$$

From these errors, we can determine the covariances of the two *a priori* and *a posteriori* error estimates to be

$$P_k^- = E[e_k^- e_k^{-T}]$$

$$\text{and} \quad P_k = E[e_k e_k^T].$$

In order to reduce the *a posteriori* error covariance, the *n* x *m* matrix *K* is introduced, also known as the Kalman gain, and is defined as

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1}. \tag{28}$$

The weight of the gain affects the level of confidence put on the measurement, z_k, and the predicted measurement, $H\hat{x}_k^-$. For instance, as the *R*, the error covariance of the measurement, approaches zero, the confidence in the measurement, z_k, increases and decreases in the predicted measurement, $H\hat{x}_k^-$. However, as the covariance of the *a priori* estimate error, P_k^-, approaches zero, the opposite occurs. The confidence in the measurement, z_k, decreases and the confidence in the predicted measurement, $H\hat{x}_k^-$, increases.

The Kalman filter process is a cycle between forming estimates of current states, then updating those estimates from measurements received, and finally, using those updates to predict the next *a priori* estimate. The cycle works by first estimating the "process state" at a time, *k*, and then acquires data consisting of measurements taken in a noisy atmosphere. Equations that are used to predict the estimates of the state and error covariance in order to form the *a priori* estimate will be called "time update equations". Using A and B from Equation 26, these equations are defined by

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1} \tag{29}$$

and $$P_k^- = AP_{k-1}A^T + Q. \tag{30}$$

Once the measurement data is applied and the estimate is adjusted, or corrected, then the *a posteriori* estimate is formed and then used to predict the *a priori* estimate at the next state. The equations used to apply the information from the measurements are to be called the "measurement update equations" and are given by the Kalman gain (Equation 28), along with the *a posteriori* state estimate equation, and the *a posteriori* error covariance estimate, all of which are given below, respectively, as

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1}$$
$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{31}$$

$$P_k = (I - K_k H)P_k^-. \tag{32}$$

From here, the process begins again using the recently found *a posteriori* estimates to calculate the next *a priori* estimates, as is demonstrated in the figure below.

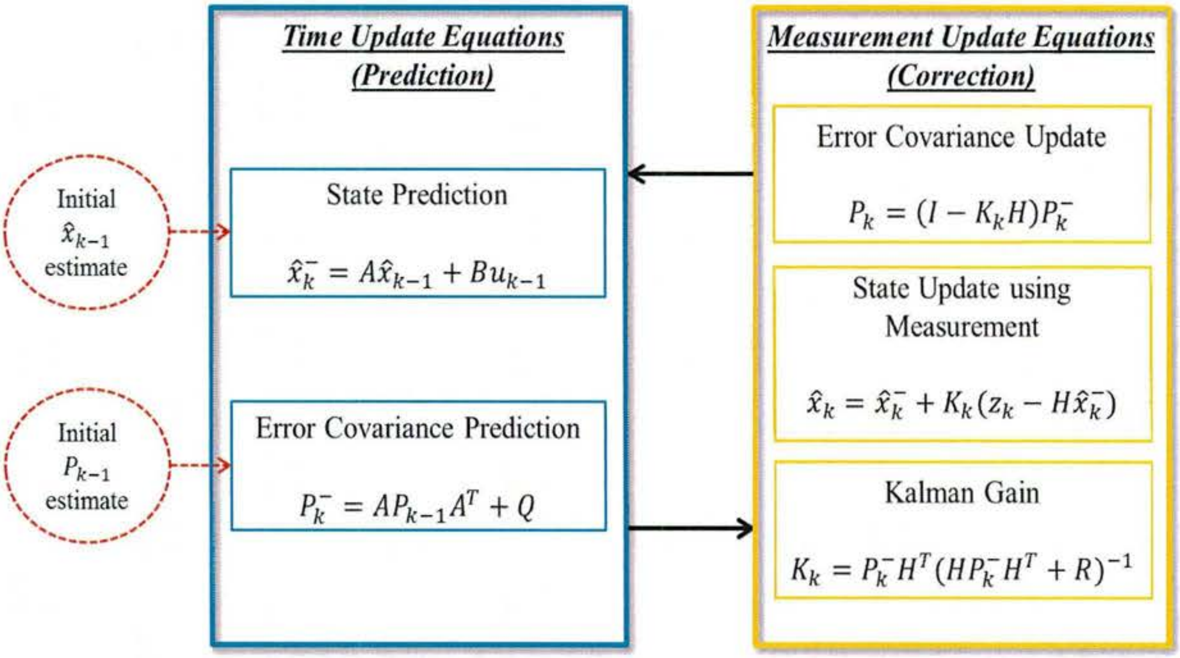

Time Update Equations
(Prediction)
Measurement Update Equations
(Correction)
Initial
$\hat{x}_{k-1}$
estimate
Initial
P_{k-1}
estimate
State Prediction
$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$
Error Covariance Prediction
$P_k^- = AP_{k-1}A^T + Q$
Error Covariance Update
$P_k = (I - K_k H)P_k^-$
State Update using
Measurement
$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$
Kalman Gain
$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$

Figure 16: An image of the recursive process of the Kalman filter using the time and measurement update Equations, 28-32. Image based on (Welch & Bishop, 2006).

When performing the Kalman filter, typically R, the covariance of the measurement noise, is measured ahead of time. However, Q, the covariance of the process noise, is not so easily found and may need to be adjusted in order to achieve the best execution of the Kalman filter. Since we assumed that Q and R would remain constant, the Kalman gain and estimated error covariance, K_k and P_k respectively, will also reach a steady-state and remain there.

2.2.3.1 Particle Filtering

Particle filtering, also known as bootstrap filtering, was first introduced in 1993 and is another filter whose goal is to estimate the state of an active system given its noisy measurements (Gordon, Salmond, & Smith, 1993). Where the Kalman filter provides the solution to a linear problem with Gaussian noise, the particle filter has the benefit of being a nonlinear filter solving non-Gaussian scenarios. Built on point mass characterizations, or particles, of the required state probability densities, the particle filter executes a sequential Monte Carlo approximation (Ristic, Arulampalam, & Gordon, 2004). A Monte Carlo model illustrates the outcomes of all the possible decisions one can make along with the risk assessment associated with each choice (Palisade Corporation, 2015). The particle filter results in a probability distribution represented as a cloud of particles in contrast to giving closed form density expressions as with the Kalman filter. The surrounding particle cloud consists of a collection of random samples and their supplementary weights which are used to calculate future estimates of the state (Ristic, Arulampalam, & Gordon, 2004).

To initialize the particle filter, an original set of particles is generated $\{(\mathbf{x}, w)\}$ where **x** is the current state vector and *w* is a weight. Just like in the Kalman filter, a future state, **x'**, is estimated based on the prior time step of the state with associated process noise, *U*, and the measurement taken, *z*, with its associated measurement noise, *V*. Unlike the Kalman filter, the predicted state equation does not need to be linear, nor is the noise required to be Gaussian, therefore, the predicted state and measurement equations will take the general form shown below, respectively.

$$\boldsymbol{x}' = F(\boldsymbol{x}, \boldsymbol{U}) \tag{33}$$

$$\mathbf{z} = G(\boldsymbol{x}, \boldsymbol{V}) \tag{34}$$

Each particle is designated a specific weight generated given the measurement vector, **z**. The updated weight, *w*', is given by Equation 35.

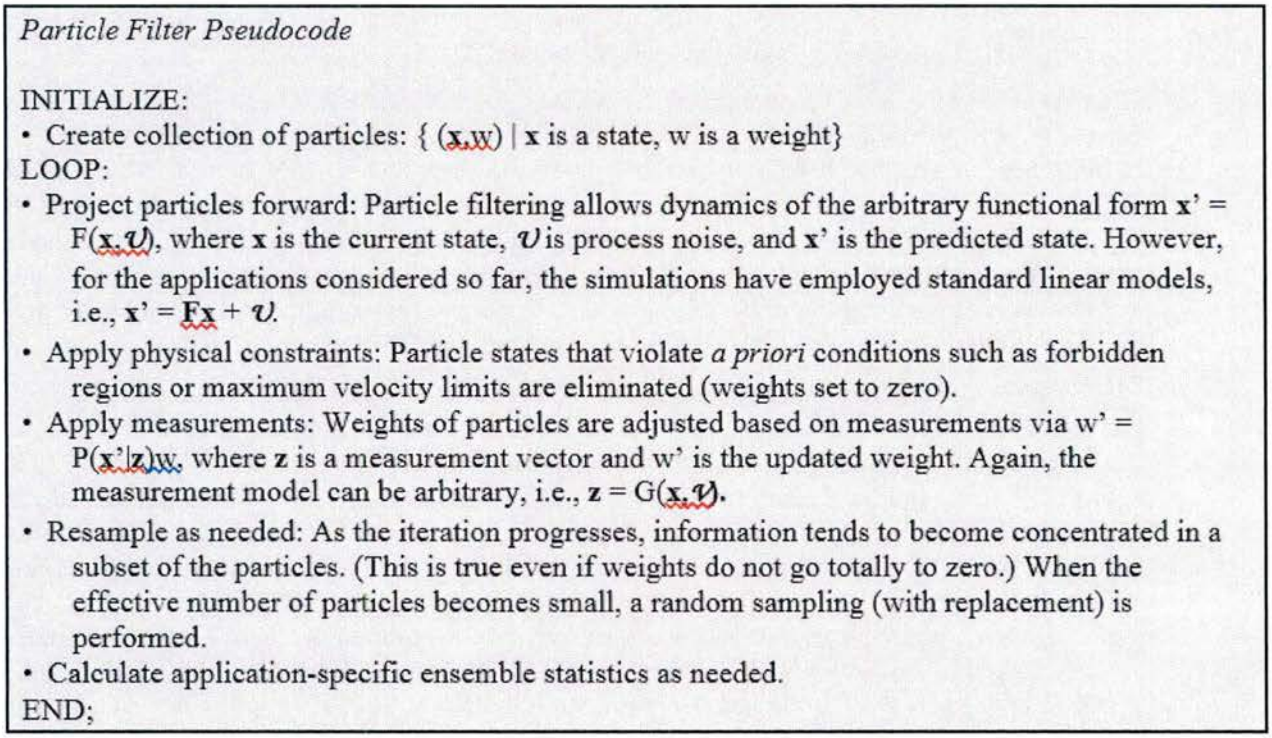
$$w' = P(\boldsymbol{x}'|\mathbf{z})\mathrm{w} \tag{35}$$

For particles whose predicted states are inconsistent with measurements or violate any *a priori* conditions, for instance breach a maximum velocity boundary, their weights are assigned as a zero, thus abolishing those particles.

A common problem that comes into play with particle filters is the problem of degeneracy. After repeating these steps a certain amount of times, the number of particles remaining to estimate future state possibilities, will reduce to a single particle. To prevent this problem, an effective sample size N_{eff} is estimated by the following equation.

$$\widehat{N}_{eff} = \frac{1}{\sum_{i=1}^{N}(w^i)^2} \tag{36}$$

In Equation 36, w^i is the weight for each sample, *i*, where $i = 1, \ldots, N$ samples. Once the effective sample size drops to reach a predetermined minimum, resampling becomes necessary. Resampling corrects the degeneracy problem by removing the low weighted samples and replacing them with multiples of the high weighted samples creating all uniform weights. The higher the sample weight is, the better chance that particle has of being selected to be multiplied. Figure 17 is a pseudocode illustrating the process of a typical particle filter. (Ristic, Arulampalam, & Gordon, 2004).



Particle Filter Pseudocode

```
INITIALIZE:
  • Create collection of particles: { ($\mathbf{x}$,w) | $\mathbf{x}$ is a state, w is a weight}
LOOP:
  • Project particles forward: Particle filtering allows dynamics of the arbitrary functional form $\mathbf{x}' = F(\mathbf{x}, \upsilon)$, where $\mathbf{x}$ is the current state, $\upsilon$ is process noise, and $\mathbf{x}'$ is the predicted state. However, for the applications considered so far, the simulations have employed standard linear models, i.e., $\mathbf{x}' = \mathbf{F}\mathbf{x} + \upsilon$.
  • Apply physical constraints: Particle states that violate a priori conditions such as forbidden regions or maximum velocity limits are eliminated (weights set to zero).
  • Apply measurements: Weights of particles are adjusted based on measurements via $w' = P(\mathbf{x}'|\mathbf{z})w$, where $\mathbf{z}$ is a measurement vector and w' is the updated weight. Again, the measurement model can be arbitrary, i.e., $\mathbf{z} = G(\mathbf{x}, \mathcal{V})$.
  • Resample as needed: As the iteration progresses, information tends to become concentrated in a subset of the particles. (This is true even if weights do not go totally to zero.) When the effective number of particles becomes small, a random sampling (with replacement) is performed.
  • Calculate application-specific ensemble statistics as needed.
END;
```

Figure 17: Particle filter pseudocode illustrating the typical process of a particle filter. (Sacha & Shaffer, 2010)

The downfall, compared to the Kalman filter, is the particle filter requires a high computational cost. The more particles that are used yield a more closely converged probability density but result in a more expensive cost. However, due to the speed of current technology, particle filters are still able to be implemented in the field nonetheless. Particle filtering is already being implemented in the field of interest addressed in this thesis: target tracking, as well as other areas, such as chemical engineering, computer vision, robotics, and financial econometrics (Doucet & Johansen, 2008).

2.2.3.1.1 Other Particle Filter Research Efforts

Goldhahn and Braca have recently performed research in this field of tracking undersea objects with a particle filter. Their research focuses on the use of autonomous underwater vehicles (AUVs) for a multi-static tracking configuration with multiple sensor and receiver pairs for anti-submarine warfare (ASW). They are simulating the use of horizontal line arrays as the receivers which suffer from port-starboard (left-right) ambiguity resulting in complications for different detection and tracking algorithms. The variation of the probability of detection due to the environment, in which the tracking is taking place, is also being applied to their simulation. They are using a particle filter to restructure the Bayesian posterior distribution of the state of the target given the collected information taken from the sensors. (Goldhahn, et al., 4-9 May 2014).

2.3 Continuous Active SONAR

2.3.1 Pulsed Active and Continuous Active SONAR

When trying to locate an underwater object, SONAR is used and can either be passive or active. Passive SONAR only listens for acoustic activity as opposed to active SONAR which transmits a signal and then listens for an echo. Conventionally, when using mobile active SONAR, a single vehicle transmits a signal which is then reflected from an object and received by the original transmitting vehicle. Once the echo, or ping, is received, the vehicle transmits another signal. This configuration, using a single transmitter/receiver, is known as monostatic SONAR (Yakubovskiy, 2009), and the mode of operation consisting of transmitting a short pulse followed by a long listen interval is referred to as Pulsed Active Sonar (PAS).

The vehicles referred to in this thesis will be assumed to be unmanned underwater vehicles (UUVs). Two types of UUVs exist: a UUV with a remote human operator, known as a remotely operated underwater vehicles (ROVs), and autonomous underwater vehicles (AUVs) which do not require any direct human operation (Dobbins, 2014). UUVs, such as the biologically inspired "Robo-lobster", can perform underwater tasks in shallow, rough waters (Singer, 2009). Other UUVs, such as a modified torpedo, like the Remote Environmental Monitoring Unit (REMUS), or a small submarine are able to operate in deeper waters (Singer, 2009). Currently UUVs are used as SONAR platforms to search for objects but future applications that have been identified include being able to position and recover devices, take action on all forms of information, and to be able to engage all types of targets (Department of the Navy, 2004).

PAS traditionally uses a monostatic configuration, in which it transmits a short signal burst, or chirp, via a frequency modulated waveform (FM) or by a short narrowband pulsed sinusoid. Even though these types of signals provide information regarding the range of the object being tracked, they may not offer much insight on Doppler. This will be discussed in more detail later. To avoid any loss of data, PAS must wait to hear the ping before transmitting another pulse (Grimmett & Wakayama, 2013). Since the speed of sound in water is only about 1500 m/s, the ping repetition interval (PRI) tends to be large relative to the transmit pulse length, giving a very low duty cycle as small as 1%. This high PRI makes tracking and classifying an object much more difficult since the rate of information recurrence is not high enough to accurately identify the signal echo versus noise, leading to the detection of more false alarms. (Hickman & Krolik, 2012).

To increase the transmission and reception rate, continuous active SONAR (CAS) can now be used. CAS uses a bistatic configuration, in which the transmitter and receiver are two different UUVs with a large enough separation analogous to the transmitter-to-object and object-to-receiver distances (Yakubovskiy, 2009). When using a bistatic configuration, the transmitter does not need to wait to hear the ping before sending another transmission, which allows CAS to have a transmit duty-cycle of theoretically 100%. This high duty-cycle allows for a much higher rate of information recurrence leading to more accurate detection, classification, localization and tracking of an object with a potentially lower number of false alarms (Grimmett & Wakayama, 2013). The downfall with having such a high duty-cycle is a decreased signal-to-noise ratio (SNR), also known as a signal-to-clutter-plus-noise ratio (SCNR), due to the continuous interference, referred to as the direct blast from the transmitter.

2.3.2 CAS Transmission Waveforms

There are different types of waveforms that can be used by CAS that result in different obtained information. One of these is a repeating linear frequency modulated waveform (LFM). An LFM allows for the opportunity to find the estimated range of the object being tracked based on the time delays for when echo is received. If using spectral processing to determine the time delay Δτ, the following equation may be used

$$\Delta\tau = \Delta f \frac{T_{PRI}}{B} \tag{37}$$

where Δf is the frequency shift obtained by using the Short Time Fourier Transform, T_{PRI} is the time duration of the ping repetition interval, and B is the total bandwidth of the LFM (Grimmett & Wakayama, 2013).

To demonstrate the information that can be found when using a LFM waveform, a continuous active SONAR scenario was constructed in MATLAB to transmit a LFM signal. The code simulated the interaction of an acoustic transmitter, a separately located receiver, and two targets and was designed based on the ray cone formulation in Figure 18. If the position of the transmitter is given at time t_1, along with the description of the receiver's motion path, the code uses the forward ray cone formulation to calculate the time the signal reaches the receiver, thus, allowing for the position of the receiver to be found. A backward ray cone can map the energy backwards in time to the moment of transmit. (Ricker, 2003).

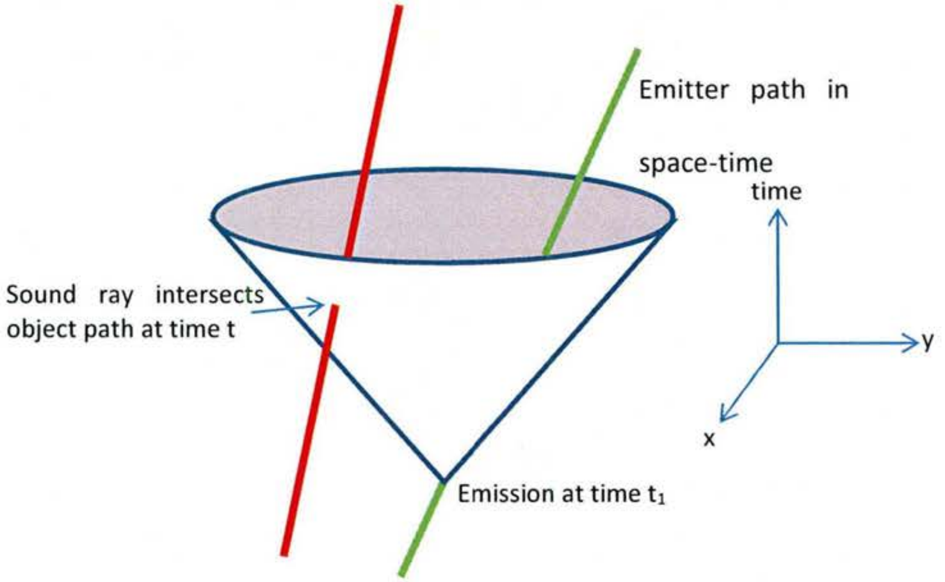


Figure 18: Forward Ray Cone. Image from NURP annual report.

The ray-cone code simulates the CAS signal received after being reflected form two different targets, and outputs the resulting spectrogram shown in Figure 19. The simulation confirmed that not only can the time delay of each of the targets be found when transmitting a LFM signal, but the time delay of the direct blast can also be detected. Learning the time delay for each target allows for the computation of range. However, there is very little information that can be observed about the frequency which means the Doppler of the targets cannot be determined unambiguously.

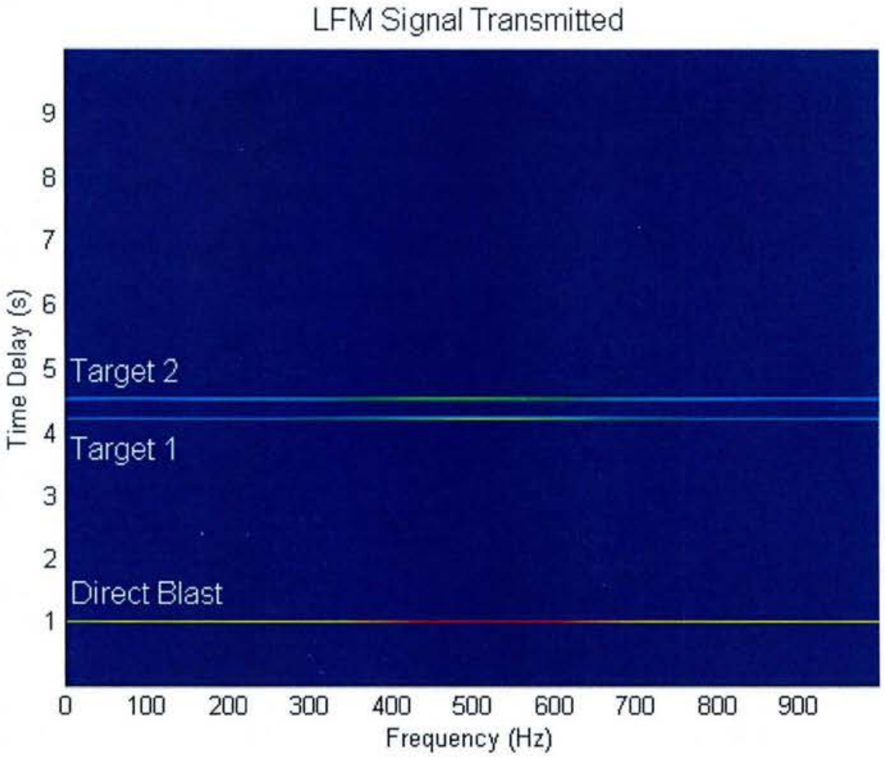


Figure 19: Spectrogram from the MATLAB simulation of the received data from the transmitted LFM signal

Different from a LFM signal, a continuous waveform (CW) can be continuously transmitted instead. When using a continuous CW signal, the Doppler shift of the target can be found and used to calculate the approximated continual bistatic range rate of change (Grimmett & Wakayama, 2013). The same MATLAB code as used previously, was run again, but simulating the transmission of a continuous CW signal. Its resulting spectrogram is shown in Figure 20. The figure shows the opposite result as when transmitting the LFM signal. The Doppler shift can now easily be observed, but the time delay can no longer be calculated. These results alone do not promote close state approximations for the detected object.

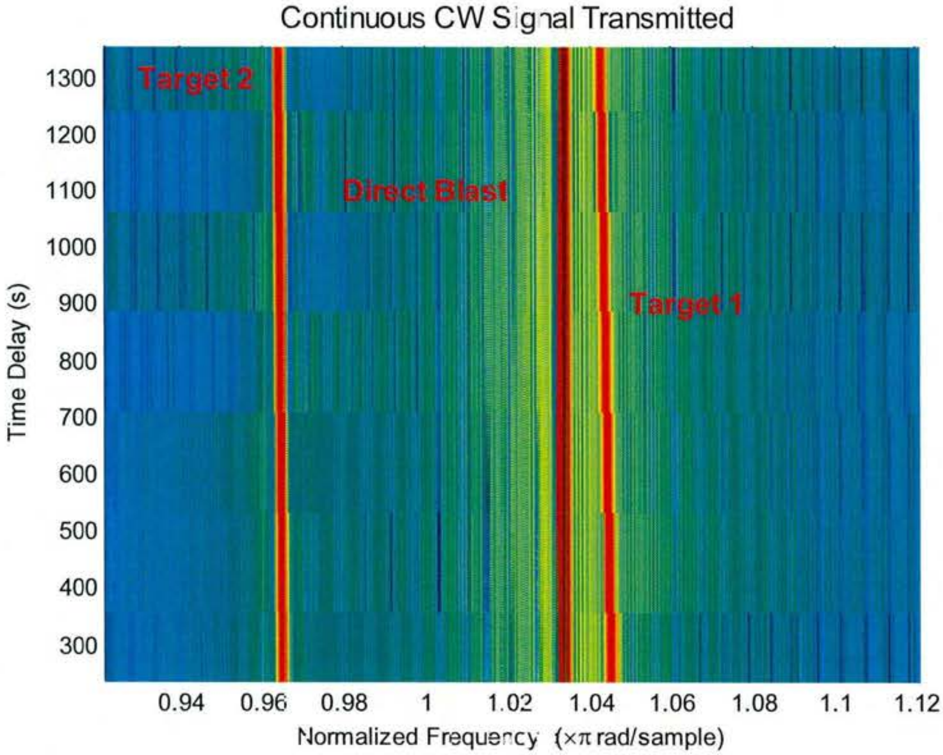


Figure 20: Spectrogram from the MATLAB simulation of the received data from the transmitted continuous CW signal

2.3.3 Other Signal Design Research Efforts

Much research is being done with different CAS waveforms and geometry configurations to try and achieve a closer estimate of the desired target state. There are refined broadband waveforms that try to combine both CWs and FMs in order to find both range and Doppler measurements of the detected object. Hickman and Krolik discuss the use of a "non-recurrent wideband linear FM signal with circular Costas frequency-staggering across chirp repetition intervals", referred to as a slow-time Costas-coded waveform. The waveform has been formulated to allow for high transmit rates while receiving both range and velocity estimates for each transmission, and also having a high SCNR (Hickman & Krolik, 2012).

Grimmett and Wakayama, on the other hand, have looked at simply using a continuous CW transmitted signal, but with using multiple transmitter-receiver pairs. They apply a Gaussian Mixture Probability Hypothesis Density filter using the Doppler-bearing measurements taken from the transmitted continuous CW signals. When they used the measurements from multiple source-receiver pairs, tracking the state of an object proved to be much more successful, as opposed to using a single source-receiver pair. (Grimmett & Wakayama, 2013).

Chapter 3: FACTOR GRAPHING FOR DCL

3.1 Detection

3.1.1 Detector Model

In a SONAR detection problem, the receiver must be able to detect the existence of the transmitted signal amongst noisy data to determine whether to pass the data along to the classifier. This section will set up a detector in the form of a loop-free factor graph. Since there are no loops, belief propagation can be used to understand the incoming data. From this data, the likelihood ratio can be found. For the SONAR problem, the detector is a binary problem producing a hard-decision between the hypotheses H_1 and H_0. If the transmitted signal is present, H_1 is the hypothesis, and H_0 denotes when the transmitted signal is not present. The received signal, R, is given by

$$R = hs + w \tag{38}$$

where s is the original transmitted signal, h is the channel gain, and $w \sim \mathcal{N}(0, \sigma_w^2)$ is zero-mean, white, Gaussian noise. It is assumed that s, h, and w are all independent from each other. This detector model can be inserted into the factor graph given in Figure 21 where $\hat{\theta}$ denotes the decision made by the Neyman-Pearson decision theory (NP) discussed in the next section. (Zarrin & Lim, 2008).

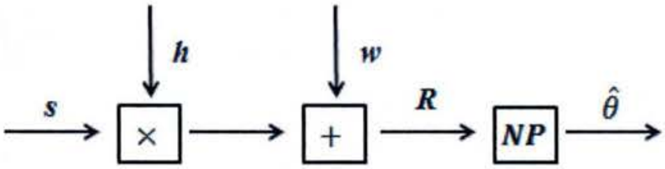


Figure 21: The detector factor graph outputting the decision, $\hat{\theta}$.

3.1.2 Neyman-Pearson Decision Theory

The decision, $\hat{\theta}$, is a binary decision of either H_0 or H_1, represented by the indices $\theta = \{0,1\}$ respectively. The Neyman-Pearson theory makes this decision by maximizing the probability of detection, P_D, while minimizing the probability of a false alarm, P_F. This method uses the likelihood ratio, $\Lambda(\mathbf{R})$ given by Equation 16. A threshold of which to test the ratio against is determined by selecting the wanted P_F and then solving Equation 18 for the threshold, λ. The likelihood ratio test is conducted by comparing the likelihood ratio to the threshold as in Equation 19 and shown again below.

$$\Lambda(\mathbf{R}) \underset{H_0}{\overset{H_1}{\gtrless}} \lambda$$

If the likelihood ratio is greater than the threshold, the factor graph will output $\hat{\theta} = 1$, and it will output $\hat{\theta} = 0$ if the likelihood ratio is less than the threshold.

3.2 Classification

3.2.1 Naïve Bayes Classifier

The naïve Bayes classifier has been established as an efficient classifier in real-world scenarios with little training time (Zhang, 2004). This being the case, it is logical to utilize this form and transform it into a factor graph. The transformation is quite simple. Figure 22 illustrates the factor graph of the naïve Bayes classifier with class node, *y*, and features **x** = (x_1, x_2, ..., x_K). The solid blue squares represent the edge factor nodes $\psi_k(y, x_k)$. To determine the closed form solutions of this factor graph, all that is required is the defining of a factor node $\psi(y)$ as the input, and the edge factors $\psi_k(y, x_k)$. The factor node $\psi(y)$ is equal to the *a priori* probability of class node *y*, $p(y)$, and the edge factors $\psi_k(y, x_k)$ equal the conditional probability for each feature x_k, $p(x_k|y)$. These functions are directed through belief propagation to produce the closed form solution given in Equation 39.

$$p(y, \boldsymbol{x}) = \psi(y) \prod_{k=1}^{K} \psi_k(y, x_k) \tag{39}$$

After substituting the probabilities into the appropriate factor nodes, the resulting equation is equivalent to the joint probability formula for the naïve Bayes classifier given in Equation 25. (Sutton & McCallum).

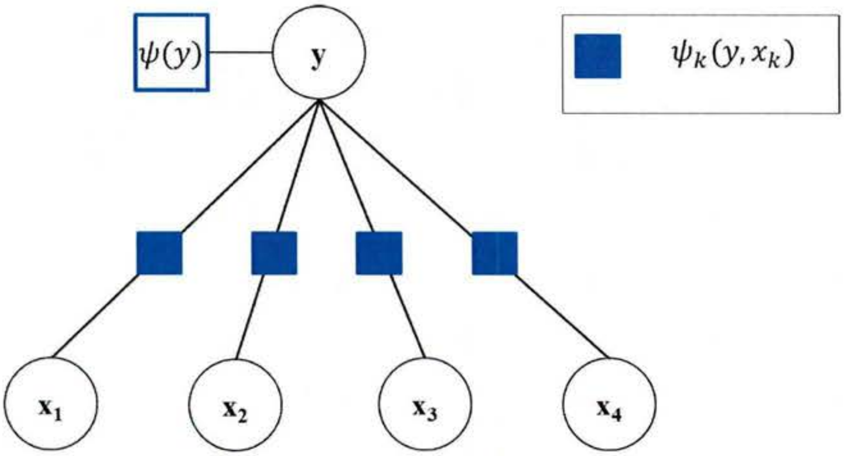


Figure 22: The factor graph form of the naïve Bayes classifier with class variable y and features **x** = (x_1, x_2, ..., x_K). *The blue squares represent the edge factors* $\psi_k(y, x_k)$. *Image based on (Sutton & McCallum).*

3.3 Tracking

3.3.1 Kalman Filter

The Kalman filter requires measurement updates from the reflected signal in the form of an update equation, along with a process update equation in order to locate the current state of the object being tracked. The state *x* at time step *k* will be defined by the linear stochastic differential equation

$$x_k = Ax_{k-1} + w_k, \tag{40}$$

where w_k is white Gaussian process noise, such that $w_k \sim N(0, \sigma_w^2)$, and *A* is an *n* x *n* matrix relating the state at the prior time step $k - 1$, to its current time step *k*. For this demonstration, *A* will remain constant and there will be no real control input. The measurement *z* at time step *k* is defined by

$$z_k = Hx_k + v_k,$$

where *v* is white Gaussian measurement noise, represented by $v_k \sim N(0, \sigma_v^2)$, and *H* is an *m* x *n* matrix that describes the relationship between the measurement and state, and will remain constant for this setting. *Q* and *R* will represent the covariances for the process and measurement noise, respectively, and will also remain constant for this demonstration.

For the scope of this research, the Kalman filter will use Equations 41 and 30 (both shown below) to predict the estimates of the state and error covariance to form the *a priori* estimate.

$$\hat{x}_k^- = A\hat{x}_{k-1}$$

$$\text{and} \quad P_k^- = AP_{k-1}A^T + Q, \tag{41}$$

where $\hat{x}_k^-$ is the real *a priori* estimate of the state at time *k* given x_{k-1} and $\hat{x}_{k-1}$ is the *a posteriori* real state estimate at time $k - 1$ given z_{k-1}. The variable P_k^- is the covariance of the *a priori* estimate error and P_{k-1} is the covariance of the *a posteriori* error at time step $k - 1$.

The *a posteriori* estimate is fashioned after the previous estimate has been adjusted from the application of the measurement data. It is then used to predict the *a priori* estimate at the next state. Three equations are used to apply the measurement updates. These equations are the Kalman gain K_k, the *a posteriori* state estimate $\hat{x}_k$, and the *a posteriori* error covariance estimate P_k, and are shown below.

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$

$$P_k = (I - K_k H)P_k^-$$

The goal is to fit a tracker into a factor graph model and have it track with a similar efficiency as the Kalman filter equations. Some variables will appear different to avoid conflicting with similar variables but will be addressed.

To set up the factor graph, the two state-space equations need to be found. If the assumption is made that the probability densities are Gaussian and the propagation is linear then the process state-space equation for the state of an object at time k is given by Equation 42.

$$x_k = Ax_{k-1} + w_k \tag{42}$$

The measurement taken of the object at time k is given by Equation 43

$$z_k = c_k^T x_k + v_k, \tag{43}$$

where c is defined by an n x m transposed matrix of ones and zeros describing the relationship between the measurement and the state. In this problem, unlike the Kalman filter, the noise w_k and v_k are not assumed to have zero-mean, while x_k and z_k are still the state and measurement at time step k, respectively. As before, A is still an n x n matrix relating the state at the previous time step $k - 1$ to the state at the current step k. As described in (Loeliger, Jan, 2004), Figure 23 shows how these two state-space equations fit into a factor graph.

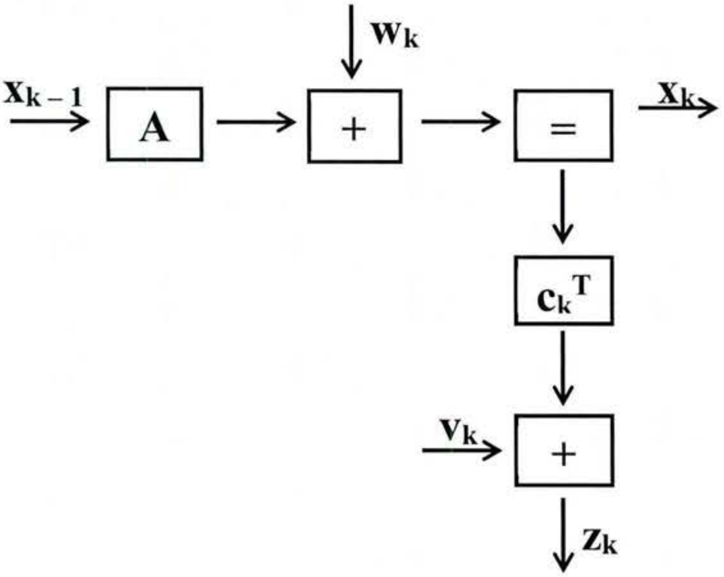


Figure 23: Factor graph of the Kalman filter with its two state-space equations

From this factor graph, Gaussian messages can be computed, resulting in the means m, and covariances V, of the state x and measurement z at time step k. By using the update rules for mean and covariance message computation outlined in (Loeliger, et al., 2007) and explained in section 2.1.2.2 of this thesis, the Gaussian messages were computed for both the state and measurement at time k, resulting in the following equations:

$$\begin{aligned}\vec{V}_{x_k} = A\vec{V}_{x_{k-1}}A^H + \vec{V}_{w_k} - \left(A\vec{V}_{x_{k-1}}A^H + \vec{V}_{w_k}\right) \times (c_k^T)^H \\ \times \left[\overleftarrow{V}_{z_k} + \vec{V}_{v_k} + c_k^T\left(A\vec{V}_{x_{k-1}}A^H + \vec{V}_{w_k}\right)(c_k^T)^H\right]^{-1} \times c_k^T\left(A\vec{V}_{x_{k-1}}A^H + \vec{V}_{w_k}\right),\end{aligned}$$

(44)

$$\begin{aligned}\vec{m}_{x_k} = A\vec{m}_{x_{k-1}} + \vec{m}_{w_k} + \left(A\vec{V}_{x_{k-1}}A^H + \vec{V}_{w_k}\right) \times (c_k^T)^H \\ \times \left[\overleftarrow{V}_{z_k} + \vec{V}_{v_k} + c_k^T\left(A\vec{V}_{x_{k-1}}A^H + \vec{V}_{w_k}\right)(c_k^T)^H\right]^{-1} \\ \times \left[\overleftarrow{m}_{z_k} - \vec{m}_{v_k} - c_k^T\left(A\vec{m}_{x_{k-1}} + \vec{m}_{w_k}\right)\right],\end{aligned}$$

(45)

$$\vec{V}_{z_k} = \vec{V}_{v_k} + c_k^T\vec{V}_{x_k}(c_k^T)^H,$$

(46)

$$\text{and} \qquad \vec{m}_{z_k} = \vec{m}_{v_k} + c_k^T\vec{m}_{x_k}$$

(47)

where $\vec{V}_{x_k}$ and $\vec{V}_{z_k}$ are the covariances of the state x and the measurement z at time step k, respectively. The means of the state x and the measurement z at time step k are represented by $\vec{m}_{x_k}$ and $\vec{m}_{z_k}$, respectively. Throughout these equations, the arrows above the covariances V, and means m, indicate the direction the messages are traveling within the factor graph.

In order to demonstrate the relevance between the Kalman filter equations and these Gaussian messages from the factor graph, the derivation for the covariance of the state x at time k, $\vec{V}_{x_k}$, will be explained and compared to the *a posteriori* equations and Kalman gain from the Kalman filter, that are used to predict the next *a priori*. First, to make the derivation clearer, certain edges of the previous factor graph will be labeled and marked by circled numbers in order to break down each step of the derivation. The edited factor graph is shown in Figure 24 below.

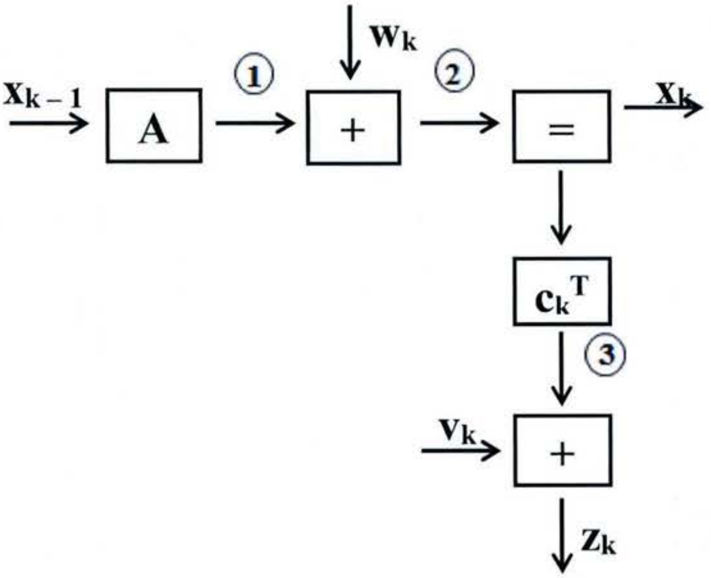


Figure 24: Factor graph of the Kalman filter with certain edges numbered and circled for derivation.

Beginning from the leaf at x_{k-1} to the circled number 1, the covariance can be written as

$$\vec{V}_1 = A\vec{V}_{x_{k-1}}A^H.$$

To find the covariance at the circled number 2, the covariances of the first two branches are simply added together resulting in Equation 48.

$$\vec{V}_2 = \vec{V}_1 + \vec{V}_{w_k} = A\vec{V}_{x_{k-1}}A^H + \vec{V}_{w_k} \tag{48}$$

Before the covariance of x_k can be found, the branch beneath the "=" node must be addressed. As before, the derivation must begin at the leaves and calculate the covariance at the circled number 3. Since the goal is to find the covariance of x_k, the covariance at 3 must be calculated for the opposite direction the arrow is pointing, which Equation 49 shows us.

$$\overleftarrow{V}_3 = \overleftarrow{V}_{z_k} + \vec{V}_{v_k} \tag{49}$$

Finally, all the pieces are known to complete the derivation of the covariance of the state *x* at time step *k*. Since the node connected to x_k is an equal sign, the last step is more complex than simply adding the two branches. Equation 50 gives the final step before substituting the previously found covariances.

$$\vec{V}_{x_k} = \vec{V}_2 - \vec{V}_2(c^T)^H G c_k^T \vec{V}_2 \tag{50}$$

$$\text{where } G \triangleq \left(\overleftarrow{V}_3 + c_k^T \vec{V}_2 (c_k^T)^H\right)^{-1}$$

Equation 44 yields the results after substituting in Equations 48, 49, and G into Equation 50 and is shown again below.

$$\begin{aligned}\vec{V}_{x_k} = A\vec{V}_{x_{k-1}}A^H + \vec{V}_{w_k} - \left(A\vec{V}_{x_{k-1}}A^H + \vec{V}_{w_k}\right) \times (c_k^T)^H \\ \times \left[\overleftarrow{V}_{z_k} + \vec{V}_{v_k} + c_k^T\left(A\vec{V}_{x_{k-1}}A^H + \vec{V}_{w_k}\right)(c_k^T)^H\right]^{-1} \times c_k^T\left(A\vec{V}_{x_{k-1}}A^H + \vec{V}_{w_k}\right).\end{aligned}$$

Once the covariance of x_k has been found the covariance of z_k can also be found by first finding the covariance of the circled number 3 for the direction it is now traveling and is given by Equation 51.

$$\vec{V}_3 = c_k^T \vec{V}_{x_k} (c_k^T)^H \tag{51}$$

Equation 51 is then used to compute the covariance of z_k shown in Equation 52.

$$\vec{V}_{z_k} = \vec{V}_{v_k} + \vec{V}_3 \tag{52}$$

Once $\vec{V}_3$ is substituted, the resulting equation yields Equation 46, shown below.

$$\vec{V}_{z_k} = \vec{V}_{v_k} + c_k^T \vec{V}_{x_k} (c_k^T)^H$$

The full derivation of the mean of the state and measurement, *x* and *z*, at time step *k* can be viewed in Appendix A.

The resulting covariance $\vec{V}_{x_k}$ in Equation 44 from the factor graph can be compared to the *a posteriori* error covariance estimate P_k in Equation 32 from the Kalman filter. This will be broken down one step at a time beginning with the comparison of the Kalman filter's P_k^- to the factor graph's $\vec{V}_2$. In the Kalman filter, P_{k-1} represents the covariance of the state *x* at the previous time step *k – 1,* which is represented as $\vec{V}_{x_{k-1}}$ in the factor graph. *Q* represents the covariance of the process noise in the Kalman filter and can be related to $\vec{V}_{w_k}$, which is the covariance of the process noise in the factor graph. Since *A* is the same *n* x *n* matrix in both models, then the Kalman filter's *a priori* covariance estimate in Equation 30, P_k^- , can be related to the covariance $\vec{V}_2$ from the factor graph's Equation 48 and is illustrated below as

$$P_k^- = \vec{V}_2$$

$$AP_{k-1}A^T + Q = A\vec{V}_{x_{k-1}}A^H + \vec{V}_{w_k}.$$

Referring to the Kalman gain in Equation 28, *H* can be compared to *c* since they both are matrices relating the measurement to the state. The variable *R* in the Kalman filter is the covariance of the measurement noise, where $\overleftarrow{V}_3$ is the covariance of the measurement noise and a measurement covariance summed together. Since the covariance of the ideal position measurement values is zero, then *R* is equal to $\overleftarrow{V}_3$. Since it has already been shown above that $\vec{V}_2$ is relative to P_k^-, then it can be said that the Kalman gain K_k is equivalent to the expression below:

$$K_k = \vec{V}_2(c_k^T)^H G$$

The expression is shown below in Equation 53 with the substitutions made.

$$\begin{aligned} &P_k^- H^T(HP_k^- H^T + R)^{-1} \\ &\qquad = \left(A\vec{V}_{x_{k-1}}A^H + \vec{V}_{w_k}\right) \times (c_k^T)^H \times \left[\overleftarrow{V}_{z_k} + \vec{V}_{v_k} + c_k^T\left(A\vec{V}_{x_{k-1}}A^H + \vec{V}_{w_k}\right)(c_k^T)^H\right]^{-1} \end{aligned} \tag{53}$$

By distributing P_k^- in Equation 32, the following equation results:

$$P_k = P_k^- - K_k H P_k^-. \tag{54}$$

After analyzing all the pieces to the above equation and relating them to their corresponding sections from the factor graph model, the following assumption stating that Equations 32 and 44 are equal can be made:

$$\begin{gathered} P_k = \vec{V}_{x_k} \\ P_k^- - K_k H P_k^- = \vec{V}_2 - \left(\vec{V}_2(c_k^T)^H G\right)(c_k^T)\vec{V}_2 \end{gathered} \tag{55}$$

By analyzing the derivation of the mean of state *x* at time step *k,* $\vec{m}_{x_k}$ in appendix A, in a similar way, the *a posteriori* state estimate $\hat{x}_k$ from Equation 32 can be shown to be similar to $\vec{m}_{x_k}$ from Equation 45 of the factor graph model. The derivation of this comparison is shown in Appendix B.

These comparisons demonstrate that when utilizing a factor graph as a tracker, the resulting equations can reduce to the *a posteriori* estimate equations in the standard Kalman filter under the assumption of zero-mean. However, it is important to note that unlike the standard Kalman filter, the factor graph does not require the noise to be zero-mean. Instead, it is able to compute a track for biased measurements if needed.

3.3.1.1 Simulations

To verify these equations derived in the previous section, some simulations in MATLAB were conducted. First, a common process path needed to be plotted for both the Kalman filter and the factor graph to track. An ideal process path was created, along with a realistic process. The realistic path is simply the ideal process with added noise. Figure 25 is the plot of these two paths.

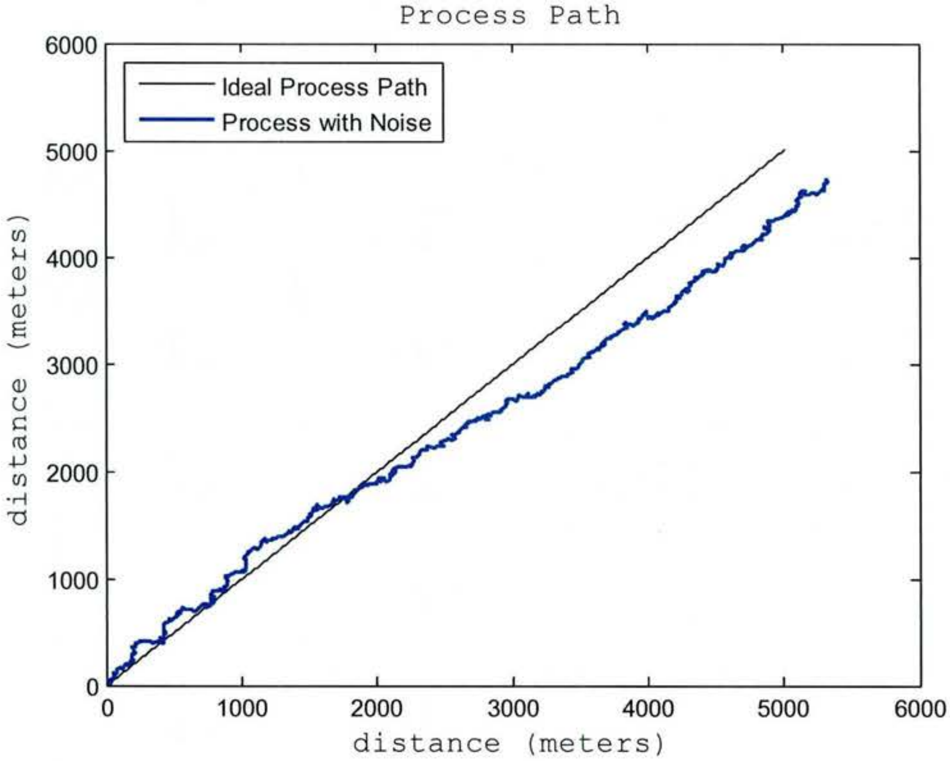


Figure 25: The ideal process path with the realistic path to be tracked by the Kalman filter and the factor graph

Once the realistic process path is graphed, both the Kalman filter and the factor graph are required to track the location of each point on the path. Figure 26 illustrates both the Kalman filter and factor graph tracking the process path.

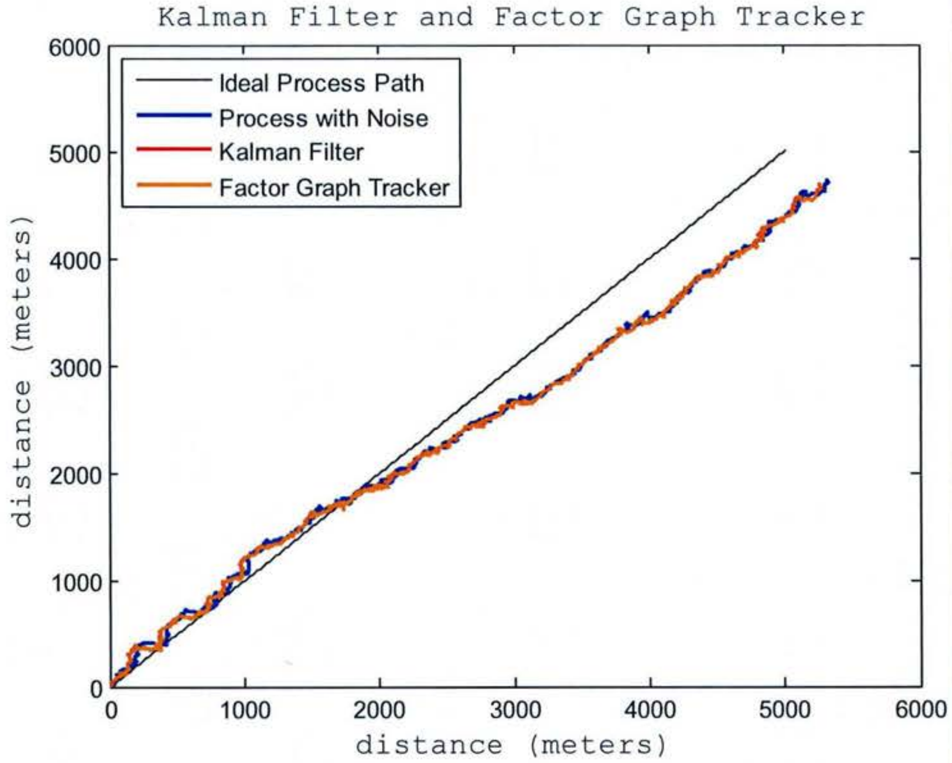


Figure 26: Plot of the Kalman filter and factor graph tracking the process path. The factor graph tracker plots the exact same path as the Kalman filter.

As seen in the graph, not only do both the factor graph and the Kalman filter successfully track the process path, but the two tracks align perfectly together. To further analyze the results of the factor graph and the Kalman filter, they were plotted against each other and then the deviations between the two were also plotted as seen in Figure 27 and Figure 28, respectively.

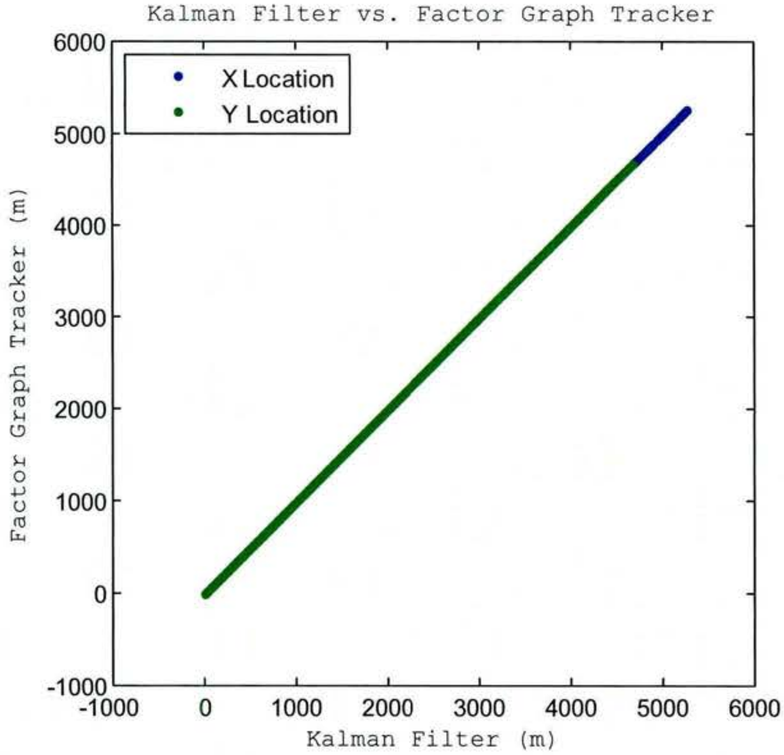


Figure 27: Plot of the x- and y-values of the Kalman filter against the x- and y-values of the factor graph.

The goal when plotting these two trackers against each other is to see a forty-five degree line across the graph showing the linear relationship of $y = x$ for both the x-values and the y-values. As demonstrated above, the result of the factor graph and Kalman filter plotted against each other is linear relationship $y = x$, where the x-values are plotted in blue and the y-values are in green. Figure 28 shows the exact differences between the values of the Kalman filter and factor graph.

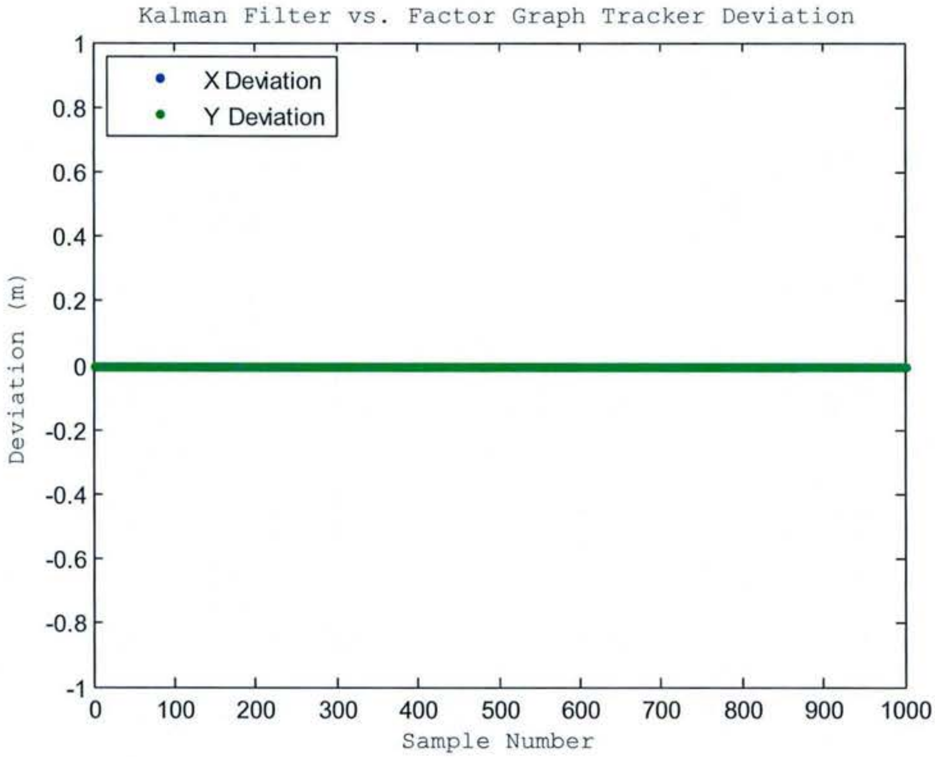


Figure 28: Plots the difference between the x- and y-values of the factor graph and the x- and y-values of the Kalman filter.

The above plot shows how different the values of the factor graph are to the Kalman filter. Since there is no deviation between the two tracks, they are identical.

These plots have proven that the factor graph tracker is capable of tracking the process path successfully. They have also confirmed that there is no difference between the standard Kalman filter and the factor graph tracker, which suggests that the factor graph tracker is an appropriate alternative to the standard Kalman filter.

Next, it is necessary to verify if the factor graph tracker is able to track the object when there is bias in the data. A MATLAB simulation was conducted in which a bias of 0.4 m/s was applied to the velocity of the object in the x-direction. Figure 29 demonstrates the speed of the object with the applied bias and the Kalman filter's and factor graph tracker's ability to track that object's speed. The figure displays the speed of the object in blue, the track from the Kalman filter in red, and the track from the factor graph tracker in orange. Figure 30 illustrates the velocity deviation of the tracks of the Kalman filter and factor graph tracker from the velocity of the object and Figure 31 shows the location deviation. In these figures, the deviation of the Kalman filter is in blue while the deviation of the factor graph tracker is in green. Last, the velocity and location root mean squared (RMS) errors were calculated for both the Kalman filter and factor graph tracker and are displayed in Table 1. The below figures and table verify the factor graph tracker's ability to track an object with bias much more efficiently than the standard Kalman filter.

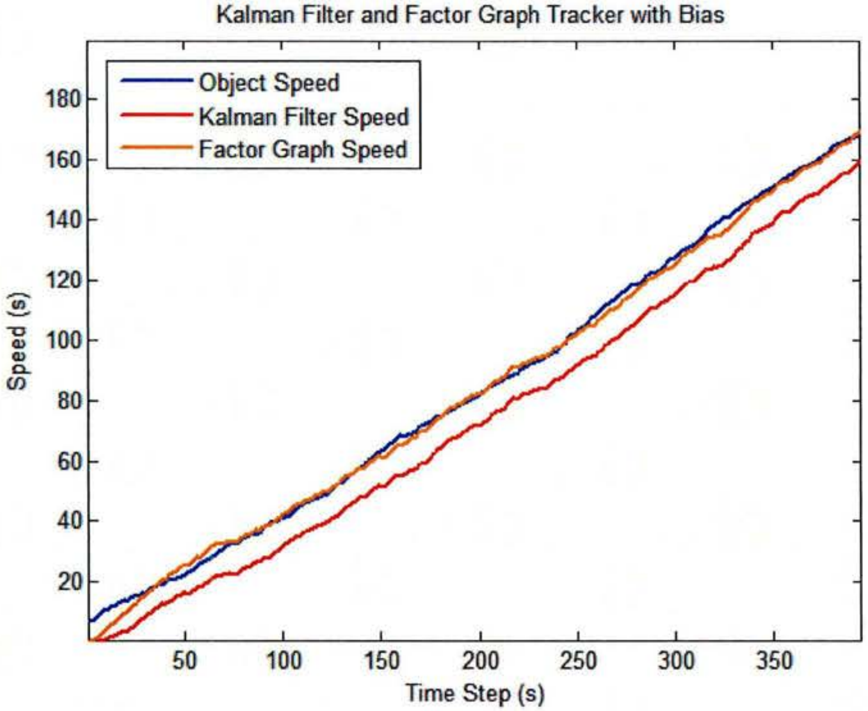


Figure 29: Plot of the Kalman filter and factor graph tracker tracking the object with a bias of 0.4 m/s in the x-direction.

Table 1: Location and velocity RMS errors for the Kalman filter and factor graph tracker when tracking an object with a bias of 0.4 m/s in the x-direction of the velocity

Root Mean Squared Errors		
	Location RMS Error (m)	**Velocity RMS Error (m/s)**
Kalman Filter	129.6882	10.4902
Factor Graph Tracker	36.6864	2.1496

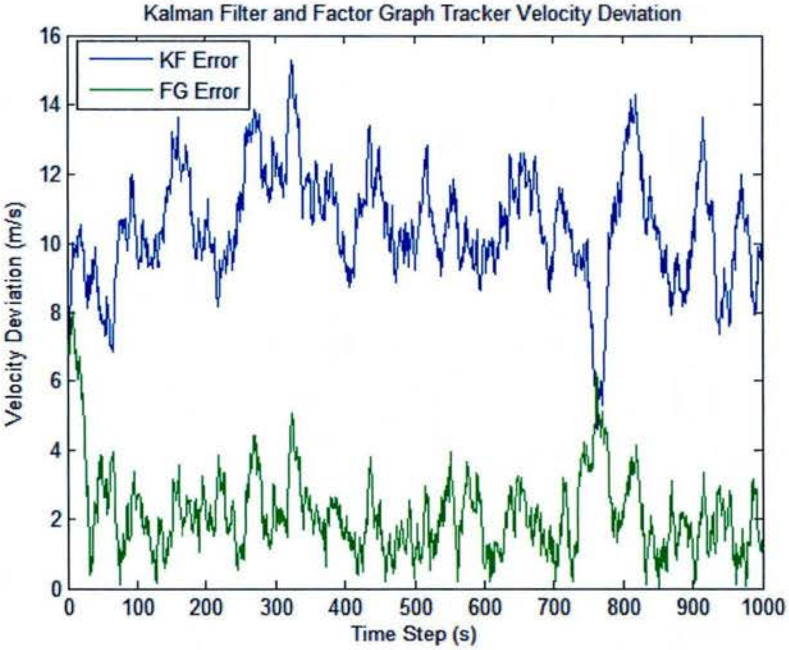


Figure 30: Plot of the difference of the velocity of the object being tracked from the velocities of both the tracks of the Kalman filter and factor graph tracker. The deviation of the Kalman filter is in blue and the deviation of the factor graph tracker is in green.

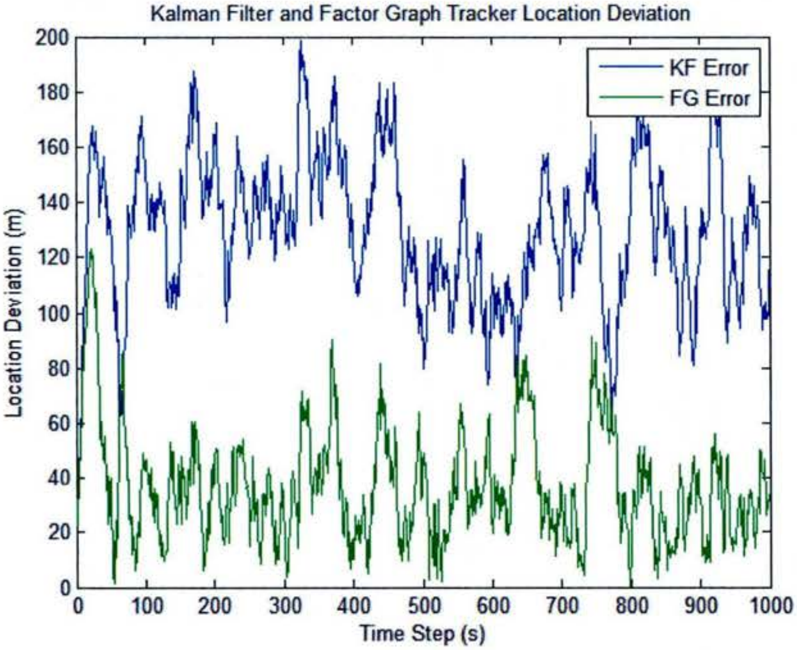


Figure 31 Plot of the difference of the location of the object being tracked from the locations of both the tracks of the Kalman filter and factor graph tracker. The deviation of the Kalman filter is in blue and the deviation of the factor graph tracker is in green.

Chapter 4: GRAPHICAL MODEL BASED CONTINUOUS ACTIVE SONAR

4.1 Particle Filter Simulation

In order to test the hypothesis that continuous active SONAR (CAS) will yield more accurate results when tracking an object verses pulsed active SONAR (PAS), a simulation of a particle filter in MATLAB was performed. The simulation would test different scenarios modeling CAS and PAS to track an object and would output the error of each. The error collected would be the total area of the particle cloud surrounding the target with its own standard deviation. The other two pieces of error collected would be the root mean square (RMS) error for the position of the target and the velocity of the target. Each of these statistics also had their own standard deviations.

4.1.1 Expected Results

The expected result should show that the error decreases with each decrease in the time step interval and should be lower when compared to the monostatic scenario. If this is the outcome, then a bistatic situation must be used to continue. Since the receiver would no longer be used as the transmitter, the next step would be to determine the appropriate messages concerning the transmitter that should be embedded within the signal or measured for a closer estimate of the object. Depending on the number of messages that could be embedded or used, the expected results should show less error when knowing the transmit time, thus being able to calculate the range from the receiver to the source of the signal. The error should also decrease when knowing the Doppler between each element, the transmitter, receiver, and object, and when knowing the velocity of the source.

4.1.2 Setup

4.1.2.1 Omnidirectional vs Forward-Directional

Receive-arrays can be either omni-directional or directional with each having a benefit over the other. If the direction of the echo is known, the directional receiver can be used and help to offer information concerning the bearing uncertainty toward the object (Siurna, Crawford, Theriault, & Armstrong). If the receiver is between the signal source and the echo from the object, and is oriented toward the object, masking by the receiver vehicle body may mitigate the interference from the direct blast of the transmitter. However, if the direction of the object is unknown, having a directional receiver could result in missing the echo if it is oriented in the wrong direction.

For these reasons, both an omni-directional receive-array and a forward-directional receive-array were tested in the particle filter at each scenario. The forward-directional receiver can receive data from anywhere within the hemisphere of the direction it is traveling; a total of 180 degrees. This will be referred to as the receiver "seeing" the item. Within the particle filter code, a specific bearing standard deviation was used when determining if the receiver could see the source and is given in Table 3 in the Basic Operation section.

4.1.2.2 Vehicle Geometries and Velocities

It was important to test the simulation using different spatial scenarios/configurations to observe if the results varied based on relative geometry, especially in the case of the forward-directional receiver. Three spatial configurations were tested to attempt to capture extremes of all possible scenarios. These geometries with each vehicle travel path are illustrated below. The transmitter is demonstrated by a green square, the receiver is a blue circle, and the target is given as a red asterisk. The arrows at the ends of each path indicate the direction the vehicle is traveling. The first two geometries use a bistatic configuration. Geometry 1 has the transmitter traveling behind the receiver, while the receiver is traveling toward the general area of the target and is shown in Figure 32. Figure 33 illustrates the second geometry, in which the source is now traveling so that it is in the view of the receiver. Geometry 3 is the monostatic case; meaning the transmission and reception of the transmission are performed by the same vehicle. This vehicle is traveling towards the target as presented in Figure 34.

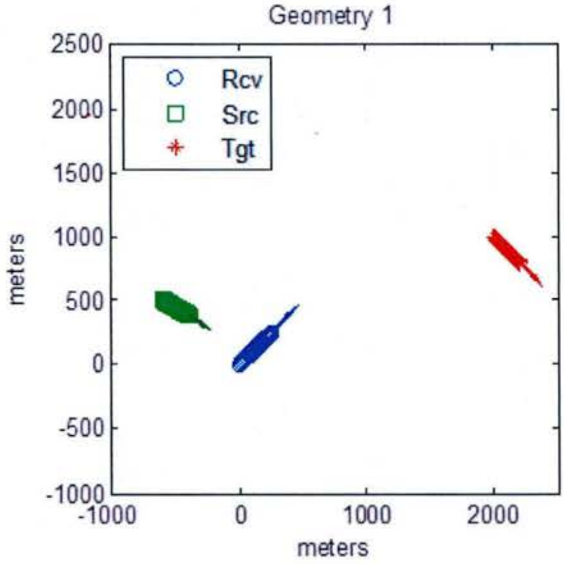


Figure 32: Geometry 1: The source's path of travel remains behind the path of the receiver while the target remains in view of the receiver's path.

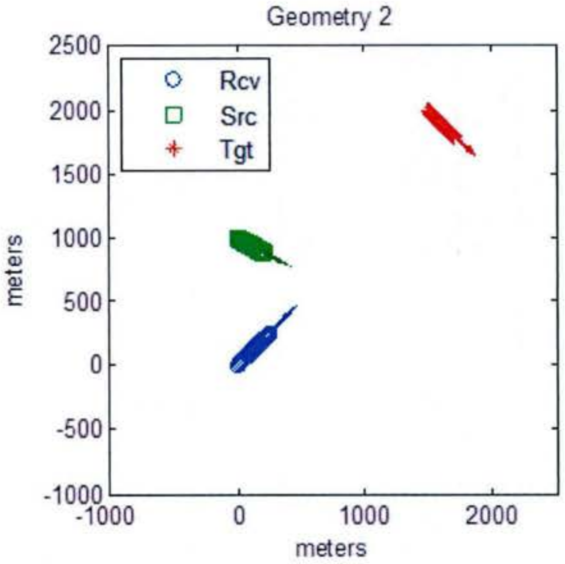


Figure 33: Geometry 2: The source's travel path remains in view of the receiver's path and is traveling between the receiver and target.

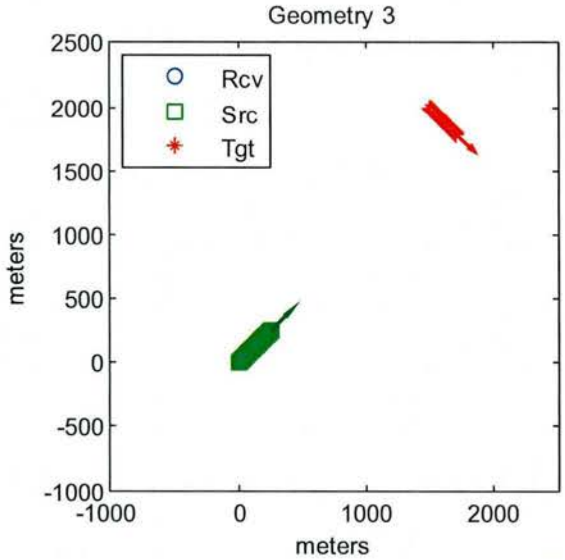


Figure 34: Geometry 3: The monostatic case. The source and receiver is one vehicle traveling toward the target.

At this time, the velocities of the vehicles were not of interest to be tested and therefore remained fixed. Table 1 describes the velocities for each vehicle.

Table 2: The velocities of the vehicles used in the particle filter simulation

Vehicle Velocities

Vehicle	**Speed m/s**	**Angle from Plot Origin (degrees)**
Bistatic Cases		
Receiver	3	45
Source	2	-30
Target	2.5	-45
Monostatic Case		
Receiver/Source	3	45
Target	2.5	-45

4.1.2.3 Transmit Rate

The transmit rate was varied to test if the ping repetition rate impacted the accuracy in tracking the object. For each scenario four different time steps were tried: one at three seconds, another at two seconds, one at one second, and the last one at half a second. Geometry 3 is the one exception to these trials. Since geometry 3 is the monostatic case, the transmitter has to wait to receive the echo before it can transmit again, as opposed to a bistatic case, where the source can continuously transmit without any risk of losing data. For this reason, only the three second time-step will be tested on the monostatic case.

4.1.2.4 Known Messages

In order to localize the object in relation to the receiver, several pieces of information are needed. First, the location of the transmitter in relation to the receiver must be known, and second, the time the signal was transmitted must also be known. Finally, the angle of arrival of the echo is needed. This information might be easy to obtain if the transmitter and receiver are stationary or if radio communications and GPS are readily available. Unfortunately, vehicles are typically in motion, and when working with undersea vehicles, GPS and radio frequencies are typically not accessible or available.

A few resources are available that can be used to infer the relative geometry between the transmitter, receiver, and object. Such as using the direct blast from the transmitter, taking bearing measurements, or taking advantage of acoustic communications in which, a message or messages can be embedded within the transmitted signal to be intercepted by the receiver.

Several message possibilities were tested, to determine which would be the most useful towards more precisely tracking an object. A message can either be an acoustic communication message embedded in the signal or it can also be a measurement provided by a SONAR sensing mechanism. Among the messages being tested was the time of transmit, thus allowing for the computation of the range between the source and receiver. Another possible message could consist of either sending the velocity of the source or just simply the speed of travel of the source. A message that can be measured is the complete bistatic Doppler from the source to the target to the receiver, and from the source to the receiver. A different standard deviation was used within the MATLAB code for each specific piece of information, which is given in Table 4. When the velocity of the source was known, two standard deviations were applied: speed and a separate bearing for velocity.

Because there is limited bandwidth for communication, it is unknown how many messages can be embedded within the SONAR signal or sent by a side channel. An example of an acoustic transceiver that could be used for underwater acoustic communication is the WHOI Micro-Modem. It can operate at 10 kHz, 15 kHz, or 25 kHz, the same as the REMUS transponders used on some UUVs, using 4 kHz of bandwidth. The WHOI Micro-Modem has the ability to use frequency-shift keying with frequency hopping which allows it to operate in shallow water with bistatic UUV configurations. After implementing error-correction, the WHOI Micro-Modem has a data rate of 80 bps. It is beyond the scope of this thesis to determine the size of the messages being sent, the encoding details, or how many messages can be sent in a given time interval. For this reason, different quantities of messages were ran and compared. (Freitag, et al., 17-23 Sept. 2005)

As it is possible for vehicles to lose clock synchronization, scenarios were also run assuming the transmit times are unknown. Finally, the type of signal being transmitted does not always allow for an accurate Doppler measurement to be taken, so different situations were run with Doppler being unknown to determine how beneficial each combination of known messages were. All possible combinations of the messages were assumed to be sent and measured in the program and all were assumed to be received. This resulted in a total of twelve combinations, which are given in Table 3. When executing the program, the source velocity message and source speed message were never both turned on at the same time since the velocity includes the speed.

Table 3: All possible message combinations tested in the particle filter simulation. The ones and zeros denote a message being used or not in that test, respectively. The source velocity and source speed were never run at the same time since velocity includes speed.

Message Combinations				
	Messages			
Combinations	**Transmit Time**	**Source Velocity**	**Source Speed**	**Doppler**
Zero Messages				
1	0	0	0	0
One Message				
2	1	0	0	0
3	0	1	0	0
4	0	0	1	0
5	0	0	0	1
Two Messages				
6	1	1	0	0
7	1	0	1	0
8	1	0	0	1
9	0	1	0	1
10	0	0	1	1
Three Messages				
11	1	1	0	1
12	1	0	1	1

4.1.2.5 Basic Operation

It is assumed that the bistatic time-difference of arrival, i.e., the difference between the echo time of arrival and direct blast, is always known for each acoustic processing cycle. Dividing by the speed of sound (assumed to be a constant 1500 meters per second) yields range between the source and target added to the range between the object and receiver and minus the range between the receiver and source. This is referred to as the *delta range* and is always known even though the individual component ranges may not be. When computing the delta range within the MATLAB particle filter code, the standard deviation for range, given below in Table 4, is applied. This range standard deviation is also used when computing the range for the source when given the transmit time. When the delta range is known, the problem of localizing the target is now narrowed down to an ellipse surrounding the source and receiver. The solution is located anywhere on the ellipse. Figure 35 illustrates this geometry where, as before, the blue circle is the receiver, the green square represents the source, and the red star is the target. The black lines represent the ranges between each connecting vehicle.

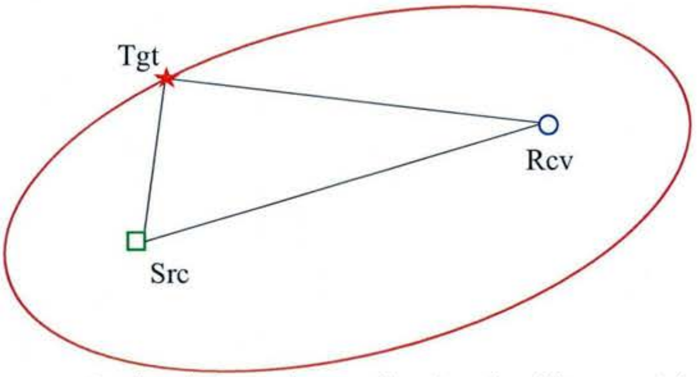


Figure 35: An ellipse representing the solution to the target location when delta range is known. The blue circle is the receiver, the green square is the source, and the red star is the target. The black lines represent the ranges between each connecting vehicle.

Another fact that is always known is the bearings of the source and target in relation to the receiver, for which a standard deviation for relative geometry bearing, also in Table 4, is applied. The one exception is if the receiver is forward-directional and cannot see the transmitter, as in geometry one.

Table 4: The standard deviations used in the particle filter simulation

Standard Deviations

Application	Sigma value
Range	40 m
Relative Geometry Bearing	2°
Doppler	1 m/s
Source Speed	1 m/s
Source Velocity Bearing	30°

The simulation gives a joint estimation of the source state and then the object state, which consist of their geometries and velocities. In order for this to be done, each particle in the simulation has its own target and source state. The state of the source is predicted from the information obtained by the receiver from the direct blast and the target echo. Each particle in the cloud surrounding the transmitter is then used to estimate the state of its corresponding particle component in the cloud surrounding the target. Last, the program cleans out any particles whose paired states show inconsistency with the measurements being observed. This simulation is done at the contact level and is not modeling actual time series. Figure 36 demonstrates a screen shot of the simulation while running. The yellow cloud is the particle cloud surrounding the transmitter and the pink cloud is the particle cloud surrounding the target.

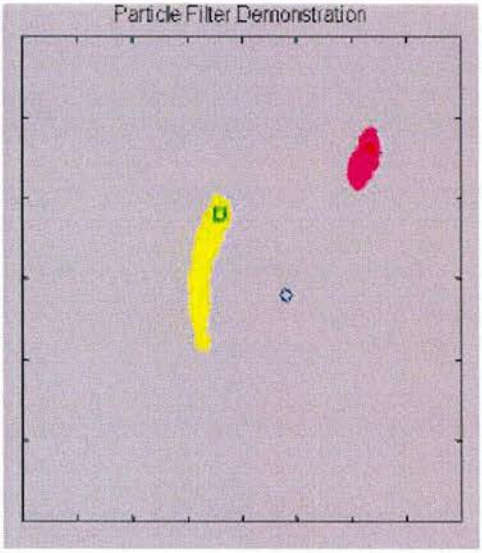


Figure 36: Screen shot during the particle filter simulation. The yellow particle cloud is surrounding the transmitter (green square), the pink cloud is surrounding the object (red asterisk), and the blue circle is the receiver.

The particle filter code uses 10,000 particles for each simulation and is performed using the computer specifications given in Table 5. It did not make use of any graphics card acceleration. The program completes ten runs at each of the mentioned scenarios allowing for the estimation of random error affecting the results. At the end of each run, the program calculates the area of the particle cloud surrounding the target, the RMS error for the target position, and the RMS error for the target velocity. For each of these statistics, a standard deviation over the ensemble was also computed.

Table 5: Computer specifications used for the particle filter simulation.

Computer Specifications	
Processor	Intel® Core™
Processor Number	i5-2400
Processor Base Frequency	3.1 GHz
Memory	4 GB

4.1.3 Statistics and Results

For each scenario, the area of the particle cloud ellipse surrounding the target was calculated using Equation 56.

$$A = \pi ab \tag{56}$$

In the equation above, a is the radius of the major axis of the surrounding ellipse, while b is the radius of the minor axis. The root mean squared (RMS) error (RMSE) for both the predicted target location and velocity were also found for each situation. Equation 57 gives the formula for calculating RMS error where $\hat{y}_i$ is the predicted value at time i for n observations, and y_i is the actual value at time i.

$$RMS\ Error = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{57}$$

Since the RMS error of the location and velocity need to be found, as opposed to an arbitrary value, the distances between the x and y RMS error values for both location and velocity were calculated.

Once the statistics for each scenario were found, the results for each specific situation where only the transmit intervals were changed, were analyzed for comparison. In every case, significantly less error was found for every decrease in the transmit interval. The following three figures demonstrate one case of the yielded target cloud area, and position and velocity RMS errors for an omni-directional receiver at geometry 1 with message configuration 11.

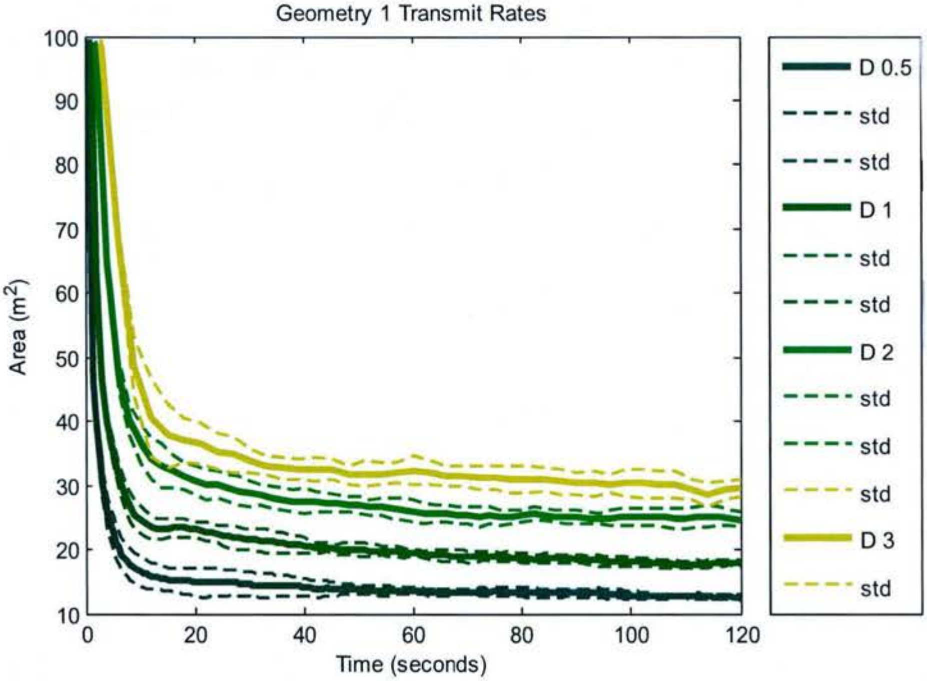


Figure 37: The area of the target particle cloud for each transmit rate for an omni-directional receiver at geometry 1 with message combination 11.

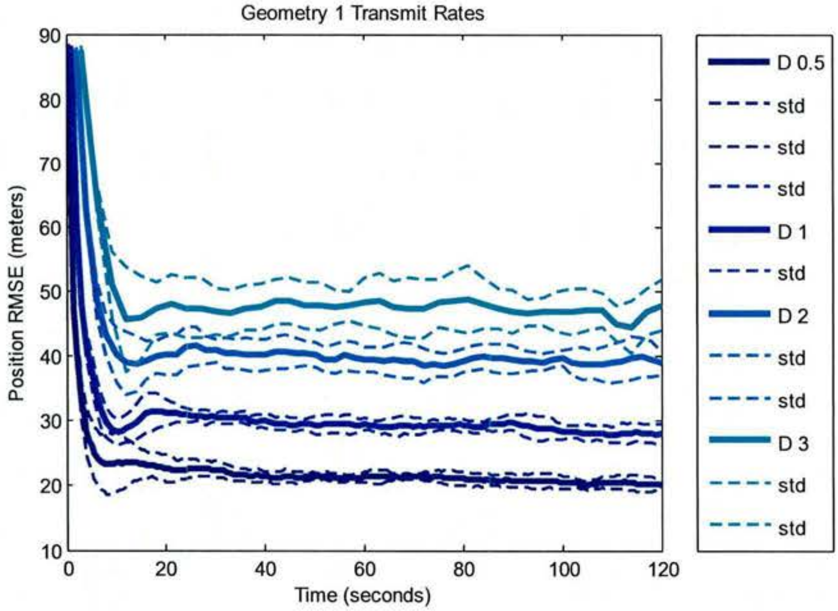


Figure 38: The RMS Error of the predicted target location for each transmit rate for an omni-directional receiver at geometry 1 with message combination 11.

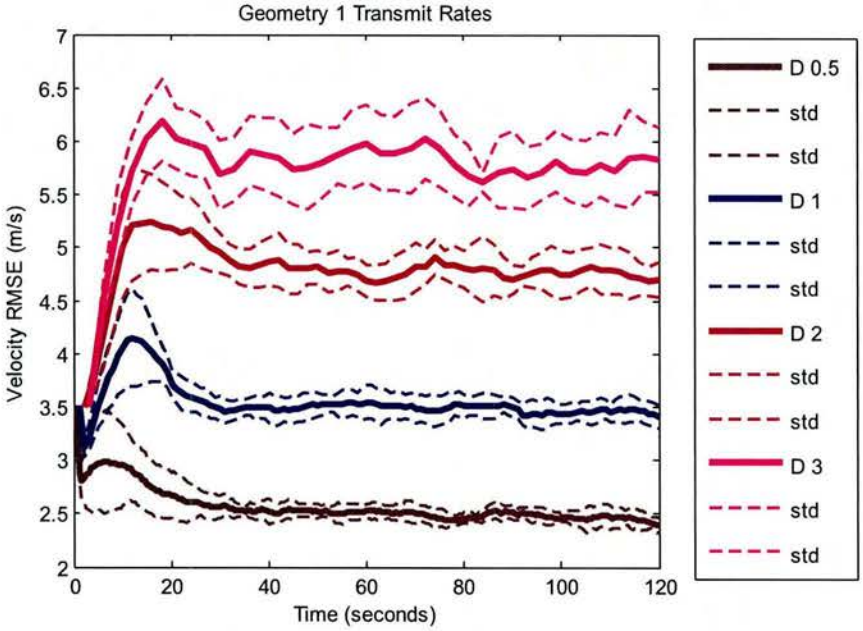


Figure 39: The RMS Error of the predicted target velocity for each transmit rate for an omni-directional receiver at geometry 1 with message combination 11.

For every scenario tried in the simulation, the same trend of results shown in Figure 37, Figure 38, and Figure 39 were imitated. Since the location and speed of the target is found with significantly less error as the update rate increases, it can be concluded that CAS will be more effective in target localization with minimum error than PAS, being that CAS has a much higher update rate. To further verify this result, two paired t-tests were run to compare the scenario with the least error for the monostatic case against bistatic cases. The monostatic case involved knowing Doppler, since it resulted in the least error. The two geometric configurations were tested against knowing zero messages for the same setup, then the means for the two t-tests were calculated and used to compare the monostatic case verses the bistatic case. In every comparison, the bistatic configuration yielded the least error across all three statistics.

Now that the effectiveness of CAS verses PAS has been proven to lead to more accurate results when locating the target, the messages and their combinations that will yield a more precise solution need to be found. Since it is unknown how many messages can be found or sent, the results will be calculated for the case of knowing zero messages, one message, two, and finally all three messages. To find these results, paired t-tests were utilized after combining both bistatic geometries together for each of the three statistics and for all message combinations with both directional receivers (forward-directional and omni-directional).

The paired t-tests were first completed comparing each single message against zero messages. Second, the single messages were compared against each other to determine which message provided a more accurate estimate of the target location. Next, each combination of two messages was compared to each other and finally, the two cases of knowing three messages were compared for each statistic over the two directional receivers. An example of a t-test plot result for a two message comparison is shown in Figure 40. Once the message combination that produced the most accurate solution was found for each limited message amount, paired t-tests were conducted to find the overall message combination for each statistic at both directional receiver types that resulted in the least error. And finally the overall message combinations yielding the least error at each directional receiver type were compared against each other for each statistic.

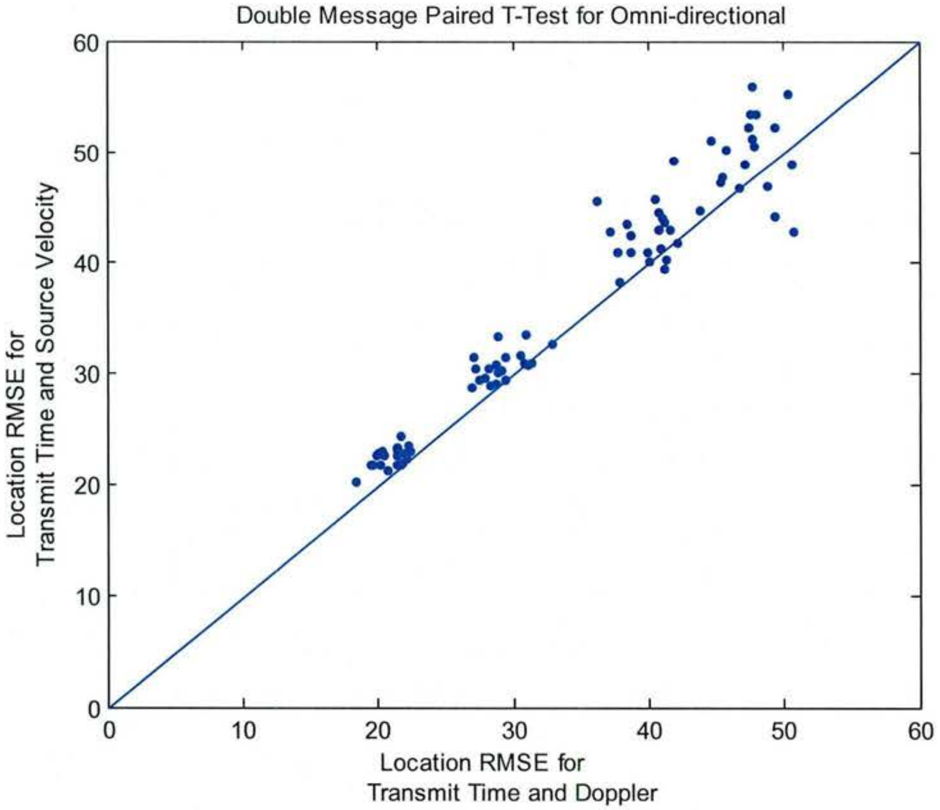


Figure 40: A paired t-test plot comparing the location RMS error for a known transmit time and source velocity message combination against a transmit time and Doppler message combination using an omni-directional receiver. The Transmit Time and Doppler combination yielded the least error.

When conducting the t-tests, the significance level, or alpha level, was chosen to be 0.05. The p-value is the probability of obtaining a test statistic equal to the observed value or more extreme when the null hypothesis H_0, is true. If the p-value is greater than the alpha level, then the null hypothesis is accepted. Otherwise it is rejected and the alternative hypothesis H_a, is assumed true. A left-tailed t-test was performed where the null hypothesis states that a combination x is greater than or equal to a combination y verses the alternative hypothesis that the combination x is less than the combination y. Since t-tests are designed to compare the means to zero, combination y is subtracted in the hypotheses to compare the expressions to zero. Once a combination is found to have less error than the one to which it is being compared, then that combination is then used for comparison to the other combinations when continuing. The hypotheses for each t-test and their resulting p-values are given in Table 6 along with the decision made for each test. For more information regarding t-tests, chapter 208 of the NCSS Statistical Software documentation clearly explains the operation and meaning of the test (NCSS Statistical Software, 2015).

Table 6: The paired t-test hypotheses, resulting p-values, and the decisions made.

Paired T-Tests						
Directional Receiver Type	Statistic	Number of Messages	$H_0 : x - y \geq 0$ (com x ≥ com y)	$H_a : x - y < 0$ (com x < com y)	P-value (α=0.05)	Accept/ Reject H_0
Forward	Location RMS Error	One	2 ≥ 3	2 < 3	0	Reject
			2 ≥ 4	2 < 4	0	Reject
			2 ≥ 5	2 < 5	0	Reject
		Two	6 ≥ 7	6 < 7	0	Reject
			6 ≥ 8	6 < 8	0	Reject
			6 ≥ 9	6 < 9	0.0138	Reject
			6 ≥ 10	6 < 10	0	Reject
		Three	11 ≥ 12	11 < 12	0	Reject
		Overall	2 ≥ 6	2 < 6	1	Accept
			6 ≥ 11	6 < 11	1	Accept
	Velocity RMS Error	One	2 ≥ 3	2 < 3	0.1572	Reject
			2 ≥ 4	2 < 4	0	Reject
			2 ≥ 5	2 < 5	0	Reject
		Two	6 ≥ 7	6 < 7	0.0008	Reject
			6 ≥ 8	6 < 8	1	Accept
			8 ≥ 9	8 < 9	1	Accept
			9 ≥ 10	9 < 10	0.0036	Reject
		Three	11 ≥ 12	11 < 12	0	Reject
		Overall	2 ≥ 9	2 < 9	1	Accept
			9 ≥ 11	9 < 11	0.9224	Accept
	Area	One	2 ≥ 3	2 < 3	0	Reject
			2 ≥ 4	2 < 4	0	Reject
			2 ≥ 5	2 < 5	0	Reject
		Two	6 ≥ 7	6 < 7	0	Reject
			6 ≥ 8	6 < 8	0	Reject
			6 ≥ 9	6 < 9	0.9946	Accept
			9 ≥ 10	9 < 10	0	Reject
		Three	11 ≥ 12	11 < 12	0	Reject
		Overall	2 ≥ 9	2 < 9	1	Accept
			9 ≥ 11	9 < 11	1	Accept

Table 6: The paired t-test hypotheses, resulting p-values, and the decisions made.

Paired T-Tests						
Directional Receiver Type	**Statistic**	**Number of Messages**	$H_0 : x - y \geq 0$ **(com x ≥ com y)**	$H_a : x - y < 0$ **(com x < com y)**	**P-value (α=0.05)**	**Accept/ Reject** H_0
Omni	Location RMS Error	One	$2 \geq 3$	$2 < 3$	0	Reject
			$2 \geq 4$	$2 < 4$	0	Reject
			$2 \geq 5$	$2 < 5$	0	Reject
		Two	$6 \geq 7$	$6 < 7$	0.0338	Reject
			$6 \geq 8$	$6 < 8$	1	Accept
			$8 \geq 9$	$8 < 9$	0	Reject
			$8 \geq 10$	$8 < 10$	0.0002	Reject
		Three	$11 \geq 12$	$11 < 12$	0.3025	Accept
		Overall	$2 \geq 8$	$2 < 8$	1	Accept
			$8 \geq 12$	$8 < 12$	0.1255	Accept
	Velocity RMS Error	One	$2 \geq 3$	$2 < 3$	0.0002	Reject
			$2 \geq 4$	$2 < 4$	0	Reject
			$2 \geq 5$	$2 < 5$	0	Reject
		Two	$6 \geq 7$	$6 < 7$	0.2714	Accept
			$7 \geq 8$	$7 < 8$	1	Accept
			$8 \geq 9$	$8 < 9$	0.592	Accept
			$9 \geq 10$	$9 < 10$	0.3224	Accept
		Three	$11 \geq 12$	$11 < 12$	0.0037	Reject
		Overall	$2 \geq 10$	$2 < 10$	1	Accept
			$10 \geq 11$	$10 < 11$	0.9993	Accept
	Area	One	$2 \geq 3$	$2 < 3$	0.0001	Reject
			$2 \geq 4$	$2 < 4$	0	Reject
			$2 \geq 5$	$2 < 5$	0	Reject
		Two	$6 \geq 7$	$6 < 7$	0.0049	Reject
			$6 \geq 8$	$6 < 8$	1	Accept
			$8 \geq 9$	$8 < 9$	0.0002	Reject
			$8 \geq 10$	$8 < 10$	0.0001	Reject
		Three	$11 \geq 12$	$11 < 12$	0.0119	Reject
		Overall	$2 \geq 8$	$2 < 8$	1	Accept
			$8 \geq 11$	$8 < 11$	1	Accept

As predicted, when comparing each message combination to zero messages known, every combination yielded less error over knowing zero messages. When only one message could be known, each statistic over every scenario unanimously resulted in the known transmit time yielding a more accurate solution. If all three messages could be known, knowing the velocity over the speed also resulted in less error across the board with the exception of one scenario. This exception is for the location RMS error when the receiver is omni-directional, in which case, it was more beneficial to use combination 12 (knowing the transmit time, the speed of the source, and the Doppler) Some varying results were found for cases limiting the number of messages to two and for statistical overall decisions.

The first set of cases to be discussed for knowing two messages and the overall message combination will be when the receiver is forward-directional. When analyzing the location RMS error, message combination 6 (knowing the transmit time and source velocity) resulted in the least error. However, combination 9 (knowing the source velocity and Doppler) had the least velocity RMS error and the smallest target particle cloud area. For all three statistics, combination 11 (knowing the transmit time, source velocity, and Doppler) yielded the least error overall, as was expected.

Next, the cases having an omni-directional receiver will be analyzed for the same two message limits. Contrary to the previous scenario with the forward-directional receiver, combination 10 (knowing the speed of the source and the Doppler) resulted in a lower velocity RMS error. Though, the smallest particle cloud area and lowest location RMS error were achieved by combination 8 (knowing the transmit time and Doppler). Overall for the omni-directional receiver, combination 11 had the smallest area and velocity RMS error but combination 12 had the lowest location RMS error. The summary of the paired t-test results are shown in Table 7 below.

Table 7: The paired t-test results for each statistic and directional receiver type show the message combinations that resulted in the least error.

Paired T-Test Results							
Directional Receiver	**Statistic**	**Number of Messages**	**Message Combination with Least Error**				
			Comb. #	**Transmit Time**	**Source Velocity**	**Source Speed**	**Doppler**
Forward	Location RMS Error	One	2	X			
		Two	6	X	X		
		Three	11	X	X		X
		Overall	**11**	X	X		X
	Velocity RMS Error	One	2	X			
		Two	9		X		X
		Three	11	X	X		X
		Overall	**9**		X		X
	Area	One	2	X			
		Two	9		X		X
		Three	11	X	X		X
		Overall	**11**	X	X		X
Omni	Location RMS Error	One	2	X			
		Two	8	X			X
		Three	12	X		X	X
		Overall	**12**	X		X	X
	Velocity RMS Error	One	2	X			
		Two	10			X	X
		Three	11	X	X		X
		Overall	**11**	X	X		X
	Area	One	2	X			
		Two	8	X			X
		Three	11	X	X		X
		Overall	**11**	X	X		X

Last, the population means of the forward-directional and omni-directional receivers were compared against each other for each of their overall least error statistics. For each statistic, the omni-directional receiver resulted in the most accurate solution.

Chapter 5: RESULTS, CONCLUSIONS, AND THE FUTURE

5.1 Results and Conclusions

5.1.1 Factor Graphing for DCL

The first goal of this thesis was to implement a factor graph for each component of the detection, classification, and localization (DCL) chain. This was achieved and explained for each section.

A factor graph was found for the detector using the model equation for a received signal. The Neyman-Pearson hypothesis test is then applied to the signal in the next stage of the factor graph leading to an output of identifying the data as either a signal detection or noise. If the received data is found to be a detection of the transmitted signal, it is sent on to the classifier for the next stage in the DCL chain.

The naïve Bayes classifier was easily transformed into a factor graph by simply identifying the equations and placements of the factor nodes. The class, y, requires an *a priori* probability, $p(y)$, which is given by the factor node $\psi(y)$. For each feature, $\mathbf{x}= (x_1, x_2, \ldots, x_K)$, connected to the class, y, a conditional probability, $p(\boldsymbol{x}|y)$, is required and is given by the edge factor node $\psi(y, x_k)$. When finding the closed-form solution of the factor graph through belief propagation, the resulting equation is equivalent to the joint probability formula for the naïve Bayes classifier.

Once the class is determined, the tracker proceeds to track the object. Implementing a factor graph into a tracker proved to be slightly more difficult. Two Gaussian state-space equations with assumed linear propagation are used, describing the predicted state at a time k of the object being tracked and the actual measurement taken of the object at time k. These state equations were implemented into a factor graph and through belief propagation, closed-form solutions were determined. The solutions showed that the factor graph tracker reduces to the Kalman filter if the noises are zero-mean. However, unlike the standard Kalman filter, the factor graph has the capability of inputting biased data to track the object.

5.1.2 Graphical Model Based Continuous Active Sonar

The second goal of this thesis was to evaluate the performance of CAS to determine if the accuracy of the target location, determined by the tracker, improved as the update rate was increased. This was done through a particle filter simulation consisting of two bistatic configurations and a monostatic configuration. The monostatic configuration was limited to a transmit interval of three seconds since a monostatic system requires the unmanned underwater vehicle (UUV) to listen for the echo between transmissions. The bistatic configurations were tested with transmission intervals of three seconds, two seconds, one second, and half of a second. Also, because every receiver may not always be omni-directional, a forward-directional receive array was tested as well to compare the results. For each scenario, the area of the target particle cloud and the root mean square (RMS) errors for the location and velocity of the target were calculated. In every case, significantly less error was found for every decrease in the transmit interval. Since the location and speed of the target is found with significantly less error as the update rate increases, it was concluded that CAS will be more effective in localizing a target with minimum error than PAS, being that CAS has a much higher update rate.

As CAS was proven to be more beneficial in target tracking for the various scenarios studied, the next objective was to determine which messages sent between the two UUVs (receiver and transmitter) were most beneficial for tracking an underwater object. A message can either be an acoustic communication message embedded in the signal or it can also be a measurement provided by a SONAR sensing mechanism. The messages tested were: the time of transmission, either the velocity of the source or just simply the speed of travel of the source, and the complete bistatic Doppler from the source to the object to the receiver, and from the source to the receiver. Because it may not be possible or beneficial for all messages to be received, all possible combinations of knowing one message, two messages, or all three messages were tested, using a particle filter simulation, for each configuration, transmit interval, and receiver type scenario. To identify the message combination with the least error, paired t-tests were conducted comparing all scenarios for the specific number of messages.

For every scenario, if only one message could be known, knowing the transmit time resulted in the least error. If two messages could be known, the results varied for different scenarios. If the receiver was forward-directional, knowing the transmit time and the source velocity resulted in the least location RMS error, but knowing the source velocity and Doppler had the least velocity RMS error and smallest target particle cloud area. If the receiver was omni-directional, knowing the transmit time and Doppler gave the least location RMS error and the smallest particle cloud area. The least velocity RMS error was found by knowing the source speed and Doppler. If all three message categories could be known, it was best to know the transmit time, source velocity (as opposed to just its speed), and Doppler, for all but one scenario. That is, when the receiver is omni-directional, knowing the transmit time, source speed, and Doppler gave the least location RMS error.

The different message combinations resulting in least error for each number of messages allowed were also tested against each other via paired t-tests. These overall message combinations resulting in the least error were found to be the combinations where all three messages could be known with the exception of one scenario. Knowing the source velocity and Doppler when the receiver is forward-directional resulted in the least overall velocity RMS error. Of the other overall message combinations, it was more beneficial to know the transmit time, source velocity, and Doppler. This also has one exception. When the receiver is omni-directional, knowing the source speed as opposed to the velocity gave the least location RMS error.

Last, for each statistic, the omni-directional receiver resulted in the most overall accurate solution. With the exception of a couple surprises, most of the results found were as predicted.

5.2 Significance of Results

Implementing factor graphs into the sections of the DCL chain allows for the potential opportunity to feedback information obtained from the results of one section into a particular node of another section. However, the application of a tracker into a factor graph yielded the most significant result of this section of the thesis. The factor graph tracker was proven to be exactly as efficient as the standard Kalman filter, which is commonly used in tracking, but with the added bonus of the noise not requiring a zero-mean, allowing for the tracker to be able to handle the input of biased data.

In the second section of the thesis, since it was proven that as the transmit interval decreased, the tracking error also decreased, the effectiveness of CAS versus PAS was also proven. This conclusion resulted in further research to determine the value of different messages that could be known when tracking an object. The utility of each of these message combinations given different scenarios were found and compared. Knowing which message combination to use for each scenario, could be found to be very useful in naval applications involving the tracking of different objects.

5.3 Future Work

One possible extension of the present work would be to use the results from the factor graph tracker to determine information regarding the object being tracked, for instance length of the object, and feed that information back into a classifier. The classifier in question may not necessarily be a factor graph but the goal of supplying feedback could still be achieved

Another extension would be to run tests with stronger bias in the input data within the factor graph tracker. The availability of this closed-form bias expression may allow easier construction of practical trackers.

Another line of investigation would be to run more detailed simulations to study the implementation of actual signals such as M-sequences, or combinations of FM and CW waveforms, with respect to direct blast cancellation and localization ability. Another aspect of this simulation would be to expand the number of geometries to address more application specific configurations. Finally, it is necessary to determine the specifics regarding message encoding for the transfer of information between the receiver and source, for example, to learn how many bits each type of message would require, and therefore, how many messages can be sent.

REFERENCES

Coughlan, J. (2009). A Tutorial Introduction to Belief Propagation. *The Smith-Kettlewell Eye Research Institute.*

Department of the Navy. (2004). *The Navy Unmanned Undersea Vehicle (UUV) Master Plan.*

Dobbins, F. (2014, April). *Military Robotics*. Retrieved from Presentation of Robotics: https://sites.google.com/site/presentationofrobotics/home

Doucet, A., & Johansen, A. M. (2008, December). A Tutorial on Particle filtering and Smoothing: Fifteen years later. Tokyo, Japan.

Duda, R., Hart, P., & Stork, D. (2001). Introduction. In *Pattern Classification Second Edition* (pp. 1-19). New York; Chichester; Weinheim; Brisbane; Singapore; Toronto: John Wiley & Sons, Inc.

Faragher, R. (2012). Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation. *Signal Processing Magazine-lecture NOTES*, 128-132.

Forney, J. G. (2005). Codes on Graphs. In E. Biglieri, *Coding for Wireless Channels* (pp. 235-271). Springer US.

Frank R. Kschischang, S. M. (2001). Factor Graphs and the Sum-Product Algorithm. *IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. 47, NO. 2*, 498-519.

Freitag, L., Grund, M., Singh, S., Partan, J., Koski, P., & Ball, K. (17-23 Sept. 2005). The WHOI micro-modem: an acoustic communications and navigation system for multiple platforms. *OCEANS, 2005. Proceedings of MTS/IEEE* (pp. 1086 - 1092 Vol. 2). Washington, DC: IEEE.

Goldhahn, R., Braca, P., LePage, K. D., Willett, P., Marano, S., & Matta, V. (4-9 May 2014). Environmentally Sensitive Particle Filter Tracking in Multistatic AUV Networks with Port-Starboard Ambiguity. *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference* (pp. 1458-1462). Florence: IEEE.

Gordon, N. J., Salmond, D. J., & Smith, A. F. (1993). Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimation. *IEEE Proceedings-F, Vol. 140, No. 2*, 107-113.

Grimmett, D., & Wakayama, C. (2013). Multistatic Tracking for Continuous Active Sonar using Doppler-Bearing Measurements. *16th International Conference on Information Fusion* (pp. 258-265). Istanbul, Turkey: ISIF.

Heeger, D. (2003-2007). Signal Detection Theory. New York, United States.

Hickman, G., & Krolik, J. L. (2012). Non-Recurrent Wideband Continuous Active Sonar. Durham, NC, USA.

Jonathan S. Yedidia, W. T. (2001). Understanding Belief Propagation and its Generalizations. *Mitsubishi Electric Research Laboratories.*

Kalman, R. E. (March 1960). A New Approach to Linear Filtering and Prediction Problems. *Transaction of the ASME-Journal of Basic Engineering*, 35-45.

Loeliger, H.-A. (Jan, 2004). An Introduction to Factor Graphs. *IEEE Signal Processing Magazine*, 28-41.

Loeliger, H.-A., Dauwels, J., Hu, J., Korl, S., Ping, L., & Kschischang, F. R. (2007). The Factor Graph Approach to Model-Based Signal Processing. *Proceedings of the IEEE Vol. 95, No. 6*, 1295-1322.

Macmillan, N. A. (2002). Signal Detection Theory. In H. Pashler, & J. Wixted, *Stevens' Handbook of Experimental Psychology Third Edition Volume 4* (pp. 43-90). New York: John Wiley & Sons Inc.

Maybeck, P. (1979). Introduction. In P. Maybeck, *Stochastic Models, Estimation, and Control, Vol. 1* (pp. 1-16). Academic Press.

NCSS Statistical Software. (2015). Chapter 208: Paired T-Test. In *NCSS Documentation* (pp. 208-1 - 208-25). NCSS, LLC.

Palisade Corporation. (2015). *Monte Carlo Simulation*. Retrieved from Palisade: http://www.palisade.com/risk/monte_carlo_simulation.asp

Ricker, D. W. (2003). *Echo Signal Processing.* Boston: Kluwer Academic Publishers.

Ristic, B., Arulampalam, S., & Gordon, N. (2004). *Beyond the Kalman Filter Particle Filters for Tracking Applications.* Boston; London: Artech House Publishers.

Sacha, J., & Shaffer, A. (2010, September 15). Particle Filter Speed Up Using a GPU. *14th High Performance Embedded Computing Workshop*, MIT Lincoln Laboratory: Lexington, MA.

Singer, P. W. (2009). Military Robots and the Laws of War. *The New Atlantis, Number 23, Winter*, 25-45.

Siurna, L., Crawford, J., Theriault, J. A., & Armstrong, B. (n.d.). In-stride Compact Active-Passive Sonar (iCAPS): Re-integrating a Canadian Technology for Littoral Applications. Dartmouth, Nova Scotia, Canada.

Sorenson, H. W. (1970). Least-squares Estimation: From Gauss to Kalman. *Spectrum*, 63-68.

Sutton, C., & McCallum, A. (n.d.). An Introduction to Conditional Random Fields for Relational Learning. Massachusetts, USA.

Van Trees, H. L. (2001). *Detection, Estimation, and Modulation Theory.* Canada: John Wiley & Sons, INC.

Weiss, Y., & Freeman, W. (Oct., 2001). Correctness of Belief Propagation in Gaussian Graphical Models of Arbitrary Topology. *Neural Computation Vol. 13, No. 10*, 2173-2200.

Welch, G., & Bishop, G. (2006). *An Introduction to the Kalman Filter.* Chapel Hill, NC University of North Carolina at Chapel Hill.

Yakubovskiy, A. (2009). Bistatic Sonar, explained. *FarSounder, Inc.*, Warwick, RI, USA.

Zarrin, S., & Lim, T. J. (2008). Belief Propagation on Factor Graphs for Cooperative Spectrum Sensing in Cognitive Radio. *New Frontiers in Dynamic Spectrum Access Networks. 3rd IEEE Symposium*, 1-9.

Zhang, H. (2004). *The Optimality of Naive Bayes.* Fredericton, New Brunswick, Canada: American Association for Artificial Intelligence.

APPENDICES

Appendix A: Mean Derivation

The derivation of the mean of state x at time step k will be shown in this appendix and refers to Figure 1 below.

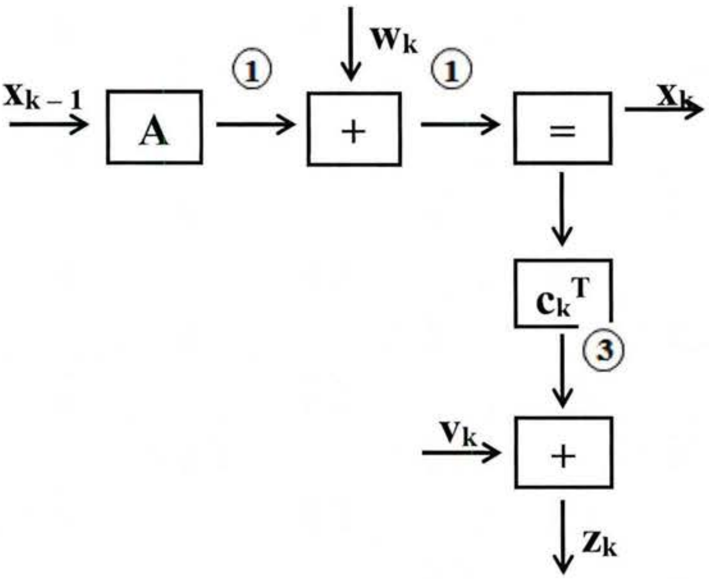


Figure 1: Factor graph of the Kalman filter with certain edges numbered and circled for derivation.

First, the mean at the circled number 1 is found and shown below.

$$\vec{m}_1 = A\vec{m}_{x_{k-1}} \tag{1}$$

The mean at the circled number 2 is found by taking the sum of Equation 1 and the mean of the process noise, and is shown in Equation 2.

$$\vec{m}_2 = \vec{m}_1 + \vec{m}_{w_k} = A\vec{m}_{x_{k-1}} + \vec{m}_{w_k} \tag{2}$$

The next step is to calculate the mean of the lower branch beginning at the leaves. This is given by the equation below.

$$\overleftarrow{m}_3 = \overleftarrow{m}_{z_k} - \vec{m}_{v_k}$$

Lastly, these equations are combined to derive the mean for the state *x* at time step *k*, given below in Equation 3.

$$\vec{m}_{x_k} = \vec{m}_2 + \vec{V}_2(c_k^T)^H G(\overleftarrow{m}_3 - c_k^T\vec{m}_2) \tag{3}$$

After substituting the known equations into Equation 3, the derived mean of the state *x* at time step *k* yields the equation shown below.

$$\begin{aligned}\vec{m}_{x_k} = A\vec{m}_{x_{k-1}} + \vec{m}_{w_k} + \left(A\vec{V}_{x_{k-1}}A^H + \vec{V}_{w_k}\right) \times (c_k^T)^H \\ \times \left[\overleftarrow{V}_{z_k} + \vec{V}_{v_k} + c_k^T\left(A\vec{V}_{x_{k-1}}A^H + \vec{V}_{w_k}\right)(c_k^T)^H\right]^{-1} \\ \times \left[\overleftarrow{m}_{z_k} - \vec{m}_{v_k} - c_k^T\left(A\vec{m}_{x_{k-1}} + \vec{m}_{w_k}\right)\right]\end{aligned}$$

To continue further and find the mean for the measurement in the direction of flow $\vec{m}_{z_k}$, the mean at the circled number 3 needs to be computed for the given arrow direction. This yields the following equation:

$$\vec{m}_3 = c_k^T\vec{m}_{x_k} \tag{4}$$

At this point, $\vec{m}_{z_k}$ can now be found by summing Equation 4 with the mean of the measurement noise and is given in the equation below.

$$\vec{m}_{z_k} = \vec{m}_{v_k} + c_k^T\vec{m}_{x_k}$$

Appendix B: Mean and *A Posteriori* State Estimate Correlation

This appendix will illustrate how the mean of the state *x* at time step *k*, $\vec{m}_{x_k}$, from the factor graph tracker, correlates to the *a posteriori* state estimate $\hat{x}_k$, in the Kalman filter. As reference, Equation 31 is placed below for the comparison.

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$

The relationship between the Kalman gain and the factor graph model is given by Equation 53 in section 3.3.1 of this paper. The measurement z_k corresponds to the mean of the measurement $\overleftarrow{m}_{z_k}$ in the factor graph. Also, *H* has already been related to the matrix *c* in the previously mentioned section.

The last piece to align with the mean from the factor graph, is the *a priori* state estimate $\hat{x}_k^-$. This Equation 41 relates to the mean $\vec{m}_2$, found in Appendix A, and below, both equations are given.

$$\hat{x}_k^- = A\hat{x}_{k-1}$$

$$\vec{m}_2 = A\vec{m}_{x_{k-1}} + \vec{m}_{w_k}$$

As explained in section 3.3.1, *A* is the same *n* x *n* matrix in both models. Since the objective is to prove that the *a posteriori* state estimate at time *k*, $\hat{x}_k$ from the Kalman filter relates to the mean of the state *x* at time step *k*, $\vec{m}_{x_k}$ from the factor graph tracker, then it must be assumed that the *a posteriori* state estimate at time step *k – 1*, $\hat{x}_{k-1}$, also relates to the mean of the state at the same time step, $\vec{m}_{x_{k-1}}$. However, the mean has an extra expression, $\vec{m}_{w_k}$, the mean of the process noise. Since the Kalman filter assumes that noise has a zero-mean, this expression can be eliminated, leaving the two equations equivalent.

$$\hat{x}_k^- = \vec{m}_2$$

$$A\hat{x}_{k-1} = A\vec{m}_{x_{k-1}}$$

For the same reason, the mean of the measurement noise $\vec{m}_{v_k}$, is also eliminated from the equation for $\vec{m}_{x_k}$.

Since all the essential parts of both the Kalman filter equation and the factor graph mean equation have been correlated when the means of the process and measurement noise are zero, it is necessary to say that the mean of the state *x* at time step *k*, $\vec{m}_{x_k}$, from the factor graph, correlates to the *a posteriori* state estimate $\hat{x}_k$, in the Kalman filter resulting in the following equations:

$$\hat{x}_k = \vec{m}_{x_k}$$

$$\hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) = \vec{m}_2 + \vec{V}_2(c_k^T)^H G\big(\overleftarrow{m}_{z_k} - c_k^T\vec{m}_2\big)$$

ACKNOWLEDGEMENTS

Many thanks are owed to the Naval Undersea Research Program (NURP), Maria Medeiros, Dr. Teresa McMullen, and the Office of Naval Research Code 33, for providing this research opportunity.

This research project and effort undertaken is sponsored by the Department of the Navy, Office of Naval Research. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the Office of Naval Research.