UNIVERSITY OF OKLAHOMA
GRADUATE COLLEGE

SCALABLE MULTI-AGENT COLLABORATION FOR RADAR TASKS

A DISSERTATION
SUBMITTED TO THE GRADUATE FACULTY
in partial fulfillment of the requirements for the
Degree of
DOCTOR OF PHILOSOPHY

By
GEOFFREY M. DOLINGER
Norman, Oklahoma
2025

SCALABLE MULTI-AGENT COLLABORATION FOR RADAR TASKS

A DISSERTATION APPROVED FOR THE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

BY THE COMMITTEE CONSISTING OF

Dr. Justin Metcalf, Chair

Dr. Dean Hougen

Dr. Nathan Goodman

Dr. Dimitris Diochnos

Dr. David Ebert

To Elyssa, Jace, Victoria, and Warren

Acknowledgments

First and foremost, I want to thank my wonderful wife, Elyssa. She is my best friend and her loving support, encouragement, resilience, and patience during this journey kept me optimistic and focused during the roughest times in my graduate studies. I want to thank my children Jace, Victoria, and Warren for believing in me and bringing me so much joy. I want to thank my advisor, Dr. Metcalf, for his guidance and mentorship throughout this journey. Thank you for inspiring me to explore these topics and calming my concerns. Your encouragement and support for your students has given me a passion I hope follows me through my career. I would like to thank my committee, Dr. Hougen, Dr. Goodman, Dr. Diochnos, and Dr. Ebert, for providing me with guidance and their wisdom. I would like to thank my leadership at USAF AFSC/SW, the 76 SWEG, and the AFOSR for funding my research. A huge thank you to Dr. Alex Stringer, my research partner and collaborator that brought understanding and joy to the research process. I can't imagine completing this degree without your friendship. Thank you to Shane and Bryan for your help and collaboration. I want to extend a huge thank you to the amazing team of researchers in the AFSC/SW and 76 SWEG office of research. Especially, Joe for his constant support and willingness to help with any task and Timmy for his excitement, amazing mind, and friendship throughout this research. I would like to thank Adam for his friendship and vision for the research effort and the culture he has helped build. Finally, I want to thank my mother, Debra Dolinger, for your love of this nerd. Your stability and support has given me a foundation to thrive.

Table of Contents

List of Tables

List of Figures

Abstract

We examine the challenge of multi-agent command and control (C2) scenarios where autonomous agents accomplish objectives by observing their environment, deciding optimal actions to meet the objective, and then act upon their environment. The most common method for these agents to observe their environment is through the use of radar. The domain of cognitive radar follows a similar perception-action cycle and accomplishes radar resource management (RRM) tasks. Modern C2 challenges with multiple autonomous agents using radar presents a unique challenge to both domains. Multi-agent autonomy is often formulated without the uncertainty and complexity of higher-fidelity radar and radar task methods often optimize singular systems as opposed to coalitions of agents. The integration of radar task methods with multi-agent systems presents a unique challenge that requires maximization of collaborative utility instead of singular performance. This work examines C2 challenges where agents have increased uncertainty and reduced resources but can and should coordinate actions. We designed a simulation environment for C2 challenges with medium fidelity radar detection simulations. Within this environment, we present agent action methods in the fields of machine learning and reinforcement learning to perform with low resources and increased uncertainty. Novel to this work is the use of game theory and meta-cognition to coordinate multiple agents for increase utility in the radar tasks of detection, track confirmation, and optimal surveillance. In order to meet the needs of the C2 domain, special attention is paid to methods that are scalable and modular.

Chapter 1

Introduction: Multi-Agent Command and Control

The early 1980s saw the expansion of the military concept of command with the inclusion of control. Modern technology has created a domain with more information and a need for agility alongside accuracy and assurance. McCann and Pigeau [1] summarize command and control (C2) as establishing a common intent to achieve coordinated action. More specifically, C2 systems look to collect data, identify intent by analyzing this data (reasoning), and use established intent to define actions (planning) [2]. Additionally, modern C2 often references the observe-orient-decide-act (OODA) model originally presented by Boyd (1996), an operational framework and guide to activities that encompass C2 [3]. Although not the only component of C2, radar systems play a major role in observing the environment, analyzing information, and taking relevant actions. Modern software-based radar systems utilize autonomy and radar signal processing methods to support C2. Command and control (C2) represents the mechanism for diverse operations to navigate an environment filled with uncertainty, complexity, and ever decreasing time to react. In all but the most trivial use cases, modern C2 operations utilize multiple platforms to achieve strategic objectives, making the addition of collaborative coordination paramount to the success of such operations.

For more than 100 years, autonomy and radar signal processing have improved military, industrial, and domestic fields largely through the use of closed-form ex-

pressions, heuristics, and probabilistic methods. Beginning with sensor management techniques, which utilized radar resource management (RRM) in the 1960s, advances in the 1990s focused on maximizing the performance of multiple radar functions such as detection, tracking, and classification. Advanced methods in constant false alarm rate (CFAR) detectors and adaptive clutter suppression further increased adaptivity in RRM [4]. These technological advances culminated in the term "cognitive radar" presented by Haykin in 2006. Cognitive radar is characterized by the perception-action cycle. The core characteristics of cognitive radar are a system that perceives an environment, has the ability to learn, and can adaptively determine the appropriate system response [5, 6].

The C2 domain has the goal of coordinating multiple assets to meet mission objectives in real time. This simple sounding goal is in reality complex and challenging. In order for command to make action decisions, the environment must be observed, processed, and summarized from complex technical information (radar return information and intel). This information is summarized and analyzed for intermediate mission impacts (tracks and entity information) and finally used to produce course of action (COA) recommendations for command. This process has to happen on a short timescale so that action decisions can have a relevant impact on the environment. Radar is an important subset of the tools used to address this challenge and plays a vital role in the success of C2 operations. Two factors have heavily influenced the difficulty of modern C2 operations: the growing scale of interconnected systems and the increased agility of actors. The introduction of unmanned aerial vehicles (UAV), or drones, has increased the number of actors in the environment, which requires dedicated effort and design to coordinate or enable autonomous action. Additionally, advances in computation speeds and the increased use of artificial intelligence (AI) have led to systems that operate at timescales orders of magnitude faster than human response capabilities. RRM is no stranger to these challenges and has a history of

research into cognitive radar advancements and optimizations to radar task execution. This makes RRM an ideal area of application and exploration for possible improvements to C2. The majority of RRM methods presented in the available research literature assume a single radar platform that can take multiple actions and complete multiple tasks. C2 environment often has multiple platforms with differing locations and capabilities that need to coordinate. This provides a useful space for exploring the formulation and execution of radar tasks with the goal of optimizing the collaborative performance of multiple systems.

The need for collaborative systems in C2 and RRM lends itself to the field of game theory (GT). GT is the study of mathematical and logical models to describe and optimize agent utility or achieve some defined success. A subset of this field is collaborative GT, which focuses on methods to optimize the utility of multiple agents collaborating instead of individual agents maximizing only their personal utility. This concept aligns well with the C2 mission and challenges. There is precedent for the use of GT in RRM and radar applications, including a focus on MIMO radar systems [7, 8, 9]. Similarly, GT has achieved success in the fields of adaptive jamming [10], trajectory optimization [11], and cognitive radar detection [12]. In each of these examples, GT is utilized to find optimal strategies. Within the realm of collaborative GT, the consensus method was explored by Deligiannis et al. [13], who optimized power allocation for beam forming, and Marden et al. [14], who approached exponential scaling of collaborative agents with a nearest neighbor approach.

The challenge of autonomous systems has a history in controls and cognitive radar. Applications of artificial intelligence and machine learning (AI/ML) were considered in radar in the early 1990's with the proposal of intelligent systems that learned from their environment using data to drive sensing and resource management [15]. Michael Wicks introduced "Sensor informed robotics" which focused on integrating the capability of sensors and sensor networks with platform actions [16]. This progression in

cognitive radar and autonomy has led to the current explosive growth of AI/ML and its application to the radar domain. The accomplishments in the field of AI/ML have resulted in its massive focus in research, development, and adoption of software in nearly every aspect of modern life. The impressive performance of AI/ML utilizing deep learning architectures, reinforcement learning (RL), and transformer-based generative methods make it a powerful tool. AI/ML techniques could provide solutions to problems beyond heuristic and probabilistic methods and provide coordination of complex systems to a scale unattainable by manual software implementation. However, these deep learning methods are largely black boxes that are difficult to understand and interpret [17]. After a model has trained on data, the values of the parameters are far from interpretable and can result in undesirable consequences like over-fitting, learned bias, hallucinations, lack of repeatability, and unpredictable behavior. A current challenge for software developers, AI engineers, and researchers is to explore the potential of AI/ML while addressing new challenges in explainability, trust, security, and certification of those capabilities. Specifically, RL has seen an increase in the rate of exploration as a tool for autonomy and cognitive radar. The use of RL and GT together provides the benefits of flexible learning from RL along with strategy optimization methods provided by GT. This has been studied extensively in the literature, particularly in the context of multi-agent RL [18, 19].

The domain of modern command and control presents a unique and under-explored challenge of accomplishing radar tasks with multiple autonomous platforms that need to collaborate. With the introduction of low-resource UAV swarms, the C2 challenge of coordinating these platforms is a scenario necessitating new methods for task competition and collaboration. This dissertation explores this intersection of radar task execution with multiple low-resource platforms and the new methods needed to autonomously coordinate and execute while remaining scalable to increases in platform count. To accomplish this exploration, I built a simulated environment with

medium fidelity radar and the C2 scenario then proposed and tested novel AI/ML, RL, and game theory methods with traditional RSP methods. (See Figure 1.1 for the breakdown of the domain and the methods explored) The primary contributions of this work include: a game theory method to collaborate RL and traditional agents to confirm highly evasive targets, a game theoretic utility representation to optimize collaborative search with moving platforms, and a scalable metacognitive detector that performs CFAR detection across multiple clutter profiles.

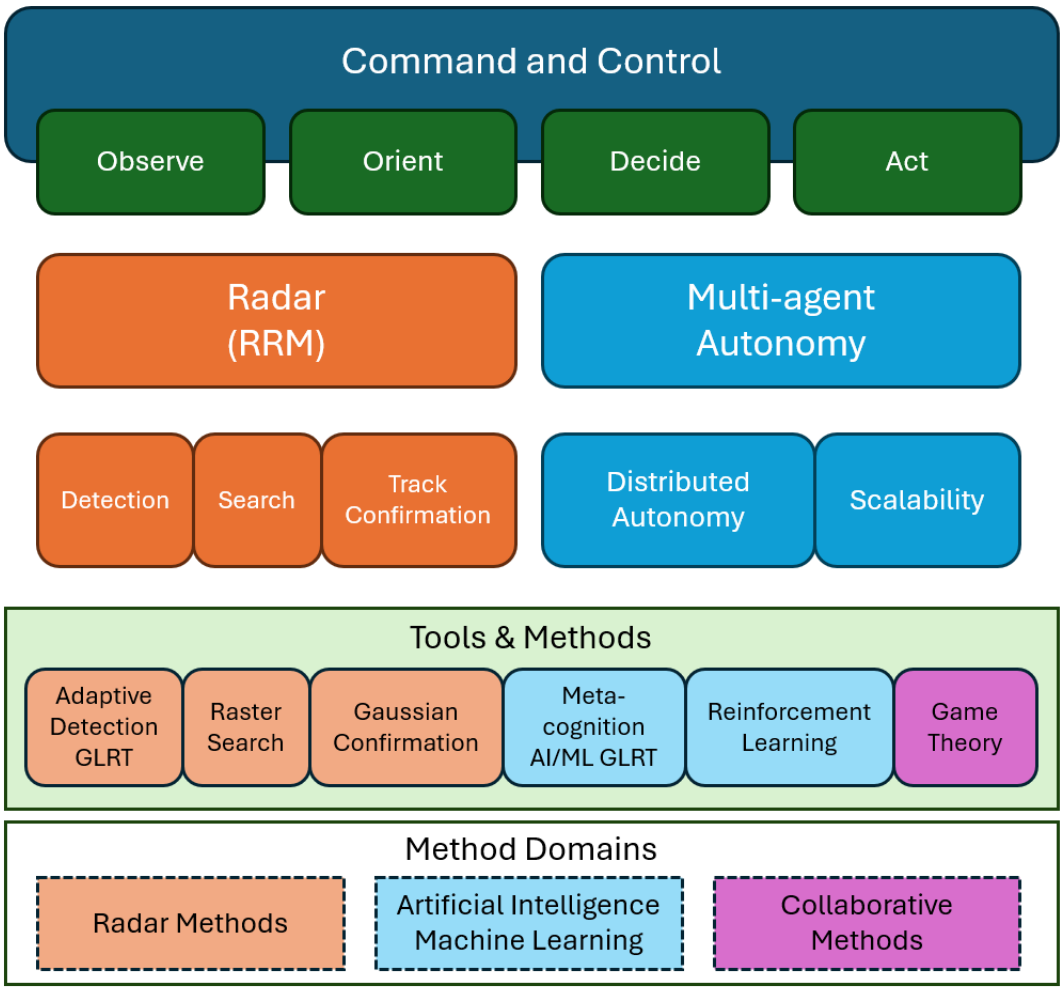


Figure 1.1: Domain and Methods Summary

1.1 Summary of Contributions

To showcase the effectiveness of collaborative, multi-agent methods that scale for autonomous systems, multiple techniques have been designed, implemented, and tested for multiple radar tasks in C2 scenarios. The work in this dissertation has one foundational simulation and three research areas that each have their own contributions.

1.1.1 C2 Scenario and Radar Simulation

The first area is not a contribution but a necessary simulation to facilitate the testing and validate the contributions for two of the following research areas. In order to test autonomous and collaborative methods, a C2 environment was necessary to generate scenarios, manage multiple agent's perception and actions, and have realistic responses to those actions. Essentially, the observation-action cycle for C2 scenarios requires a simulation to facilitate multiple factors in real time. As the work in this dissertation is focused on radar tasks, the observation portion of the simulation has an active radar component. Additionally, agents act as cognitive radar actors by making action selections that cause the radar simulation components to illuminate physical space in a given scenario.

The simulator has three levels of fidelity as it improved during the journey of this dissertation. The lowest fidelity version estimates the probability of detection using the radar range equation and a simulated phase array radar. The medium fidelity system utilizes more advanced clutter profiles to generate support samples for range and Doppler gates. These samples are used with an adaptive detector to produce detections and false alarms. The final simulation models a full pulse-Doppler radar with transmit waveform, simulated return signal with noise and clutter, ranged-Doppler processing, and a cell-averaging CFAR detector. Each version of the simulator has the same C2

environment but increasing fidelity in the radar observation simulation. As this simulator is not a research contribution it does not have a dedicated chapter, but given its importance in interpreting results a detailed description is included in Appendix A.4.

This simulator was used in the research topics: "Multi-Agent Track Confirmation" and "Scalable Multi-Agent Surveillance with Utility Representation". Additionally, it is explained and was used for three publications supporting this dissertation's contributions: "Collaborative Game Theory and Reinforcement Learning Improvements for Radar Tracking" [20], "Multi-agent track confirmation utilizing reinforcement learning and game theoretics" [21], and "Utility Representation for Collaborative Surveillance" [22].

1.1.2 Multi-Agent Track Confirmation

The first research area explores the impact of RL and GT methods for a multi-platform mission to confirm a track of a highly evasive target following an initial detection. A C2 scenario focused on the RRM task of track confirmation was designed and implemented with multiple radar backend simulations. Track confirmation focuses on the process of confirming that a detection is a target of interest instead of a false alarm. This requires multiple subsequent detections of the target in a short time period. Traditionally a radar system will illuminate the same space additional times given the assumption that either the target is not evasive or the time period to re-illuminate the target is such that it can not effectively leave the area of illumination.

This work is modeled after close range UAV detections, where both the target is evasive and the time between illuminations is longer. Additionally, the evasive target is designed to turn perpendicular to an illuminating source and thus has a relative velocity close to zero which is hidden in a ground clutter ridge. Given the evasive target and lower radar time resources, this work explores the viability of low-information tracking methods and the impact of game theory given multiple platforms that are

separated in space. Scenarios were simulated with medium-fidelity radar and clutter, and the scenario success, action error, and probability of detection were recorded. Additional analysis was conducted regarding the impact of platform separation and false alarms.

The core contributions were the exploration of low information tracking methods, including the AI/ML method of reinforcement learning, and the impact of collaborative game theory for multi-agent autonomy. Given the increase in UAVs and distributed autonomous systems, this research impacts both the RRM task of track confirmation as well as multi-agent collaboration. An early implementation of this work was published in the proceedings for the International RadarCon 2023 [20]. An expanded journal publication was published in IET Radar, Sonar, & Navigation in January of 2025 [21]. This research was funded under the Air Force Office of Scientific Research (AFOSR) grant with award number F4FGA03158J001.

1.1.3 Scalable Multi-Agent Surveillance with Utility Representation

The next research method was a first principles based utility method to optimize surveillance in C2 scenarios. Surveillance is the first RRM task and represents the challenge of searching space for potential targets of interest. This challenge is a balance of exploration of unknown space and exploitation of domain knowledge regarding likely regions of target presence. Traditionally, search methods assume all space has equal importance and should be given equal priority in the search algorithm. However, C2 scenarios often have spaces too large to fully search, and optimization is necessary. This search scenario was expanded to a multi-agent challenge where I established collaborative methods that scale with low computation for an unlimited number of platforms and can maximize multi-agent utility with and without direct information sharing.

These scenarios included C2 inspired regions of greater importance that are approached by targets of interest. I designed a GT based utility representation of the space that utilized platform position, the C2 based priorities, and a beta distribution to balance exploring new space and re-searching priority locations more often. This representation enabled passive and active collaboration of platforms that scales linearly. This representation was tested against multiple other methods to compare the number of targets detected and the target distance from vital locations when found.

The core contribution of this research is a scalable optimization for multi-agent surveillance. The C2 surveillance challenge requires the collaboration of multiple agents and approaches surveillance with domain priorities that should be utilized. The beta distribution based balancing of exploring unknown regions and revisiting high priority spaces is applicable to other search methods and provides a dynamic solution. This work has been accepted and will be presented in the 2025 International Radar conference [22]. This work was also funded under the AFOSR grant with award number F4FGA03158J001.

1.1.4 Scalable Meta-Cognitive Radar Detection

The final research contribution was the development of a meta-cognitive adaptive detector that made CFAR detection across multiple spherically invariant random variable (SIRV) types of clutter. I collaborated with a research partner and developed an adaptive detection system that used ML to determine statistical information about clutter samples and then dynamically adjusted the threshold to maintain a CFAR for samples under test (SUT). I utilized divergence methods to determine statistically distinct regions across multiple distributions. I designed an ML discriminator to determine the most likely distribution region. As part of the meta-cognitive structure, a second layer of specialized neural networks determined the optimal threshold.

An extension to the Meta-Cognitive detector work analyzed multiple clutter dis-

tributions and fit them into a single system. Clutter distributions have overlapping regions and high complexity when analyzed in full. This system broke the problem into manageable regions and resulted in finely-tuned threshold selectors. This resulted in CFAR performance for the full system. This system also produced smaller ML models that train quickly and operate at real-time speeds. I completed expansion work on this system to dynamically scale the system for different users with different clutter data or expectations. The original system was the result of training multiple models with experts exploring and testing to find the optimal configuration. The dynamic system I designed will automatically use data or user input to self-form, train, and test models so that this system can be used by a radar system without reliance on the original designers.

The core contributions for this research includes a statistical divergence method to determine statistically unique regions, an adaptive detector that has CFAR performance across multiple clutter distributions, and a scalable self-forming application that enables adoption for radar systems. This work was published in the IEEE Transactions on Aerospace and Electronic Systems as part of the Special Section on Deep Learning for Radar Applications [23] in 2023. This work was also funded under the AFOSR grant with award number F4FGA03158J001.

Chapter 2

Radar Background

2.1 Introduction

This dissertation will focus on scalable, multi-agent technologies within the command and control (C2) domain. The initial steps of C2 frameworks focus on "observe" and "orient" tasks. For modern systems, the domain of radar and radar signal processing is the primary mechanism to accomplish observation and orientation. Therefore, the work in this dissertation uses radar simulation and signal processing methods as the environment and agents for research exploration. These methods provide a driving force in applicable AI/ML and game theory based methods. The work presented uses a radar simulation and addresses scaling and multi-agent solutions to radar tasks. In this chapter, I will discuss signal formulation for a pulsed-Doppler radar, radar detection methods, C2 and radar resource management(RRM), and the specific RRM tasks of search and track confirmation.

2.2 Pulsed-Doppler Signal Formulation

The following sections will describe the formulation of a pulsed-Doppler radar transmitting an active radar signal and receiving signals including potential targets of interest, environmental noise, and clutter from the environment. Different detection

methods will utilize data at different levels of processing. This section will explain the signal formulation process from simulation of the transmit signal to range-Doppler processing and provide explanation for modeling realistic clutter and the effects of platform motion. The majority of the information in this section comes from "Fundamentals of Radar Signal Processing" by Mark Richards [24].

2.2.1 Radar Range Equation

Within radar signal processing, an active radar is characterized by a transmit signal of electromagnetic energy often referred to as $\mathbf{x}$. The corresponding return signal is formulated as a discrete-time measurement of reflected electromagnetic (EM) energy, ambient energy, and other unwanted EM sources. This return signal formulation is shown in equation 2.1:

$$\mathbf{z} = \mathbf{y} + \mathbf{n} + \mathbf{w} \tag{2.1}$$

Where $\mathbf{z}$ is the return signal, $\mathbf{y}$ is the potential return of target(s) of interest, $\mathbf{n}$ is the noise within the environment, and $\mathbf{w}$ are sources of interference. Often noise and clutter are represented together. Each component will be explained in more detail in the following sections. A common estimation of return power (P_r) is the radar range equation:

$$P_r = \frac{P_t G_{tx} G_{rx} \lambda^2 \sigma}{(4\pi)^3 L_s L_a R^4} \tag{2.2}$$

This equation utilizes the radar parameters to estimate the return power with P_t as the transmit power, G_{tx} the radar system gain during transmission, G_{rx} the receiver gain, λ the carrier wavelength, σ the radar cross section of a potential target, L_s is the system loss factor, L_a is the atmospheric loss, and R is the range. Additionally, a simple formulation for the noise power (P_n) can provide an estimate for the signal-to-noise ratio (SNR). This noise power is often estimated as:

$$P_n = k_b \beta F T_s \tag{2.3}$$

For noise power the important variables are Boltzmann's constant (k_b), (β) is the signal bandwidth, (F) is the noise figure, and (T_s) is system temperature. This can be combined to provide an estimate for the SNR which is useful for lower fidelity estimations. The final SNR equation is:

$$SNR = \frac{P_t G_{tx} G_{rx} \lambda^2 \sigma}{(4\pi)^3 k_b T_s \beta F L_s L_a R^4} \tag{2.4}$$

SNR calculations are vitally important because all signal processing methods need sufficient SNR or targets of interest can not be distinguished from ambient noise, let alone environmental clutter or active interference. For higher fidelity discussions in this section, the signal-to-interference-and-noise ratio ($SINR$) will be considered. $SINR$ will include the environmental and active interference source in it's calculation.

2.2.2 Pulsed Radar Transmit Signal

The following section will discuss a high-fidelity representation of radar transmit and receive signals. This is the foundation for later discussed signal processing methods used for radar detection. For pulsed-Doppler radar, a transmit signal is characterized by a carrier signal and a time-varying signature envelope ($a(t)$). The general form for a transmitted signal is:

$$x(t) = a(t)\cos(2\pi F_0 t + \theta(t)) \tag{2.5}$$

Where F_0 is the frequency for the carrier signal, t is time, and $\theta(t)$ is a time varying phase. Note that the envelope $a(t)$ and the phase $\theta(t)$ are not always present in every radar system. Although the return signal will contain additional noise and clutter

information, it will also contain reflections of this transmitted signal. In modern radar systems, this signal is usually sampled using a "quadrature" receiver. This provides radar information in a form called IQ-data, which represents the complex envelope. To understand the form of this data, consider the original transmit signal. Using the trigonometric identity: $\cos(A + B) = \cos(A)\cos(B) - \sin(A)\sin(B)$ the original signal can be written as:

$$\begin{aligned} x(t) &= a(t)\cos(2\pi F_0 t)\cos(\theta(t)) - a(t)\sin(2\pi F_0 t)\sin(\theta(t)) \\ x(t) &= x_I(t)\cos(\theta(t)) - x_Q(t)\sin(\theta(t)) \end{aligned} \tag{2.6}$$

These components of the complex envelope are called the in-phase component ($x_I(t) = a(t)\cos(2\pi F_0 t)$) and the quadrature component ($x_Q(t) = a(t)\sin(2\pi F_0 t)$). Given their slow variation compared to the high frequency carrier these are considered narrowband signals and can be represented as a complex envelope of form: $\tilde{x(t)} = x_I(t) + jx_Q(t) = a(t)\exp[j\theta(t)]$. The in-phase and quadrature signals are sampled within the receiver hardware by utilizing an oscillator with two mixers that are $90°$ out of phase. Some of the detection methods used in this dissertation will utilize IQ-data, but the following discussion will explore formulating the reflected information from potential targets of interest, ambient noise, and environmental clutter. This work assumes no active interference as the field of electronic warfare and jamming is out of scope but can be explored in more detail [25].

The following section will explain the formulation of a transmit signal and a receive signal containing potential targets of interest and ambient noise for a single waveform used as part of the simulation in this work. Earlier the transmit signal was formulated as a carrier signal and a complex envelope ($a(t)$). This envelope can take on different waveforms and is often windowed. The waveform can affect radar signal processing in multiple ways, including SNR and the range and Doppler resolution. There are multiple classes of waveform, including frequency modulated (FM), phase

codes, frequency codes, and random noise. Additional discussion of waveforms and their impact can be found in an overview by Shannon Blunt and Eric Mokole [26]. The most common waveform from the FM class is the linear FM (LFM). This is often used because it is simple to implement with hardware and has positive Doppler properties. The research in this work uses an LFM waveform and will be the focus of the discuss on formulation hence forth. The transmit signal is a baseband signal that is modulated to the carrier frequency. The baseband component is the waveform and is the focus of the following formulation. This baseband wave form can be represented as its baseband form $\tilde{x}(t) = a(t)\cos(2\pi F_c t)$ and the corresponding complex envelope $s(t) = a(t)\exp(j\theta(t))$. For profiling the LFM the focus will be on the complex envelope. The amplitude envelope ($a(t)$) is typically a rectangular windowing function. In this work the Hanning window was used for the amplitude envelope with the form:

$$w(n) = 0.5(1 - \cos(2\pi n/N), \text{ where } 0 \leq n \leq N \tag{2.7}$$

In this case $N = \frac{T_p}{T_s}$, where T_p is the transmit pulse width and the sampling interval is $T_s \geq \frac{1}{2\beta}$, with β as the bandwidth of the signal. Considering that the LFM is a frequency modulation technique, the majority of the waveform is within the phase component. For the LFM the phase function is a linear sweep from $[\frac{-\beta}{2}, \frac{\beta}{2}]$ and is represented as: $f_{LFM}(t) = \pm(\frac{-\beta}{2} + \frac{\beta}{T_p}t)$. The phase function is derived with the following equation:

$$\begin{aligned} \theta_{LFM}(t) &= 2\pi \int_0^t f_{LFM}(\zeta)d\zeta + \theta_0 \\ \theta_{LFM}(t) &= \pm 2\pi(\frac{-\beta}{2}t + \frac{\beta}{2T_p}t^2) + \theta_0 \end{aligned} \tag{2.8}$$

By combining, the final LFM complex envelope is:

$$s(t) = a(t)\exp(j2\pi(\frac{-\beta}{2}t + \frac{\beta}{2T_p}t^2) + \theta_0) \tag{2.9}$$

The final baseband signal takes the form:

$$s(t) = a(t)\cos(2\pi(F_c - \frac{\beta}{2})t + \pi\frac{\beta}{T_p}t^2 + \theta_0) \tag{2.10}$$

In most radar literature, the term gamma ($\gamma = \beta/T_p$) is used to represent the chirp rate for an LFM waveform. The variable θ_0 is the phase offset and is usually a random constant in simulation. An important variation to discuss is that often the chirp is a triangle as opposed to a sawtooth. The main reason for this is that the up frequency followed by the down frequency plays an important part in measuring the Doppler frequency in later processing. The LFM used in the simulator for this work used this up beat and down beat triangle frequency chirp method. A visual representation of an LFM complex envelope is shown in Figure 2.1.

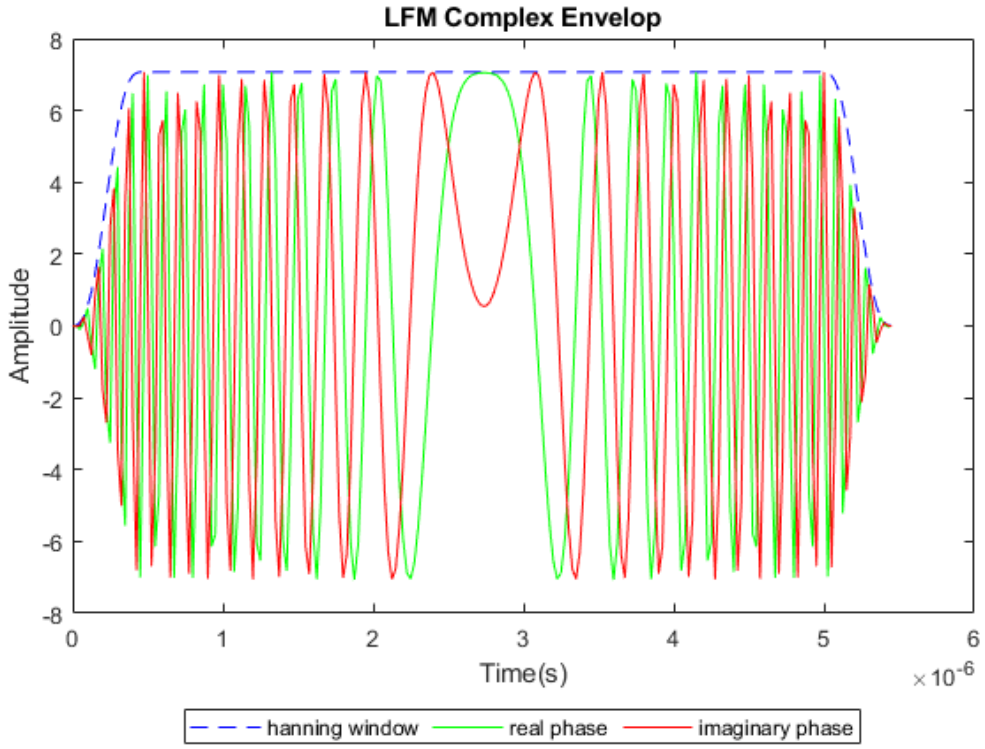


Figure 2.1: Complex Envelope for a Triangular LFM

In pulsed-Doppler radar this transmit signal is repeated for M pulses. It is through

these M pulses that the Doppler information can be processed and differentiates pulsed-Doppler radar. The rate at which the pulses are repeated is called the pulse repetition frequency (PRF), and its reciprocal is called the pulse repetition interval ($PRI = \frac{1}{PRF}$). The coherent pulse interval (CPI) is the product of the M pulses and the pri.

2.2.3 Target of Interest Formulation

Now that we have a realistic representation of a transmit signal, the following section will discuss the process to represent the range and velocity impacts of a target of interest on the return function. Assuming there is a target at range (R), and a relative velocity (v_r), the return signal will have a time delay due to range and a Doppler shift due to the relative velocity. The relative velocity is the radial component of velocity between the target and the platform. The range component is related to the time it would take the EM wave to travel to and from the target. This delay is $\tau = \frac{2R}{c}$, where c is the speed of light. Considering the repeated pulses and the velocity of the target, the range at each pulse differs in position. Typically the variable $m = [-M/2, -M/2+1, \cdots, 0, \cdots, M/2-1]$ is used to represent the pulses centered at 0. M is usually chosen to be a base of 2 (64, 128, 256, etc.) and has the trade-off that more pulses improve Doppler resolution but also increase the CPI. The real time delay is a function of m taking the form:

$$\tau(m) = \frac{2(R - v_r \cdot m \cdot pri)}{c} \tag{2.11}$$

The range at each pulse is increased by the target velocity. The effect of the velocity is to increase or decrease the frequency of the return signal compared to the transmitted signal. The Doppler shift (F_D) is a function of the relative velocity and is:

$$F_D = F_0(\frac{c + v_r}{c - v_r}) - F_0 \approx \frac{2v_r}{c} F_0 = \frac{2v_r}{\lambda_0} \tag{2.12}$$

This approximation holds true if the relative velocity is much lower than c which is true for terrestrial targets. The final component is the scaling of the return power expected for the signal. From 2.13, This ratio of the return power given the transmit power is:

$$\alpha = \sqrt{\frac{P_r}{P_t}} = \sqrt{\frac{G^2\lambda^2\sigma}{(4\pi)^3 R^4}} \tag{2.13}$$

By combining all of these components together the return signal will be time shifted by $\tau(m)$, scaled by alpha α, and Doppler shifted by the Doppler frequency F_d. The receive waveform has the form:

$$\tilde{y}(t) = \alpha a(t - \tau(m)) \cos(2\pi(F_0 + F_D)(t - tau(m)) + \theta(t - \tau(m)) \tag{2.14}$$

The final form of the complex envelope is then:

$$\begin{aligned} y(t) &= \alpha a(t - \tau(m)) \exp(-j2\pi(F_0 + F_D)\tau(m) + j2\pi F_D t + j\theta(t - \tau(m)) \\ &= \alpha a(t - \tau(m)) \exp(j\theta(t - \tau(m)) \exp(-j2\pi(F_0 + F_D)\tau(m) + j2\pi F_D t) \end{aligned} \tag{2.15}$$

Recall that $\theta(t)$ for an LFM was given in equation (2.8). Although this complex envelope does have all of the components needed to simulate a target, one can utilize both the time and frequency domain, through the use of a Fourier transform. This simulates the target with a series of simple steps. Considering that the delay τ is for each sample m, the following steps will include the sample dimension, as these changes are needed for each sample. The steps to calculate a return envelope from the transmit envelope are as follows:

1. Define the length of the samples for a desired range $L = \frac{t_{max} - t_{min}}{T_s}$ where $t_{max} = \frac{2R_{max}}{c} + T_p$ and $t_{min} = \frac{2R_{min}}{c}$.

2. Complete a length L fast Fourier transform (FFT) and an FFT shift of $x(t, m) \rightarrow X(f, m)$

3. Generate a frequency range (f) of length L from $-F_s/2$ to $F_s/2$.

4. Implement the time shift due to $\tau(m)$ in the frequency domain by multiplying $X(f,m)_{ts} = X(f)\exp(-j2\pi f(\tau(m) - t_i))$

5. Conduct an inverse FFT shift and then a inverse FFT ($X_{ts}(f,m) \rightarrow x_{ts}(t,m)$).

6. Finally, apply the Doppler shift and alpha scaling in the time domain using: $y(t,m) = x_{ts}(t,m)\alpha\exp(j2\pi F_D t)\exp(j\theta_0)$

This pulsed-Doppler return now includes the target information for each sample in the CPI. This process can be repeated for multiple targets of interest that are expected to be in the beam-width of a transmitted signal.

2.2.4 Thermal Noise

Now that a received CPI can be simulated with targets of interest, a realistic signal will contain both noise and clutter from the environment. Thermal noise is due to the motion of electrons within any material that is greater than 0° K. This noise is present in the receiver hardware and propagates through the radar system before signal processing can be performed. To represent this loss, the term F is used as the noise figure of the radar system. A detailed discussion and methods for calculating the noise figure for specific radar systems can be found in [27]. For many simulations, a reasonable noise figure between 2-10 dB is selected. The noise power is typically estimated using (2.3). Thermal noise is modeled as additive white Gaussian noise (AWGN). Gaussian white noise is characterized by having zero mean, is independent with regards to time, has constant power across all frequencies, and follows a Gaussian distribution. In reality there is a high frequency component of radar electorincs, but in the operation range of radar systems, it is fair to assume thermal noise is AWGN. The process to generate complex AWGN to be added to the return CPI $y(t,m)$ is shown in the following equation:

$$n(t,m) = \sqrt{P_n/2}\mathcal{N}[0,1](t,m) + j\mathcal{N}[0,1](t,m) \tag{2.16}$$

Where $N[0,1](t,m)$ is the normal distribution with mean of 0 and standard variation of 1 for all terms in t and m. This noise is then added to the CPI $y(t,m)$. The visual impact of this thermal noise is shown in section 2.2.7 in Figure 2.4 following the discussion of range-Doppler processing.

2.2.5 Matched Filter

Now that we have a formulation for the return signal, it is important to address the most common processing method performed on return signals, called the matched filter. The matched filter is designed to maximize the signal-to-noise ratio at some time for a given signal. For a reflected transmit signal within a return signal, a matched filter is designed to be a time revered, conjugated copy of the transmitted waveform. The time and frequency versions of the matched filter are shown in the following equations:

$$\begin{aligned} h(t) &= \alpha x^*(T_M - t) \\ H(F) &= \alpha X^*(F) \exp -j2\pi F T_m \end{aligned} \tag{2.17}$$

Where T_M is the time where we want to maximize the SNR and typically is equal or greater than the transmit pulse ($T_M \geq T_p$) in order to maintain causality. The Cauchy-Schwarz inequality is used to derive the matched filter and can be explored in detail in [24]. The term α is a scaling constant that is usually set to 1 during the analysis of signal returns. The filtering process is a convolution of the signal and the filter. In practice, matched filter processing is usually done in the frequency domain as the convolution corresponds to multiplication. The process for applying the matched filter is the following steps:

1. Given the time based return signal $y(t,m)$.
2. Complete a FFT and an FFT shift for the return signal of $y(t,m) \rightarrow Y(f,m)$

3. Generate the matched filter by taking the conjugate of the transmit signal and reversing the signal order $h(t) = x^*(-t)$.

4. Convert the matched filter to frequency domain with FFT and FFT shift. $(h(t) \rightarrow H(f))$

5. Multiply the matched filter and the return signal in frequency domain: $Y_m(f, m) = Y(f, m) \cdot H(f)$

6. Finally, return to the time domain with an inverse FFT shift and inverse FFT: $Y_m(f, m) \rightarrow y_m(t, m)$

The matched filter is not the only range processing method, but it is simple, effective so it is the method method used in this dissertation. The matched filter is an important step for range-Doppler processing in section 2.2.7. Matched filtering is also used to derive the ambiguity function, which is an important tool for analyzing the effect of the waveform in time and Doppler frequency. The work in this dissertation is not focused on waveform optimization, and the LFM was chosen for its positive range resolution characteristics, which are important for radar simulations using these methods. More information on the ambiguity function can be found in [24].

2.2.6 Clutter Simulation

Clutter is a passive form of interference that is the result of environmental reflections that are not targets of interest. There are multiple forms of clutter with an extensive literature explaining their formulation and foundation. Two sources with detail on clutter simulation include chapter 2 of "Fundamentals of Radar Signal Processing" [24] and chapter 11 of "Academic Press Library in Signal Processing" by Greco and Watts [28]. The following section will discuss two methods relevant to a scenario of an airborne pulsed-Doppler radar illuminating targets with a ground-based, environmental background.

The first form of clutter is due to the environment over different ranges. There are three major methods to represent this clutter: point scatterers, volume scatterers, and area scatterers. Point scatters use the same method as targets of interest as they represent strong reflection points in the environment. Often, the clutter simulation methods use a distribution to spread the points in the environment. Point scatters are computationally expensive to simulate and need appropriate density to map to physical features. Volume scatters and area scatters utilize the normalized clutter reflectivity (σ^0) to represent the distribution of clutter power in a similar method as RCS to represent the reflectivity of a target. Volume scatters are usually used for platforms that have weather as the radar environment background. Figure 2.2 from [28] shows a visual of a volume scatter scenario.

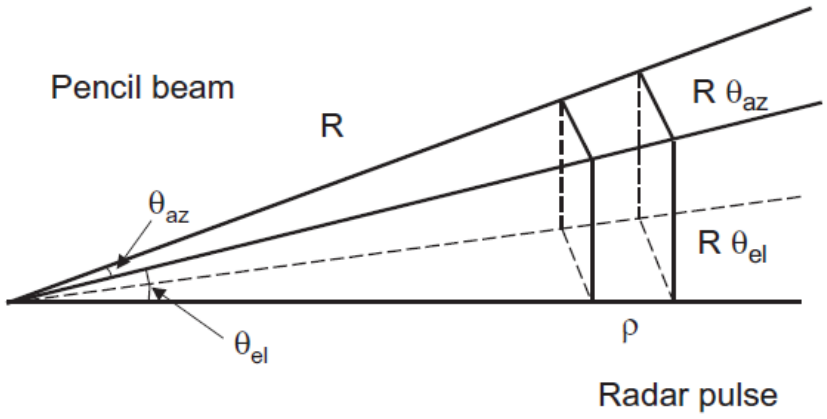


Figure 2.2: Volume Clutter Illumination

The equation for the illuminated volume and the corresponding volume reflectivity η is:

$$\begin{aligned} V_c &= \alpha \rho R^2 \theta_{az} \theta_{el} \\ \eta &= \frac{\sigma}{V_c} \end{aligned} \tag{2.18}$$

Where α is a scaling factor due to the compressed waveform and is 1 for

rectangular-shaped pulses and 0.753 for Gaussian-shaped beams. The range resolution is ρ. θ_{az} and θ_{el} are the azimuth and elevation beam width. According to [28], the value of reflectivity (Z) for weather patterns tends to range from -35 to 80 dBZ. Note that dBZ is a decibel conversion of the units mm^6/m^3 and is 180dB greater than volume reflectivity η with units of m^6/m^3. The full volume scatter power equation is a modification of the radar range equation and is:

$$P_{vc}(R_0) = \frac{P_t G^2 \lambda^2 \eta \rho \theta_{az} \theta_{el}}{(4\pi)^3 R_0^2 L_s L_a(R_0)} \tag{2.19}$$

For area scatterers there are two cases based on if a radar illumination beam is beam-limited or pulse-limited. The different cases are illustrated in Figure 2.3.

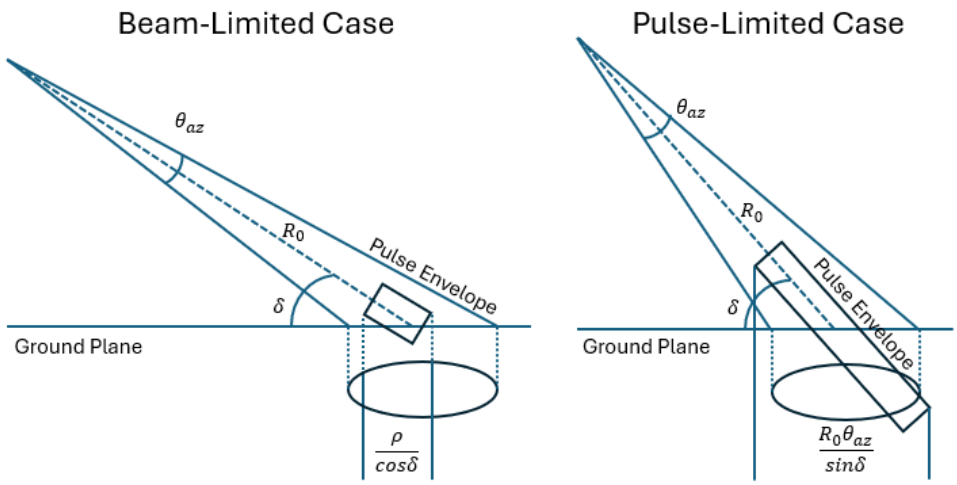


Figure 2.3: Area Scatterer Examples

The boundary cases are differentiated with the following inequality:

$$\begin{cases} \text{Beam limited:} & \frac{\rho}{R}\tan\delta > \theta_{az} \\ \text{Pulse limited:} & \frac{\rho}{R}\tan\delta < \theta_{az} \end{cases} \tag{2.20}$$

The power equation for the beam limited case is the range equation modified by the area backscatter for one instance of $R^2\theta_{az}\theta_{el}/\sin\delta$, to make the following equation

for clutter power:

$$P_{ABL}(R_0) = \frac{P_t G^2 \lambda^2 \sigma^0 \theta_{az} \theta_{el}}{(4\pi)^3 R_0^2 L_s L_a(R_0) \sin\delta} \tag{2.21}$$

For the pulse limited case the one instant back scatter is $R\rho/\cos\delta$ and has a corresponding clutter power of:

$$P_{APL}(R_0) = \frac{P_t G^2 \lambda^2 \sigma^0 \rho \theta_{el}}{(4\pi)^3 R_0^3 L_s L_a(R_0) \cos\delta} \tag{2.22}$$

The final step to determine the power from an area scatterer is to determine the normalized clutter reflectivity (σ^0). There are multiple models used to estimate this parameter. The simplest model is the constant gamma method which uses the equation $\sigma^0 = \gamma sin\delta$. This is used to estimate the plateau region of empirically measured back scatter from [29]. A general estimation of gamma for different terrain is provided by Barton [30] and is replicated in Table 2.1.

Terrain	**Mean γ (dB m^2/m^2)**
Mountain, Urban	-5
Wooded hills	-10
Rolling hills	-5
Farmland, desert	-15
Flatland	-20

Table 2.1: Typical Values of γ for Different Types of Terrain

The second clutter type to explore is ground clutter. This clutter is present for platforms illumining targets with a background that is the ground. It is processed as a band near 0 Hz in Doppler that represents the stationary or slow moving ground environment. For a pulse-Doppler radar this uses the clutter power from the area clutter method described above but models the Doppler component appropriately. The primary source for this Doppler modeling comes from the work by Hassanein *et al.*

[31]. This work estimates a clutter covariance matrix (CCM) to generate the samples with the following equation representing the CCM values.

$$\tilde{r}(n) = P_c e^{-8(\frac{\pi n T_p \sigma}{\lambda})^2} \tag{2.23}$$

Where T_p is the pulse repetition interval, and λ is the wavelength. The $N \times N$ Clutter Covariance matrix is then constructed, with each component being calculated with (2.23), where n is the distance from the diagonal of the matrix using the following structure:

$$\begin{pmatrix} \tilde{r}(0) & \tilde{r}(1) & \cdots & \tilde{r}(N-1) \\ \tilde{r}(1) & \tilde{r}(0) & \cdots & \vdots \\ \vdots & \vdots & \ddots & \tilde{r}(1) \\ \tilde{r}(N-1) & \cdots & \tilde{r}(1) & \tilde{r}(0) \end{pmatrix} \tag{2.24}$$

This CCM can then be used to generate samples from a multivariate Gaussian distribution. Together this provides a clutter signal ($w(t, m)$) that can be added to the noise and potential target return signal to represent a simulation of the return signal (z) as given in (2.1). The visual representation of the full signal is shown in section 2.2.7 in Figure 2.4 following the discussion of range-Doppler processing.

2.2.7 Range-Doppler Processing

Considering that a target of interest will delay the signal in time given its range and induce a Doppler shift due to its relative velocity, we would want to be able to distinguish targets given both features. This process is called range-Doppler processing as we will take the return CPI $z(t, m)$ and process the range from the time axis (fast time) and the velocity/Doppler information from the samples m (slow time).

As discussed in Section 2.2.5, matched filtering compresses the fast time response into a range axis. For the slow time axis we will use the FFT and a FFT shift to trans-

late the samples into the frequency domain. The result will display range and Doppler cells in two different axis that visualize targets and clutter. This range-Doppler data can be used in future detection methods like cell-averaging constant false alarm rate (CA-CFAR) detection.

The range Doppler plot is effective at the visualization of a CPI, therefore the following figures are used to display the simulation features. Figure 2.4 shows the range-Doppler plot ($z_{RD}(R, f_D)$ or $z_{RD}(R, v_r)$) for a full return signal $z(t, m)$ which contains a single target at a range of $R = 6940m$ and a relative velocity of $v_r = -122m/s$ and has both thermal noise n and area scatter clutter w.

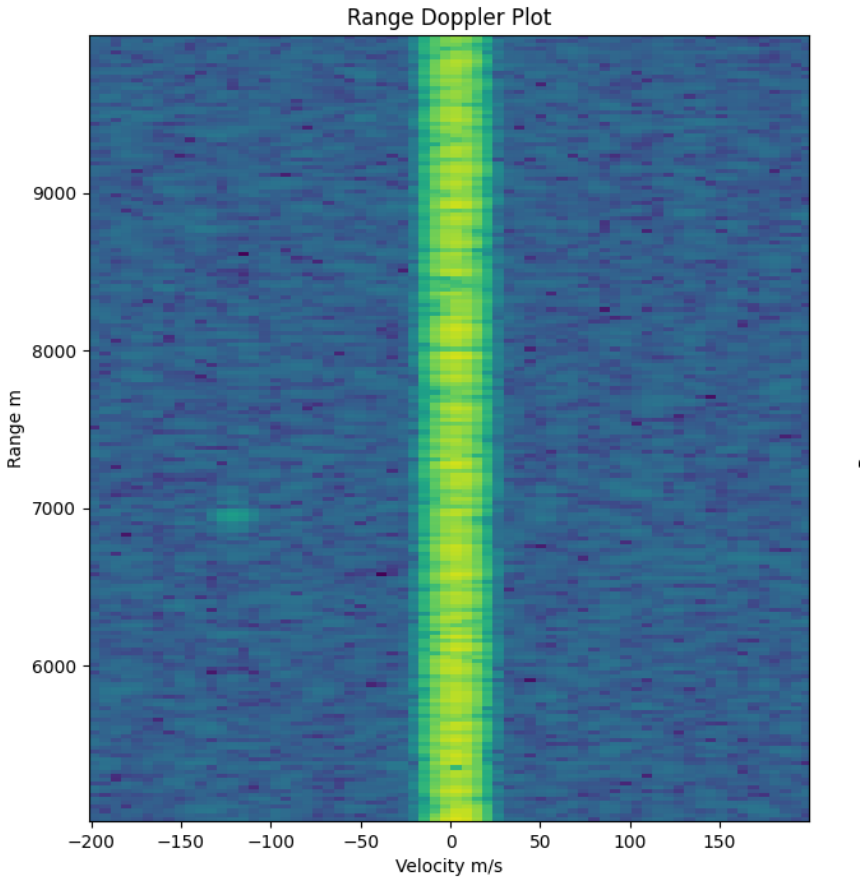


Figure 2.4: Example Range Doppler Plot

As can be seen in Figure 2.4, the target of interest is present (close to the true location), but the range, velocity resolution, and thermal noise obscure the true information. This figure visualizes the ground clutter from an area scatterer and the clutter ridge from the work in [31].

2.3 Radar Detection

Detection is a foundational task within radar signal processing, as most other tasks are dependent on the information provided during the detection process. In simple terms, detection is the determination of whether a target of interest is present within a return signal or a CPI of return information. Beyond the determination of the presence of the target is the additional estimation of the range and relative velocity of a target if detected. An important aspect of detection is understanding the relationship between detection and false alarms. This problem assumes a pulse-Doppler radar that transmits m pulses, comprising a coherent processing interval (CPI). The return signal is received and digitally sampled to create m pulses of length n. A single range bin within the CPI is a complex signal, $\mathbf{z}$ with the form:

$$\mathbf{z} = [z_1, z_2, ..., z_m]^T \tag{2.25}$$

The function of a detection system is to determine whether or not $\mathbf{z}$ contains a signal return from a target of interest. In the event that it does, then $\mathbf{z}$ is comprised of an additive combination of the signal of interest and interference. Otherwise, $\mathbf{z}$ consists entirely of interference. This can be written as a binary hypothesis test. The two hypotheses can be represented as:

$$\begin{cases} \mathbf{H_0}: & \mathbf{z} = \mathbf{n} + \mathbf{w} \\ \mathbf{H_1}: & \mathbf{z} = \mathbf{s} + \mathbf{n} + \mathbf{w} \end{cases} \tag{2.26}$$

In (2.26), $\mathbf{n}$ is the m-dimensional complex noise vector, $\mathbf{n}$ is the m-dimensional complex interference vector and $\mathbf{s}$ is the m-dimensional target return vector. The results of this hypothesis test and the truth of the signal return results in four possible situations. These four hypothesis results are shown in Figure 2.5.

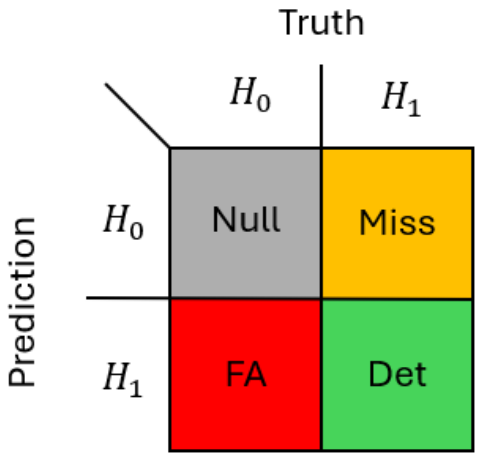


Figure 2.5: Hypothesis Test States

The two desired states are when prediction matches truth which happens in the: null state and detection state. The other two conditions are a disconnect from prediction and truth and negatively influence performance: miss state and false alarm (FA) state. For a series of N events the two ratios of importance for detection systems are the probability of detection ($P_D = \frac{\sum Det}{(\sum Det + \sum Miss)}$ for the H_1 events) and the probability of false alarm ($P_{FA} = \frac{\sum FA}{(\sum FA + \sum Null)}$ for the H_0 events). As the number of empirical events approach infinity the P_D and P_{FA} approach the true values for the system and environment. The following section will explain multiple detection methods starting with P_D estimates, a constant false alarm (CFAR) method and an expansive discussion of space-time adaptive processing and adaptive detection methods.

2.3.1 Probability of Detection Approximations

For simple simulations often the goal is to represent a probability of detection given the conditions of the radar system and the environment. These method are simple but lack a P_{FA} condition or assume a constant P_{FA}. The first method is a Swerling 1 estimation method from Richards [32]. 2.27:

$$P_D(R) = P_{FA}^{(\frac{1}{1+\mathrm{SNR}(R)_{lin}})} \tag{2.27}$$

This method uses desired P_{FA} and an SNR calculation from 2.4. A second simple method is derived from Albersheim's equation [33]. This equation estimates the SNR from the P_D, P_{FA}, and the number of samples N. Albersheim's equations are given as

$$\begin{aligned} &A = \ln(\frac{0.62}{P_{FA}}),\ B = \ln(\frac{P_D}{1-P_D}) \\ &\mathrm{SNR}_{dB} = -5\log_{10} N + (6.2 + \frac{4.54}{\sqrt{n+0.44}})\log_{10}(A + 0.12AB + 1.7B) \end{aligned} \tag{2.28}$$

The derivation provided by Richards in [33] shows a derivation for estimating P_D using Albersheim's equations:

$$\begin{aligned} &Z = \frac{SNR_{dB} + 5\log_{10} N}{6.2 + \frac{4.54}{\sqrt{N+0.44}}}), \quad B = \frac{10^Z - A}{1.7 + 0.12A} \\ &P_D = \frac{1}{1+e^{-B}} \end{aligned} \tag{2.29}$$

These simple estimation methods do not make a hypothesis test decision for a series of events. Instead, they provide an estimated P_D which can be used to determine a detection by generating a random uniform number ($U[0,1]$) and then comparing if it is greater or less than the estimated P_D.

2.3.2 Cell Averaging CFAR

A detection method often used on range-Doppler processed data is the cell-averaging CFAR method. This method uses a threshold to make a detection determination for a cell under test (CUT) by utilizing the mean of neighboring cells and a desired probability of false alarm. The logic behind this method is that these support cell represent the H_0 case which includes only noise and clutter. Since the H_1 case will also contain a target of interest then if we use the mean of the support samples as part of the threshold then a target will return a power value high enough to pass some threshold.

The equations for this method are:

$$\begin{aligned} \lambda &= \alpha \tilde{P_n} \\ \tilde{P_n} &= \frac{1}{N} \sum_{m=1}^{N} x_m \\ \alpha &= N(P_{fa}^{-1/N} - 1) \end{aligned} \tag{2.30}$$

In these equation λ is the threshold value, $\tilde{P_n}$ is the estimated noise/clutter power, N is the number of support samples, x_m are individual support samples, and P_{fa} is the desired probability of false alarm. The final detection determination uses the following inequality: if $x_{CUT} > \lambda$ then a detection is assumed. In order to determine the support samples two parameters are used in cell-averaging CFAR: guard cells G and training cells T. The guard cell parameter are the rectangular offset of cells directly around the CUT that are not included in the support samples. The train cell parameter determine the offset beyond the guard cells that will constitute the support samples. A visual representation is shown in figure 2.6.

Cell averaging CFAR can be performed across single or 2 dimensional data. The following is the amount a samples for range Doppler data given the guard and training

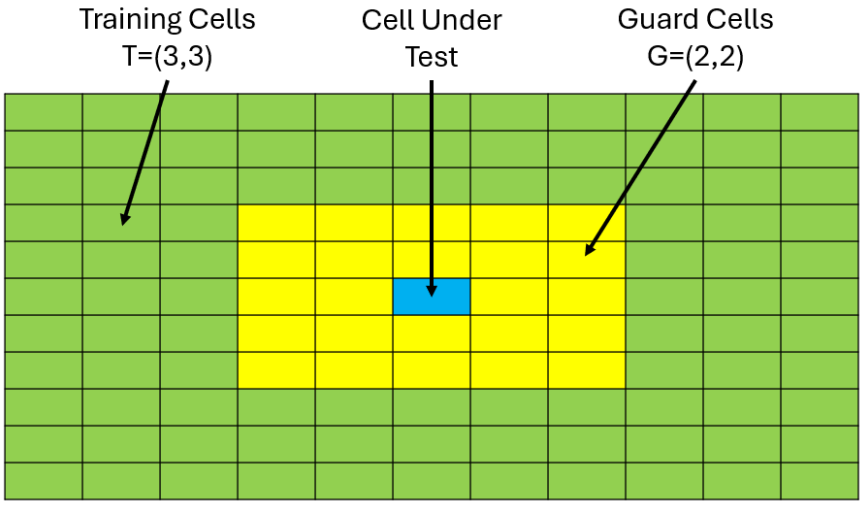


Figure 2.6: Example of 2D Cell Averaging CFAR

parameters: $N = (2G_R + 1 + 2T_R)(2G_D + 1 + 2T_D) - (2G_R + 1)(2G_D + 1)$, where $G = (G_R, G_D)$, $T = (T_R, T_D)$, R is the range dimension, and D is the Doppler dimension. This evaluation is performed on every cell of interest in the range and Doppler data. Because the cells near the edges need support samples it is typical to process more data than will be evaluated or to use a padding method.

Since every cell in the RD data needs to be evaluated, two nested loops can be used but this is computationally inefficient. Conveniently, the cell averaging CFAR method can utilize a convolution operation to process the full RD data much more efficiently. A practical implementation of cell averaging CFAR is as follows:

1. Find N given G and T.
2. Calculate α for the operation given P_{fa} and N.
3. Build a 2-D window W as seen in Figure 2.6, where the value of training cells is $1/N$ and the guard cells and CUT are 0.
4. Use convolution to create an estimation matrix $\tilde{\mathbf{P}}_\mathbf{n}$ from the original RD data $z_{RD}(R, f_D)$: $\tilde{\mathbf{P}}_\mathbf{n} = W \cdot z_{RD}$.

5. Create the CA-CFAR plot (z_{CACFAR}) by compareing each point in the RD data to the threshold: For $z_{RD} > \alpha\tilde{\mathbf{P}}_{\mathbf{n}}$, $z_{CACFAR} = 1$ and for $z_{RD} \geq \alpha\tilde{\mathbf{P}}_{\mathbf{n}}$, $z_{CACFAR} = 0$.

An example of a CA-CFAR operation is shown in Figure 2.7 with a target at a range of $R = 4677\text{m}$ and a relative velocity of $v_r = -83\text{m/s}$ along with the presence of thermal noise n and area scatter clutter w.

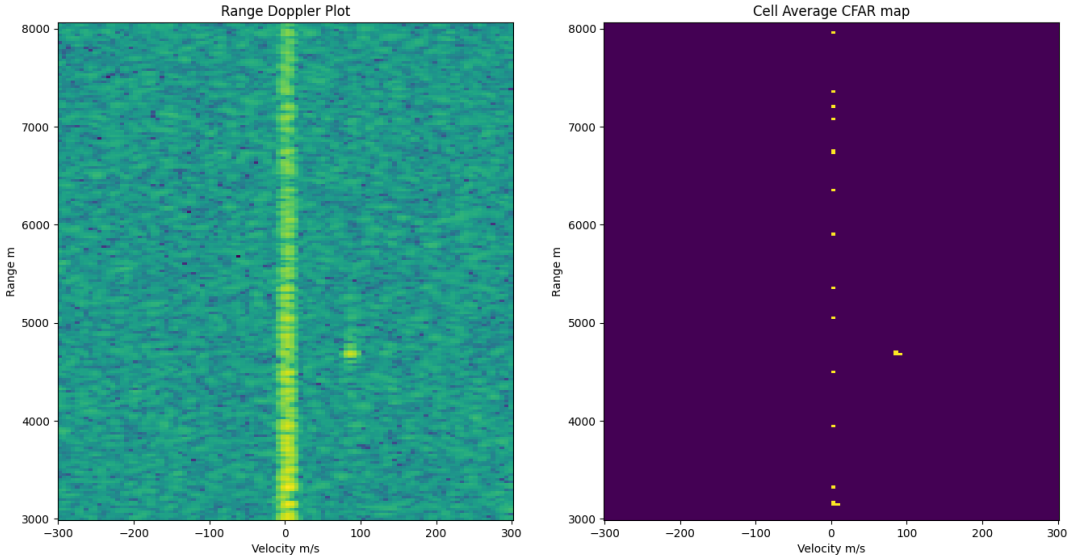


Figure 2.7: Example of CACFAR

Given that larger power returns from targets can be detected by CA-CFAR in multiple adjacent bins, an association is often made to find a center point and represent the bins as a single detection for the list of detections returned.

2.3.3 STAP Detection and Non-Gaussian Clutter

The following is a discussion of the literature around radar detection with a specific focus on adaptive detection utilizing space-time adaptive processing (STAP) given raw I/Q data from a pulsed-Doppler radar. Throughout the literature, the interference is modeled as $\mathbf{n} = \sqrt{\tau}\mathbf{x}$, where τ is referred to as the texture and $\mathbf{x}$ is the speckle. Here,

τ is positive and represents localized clutter power. $\mathbf{x}$ represents local back-scattering and is typically modeled as an m-dimensional Gaussian random vector with zero mean and normalized Hermitian covariance matrix $E[\mathbf{n}\mathbf{n}^H] = \mathbf{M}$. For convenience in simulation, here we adopt the model where $\mathbf{M}_{i,j} = \rho^{|i-j|}$ $(1 \leq i, j \leq m)$. The correlation coefficient, ρ, is positive in the range $0 \leq \rho \leq 1$, with higher values modeling more correlated interference and lower values modeling less correlated interference. The results presented in this dissertation are general and applicable to any scenario where the interference term is correlated. In practice for a single channel radar a Doppler power spectral density model (PSD) would be adopted [34], and for a STAP application a more advanced covariance model would be introduced [35].

The target return signal is modeled as $\mathbf{s} = \alpha\mathbf{p}$. α is an unknown positive random value representing the amplitude of the signal return. This is meant to capture the effects of channel propagation. $\mathbf{p}$ is the m-dimensional known steering vector $\mathbf{p} = [e^{-j2\pi f_D T_s}, ..., e^{-j2\pi m f_D T_s}]$, where f_D is the Doppler frequency and T_s is the pulse repetition interval. We can now rewrite the previous hypotheses as:

$$\begin{cases} \mathbf{H_0}: & \mathbf{z} = \sqrt{\tau}\mathbf{x} \\ \mathbf{H_1}: & \mathbf{z} = \alpha\mathbf{p} + \sqrt{\tau}\mathbf{x} \end{cases} \tag{2.31}$$

In addition to the sample/cell under test, it is assumed that there is a number K, of additional samples that contain only interference. These are typically referred to as training or secondary data in the literature; here they are represented as $\mathbf{z}_k = \mathbf{n}_k$ where $E[\mathbf{n}_k\mathbf{n}_k^H] = E[\mathbf{n}\mathbf{n}^H] = \mathbf{M}$.

2.3.4 Clutter and Spherically Invariant Random Processes

An important component of a CUT in the above problem statement is the interference or clutter represented with n. This clutter represents the reflected transmit energy due to the environment which can increase the signal to interference and noise

ratio (SINR). The most common method for detection is to establish a threshold value that is compared with a test statistic. For adaptive detection the test statistic is usually related to an estimate of the clutter covariance matrix (CCM). Section 2.3.7 will go into specific detail regarding specific adaptive detector test statistic methods. Before discussing the development of adaptive detector test statistics and threshold selection the modeling of clutter should be explored.

The most common representation for clutter is the Gaussian case. This is largely influenced by the central limit theory assumption that random samples from an independent stochastic process will be Gaussian. Due to the nature of Gaussian distributions, clutter of this type is more simple to model and some detectors can use Gaussian assumption to generate closed form solutions. Although many natural process can be accurately represented with a Gaussian distribution, many types of environments (ocean, urban, etc.) cannot and need a different representative distribution. A common clutter representation for non-Gaussian clutter is the use of spherically invariant random processes (SIRPs) and their sample vectors (SIRVs). SIRPs and SIRVs are special families of random processes and vectors with key characteristics that make them useful in modeling clutter [36]. SIRVs are random vectors that can be entirely defined by a mean vector, covariance matrix, and a first order characteristic probability density function (pdf). SIRPs are random processes from which any sampled vector will be a SIRV. In [37], SIRVs have certain properties that make them useful as a representative distribution and enable methods for sample generation, which is important for simulated research.

Rangaswamy *et al.* define four theorems of SIRVs that enable this behavior, along with two methods for simulating clutter as SIRVs. The first two of these theorems are especially relevant for research in this dissertation. The first states:

Theorem 1. *If* $\mathbf{z} = [z_1 z_2 ... z_N]$ *is a complex SIRV, there must exist a non-negative random variable S with PDF* $f_S(s)$ *that can be used to condition* $\mathbf{z}$ *to generate a*

complex Gaussian random vector, $\mathbf{x} = [x_1 x_2 ... x_N]$ *with zero mean and covariance matrix* $\mathbf{M}$.

It follows from Theorem 1 that we get $\mathbf{z} = S\mathbf{x}$. It is then clear that by setting $S = \sqrt{\tau}$, we get the form of H_0 in (2.31) with the conditioning variable $\sqrt{\tau}$ representing the texture and the zero mean Gaussian vector $\mathbf{x}$ the speckle. Note that S and $\mathbf{x}$ are assumed to be statistically independent, allowing for unique control of the texture and speckle elements. This is a powerful relationship that allows any SIRV to be generated as a combination of a zero mean Gaussian distribution and a characteristic PDF. Theorem 2 extends this relationship:

Theorem 2. *When a SIRV,* $\mathbf{z} = [z_1 z_2 ... z_N]$*, with characteristic function* $f_S(s)$ *undergoes a linear transform:*

$$\mathbf{y} = \mathbf{z}\mathbf{A} + \mu \tag{2.32}$$

The result, $\mathbf{y}$*, is also a SIRV with the characteristic function* $f_S(s)$*, mean* μ*, and covariance* $\mathbf{M} = \mathbf{A}\mathbf{A}^{\mathbf{T}}$.

Based on Theorem 2, it is possible to transform a SIRV with zero mean and identity covariance matrix into a SIRV with the same characteristic PDF and any desired covariance structure and mean using a simple linear transformation.

2.3.5 Generation Methods for Specific SIRVs

The research in this dissertation often simulates a clutter component in return samples that utilized a SIRV of the Gaussian family. In general there are four common methods for generating random variables. Distributions like uniform, Gaussian, and gamma are readily available with modern software (e.g. Python sci-py, Matlab) The second method uses a uniform distribution with the inverse of the cumulative distribution function(cdf) [38, 39, 40].

Compound Gaussian distributions have been broadly applied in the literature to

better model these more challenging clutter scenarios. In this work, three of the more commonly used clutter distributions are used in the development and testing of the proposed system: Gaussian, Pareto, and the K-distribution.

As was described in Section 2.3.4, the interference ($\mathbf{n}$) is modeled as a multiplicative combination of speckle ($\mathbf{x}$) and texture (τ). In the work of this dissertation, speckle is modeled as a complex zero-mean Gaussian vector with Hermitian covariance matrix ($\mathbf{M}$) defined by the one-lag correlation coefficient (ρ). The clutter texture is used to represent the local power level of the clutter and is modeled as a positive real value that remains constant over the course of a signal but can vary between signals. Modeling the clutter in this way as $\mathbf{n} = \sqrt{\tau}\mathbf{x}$ means that the clutter texture can be used to easily adjust the distribution of the generated data so that if fits either a Gaussian or compound Gaussian distribution ([41],[42]). As described in [43] this results in the PDF of a clutter-only signal taking the form:

$$f_{\mathbf{n}}(\mathbf{n}) = \frac{1}{(\pi)^N ||\mathbf{M}||} \int \frac{1}{(\tau)^N} e^{-\frac{\mathbf{n}^H \mathbf{M}^{-1} \mathbf{n}}{\tau}} f_\tau(\tau) d\tau \tag{2.33}$$

This allows for the generation of different clutter distributions based on the selection of τ. When the texture parameter is set to $\tau = 1$, equation (2.33) simplifies to:

$$f_{\mathbf{n}}(\mathbf{n}) = \frac{1}{(\pi)^N ||\mathbf{M}||} e^{-\mathbf{n}^H \mathbf{M}^{-1} \mathbf{n}} \tag{2.34}$$

and the clutter is modeled as a complex Gaussian distribution.

In order to generate the K-distributed clutter model, τ is set to be Gamma distributed with shape parameter a and scale parameter b:

$$f_\tau(\tau) = \frac{1}{(b)^a \Gamma(a)} (\tau)^{a-1} e^{\frac{-\tau}{b}} \tag{2.35}$$

Where $\Gamma(\bullet)$ is the Gamma function. The amplitude PDF of the clutter can then be represented as a K distribution [43]:

$$f_{|\mathbf{n}|}(|\mathbf{n}|) = \frac{2|\mathbf{n}|}{b\Gamma(a)} \left(|\mathbf{n}| \sqrt{\frac{1}{2b}} \right)^{a-1} K_{a-1} \left(|\mathbf{n}| \sqrt{\frac{2}{b}} \right) \tag{2.36}$$

Where $K(\bullet)$ is the modified Bessel function of the second kind.

Pareto distributed clutter can be modeled by using an inverse gamma texture with shape parameter a and scale parameter b: [44],[41]:

$$f_{\tau}(\tau) = \frac{1}{(b)^{a}\Gamma(a)} (\tau)^{-(a+1)} e^{\frac{1}{-b\tau}} \tag{2.37}$$

In each of these cases, adjusting the shape and scale parameters for the clutter texture τ also modifies the shape and scale (respectively) of the K and Pareto distributions (as one would expect). As such, these will here on out be referred to in the broader context as the shape and scale parameters for the clutter distribution. All interference samples are assumed to be independent.

2.3.6 Divergence Measurements of Distributions

Within the field of statistics there exist multiple ways to characterize and compare the probability density function of distributions. A single pdf is often characterized by summary statistical measures like the mean, covariance, standard deviations, skewness and kurtosis. These tools give insights into a single distribution but often we want to compare two distributions. Although there are multiple ways to compare two distributions like summary statistic comparison and hypothesis testing methods (e.g. T-test, Z-test, or Kolmogorov-Smirnov test) this section will focus on divergence measurement methods as they are the primary methods utilized in later research.

The core divergence method I will discuss here is the Kullback-Leiber divergence (KLD) [45] and its symmetrical expansion the Jensen-Shannon divergence (JSD) [46]. The KLD is a likelihood based divergence measurement between two distributions. The general form assumes two distributions P and Q.

Typically P represents data and Q represents a reference distribution for comparison. The KLD has the form:

$$D_{KL}(P\|Q) = \int_{-\infty}^{\infty} P(x)\log[\frac{P(x)}{Q(x)}]dx \tag{2.38}$$

The KLD provides a measure of statistical distance by calculating the relative entropy between two probability distributions (P and Q). For discrete distributions within a space X the KLD is:

$$D_{KL}(P||Q) = \sum_{x\in X} P(x)log(\frac{P(x)}{Q(x)}) \tag{2.39}$$

The KLD is not symmetric, ($D_{Kl}(P||Q) \neq D_{Kl}(Q||P)$). The KLD has been used with SIRVs recently as a measure of statistical uniqueness given different distributions. Bomburn *et al.* has explored the use of the KLD with both SIRVs [47] and with the product of a multivariate generalized Gamma and a Uniform distribution [48]. An important principle explored in these papers includes probabilistic independence of SIRVs:

$$p_Z(\mathbf{z}) = p_\tau(\tau)p_X(\mathbf{x}) \tag{2.40}$$

Additionally, a chain rule expansion of the KLD can represent the texture and speckle components separately:

$$\begin{aligned} &D_{KL}(p_Z(\mathbf{z})\|q_Z(\mathbf{z})) = \\ &D_{KL}(p_\tau(\tau)\|q_\tau(\tau)) + D_{KL}(p_X(\mathbf{x})\|q_X(\mathbf{x})) \end{aligned} \tag{2.41}$$

The work in [47] has derivations for closed form KLD for Inverse Gamma and Fisher distributions with additional KLD derivations for Weibull in [49]. The Kullback-Leiber divergence provides a measure of statistical distance by calculating the relative entropy between two probability distributions (P and Q).

The Kullback-Leiber divergence is not symmetric ($D_{Kl}(P||Q) \neq D_{Kl}(Q||P)$). In order to have a symmetrical divergence method utilizing the KLD I will now discuss the JSD. The JSD is symmetric divergence to the average of two distributions based on the KLD method. The JSD is symmetric by calculating the Kullback-Leiber divergence with respect to a new divergence $M = \frac{1}{2}(P + Q)$ which is the average of the original distributions. The Jensen-Shannon divergence is:

$$JSD(P||Q) = \frac{1}{2}D_{KL}(P||M) + \frac{1}{2}D_{KL}(Q||M) \tag{2.42}$$

2.3.7 Adaptive Detectors

Adaptive detectors are a class of detectors that utilize a test statistic that is typically derived from statistical analysis of support samples. This test statistic is compared to a threshold value to determine if a target of interest is present (H_1) or if only noise and interference are present (H_0). Although multiple adaptive detectors are commonly used like the adaptive matched filter (AMF) [50, 51], adaptive covariance estimator (ACE) [52], the adaptive Roa detector [53], and multiple two-stage algorithms [54, 55]; the work in this dissertation utilizes Kelly's generalized likelihood ratio test (GLRT) [56]. The GLRT is an adaptive radar detector for use in the presence of unknown homogeneous Gaussian noise. This detector has been used extensively in the field and has decades of expansive investigation [57, 58, 59, 60, 61, 62]. The GLRT is an extensions of a likelihood ratio test (LRT). This likelihood test is based on the joint probability density functions (PDFs) for the return signals that either contain a target (H_1) or not (H_0). The general from for these joint PDFs is shown in 2.43 [56]:

$$f_i[\mathbf{z}, \mathbf{z}_1, ..., \mathbf{z}_K] = f[\mathbf{z}|_{\mathbf{H}_i}] \prod_{k=1}^{K} f[\mathbf{z}_k], \quad i = 0, 1. \tag{2.43}$$

If the signal consists only of interference (H_0) and is assumed to be zero mean Gaussian and identically distributed as the train samples, then this joint PDF becomes:

$$f_0[\mathbf{z}, \mathbf{z}_1, ..., \mathbf{z}_K] = \left(\frac{1}{\pi^N ||\mathbf{M}||} e^{-trace(\mathbf{M}^{-1}\mathbf{T}_0)} \right)^{K+1} \tag{2.44}$$

$$\mathbf{T}_0 = \left(\frac{1}{K+1} \right) \left(\mathbf{z}\mathbf{z}^H + \sum_{k=1}^{K} \mathbf{z}_k \mathbf{z}_k^H \right) \tag{2.45}$$

where N is the number of pulses in each sample, equivalent to m in (2.25). The interference covariance matrix $\mathbf{M}$ must be estimated from the support samples.

When z contains both a target of interest and interference, the PDF becomes:

$$f_1[\mathbf{z}, \mathbf{z}_1, ..., \mathbf{z}_K] = \left(\frac{1}{\pi^N ||\mathbf{M}||} e^{-trace(\mathbf{M}^{-1}\mathbf{T}_1)} \right)^{K+1}, \tag{2.46}$$

$$\mathbf{T}_1 = \left(\frac{1}{K+1} \right) \left((\mathbf{z} - \alpha\mathbf{p})(\mathbf{z} - \alpha\mathbf{p})^H + \sum_{k=1}^{K} \mathbf{z}_k \mathbf{z}_k^H \right). \tag{2.47}$$

The return signal amplitude α in addition to $\mathbf{M}$ must be determined or estimated for the return signal fo the H_1 case. The true interference covariance matrix is typically estimated as 2.48 though other methods exist:

$$\mathbf{M} \approx \mathbf{S} = \sum_{k=1}^{K} \mathbf{z}_k \mathbf{z}_k^H \tag{2.48}$$

A likelihood ratio test provides a test statistic by comparing the ratio of two likelihood estimations. These likelihood estimations provide a statistical model which maximizes the PDF for some parameter. The final form including the hypothesis cases is:

$$\lambda = \frac{\max_{\mathbf{M},\alpha} f_{\mathbf{z}|\mathbf{H}_1}[\mathbf{z}, \mathbf{z}_1, ..., \mathbf{z}_K]}{\max_{\mathbf{M}} f_{\mathbf{z}|\mathbf{H}_0}[\mathbf{z}, \mathbf{z}_1, ..., \mathbf{z}_K]} \gtrless_{\mathbf{H_0}}^{\mathbf{H_1}} \lambda_0 \tag{2.49}$$

By combining (2.44) and (2.46) with (2.49) and performing the maximization, the final test statistic takes the form:

$$\lambda = \frac{1 + \left(\mathbf{z}^H (\mathbf{M})^{-1} \mathbf{z}\right)}{1 + (\mathbf{z}^H (\mathbf{M})^{-1} \mathbf{z}) - \frac{|\mathbf{p}^H (\mathbf{M})^{-1} \mathbf{z}|^2}{\mathbf{p}^H (\mathbf{M})^{-1} \mathbf{p}}} \tag{2.50}$$

By using the CCM estimation (2.48) and rearranging the test statistic to the form $\eta = \frac{\lambda - 1}{\lambda}$ we get the well known form of the GLRT test statistic from literature:

$$\eta = \frac{|\mathbf{p}^H \mathbf{S}^{-1} \mathbf{z}|^2}{(\mathbf{p}^H \mathbf{S}^{-1} \mathbf{p})[1 + (\mathbf{z}^H \mathbf{S}^{-1} \mathbf{z})]} \tag{2.51}$$

The thresholds is derived from the desired probability of false alarm (P_{fa}) and assumptions regarding the clutter:

$$P_{fa} = P(\lambda > \lambda_0 | \mathbf{H_0}) \tag{2.52}$$

Kelly [57] provides an estimation of the threshold given a target (P_{fa}) for the GLRT detector and a Gaussian assumption:

$$P_{fa} = \frac{1}{\lambda_0^{K-N+1}} \tag{2.53}$$

Considering this is only a function of K, N, and λ_0, the GLRT has been proven to be CFAR with respect to the clutter covariance in the Gaussian case.

2.4 Command and Control and Radar Resource Management

2.4.1 Command and Control

Command and Control is a domain that represents the mechanism for diverse operations to navigate an environment that is filled with uncertainty, complexity, and ever decreasing time to react. In the late 90's, C2 was summarized as establishing a common intent to achieve coordinated action [1]. As technology has advanced the framework for C2 was expanded to include systems that look to collect data, identify

intents from analyzing this data (reasoning), and break down intent into actions (planning) [2].

Over the years there have been multiple models to describe the C2 process with the most common being the observe-orient-decide-act (OODA) model originally presented by Boyd (1996) as an operational framework and guide to activities that encompass C2 [3]. A common framework that coincides with the radar domain and represents additional challenges is the Find, Fix, Track, Target, Engage, Assess (F2T2EA) framework [63]. In 1990, US General John Jumper proposed the C2 framework find-fix-track-target-engage-assess to address the OODA loop in more detail. This expanded framework overlaps well with the goals of radar systems as they are the dominate method to observe and orient in the environment and by applying the find, fix, track, and target tasks. The RAND organization has assessed the viability of AI/ML in C2 to fulfill autonomous needs in the target, engage, and assessment steps [64]. Additionally, AI for decision support has been aligned to C2 with multiple viable tools [65].

2.4.2 Radar Resource Management

Advances in modern radar have led to a range of new capabilities for searching and tracking targets through a space. In particular, there has been extensive research into effectively coordinating multiple beams to minimize the time it takes to search a space and maximize the number of tracks that can be maintained. Multiple beams can be generated in multiple-input and multiple output (MIMO), phased-array, and digital array radars. Coordinated multi-agent search and track tasks are commonly found in drone swarms and satellite constellations. Whether coordinating multiple beams on a single platform or multiple radar systems on independent platforms, that coordination is generally framed as a limited resource optimization problem. This formulation flows naturally from the goals and restrictions of radar platforms. Ideally,

a sensing system provides a full and current understanding of an operational space. However, each platform has a limited power budget, processing time requirements, and movement/steering restrictions. Each limits the amount of space the platform can effectively observe.

The field of RRM has arisen to identify methods to optimize the utilization of multiple radar beams. In general, RRM defines a set of tasks that need to be performed and a number of agents that are available to perform those tasks. In RRM, an agent refers to an available beam, and agent tasks are typically broken out into three categories: 1) search, 2) confirm, and 3) track. A search task directs an agent to observe an area to see if an untracked target exists there. The confirm task directs agents to observe an area where an initial detection occurred to attempt to establish a track. A track task involves having agents observe the next expected location of a tracked target to update its position and velocity information.

A cost value is defined for each available task based on the radar resources required to perform and the impact if not performed. For instance, updating the track of a target far from the available agents generally requires more power and time resources than doing so for closer targets. However, the longer that distant target goes unchecked, the more likely it is that the track will be lost. RRM algorithms seek to optimize the allocation of available resources in order to minimize the overall cost to a scenario. This optimization becomes more challenging as the search space and number of tracks increase.

RRM research largely focuses on mathematical representations of the cost associated with various tasks and in exploring different optimization algorithms. Common optimization schemes applied to this space are the quality of service (QoS) method [66, 67, 68] and methods that apply the Cramer-Rao lower bound (CRLB) [69, 70, 71, 72, 73].

2.5 Radar Search

The radar search function is a task within traditional RRM steps. Search is often also referred to as surveillance, and represents the use of radar resources (energy and time) directed at azimuth and elevation angles with the goal of detecting potential targets of interest. The search function could be a sequence of radar illumination actions in a set pattern or it could be determined at each radar action cycle based off some criteria or policy. Traditionally, the search function is considered along with the track function by an RRM scheduler. In those scheduler based systems the goal is to balance radar resources between updating new tracks and search the radar space for potential new targets. The classic challenge becomes a matter of maintaining as many tacks as possible while still searching the radar space often enough to cover the space. In both cases the illumination action space for the radar is the same as a function of azimuth, elevation, power, and dwell time [74]. The following section will examine three aspects of the search function from RRM: 1) search methods based on a pattern or sequence of illumination to cover a space efficiently, 2) methods to represent the search space to influence intelligent search decisions, and 3) intelligent search method given varying degrees information or representations of the search space.

The first class of search pattern has its foundation in a geometric analysis of sequential illuminations often referred to as a raster scan method. The simplest raster scan will partition the search space into a Cartesian space and then progress in either a row or a column sweep. Typically, the raster scan will include an overlap of the previous illumination in order to reduce the likelihood that a moving target will evade in the scan direction. Once a column or row has been swept, then the pattern will move in the other Cartesian option. This will continue until the full space is scanned, and then it will repeat [74]. If information is available regarding the geometry or intended direction of the target then a modified raster can be developed. The work in [75] explores a

hallway example with a target attempting to pass a space undetected as well as a target within a circle attempting to exit the space. In both cases the authors devise a pattern and bounds to assure that given the target's maximum velocity and the search agents time resource that the target will be found. The second class of pattern based search methods implement a stochastic approach with random or pseudo-random search patterns. In these methods the search agent will illuminate areas based on a distribution (typically uniform or Gaussian) and will search without replacement from a discrete representation of the radar space.

The second aspect of search research and implementation is focused on different ways to represent the radar search space. The default assumption is that all space has equal importance and that there is no a priori knowledge. Many methods will use this assumption but research has shown that if you have a priori knowledge you can improve search efficiency. Additionally, even without initial assumptions, some representation can improve search efficiency with a representation that updates following the results of previous radar search illuminations. For this class of representation it is common to descretize the space and then assign some value or probability to each potential search location. The work in [76] assumes that targets have specific radial velocities and radar cross sections but will appear in the space at different ranges given a known probability distribution. Each possible action the radar can take is modeled with a task quality and utilizes a cumulative probability of detection to drive action selection. Conversely, [77] represents the search space with a probability of target existence and modeled a scenario with environmental generators of targets (i.e., an airport). This information is used to drive search action selection by selecting the action to cover that areas of highest target probability. The work by Chen *et al.* [78] represents a top down view of the radar space (x and y Cartesian) with cells that contain a likelihood value for a hidden target based on the stationary features of the environment. Finally, an entropy based method for search is presented in [79]. They model a

probability of target presence with an entropy of 1 for probability of 0.5. There is no a priori information but as areas are searched (initially randomly), the probability is either increased or decreased, which updates the entropy toward 0. The goal is to maximally reduce the entropy in each time step. [80] represents previous scan data, and multiple beam's range and Doppler information to develop a large multi-dimensional map.

The third aspect will focus on the methods that have been employed to determine radar search actions often with a representation presented in the second aspect. A simple approach when utilizing a space representation is to perform a summation of some utility and identify the location of greatest value of said utility. The work in [81] utilizes the linear solver CPLEX to create sub-regions to then perform raster scans. Another common method is to use Bayesian theory to update the posterior probability distribution given the information during a radar illumination [82]. Following the feature representation presented in [80] the authors use a Hough transform to find the features representing target. The work in [83] utilizes a genetic algorithm with a fitness function focused on expected discovery distance and expected discovery time. Finally the multi-agent challenge for surveillance has precedence using game theoretic concepts like price of anarchy to coordinated multi-platform search coordination [84]. These methods are typically paired with a space representation or some features of the search space. This is due to the fact that if there are no environmental factors or a priori information, then all space should be searched with *equal* importance, and a method like a raster scan or uniform random is optimal.

2.6 Radar Track Confirmation

While detection algorithms are designed to keep the P_{fa} low, false alarms still occur regularly in real radar systems. This is due to a combination of the large number of measurements that are taken and suboptimal performance in the interference rejec-

tion of the detector [85]. If every detection not associated with an existing track was used to establish a new one, then many false alarms could produce a track. This could result in the track list growing rapidly with false tracks, consuming significant radar resources to monitor targets that don't exist. To avoid that scenario, a process called track confirmation and association is performed [85].

Track confirmation is a relatively simple but important task. When a detection is observed, it is first evaluated to see if it should be associated with an existing track. Association is typically done using one of various probabilistic [86, 87] or joint probabilistic [88, 89] association methods. If a detection is not associated with an existing track it is flagged as a potential track. That potential track must then be verified through multiple repeated detections in rapid succession.

Chapter 3

Machine Learning and Game Theory Methods Background

3.1 Introduction

The work in this dissertation focused on scalable multi-agent technologies within the command and control (C2) domain utilizing radar for the initial steps of the OODA loop. The pivotal decide and act steps have historically been completed by human users with varying degrees of computer based assistance. As the modern C2 challenge increases in scale, complexity and agility, additional domains in autonomy have been explored and show great potential. A promising and powerful tool for automation and autonomy in the last several decades is the emergence of AI/ML. The autonomy challenge has demanded methods for multi-agent coordination as realistic environments expand into interconnected challenges where harmful action can emerge from allied systems due to lack of coordination. Additionally, these systems need to scale from single or low system collaboration to large distributed systems. This dissertation utilizes the field of game theory to explore and address the growing need for multi-agent collaboration. In this chapter, I will discuss the fundamentals of machine learning, the ML method of reinforcement learning (RL), concepts of multi-agent RL and meta-cognition, and the field of game theory.

3.2 Machine Learning

The field of machine learning has exploded in both popularity and applications in the last 20 years. This includes application in engineering and radar signal processing. ML is a subset of the larger field of artificial intelligence which is the utilization of machines or computing to replicate the capabilities or task performed through human intelligence. AI encompasses a very large range of methods from state machines, heuristic methods, and mathematical algorithms to modern trillion parameter deep learning models. The following discussion will present and discuss the field of machine learning but focus specifically on the deep learning methods used in this dissertation.

ML is a subset of AI that includes the requirement that the parameters of a model or equation are not predetermined, but instead determined through interaction with data and/or an environment. The modern advancements in computational power has lead to increase innovation in a subset of ML called deep learning (DL). DL is characterized by the use of neural networks as the foundation for a machine learning model. There are multiple model configurations in the domain of DL that are focused on different applications and challenges. Figure 3.1 is a visualization of only a portion of deep learning methods. Specifically these are methods that pertain to the C2 and RMM domains. Additional detail will be given to the methods used in the research in this dissertation, but an overview of these methods can be found in [90, 91, 92].

As shown in Figure 3.1, most ML methods can be classified in four main categories based on the methods used for models to learn parameters as well as the expectations of the data or environment. Most ML literature includes the categories: unsupervised learning, supervised learning, semi-supervised learning, and reinforcement learning. Unsupervised learning is a method of training a model to identify the underlying patterns held within data or in the case of autoencoders to learn a mapping of input infor-

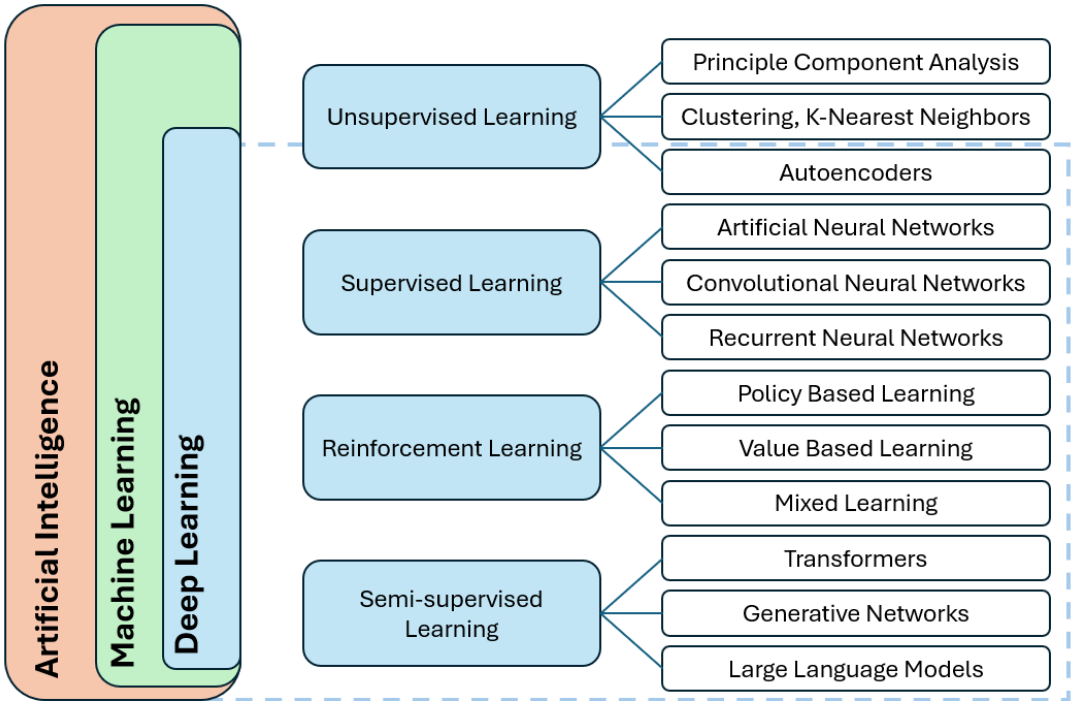


Figure 3.1: Methods in Machine Learning

mation to some output data. Supervised learning is learning with data and label pairs. The label is a truth value corresponding to input data that the model learns to predict. Semi-supervised methods typically contain some form of truth or goodness metric but typically have the model learn some task or behavior with a mixture of truth information and the impacts of model performance. Popular generative models [93] and large language models are examples of semi-supervised learning [94]. Reinforcement Learning is a subset of semi-supervised learning that has a model act on a dynamic environment and learns based on its impact upon the environment [95].

The work in this dissertation utilized supervised learning artificial neural networks (ANN) and reinforcement learning extensively and will be discussed in detail in the following sections. Additional discussion is made regarding multi-agent architectures and methods in machine learning.

3.2.1 Neural Networks and Supervised Learning

The oldest form of DL is also the core building block of modern DL models, the perceptron. The perception was introduced in 1943 by Warren McCulloch and Walter Pitts [96] and is modeled after the biological neuron found within most animal life. This mathematical representation of the neuron, the perceptron, is relatively simple with an example shown in Figure 3.2 and a mathematical representation:

$$\hat{y} = f(w_0 + \sum_{i=1}^{m} w_i x_i) \tag{3.1}$$

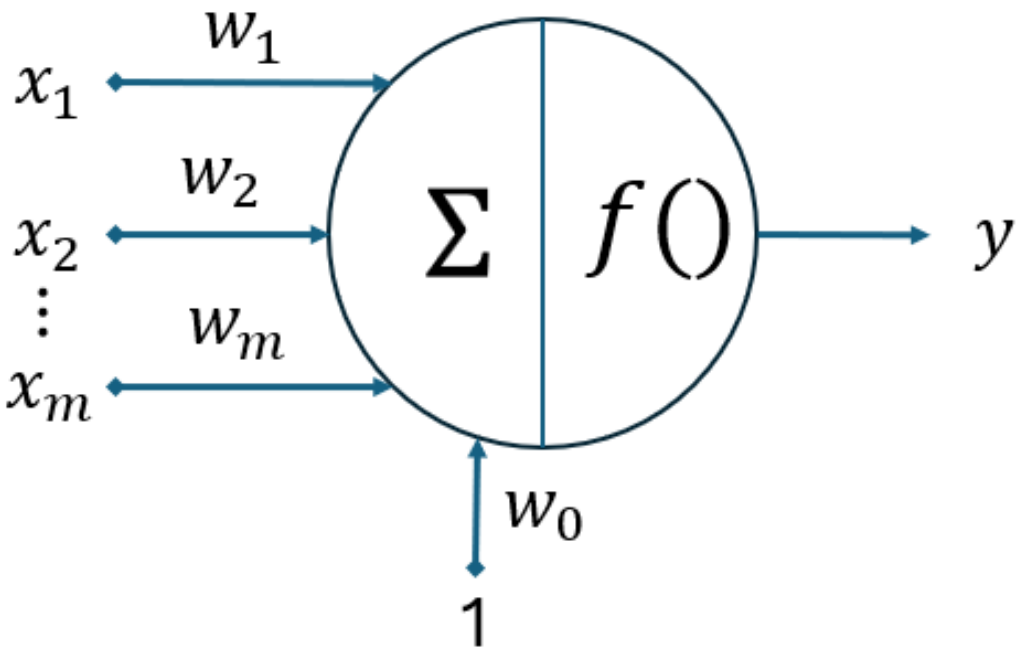


Figure 3.2: Simple Perceptron

A single perceptron can have m inputs that are weighted and summed with a bias weight. This summation is then processed by an activation function. The learned component for a perceptron are the weighting parameters. Activation functions in practice are required to be non-linear and differentiable. The reason for this requirement is due

to the solution to the exclusive or (XOR) problem famously presented by Minsky and Papert in 1969 [97]. This work made the case that a single layer of perceptrons were unable to solve an XOR data set. Even with a non-linear activation function a single-layer continuous boundary of any form cannot separate this data. Admittedly they mentioned the use of multi-layer perceptrons, but at the time they lacked a method for updating weights in a multi-layer perceptron.

A major breakthrough for neural network systems came with the work by Rumelhart in 1986 with the introduction of backpropagation [98]. The weights of the perceptron were updated by using the error from a prediction to change the weights based on their contribution to an incorrect prediction. This took the form of:

$$w_i = w_i - \epsilon(y_j - \hat{y_j})x_i \tag{3.2}$$

Where w_i was the weighting parameter for i inputs of a perceptron, y_j is the truth or label corresponding to data input x_i for j outputs, and $\hat{y_j}$ is the predicted output.

This process works well for a single layer but it presented a challenge for multiple layers. The introduction of backpropagation utilized the chain rule in calculus to propagate the error gradient through layers. This combined with a matrix representation and linear algebra creates an elegant solution to update all weights efficiently that also was computationally efficient for parallel processing units like graphic processing units (GPUs). A full multi-layer perceptron also called a dense neural network or ANN is shown in Figure 3.3.

To better under stand the math for back propagation, first we will define the forward process for each layer:

$$\begin{aligned} Z^{(L)} &= W^{(L)}X^{(L)} \\ A^{(L)} &= f_{a(L)}(Z^{(L)}) \\ C^{(L)} &= f_c(Y^{(L)}, \hat{Y}^{(L)}), \text{for output layer} \end{aligned} \tag{3.3}$$

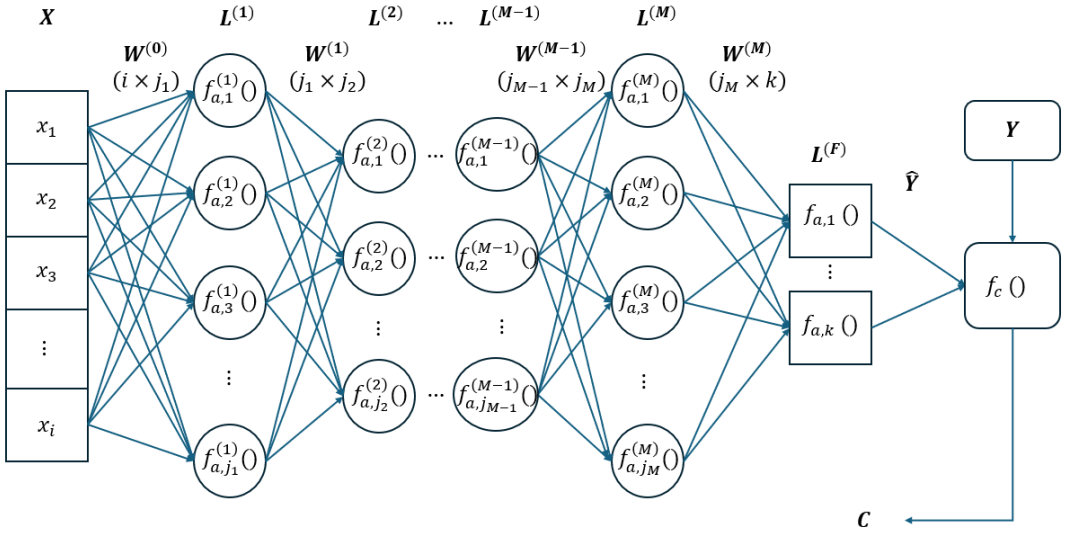


Figure 3.3: Artificial Neural Network

For this matrix representation, $Z^{(L)}$ is the cross product or sum of the product of weights and inputs, $A^{(L)}$ is the activation output of a layer (this is $\hat{Y}^{(L)}$ for the output layer), and $C^{(L)}$ is the cost at a layer. The equation to update the weights of a layer is:

$$W^{(L)} = W^{(L)} - \epsilon \frac{\delta C^{(L)}}{\delta W^{(L)}} \tag{3.4}$$

Effectively, this process assigns gradient associated with each weight given that weight's contribution to the cost from the loss function. In order to determine this back propagation used the chain rule to determine three gradients and uses matrix multiplication to update weights. The three gradients relate to the series of equations in (3.3) and are:

$$
\begin{aligned}
\frac{\delta C^{(L)}}{\delta W^{(L)}} &= \frac{\delta Z^{(L)}}{\delta W^{(L)}} \frac{\delta A^{(L)}}{\delta Z^{(L)}} \frac{\delta C^{(L)}}{\delta A^{(L)}} \\
\frac{\delta C^{(L)}}{\delta A^{(L)}} &= X^{(L)} = A^{(L-1)} \\
\frac{\delta A^{(L)}}{\delta Z^{(L)}} &= f'_{a(L)}(Z^{(L)}) \\
\frac{\delta C^{(L)}}{\delta A^{(L)}} &= f'_c(Y^{(L)}, A^{(L)}) = 2(A^{(L)} - Y), \text{for L2 norm}
\end{aligned}
\tag{3.5}
$$

The elegance of this method is that cost associated from the prediction in the final layer and the loss function can be propagated through any number of layers and the weights can be updated based on their contribution to the error in the prediction. This efficient updating method is the source of the requirement that activation functions f_a and loss/cost function f_c are differentiable.

Now that the form of an ANN is presented the discussion of the function and setting used for model formulation is important. These additional design selections are referred to as hyper-parameters. The first set of hyper parameters to discuss are common activation functions. In order to use back propagation these functions must be differentiable. A common list of activation function and the situations to use them are included in Table 3.1

All of the above are common hidden layer activation functions, but some suffer from issues identified by the community. The sigmoid and tanh functions suffer from the "vanishing gradient problem" which happens when the input become very large or small then the corresponding gradient approaches 0. This cause an issue were learning can either stop or slow down significantly. The standard ReLU can suffer from the "dying ReLU problem" because any negative value will become 0. This means that if a neuron can become negative then the neuron will "die" as the gradient cannot change future weights.

When it comes to output activation function, the list should reflect the problem that is being solved. The most common problems solved by ANNs are either regression or

Activation Function	**Equation**	**Challenges**
Sigmoid	$f_a(x) = \frac{1}{1+e^{-x}}$	vanishing gradient problem
Hyperbolic Tangent (tanh)	$f_a(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	vanishing gradient problem
Rectified Linear Unit (ReLU)	$f_a(x) = \max(0, x)$	dying ReLU problem
Leaky ReLU (LeLU)	$f_a(x) = \begin{cases} x, & x \geq 0 \\ \alpha x, & otherwise \end{cases}$	N/A
Exp. Linear Unit (ELU)	$f_a(x) = \begin{cases} x, & x \geq 0 \\ \alpha(e^x - 1), & otherwise \end{cases}$	N/A

Table 3.1: Commonly Used Hidden Layer Activation Functions

classification problems. A regression problem trains the model to predict continuous values in some range and a classification problem looks to predict if the input data is part of some class. Both the type of problem and the range will often necessitate different output activation functions. Common output function are presented in Table 3.2

Activation Function	**Equation**	**Utilization**
Linear	$f_a(x) = x$	Regression
Sigmoid	$f_a(x) = \frac{1}{1+e^{-x}}$	Binary and Multi-label classification
Softmax	$f_a(x) = \frac{e^{x_i}}{\sum_i^N e^{x_i}}$	classification of N classes

Table 3.2: Commonly Used Output Layer Activation Functions

The next hyper-parameter to discuss is the types of cost function typically utilized for ANNs. It is important to use a loss function that supports success for the challenge the model is training to predict. Some of the functions are useful for regression or classifications and others are hybrid methods or address some derived component of fitness or goodness the model should attain. Common cost/loss functions include:

Cost/Loss Function	**Equation**	**Utilization**
Mean Absolute Error	$f_c(y, \hat{y}) = \frac{1}{n}\sum_i^n \lvert y^i - \hat{y}^i \rvert$	Regression
Mean Squared Error	$f_c(y, \hat{y}) = \frac{1}{n}\sum_i^n (y^i - \hat{y}^i)^2$	Regression
Huber Loss	$f_c(y, \hat{y}) = \begin{cases} \frac{1}{n}\sum_i^n (y^i - \hat{y}^i)^2, \\ \qquad \lvert y^i - \hat{y}^i \rvert \leq \delta \\ \frac{1}{n}\sum_i^n \lvert y^i - \hat{y}^i \rvert, \\ \qquad \delta(\lvert y^i - \hat{y}^i \rvert - \frac{1}{2}\delta) > \delta \end{cases}$	Regression
Binary Cross-Entropy	$f_c(y) = \frac{1}{n}\sum_i^n -y_i log(p_i) + (1 + y_i) log(1 - p_i)$	Classification
Categorical Cross-Entropy	$f_c(y) = \frac{-1}{n}\sum_i^n \sum_j^m y_{ij} log(p_{ij})$	Classification

Table 3.3: Commonly Used Output Layer Cost Functions

The list above are common loss functions, but it is important to understand that significant research is underway regarding custom cost functions as well as utilizing distribution based approaches like the Kullbeck-Leibler divergence and Cosine Similarity.

A common problem with ANNs is a result called overfitting. Overfitting is when

a model learns a dataset so well that it learns the predict the noise within the data instead of the underlying function or problem. To avoid overfitting, the methods of drop out and normalization are common. Dropout is a method to ignore updating some percentage of weights at random. While normalization is smoothing effect that is applied to weights in the same layer after updating. Additionally, it is very important when training and testing a model to maintain independent training, validation, and testing subsets of the data/label pairs. At the end of some training interval, called an epoch, the current model is tested against the validation training set which is used to measure the convergence of the model instead of the of the training set. It is very easy for a model tested on its training set to overfit.

The final hyperparameters can have significant impacts on learning for ANN models. The learning rate ϵ controls how much parameters change for each update. Typically, learning rate is a small number ($\epsilon \approx [0.01, 0.00001]$) as the cost for uncontrolled learning is usually more problematic than needed longer training cycles. Typically training is done in batches where some batch of data is trained before each parameter update. The final consideration for training a model is the stopping criteria. This is the method to determine when to stop training. The simplest method is a set number of epochs before stopping. This is not very effective as often a model can over train if run too long or can plateau if not trained long enough. One common stopping criteria is based on evaluating an epoch for some success criteria. Another method is called patience. Patience will look at the change in performance for a model and save the best model over time. Then, it will stop if a new iteration has not surpassed the best saved model after some time period.

3.2.2 Reinforcement Learning

Reinforcement Learning (RL) is a method within machine learning that utilizes feedback from an environment to train agents to make decisions. This process has an

agent make an action selection a_t based on an observation w_t of the environment state s_t. In turn, those actions will impact the environment and a reward r_t is returned to the agent providing context regarding the impact of the action selection. The agent will then utilize machine learning methods to update action selection for the future. Finally, the environment will provide the updated state and the process will iterate. A visual representation of this process is shown in Figure 3.4 [99].

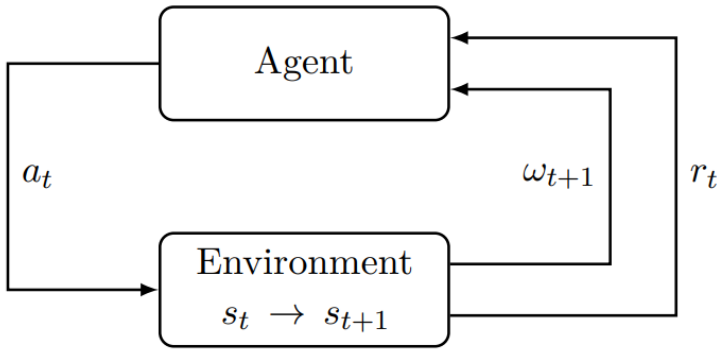


Figure 3.4: Agent-Environment Interaction

A core aspect of RL is that the agent uses trial and error experience of the environment as opposed to a priori knowledge of the environment state. The goal of RL is for the agent to learn good behavior or strategy for action selection.

The majority of RL methods assume a Markov decision process which is modeled by a 5-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$.

- $\mathcal{S}$ is the state of the environment.
- $\mathcal{A}$ is the action space, which is the output for the agent. The action state space represents all possible actions the agent can make toward the environment.
- $\mathcal{T}$ is the state transition function, which describes the probability that the environment changes from state s to state s' after the cognitive agent takes action a. Mathematically, this is written as

$$\mathcal{T}(s_t, a_t, s_{t+1}) = \mathbb{P}(s_{t+1}|s_t, a_t) \tag{3.6}$$

- $\mathcal{R}$ is the reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, R_{max}]$ of the environment, which is a reflection of the objectives of the environment.
- γ is a discount factor which decays the impact of past rewards in cumulative reward calculations.

There are various RL methods but each will focus on some combination of *policy*, *value*, and *models* of environments. The work in this dissertation focuses on model-free RL algorithms for an extensive overview of model-based RL see [100]. A break down of model-free RL algorithms can be seen in Figure 3.5 from [101].

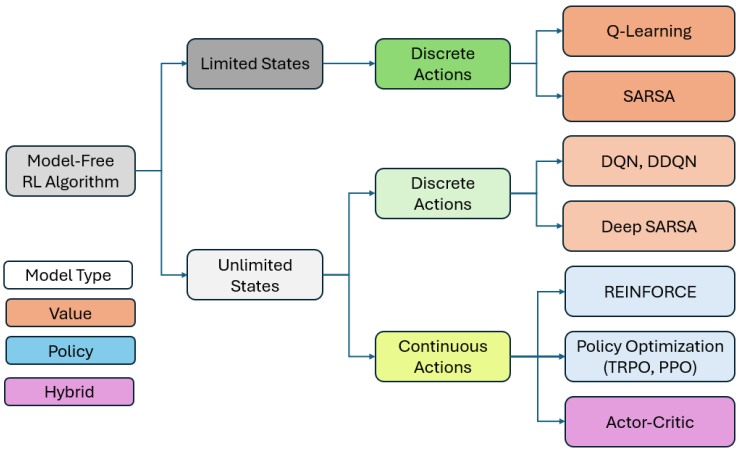


Figure 3.5: Model-free RL Algorithms

Policy represents how an agent selects actions and is represented as $\pi(s) : \mathcal{S} \rightarrow \mathcal{A}$ for direct action selection or $\pi(s, a) : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ representing a probability of action selection given a state. The value function is the expected return given a policy selection. This is expressed as:

$$V^{\pi}(s) = \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s, \pi] \tag{3.7}$$

An additional equation of interest used across RL is the concept of the Q-value function $\mathcal{Q}^{\pi}(s,a) : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ which represents the expected return for an agent that takes a specific action a given a state s while it follows a policy π. This is expressed as:

$$\begin{aligned} \mathcal{Q}^{\pi}(s,a) &= \mathbb{E}[G_t | s_t = s, a_t = a, \pi] \\ G_t &= \sum_{k=0}^{\infty} \gamma^k r_{t+k} \end{aligned} \tag{3.8}$$

The final concept important to RL is that of the advantage function, which represents the expected advantage of action selection given value and Q-value. This takes the form:

$$\mathcal{A}^{\pi}(s,a) = \mathcal{Q}^{\pi}(s,a) - V^{\pi}(s) \tag{3.9}$$

There are multiple versions of RL that utilize policy, value, Q-value, and advantage to encourage learning. The following is a detailed discussion of two methods appropriate to signal processing and radar game models. [99]

The first method is called the double deep Q-network (DDQN). In a deep Q-network the estimated Q-function is represented as:

$$\mathcal{Q}^{*}(s,a) = \mathbb{E}_{s' \sim \epsilon}[r + \gamma \max_{a'} \mathcal{Q}^{*}(s',a') | s,a] \tag{3.10}$$

where $\mathcal{Q}^{*}(s,a)$ is the estimated Q-value given a state-action pair. A basic DQN will train a deep neural network (DNN) to be the agent policy by using the Q function for back propagation and neural update. In practice the DQN will overestimate the Q value of the system which leads to unstable policies and reduced performance. The DDQN has a primary DQN (θ) that learns for same period τ and then transfers the neural network parameters to another target DQN (θ') with identical structure. At

each τ period the parameters in the target network are updated by a combination of its historical parameters and the primary DQN:

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta' \tag{3.11}$$

The updated target model makes action selection while the primary model learns every time step before the next update step. This balanced structure reduces unstable behavior as the policy DQN is updated slower and has a smoothed understanding of the system's Q-value. [102]

The second method to discuss is the actor-critic model (AC). The AC model has two DNN's with the actor providing policy and the critic estimating the value (or Q-value). The actor utilizes a gradient policy method from the REINFORCE RL method. The core concept is that policy methods define an objective function:

$$J(\theta) = \mathbb{E}[\sum_{t=0}^{T-1} r_{t+1}|\pi_\theta] \tag{3.12}$$

The gradient of the REINFORCE objective function is calculated to update the policy parameters:

$$\bigtriangledown_\theta J(\theta) = \mathbb{E}_\tau[\sum_{t=0}^{T-1} \bigtriangledown_\theta \log \pi_\theta(a_t|s_t) G_t] \tag{3.13}$$

The critic in the AC method decomposes the gradient of the objective function and applies the expected value calculation to the G_t function 3.8 as a substitute for the Q-function. A recent update to AC models called A3C updated the Q-function to the advantage function in equation 3.9:

$$\bigtriangledown_\theta J(\theta) = \mathbb{E}_\tau[\sum_{t=0}^{T-1} \bigtriangledown_\theta \log \pi_\theta(a_t|s_t)] A(s_t, a_t) \tag{3.14}$$

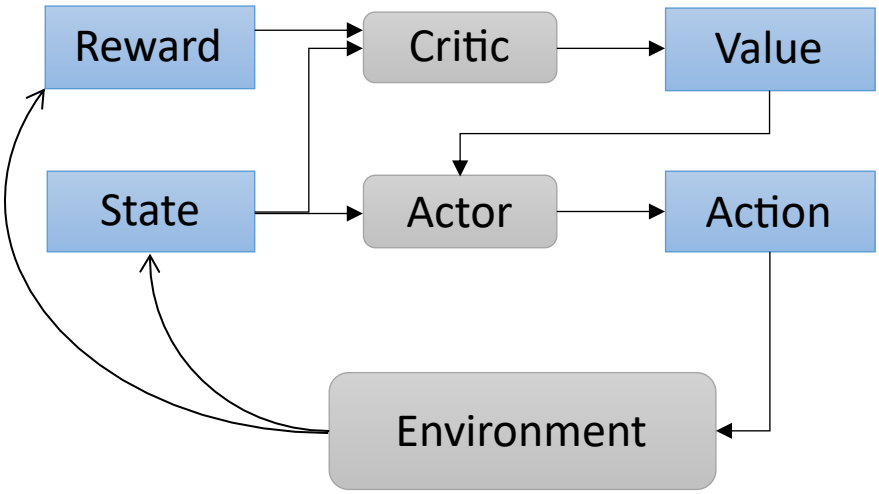


Figure 3.6: Actor-Critic Architecture

A depiction of the AC methods is depicted in Figure 3.6 [103, 99].

The work in this dissertation uses Asynchronous Advantage Actor Critic (A3C), of which there are two independent sets of parameters: Critic (ϕ) and Actor (θ) parameters both optimizing a single policy (π). The Critic's parameters are optimized to create a function ($V^{\pi}(\omega|\phi)$) representing the value of a state (ω) when following the learned policy. The quality of a state ($Q(\omega, \alpha)$) is the discounted reward of the state-action (α) pair in question given future states achievable in a single episode (T). As the state space is large, we estimate the Q function by calculating the discounted sum of rewards when following an execution of the current policy through a single episode:

$$Q^{\pi}(\omega, \alpha) = \sum_{t}[R(\omega_t|\alpha_t) * \gamma^{t-1}] \tag{3.15}$$

where $t \in T$ is a state-action pair in the ordered set of episode state-action pairs (T), $R(\omega_t|\alpha_t)$ is the reward for the state-action pair, and $\gamma \in [0, 1]$ is the discount factor for future rewards received. As the value function needs to be biased towards the quality of a state to recognize future potential rewards, we optimize the critic's pa-

rameters by the mean squared error between the value function and estimated quality of a state, making the loss function as follows:

$$\mathcal{L}(\phi) = \frac{1}{|T|} \sum_{t} [V^{\pi}(\omega|\phi) - Q^{\pi}(\omega, \alpha)]^2 \tag{3.16}$$

The Actor is responsible for the action made given a state, known as the action distribution ($\pi_\theta(\alpha|\omega)$). As actor-critic is a temporal-difference algorithm, we want to optimize the Actor's parameters by discovering the advantage of taking one action over another. We define the advantage function as the difference between the estimated quality of a state-action pair following the policy and the estimated value of a state, shown below:

$$A^{\pi}(\omega, \alpha) = Q^{\pi}(\omega, \alpha) - V^{\pi}(\omega|\phi) \tag{3.17}$$

To bias the actor model towards a better action, the output of the actor is treated as the mean of the action distribution. To update the parameters of the model, the loss function is defined as the normalized log-probability of the taken action and the advantage of the action taken:

$$\mathcal{L}(\theta) = -log(\pi_\theta(\alpha|\omega)) * A^{\pi}(\omega, \alpha) \tag{3.18}$$

This is derived from Proximal Policy Optimization (PPO) [104] and allows the model to compare the temporal-difference between action distributions. This encourages convergence to better actions when optimizing both loss functions independently. RL methods have the advantage of being very flexible and only relying on observations with limited information and the mechanism design of reward functions. This comes at the cost that it can be difficult to optimize the observations and the rewards to encourage learning desirable strategies.

A modern update to RL includes the use of multi-Agent RL (MARL) methods to train agents that learn multiple related objectives with the same model [18, 19]. The field of meta-cognition utilizes a hierarchy of single task agents and a selection mechanism to determine the appropriate task given the environmental state. Meta-cognition has the advantage of being modular and allowing agents to be replaced with other methods [105]. Nevertheless, MARL has shown that when tasks are related, then learned behavior is more consistent and multi-task and multi-agent coordination can be learned in a single model. RL methods have the advantage of being very flexible and only relying on observations with limited information and the mechanism design of reward functions. This comes at the cost that it can be difficult to optimize the observations and the rewards to encourage learning desirable strategies.

3.3 Game Theory: Representations

Game theory (GT) studies the interactions between agents in an environment or game. These interactions are logically and mathematically modeled to describe and optimize an agent's decisions. The games are represented by rules and objectives to achieve a specific desired end state. These rules and objectives are used to limit the complexity of games which allows for mathematical modeling. The strategic form game is the most common form studied in GT. This game utilizes a payoff function, u, which assigns a utility value to actions which are then used to determine a course of action. Strategic form games allow for adversarial, collaborative, and multi-objective interactions between agents.

$$\begin{aligned} &\mathcal{G} = (\mathcal{K}, (\mathcal{S}_k)_{k\in\mathcal{K}}, (u_k)_{k\in\mathcal{K}}) \\ &\text{where } u_k : \mathcal{S}_1 \times \cdots \times \mathcal{S}_k \longrightarrow \mathbb{R} \\ &(s_1, \ldots, s_k) \longmapsto u_k(s) \end{aligned} \tag{3.19}$$

This will often include the notation: $s_{-k} = (s_1, \ldots, s_{k-1}, s_{k+1}, \ldots, s_K)$, which

represents the strategies taken by all agents other than agent k [106].

Depending on the variations to a strategic form game, the goal for agents in this game is to either minimize cost or maximize utility for themselves. The "solution" to the game is a mixed concept called optimality which relates to a stable equilibrium or dominant strategy. The primary optimality for strategic form games is the Nash equilibrium (NE) which is a point in the game where no player can gain greater utility by changing their strategy as long as all other players maintain their current strategies [107]. It is possible for a game to have more than a single NE as any local maxima in the utility function can result in an equilibrium state. The strategy methods section will breakdown common methods to encourage solutions toward the global NE or transform the game function to enable optima variations. Strategic form games can be adversarial, collaborative, or mixed objective focused. These variations are represented when the s_k and s_{-k} vectors map to unique utility functions for different agents.

Extensive form games are a strict representation of adversarial and collaborative games, similar to the standard form, except they require perfect information and utilize a tree structure to represent all strategy options with leaves terminating in utility values. The extensive form is more complete than the strategic form and is intuitive for full game solutions. The main drawback in using the extensive form is that it requires full information of all strategies and the resulting utility, which makes it infeasible for many applications. [106]

Where strategic form games can be either adversarial or collaborative, the coalition form games are only collaborative and assume that all agents are willing to work together to maximize the utility of the system. Following the assumption that all agents are willing to work together, coalition form games focus on solutions that form coalitions (subsets) of agents. This game form is concerned with the size and amount of coalitions, when they should divide or sustain membership, and the optimal configuration to maximize utility. Inside each coalition, the

players may choose strategies, similar to a strategic-form game, but the goal is to analyze the formation of the coalitions given communication between the players. Coalition form games consists of $\mathcal{K} = \{1, \ldots, K\}$ players with a total of $2^k = \{\emptyset, \{1\}, \{2\}, \ldots, \{K\}, \{1,2\}, \ldots, \{1, \ldots, K\}\}$ possible coalitions ($\mathcal{C}$). Each of these coalitions has a value function (v) that represents the utility of the players within a coalition. The general framework for coalition form games is:

$$\begin{aligned} v : 2^{\mathcal{K}} &\longrightarrow \mathbb{R}^{\mathcal{K}} \\ \mathcal{C} &\longmapsto v(\mathcal{C}) \end{aligned} \tag{3.20}$$

Strategic form collaborative games focus on a balancing of utility or mechanisms to discourage disruptive actions. Coalition form games, instead, focus on maintaining stable coalitions as the environment changes, not just resource allocation and individual utility maximization. [108]

This work focuses primarily on collaborative, strategic form games. This assumes that agents are willing to work together and form coalitions with the intention of achieving an optimal coalition configuration to maximize utility. This form of game can converge to a point where future agent decisions or actions will not increase utility for the agents. When this condition happens it is called a solution. However, for complex games, such as those in the radar domain, a solution may not exist or may be unsolvable in a closed form. Collaborative GT applications include: information sharing for informed agent strategy; game solution design; and established methods like consensus, price of anarchy, and regret management.

3.4 Game Theory: Agent Methods

As discussed above, the primary solution for strategic form games is related to optimality (π). The original representation of optimality is the NE. Both local and global NE can exist in complex games. In order to drive solutions that align with the

global NE, multiple methods have been proposed which modify the NE interpretation. These methods include: Pareto optimality, social optimality, price of anarchy, and mechanism design techniques.

The Pareto optimality is the point where it is not possible to increase the utility of one player without decreasing that of at least one other. This optimality encourages stability in a game where a strategy cycle among players is present [109]. The social optimality looks to maximize the utility in the system as a whole. It represents a solution for the combination of player strategies that results in the greatest total utility regardless of individual agent results. The price of anarchy is a measure of performance loss given selfish actions among agents. Specifically, it is a ratio of the utility from the current strategy divided by the worst possible NE in the system. Price of anarchy is represented as:

$$PoA = \frac{\max_{s \in S} \sum_{k \in \mathcal{K}} u_k(s)}{\min_{s \in S^{NE}} \sum_{k \in \mathcal{K}} u_k(s)} \tag{3.21}$$

Price of anarchy can be used to weight the utility of game states due to the social welfare cost of strategies [110]. The last solution mechanism related to the NE is called mechanism design. Mechanism design looks to improve system efficiency and stabilize to desirable states by applying a transformation to the utility function and then obtaining the NE for the transformed game [111]. More broadly, mechanism design, also referred to as Reverse GT, can alter the design of games in such a way as desirable solutions are both possible and stabilize to a NE.

A second class of strategic game solutions keeps the game and NE the same but modifies the solution concept. The two most common methods to accomplish this are the correlative equilibrium and the Nash bargaining solution (NBS). The correlated equilibrium method is a randomized assignment of action recommendations to agents where all agents have no benefit to deviate. This is represented as a joint probability

distribution calculated across all strategy combinations. The main purpose of correlated equilibrium is to use probability distributions to recommend actions leading to NE that incentivizes agents to accept[112] [106]. Parameters of the correlated equilibrium include using either public signals providing recommendations to all agents or private signals providing recommendations only to a subset of agents. A variation called coarse correlated equilibrium assumes players decide whether to commit to the recommendation before receiving a public or private signal.

The NBS uses a communication mechanism to encourage bargaining between agents to explore the options with the maximum product of utility among strategies from a disagreement point. NBS requires the utility function $\mathcal{U}$ to be a convex set as it provides a solution of:

$$\begin{aligned} &\max_{(u_1,u_2)\in\mathcal{U}} (u_1-\lambda_1)(u_2-\lambda_2) \\ &\text{subject to: } u_1 \geq \lambda_1, u_2 \geq \lambda_2 \end{aligned} \tag{3.22}$$

where (λ_1, λ_2) is a "disagreement point" within $\mathcal{U}$. Originally NBS applied to only two player games but it can be expanded to multiple players. This expanded form still needs a convex utility function and is often updated to a coalition form solution as the NBS is improved by considering coalitions[113].

The simplest form of GT, or lack of GT, is the best policy method. This method has all agents implemented separately and then they select the action from their policy which best maximizes their utility. A variation is the collaborative best policy method in which agents share information. After information is shared the individual agents are free to follow their best action policy, but usually incorporate the new information. Both of these methods are often included with other learning agents as they represent a strategy instead of an agent themselves.

The best response dynamic (BRD) is a type of evolutionary game theory that updates strategy rules, where player's strategies in the next round are determined by their

best responses to some subset of the population. Two common variations of BRD are the Gauss-Seidel method [114] and the Lloyd-Max algorithm [115]. The Gauss-Seidel method is a sequential application of each agent using BRD in turn until convergence. This method originated from iteratively solving a linear system:

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$$
$$\text{given time } t\text{: } x_1(t+1) = \frac{y_1 - a_{12}x_2(t+1)}{a_{11}} \tag{3.23}$$
$$\text{then: } x_2(t+1) = \frac{y_2 - a_{21}x_2(t+1)}{a_{22}}$$

where x_k is the action of an agent, y_k are utility observations by agents, and a_{ij} are assumed to be known conditions of the game. This assumption is a strong restriction as it requires understanding of the game mechanics. The Lloyd-Max algorithm is similar to k-means clustering as it partitions a space into cells and aligns the cell centroids to the optimal regions in the space. When applied to the utility function this iterative method can lead to convergence among multiple agents.

The regret management algorithm utilizes an evaluation of regret to apply a probability of action selection. Each agent assigns a regret value for a selected action $a_{k,n}$ by calculating the difference between always doing this action versus following the current strategy. This is expressed as:

$$r_{k,a_k,n}(t+1) = \frac{1}{t}\sum_{t'=1}^{t} u_k(a_{k,n}, a_{-k}(t')) - u_k(a_k(t'), a_{-k}(t')) \tag{3.24}$$

Regret management assumes that the agents have a method to estimate their own utility. As the algorithm iterates, it builds an action distribution $\pi_k(t) = (\pi_{k,a_{k,1}}, \ldots, \pi_{k,a_{Nk}})$. This is updated by the regret calculations where positive regret implies that the given action should have higher probability. Similarly, negative regret

implies that the action should have a lower probability. The system stabilizes its action distribution when regret approaches zero. [116]

The final category of agents includes the collaborative methods consensus and merge-split. The consensus algorithm utilizes full information sharing among collaborative agents to reach an optimal consensus strategy. It operates on the assumption that players will share full information and that a strategy exists that optimizes collaborative utility. A simple version of consensus utilizes a weighting matrix β. This matrix represents the confidence each agent has in its collaborative partners. This is expressed as:

$$a_k(t+1) = a_k(t) + \sum_{j \in \mathcal{A}_k} \beta_{k,j}(a_j(t) - a_k(t)) \tag{3.25}$$

A simplified version allows the convergence matrix to collapse the relationship with the weighting matrix (C) with the form $a(t+1) = Ca(t)$ and reflects a dynamic weighting of collaborative action intent influencing an agent's action selection. [117] There are multiple versions of consensus implementation, but a core characteristic is information is shared and action policy is modified in light of the additional information.

3.5 Collaborative Game Theoretic Examples for RRM

A specific class of GT utilized in this dissertation are collaborative methods for coordinating agents. These examples are all characterized by game designs and GT methods that encourage agents to meet shared goals. The applications discussion will focus on those with minimal information sharing, direct collaboration, and examples with integrated collaborative mechanisms. The work in [118] specifically explored producing a collaborative outcome while using non-cooperative approaches. They used reward and punishment structure within a zero-sum distributed Goore game

model. The Goore game model asks agents to make action selection without any shared information or observation of other agent actions; then the game rewards or punishes individuals based on a collaborative objective and their part in the result. The agents attach a probability to action based on rewards. It was shown that the agents will settle to a collaborative NE if one exists. Song et. al. utilize a two step process to structurally enable collaborative search with multiple UAVs. In their scenario, multiple UAVs will perform an optimized domain search pattern until a target is detected. Then the group identifies the closest searcher and they engage in a local search algorithm based on a matrix of action options and their estimate of target motion to hunt the target. The remaining members continue the search for other targets. Together the framework leverages collaborative search and strategic finding methods for improved performance over heuristics and other pure strategy methods. [119]

This next class of collaborative search examples have individual agents that optimize personal utility but include a structure or hierarchy that causes a collaborative system response. The authors in [120] present a search game with different regions with underlying receiver operating characteristics (ROC) representing the trade-off of detection rate and probability of false alarm. There are a set of searchers and an intelligent planning agent. As the searchers explore the region with random sweeps the planner estimates the ROC in the map. The centralized planner then orchestrates the search patterns of the search agents in order to improve overall detection rates. This single planner creates a collaborative search result without controlling direct actions, but by influencing the underlying search agents. The work in [13] investigated GT approaches to a joint beamforming and power allocation problem. It initially tests selfish agents with no collaboration and compare it to a leader-follower consensus modification with a Stackelberg game. The Stackelberg game included a surveillance agent as a leader which sets an interference cost. The follower agents then select actions while leveraging the interference cost. Then the leader selects its action to optimize any gaps

in the follower's actions. This results in a mechanism to discourage interference with an intelligent optimization of the remaining search space to be covered.

The last two examples of collaborative search use integrated GT approaches to orchestrate collaborative action among players. The consensus algorithm is utilized in [14] with a modification to address the exponential growth issue for all collaborative agents analyzing and optimizing with all other agents. They implement a local neighbor optimization which modifies each agent's utility function to account for the action of its (n) neighbors. The neighbor list is relative to proximity and changes throughout the game. They showed that this local optimization causes a positive global collaborative response. The final example [121] utilizes a particle swarm optimization to address collaboration by multiple agents. Each particle is treated as a possible strategy for the agents. The particle swarm algorithm then causes the particles to concentrate around winning strategies. This grouping drives action selection probability. These collaborative search methods demonstrate the breadth of games, agents, and GT methods that can be viable in the domain of search and evade.

Chapter 4

Multi-Agent Track Confirmation

4.1 Introduction

This chapter will focus on the exploration of multi-agent methods to address the command and control (C2) challenge of confirming highly evasive targets with limited radar resources. This challenge is modeled as a search and evade scenario for a specific radar resource management (RRM) task. Traditionally, the challenge of track confirmation is addressed by having a radar platform illuminate a potential detection again. The logic is that it is increasingly unlikely that a false alarm will happen in the same range bins in the case of repeated illuminations with the same or very similar elevation, azimuth, and power settings. This assumption holds for radar platforms with much greater resources (time budget and computational speeds) than the potential motion of a target of interest. Essentially, if we assume that a subsequent illumination happens quickly enough that a moving target will not have moved enough to be missed. Additionally, even if the target has moved, if it moved in a simple linear or constant heading then it can be associated with the original detection and a track can be established. In these scenarios the RRM task of track confirmation is a trivial effort of repeated illumination. The work in this chapter explores the situations where this assumption does not hold and a target is evasive while the platform has reduced radar

resources. In this scenario, a repeat illumination is unlikely to find the target, instead this becomes a search and evade challenge where the platform is trying to accomplish low history tracking and space reduction strategies.

The reasoning for presenting this scenario and exploring this space is inspired by the C2 reality, that multiple UAVs with reduced resources and computational capacity will be engaging with other smaller and highly evasive UAVs. In this scenario, the targets are more evasive and the platforms have less time to attempt to search and find targets. This creates an interesting and underexplored domain in RRM that can benefit from both AI/ML tools and includes the potential for multi-agent collaboration. This scenario is very difficult for any single low-resource radar platform to effectively confirm another evasive target. But given that this C2 scenario will have a swarm of collaborative platforms, this work explores multi-agent strategies for success. When combined together, this track confirmation challenge becomes a challenge for low information collaboration of multiple radar systems.

An additional C2 based need that I choose to address with the methods explored is the desire for potential human-machine pairing. This drives an engineering need for systems to be modular in design such that a radar task could be conducted using the AI/ML methods explored in this work, traditional radar methods, or a human user. Effectively, this pushes a design philosophy toward hierarchical architectures with each component focused on a specific role and well-defined input and output requirements. Therefore, I avoided methods that integrate multi-objective or multi-agent action selection that are determined with internal ML mechanisms. The reasoning for avoiding these methods also supports the C2 need for explainability and trust in systems.

The reality of the C2 domain is that the performance of a method is only one aspect of the adoption of new technology. When the impact of a method has high cost and the capacity for false alarms or false confidence is low, then there exists a high bar to entry for adoption. Therefore, in this chapter I approach the exploration

of methods with the goal that AI/ML methods are either explainable or have focused action spaces so that their behavior has a tighter scope and erroneous behavior is easier to isolate and analyze. Practically, this means that the methods explored will focus on specific singular objectives, and thereby avoid monolithic structures, that looks to solve multiple objectives using internal integration.

The final constraint driven by C2 operations is a practical aspect of potential software methods. In many cases, C2 operations need software to be integrated on embedded systems and it needs to operate in real time with the other systems in the operation. This incentivizes the method to have a smaller computational footprint. For AI/ML methods, this means focusing on lower parameter counts and efficient feed-forward run times. It is important to note that AI/ML methods are often computationally expensive during training, but the feed-forward or off-line implementation has lower computational requirements. As a result, much of the work in this section will implement simple ANNs and efficient game theoretic methods.

As with any engineering challenge, there is a trade-off for adhering to these constraints. The performance of integrated multi-agent learning referenced in section 3.2.2 is promising but its internally integrated nature is often at odds with the modularity constraint. Similarly, many advanced AI/ML techniques utilize transformers the translate input states into difficult to explain latent spaces. The use of latent space based methods provide enhanced performance and flexibility, but has to be used with the understanding that they increase the difficulty of explainability. Research is underway to bridge the gap of latent space explainability [122], but as of this work it is not mature enough to be broadly considered explainable. Finally, many promising advanced ML methods require very large parameter counts. This imposes a challenge as training these large models requires super computing access and has large computational requirements even in off-line or feed-forward modes.

This chapter will follow my research over the last two years with a conference

publication, journal publication, and additional testing. This work was accomplished with a philosophy that focused on testing a low assumption baseline implementation and then increase to fidelity of the simulation to approach realism. Additionally, this iterative process resulted in changes and improvements to the methods proposed. Although the work was completed in three separate time frames and tested to completeness for publication, there is significant overlap in the simulation and methods. In order to streamline the presentation of this work the following sections will discuss the problem space, simulation designs, heuristic evasive target behavior, implementation of track confirmation agent and multi-agent collaboration, results, and discussion of all three iterations together with insight into the differences and the reasons for the changes.

The initial work was published in the proceedings for the International RadarCon 2023 in [20]. The next improvement was an expansion upon this baseline concept and implementation. The simulation is updated with a medium fidelity radar simulation that implements detection clutter ridges with reduced probability of detection as Doppler frequency and radial velocity approach zero and well as the impact of false alarms. This work was published in an expanded journal in IET Radar, Sonar, & Navigation in January of 2025 [21].

4.2 Problem Space - Mechanism Design and Radar Challenges

The main problem space for this work is the search and evade type of game in GT with the underlying environment and agent observation modeled as a radar challenge. In this work, the term game will be used in the context of a game mechanism from GT. This includes the physical representation of the environment, the agents whose actions impact the environment, the methods and restrictions to agent perception and actions within the environment, and the rules governing gaining utility in the game and ending the scenario.

This search and evade game is a strategic challenge that has increasing difficulty for agents when information is limited. In this game, the challenge for the agents is the find and confirm the position of an evasive agent. In this case the intelligence, surveillance, and reconnaissance (ISR) agents will undergo multiple steps in the RRM domain. This challenge is critical when a highly evasive target of interest has conflicting objectives and engage in a adversarial exchange. This will entail surveillance of a space, identifying a target of interest, and establishing a track confirmation. In this game mechanism the ISR agents will survey objective points, one of which the target is attempting to reach.

RRM is characterized by three common tasks: search, confirm, and track. The search task surveys a radar space that could contain potential targets of interest. The goal of the search task is to find those targets as efficiently as possible. Track confirmation is a required task given that the process of detection could result in false alarms. Therefore, a potential detection needs additional detections to confirm that it was a target and not a false alarm due to noise or interference. Once a track is establish following confirmation, the the tracking task predicts the future position of a target and utilizes resources to maintain the track with updated detections. Track confirmation has increasing complications with a highly evasive target as it may no longer be in the same location as a detection when confirmation attempts are made. If a radar system has limited resources and a target is highly evasive then confirmation becomes a challenge similar to tracking but with little to no historical information about the potential target. It is this challenge that inspired the original game design to explore low information track confirmation methods and the potential of multi-agent collaboration using game theoretic principles.

Traditional RL techniques have been used with deep learning models to create agents that learn strategies from the environment [99]. These learning agents are used to support traditional controls and signal processing tasks in multi-agent and

distributed systems. Subsequently, there is a corresponding need for effective coordination of these agents. The field of GT has historically provided this coordination by modeling a game representation of an environment with structured rules and solution mechanisms. GT focuses on game adaptation to physical environments, methods to analyze adversarial behaviors, and coordinating responses from collaborative agents. The combination of learning agents and GT methods provides the tools to develop cognitive systems that explore dynamic solutions for complex radar challenges. This chapter explores the impact of collaborative GT methods applied to RL search agents within a search and evade challenge as a strategic form game [106] of a command and control challenge.

In modern command and control there are multiple evasive agents, search agents, and naive actors. This results in a complex environment and establishes a growing need for cognitive systems and multi-agent coordination for successful operation. There are core challenges to adapting GT concepts to the RL domain. The following research describes my development of a realistic radar search and evade game followed with exploring collaborative GT methods and how to apply them to deep RL agents with a comparison to a Gaussian baseline.

4.3 C2 Scenario Design

This section will explain my design of a baseline implementation of a two agent collaborative approach to track confirmation in a search and evade game. This work will focus on the track confirmation task, but also includes a heuristic search task utilized by all agents. The mode selection implementation for each agent is a naive heuristic that will always select the track confirmation task when a potential target is detected and will select the search task if the target is lost. In addition to exploration in the track confirmation task the two agents will use a game theory based operations method to improve coordinated utility. This work was published in the paper [20]

and presented at the international radar conference 2023 in Sydney Australia, and was included in the best papers award list.

In order to limit the complexity for this initial research, I decided to limit the observable area and potential locations for objectives and targets to a 10 km by 10 km two-dimensional space. The view of the observable space is a top down view and subsequently limits the radar actions to range and azimuth. I designed the underlying simulation to scale for future research such that the simulation can generate any number of radar agents, targets of interest, and objective points. This was developed using the gym environment [123] in Python and included classes to represent important features and actors. For this initial research I made the design decision to include two radar platforms, one target, and four objective points. I made this decision to limit the amount of computation and time needed to test the simulation as well as to limit the amount of variable actors, so that I could have greater confidence in results and isolate variance while still receiving generalized results. My colleague, Timothy Sharp, coded a render function that could visualize videos of scenarios for debugging issues and for presentation. A frame of one of these videos and representation of the game is shown in Figure 4.1.

For this initial implementation the two radar platforms were positioned at -3,400 meters in the y-dimension and along the x-axis at 3,333 and 6,666 meters. The four objective points were uniform randomly initialized within the 10km by 10km game area but not within 100 meters of an edge. The game was design to test the track confirmation task and therefore began each scenario with a target detection and each radar agent beginning to attempt subsequent detections to confirm the target. The target internally selected one of the objective points as its goal and was initialized at a uniform random position offset by a 3 km radial to its goal. The inspiration for this design is based off of a C2 scenario and challenge in which targets are more evasive and UAV swarm platforms have reduced radar resources. Additionally, many

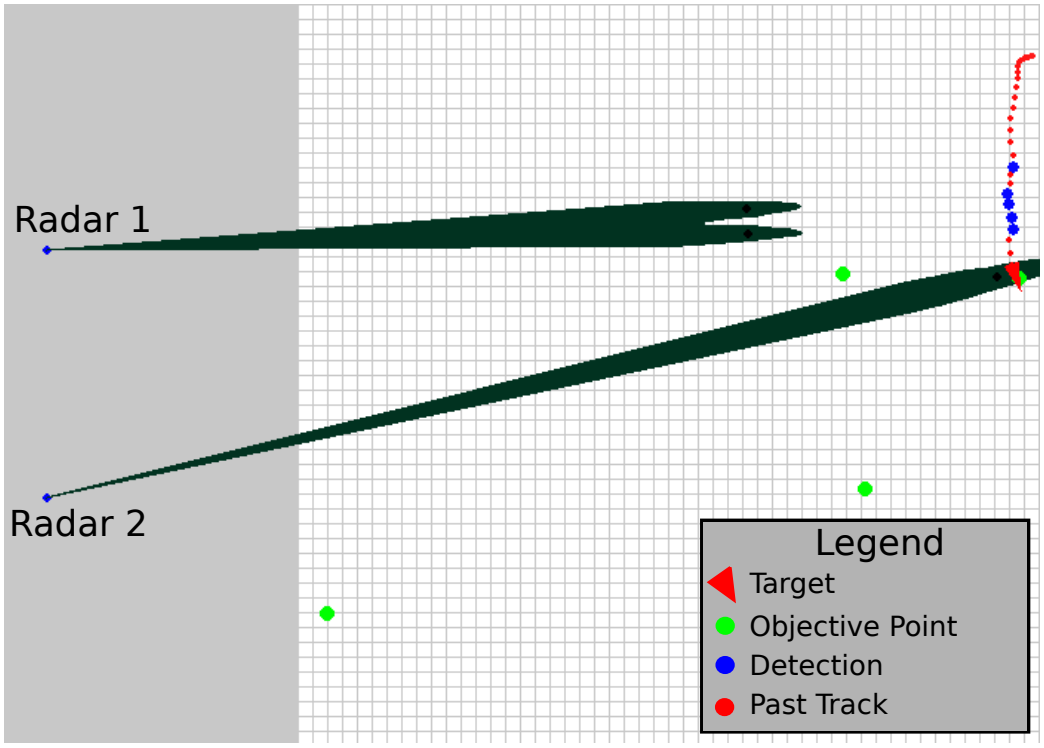


Figure 4.1: Render of the Game

tracking scenarios in the literature will have a target that is following a path that is often linear or constant velocity/acceleration. Regardless, the target is not heading toward anything in particular. But the C2 challenge I wanted to represent here was one in which a target is of interest because it is a threat to a specific strategic area. This was the inspiration for the objective points. These points act as a generic representation of a geographical area, that the surveillance agents do not want hostile targets to reach. The target is trying to move within a set radius of one of the objective points before the radar agents can establish a track confirmation. In this case the objective points represent known information for the surveillance radar agents and would need to play a role in the scenario design. In the scenario, the targets are initialized with a single objective of multiple possible objectives to add additional complexity and not present a trivial challenge where the observation agents would have information of exactly where the target is heading.

For each game scenario the target of interest was detected and the radar platforms were given information as if they had just received a detection. The target has probabilistic logic to move toward its goal objective point. The radar platforms kept an internal detection window of size $m = 10$ that would remove the t_{-m} at each time step and slide all window locations forward and then fill the t_0 location with a 1 for a successful detection and a 0 for no detection. This window was used to calculate one of the end conditions to the scenario and was utilized for mode switching. If the $sum(win) > 6$, then a track is confirmed, and the game is considered a win for the radar platforms. The other terminal condition is if the target maneuvers to within 100 meters of its goal objective, then the scenario is a loss for the radar platforms.

The simulation for detection by the radar platforms utilizes the radar range equation to estimate the probability of detection. This work was an initial exploration of the agent track confirmation methods and game theoretic solutions and focus less on the fidelity of the radar. The back end radar system estimated the signal to noise ratio given a constant noise power and the expected return power assuming a digital phase array. The specifics for the simulation are included in Appendix A, Section A.2 because while the method was not particularly novel, it was important to have a realistic action-state-observation cycle to test the C2 scenario.

The following discussion will present the scenario for the expanded research presented in the IET journal paper [21]. This work included minor parameter updates to the scenario and used the medium fidelity radar simulation with adaptive detection. (See Appendix Section A.3 For the updated game representation). I included a free to use background graphic from the US National Park Service depicting Crater Lake.

The objective point generation, the windowing method for success criteria, and the loss criteria of the target approaching objective point are the same as presented in the initial exploration. This new expansion instead focuses on additional tasks as well as the impact of a clutter ridge and false alarms. The evasive target logic is the same as

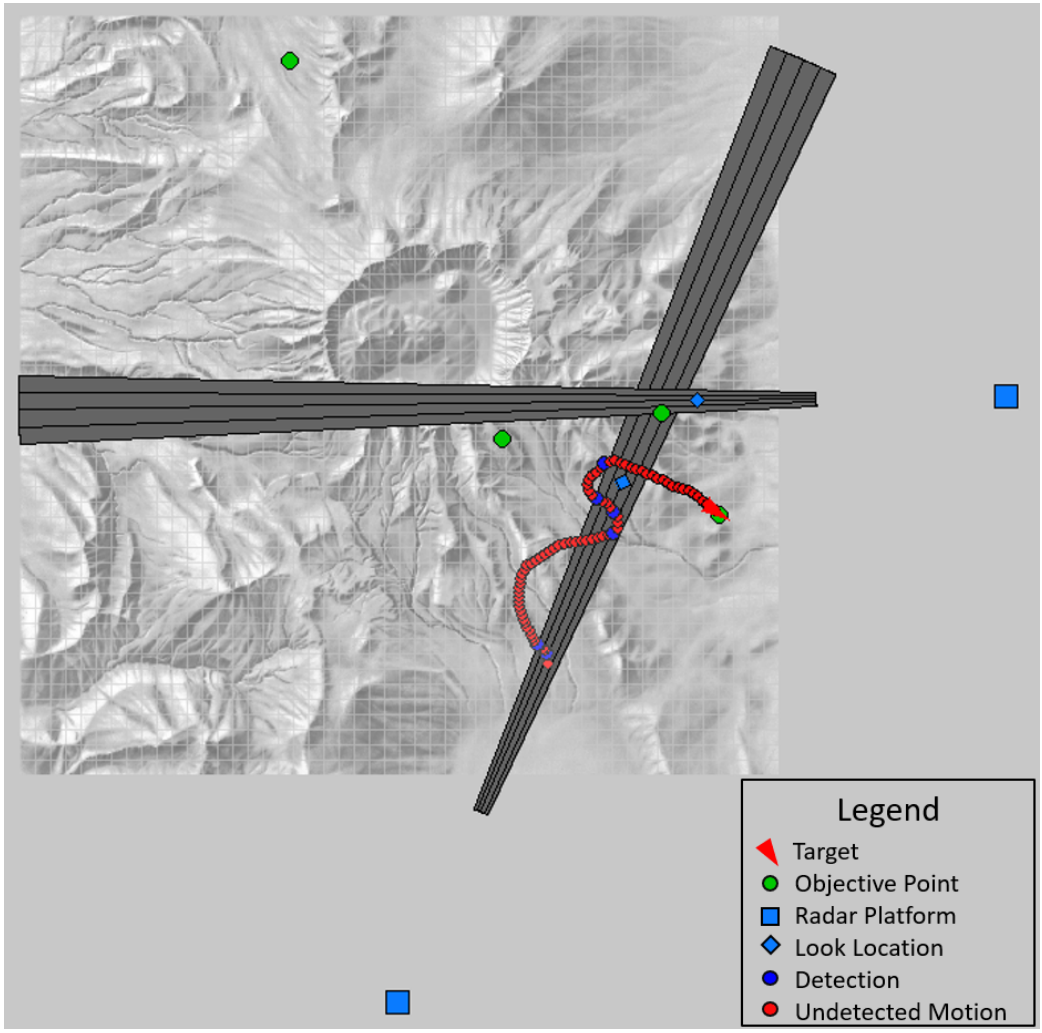


Figure 4.2: Render of the Game.

Background: US National Park Service, Bill von Allmen, Crater Lake

www.shadedreliefarchive.com/Crater-Lake.html

was presented for the initial exploration. The evasive target logic is still effective and valid for an evasive target, but is now able to impact the trackers effectively.

The majority of the map is very similar to the initial work. One update I made to create smoother motion was to implement the updates on the target motion at a faster time scale than the radar agent action space. This expanded research utilized a

radar action rate that was a fourth of the rate used in the initial work. The new time scale ($\Delta t_{target} = \frac{1}{4}$) means that velocity and angle updates were calculated more often but were also updated at a fraction of their normal value. This had a small effect on evasiveness and motion was smoother, but mostly provided rendered video that looked less jumpy.

This expanded scenario and game simulation also included significant updates in the fidelity of the radar platform detection and false alarm capabilities. This increase in fidelity was necessary to test the viability of the methods proposed in this chapter. But the increased fidelity was an implementation of existing methods in the literature as opposed to a contribution of new novelty. Therefore the radar details and specifics are included in appendix a section A.3 (GLRT Dynamic Simulation). Some of the discussion in this chapter and during conclusions are built upon the false alarm and higher fidelity clutter included in this version of the simulation. The reasoning for this radar simulation update was two fold. First, the new implementation processes an adaptive filter detection that conducts a two-hypothesis detection test. This means that there are false alarms, and the parameters of the detector were set so that false alarms could happen frequently enough to affect agent success. Given that the reaction of a track confirmation agent to false alarms is a real problem in the domain it is important to include and analyze. Second, the updated clutter profile included a clutter ridge which results in lower detection rates when the radial velocity of a target approaches zero. This is a realistic maneuver in a C2 environment and was part of the original logic for the target maneuverability. The fact that these evasive targets have a tendency to move orthogonal to an illuminating platform both increases the difficulty and complexity of the challenge but also highlights the importance of the collaborative coordination.

The final testing was identical to the journal expansion scenario. The only change was with the radar detection methods processed in the backend of the simulation. This testing and results utilized a range-Doppler processing and cell averaging CFAR

detector. This process is more realistic then the previous two methods and is described in Appendix A, Section A.4. Given the similarity to the journal expansion, this method will not be referenced until the results discussion.

4.3.1 Evasive Target Design

I designed the targets with a probabilistic heuristic that determined their motion behavior. The original design was more basic then the final implementation as my colleagues, Timothy Sharp and Joseph Karch, worked on a truncated normal improvement that resulted in both more realistic and interesting target behavior. Both the initial exploration and the journal expansion work used the same evasive target logic. The methods presented here fully explain their design and parameter tuning that was inspired by my original design with standard and evasive modes. There are two behavioral modes for a target which is based on environmental results from the radar agent(s) action selection in a given time step. The first behavioral mode is when the target is not illuminated by a radar agent and it is progressing toward its target objective point. The target retains a heading angle and a velocity. This information is used to update the corresponding position for each time step. The angle is updated with a random angle offset given as:

$$\theta_t = \theta_{t-1} + (\Delta_{\theta,max} \times N(x)) \tag{4.1}$$

where θ is the heading of the target, i is the time step, $\Delta_{\theta,max}$ is the maximum allowable change in angle (here set to 30^o), and $N(x)$ is the truncated normal distribution:

$$N(x) = \begin{cases} \frac{1}{\sigma}\frac{Z\left(\frac{x-\mu}{\sigma}\right)}{\Phi\left(\frac{1-\mu}{\sigma}\right)-\Phi\left(\frac{-1-\mu}{\sigma}\right)}, & -1 \leq x \leq 1 \\ 0, & otherwise, \end{cases} \tag{4.2}$$

where $Z(\bullet)$ is the PDF and $\Phi(\bullet)$ the CDF of the standard normal distribution with zero mean and unit variance. For this mode the mean (μ) and standard deviation (σ) dynamically reduce as the target approaches the objective point:

$$\mu = \min\left(1, \max\left(-1, \frac{\theta_{obj}}{\Delta_{\theta,max}}\right)\right) \tag{4.3}$$

$$\sigma = \frac{\min\left(100, \frac{R_{T,obj}+5}{200}\right)}{\Delta_{\theta,max}} \tag{4.4}$$

θ_{obj} is the angle between the current target heading and its target objective, $R_{T,obj}$ is the distance between the target and objective, $\min(\bullet)$ refers to the minimum and $\max(\bullet)$ refers to the maximum operators. This behavior encourages the target to progress toward its objective point but retains some randomness in its path to confuse and avoid detection. This modification to the original pseudo-random behavior was a key improvement by Sharp and Karch as it stopped the target from circling a goal or moving in random directions as it was close to the goal. This improvement means that as the target got closer to the goal, it was less random and operated in a logical and consistent behavior.

The velocity is updated at each target time step. The velocity has a simple update given in equations 4.5 and 4.6.

$$a = \begin{cases} U(0, a_{max}), & V_{t-1} < \frac{V_{max}}{2} \\ U(0, -a_{max}), & V_{t-1} > \frac{V_{max}}{2} \end{cases} \tag{4.5}$$

$$V_t = V_{t-1} + a \cdot t \tag{4.6}$$

In these equations a is the acceleration, V is the velocity, t represents time, and $U(\cdot)$ represents a uniformly distributed random value between the internal arguments. For this work $V_{max} = 300$ m/s.

The second behavioral mode I designed happens when the target is illuminated. The logic for this behavior utilizes the realistic clutter response where the probability of detection reduces as the radial velocity of a target approached 0. What this means from the target's perspective is that when the target is illuminated we assume it will know that a source may have detected it and the angle from the illuminating source. If possible, the target can turn or veer to travel perpendicular to the the angle of illumination. When the target does so, it will move into the clutter ridge and reduce its effective probability of detection. Given the basic radar simulation and detection method used in this initial work, this aspect of evasion does not impact these early tests, but I do implement this aspect with the expanded simulation methods. An additional advantage to moving perpendicular for the target is that the target is more likely to leave the area of a searching radar beam if the radar tries to confirm a detection by looking at the same location. This is a common method and still served as an intelligent aspect of target evasion for the initial research.

Given the above logic, I designed it so that if a radar agent directs a beam in the direction of the target with a detectable transmit power then the target knows it is illuminated and the incident angle of illumination detection, θ_{det}. Within this mode the target will take evasive movements in order to avoid detection. The goal for the target is to accelerate and to veer into a perpendicular direction of the angle of illumination. The logic is to move into the 0 Doppler clutter ridge and reduce P_d for the radar that illuminated it. The following discusses the specifics of the evasive behavior for the illuminated response mode.

In my original design the target had a static probability to veer or maintain its current heading when illuminated, but this proved to create situation where the target was too evasive for the radars to win, but was veering often enough that it did not reach the goal objective. This resulted in excessive game times, and so my colleagues Sharp and Karch implemented a dynamic probability to veer. This probability reduced as the

target approaches the objective point. The closer the target is to the objective point, the more likely it is to increase velocity and continue heading toward the objective point. This probability is:

$$P_{veer} = 0.5 - \frac{\tan^{-1}(\frac{R_{T,obj}-1000}{200})}{\pi} \tag{4.7}$$

A random variable ($x_{veer} = U(0,1)$) is generated and compared to result of P_{veer} given the range from the target to the objective, $R_{T,obj}$. The subsequent behavior is represented as:

$$V_t = \begin{cases} V_{t-1} + a_{max}, & x_{veer} < P_{veer} \\ \text{standard mode, (4.5-4.6)} & x_{veer} > P_{veer} \end{cases} \tag{4.8}$$

The new target heading is calculated by comparing the target's current heading, θ_f and illumination detection angle, θ_{det}. Figure 4.3 depicts the behavior for determining the target's new action heading. This heading angle update is calculated as:

$$\theta_t = \begin{cases} \text{evasive mode, (4.10),} & x_{veer} < P_{veer} \\ \text{standard mode, (4.5-4.6)} & x_{veer} > P_{veer} \end{cases} \tag{4.9}$$

Once in evasive mode, the target determines a target heading as a mean of the tangential angle to the illumination source with a uniform distribution with an offset of $30°$, $\theta_{tangent} = U(\theta_{Truth,tangent} - 30, \theta_{Truth,tangent} + 30)$ and $\theta_{Truth,tangent} = \theta_{det} \pm 90°$. Then on each time step the heading angle updates to either the target tangent heading or a max angle change ($25°$), whichever is lower. This has the following behavior:

$$\theta_t = \theta_{t-1} + max((\theta_{tangent} - \theta_t), \theta_{max} \tag{4.10}$$

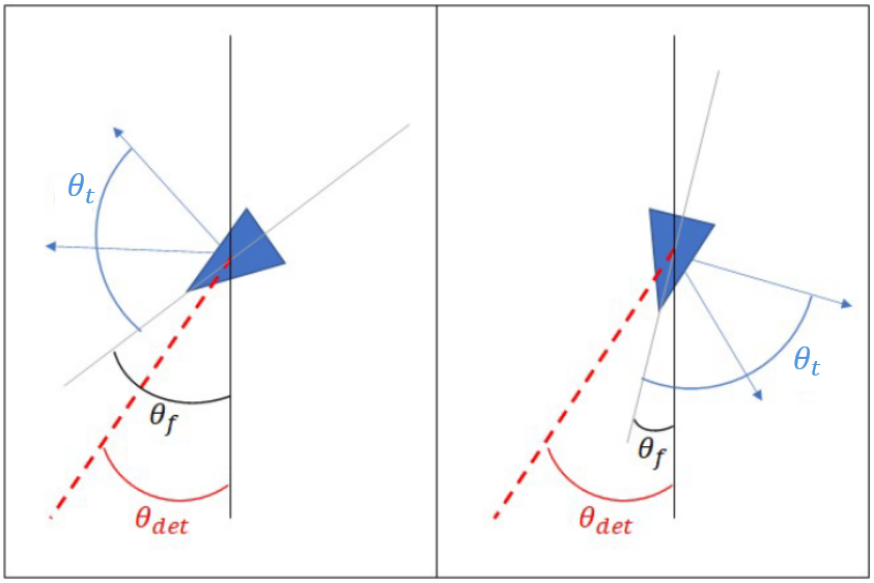


Figure 4.3: Target Heading for Evasive Behavior

4.4 Track Confirmation Agents

Each detector is assigned an agent with a policy to determine the action location for beam forming. After each time step an observation of the game state is passed to each agent, consisting of the data shown in Table 4.1. It is an important distinction that this is the maximum observable state that an agent method could use, but each agent may use a subset of this information for its personal policy decision.

This observation state was used by the agent to decide the next look location, which is then processed into beam splitting, transmit power, and azimuth. During a scenario the agents are initialized in the track confirmation mode. If a target is not detected in some time then the operations agent will switch to a search mode. The process to switch to the search mode is based on consecutive missed detections (when the number of missed detections exceeds the parameter (n_{TL})). The next look location is determined based on which mode the agent is in: search or track.

If the radar agent does not have a detection in a set number of turns, then they will

Name	Symbol	Description and Bounds
Detection History Flag	F_D	1 if a detection is made, decreasing linearly to 0 after n_{TL} turns [0,1]
Detection Location	(x_d, y_d)	Coordinate pair of the last successful detection [0,10000]
Radial Velocity	(Vx_r, Vy_r)	Coordinate pair of the target radial velocity to agent [-300,300]
Platform Location	(x_p, y_p)	Coordinate pair of the radar agent location [-3000,13000]
Detection Flag	F_t	Boolean flag if previous action resulted in a detection [0,1]
Previous Action	(A_{x-1}, A_{y-1})	Coordinate pair of the previous action [0,10000]
Objective Locations	$(x_{obj1}, y_{obj1}, \ldots, x_{objn}, y_{objn},)$	Four Coordinate pairs of objective points [0,10000]

Table 4.1: State Array

move to a search mode in order to recover the target for confirmation. The number of turns until the track is lost is represented by η_{TL}. Regardless of the confirmation agent, the same search method is implemented during this recovery phase. The search mode is a simple Gaussian track that utilizes the a priori knowledge that the target is interested in one of the objective points. The search method is implemented with a Gaussian action selection with a mean at the center of one of the objective points chosen with a uniform random selection probability. The action method is defines as:

$$\begin{aligned} &(Ax_{search}, Ay_{search}) = N(\mu_{obj}, (800, 800)) \\ &\mu_{obj} = \text{uniform selection among objective points.} \end{aligned} \tag{4.11}$$

When a radar agent makes a detection in this search recovery mode then it will begin track confirmation and the track window is reinitialized.

4.4.1 Naive Gaussian

The first track confirmation method used in this work is treated as our baseline and is a naive Gaussian confirmation method. The main differences are recorded here and are based on radar parameter changes for the higher-fidelity implementation. This confirmation method only utilizes a subset of the observable state. Specifically, the naive Gaussian only utilizes the detection location pair, (x_d, y_d). The detection location is used as the mean of a Gaussian selection for the next action. The naive Gaussian action selection is:

$$(Ax_{guass}, Ay_{gauss}) = N(\mu_{det}, (\frac{V_{max}}{3}, \frac{V_{max}}{3})) \tag{4.12}$$

The standard deviation is based on the expected maximum velocity of a target of interest. For this work $V_{max} = 300$ as this approaches the sound barrier velocity. The max velocity is divided by 3 for the standard deviation to represent the assumption that the target is most likely to maneuver within 3 standard deviations and be limited by the max velocity expectation.

4.4.2 Gaussian Velocity Predictor

The second method implemented in this work is a heuristic Gaussian confirmation method that utilizes detection history to estimate a linear velocity, which is used to bias a Gaussian action selection. This method was not used in the initial iteration, but was tested for the journal expansion and the range-Doppler processed versions. The reason for this track method came from exploring tracking methods like the Kalman filter family. Ideally, a Kalman filter, like the unscented KF, should be considered

for a tracking task. This is due to the common use and Bayesian based confidence to predict consistent motion. Even though the target is evasive, there are likely to be periods where it's motion is conducive to a motion prediction method. The problem with KF variants during testing was that they need a history of detections to converge and begin providing accurate prediction. In this scenario a detection history will be very low. This inspired the method presented here as it is a simple linear prediction. My idea was to gain some motion prediction benefit, even though a low detection history would be provided. The information from the observable state utilized by the Gaussian velocity predictor is a subset but includes historic information from the state. This method computes a linear velocity estimation and requires at least two detections to begin making velocity estimations. For the first time step of confirmation attempt, the method will use the naive Gaussian method in section 4.4.1.

Initially and when the predictor has a successful prediction, the detection location $(x_{d,t0}, y_{d,t0})$, the current time step, t_0, and previous estimated velocity (Vx_{t0}, Vy_{t0}) are stored. At each time step the predictor will use the current detection location, (x_d, y_d), the current time step, t and detection flag, F_t. Then at each time step the linear velocity vector is estimated using:

$$Vx, Vy = \begin{cases} (Vx_{t0}, Vy_{t0})\eta_{vel}^{t-t_0} & F_t = False \\ \frac{(x_d, y_d) - (x_{d,t0}, y_{d,t0})}{t - t_0} & F_t = True \end{cases} \tag{4.13}$$

If the previous time step did not have a successful detection then the velocity vector is the same as the previously stored estimate but with a time based decay ($\eta_{vel} = .9$). If the difference between t and t_0 is greater then an uncertainty limit (3 for this work) then the process will reset to the initialized mode which does a standard naive Gaussian method (section 4.4.1). This velocity estimate is then used to determine the mean and standard deviation for the Gaussian selection method as:

$$
\begin{aligned}
\mu_{gaussP} &= (x_d, y_d) + (Vx, Vy) \\
\sigma_{gaussP} &= max(\|((Vx, Vy))\|, \frac{V_{max}}{3}) \\
(Ax_{gaussP}, Ay_{gaussP}) &= N(\mu_{gaussP}, \sigma_{gaussP})
\end{aligned}
\tag{4.14}
$$

This method still utilizes a Gaussian selection for the final step in action selection given the uncertainty of the detection range resolution and that the target will violate the assumption of constant velocity and linear motion.

4.4.3 RL Agent Design and Training

For part of this exploration, I wanted to explore RL as a method for action selection for the track confirmation task. While completing this research, I designed the observable state and the initial reward structure with my colleagues, Dr. Stringer and Bryan Lavender. During the design process I approved changes and discussed ideas, but they worked with me to code the models, update the state and reward, and conduct the training and testing for the final models.

The final track confirmation method was an RL agent trained in a modified version of the full game scenario. The discussion of the RL agent in this section will focus on the environment design, the observable state, the action expectation, the RL design structure, and the training and testing results. The main motivation for looking into RL as a potential tracker is that I built a C2 scenario and have a simulation that can be used as a training environment. Additionally, this track confirmation challenge has an evasive target and a low information state. The radar must establish a track with multiple detections when both false alarms and the variable probability of detection result in actions that do not always succeed. False alarms affect any method that overly trusts a "detection" as it could lead the agent astray. The motivation here was to explore if RL could generalize a low information tracking strategy.

The modified environment for the training of the RL agent uses the game mech-

anism described in section 4.3, but is truncated to encourage learning the patterns a confirmation tracker needs in order to effectively implement action selection. The RL environment has only a single agent and four objective points present with an initialized state that has detected the target. The end conditions of the game are if the agent establishes a track using the window method or is if they lose the track. There is no search mode implementation or secondary track confirmation due to recovery during search. This decision, to only have a single agent, was motivated by the modular design philosophy. This meant that these agents are trained in a solo environment, and then the future collaborative methods are applied to them the same as any other track confirmation method.

The observable state, ω, utilized as an input to the RL agent, is the full observable state presented in Table 4.1 with a modification. The observable state is modified such that all values except the detection history flag (F_D) and the detection flag (F_t) are scaled so that they range from -1 to 1 for radial velocity (Vx_r, Vy_r) and 0 to 1 for all other values.

For the RL model, an A3C algorithm with PPO optimization was selected. The primary motivation for using the A3C algorithm with the PPO optimization is due to the background literature review from Section 3.2.2. Included in that section is discussion about the appropriate types of algorithms to be used in different scenarios and in this case we have a continuous state and a continuous action space. Additionally, this C2 challenge includes a confidence-based disconnect between an action and its impact on the environment. This means that even if an action given a state would result in a reward in one scenario the exact same action-state pair may not in another. This is the reality of the radar domain that probability of detection and the potential of false alarms will mean that this action-state mapping is stochastic. Therefore, the A3C algorithm provides an interesting balance between optimizing policy as well as estimating a value function (in this case the advantage function). This estimation with

deep neural networks is important because a generalized solution is necessary, given the uncertainty of an action-state pair and its impact on the environment. Both the actor and the critic are three-layer deep neural networks. The action output $(Ax_{RL,offset}, Ay_{RL,offset})$, a sample from the actor policy ($\pi_\theta(\alpha|\omega)$), is the normalized offset from the prior detection point (x_{last}, y_{last}) and is scaled then summed to determine the global action selection (Ax_{RL}, Ay_{RL}) for the game:

$$\begin{aligned}(Ax_{RL,offset}, Ay_{RL,offset}) &= \pi_\theta(\alpha|\omega) \\ (Ax_{RL}, Ay_{RL}) &= (x_d, y_d) + w_{RL}(Ax_{RL,offset}, Ay_{RL,offset})\end{aligned} \tag{4.15}$$

Various action scaling factors were tested, so it will be treated here as one of the hyperparameters. During this process, I and Mr. Lavender explored multiple variations of the RL model. This exploration included utilizing a subset of the state, multiple reward functions, and a transfer learning approach. The state exploration showed that the detection flag, detection location, and previous action were the most important input features for the RL agents. A total of 12 RL agents were trained between the two years of research and the three simulation implementations. One agent was trained with an internal recurrent neural network, but it had issues training quickly and due to the uncertain nature of the challenge it tended to converge on behavior that observed the center of the map instead of attempting to track or move based on the previous detection location. Multiple agents were trained with sub sets of the state. The first had a state that only included five features: $[F_D, x_d, y_d, A_{x-1}, A_y - 1]$. During investigations of trained models weights and their activation contributions, I found that these 5 where the most important for action selection. The second medium sized state had seven features: $[F_D, x_d, y_d, V_{xr}, V_{yr}, A_{x-1}, A_{y-1}]$. This version included the relative velocity information with the reasoning that that information could inform the general direction a target was heading and improve action selection. Both of these state variations resulted in models that converged with a maximum reward within 5% of the best models that used the full state.

A major area of exploration was with the detection mechanism during training. Originally the detection mechanism (range equation, GLRT with synthetic samples, and range-Doppler processing with cell averaging CFAR) were used, but they resulted in difficulties for the models. The main issue I found during RL exploration and training was that if the model had inconsistencies in their actions and the impact on the environment then the policy had issues converging. The best models used illumination as the method for determining successful action and reward. Illumination occurred when a target is within a detector's 3 dB beam width and under the cutoff range in Table A.4. A detection was processed using the radar simulation (range equation, GLRT, or range-Doppler and cell averaging CFAR) and was subject to the clutter profile. This choice was made so that the RL agent learns motion patterns for the target and is not punished by the realistic P_D drop as Doppler frequency/radial velocity approaches zero. Although the clutter impact is realistic, it was deemed not to be relevant to initial training of the RL agents. In the early phases of training, the RL agent needs to learn the basic motion characteristics of targets. When the clutter response was included during RL training, the agent often made an action selection that held the target in its beam but was not rewarded. This confused the agent and made rewards tied to aspects outside the agent's control.

The final area of RL exploration was with multiple reward functions. The first reward functions mirrored the testing scenario with a small reward based on the detections in the sliding window increasing, followed by a large reward for track confirmation and a large punishment for losing the target completely. The agents were trained in games with one detector, one target, and four objectives. Two primary variations of reward function were developed and tested when training RL agents for this game. The first established a two stage reward. One small reward was provided for each time step in the game of $+1$ for a detection and -1 for a missed detection. A larger reward of ± 50 was administered upon reaching one of a set of win/loss conditions. Each

training game was considered won if four out of the last ten measurements resulted in a detection, while a loss occurred if either the target reached its objective or the agent had five consecutive time steps without a detection. The intent of this reward function was to teach the agent both the importance of getting detections and of quickly establishing tracks. This resulted in learning but the model seemed to get a medium reward (a few detections in the window) and then stopped exploring to increase the reward. Further investigation showed that this was due to reward sparsity. The loss condition was much more common than the win condition and the agent would settle to get negative reward with slight improvements (achieving 3 detections then losing the target). Since the large positive was very hard to achieve for the learning agent it did not receive it often and had difficulty reinforcing what components of its action resulted in the success.

Through investigation, I found that a simpler reward with larger rewards for illumination provided the best results. Additionally, this reward was calculated immediately after an action. Since the model did not have memory beyond a single time step (past actions in the state $[A_{x-1}, A_{y-1}]$) a reward for a series of actions resulted in confusion and policies that converged toward the mean of actions toward the center of the map. The transfer learning method initially trained a model with the ability to cheat so that it was provided exactly the information of where the target was likely heading. This resulted in a successful model but did not help the model when tested to perform in the low information state. Instead the model was heavily affected by false alarms and tracked those false detections away from the target.

Due to the disparities between these approaches, it is not possible to directly compare the performance of the reward functions based on the average reward. So, in order to compare these models, they were each integrated with the main game and tested as the tracking mode detectors in $1,000$ randomly generated games. They were evaluated based on their win percentage in the main game. Agents trained presented

here were the top performing for the original simulation, the GLRT expansion, and the range-Doppler processing. Of the 12 models, multiple came within 5% of the best models. This testing showed that there was an upper limit to the expected RL performance that likely needs a re-evaluation of the RL structure or the training process to exceed this limit.

Two models were used for different phases of this effort. The first model was for the initial work and had a reduced state and was trained in the original simulation (Section A.2). This model's architecture is summarized in Table 4.2 and its hyperparameter settings are summarized in Table 4.3.

Actor Model			
Layer	**Input Dim.**	**Output Dim.**	**Activation**
Input	13	13	N/A
Hidden	13	128	ReLU
Output	128	2	N/A
Critic Model			
Layer	**Input Dim.**	**Output Dim.**	**Activation**
Input	13	13	N/A
Hidden	13	128	ReLU
Output	128	1	N/A

Table 4.2: Model Architecture - Original Simulation

The initial reward function provide the agent with a reward of $+5$ after each time step in which the target was detected and a -5 after each time step if failed to detect the target. No win condition was set and each training game would end when either the target reached its objective or the detector lost the track (five consecutive misses). If the target reached the objective before the detector lost the track, the game simply ended without any additional positive or negative reward. A reward of -10 was pro-

Parameter	Value	Parameter	Value
Actor learning rate	10^{-4}	Critic learning rate	10^{-4}
Actor Optimizer	Adam	Critic Optimizer	Adam
Batch Size	2	Discount Factor	0.9
PPO Clip Ratio	0.2	Action Scale Factor	$2,000$

Table 4.3: Model Hyperparameters - Original Simulation

vided if the agent lost the track. This approach was simpler and focused more heavily on teaching the agent to simply get a detection after each action.

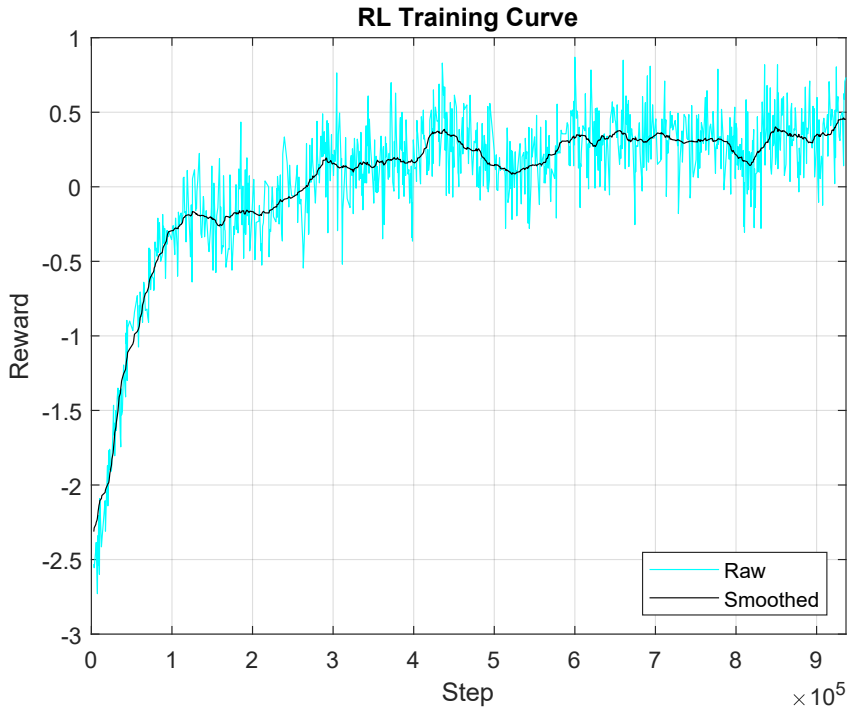


Figure 4.4: Training Curve for the RL Tracking Agent.

The final model was trained for $\sim 935,000$ steps and converged to an average reward of just under 0.5. The model converged after $300,000$ steps. The training curve for this model is depicted in Figure 4.4. The final model was selected by integrating

the trained model into the game detectors and measuring the win rate for each in the final game setting.

This process of exploration with the expanded game and the more advanced simulation methods resulted in the final model which is summarized in Table 4.4 and its hyperparameter settings are summarized in Table 4.5.

Actor Model			
Layer	**Input Dim.**	**Output Dim.**	**Activation**
Input	18	64	ReLU
Hidden ($\times 3$)	18	64	ReLU
Output	64	2	N/A
Critic Model			
Layer	**Input Dim.**	**Output Dim.**	**Activation**
Input	18	64	ReLU
Hidden ($\times 3$)	18	64	ReLU
Output	64	1	N/A

Table 4.4: Model Architecture

Parameter	**Value**	**Parameter**	**Value**
Actor learning rate	10^{-4}	Critic learning rate	10^{-4}
Actor Optimizer	Adam	Critic Optimizer	Adam
Batch Size	10	Discount Factor	0.9
PPO Clip Ratio	0.2	Action Scale Factor (w_{RL})	$1,000$

Table 4.5: Model Hyperparameters

The reward function for an RL agent reflects the action selection's impact on the environment and will reinforce desired results with positive rewards and discourage

undesirable behavior with negative rewards. The reward design and tuning is often the most influential and difficult process for RL utilization. We decided to use a simple reward structure to better isolate the factors leading to convergence and diagnose undesirable behavior. The reward structure used for the final agent is in equation 4.16:

$$Reward = \begin{cases} +25 & \text{target is illuminated} \\ -5 & \text{target not illuminated} \end{cases} \tag{4.16}$$

The final model was trained for 100 epochs. Each epoch was a series of 20 games and the average number for time steps in those games was recorded to be 8.55. This means that the agent was trained for $\sim 200,000$ steps and converged to a cumulative reward of 29.24. The training results are shown in Figure 4.5:

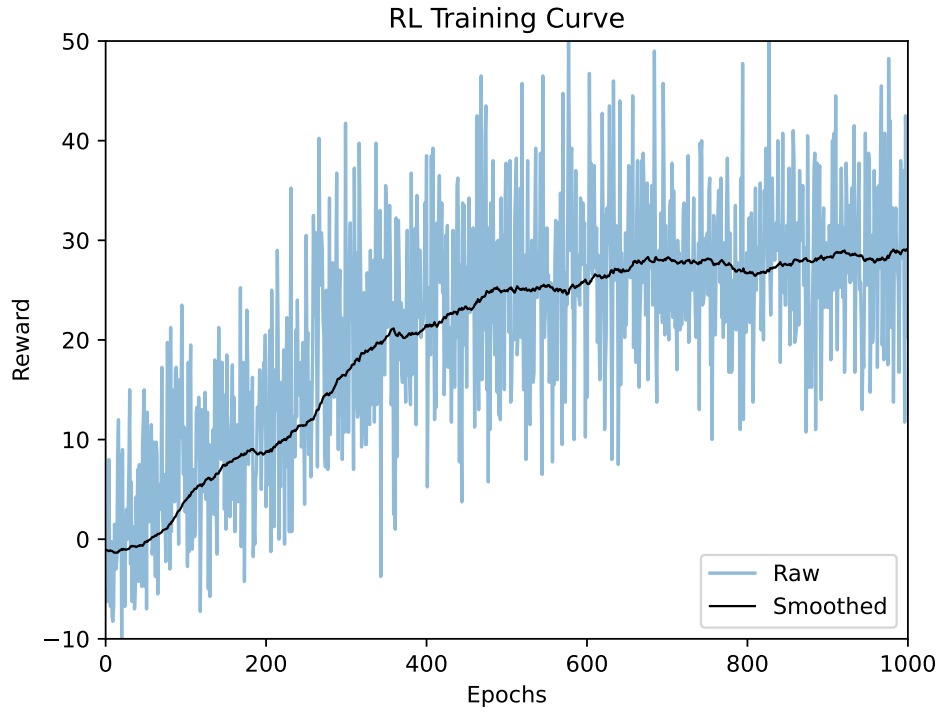


Figure 4.5: RL Rewards per Epoch During Training

The agent converged its reward at ~ 650 epochs. This model was saved and the actor policy (π) was implemented in a feed-forward mode to act as the track confirmation agent. The results of the agent and its integration with the GT are discussed in results section 4.5.

4.4.4 Game Theory Implementation

The final area of novelty I researched is the impact of game theory (GT) principles to impose collaboration upon multiple agents. I provided the full design, implementation and ideation for this work and hold sole novelty. Section 3.3 goes into detail on GT as a field but it is important that the work here explores modified applications of GT methods and not a solution focused application of GT. A major subset of GT is to model and define a game and then utilize solution methods to derive optimal strategies to solve the presented game. The application of GT in this work looks only at the collaborative strategy methods in the literature that are used to maximize utility for a multi-agent game. The main reason to avoid solution methods is that those solutions often need to have games with more rigid rules, high or full information, and/or assumptions about environments and agents. This work is focused on the radar domain and it is important that any simulation or game is representative of reality and the radar challenge. Given that necessity, it is difficult to have a close form solution in the GT solution sense. Regardless, the field of GT has great potential to improve multi-agent task generation. The approach I decided to take is to examine collaborative GT methods, and adapt them into the research as strategic modification methods in the hopes of driving improved performance even if a solution is unattainable.

The final steps of implementation for this work focused on the application of three GT based collaborative methods. This first method is a baseline method that is the lack of collaboration. This method is called non-collaborative best policy/response (NCBP/R) and means that each agent will not share information but will take the

action that its policy determines given the observable state.

The first GT method, NCBP/R, is implemented as a baseline to test the effectiveness of the track confirmation methods without any information sharing or state/action override. Both agents will see their own observable state based on the confirmation method used. This often results in different agents existing in track confirmation and search modes at different times. Because these agents are not sharing any information, they will often be in different modes between track confirmation and search recovery. Their beams will overlap and they do not attempt to optimize actions given multi-agent information.

The second method is a naive collaborative method called collaborative best policy/response (CBP/R). This method will have agents share detection information such that if one agent has a detection then the second agent will gain that information and make policy decisions as if it had also made the same detection. These agents will continue to make personal action selection based on their policy but there is no mechanism to try and optimize collaborative utility.

The final GT method is a modification of the consensus method inspired by the Stackelberg game type. The Stackelberg game is simply a game structure where actors take turns to implement their actions as opposed to simultaneous actions. The consensus method attempts to maximize the utility of of all agents in a collaborative network or coalition as opposed to each agent maximizing its personal utility. In this research I refer to the consensus method as a leader-follower (LF) consensus method. This method was very similar to the initial implementation and the expanded simulations. There are a few updates to the LF consensus algorithm for the expanded simulation and results.

For this LF strategy, information is shared between agents. When in track confirmation mode, one agent is assigned the role of leader and the other the role of

follower. The inspiration for the LF method was to force a following agent to augment the action selection of a leader and the search in areas near the leader but not directly overlapping. The updated method utilizes the following logic and equations. When the LF GT method is being used, the two agent's will determine the "leader" as the agent with the most recent detection in the previous time step. If both agents have a detection then one is chosen at random. If both agents get detections on the same time step or last received a detection at the same time step, then the closer agent takes the role of leader. The leader continues to take personal optimal actions while the follower obeys a heuristic to support the leader's actions. This heuristic directs the center of the follower beam as a Gaussian offset with the mean as the difference between the closest leader $3dB$ beam edge and the communicated detection location. Equations 4.17 shows the selection for the follower action and subsequent beam(s):

$$\begin{aligned} & N(\mu_{follow}, 200m) \\ & \mu_{follow} = \frac{(x_{lead3dB}, y_{lead3dB}) + (x_{det}, y_{det})}{2} \\ & (x_{lead3dB}, y_{lead3dB}) = argmin_{(x_{i,lead3dB}, y_{i,lead3dB})} \\ & Range[(x_{i,lead3dB}, y_{i,lead3dB}), (x_{follow}, y_{follow})] \end{aligned} \tag{4.17}$$

Where $Range(\bullet) = \sqrt{a^2 + b^2}$, $(x_{i,lead3dB}, y_{i,lead3dB})$ is the list of $3dB$ edge points generated by the leader action's beam, and (x_{follow}, y_{follow}) is the location of the follower detector.

$$\begin{aligned} & N(\mu_F, 200m) \\ & \mu_F = \frac{(x_L, y_L) + (x_D, y_D)}{2} \\ & (x_L, y_L) = argmin_{(x_{i,L}, y_{i,L})} < (x_{i,L}, y_{i,L}), (x_F, y_F) > \end{aligned} \tag{4.18}$$

Where $< \bullet >$ is the Euclidean distance between points, $(x_{i,L}, y_{i,L})$ is the list of $3dB$ edge points generated by the leader action's beam, and (x_F, y_F) is the location of the follower detector.

This leader-follower method was designed with the goal to cause the agents to not only search together but to avoid wasting potential utility by directing their beams to overlap. There will always be some overlap and it is not strictly a disadvantage as even a beam directly on the target can fail to detect and the simulator implements a probability of detection. Nevertheless, the optimal strategy would be for the follower to search near the leader but to effectively extend the search space by being offset to the leader action. For this initial research I utilized a heuristic method and included the variance of a Gaussian selection to continue providing flexibility.

For the expanded work the LF method was modified to include the velocity information to improve performance. The mean for this selection utilizes the leader action, the beam width, and the leaders radial velocity. Using the beam width and number of the beams the algorithm determines the radial edges from the action:

$$\begin{aligned} &([x_{edge1}, y_{edge1}], [x_{edge2}, y_{edge2}]) = \\ &[R_L \cos(\theta_L \pm \theta_{bw} n_{beam}), R_L \sin(\theta_L \pm \theta_{bw} n_{beam})] \end{aligned} \tag{4.19}$$

In (4.19), R_L represents the range between the leader action and the leader's location, θ_L is the azimuth angle between the leader location and the action coordinates, θ_{bw} and n_{beam} are the beam width and number of beams in the selected beam splitting mode. Once the edge pair is calculated then the follower determines which edge location is closest to its location. Finally, the leader's radial velocity is added to the closest edge to determine the mean of the follower action. The logic behind this method is that the follower should focus a beam that will support the leader if the target evades. Neither radar knows where the target might evade, but the radial velocity information does tell the radar agents if the last detection had the target moving toward or away from the leader and a rough magnitude of that velocity. The standard deviation is the maximum between the norm of the leader radial velocity and a third of the max velocity. In other words, define the following as:

$$
\begin{aligned}
\mu_{follow} &= (x_{edge}, y_{edge}) + (Vx_{rL}, Vy_{rL}) \\
\sigma_{follow} &= max(\|(Vx_{rL}, Vy_{rL})\|, \frac{V_{max}}{3}) \\
(Ax_{follow}, Ay_{follow}) &= N(\mu_{follow}, \sigma_{follow})
\end{aligned}
\tag{4.20}
$$

An example of this behavior is also depicted in Figure 4.6. In addition to the action override for the follower, the followers state is also modified using the CBR method so that if the follower becomes the leader on following time steps then the state will be in a usable form for a track confirmation that uses state history information.

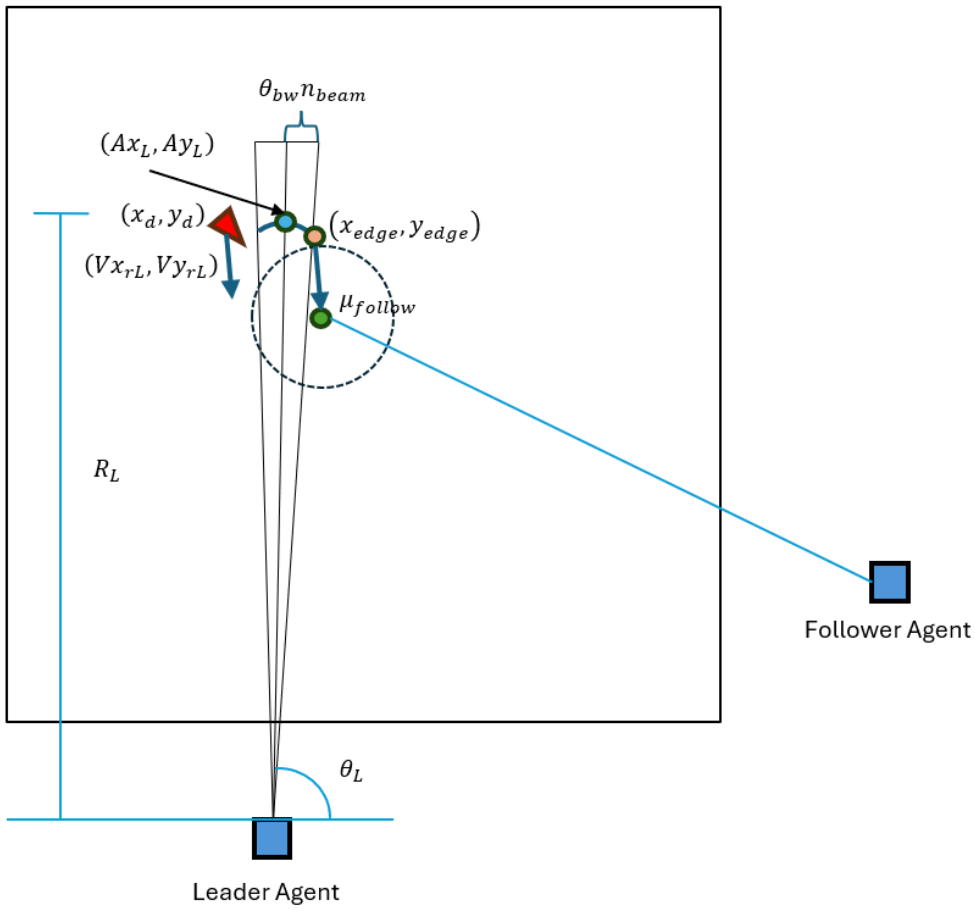


Figure 4.6: Leader Follower Visual Example

4.5 Track Confirmation Results and Discussion

This work was tested three times with a different radar simulation and expanded methods for a journal publication. Therefore the following results section will be organized by the different metrics and analysis topics. For each testing metric or analysis testing it will present each version that is relevant and discuss both the result and insights regarding the impact of the simulation and method differences. The first section will discuss the methods for testing and recording of metrics.

4.5.1 Testing Set-up

For the original range equation based simulation a total of six combinations of methods (two tracking agents and three GT methods) played 10,000 games each. Each combination of agent and GT was executed and analyzed for two variations on the parameter n_{TL}.

This initial exploration into a multi-agent track confirmation RRM challenge applied a track confirmation, search method, and collaborative GT heuristics. I set up the testing scenarios with variance, but with enough consistency so that a Monte Carlo implementation would enable confidence in the impact of the agent methods and the game theory applied. The following is a discussion of the results from this initial research and will focus on agent methods and their impact followed by the GT method impact and a final analysis of their combined impact.

The core metrics analyzed for this initial conference paper included win rates for each combination of track agents and GT, and the turn that games ended. A win, as described in the game scenario section, is if six detections are made in a sliding window of ten and a loss is due to the target moving within a radius of 200 m of its target objective point.

The expanded game for the journal was designed with the adjustable parameters to support different testing methods. The goal of the testing is to explore the effectiveness of the three track confirmation trackers (naive Gaussian, Gaussian velocity predictor, and RL agent) as well as the impact of the three GT methods utilized with those track confirmation methods. The first testing method looks at the performance of the track confirmations without collaborative GT and for a single agent. This testing was conducted with a game simulation containing a single target of interest, four objective points, and a single agent. The position of the agent is always 3000 m from the surveillance map in either the x or y coordinate.

This expanded game also used the metric of win rate but tested a sweep of agent starting angles in order to get a greater insight into the impact of the clutter ridge. This implementation of the game featured four objective points and two agent radars. The first radar is placed randomly with an offset of 3000 m in either x or y coordinate, while the second is given an angle offset centered at the middle of the map. This allowed us to test a sweep of different angles between radars to see the impact of the orthogonality of the beams on the agents and GT methods to confirm the track of a target. These sweeps were run at $15°$ increments from $30°$ to $180°$. Each sweep had a total of 500 independent game scenarios for a total of $5,500$ tests for each combination of track confirmation method and GT pair.

Beyond the win-rate metric this journal explored several additional metrics in order to gain additional insight into the impact of the track agents and the collaborative methods. The additional areas of analysis include: probability of detection, percent overlap of action beams, and the impact of false alarms. These additional analysis metrics were introduced due to an expanded scope for the journal and due to the introduction of a clutter ridge and false alarms. The original range equation simulation did not have a clutter ridge or false alarm impacts to analyze. The updated range-Doppler and cell averaging CFAR simulation has the same testing set-up and metrics for anal-

ysis as the GLRT journal simulation. The following section will include the results in order of success metric and analysis topic.

4.5.2 Track Confirmation Results

4.5.2.1 Results - Win Rate

The win rates for the initial simulation is shown in Table 4.6:

Gaussian Track Agent		
GT Method	**Win Rate** ($n_{TL} = 5$)	**Win Rate** ($n_{TL} = 3$)
NCBP	40.10%	43.15%
CBP	44.65%	49.04%
LF	55.51%	58.15%
AC-PPO RL Track Agent		
GT Method	**Win Rate** ($n_{TL} = 5$)	**Win Rate** ($n_{TL} = 3$)
NCBP	53.59%	53.96%
CBP	53.38%	54.64%
LF	59.62%	60.52%

Table 4.6: Win Rate Results (10,000 games each method)

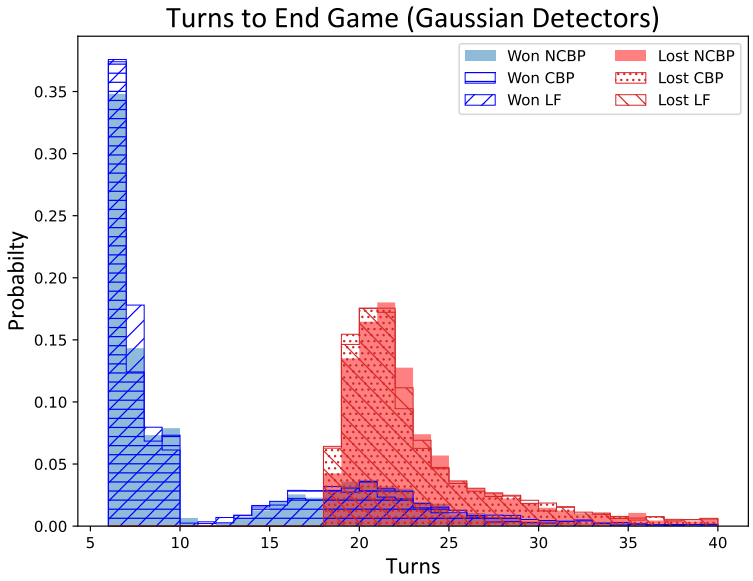
The initial results showed a few trends that inspired the pursuit of additional testing. First the scenario was designed with appropriate difficulty to gain meaningful results. Within the field of GT a challenge needs to have a baseline success rate that is not too close to 0% or too close to 100%. This need is so that the impact of different agents and collaborative methods can be observed and analyzed. In this case the baseline method, naive Gaussian track confirmation method with no collaboration (NCBP), had a win rate of 40.10% and 43.15% for $n_{TL} = 5$ and 3. For a simulation with no clutter ridge or false alarms, the RL agent performs significantly better than

the Gaussian, with an increased win rate of $10 - 13\%$. The parameter for turns to enter search (n_{TL}) had minimal impact on both methods but was slightly positive for entering the search mode earlier and potentially recovering a lost target.

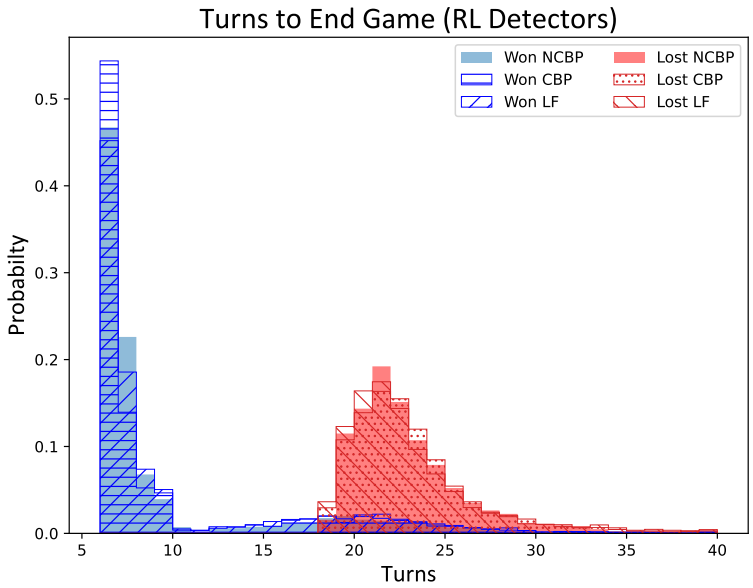
In order to have deeper insight into the impact of mode-switching for the different agents, I recorded the length of games regardless of outcome. This information was processed into histograms representing the turn when the game ended for each track confirmation agent and the corresponding outcome. Figure 4.7 gives more insight into which turn the end condition occurred and which agent won. The earliest turn a game could be won is on turn six (the window size to establish a track).

Analysis of these histograms show that the RL agent is much more likely to win after the initialized detection on turn zero. Although both methods do have some games won after a search mode switch, the target is evasive enough that once lost, it is much more likely for the agents to lose. The following sections will now analyze the impact of the GT methods on the win rate for both the naive Gaussian and the RL track confirmation methods.

The first GT method to review is the impact of the CBP GT method. As expected the fact that the agents looked in the same area of the target improved win rates for both methods. The naive Gaussian method saw an increase in win rate of $5 - 6\%$. The histogram in Figure 4.7a shows that the majority of this increased win rate was on turn six. This means that additional sharing of information increase the probability of the two agents getting subsequent detections and winning in those first ten turns more often. Effectively, the pair got to look twice with the Gaussian track at each time step, and better detect the target if it does not evade out of their search zone. On the other hand, the RL agents see almost no change in their win rate due to this collaboration. This is due to the two RL agents being copies of each other. Given that their observable state will be very similar I found that they consistently looked in the same place. The histogram in Figure 4.7b supports this assertion as the win rate on turn 6 increases by



(a) Histogram of Turns to End for Gaussian Agent.



(b) Histogram of Turns to End for RL Agent.

Figure 4.7: Histogram of Turns to End a Game ($n_{TL} = 3$)

10% for just CBP. This method is increasing the probability of maintaining the target track from game start but loses win rate from a second detection and track later in the game.

The LF GT method had a noticeable increase in win rate for both the Gaussian and RL tracking agents. The greatest gain was experienced by the Gaussian agent at $9-11\%$. The Gaussian agent, using the LF method, gains the benefit of collaboratively searching the same localized region without the inefficiency of both agents generating beams that are mostly overlapping. The RL agent also saw improvement from the LF method but to a lesser extent (6%). The histogram shows that the majority of the gained win rate is in the turns six to ten. This means that the follower action is supporting the RL agent with a larger effective search region. Given the LF method is a Gaussian override, it improves coverage with less look overlap but it is not optimal for RL. Ideally, an RL agent trained to be in a follower mode could provide more support and a greater increase in win rate.

The mean win rates for the journal expansion and the range-Doppler processing is shown in Table 4.7.

Additionally, the expanded simulations both were tested for a sweep of angle difference of the two platforms. The Figure with these results are shown in Figure 4.8 and Figure 4.9.

This expanded work has multiple differences from the initial results discussed above. Most importantly, I designed an additional tracking agent and the higher fidelity radar simulation introduced both a clutter ridge and false alarms. This additional tracking agent (Gaussian Velocity Predictor) was inspired by traditional tracking methods in the family for the Kalman filters [124]. Some early testing attempted to use the extended Kalman filter to help with reducing noise and tracking the target, but that class of Bayesian trackers needed several detection points to converge. The proposed

GLRT Simulation			
Track Confirmation	**GT**	**Test Run**	**Win Rate**
Naive Gaussian	NCBR	5500	19.22%
Naive Gaussian	CBR	5500	62.76%
Naive Gaussian	LF Consensus	5500	69.51%
Gaussian Velocity	NCBR	5500	58.95%
Gaussian Velocity	CBR	5500	84.36%
Gaussian Velocity	LF Consensus	5500	78.93%
RL Agent	NCBR	5500	37.62%
RL Agent	CBR	5500	59.24%
RL Agent	LF Consensus	5500	60.36%
Range-Doppler Simulation			
Track Confirmation	**GT**	**Test Run**	**Win Rate**
Naive Gaussian	NCBR	5500	3.84%
Naive Gaussian	CBR	5500	29.25%
Naive Gaussian	LF Consensus	5500	44.96%
Gaussian Velocity	NCBR	5500	31.44%
Gaussian Velocity	CBR	5500	67.44%
Gaussian Velocity	LF Consensus	5500	60.29%
RL Agent	NCBR	5500	18.09%
RL Agent	CBR	5500	30.76%
RL Agent	LF Consensus	5500	37.85%

Table 4.7: Mean Win Rates

confirmation track challenge has limited information, and by the time enough detections occur a confirmation agent has succeeded or the target has evaded enough that

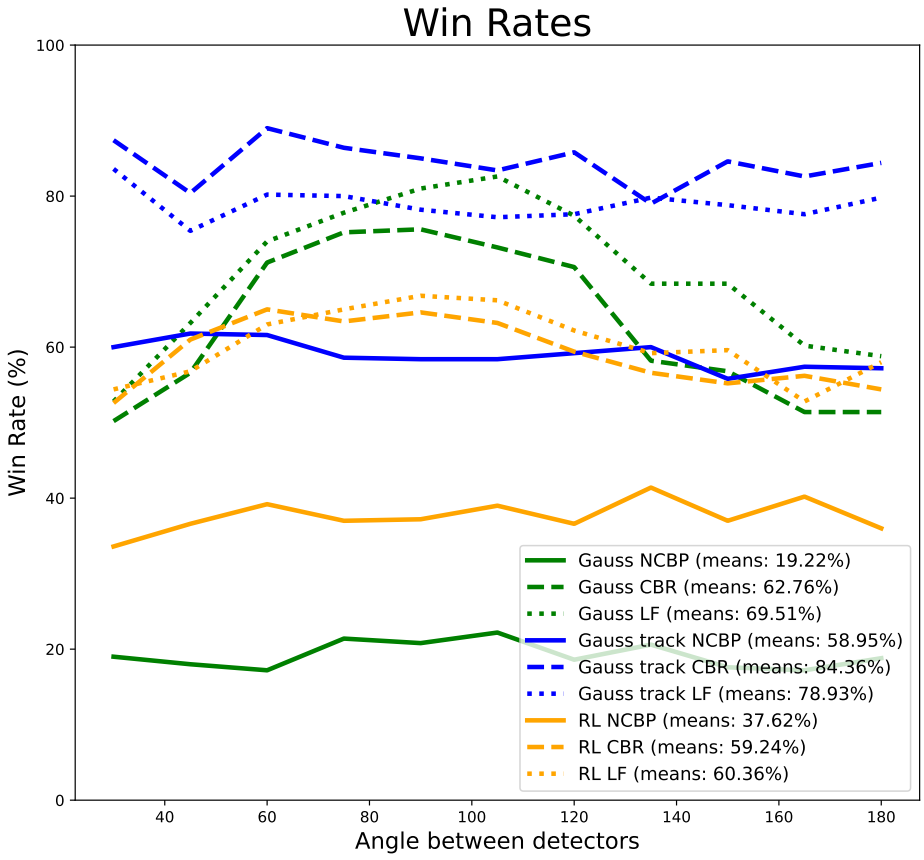


Figure 4.8: Win Rate Plots for Angle Offset Between Detectors - GLRT Simulation

detections are not frequent. This motivate the heuristic design of the velocity predictor so that it would utilize the information from multiple detections, but still keep the Gaussian variability to reflect the uncertainty in the evasive target behavior.

The clutter ridge implementation (Section A.3.2) combined with the evasive target behavior to truly highlight the necessity of multi-agent collaboration in this space. Once a target was illuminated it had a tendency to move orthogonal to the illuminating source. This means that its relative velocity would approach 0 and the target would become hidden in the clutter ridge and greatly reduce P_D (see Figure A.1). This

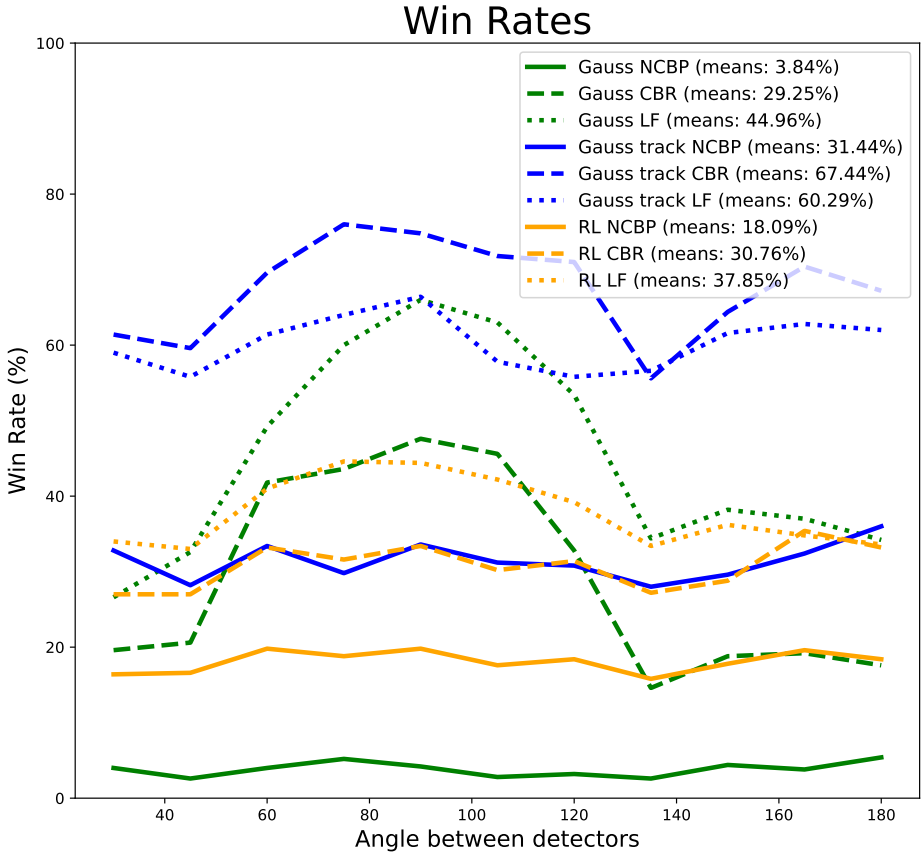


Figure 4.9: Win Rate Plots for Angle Offset Between Detectors - Range-Doppler Simulation

feature is also the inspiration for the sweep testing in these methods. This way I can analyze and discuss the impact of agent separation for detection of this class of evasive targets. The impact of the false alarms is analyzed in detail later but it imposes a real challenge for tracking methods that rely too heavily upon the certainty of any specific detection.

The naive Gaussian with NCBR had the lowest win rate at just below 20%, which was in line with our expectation as the system parameters were designed to make

the default game difficult. The range-Doppler simulation shows this fact even more profoundly as the baseline method has 4% win rate. This is because the agents are not collaborating and the target will turn orthogonal to single illuminating agent. In this case the naive Gaussian will lose the target and have very little chance to find the target again. The reason this is even more pronounced for the range-Doppler simulation is that the CA-CFAR detection is even more sensitive to the clutter ridge. Effectively a target could be resolved as it approach the clutter ridge much more effectively with the GLRT than the CA-CFAR. Since the Gaussian velocity predictor utilizes state history to predict velocity it was expected to perform better. The RL had a greater increase in win rate over the naive Gaussian, but did not benefit for the second agent as much as the probabilistic methods.

In all cases the GT methods showed improvement over the absence of collaboration represented in NCBR. The information sharing with CBR increased win rate as both confirmation agents made action selections in the same local region as the past target detection location. Although CBR improved win rate by $> 20\%$, it still had the naive Gaussian and RL agents waste potential utility by having beams overlap heavily and not explore unique space. The Gaussian Track method was improved most by CBR. Further analysis showed that this was due to the way that each agent in this mode would collaborate and update their personal velocity estimations. If a platform had a personal detection they would trust their detection information. If only a single platform had a detection then they would share the detection location. In either case, each would use it's own detection history to derive the velocity estimation and determine the next action. This means that information was shared but actions did not necessarily overlap and waste utility. The improved LF method from the original work in [20] utilized the radial velocity information to not only have each agent's action in the local area of detections but to have the supporting agent increase collaborative utility by looking away from the leader agent by the magnitude and direction of the

radial velocity. For the Gaussian and RL methods, the LF had a slightly greater impact on win rate. Although the win rate for the range-Doppler processed simulation were lower the general trend between methods was similar to the GLRT. This confirms that the impact of the clutter ridge and the performance of the trackers and GT methods are consistent.

An interesting and expected behavior seen in the angle sweep plots is that for the collaborative methods (CBR and LF) that the angle difference between the platforms has an impact upon win rate. As expected the closer that the angle difference approaches 90% the maximum win rate occurs. Further analysis shows that this is due to the fact that the evasive target will attempt to move so that its velocity vector it orthogonal to the illuminating source. But if the collaborative agents are orthogonal to each other then the target cannot hide in the clutter ridge against both observers. Similarly as the agents are positioned closer to 0% and 180% their performance degrades.

A result shown in these plot that was discussed some in the section on RL 4.4.3 is that the RL seems to hit a performance ceiling in both its personal performance as well as the impact of the GT methods. Further investigation into the challenge with RL has lead to the following insights: The RL was trained alone and utilized the illumination for reward. This was necessary for convergence of a model, but also caused the model to converge an deterministic state action estimates. This means that the given a previous detection and other similar state information the RL will typically take a similar action with little variance. This means that it will not benefit from CBR very much as both agents will typically look at the same spot given the similarity of their state. This can improve P_D if the target is illuminated but it also narrows the space illuminated resulted in a lower chance of target illumination. This is why the RL-CBR pair shows lower win rate gains and does not have the same angle response as the other track methods. Also the rigid policy of the RL does not have the time history of the velocity predictor or the variability of the Gaussian methods. This is why the

non-collaborative RL performs between the other trackers with no collaboration, but gains the least from the GT. Future effort in RL will likely need to completely re-analyze the state, the model (giving memory with an RNN), and/or investigate MARL option to train multiple agents simultaneously.

4.5.2.2 Results - Probability of Detection

Although a metric like win rate for this scenario provides information on the performance of the methods and GT, I want to include an additional analysis of the probability of detection for each combination of methods and GT. For these results, the individual look actions across the angle sweep testing were analyzed, but only the actions taken in a track confirmation mode (not search) were considered. The probability of detection results are shown in Figure 4.10 for the GLRT method and Figure 4.11 for the range-Doppler and CA-CFAR method. This represents the joint P_d of the platforms so that if either platform has a detection, then it is counted as a detection.

The P_d results follow a pattern similar to the win rate results and confirm that the confirmation scenario with the detection window is correlated to the individual detection rate. An interesting trend that is shown in both the win rate and P_d figures is that the naive Gaussian and RL methods show the best results as the starting position of the platforms approaches $90°$, with a degradation in performance as the platforms approach $30°$ and $180°$. This is likely due to the impact of the realistic clutter profile used in the simulation, which reduces P_d as the Doppler frequency and corresponding radial velocity approach zero. The question arises as to why the Gaussian velocity predictor does not see this same effect. Although some smoothing could be attributed to the variance of the scenarios, a total of 500 scenarios with ≈ 5000 radar actions and P_d calculations were analyzed for each angle difference, which indicates there are additional factors involved. The velocity prediction method differs because it utilizes detection history information, but its actions are derived from velocity estimations and

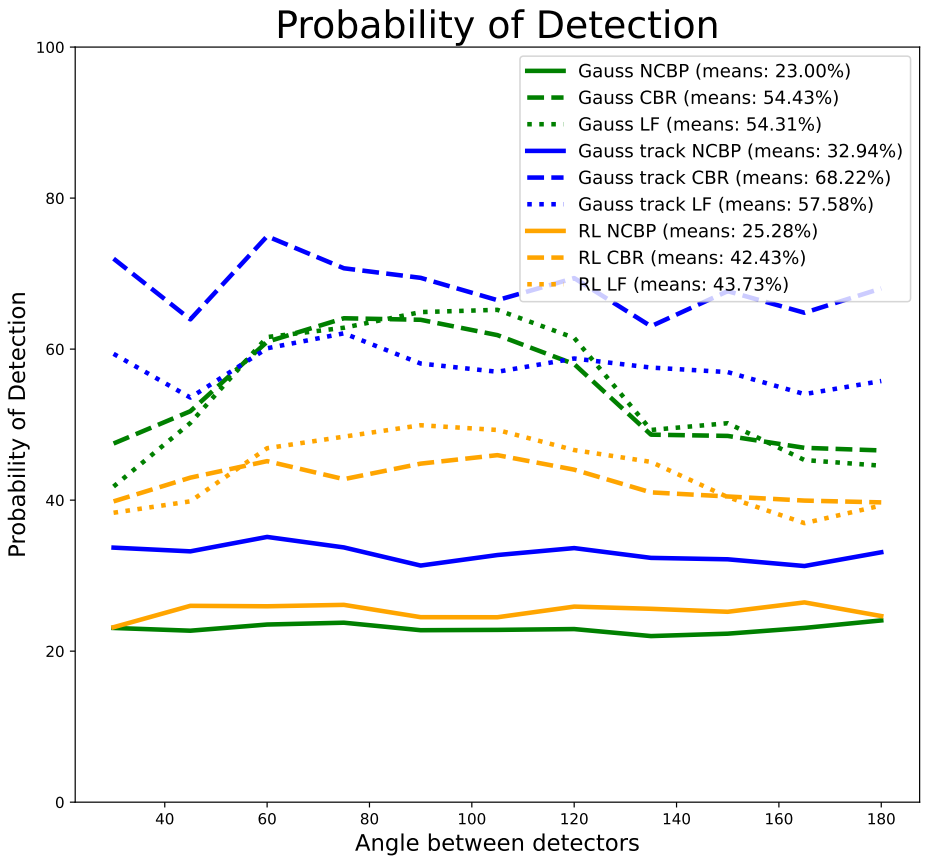


Figure 4.10: Mean P_d for Angle Offset Between Detectors - GLRT Simulation

not directly from the detection location. This is the reason it does not see a P_d response to the clutter based on orthogonal spacing of the platforms. Even if the platforms do not detect an illuminated target, their next look will progress with their velocity estimation (with a small decay). Therefore, a missed detection does not hamper their next action as much as the other two methods that do not incorporate historical information.

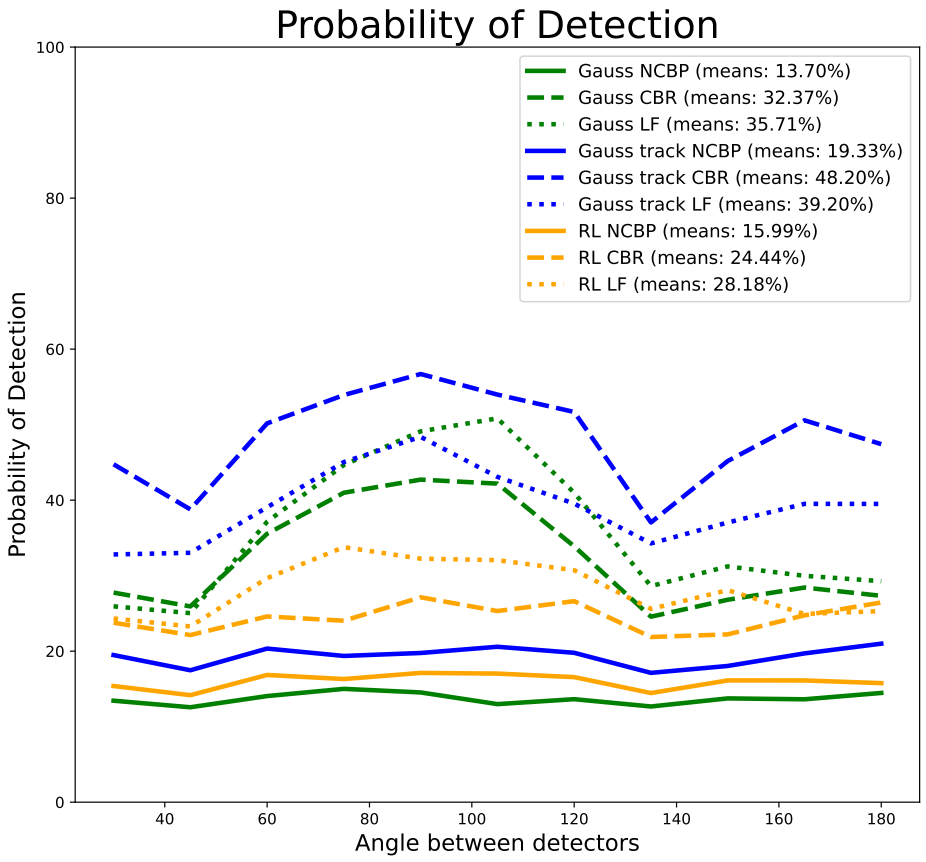


Figure 4.11: Mean P_d for Angle Offset Between Detectors - Range-Doppler Simulation

4.5.2.3 Results - Action Error

Figure 4.12 shows the heatmaps of $2,000$ tests performed to further understand the actions of the agents and the paths of the targets. The two radars were located below and to the right of the target starting location ($90°$ separation) and the objective above and to the left. The tracking heatmaps clearly show that the spread of look locations were along the beam axis. In addition, when compared to the naive cases, the target behavior was pushed away from the objective far more by the tracking agents. This

is likely the source of the additional error in the histograms and confirms the idea that the constant velocity or a policy to lead the target ends up harming the methods when the target is more likely to veer and stay in a local region. This means that the more an agent relies on assumptions regarding linear or near linear motion, the more it harms performance when the assumptions are violated.

Within Figure 4.12, Blue rectangles represent look locations, and red hexagons represent target locations. These plots are histograms, but the RL look locations are normalized logarithmically to better highlight the look structure. These heatmaps do show that the RL policy learned to lead the target based on detections. But since future detection where less reliable compared to the illumination in training, the RL made logical action selection but its lack of space exploration resulted in losing the target more often.

4.5.2.4 Results - Action Distance and Beam Overlap

The previous discussions were focused mainly on the impact of action selection methods alone or a broader analysis in the context of the track confirmation scenario. The following is an additional analysis and discussion focused on the two GT methods and the impact they had on action selection and overall win rate. The first analysis metric was the distance between action selections when using a collaborative method. These results are shown in Figure 4.13 for the GLRT simulation and Figure 4.14 for the range-Doppler simulation. These were conducted for a sweep of platform angles and includes the mean of actions over all angles.

The first and most obvious insight is that the distance for LF is greater than CBR and has less variance. This is expected as the LF method is a heuristic action override, so it should only have variance due to two conditions. First, variance occurs as a result of the Gaussian component in the follower look selection. Second, variance occurs as a result of target location and platform locations. The CBR has lower action distance

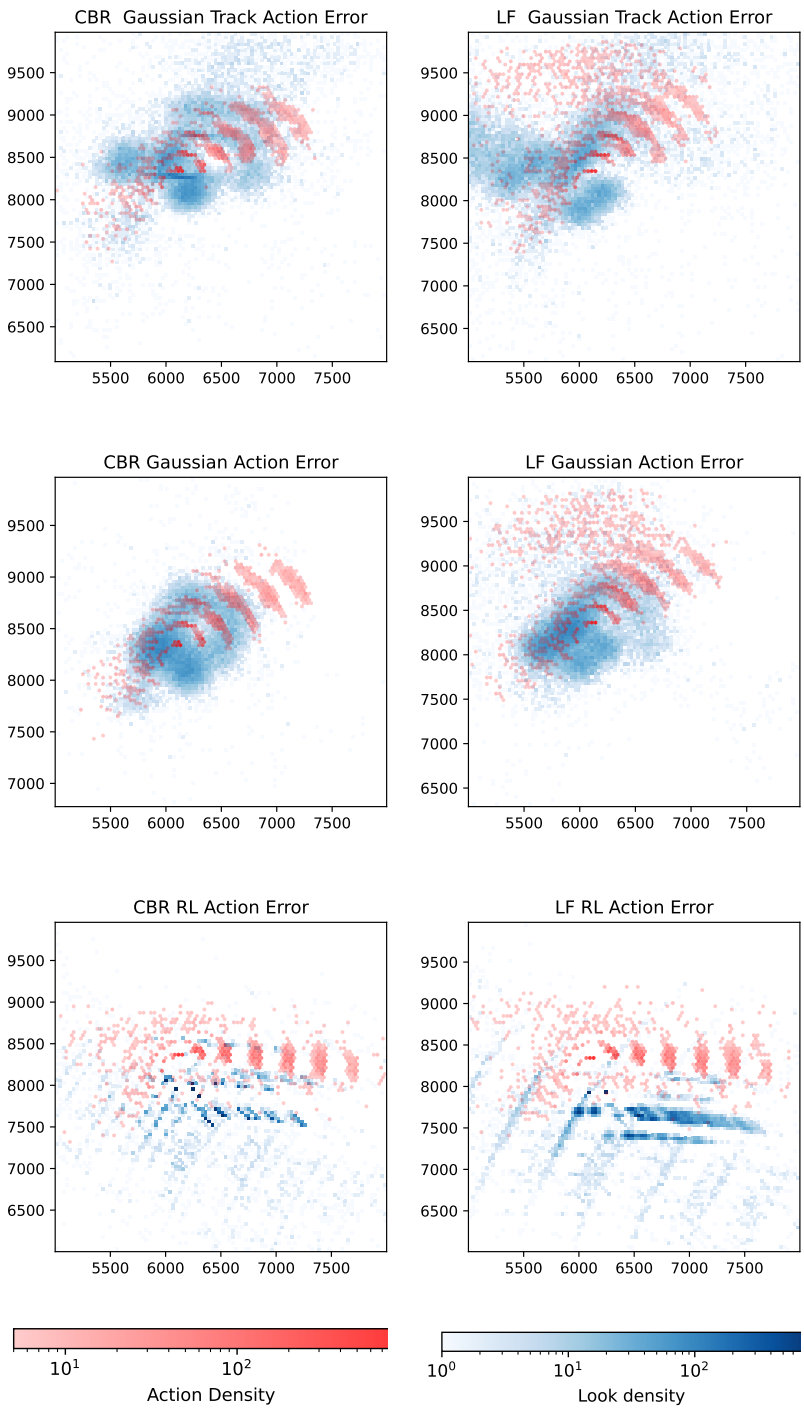


Figure 4.12: Heatmaps of Look Locations and Target Locations.

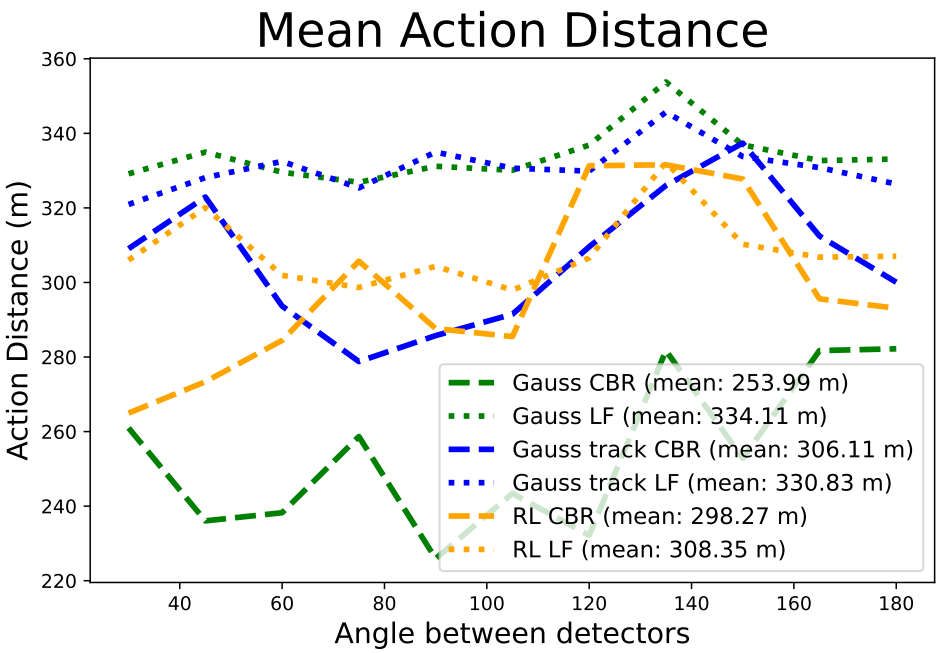


Figure 4.13: Action Distance Between Two Platforms - GLRT Simulation

given that if only one platform has a detection, then both platforms will share and make action selections based off the exact same detection location. The exception is the CBR-RL pair for the Range-Doppler simulation. Additional analysis found that the newest RL models had a state that included the velocity and platform information. This resulted in the actions in the CBR to have more separation as the states were different enough to increase distance in the action selections.

The second form of analysis of the GT methods investigates the degree of overlap for the beams generated by each of the platforms. To calculate these percentage values, the 3 dB beam was drawn for each platform based off their action selection. Then, a 1 km radius circle was drawn around the target. The total area of the circle that was covered by either of the beams and the total area covered by both of the beams are divided by the full circle area to represent the percent of overlap. A radius of 1 km was chosen because in a game window there could be up to 5 turns of missed detection

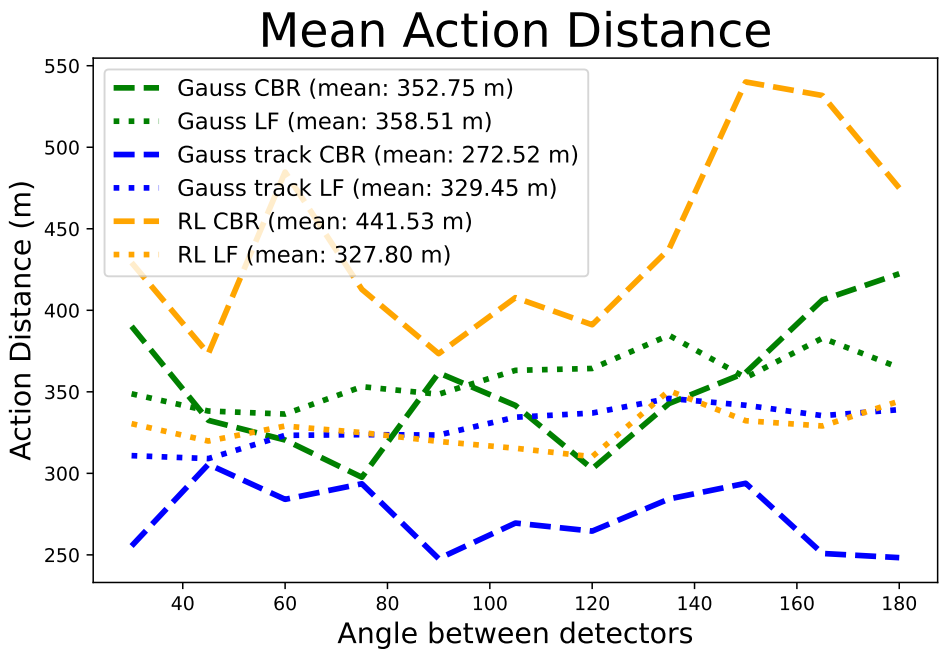


Figure 4.14: Action Distance Between Two Platforms - Range-Doppler Simulation

before the agents switch to search mode and the mean velocity of the target was 200 m/s. Therefore, the 1 km radius represented the reasonable space the target could maneuver in a game window. These results are shown in Table 4.8. In this table, coverage refers to the portion of the map that was observed by any platform, while overlap specifically refers to regions that are simultaneously viewed by both.

In general, this shows a trade-off between the coverage and the overlap. If the target has maneuvered to be orthogonal to one detector then an overlapping look will increase the probability of detection. On the other hand, if the target is performing evasive maneuvers, then increased coverage of possible target locations is of greater importance. For the naive Gaussian and RL methods the win rate is higher with LF consensus and they optimized search area over a double beam look compared to CBR. The highest performing method, Gaussian velocity prediction using CBR, seems to have balanced beam coverage with beam overlap. This is due to each platform sharing

GLRT Simulation			
Track Confirmation	**GT**	**Coverage**	**Overlap**
Naive Gaussian	CBR	56.0%	18.7%
Naive Gaussian	LF Consensus	58.6%	18.1%
Gaussian Velocity	CBR	60.2%	19.9%
Gaussian Velocity	LF Consensus	59.1%	17.7%
RL Agent	CBR	52.1%	16.2%
RL Agent	LF Consensus	54.1%	15.6%
Range-Doppler Simulation			
Track Confirmation	**GT**	**Coverage**	**Overlap**
Naive Gaussian	CBR	41.6%	10.7%
Naive Gaussian	LF Consensus	44.6%	11.5%
Gaussian Velocity	CBR	45.4%	12.2%
Gaussian Velocity	LF Consensus	45.0%	11.4%
RL Agent	CBR	37.7%	8.6%
RL Agent	LF Consensus	41.8%	10.3%

Table 4.8: Percent Overlap for 1km Radius of Target

information but also utilizing their own personal state history and velocity estimation.

4.5.2.5 Results - False Alarm Impacts

The next area of testing captured the impact of false alarms on agent performance. The GLRT radar simulation and the range-Doppler simulation included the possibility of false alarms. The game simulation had a $P_{FA} = 10^{-5}$ for the GLRT version and a $P_{FA} = 10^{-8}$ for the range-Doppler version. The main reason for this difference is that the GLRT had 16 range gates and 16 Doppler/velocity gates where the range-

Doppler version had 560 range bins and 128 Doppler/velocity bins. This increase in evaluated gates/bins was given to the increased number of bins during evaluation. For both simulations if a detection and false alarm was added to the detection list from a single illumination then an association process based on the distance to a previous detection would reduce the detection list to a single return. This means that not all false alarms had an impact on the agents, only those that happened in the absence of a target detection.

The parameters were tuned so that false alarms were frequent enough to impact the agents but not so frequent as to invalidate actions due to uncertainty. During scenarios, I recorded if false alarms were associated and assumed to be detections. In $\approx 40\%$ of games a false association happened and could affect the agent's behavior. Table 4.9 shows the win rates of each agent GT combination when a false alarm was present or not. For the GLRT version it shows that false alarms had a distinct but minimal impact on the win rate of the games. In the case of the range-Doppler simulation the difference is much more pronounced. The presence of false alarms always resulted in a low win rate regardless of method. The Gaussian velocity predictor was most affected by the false alarms, followed by the RL. This makes logical sense, as a false alarm would be trusted and then result in a false velocity or negatively affect a policy's action selection. For the velocity predictor this false velocity would negatively affect the next three actions as the predictor will follow that velocity estimate. Conversely, the naive Gaussian and the RL agent would only have the state for that time step corrupted by the false alarm. These results not only show the validity of including false alarms as they are a real challenge for RRM, but that agents that perform extremely well with perfect information will likely struggle as the certainty of the detection history reduces.

GLRT Simulation						
Agent	**NCBR**		**CBR**		**LF**	
Naive Gaussian	19.2%		62.8%		69.5%	
No FA	21.4%	+2.2%	65.3%	+2.5%	72.2%	+2.7%
FA	16.8%	-2.4%	60.0%	-2.8%	66.6%	-2.9%
Gaussian Predictor	58.9%		84.4%		78.9%	
No FA	63.5%	+4.6%	85.9%	+1.5%	80.8%	+1.9%
FA	52.2%	-6.7%	82.7%	-1.7%	76.8%	-2.1%
RL	37.6%		59.2%		60.4%	
No FA	41.8%	+3.8%	60.9%	+1.7%	61.8%	1.4%
FA	32.3%	-5.3%	57.2%	-2.0%	58.8%	-1.6%
Range-Doppler Simulation						
Agent	**NCBR**		**CBR**		**LF**	
Naive Gaussian	3.8%		29.3%		44.9%	
No FA	4.8%	+1.0%	34.3%	+5.0%	52.1%	+7.2%
FA	2.8%	-1.0%	20.2%	-9.1%	28.6%	-16.3%
Gaussian Predictor	31.4%		67.4%		60.3%	
No FA	38.5%	+7.1%	74.7%	+7.3%	68.4%	+8.1%
FA	19.3%	-12.1%	44.6%	-22.8%	38.5%	-21.8%
RL	18.1%		30.8%		37.9%	
No FA	23.4%	+5.3%	36.7%	+5.9%	44.9%	+7.0%
FA	10.1%	-8.0%	19.7%	-11.1%	24.1%	-13.8%

Table 4.9: Win Rates Given False Alarms

4.5.2.6 Results - Computation Cost

A final area of analysis focused on the computational cost of the three track confirmation methods. For this analysis, I ran each method and had it make action selections

for random target locations and states that they experienced during the simulation. A total of $100,000$ iterations of each method were run, and the mean was computed. These results were run on a Dell Precision 7750 with an NVIDIA quarto RTX 3000 GPU, an Intel Xeon W-10855M CPU with 2.80 GHz speed and 128 Gb of RAM. The naive Gaussian method averaged $1.89\ \mu s$ per action, the Gaussian velocity predictor averaged $24.61\ \mu s$, and the RL in feedforward mode took $659.4\ \mu s$. As expected, the simplest method took the least amount of time with the same computational resources. Even though the Gaussian velocity predictor was 13 times more costly and the RL was 349 times more costly, they were all still able to operate in real-time for radar scheduling needs simulated here. This is an important engineering trade-off, but assuming a radar system has similar or better computational resources, every method would perform in the available radar time budget.

Overall, these results confirm the effectiveness of GT methods, as both the act of information sharing and the optimization of shared utility provided marked improvement for the agents. The track confirmation method provided interesting insights into the benefit of using additional information to improve action selection. It also illustrated how individual first principles based calculations can both avoid wasted utility and outperform heuristic methods. The RL agents show that as a problem space increases in complexity, there is potential in flexible methods that make few inherent assumptions but also observe underlying patterns through Monte Carlo training to allow an ML model to find patterns and strategies from the environment. Methods like deep RL are less inherently understandable and require hyper-parameter exploration and tuning, but the potential flexibility and performance in high complexity or low information scenarios is worth exploring as potential options.

4.6 Conclusions

This work explores the RRM task of track confirmation but within the C2 challenge of UAV swarm search and evade. This adds the constraints that the target of interest is highly evasive with an intelligent behavior to utilize the 0 Doppler clutter ridge to avoid detection. Additionally, the ISR radar platforms have limited resources and are unable to re-illuminate a target detection for confirmation as the target may no longer be in the same space. The track confirmation task is performed to confirm that a detection is a target of interest and not a false alarm. False alarms will happen but if a detection is repeated and associated as the same source then confidence in detection greatly increases and a track is established. Traditionally, track confirmation is simply a re-illumination of the space. This method is effective when the radar platform has the resources and agility to illuminate the potential target quickly enough that it is close to the location of the initial detection. Alternatively, if the target does move too far from the initial detection, but follows a constant velocity path then the track can still be confirmed with a local search. This chapter address the C2 scenario where these assumption do not hold true.

In order to address this new challenge I design and implemented three core areas of contribution. First I was able to code an expandable simulation and environment in python that will prove invaluable for continued research. There exist multiple tools for simulation and environment including Matlab Radar toolboxes, python gym environment, and hi-fidelity simulations like RFview and AFsim. Although any or all of these could be used to explore either high-fidelity radar challenges or an interactive search and evade scenario with multiple agents, none could do both. Additionally, I would argue that there is value in implementing and understanding the radar and game mechanism components of the environment as it improves understanding of those domains and enables updates and improvements for the future. This simulation environment

had three increasing fidelity radar detection methods over two years of iterative research and learning. A current challenge in the simulation domain is that high fidelity radar simulations exist but require extreme computation. With in the RL and GT work there are multiple "games" that are representative and computationally efficient, but they tend to lack the realism and uncertainty that is a reality of the radar domain. This iterative simulation found a middle ground where radar realism and medium fidelity were included, but the computation and GT game design principles built a scenario that is agile enough for training RL and testing C2 scenarios. This contribution was not directly novel to radar or GT, but was a necessity to test and evaluate the other contributions in this work.

The second major contribution is that I was able to implement tracking methods for evasive targets in the absence of a detection history. This is a difficult challenge especially when the C2 scenario was formulated to have low radar resources and evasive targets, but the addition of collaborative multi-agent tactics resulted in performance improvement. This multi-agent UAV swarm scenario is an under-explored and growing challenge in the C2 autonomy domain. Additionally, both the low history tracking and the collaboration were implemented in such a way that it is modular to the human-in-the-loop C2 need. The final research benefit is the novelty of exploring advanced behavioral techniques like RL. Even the non-ML method were novel in their C2 knowledge added design. This means that the heuristics developed were based on the C2 information available to the agent. A total of three low history tracking methods (naive Gaussian, Gaussian velocity predictor, and multiple RL models) where implemented, tested, and analyzed. A final component of research novelty is the process of adapting GT methods focused on collaborative utility for multiple agents. Specifically a baseline of no collaboration, a strict information sharing, and a consensus leader-follower method were developed and tested. This application of GT showed improvement for the multi-agent case by establishing cooperation and optimizing col-

laborative utility for multiple agents.

Together these contributions provided an environment to test medium-fidelity radar tasks in a multi-agent C2 scenario. The core research goal to address the C2 autonomy challenge was to accomplish a difficult under explored challenge (track confirmation with low resources and evasive targets) but also to highlight the viability and importance of multi-agent utility maximization (using GT). The results showed that in a realistic scenario, agents should utilize information and history (Gaussian velocity predictor and RL), and they should include generalization mechanisms (i.e. probabilistic action variation). This main reason for the generalization mechanisms is that the realistic presence of false alarms can distract and lead astray methods with too rigid of an action policy. The RL research and exploration showed that the tool has promise but needs additional research and creative re-evaluation to improve. The RL methods were explored extensively but future research needs to explore completely new architectures and the potential of multi-agent training scenarios and integrating GT concepts into the training process. The application of probabilistic heuristic GT to a solo trained RL agent had diminished impact.

The implementation of GT was very successful and proved a vital component to accomplishing detection with a clutter ridge and evasive target. the evasive target design was intelligent and would balance approaching it goal and attempting to move orthogonal to an illuminating platform. Effectively, this would reduce its Doppler component to enter the clutter ridge and greatly reduce probability of detection. Extensive testing showed that the GT improved agent success in general. As platforms approached orthogonal positions relative to the target their collaborative success increased. These result validate that this C2 scenario requires new low history methods and that collaborative action augmentation can greatly improve objective success for collaborative agents.

The contributions in this work show the value in a collaborative multi-agent ap-

proach to integrated C2 challenges in autonomy. The domains of radar, autonomy, and multi-agent collaboration all have deep research and advances in solving problems within their domain. A key contribution in this work is exploration and testing of effective methods when all of those domains interact together to represent more complex and dynamic challenges. The likely reason that this interconnection of domains has less research is that it requires both a depth and breath of topic in multiple domains. Nevertheless this research is relevant because a method that is overly optimized in one domain may have reduced performance when integrated with the other domains in play for C2. An example in this work is that the RL in the simplified range equation simulation performed better than the baseline in all cases. But once the uncertainty imposed by higher-fidelity radar (clutter ridge and false alarms) was present then the GT made a much larger impact and the baseline tracking with GT surpassed the RL.

Part of the research philosophy for this work was to create track confirmation trackers and collaborative methods that were modular to enable human-in-the-loop possibilities. The proposed method succeeded but there was a cost. This came specifically in the result for the RL models. The exploration and results for the RL models showed that either there is a limit to its performance in this space or more likely the models need a training environment that is multi-agent. This has the potential of improved performance and the field of MARL show promise in this direction, but as the process becomes more integrated in the deep learning networks the more difficult modularity becomes. A single RL model could be replaced with another model or person, but single actor in a multi-agent model can't be replaced. Care was taken for the non-ML methods to generalize and scale with changes in environment as they were based on radar parameters or expectations about the targets. The additional issue with RL is that additional effort is required to build a model that generalizes to changes in environment or scenario. Design choices like action factors and state normalization were

implemented to encourage generalization, but large changes in scenario would likely need a retrained model. Even with the constraint, RL and ML has shown promise in multiple domains as the technology learns from data and can generalize solutions and strategy for complex problem spaces.

Although these results effectively modeled and explores this C2 challenge, considerable work is needed to build confidence, scale to realistic operations, and integrate this with real systems. As discussed, the RL models in this space could warrant a full graduate work to itself. The RL models should explore multi-agent training scenarios including MARL. The RL model could benefit from an architecture with some form of memory like an RNN. Finally the RL exploration could benefit from research into reward function. For the multi-agent RL case the use of GT methods like price of anarchy could be utilized to build dynamic collaborative reward functions. This work did explore GT as a collaborative solution but the field of collaborative GT includes multiple algorithms for utility optimization. The topics of price of anarchy, regret management, and confidence-bargaining method are promising and should be explored in addition to the consensus methods in this work.

Finally, this C2 scenario was simplified to only two stationary platforms and one target. A major area of expansion is scaling this scenario with many agents and multiple evading targets. Additionally, the platforms in this simulation could move but movement was not tested due to maintaining a complexity scope. Testing with motion is much more realistic and should be explored. A challenge imposed by this increase in scale is balancing tasks and collaborative resources. If there are 20 agents then not all 20 should look for the same target when other targets may exist. This could be addressed with a centralized control, but modern C2 scenarios are concerned with contested environment and a decentralized autonomy is a growing need. Proposed ways to scale in a decentralized way is to combine localization and implement a confidence-negotiation structure from coalition GT. A confidence-negotiation structure will have

agents evaluate the importance of possible tasks and then negotiate task selection with other agents to find an appropriate response of swarm resources.

The work presented in this chapter built an environment to test multi-agent track confirmation with C2 challenges. It is core to my research goal to utilize AI/ML and GT within RRM and C2 challenges. The C2 scenarios are domain relevant to the research, but the utilization of realistic radar is required to have confidence in the novel methods. The building of the scenario and the radar simulation greatly improved my understanding of radar signal processing and learning the challenges of the C2 domain. My research with the track confirmation methods improved my understanding of the balance between strategy and the need for generalization given uncertainty. A core interest when exploring C2 was the need for collaboration. During my career a challenge was integrating systems where individual components were optimized for their personal utility. When those systems where integrated they often harmed to collaborative utility because their local optimization was wasteful or counter-productive to the collaborative objective. The GT methods, focused on maximizing collaborative utility, was foundational to my research goals and showed promise from these results. These contributions were peer reviewed and evaluated by the community through a conference paper [20] and a journal paper for the expanded work [21].

Chapter 5

Scalable Multi-Agent Surveillance with Utility Representation

5.1 Introduction

As a continuation of my research into scalable multi-agent methods for radar resource management (RRM) tasks the next area of focus was inspired by the C2 challenge for space surveillance. Similar to the work presented in Chapter 4 the inspiration for this work examined the C2 scenario of multiple smaller UAV's as intelligence, surveillance, and reconnaissance (ISR) agents with the goal of accomplishing an RRM task. In RRM, the goal is to manage the resource utilization across multiple tasks. The most basic of these tasks are search, confirm, and track. Searching involves a systematic method for surveying unknown space with the goal of detecting potential targets of interest. Traditional search methods usually attempt to solve the problem under the naive assumption that all space is equal in importance. The second assumption is that updating and recovering lost tracks are of higher priority than searching space. In this case search is often what an RRM scheduler will do when it has left over resources.

The research in this section explores search while challenging these assumptions and expands to the multi-agent case with platform movement. In many real-world C2 scenarios, the search space may be too large to cover in a reasonable time. A solution to this time constraint is to rank small subspaces by importance. In many

C2 applications, intelligence information is used to assign importance to areas of the search space. Some algorithms incorporate prior information, but many are relatively rigid in the granularity of how the search space is segmented. The search method proposed balances exploration of radar space with exploitation of prior knowledge to meet the objective to find targets of interest. This method prioritizes higher probability of detection by illuminating more of the local area and optimizes collaborative utility when multiple platforms search together.

In a C2 scenario, it is reasonable to assume that a radar surveillance has physical locations that are of importance and therefore all points in space do not have equal priority. This led to a literature review (section 2.5) into the field of knowledge-aided search in which prior information can and should drive new search actions. The research includes a class of solutions focused on representations of the search space which use measures like entropy, probability of target existence, and radar action utility. This concept of a representation, given prior information, inspired the utility search framework presented in this section.

The goal of the utility representation is to balance exploitation of prior information and likelihood of success with exploration of unknown space. To accomplish this I designed an efficient representation with multiple layers that perform this balance. The first layer will estimate the likelihood of an action's success by using a Swerling 1 P_D estimation with the radar range equation. The second layer, called the objective layer, will represent C2 intelligence and encode the importance to survey areas of strategic importance. The final information layer utilized a novel beta distribution implementation that drives searching new areas but will revisit higher priority locations more often than low priority regions.

The final aspect, presented in this representation, is the ability for multiple search agents to share their search history and then search the space while maximizing collaborative utility. Even more significant is that this representation does not need action

sharing in real time to maximize utility as agents utilize their personal location to guide search selection. This results in improved collaborative search without making unrealistic communication simultaneity assumptions. When information can be shared then the collaborative utility is optimized to minimize beam overlap. The sharing process was designed to be scalable regardless of the number of agents.

I want to discuss my choices regarding the two-dimensional representation of the search scenario for this work. Given that a radar sensor on a platform transmits a propagating wave which both expands and loses power with range the actions are typically represented in a polar coordinate system. Specifically, the radar actions are represented with the azimuth angle, the elevation angle, and the range a beam of a given power can propagate and receive return energy that could be detected. If a researcher intends to reduce complexity they will often reduce the problem to only two dimensions. For RRM actions the most common choice is to assume that the radar action space should be represented with only azimuth and elevation. This representation is convenient as the space can then be represented by a grid, as shown in Figure 5.1.

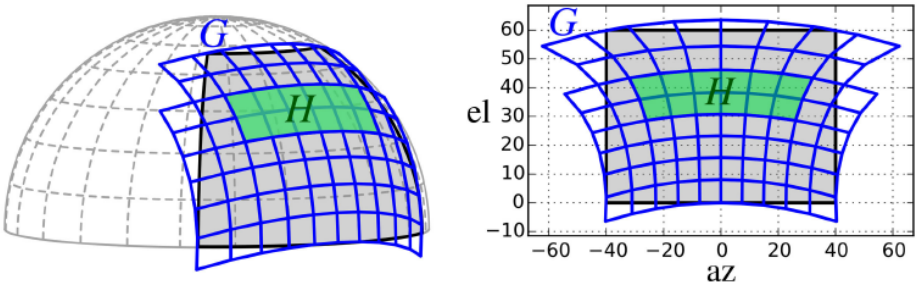


Figure 5.1: Example of Azimuth and Elevation Representation [81]

In this work I make the choice to represent the RRM space instead from a top down view of the geography (x and y Cartesian coordinates) for the surveillance space. The reasoning for this type of representation is due to two C2 driven factors and one assumption regarding mobile platforms (e.g. UAV drones). A visualization of the global scenario is shown in Figure 5.2.

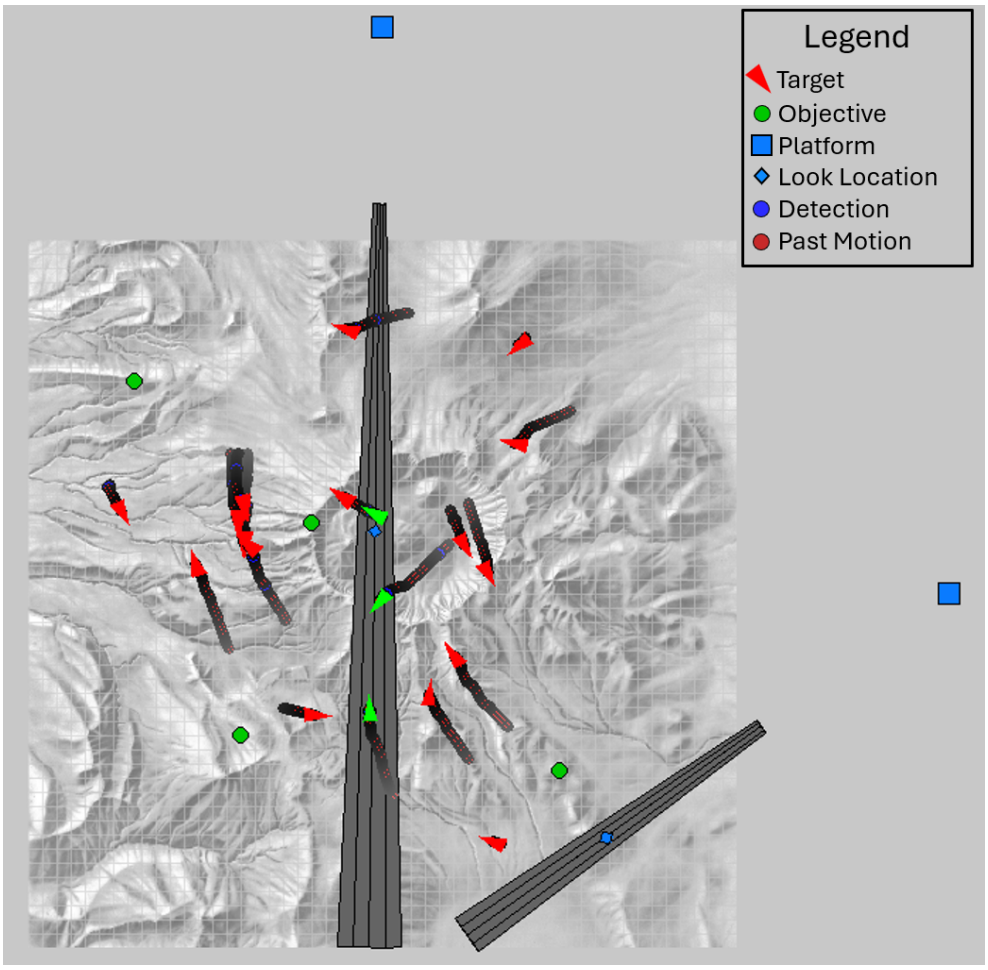


Figure 5.2: Utility Search C2 Scenario

The first C2 assumption is that the location of areas that need to be surveyed are going to be geographically located in the environment and relevant regardless of the radar action capability. For a single platform that is stationary these are the same but once platforms are going to coordinate they need to be able to understand the surveillance area with a global representation. The second factor is that these platforms are capable of moving and an azimuth-elevation selection will illuminate a different location from time-step to time-step depending on the motion of the platform. Finally we assume that targets of interest will have greater variance in their x-y coordinates as opposed to the z Cartesian dimension. Therefore elevation of a moving platform is

the more appropriate dimension to hold constant for a two dimensional C2 scenario.

The representation and testing in this chapter provides a contribution to the field of C2 surveillance where detecting targets of interest as they approach key objectives is more important then a full space search. Beyond the C2 relevance, these method can optimize search when the radar scheduler resources are insufficient to search the entire space. This design can optimize collaborative search by multiple agents in the same surveillance space. This collaboration can succeed both with and without real-time information sharing and is linearly scalable as the number of agents increase.

The following sections will present the utility map representation framework, the method I designed and implemented to utilize this representation, the comparison methods selected, and the testing procedure and results discussion. The following discussion will explain the details for how each sub-layer is calculated and will include my ideation to use radar principles to design the layers. The majority of the work detailed in this section is my contribution from concept to implementation. My research colleague, Timothy Sharp, contributed with the integration of the layers into the full simulation, the planning of the layer design and software implementation, and supported testing. I came up with the design of each layer, the framework and the search methods, implemented the majority of the layer code, integrated the system, and conducted testing. This work has been accepted and will be presented in the 2025 International Radar Conference [22] and was funded under the AFOSR grant with award number F4FGA03158J001.

5.2 C2 Design and Simulation

The scenario used for evaluating and testing the proposed methods is detailed in the radar simulation Appendix A. Specifically for this work the conference paper utilized the the GLRT with clutter implementation detailed in section A.3. Additionally,

testing was performed including platform motion with the most recent radar simulation which performed range-Doppler processing and accomplished detection with the cell-averaging CFAR method (Section A.4). The parameters in both cases did not change except that this work utilized a transmit 3dB beam width of $\theta_a = 0.5^o$ instead of the $\theta_a = 1^o$ utilized in Chapter 4.

5.3 Utility Search Representation

This search representation incorporates multiple strategies in a collaborative framework. It uses two separate layers: the utility layer and the information layer. The utility layer represents any layer that informs the agent regarding higher priority locations. The information layer is a decaying layer with a history of illuminations that are subtracted from the utility layer to encourage the platform to search new locations.

The approach to this C2 inspired search representation is to evaluate multiple aspects of the environment and a platform's location and how they impact the priority to survey space. To do this I have devised a search representation that incorporates multiple representation strategies together to gain benefit from each in a collaborative framework. This representation has two separate macro layers: the utility layer and the information layer. The utility macro layer represents any layer that informs the agent regarding higher priority areas or which will result in higher probability of successful detection. The information layer is a decaying layer with a history of illuminations that is subtracted from the utility layer to encourage the platform to search new locations. Before discussing specific layers and their design I will explain the general form of a layer.

There are multiple ways that a command and control space could be represented. As discussed in section 5.1, I chose to use a global top down x-y Cartesian coordinate

representation of a radar search space. Given that a platform can be placed anywhere around the exterior of the search space and that the platforms can move, I decided to use a simple grid discretization in x and y dimensions. For each individual platform, its radar beam will spread radially and thus not fit any grid perfectly, but a radial representation for any single platform would not be representative for other platforms located elsewhere. Given that all platforms will have this spread effect I choose to represent the space in a way that is easy to understand for a C2 operator. For the work found here all layers are the same 10km by 10km space with a grid size of 10m by 10m for a 1000 by 1000 grid map. Figure 4.2 shows a visual representation of the view geological map used in this work. Figure 5.3 is a representation of the utility

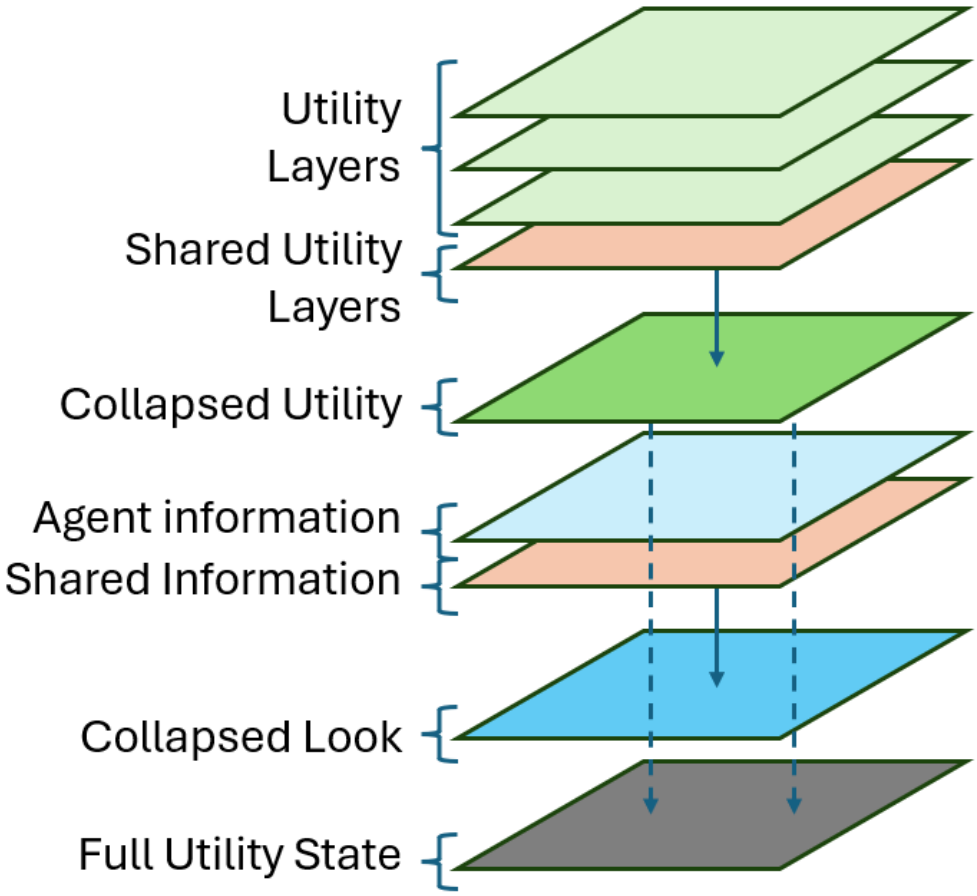


Figure 5.3: Collaborative Utility Search Representation Layers View

map frame work and will be used to aid in the description for each individual layer that follows.

The following sections will explain the reasoning and equations to calculate the utility or the information for each layer. These equations will often use the range and/or the angle of a point in the (x, y) plane compared to some point (platform, objective point, target prediction location). To do this efficiently the grid representation is saved as a mesh grid matrix where each location has identical columns of the x global location or rows of the y global location, respectively. These matrices are then used as the inputs to calculations used for future layer calculations. The common equations are shown in (5.1)

$$
\begin{aligned}
X &= \begin{bmatrix} [x_0, \ldots, x_0]^T & [x_1, \ldots, x_1]^T & \ldots & [x_n, \ldots, x_n]^T \end{bmatrix} \\
Y &= \begin{bmatrix} [y_0, \ldots, y_0] \\ [y_1, \ldots, y_1] \\ \vdots \\ [y_n, \ldots, y_n] \end{bmatrix} \\
R_{point,xy} &= \sqrt{(X - x_{point})^2 + (Y - y_{point})^2} \\
\theta_{point,xy} &= \arctan(\frac{(Y - y_{point})}{(X - x_{point})})
\end{aligned}
\tag{5.1}
$$

The two common calculations used in layer representations are the relative range ($R_{point,xy}$) and angle ($\theta_{point,xy}$) of each point in the grid to a source location.

As discussed earlier the two macro layers, utility and information, are the weighted sum of the other sub-layers in their layer group. This can include negotiated or shared layers from other platforms. The weighting variables are parameters used to tune the impact of local layers and are used for shared layers as an option for trust based confidence in future work. For the current implementation the weights are all set to 1. The two layers are the weighted sum of the other sub-layers in their layer group. For

this implementation the utility layer has a platform layer unique to each platform and an objective layer that includes C2 intelligence information about the environment. This can include shared layers from other platforms. The weighting variables are parameters used to tune the impact of local layers and shared layers. Once all the sub-layers are weighted and summed, the utility macro layer is normalized from 0 to 1.

The information layer retains a memory of the previous information layer which has a beta distribution based decay. For this implementation the default information layer is unique to a platform. For a collaborative representation the information layer will include the history from other allied platforms. The information layer is subtracted from the utility layer to create the final representation. The combined representation contains the available utility in the space and it used by the agent methods to drive search.

5.3.1 Platform Layer

The first layer in the collaborative utility representation is the platform layer. The idea for this layer was inspired by the work in [76] which took in account the position and capability of the platform to inform a utility measure of radar actions. The platform layer represents an estimated probability of detection if a target was present. To do this estimation, I decided to use a Swerling 1 model with a range equation based estimation. The following are the equations used to calculate the estimated probability of detection for the platform layer. The first calculation is to determine the signal SNR based of of radar parameters and range.

$$SNR(R) = \frac{P_t G_{tx} G_{rx} \lambda^2 \sigma}{(4\pi)^3 R^4 k T_s B_n L} \tag{5.2}$$

Where SNR is the signal to noise ratio, P_t is the transmit power, G_{tx} is the transmit gain, G_{rx} is the receiver gain, λ is the carrier wavelength, σ is the radar cross section

for the target, R is the range, k is Boltzman's constant, T_s is the system temperature, B_n is the noise bandwidth of the receiver, and L is the total system loss. In order to simplify the calculations and to appropriately incorporate the components of the phased array simulation, I decomposed the radar range equation into three components as:

$$SNR_{db} = 10\log_{10}(G_{loop}) + 10\log_{10}(\sigma) - 40\log_{10}(R) \tag{5.3}$$

Here the variable G_{loop} represents the loop gain and can be represented with the following subset of the range equation.

$$G_{loop} = \frac{P_t G_{tx} G_{rx} \lambda^2}{(4\pi)^3 P_n} \tag{5.4}$$

Where P_n is the noise power. We are utilizing a phased array so the calculation for several variables above include additional parameters. The calculations for these components are as follows:

$$\begin{aligned} P_t &= nG_{pc}P_{tx,ele}10^{G_{ele}/10} \\ G_{pc} &= BW\tau \\ G_{tx} &= G_{rx} = \frac{DBW}{az_{bw}el_{bw}} \\ P_n &= KT_s10^{F_N/10}f_s \end{aligned} \tag{5.5}$$

Where n is the number of elements in the phased array, G_{pc} is the pulse compression gain, $P_{tx,ele}$ the the transmit power of a single element, G_{ele} is the element gain, BW is the signal bandwidth, τ is the pulse width, D it the directivity of the array, az_{bw} and el_{bw} are the azimuth and elevation beam-widths, F_N is the noise figure, and f_s is the sample rate. Together all of these parameter use the radar range equation to calculate the estimated SNR at a given range R. Now that we have the estimated SNR we can use the Swerling 1 probability estimation method from [32]:

$$U_{plt} = P_D(R) = P_{FA}^{(\frac{1}{1+SNR(R)_{lin}})} \tag{5.6}$$

Where P_D is the probability of detection, P_{FA} is the probability of false alarm, and SNR_{lin} is the linear version of the SNR value. For the research in this section table 5.1 has the components that were used for implementation and testing.

Parameter	**Value**	**Parameter**	**Value**
Array elements	16×16	Center freq.	2 GHz
Sample rate	100 Mhz	Bandwidth	30 MHz
Pulse width	$1\ \mu$ s	PRF	8.5 khz
Max Power	6 W	Beamwidth	$\theta_a = 0.5^o$
n	16	Sample Support	64
P_{FA}	10^{-5}	τ	$1 \cdot 10^{-5} sec$
T_s	$290K$	Max Velocity	318 m/s
f_s	$100 \cdot 10^6 Hz$	F_N	$4db$
D	2600	β	$3 \cdot 10^5 Hz$
$P_{tx,ele}$	$3dbW$	G_{ele}	$10db$
K	$1.38 \cdot 10^{-23}$		

Table 5.1: Radar Parameters for P_D Estimation & Swerling-1

Using these parameters resulted in the following P_D curve (Figure 5.4) given range for the radar space I will use in testing.

Additionally, the plot includes the Monte Carlo results from the radar simulation using a Doppler frequency based clutter and STAP detection with the GLRT as described in section A.3. Included are the results for different P_D given the impact from the radial velocity clutter as it approaches the $90°$ orthogonality. The utility representation cannot know the actual P_D but it does show that it is a relatively good estimation and should serve to drive platform based utility. Figure 5.5 is a visual representation

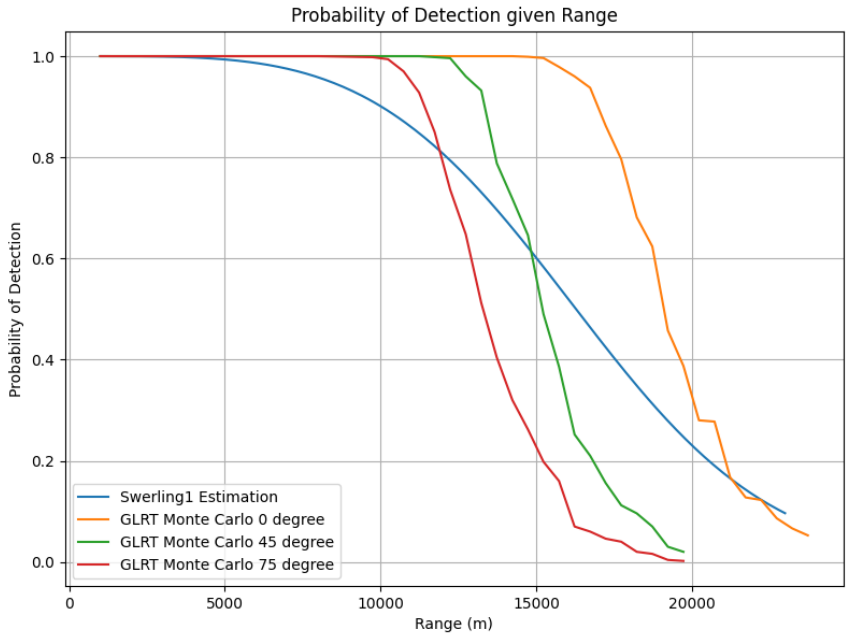


Figure 5.4: Probability of Detection for Estimator and Actual Simulation

of the platform layer for a sample scenario where the platform is at to location [13000, 5000].

5.3.2 Objective Layer

The second layer in the collaborative utility representation is the objective layer. The objective layer concept was inspired from the C2 influence on a surveillance scenario. A core inspiration for the objective layer was a combination of two different methods from the literature. The first source was the work of [77] which represented locations in the search space as potential generators of targets of interest. The second was the work in [79] which utilized a top down visualization of the search space and modeled search locations with entropy based off a probability of target presence.

For the design of the objective layer, I chose to represent the information of objective points that are of defensive value. This is the inverse of the generator concept,

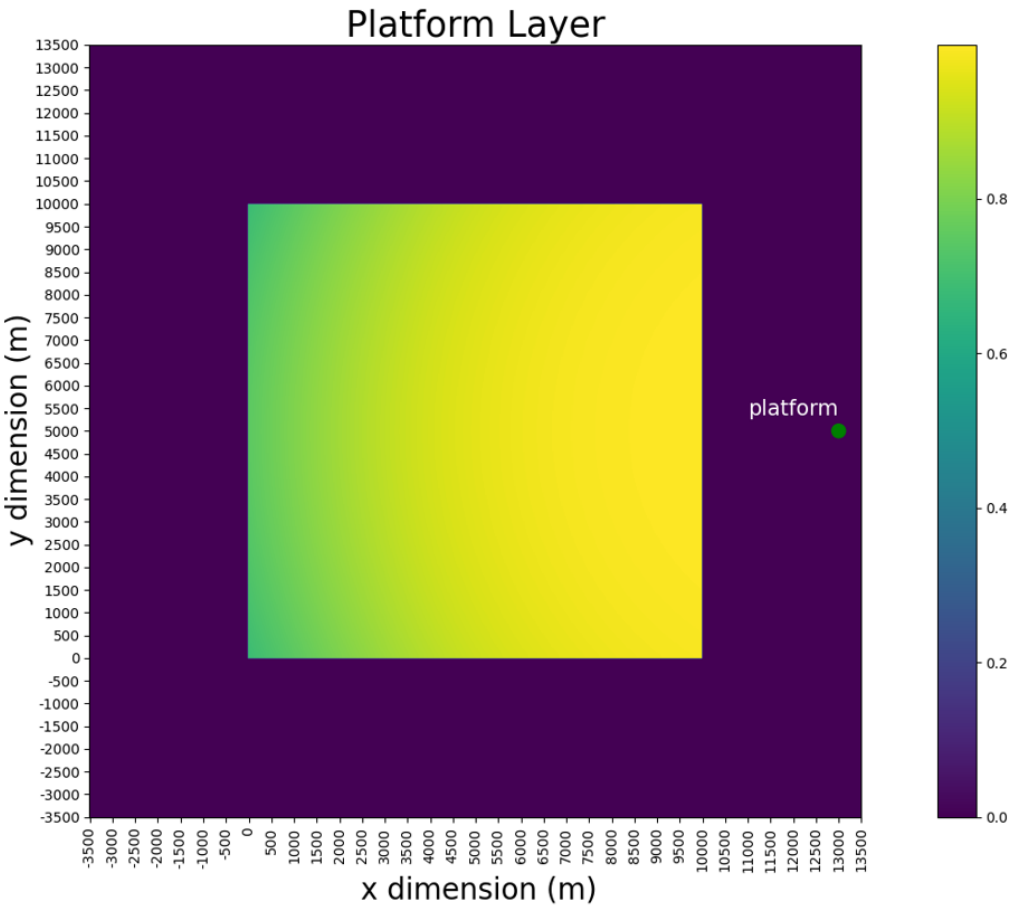


Figure 5.5: Platform Layer with Swerling 1 P_D Method

as it is instead areas that I assume targets of interest would approach. This motivation challenges the assumption that the entire search space has equal priority. Instead, it assigns higher priorities to identify targets approaching objective points. To represent the need to survey around the objective points, I designed a utility offset to drive action around them. I experimented with both a Gaussian and exponential representation for the utility around the objective points. I decided to use a Gaussian offset following a review of the work by [75], as McGee et al. utilized an assumption about a target's maximum velocity to design a geometric pattern that could assure the target could not escape some space. The idea for the Gaussian offset in the objective layer is that, given an assumption regarding a potential target's maximum approach velocity, we can set

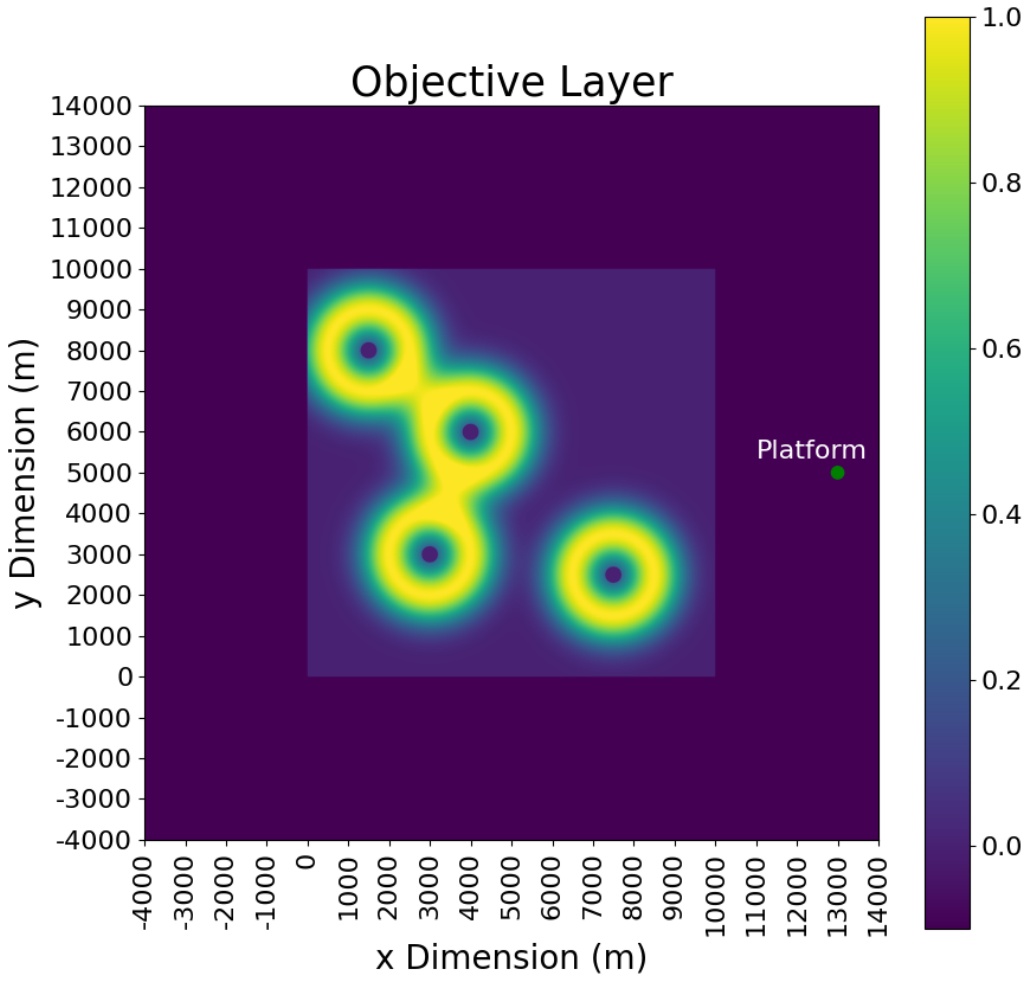


Figure 5.6: Objective Layer with Gaussian Utility

a mean offset from the objective point and a standard deviation that should maximize the probability of surveying the target before it reached an objective point. A visual representation of the objective layer is shown in figure 5.6, and the foundational equations are:

$$\begin{aligned} \mu_{obj} &= \frac{4 * V_{\mu} + R_{blind}}{\Delta t} \\ \sigma_{obj} &= \frac{2 * V_{\mu} + R_{blind}}{\Delta t} \\ U_{obj,xy} &= N(R_{obj,xy} - \mu_{obj}, \sigma_{obj}) \text{ see (5.1)} \end{aligned} \tag{5.7}$$

Where V_μ is the mean expected velocity of the targets, R_{blind} is a blind range where the target has reached the objective point, Δt it the number of radar actions per second, $R_{obj,xy}$ is the range for an (x, y) pair from the objective point, and $U_{obj,xy}$ is the utility due to the objective point at a given (x, y) location. For the work in this section target velocity is between $V = [100, 300]m/s$ with $V_\mu = 200m/s$ and $R_{blind} = 200m$.

5.3.3 Utility Layer

The utility search representation is design to allow any number of personal sub-layers for a platform (platform layer, objective layer, etc.) and other shared layers form collaborative platforms (other objectives). Once all of the layers are generated of shared then the macro-layer, utility layer, is collapsed and normalized. The process to collapse and normalize the layers is:

$$\begin{aligned} U_{util} &= \sum_{i=0}^{layers} U_i \\ U_{util_norm} &= \frac{U_{util} - min(U_{util})}{max(U_{util}) - min(U_{util})} \end{aligned} \tag{5.8}$$

A visual representation of the combined utility layer is shown in Figure 5.7.

Before incorporating the information layers, this collapsed and normalized layer, biases the utility for both the estimate of the platform's performance and the C2 knowledge based areas with a priority to survey. The combined utility will incentivize radar actions that illuminate as many objective regions as possible and prioritize looking closer given the increased P_D.

5.3.4 Information Layer

The information layer represents the information gained by previous actions performed by a platform. It utilizes the beam width information derived from the range

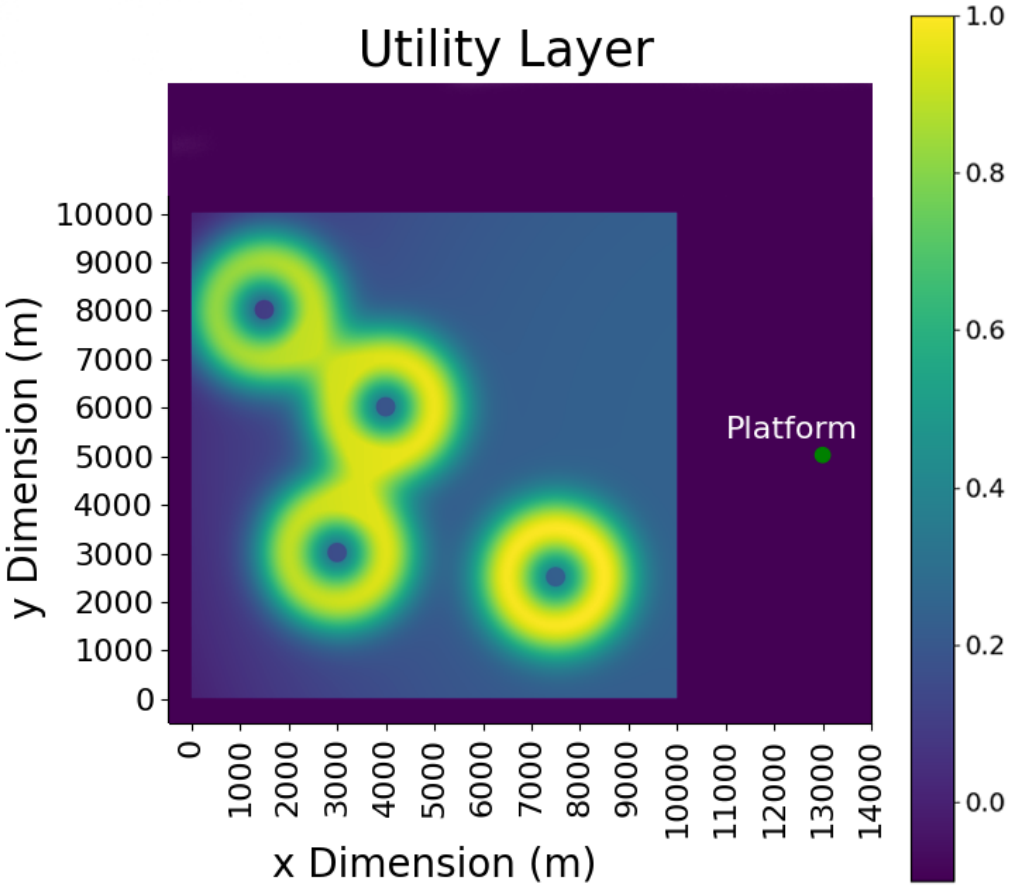


Figure 5.7: Normalized Utility Layer

and beam splitting in Table A.4. Two matrices representing the map grid (1000 x 1000) calculate the information layer each time an action is selected. A time matrix, $\mathbf{T}$, and a beam splitting mode matrix, $\mathbf{B}$, are updated at all indices within the 3db beam width cutoff for a selected action. New actions, that overlap indices already containing values, override previous values.

The information layer is inspired by how the illumination of a region lowers the entropy of that space [79]. Similar to the way entropy grows over time in many systems, this information should decay with time and incentivizes searching new spaces with platforms revisiting previously observed locations after an appropriate amount of

time. This layer included a lot of exploration regarding how to represent the balance of exploring new space and revisiting high utility locations that were previously illuminated. Because the information layer is subtracted from the utility layer this can be represented by a decay in the weight of the previous illuminations. I decided to use the platform layer values, but only for the points that are within the illumination beam. The 3 dB beam and the azimuth angle are used to draw the outer lines of the beam. Then the points between the lines are updated on the information layer matrix with planform layer values. Since the platform layer is the estimated P_D, it was logical that it also represented the probability of finding the target in an illumination and would fairly represent the gain in certainty due to an illumination given range.

The decay was the first major area to explore and multiple iterations before the final method was designed. The first decay methods were a linear decay and a multiplicative decay given time. The initial decay method was chosen because it does not require a history of the map and was easy to implement. To the linear decay method a rate $0 \geq \epsilon_{lin} \geq 1$ is chosen based on how quickly the refresh rate should be. The naive decay could also be formulated as $\epsilon_{lin} = \frac{1}{\eta_{reset}}$ where η_{reset} is the desired turns to look again for full utility of 1. This was simple to implement as the entire map could subtract ϵ_{lin} and then any negative values would be set to 0. Similarly, the multiplicative method would have a decay rate $0 \geq \epsilon_{mult} \geq 1$ and then multiply the entire map by $1 - \epsilon_{mult}$. This would slowly decay the values toward zero. The challenge with both of these methods came when trying to tune the parameters. If you refresh too often then the agents look at the same few locations and do not explore. Conversely, if the rate was too low then the agents will assume an early area has been illuminated and not look again until the whole space is explored. In fact if ϵ_{lin} or ϵ_{mult} is zero then the method becomes an ordered search of the space with no priority besides order.

At this point I discussed the issue with my research colleagues and we explored options. I expressed that, ideally, I wanted to refresh searching areas of C2 importance

more often but still not immediately search space recently illuminated. My colleague Dr. Stringer suggested that the beta distribution [125] could balance two variables overtime. I dug into the beta distribution and found that its parameters worked perfectly when mapped to variables already used in the scenario. The equation for the beta distribution is shown in (5.9):

$$f_{beta}(x, \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1} \tag{5.9}$$

The key parameters of the beta distribution are the input variable $0 \leq x \leq 1$ and the shape parameters $\alpha, \beta > 0$. When x approaches 0, the β component dominates the distribution, and conversely when x approaches 1, α will dominate. The independent variable $0 \leq x \leq 1$ would drive a change in the beta distribution and mapped well to the time since the last illumination. But this variable needed to range from 0 to 1 so the following is a time based translation for all x points within $\mathbf{X_{x,y}}$ as a matrix of all points in the representation with this normalized time difference.

$$\mathbf{X_{x,y}} = \frac{t_0 - \mathbf{T_{x,y}}}{t_0 - \mathbf{T_{x,y}} + t_{max}/2} \tag{5.10}$$

Where t_0 is the current time and the variable t_{max} is the upper limit in time at which the decay will be 1 or greater. The matrix $\mathbf{T}$ is the same size as the representation and includes the time that those locations were illuminated. This matrix is initialized with a negative number (-1), and then, each time an illumination action is processed, the illuminated grid points are set to the current time. This is an efficient way to keep track of time change for the entire representation as only the matrix is saved and only illuminated points are updated. The α and β parameters are defined as:

$$\alpha = 2 \quad \beta = .5 * U_{x,y} + 1 \tag{5.11}$$

The α parameter is a constant that will cause full decay as x approaches t_{max}. The β component plays an important role as it uses the current utility (platform and

objective) at each point to decay the information content at different rates. When the utility at any point is maximum (i.e., 1), the beta distribution will fully decay at $x \approx \frac{t_{max}}{2}$. A point with a utility value of 0, will decay completely at t_{max}. This causes the decay to be dynamic with utility value as the decay is low if time since last illumination (x) is near 0 but then balance out as time approaches t_{max}. The dynamic decay behavior is desirable as a platform utilizing this decay method will refresh areas of higher utility (prior knowledge) more often. The decay is then used by the full information layer (5.12):

$$I_{x,y} = [1 - c_{beta} f_{beta}(x, \alpha, \beta)] \cdot U_{plt}(\mathbf{B}) \tag{5.12}$$

Where c_{beta} is a weighting constant ($c_{beta} = .75$) to cause maximum decay at t_{max} ($t_{max} = 8s$) and $U_{plt}(\mathbf{B})$ is a Swerling 1 P_D (5.6) estimation for the matching power level given beam splitting mode $\mathbf{B}$.

The final component of the information layers is the ability to utilize the shared illumination history for collaborative platforms. Collapsing of the the layer that scales with any amount of collaborative agents is accomplished by taking the maximum information at each point between all information layers.

$$I_{collab} = \max_{i=1 to N} I^{(i)}_{x,y} \tag{5.13}$$

A visual representation of the Information layer is shown in figure 5.8 where two agents are sharing information layers after 10 actions. The effect of the beta distributed decay can be seen as the objective points from Figure 5.7 refresh sooner. The combination of the utility and information layers for the full representation ($U_{full} = U_{util} - I_{collab}$) is shown in Figure 5.9:

This visualization illustrated the balance of exploration and exploitation provided by the beta distribution. Effectively, the regions with greater utility will increase faster,

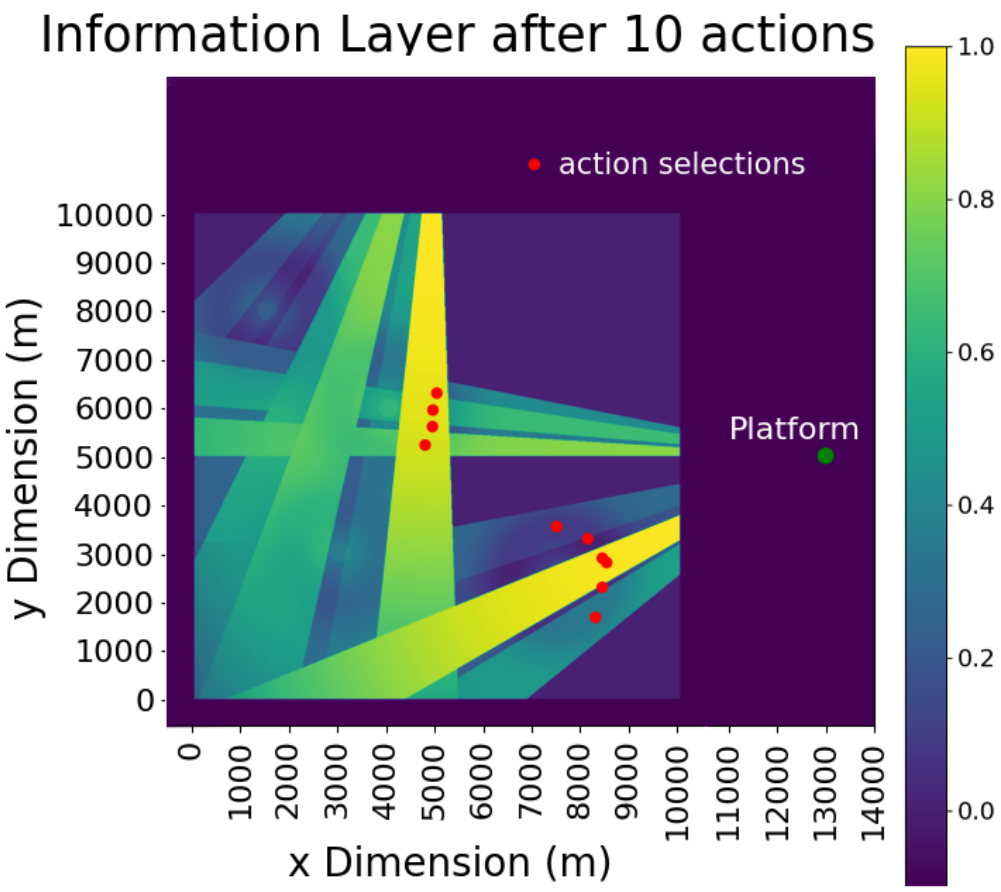


Figure 5.8: Information Layer

given the decay in the information layer. The dynamic beta distribution decay causes a utility maximizing searcher to refresh areas of high utility often but to search new locations with remaining resources.

5.4 Action Selection Methods

This section discusses an action selection method employing the utility representation and multiple comparison methods. The primary method used with the full utility representation is a two step maximization method which optimizes the location that will illuminate the greatest total utility (within a utility representation). Initially this

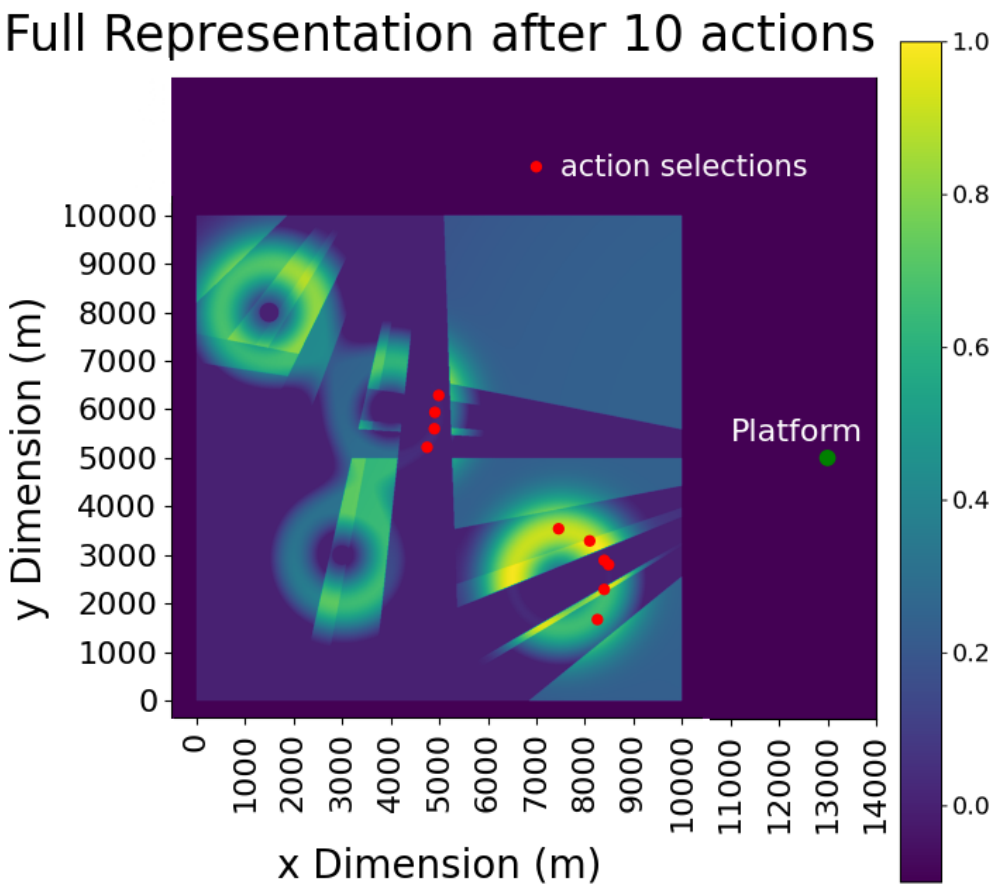


Figure 5.9: Full Utility Representation

was accomplished by scanning the possible utility of a targeted beam at each discretized cell within the map. That proved to be very computationally expensive. In order to reduce the computational cost, I worked with my colleague, Mr. Sharp, to use a pooling method to determine the general area of utility concentration. Then the local area was searched for each action to maximize utility in the local region.

The utility evaluation selects a cell in the map (x_i, y_j) and generates the look information for that beam ($I_{x_n,y_m} = \sum_{i,j}^{i\in n, j\in m} I_{x_i,y_j}$), where (X_n, y_m) are all index pairs within the beam. The beam information uses the same Swerling 1 P_D estimation from the platform layer section. This beam value acts as a weight that multiplies with the

current utility representation as:

$$U_{B:x_n,y_m} = \sum U_{plt:x_n,y_m} * I_{x_n,y_m} \tag{5.14}$$

After calculating all x,y points in the map, the point with the highest total utility was selected for action selection. This method selects the point that generates the most utility, but is computationally costly for all one million points within the representation matrix of size 1000 x 1000. This led to a two-step solution that used average pooling as preprocessing to locate an action in the top 1% of utility coverage.

Average pooling is used in image processing and machine learning to reduce the size of a large matrix or image by averaging values within a pooling window to create a smaller matrix. The pooling size was set to $p = 10$, which resulted in a reduction of 10 in both dimensions, creating a matrix of size 100 x 100, with each point representing a 10 x 10 location in the original representation. Using this pooled matrix, the maximum cell of the $10,000$ available cells was identified. Then, the utility calculation above was calculated for the 100 points in that subspace, and the point with the greatest utility was selected as the action point. This two-step method was 182 times faster than the original and found a solution with $< 1\%$ error compared to the action with maximum coverage. This maximization method was performed on two variations of the utility representation. The first used a representation for each platform with only the information layer from their own actions. The second utility representation explored the option to share information layers between radars. A visual representation of the pooling operation is shown in Figure 5.10.

Two raster methods were utilized for comparison. The first was a full raster of the 10k x 10k space. This raster method utilized the three beam-splitting modes and, for each mode, swept the angle in a clockwise motion between the ends of the map with an increment of the total 3dB beam width. This is shown in Figure 5.11 as this method will radially cover the map space at each range mode.

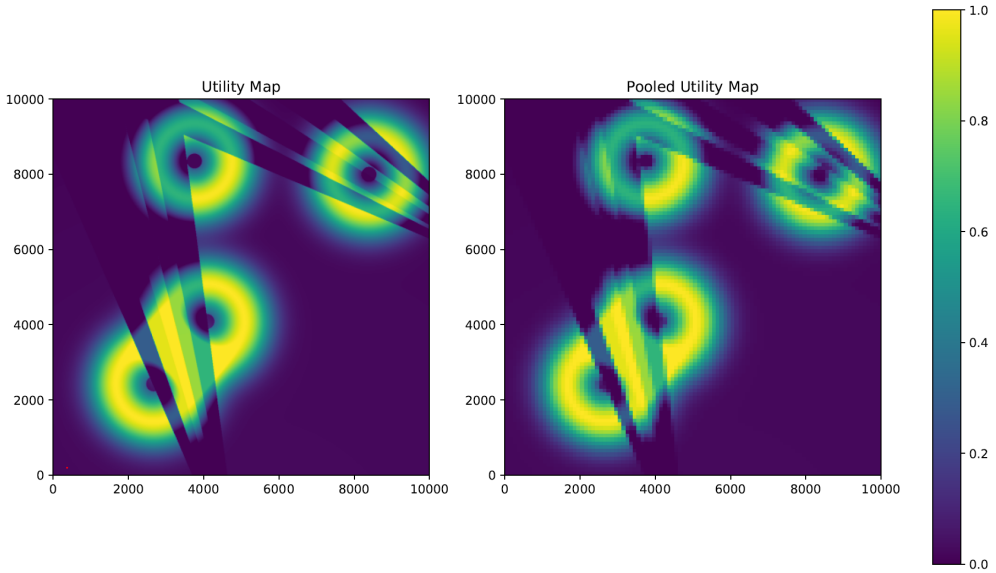


Figure 5.10: Example of Average Pooling

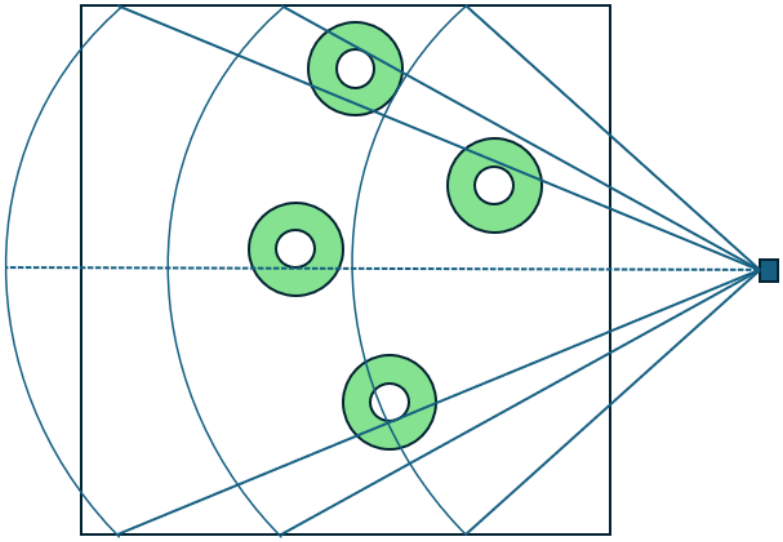


Figure 5.11: Raster Scan of the Full Space

The second raster method (focused raster), utilized the prior information proved by the objective points. For this raster, the radar would select one of the objective points with a uniform random chance (without replacement). Once an objective point was selected, the focused raster would calculate the point with an offset of $\mu_{obj} = 1200m$

and $\pm 90^{\circ}$ from the angle between the radar and the target objective point. Using these edge points, a similar span of raster angles was swept. The raster scan was performed for each of the three beam width modes in a clockwise motion. A visualization of this method is shown in Figure 5.12.

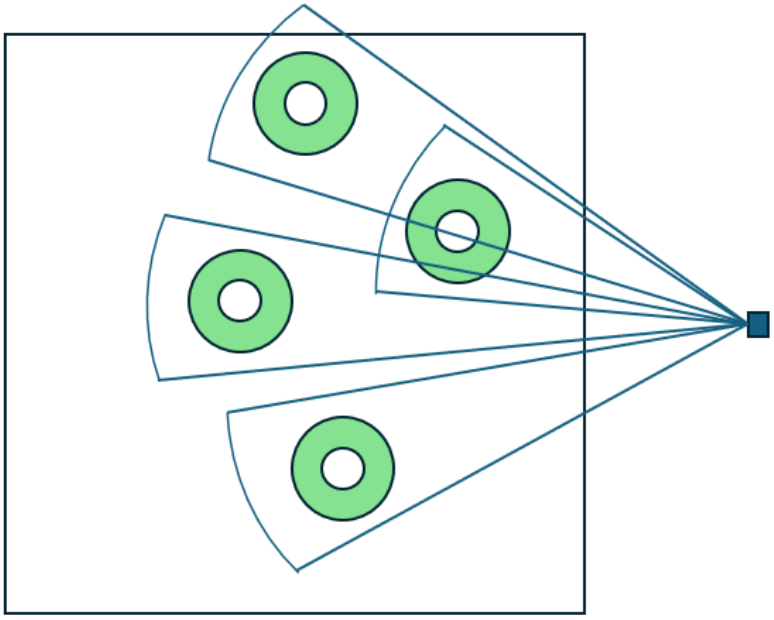


Figure 5.12: Focused Raster Scan of Objective Points

5.5 Utility Search Results and Discussion

The testing of the utility representation was conducted with two scenario and radar simulation versions: the GLRT version (Section A.3) and the range-Doppler processing version (Section A.4). Both scenario simulations implemented the utility representation and action selector, with and without collaboration, and the two raster comparison methods. The baseline comparison had stationary platforms. Additional testing was conducted with only the range-Doppler processing simulation that had the platforms moving in a circular pattern around the main viewing area. The following section will describe the testing set-up for both the stationary and the moving platform

surveillance scenarios.

5.5.1 Test Scenario Set-up

The testing for both the utility representation and the action selection method was performed in two medium-fidelity gym environments presented in Section A.3 and Section A.4. Twelve scenarios were run to explore the impact of the four action selection methods (full raster, focused raster, utility selection, collaborative utility selection) with three amounts of radar platforms present (1,2, and 4). For each scenario, the physical space is restricted to a 10k by 10k space. Up to four platforms are initialized in four locations outside the simulation space ([5000, 13000], [5000, -3000], [13000,5000],[-3000, 5000]). Each platform had a random, unique position from these options. Four objective points are randomly generated between $x = [1000, 9000]$ and $y = [1000, 9000]$, and no objective points were within $1000m$ from each other. Each scenario initializes with a number of targets equal to twice the number of radar platforms, and then at each time-step, each objective point has a probability of generating a new target ($P_{gen} = .25$), continuing until 30 targets are generated. Targets randomly select an objective point and are spawned at a random point within 4 km of the objective point. Each target travels at a constant velocity (200 m/s) toward the objective. At each time step, the platforms select an action coordinate, and a beam with the corresponding beamwidth and power is generated (table A.4). Detection is accomplished with the GLRT or cell averaging CFAR depending on the simulator being used. If the detection method determines a detection and a target is present (not a false alarm), then the target records the time it was found and its distance from its objective point. Targets within 200 m of the objective are recorded as missed. The simulator continues to run until all targets are detected or arrived at the objective.

Three metrics are stored from these scenarios: the number of targets found, distance from the objective point, and the percent of the utility representation coverage at

each time step of the scenario. 100 unique configurations of random objective points were generated and tested across the twelve scenarios with randomly selected radar platform locations. For each scenario the mean and standard deviation of the metrics for all 100 configurations were analyzed and presented in Section 5.5. The distance metric only track detected targets (missed targets would be $< 200m$ and bias the results). The utility metric was the mean utility after time step 10 for each configuration. Turn 10 was selected given plots of that utility over time showed that utility began at 0 and grew. Following turn 10, all methods had stabilized and the mean represents the effective utility provided by the search method.

This work was originally run with the GLRT simulation and accepted for publication at the international radar conference 2025 [21]. Additional testing was conducted with the final radar simulation which generated a transmit signal and conducted range-Doppler processing and used the cell averaging CFAR method for detection. This method also used a clutter profile that included a clutter ridge. Using this scenario simulation I did additional testing which included the motion of the platforms. For this testing the radar platforms moved in a circular motion with a constant radius of 8000m from the center of the scenario map. The platforms all moved in a counter-clockwise direction with a constant velocity $V_p = 100m/s$ tangential to this circular path. The goal for this motion based testing was to explore the effectiveness of the utility representation and the comparison raster methods when moving. Typically a raster method assumes a stationary position to ensure that the intended space is covered. If the platform is moving then one of two scenarios happen to the raster actions. The following figures 5.13 show the raster actions and what happens if the platform moves in the same radial direction or in the opposite direction.

The results and discussion will expand on what is seen in the plots. The process of a raster scan with motion will result in a heavy overlap on a focal point based on the rate of raster and the motion of the platform (as seen in the opposite motion plot). If

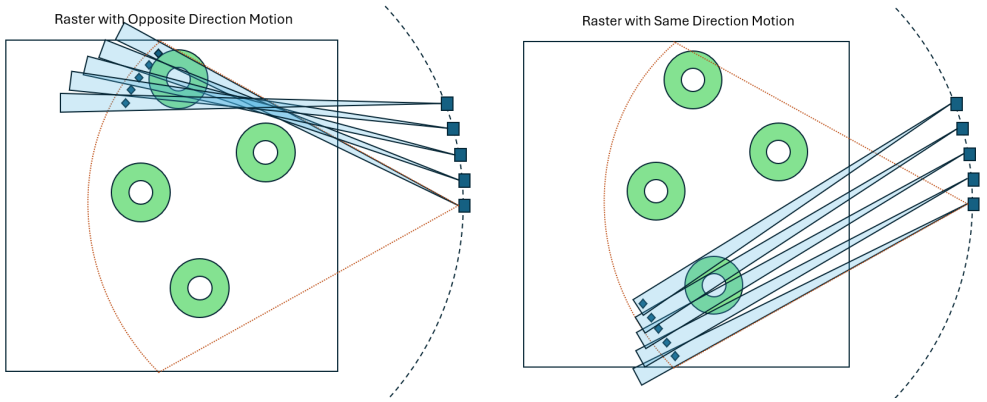


Figure 5.13: Raster with Motion

the motion is with the direction of the raster then there is less overlap but areas of the raster that should be covered will not be covered. The motion based impact on raster methods will be discussed and observed in the following results.

5.5.2 Results and Discussion

This results section will discuss the three core metrics: targets found, distance of target from its objective, and utility score. Within the discussion of each metric I will begin with the results from the GLRT simulation published in [22] followed by the new results that includes the motion testing. The targets found from 30 total in a scenario is shown in Figure 5.14 for the GLRT simulation.

The results from the scenarios indicate multiple interesting and promising trends. Observing the targets detected, the utility methods consistently found 10-20% more targets than the raster methods, with the greatest gain in the two-platform scenario. These results confirm that the use of both prior C2 information in the objective layer and the encoding of the platform location in the platform layers had an impact. This representation includes layers that drive searching priority C2 locations (objective layer), prioritizes beams with higher P_D (platform layer), and balances searching new space versus rechecking high activity locations. The focused raster also used prior

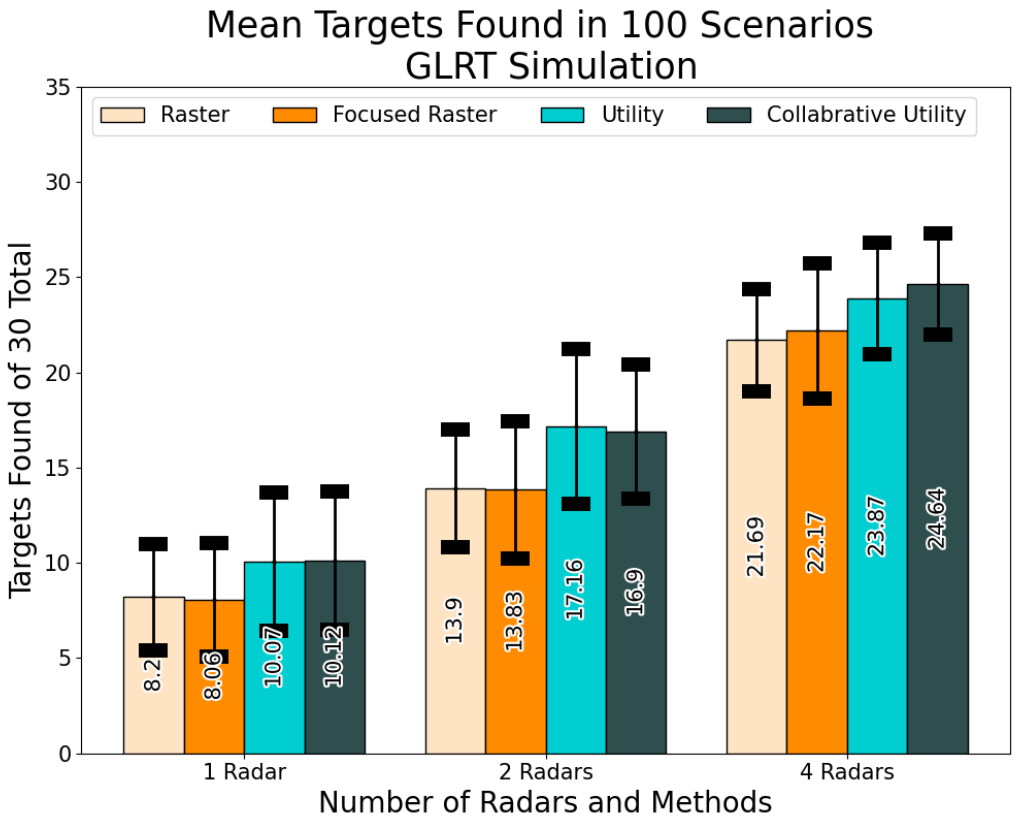


Figure 5.14: Target Metric for GLRT Simulation

information, but the utility method's platform layer drove agents to search objective points closer to themselves and/or select beam actions that would cover multiple objective points. The full raster method would maximally explore the space but not use any C2 information. The success of the utility method is that both exploration and exploitation of prior information enable more efficient search. The targets found results from the range-Doppler simulation for both the stationary and circular motion is shown in Figure 5.15.

Figure 5.15 shows the number of targets found out of 30 total in the scenario. The first promising result is that the results for this simulator differ from the first testing but the general trend remains that the utility method find more targets than the two raster methods. In order to better understand the difference in performance, Table 5.2 shows

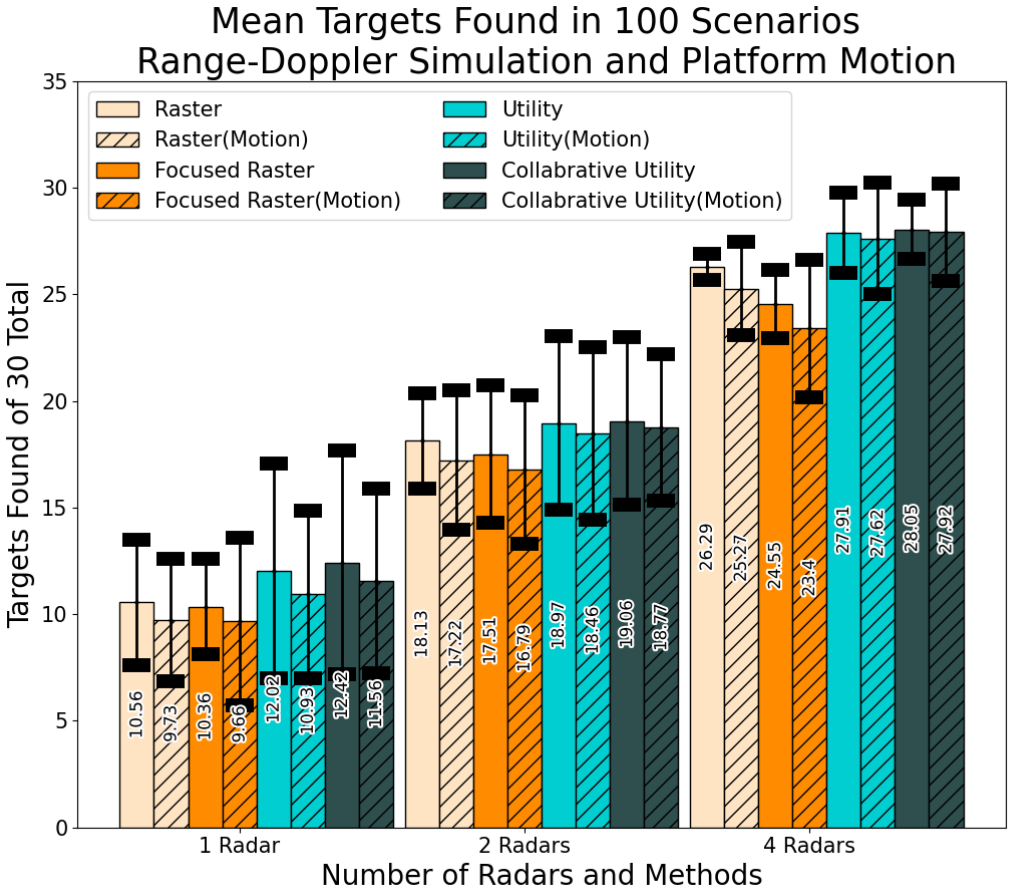


Figure 5.15: Target Metric with Motion for Range-Doppler Simulation

the percent difference from the lowest performing method. This percent difference is calculated as $D = \frac{|M_i - M_{min}|}{M_{min}} \cdot 100$ where M_i are each method and M_{min} is the lowest scoring method.

These difference percentage values help with understanding the impact of the methods as well as the platform motion. The first observation is that the utility methods perform better $(7 - 15\%)$ then either raster method, with the focused raster as the lowest performance. It is important to point out that all methods were tested with the same random scenario configurations.

The motion causes each method to perform worse than the same method when

Search Method	**Percent Diff-Min** 1 Radar (static, moving)	**Percent Diff-Min** 2 Radars (static, moving)	**Percent Diff-Min** 4 Radars (static, moving)
Full Raster	1.9%, 0.7%	3.5%, 2.6%	7.1%, 9.9%
Focused Raster	0%, 0%	0%, 0%	0%, 0%
Utility	16.0%, 13.1%	8.3%, 9.9%	13.7%, 18.0%
Collab Utility	19.9%, 19.3%	8.9%, 11.8%	14.3%, 19.3%

Table 5.2: Percent Difference Targets Found from Minimum Method

Search Method	**Loss From Motion** 1 Radar	**Loss From Motion** 2 Radars	**Loss From Motion** 4 Radars
Full Raster	−7.86%	−5.02%	−3.88%
Focused Raster	−6.76%	−4.11%	−4.68%
Utility	−9.07%	−2.69%	−1.04%
Collab Utility	−6.92%	−1.52%	−0.46%

Table 5.3: Percent Target Found for Motion

stationary. But the utility method show a gain in performance compared to the raster methods. This means that the utility methods are much more resilient to the impact of platform motion. The main reason for this is that the utility methods determine the highest utility action at each look and the platform layer is calculated for the new position. This motion was in the same direction as the raster (see Figure 5.13) and as shown in that figure had a reduction in effectiveness for the raster methods that was much greater than the impact on utility methods. This is the better scenario for area coverage and we would expect a motion in the opposite direction to be more punishing. Table 5.3 shows the decrease in performance for each method when motion is included. The impact of motion decreases with more surveillance radars, but the

utility methods have much less loss due to motion.

One final observation is how close the performance of the two utility methods are. The collaborative utility methods has the radars share their action and performs better as it encourages the platforms to not waste resources searching recently illuminated space. But the agents without sharing are close in performance. This similarity is due to the influence of the platform layer for each radar and the distance of the platforms from each other (over 11 km and 90-180° relative to the map center). The default method selects actions that have higher expected P_D, which given their distance, will be unlikely to overlap and thereby waste collective utility. This is desirable because, in a contested environment, the further platforms are from each other the more likely delay in communication or denied communication becomes. In that scenario they will still perform well as they are unlikely to illuminate the same space. If the radars are closer then they would need to share looks to avoid wasting resources but they are also more likely to establish communication.

For the distance metric, the larger the number, the sooner the target was found. This is a desirable result, as earlier detection allows more time for a response. The results for the GLRT Simulation are shown in Figure 5.16. An interesting trend is that for the single platform case, the full raster finds targets earlier. This is expected, as it searches space further away from the objective points, but as the number of platforms increases, it is the utility methods and the focused raster that increase the distance of targets found. Logically, a single platform covering all objective points will have to find targets closer, given it must cover all of the objectives. As the number of platforms approach the number of objectives, the utility methods will automatically divide up the objective points and improve both in the number of targets found, and find them at further distances.

The distance for all targets including those that are not detected (with a distance of 0m) is shown in Figure 5.17. This represents the distance biased by the number

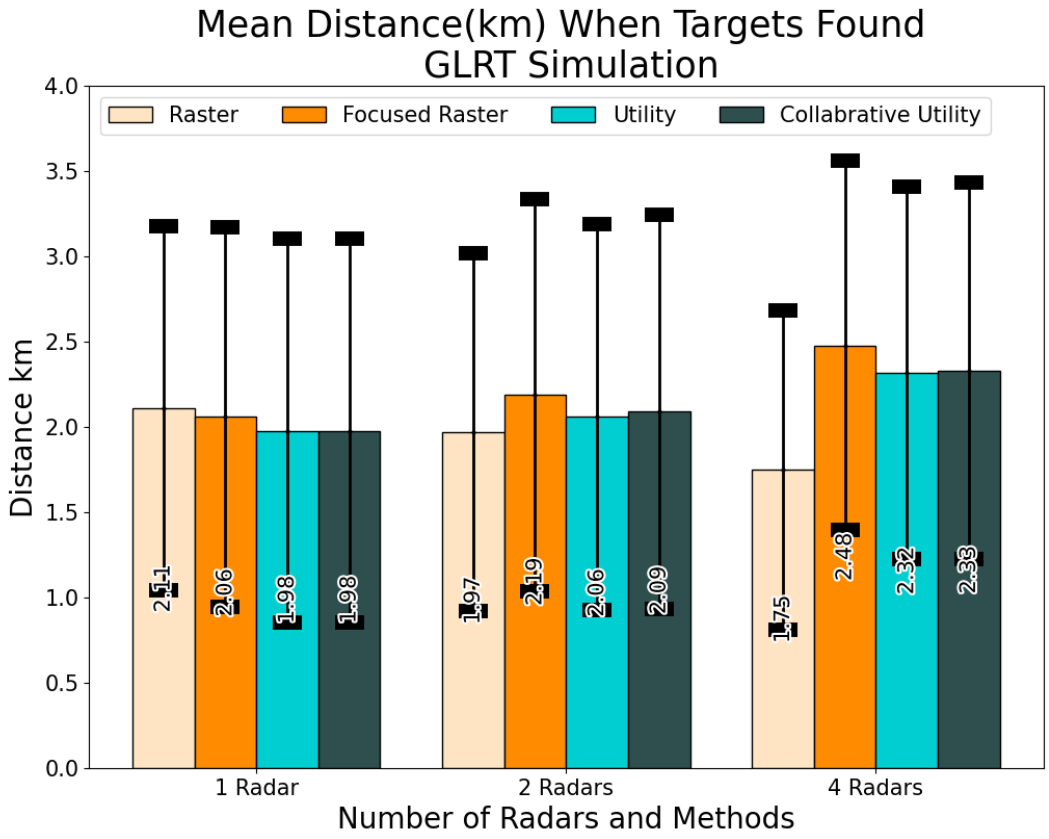


Figure 5.16: Distance Metric for GLRT Simulation

of targets found and provides insights into the realistic distance when target success is included. The standard deviation is not presented as the inclusion of the missed targets, with a distance of 0, causes large standard deviations that do not contain useful information.

The distance metrics for the range-Doppler simulation show the same trend as seen in the published testing. The impact of platform motion seems to be minimal regarding the distance that targets are found. One aspect of these results is that the variance is very high for all methods, but the 4 radar scenario does show that the focused raster and the utility methods find targets further from the objectives.

Similarly to the original simulation result, Figure 5.19 is the distance for all targets

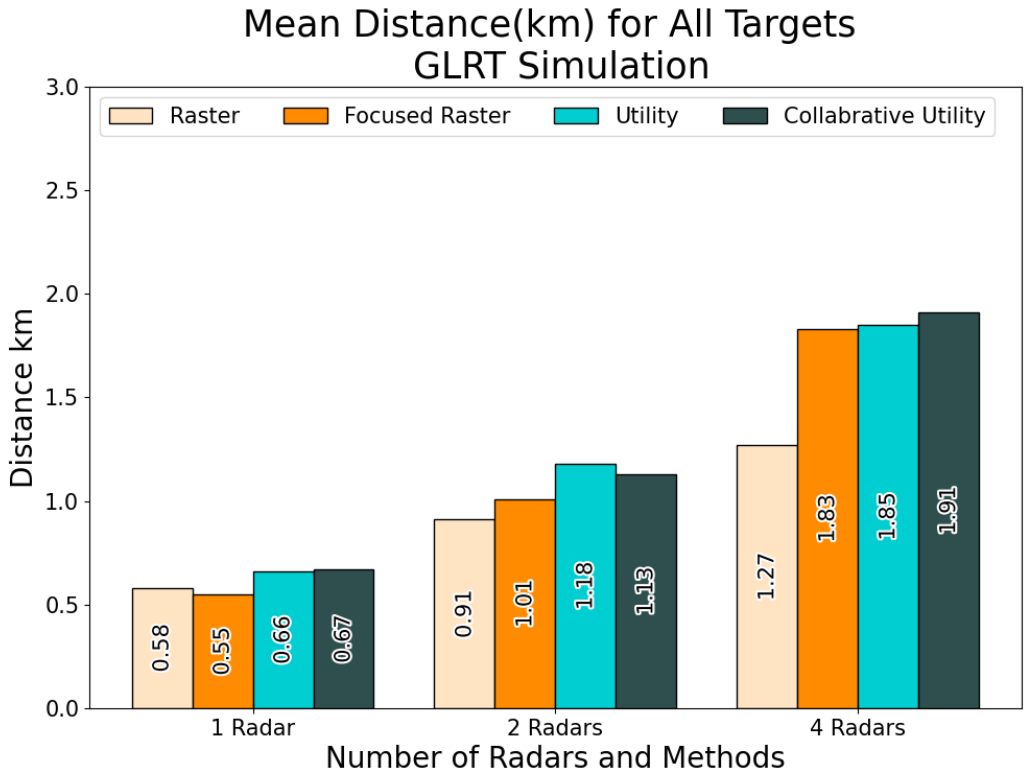


Figure 5.17: Distance Metric for All Target for GLRT Simulation

including those that are not detected.

The utility metric is a measure of the effectiveness of each method to reduce the utility in the representation. As expected, the method that maximizes utility succeeds compared to the traditional raster methods. The utility results for the GLRT are shown in Figure 5.20 and the results for stationary and motion testing with the range-Doppler simulation are shown in Figure 5.21

The single platform case is a baseline and indicates a trade-off between the targets found vs the effective range. The exploratory method, full raster, finds targets at the furthest distance but finds fewer than the objective point-focused methods (focused raster and the utility selectors). As the number of platforms increase, the utility method increases in both targets found and distance found over the full raster, but the margin is

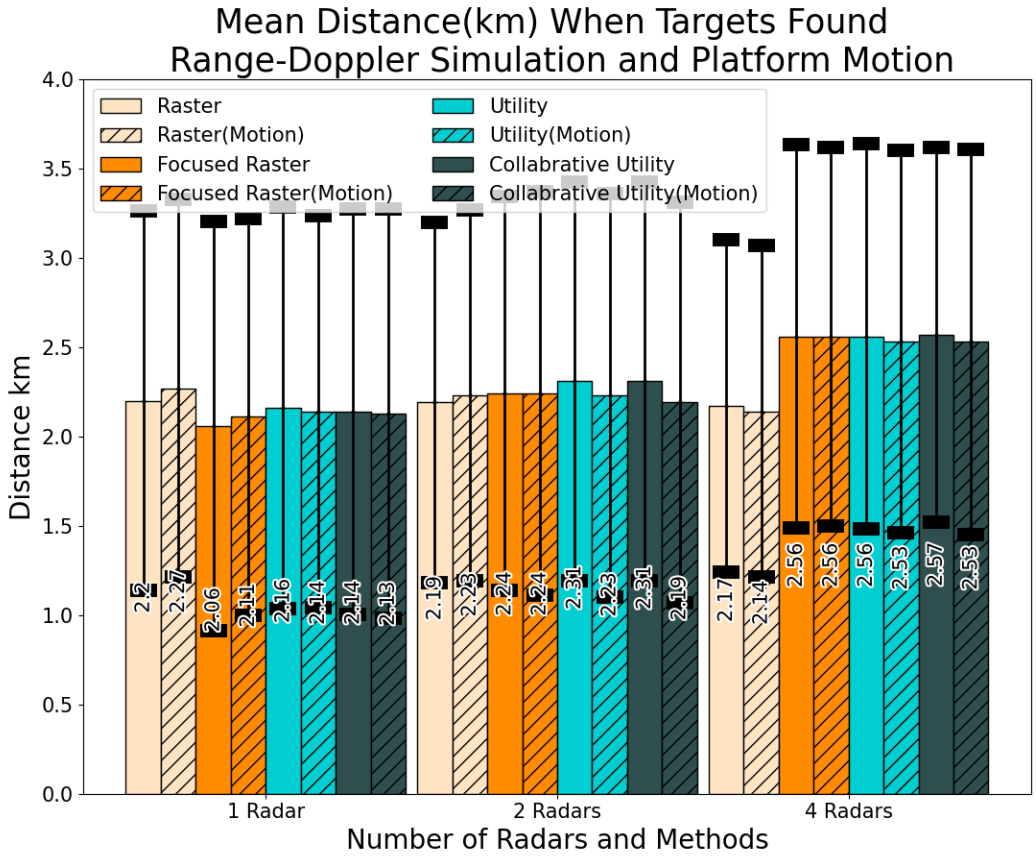


Figure 5.18: Distance Metric with Motion for Range-Doppler Simulation

less as the number of platforms reaches four, since they more saturate the surveillance space. These results indicate the utility method balances exploration and exploitation of the C2 biased space more effectively than the raster methods (optimized for either exploration or exploitation).

An important trade-off is computational cost as utility methods are costlier even with the pooling optimization. Testing of the methods indicated the raster methods (implemented on a Lambda Tensorbook with an i7-11800H, RTX 3080 Mobile, and 64Gb of RAM) averaged 0.1 ms per action, while the two-step utility method needed 600 ms per action. This cost is greater by a factor of 6000, but if an operational system has similar or greater processing, then the utility representation would be capable of

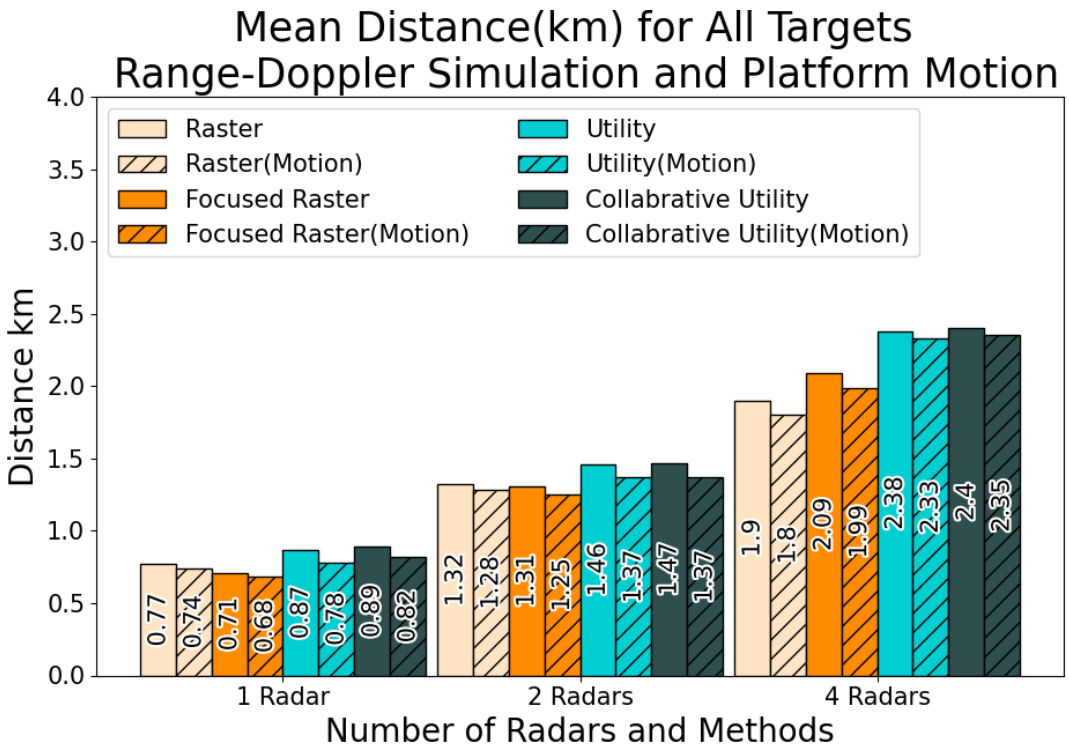


Figure 5.19: Distance Metric for All Targets with Motion for Range-Doppler Simulation

real-time surveillance. The computational complexity was tested using different size representations for the utility search (from the original 1000×1000 map size to a test run with 100×100 map size that ran 70.15 ms). This reduces the computational cost to only 700 times greater. The visuals from Figure 5.10 show that it is likely that both the map size and the pooling could be increased to reduce computational costs. A future area of improvement would be to limit test how low a resolution of the utility map could still perform well.

5.6 Conclusions

The work in this chapter explores the RRM task of search within a C2 scenario and with multiple collaborative platforms. Traditionally, search of a space treats all

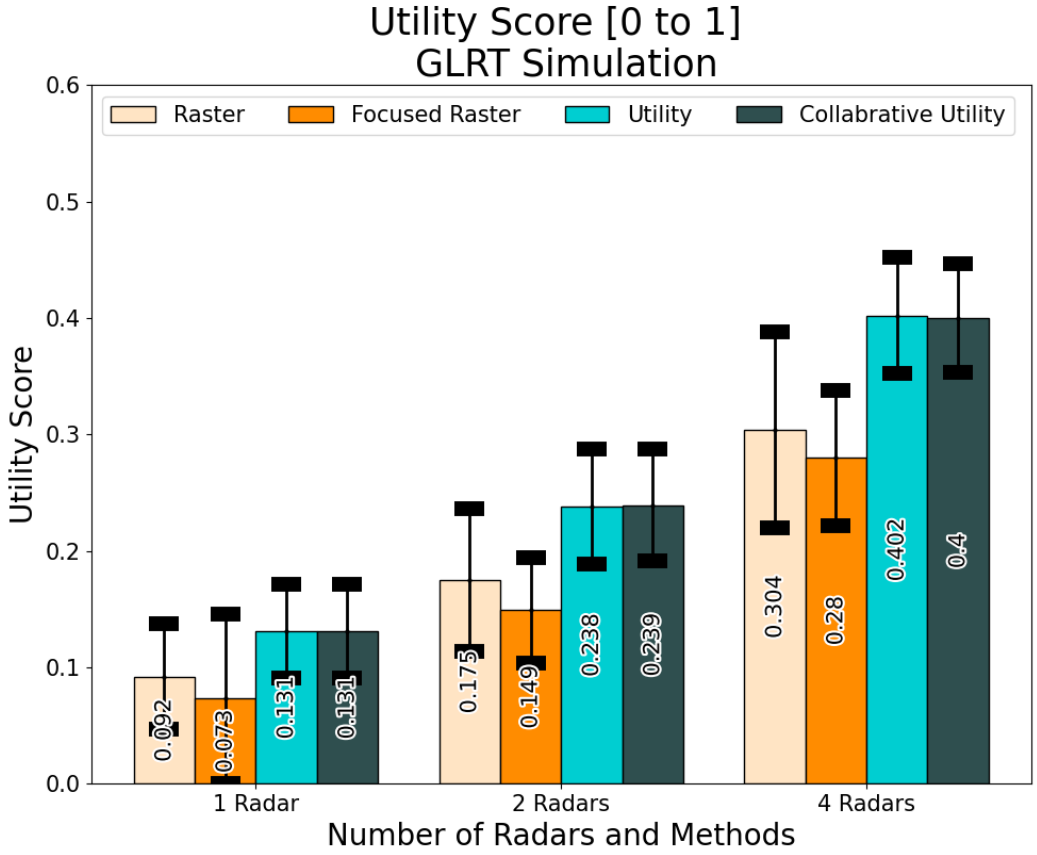


Figure 5.20: Utility Metric for GLRT Simulation

space as having equal importance and the entire space is cycled given radar resources and beam width. The field of knowledge-aided search revealed multiple methods to optimize search by incorporating information about the platform's location and capability, representing space priority with areas of target generation, and entropy and utility methods to represent space already searched. All of these methods where inspirations for the proposed utility representation.

The main contribution from this representation is the combination of these methods. The platform's expected P_D is used to bias the platforms to prioritize searching space with a higher chance of detection over locations further from the platforms. The inclusion of C2 knowledge in the form of objectives in the map drives opti-

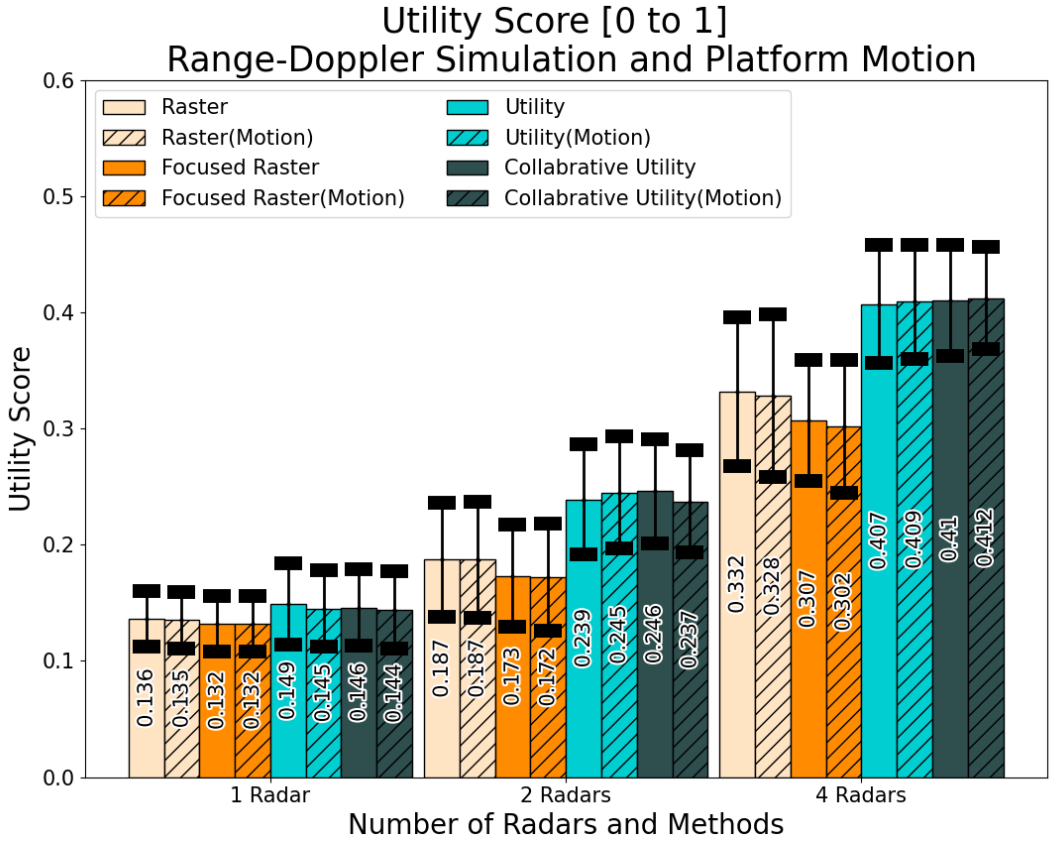


Figure 5.21: Utility Metric with Motion for Range-Doppler Simulation

mized surveillance of higher priority locations. Finally, a novel beta distribution decay method was used to balance the challenge of exploring new locations versus the need to research higher priority locations as uncertainty increases over time. This representation improved search for a single radar, but an expanded contribution is that this representation can efficiently include search history from collaborative radars to improve multi-agent performance even further. Also the method for sharing information is efficiently scalable as the layers are collapsed and stored efficiently.

This work showcases a utility search method that uses prior information from domain-specific scenarios to outperform more traditional search methods. Not only does this improve on naive search methods, but the flexibility and granularity allow

for noticeable performance improvements over other methods relying only on domain information. The utility search method is dynamic, different domain information can be encoded, allowing for use in complicated C2 applications. Not only can the utility layers be set up to bias individual agents into collaborating, but by building in collaboration, they can avoid wasted resources via sharing layers. The results indicate that applying the utility search method to a scenario with targets moving toward objective points consistently results in a greater than 10% improvement in targets found along with lower time-to-detect. This indicates dynamic and balanced exploration of the radar space and exploitation of prior knowledge has benefit for the surveillance task. Multiple platforms using this method will collaborate automatically when further from each other. In close proximity, they can share information with low computational cost, continuing collaborative actions that avoid wasted radar resources.

The contributions in this work show the value of incorporating prior knowledge about the platforms and the surveillance space to optimize search and the value of collaborative representations to improve multi-agent RRM tasks. The domain of radar search, distributed autonomy, and multi-agent collaboration all benefit from this method. The three core layers are all inspired by methods in the literature and formulated with radar and C2 concepts. This work shows novelty in the combination of these methods in an efficient format and provides a novel decay method using the beta distribution. This decay method balances the search new space vs. revisit challenge in surveillance with an elegant equation. This method also explored computational cost improvements and provides a linearly scalable ability for multiple agents to maximize collaborative utility.

This method shows great potential, and there are many places where it can be improved. For the multi-platform C2 domain, autonomous collaborative radar tasks are of value, given increasing complexity and volume of systems. Additionally, the potential of collaboration without direct communication is desirable for contested spectrum

scenarios. This paper used simple prior information to generate the utility layers but as the domain becomes more complicated, generating good utility layers may become more difficult. Analyzing what makes a good utility layer, and how different layers should be weighted will be very useful. Changing the way utility is removed after an action is taken will change the structure of the utility map which affects future actions. There may be additional performance improvements based on allowing overlapping beams or removing more utility around the beam to space out the search beams. This section of the utility search algorithm is another area for future research.

Motion was tested in this work, but all of the platforms move in the same counter-clockwise direction at the same rate. An area of future effort is to explore the impact of the collaborative representation as platforms move closer to each other. The authors hypothesize that when platforms are closer, it is more likely that illuminations will cover the same space. This should be accomplished with increased communication between agents, and is likely unnecessary when the agents are far away and the burden of communication increases. The results show that the utility method is a dynamic method that balances exploitation of prior knowledge, exploration of space, and collaborative optimization by selecting high probability actions given physical distances and/or sharing collaborative information. Investigating how the search method interacts with a radar scheduling system managing search and track actions will be important. This may highlight an advantage of the utility method in adapting its search actions based on the tracking actions.

The work presented in this chapter built tested multi-agent surveillance within a C2 scenario. It is core to my research goal to build scalable multi-agent methods to address RRM tasks and C2 challenges. The C2 scenarios are domain relevant to the research, but the utilization of realistic radar is required to have confidence in the novel methods. My research with this collaborative utility representation improved my understanding of the balance between exploring new space and exploiting prior

knowledge for effective surveillance. A core interest when exploring C2 was the need to optimize collaborative utility but also to has efficiently scalable methods. Often a more complex method will only work for single or small amounts of integrated agents. The collaborative representation, focused on scalable coordination and utilizing prior knowledge with balanced exploration and revisit mechanisms, was foundational to my research goals and showed promise from these results. These contributions were peer reviewed and evaluated by the community through a conference paper [22].

Chapter 6

Scalable Meta-Cognitive Radar Detection

6.1 Introduction

This chapter will focus on the first area of research I conducted within the field of cognitive radar. I will detail my inspiration, logic, and process regarding decisions made during the research process. This first research effort included collaborative design and development of a novel ML-based adaptive detection architecture. Utilizing my understanding of challenges in the radar detection domain, I supported and refined an idea for a meta-cognitive adaptive detector. This included understanding the challenges for adaptive detection when clutter distributions are non-Gaussian and helping to design a flexible meta-cognitive architecture to dynamically adapt to radar returns with changing clutter distributions. The architecture added capability and flexibility to a specific detection task given a distribution assumptions and included a second layer that perceived the environmental clutter distribution and selected the appropriate task parameters. This meta-cognitive architecture was designed, implemented, tested, and published in [23]. I will note here that the initial idea for this research was proposed by my colleague, Dr. Alexander Stringer. I provided support in refining the architecture and was a major contributor to this work with a focus on the meta-cognitive components. I will take care to distinguish between our contributions.

My initial research was motivated by a desire to add flexibility to existing systems and address the challenge of scaling problems into the multiple agent domain. This research interest drove collaborative research to address a class of challenges in radar where a method exists to solve a problem but has core assumptions that cause troublesome edge cases. The primary author's motivation for this research came from a literature review on adaptive detection techniques. They came to the conclusion that existing detection algorithms made assumptions that clutter interference was known and relatively consistent. Attempts to operate over a broad range of distributions would largely develop lookup lists of test statistics and thresholds. Then these parameters were selected with heuristic methods to approximate distribution followed by selection from the lookup tables.

Early work with ANN's showed their potential as universal function approximators. A literature review of ML methods for adaptive detection revealed other researchers had similarly explored the use of deep learning ML for adaptive detection. Some focused on using ML to directly classify detections but resulted in similar or worse detection performance to likelihood based detectors, but added additional uncertainty by utilizing less interpretable ML methods. Others utilized deep ML to identify distribution types and parameters fitting to the lookup tables previously discussed [126]. Additionally research was conducted to identify distributions and set the corresponding threshold with regression ML [127]. Each of these methods met challenges because they could not both generalize across multiple distribution and provide high accuracy for heavy-tailed regions where slight changes in threshold have large impacts on CFAR behavior.

The primary author identified the potential for a meta-cognitive approach and I helped refine the problem statement and potential solution space. We identified multiple issues that cause previous methods to struggle and devised methods to address them. The first issue was regarding the generalized need versus finely-tuned distri-

bution response and led to the proposal of multiple specialized detectors. The next issue was that methods to distinguish between distributions often struggled because they would use unprocessed random samples from distributions and expected ANNs to classify patterns. Although ANN's are adept at finding patterns, random sample data often leads to difficulty converging. A final issue was that systems would train on live data which often contained mixed distributions or non-distribution features which confused model classification.

When approaching the challenge of adaptive detection (described in detail in section 2.3.4) a common method for modeling clutter interference is the use of SIRVs. SIRVs are characterized by their shape and scale parameters and can differ greatly as their parameters change. The shape parameter typically has an exponential response at low values but approaches Gaussian at high values. This presents a challenge when attempting to determine a SIRV class distribution and its shape parameter with one model or prediction method. This ambiguity is what inspired the concept of not only distinguishing between distribution but also to identify shape parameter regions that are locally similar.

The principle author proposed the use of a meta-cognitive approach which I helped refine and implement. In this approach we designed a discriminator to analyze processed simulated data of multiple SIRVs and then make a classification for both the distribution type and which region to which it most likely belonged. Once a region was selected, multiple ANNs were trained for each distribution and region combination to determine the fine-tuned threshold value used for the detection criteria. The following section will explain the architecture and design of the original system. Then I will detail a follow-on effort I led to formalize the distribution region division process and my work to implement the meta-cognitive system as a micro-service and containerized software ready for orchestration in a cloud system. Finally, I will present and discuss the results from the dynamic expansion and how this research contributes to the radar

domain.

6.2 Proposed Approach

The architecture described in this research is a meta-cognitive detector presented in figure 6.1.

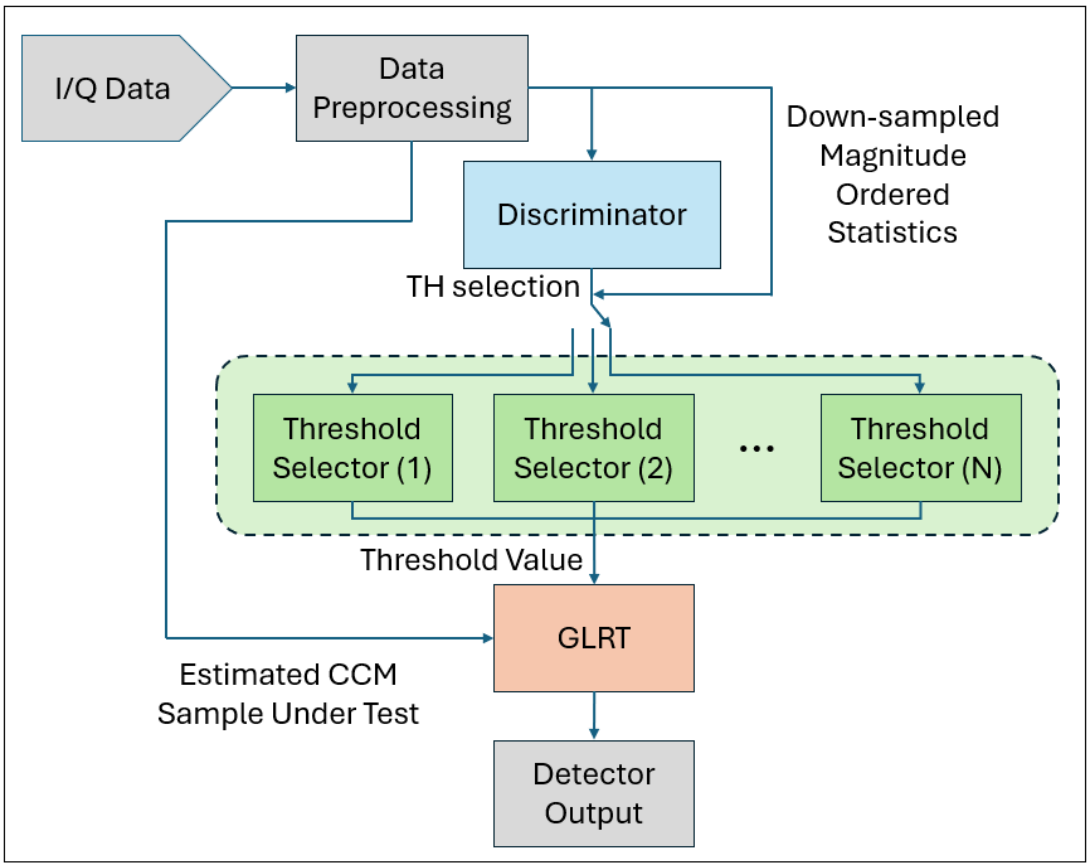


Figure 6.1: Meta-Cognitive Architecture

The following subsections of this proposed approach section will go through each of the component sections of the meta-cognitive architecture and explain the thought process, design details, and implementation. The discussion will start with a description of the original raw I/Q, the GLRT adaptive detector implementation the process for selection of the task distribution regions, the discriminator design and training, and

the threshold selectors design and training.

6.2.1 Input Data

The expected input data for this system is raw I/Q radar data. Specifically the data contains a sample within a range cell referred to as a cell under test (CUT), z_{CUT}, of length $N = 16$. Per section 2.3.3 the CUT will contain interference but may contain a target which is desired to be detected. Together this characterizes the binary hypothesis test for adaptive detection.

$$\begin{cases} \mathbf{H_0}: & \mathbf{z} = \mathbf{n} \\ \mathbf{H_1}: & \mathbf{z} = \mathbf{s} + \mathbf{n} \end{cases} \tag{6.1}$$

In (6.1) $\mathbf{n}$ is the N-dimensional complex interference vector and $\mathbf{s}$ is the N-dimensional target return vector. Throughout the literature, the clutter interference is modeled as $\mathbf{n} = \sqrt{\tau}\mathbf{x}$, where τ is referred to as the texture and $\mathbf{x}$ is the speckle. Here, τ is positive and represents localized clutter power. $\mathbf{x}$ represents local back-scattering and is typically modeled as an m-dimensional Gaussian random vector with zero mean and normalized Hermitian covariance matrix $E[\mathbf{n}\mathbf{n}^H] = \mathbf{M}$. For convenience in simulation, here we adopt the model where $\mathbf{M}_{i,j} = \rho^{|i-j|}$ $(1 \leq i, j \leq m)$. The correlation coefficient, ρ, is positive in the range $0 \leq \rho \leq 1$ with higher values modeling more correlated interference and lower values modeling less correlated interference.

The target return signal is modeled as $\mathbf{s} = \alpha\mathbf{p}$. α is an unknown positive random value representing the amplitude of the signal return. This is meant to capture the effects of channel propagation. $\mathbf{p}$ is the N-dimensional known steering vector $\mathbf{p} = [e^{-j2\pi f_D T_s}, ..., e^{-j2\pi m f_D T_s}]$, where f_D is the Doppler frequency and T_s is the pulse repetition interval. We can now rewrite the previous hypotheses as:

$$\begin{cases} \mathbf{H_0}: & \mathbf{z} = \sqrt{\tau}\mathbf{x} \\ \mathbf{H_1}: & \mathbf{z} = \alpha\mathbf{p} + \sqrt{\tau}\mathbf{x} \end{cases} \tag{6.2}$$

An additional $K = 2 \cdot N = 32$ length $N = 16$ support samples were generated that contained only the clutter interference components. These support samples were used during the pre-processing stage for input to the discriminator and the threshold selectors as well as the required components of the adaptive detector. The clutter component for the CUT and the support samples was generated in Matlab using speckle/texture based SIRV generation techniques for Gaussian, Pareto, and K-distributed random variables.

6.2.2 Adaptive Detector

Now that the form of the data has been established it is important to understand the role of the adaptive detector, how it generates a test statistic and the role that a threshold value plays in detection. A core philosophy for this research was to explore using AI/ML to augment existing state of the art radar detection methods as opposed to replacing them. Therefore, the role of the adaptive detector and its input requirements provides areas for added capability due to AI/ML based techniques. The primary adaptive detector used for this work was Kelly's GLRT. The AMF was also implemented and used for comparison. For a full derivation of the GLRT and AMF test statistic see section 2.3.7. The well known form of the GLRT test statistics is:

$$\eta = \frac{|\mathbf{p}^H\mathbf{S}^{-1}\mathbf{z}|^2}{(\mathbf{p}^H\mathbf{S}^{-1}\mathbf{p})[1 + (\mathbf{z}^H\mathbf{S}^{-1}\mathbf{z})]} \tag{6.3}$$

Where $\eta = \frac{\lambda-1}{\lambda}$, λ is the threshold value, $\mathbf{S}$ is the estimated clutter covariance matrix, $\mathbf{z}$ is the CUT, and $\mathbf{p}$ is the steering vector, The steering vector is generated given the radar system parameters and the CUT is generate for hypothesis testing according to (6.2). The remaining components, estimated clutter covariance and threshold value,

are determined from the additional K support samples containing only interference information. The real clutter covariance matrix is unknown and is estimated using the following method:

$$\mathbf{S} = \sum_{k=1}^{K} \mathbf{z}_k \mathbf{z}_k^H \tag{6.4}$$

$\mathbf{z}_k = \mathbf{n}_k$ represents the support samples which contain only interference information. This is a baseline CCM estimation method and requires a minimum of $K = 2 \cdot N$ supporting samples in order to ensure that the inverse CCM is stable. Extensive research has been conducted to increase accuracy of CCM estimation while relying on few samples. Many methods will utilize assumptions about the form of CCMs like persymmetry, Hermitian, Toeplitz, and Low-rank dominance combine with reconstruction methods to estimate the CCM with fewer samples. But for this initial implementation we generated the minimum number of support samples to estimate the CCM with (6.4).

The threshold value is more difficult to determine as it depends on the target probability of false alarm P_{FA} and assumptions regarding the distribution of the clutter. For interference with a zero-mean Gaussian distribution assumption there exists a closed form calculation for the P_{FA} and inversely the ideal threshold for a target P_{FA} [57]:

$$P_{FA} = \frac{1}{\lambda_0^{K-N+1}} \tag{6.5}$$

$$\lambda_0 = (P_{FA})^{\frac{-1}{K-N+1}} \tag{6.6}$$

The GLRT has been proven to be a CFAR detector with respect to clutter that follows the zero-mean Gaussian assumption. When this assumption does not hold the GLRT can still provide CFAR -like behavior with the appropriate threshold. For non-Gaussian distribution this threshold is typically found using empirical methods and then stored in a look-up table [126]. I spent considerable effort mapping, storing and

training ML to provide threshold values for non-Gaussian distributions in the following sections.

6.2.3 Distribution Region Selection

The next phase in our research was focused on the criteria for dividing up the threshold regions. This process had an impact on the discriminator by determining the output classes and had a direct correlation to the possible accuracy of the trained ANN. Our research proposed the concept of distribution regions to frame the fine-tuned task-based threshold selectors so that they focused on statistically distinct distributions but covered a similar enough region to gain high-fidelity threshold predictions. As discussed in 2.3.4 the SIRVs, Pareto and K-distribution are defined by the shape and scale. The shape has the greatest impact in the distinctness of the distribution. Therefore, we used a halving algorithm presented in 1 to sweep the threshold values to support analysis for unique regions. Additionally, I conducted a Jensen-Shannon analysis of perspective regions to confirm that these regions where statistically distinct. The following section will detail this pre-design analysis for distribution region selection.

Algorithm 1 uses a Monte Carlo method to identify optimal thresholds that maintain a CFAR-like detection response. Specifically this algorithm will iterate toward the optimal threshold for the target P_{FA} and then it will select a new point in the direction of the target P_{FA}. This new point will either increase or decrease by the threshold delta ($\lambda_{0,\Delta}$) unless the specific event where any selected point has switched between increasing or decreasing [Event($sgn(P_{fa,target} - P_{FA})$]. Once this event has happened then new points will halve the threshold delta until it converges on the optimal threshold given some margin (ϵ). Monte Carlo methods for identifying are commonly used to identify thresholds for covariance inversion detectors like the GLRT due to a lack of closed form expressions to calculate thresholds in non-Gaussian clutter. It is, of course, infeasible to use Monte Carlo methods in real-time systems, so many classical

Algorithm 1 Threshold Search

for $C_n \in \mathbf{c}_{dist}$ **do**

for a_i in $Region(C_n)$ **do**

Initialize $\lambda_0 = \lambda_g$ and $\lambda_{0,\Delta} = \Delta_0$

while $|P_{FA} - P_{fa,target}| \geq \epsilon$ **do**

Generate $10^2/P_{fa,target}$, Z_K samples with $a = a_i$

Calculate λ with GLRT for all samples

P_{FA} = $\frac{\sum Event(\lambda \geq \lambda_0)}{P_{fa,target} \cdot 10^2}$

$\lambda_0 \leftarrow \lambda_0 + sgn(P_{fa,target} - P_{FA})\lambda_{0,\Delta}$

if Event(sgn($P_{fa,target} - P_{FA}$) change has happened) **then**

$\lambda_{0,\Delta} \leftarrow \frac{\lambda_{0,\Delta}}{2}$

end if

end while

end for

end for

approaches to dynamic thresholding in non-Gaussian scenario use them to populate a look-up table of threshold values. Such approaches are limited by their size and resolution, limiting their utility in real-world applications. Here, we use Algorithm 1 to analyze and select threshold regions but in section 6.2.6 it is also used for generating training labels.

In algorithm 1 the classes of distributions and their distribution regions are reflecting the the set $\mathbf{c_{dist}} = [C_G, C_{K1-Kr_k}, C_{P1-Pr_p}]$. For the set classes representing K-distribution (C_{K1-Kr_k}) and Pareto (C_{P1-Pr_p}) they contain a finite number of distribution regions. (r_k and r_p respectively) Within each region is a finite set of shape parameters (a) that I swept through to generate an observable curve as well as generate supervised labels for the threshold selectors discussed in section 6.2.6. This halving algorithm initialized two threshold parameters: a guess threshold value ($\lambda_0 = \lambda_g$) set

to the ideal Gaussian threshold and the halving delta ($\lambda_{0,\Delta}$). The threshold delta was set to .1 and had the trade-off that is two small it would take additional iterations before the first halving but if too large it will take additional iterations to halve enough to meet stopping criteria. The $P_{fa,target}$ is the desired probability of false alarm and P_{FA} is the current P_{FA} calculated given the current threshold. Figure 6.2 is a visual representation of the halving algorithm. It is important to note that this method is computationally light and converges quickly but only for a problem surface that is monotonically increasing or decreasing. For more varied problem surfaces a multi-agent mountain climber method or a genetic algorithm would be needed.

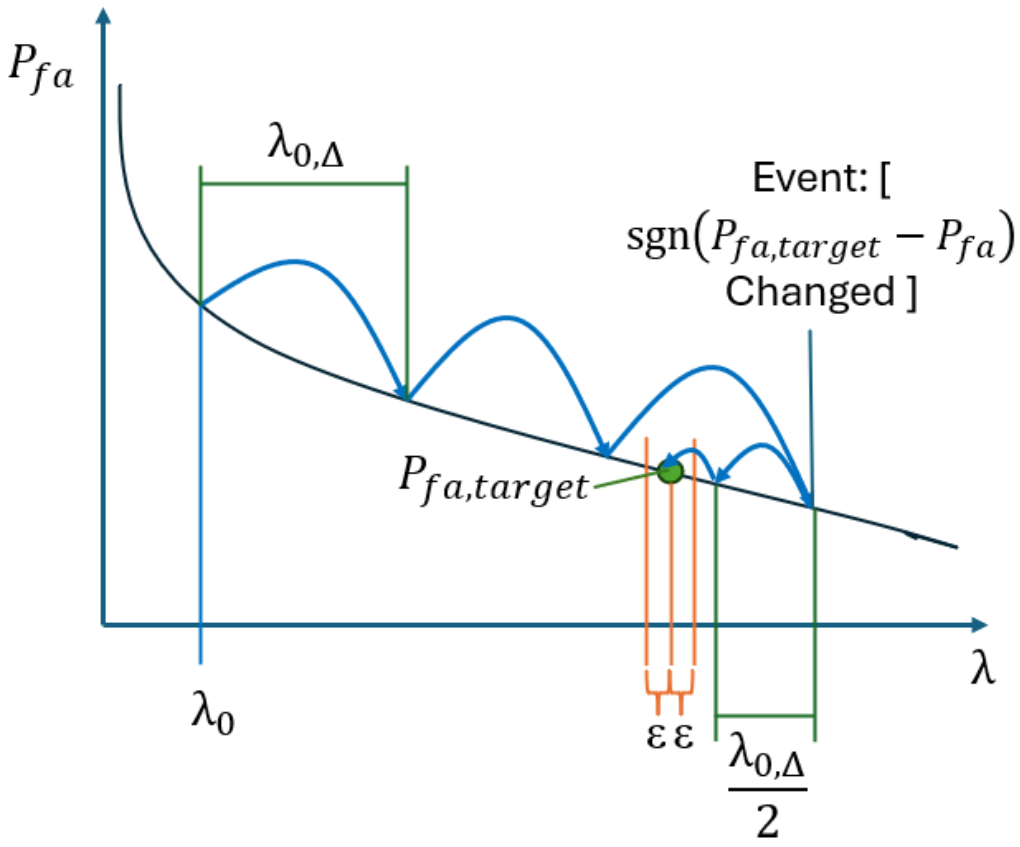


Figure 6.2: Illustration of Halving Algorithm 1

Using this halving method to empirically determine a threshold within some ϵ error

for a given distribution and shape parameter, I was able to sweep the K-distribution and Pareto distribution for analysis. Figures 6.3 and 6.4 show the results of these sweeps for K and Pareto respectively.

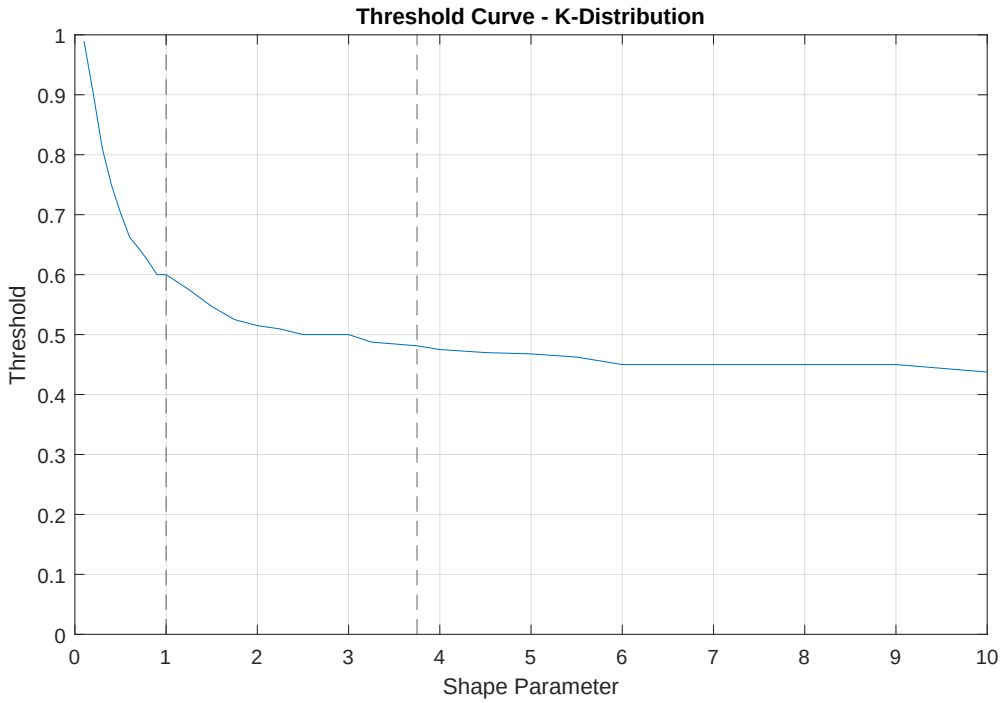


Figure 6.3: Optimal Threshold Labels (K-Distribution)

I used these curves to determine nearly linear regions. The reason for defining linear regions is to enable the threshold selector ANNs to achieve high-fidelity predictions. An important parameter for the meta-cognitive design is the number of regions. Ideally, more regions will enable higher fidelity selector regions, but could create more confusion for the classification of the discriminator and increases memory needs and training time for the additional ANNs. The linearity of the threshold regions is only one aspect for determining the distribution regions.

The second factor about the nature of SIRV distributions and the design for the regions we analyzed had to do with the distinctness of the regions. To accomplish

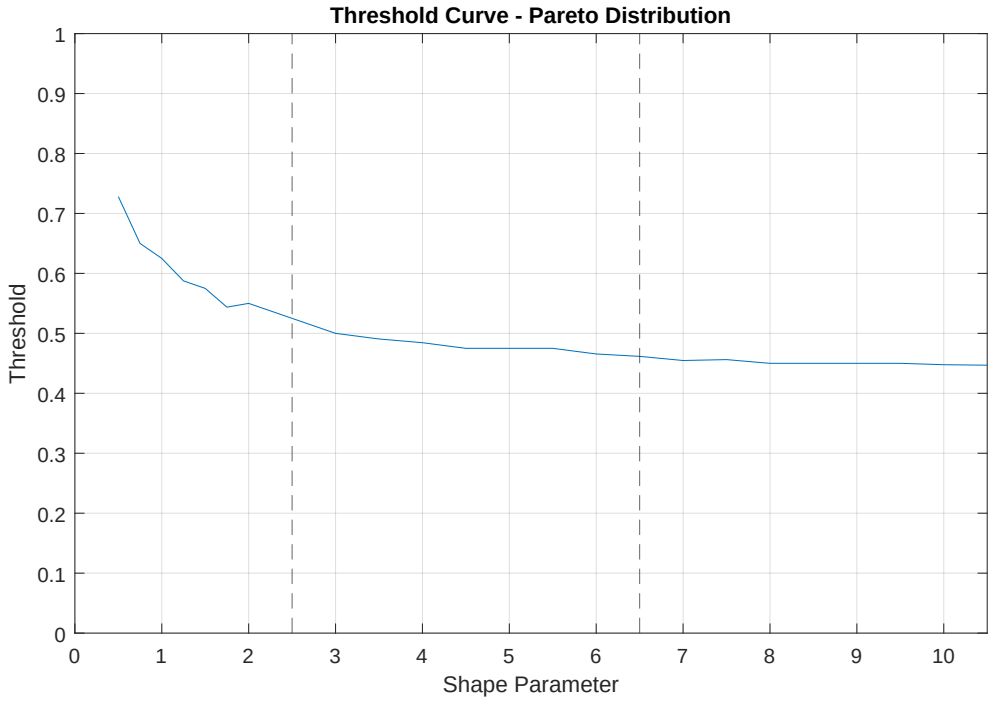


Figure 6.4: Optimal Threshold Labels (Pareto)

this I conducted a JSD (see (2.42)) for all regions and visualized it in a heat map. This JSD matrix is shown in Figure 6.5 and was used to evaluate how similar a set of regions were with 0 being identical and 1 being completely dissimilar. To ensure consistency a total of 20,000 samples of $\mathbf{Z}_K$ (length 512 complex values) for each class were generated and the average JSD over those samples is presented in Figure 6.5.

In this case the shape parameter used was the center of the regions with these initial regions displayed in table 6.1.

Both the sweep of the threshold values at a target P_{FA} and the JSD matrix were tools used to determine the initial selection of distribution regions. The specific regions in table 6.1 and figure 6.5 were the regions selected for the initial design, but I also used the JSD matrix method to test multiple parameter variations including a

Classes	Gaussian	K-Dist	K-Dist	K-Dist
a Regions	-	0.1 to 1	1.25 to 3.5	3.75 to 10
a Training label	0	1	2	3
Classes	-	Pareto	Pareto	Pareto
a Regions	-	.5 to 2.5	3 to 6.5	7 to 10.5
a Training label	0	4	5	6

Table 6.1: Clutter Ranges and Labels

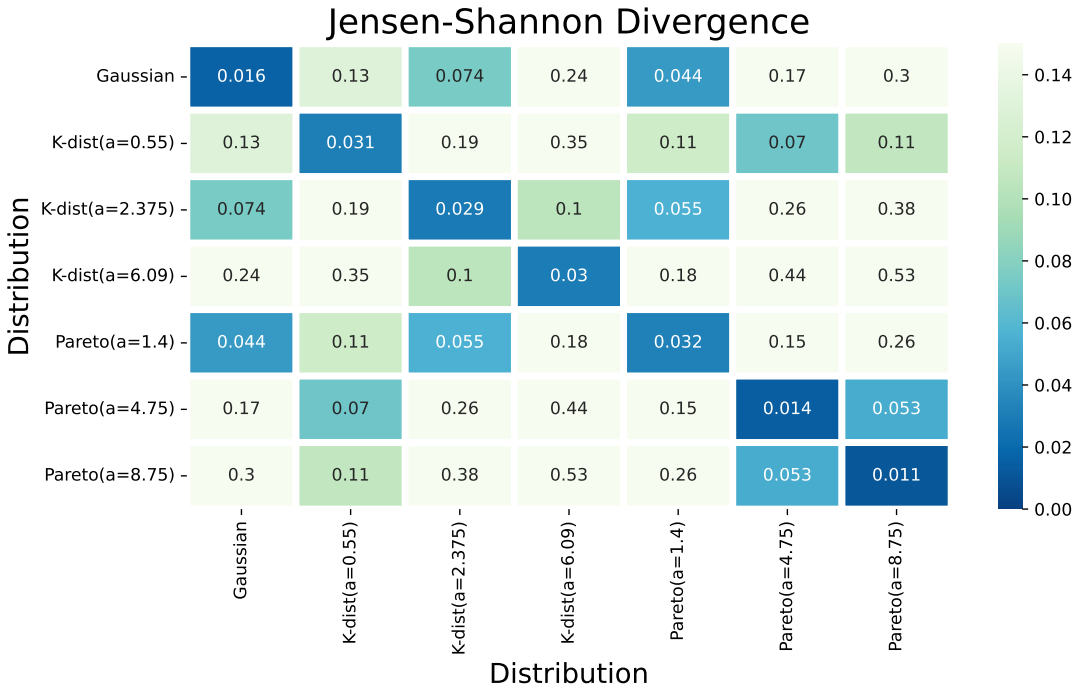


Figure 6.5: JSD for Gaussian and 3 Regions of K and Pareto

total number of regions = $[3, 5, 7, 10]$. In all cases the region boundaries were visibly determined using the threshold sweep figures 6.3 and 6.4. For the sake of simplicity and utilizing these tools I worked with my research partner Dr. Stringer to select the final regions which drove the design and training of the meta-cognitive structure.

6.2.4 Data Preprocessing

In the architecture presented in Figure 6.1, the raw I/Q data is preprocessed before it can be used by the ML components and the GLRT test statistic. The first preprocessing step is the estimation of the clutter covariance matrix (CCM). The inverse of the CCM is used by the GLRT with the CUT to generate the test statistic to be compared with a threshold. For the initial implementation the primary author estimated the CCM from the (32 X 16) support samples provided in z_K by using the estimation method (6.7) [128].

$$\mathbf{S} = E[\mathbf{n}_k \mathbf{n}_k^H] = \sum_{k=1}^{K} \mathbf{z}_k \mathbf{z}_k^H \tag{6.7}$$

The CCM preprocessing component was separated from data generation to enable modular expansions in the future by exploring low sample support CCM and inverse CCM generation. In follow-on work conducted by the primary author, he explored the use of generative ML methods to directly generate the inverse CCM. This work is found in [129]. My contribution involved data generation, testing, and paper editing therefore a full exploration is not included here as the novel contribution was Dr. Stringer's. For this initial work the CCM was calculated with using (6.4) and drove the need for $K = 2 \cdot N$ support samples in z_K.

The second area of preprocessing I performed is in an effort to process and format the raw I/Q data into a form that a deep neural network can learn relevant patterns to classify. ANNs are very effective at learning patterns within training data but they focus on the features that have the most correlation. If one feature dominates others it will dominate classification. Additionally, if those dominate features are not causally related to the desired solution or are heavily dominated by random information then the ANN will struggle. Therefore, the process for the preprocessing design is closely related to my ideation and exploration while designing and training the discrimina-

tor. This preprocessing will be discussed in the following section. Section 6.2.5 will briefly explain the exploration process as well as the final preprocessing that maximized the ANN's pattern finding capability and made the system less dependant on support sample length and amount.

6.2.5 Discriminator

A core component to the meta-cognitive detection systems success is tied to the accuracy of the discriminator. The goal of the discriminator is as a classifier that will learn patterns in the (32 X 16) support samples which contain only clutter information. In section 6.2.3, I discussed the process for determining the distribution regions based on the linearity of the threshold curves and the statistical uniqueness of the regions. This analysis provided the initial distribution regions in table 6.2.3. I utilized the Jensen Shannon divergence in figure 6.5 to affirm that the regions were mostly unique ($JSD < .03$ on diagonal and mostly $JSD > .1$ elsewhere). Nevertheless, the nature of SIRV distributions means that there will be confusion for neighbor regions and for higher-shape regions as they approach Gaussian. This presented a challenge in both the input data form and the architecture with the goal of not just attaining a high accuracy but also low confusion.

During this process, I explored a sweep of data forms and architectures before settling on the final form. Initially, I left the raw I/Q in its original form with an input size of 512 X 2. The 512 is a flattened version of the 32 x 16 support samples and the second dimension includes the real and imaginary components. This lack of preprocessing heavily confused every architecture I tried. The real and imaginary components were independent and the ANN was trying to find correlation on that second dimension. The first data processing was to train with the magnitude of the real and imaginary components. Utilizing the 512 x 1 input showed better results but still had issues and the ANN architectures were trying to find correlation between

neighboring measurements in the samples. The data was generated with $\rho = .9$ so there was correlated information but the random format caused a challenge. The next change was to order the data into an ordered statistics form. This data form performed very well, but I postulated that the full 512 samples where not needed and resulted in a larger model than necessary. Additionally, my research partner and I discussed a desire to make a system that was input size agnostic. We did not want to design the system to always have 512 input samples so we decided to explore down-sampling. Down-sampling provided two benefits assuming we could find a size that did not sacrifice important corollary information. The first was a smaller, faster model the second was that as long as we have support data greater than our down-sampled input then the system would work.

The final data form was as follows: The data was input as the (16×32) matrix of support samples and were vectorized into 512 complex data points, and then the magnitude was ordered. These ordered statistics are then down sampled by a factor of 8 to reduce the dimensionality that the machine learning must address. As a result, a down sampled set of uniformly spaced ordered statistics with $n = 64$ samples is then available for the Discriminator to learn as:

$$\begin{aligned} \mathbf{z}_{os} &= Or(vec(\mathbf{z}_K)) = [|z_{K,(1)}|, \ldots, |z_{K,(512)}|]^T \\ \mathbf{z}_{ds} &= \downarrow_8 \mathbf{z}_{os} \\ \mathbf{z}_{ML} &= \left[z_{ds,(1)}, z_{ds,(2)}, \ldots, z_{ds,(n)}\right]^T \end{aligned} \tag{6.8}$$

where $Or\left(\bullet\right)$ is the order function, $vec\left(\bullet\right)$ refers to vectorization, and $\downarrow_8 \left(\bullet\right)$ denotes down-sampling by a factor of 8. Using magnitude down sampled ordered statistics as the inputs to the models has two key benefits as explored above but summarized here. First, it provides additional distribution information to the ML models. There is a high degree of variability to the values contained within raw signal returns, and training an ML agent on such data will typically cause them to either over fit or fail to con-

verge to a solution. Using a set of ordered statistics requires very little pre-processing, reduces the variability of the inputs to the ML models, and provides crucial information related to the underlying distribution that the models were demonstrated to learn. This is in part what allows the proposed system to generalize effectively. In addition, the use of a constant number of down sampled ordered statistics makes the proposed system largely agnostic to the amount of sample support available, as long as at least 64 (in this case) values are available. At worst, some models in the system may require retraining, but no structural/architectural changes would be needed. Testing has not been performed in this work to identify the minimal number of ordered statistics required for model training, but future work will examine this optimization problem.

With the data in a form that supports classification by the ANNs; I then explored a few ML architectures. The following discussion will briefly include some architectures I explored as well as the final architecture. The first architecture left the samples in the original 32 x 16 matrix and tried using a CNN. CNNs use image processing filters or kernels to learn spatial correlation in multi-dimensional data. The initial idea for a CNN was that the matrix of support samples represent range bins in fast time and samples in the slow time. This could help if there was signal returns, but we are analyzing support samples with only clutter information. The CNN was overly complicated for this application and struggled with over-fitting. The next architecture I explored was a LSTM with the magnitude down-sampled order statistics. Although it did well I postulated that the sequential pattern recognition of an RNN was unnecessary. Therefore I considered the benefits of designing the system with the simplest ML structure that retained flexibility and solved the problem. This provided the benefits of lower computational costs and a lower bar for explainability.

The final design was a simple two layer ANN classifier. The Discriminator was a classification ANN which takes the down-sampled magnitude ordered statistics as inputs. This classifier trained to identify the patterns in seven classes of clutter and

outputs a probabilistic classification of which classes the input aligns to most. The K- and Pareto distributions represent regions in shape parameter a, but the ANN truth labels are a classification label ranging between 0 and 6. The individual samples were generated with a random shape parameter in the labeled region. Regions and labels are shown in Table 6.1.

The model design is shown in Figure 6.6 and is a two hidden layer DNN using the exponential linear unit (ELU) activation function. The output is a probabilistic classification and uses sparse categorical cross entropy (SCCE) as the loss function. Given output classes as $\mathbf{c_{dist}} = [C_G, C_{K1}, C_{K2}, C_{K3}, C_{P1}, C_{P2}, C_{P3}]$, the output takes the form $[P(\mathbf{z}_{ML} \sim C_G), P(\mathbf{z}_{ML} \sim C_{K1}), \ldots, P(\mathbf{z}_{ML} \sim C_{P3})]$ with a summed probability of 1.

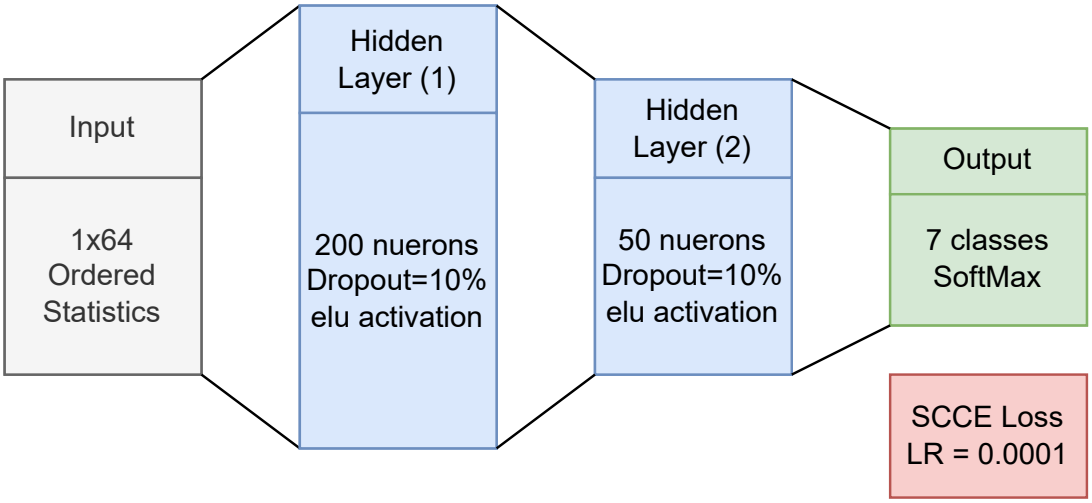


Figure 6.6: Discriminator Model

The training specifics for the discriminator includes the following parameters and data generation specifics. The Discriminator model was written in Python and trained with Tensorflow. For training, the data was generated as 20,000 length 512 complex values ($\mathbf{Z}_K$) for each of the seven classes in C_{dist} with the average shape parameter a from Table 6.1. The samples were pre-processed into $\mathbf{z}_{ML}$ from equation (6.8) and

labeled with a class target $[0$ to $6]$. The training set was 75% of the total data with a training validation set of 12.5% and the evaluation testing set of the final 12.5%. The Discriminator ANN was trained with a learning rate of 10^{-4} and had a stopping criteria using a patience factor of 30. The patience factor represents a stopping criteria for training through evaluation of the validation data set. Specifically, training ends when the number of successive epochs with validation metric (loss/accuracy) no better then the stored best metric has occurred. This method is used to have statistical confidence that the ANN has trained and is either stable or beginning to over-fit. Following the patience stopping condition the model weights with the best metric are returned.

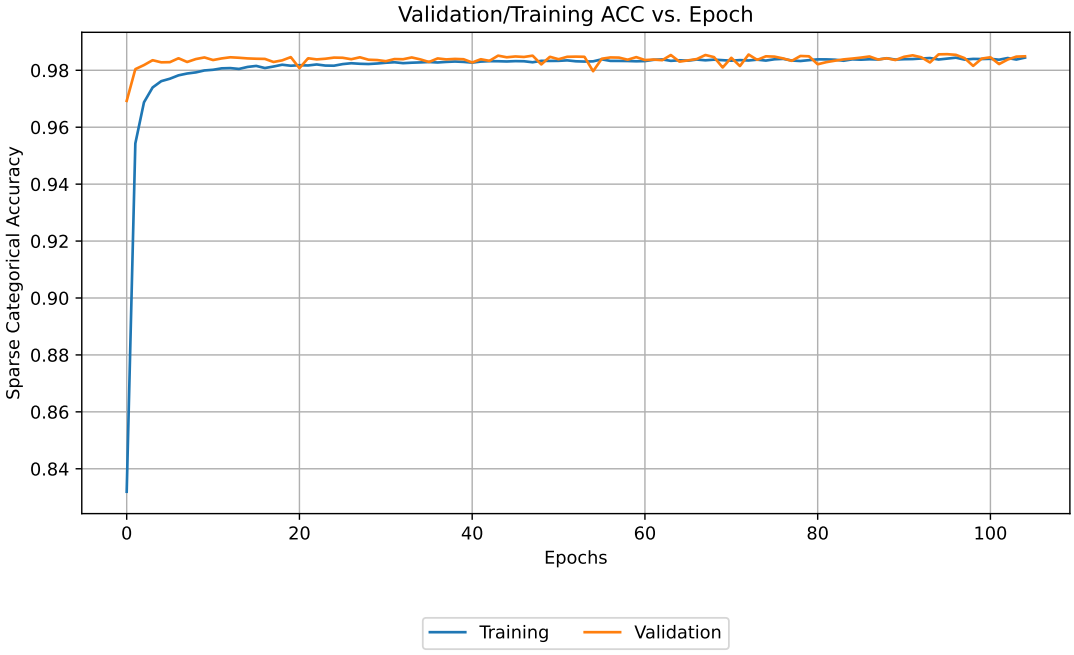


Figure 6.7: Discriminator Training Accuracy

The Discriminator training and validation accuracy are presented in Figure 6.7 and show a stable training for the ANN model. These training results are very promising for the success of the meta-cognitive system due to the very high classification accuracy ($\sim$ 98%) and near zero confusion in classification. Compared to the Jensen Shannon results it is evident that the distributions had enough uniqueness in their or-

der statistics for the Discriminator to identify the clutter class with high accuracy. The classification confusion matrix for the discriminator is shown in Figure 6.8.

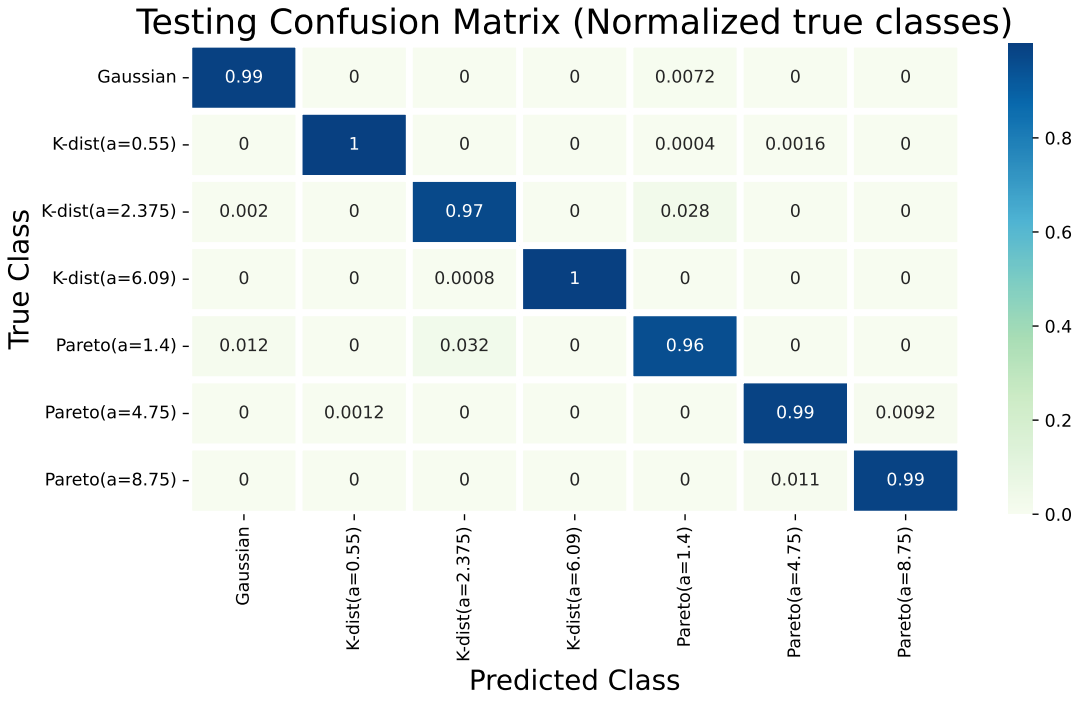


Figure 6.8: Discriminator Confusion Matrix

These results were very encouraging and showed that the exploration and care taken during region determination, data preparation, and model design resulted in a focused solution with high accuracy. This meant that the Discriminator could determine what distribution and distribution region the support samples contain with high accuracy and low confusion. Within the meta-cognitive system this provided a high quality model which used situational awareness of the environment (support sample clutter composition) and then determine the specific task (threshold selector) to select the high fidelity solution for the detecting problem. The following section 6.2.6 will explore my role in the design and training of the specialized threshold selectors.

6.2.6 Threshold Selectors

As previously discussed our metacognitive detection approach is developed to maintain CFAR-like behavior for potential targets in diverse clutter environments. Based on literature review regarding challenges in this space we knew that attempt to treat distributions as singular classes of clutter and determining threshold values for large ranges of those distribution resulted in a generalized solution that was lower fidelity. I did analysis to determine distribution regions and designed, trained, and evaluated a high accuracy, low-confusion discriminator. Now that the Discriminator could accurately determine the distribution and region for the support samples I worked with my research partner to develop the initial 6 Threshold Selectors. Dr. Stringer designed the architecture and I generated the threshold labels. We both trained and tested models but the majority of this task level design was implemented by Dr. Stringer.

For these ANNs, seven regions were identified and each had a threshold selector implemented and trained to perform in that region with the Gaussian GLRT detector. The first region was the simple Gaussian case for which the GLRT was designed. In this region, the threshold was selected using the closed form expression found in Equation (6.5). The other six regions were pulled from the K- and Pareto distribution models: 1) low shape parameter K, 2) medium shape parameter K, 3) large shape parameter K, 4) low shape parameter Pareto, 5) medium shape parameter Pareto, and 6) large shape parameter Pareto. The shape parameter values for these regions can be found in Table 6.2, and plots of the ideal threshold values can be found in Figures 6.3 and 6.4. The vertical dashed lines in these figures depict the transition boundaries between each of these regions, as previously discussed in the section 6.2.3 the threshold curves are approximately linear within each of these regions. Regression ANNs were used to set the thresholds used by the final GLRT. In this work the regression ANNs trained to select thresholds were referred to as the Threshold Selectors.

Filter	Shape Parameter (a) for Threshold Labels
K_1	$[0.10, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, 0.90, 1.00]$
K_2	$[1.25, 1.50, 1.75, 2.00, 2.25, 2.50, 2.75, 3.00, 3.25, 3.50]$
K_3	$[3.75, 4.00, 4.25, 4.50, 5.00, 5.50, 6.00, 7.00, 8.00, 9.00, 10.00]$
P_1	$[0.50, 0.75, 1.00, 1.25, 1.50, 1.75, 2.00, 2.50]$
P_2	$[3.00, 3.50, 4.00, 4.50, 5.00, 5.50, 6.00, 6.50]$
P_3	$[7.00, 7.50, 8.00, 8.50, 9.00, 9.50, 10.00, 10.50]$

Table 6.2: Shape Parameter (a) for Threshold Labels

The input to the ANNs are the down-sampled magnitude order statistics $\mathbf{z}_{ML}$. The model for each task level ANN is shown in Figure 6.9.

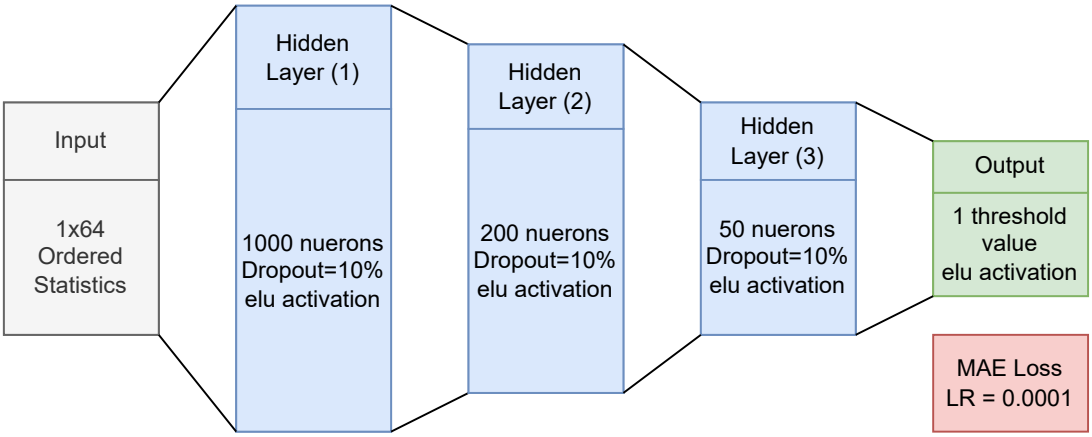


Figure 6.9: Threshold Selector Model

In order to complete supervised training, the ANNs needed labels for the optimal threshold at discrete points within their region. The training labels were generated using the halving algorithm 1.

Each of the six Threshold Selectors used for the full system were trained on data generated using the shape parameters listed in Table 6.2. $20,000$ test samples $\mathbf{z}_{CUT}$ and associated support samples $\mathbf{Z}_K$ were generated for each. All samples generated

for training of the threshold setting agents were clutter only signals. 75% of the data was used for training, 12.5% for validation, and 12.5% for testing. Training data and labels were jointly shuffled prior to training the models. Each model was created and trained in Python with Tensorflow using the MAE loss function. This was selected due to the threshold output being bounded between 0 and 1. The training loss curves for the Threshold Selectors can be found in Figures 6.10 and 6.11.

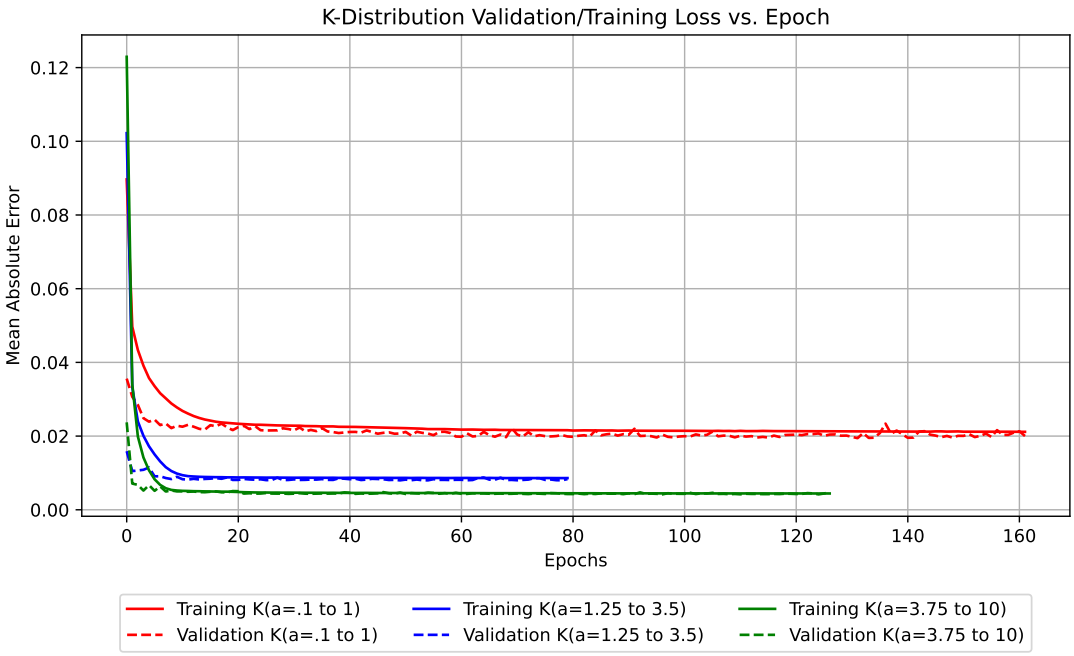


Figure 6.10: K-Distribution Training Loss

After training, each of the Threshold Selectors was integrated with the GLRT detector and performance was verified experimentally. Performance verification was done by generating $100,000$ test samples and associated support samples at each of the distributions/shape parameters listed in Table 6.2. H_0 and H_1 samples were generated and used to verify the P_{FA} and P_d performance of the detectors. Note that the P_d performance was not the focus of this work and was only verified with a SIR of 35dB. The P_{FA} performance is depicted in Figures 6.12 and 6.13, while the P_d performance is shown in Figures 6.14 and 6.15. In these figures, the vertical dashed black lines rep-

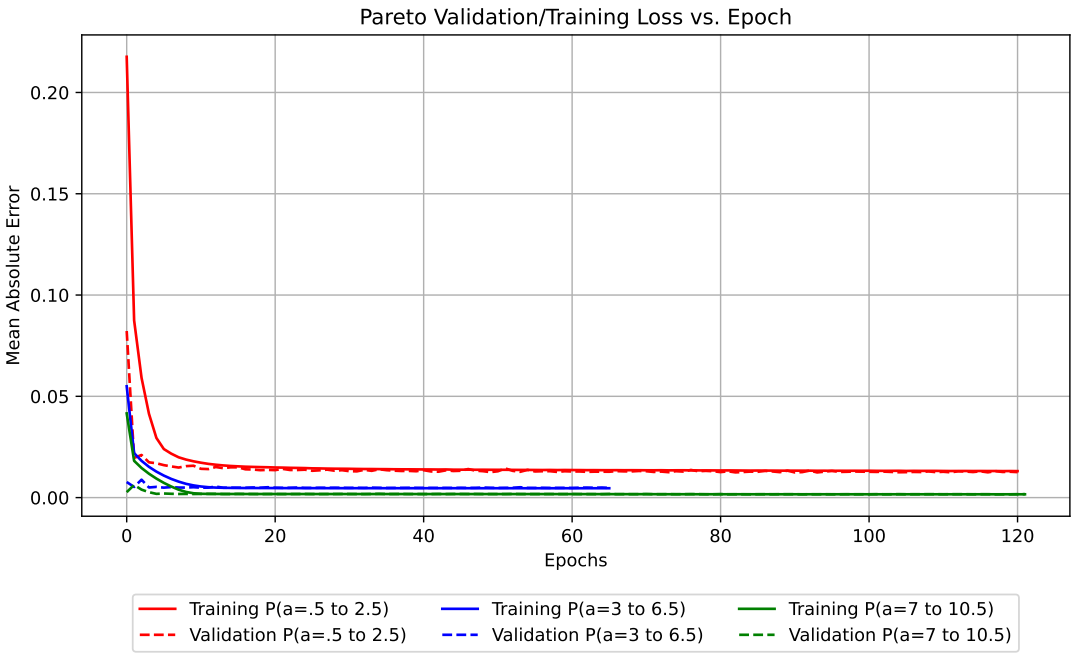


Figure 6.11: Pareto Training Loss

resent the transition points between Threshold Selectors. The vertical dashed red line in Figures 6.12 and 6.14 shows the transition into an extremely heavy-tailed clutter region that adaptive detectors traditionally struggle with. This proved to be a difficult region that I will explore in detail in section 6.3. Additionally our struggle with this difficult heavy-tailed region inspired the follow-on expansion work in section 6.4

6.3 Full System Testing and Results

The previous sections discussed the meta-cognitive architecture and my ideation and motivation to implement each of the sub-components of the architecture. The full system testing was conducted on the fully integrated systems. The process flow for the full system is as follows: 1) the input data is raw I/Q data with a sample or cell under test $\mathbf{z}_{CUT}$ and the 32 x 16 support samples $\mathbf{z}_K$. 2) This data was processed into ideal form in two ways for different sub-components. The ML components utilized

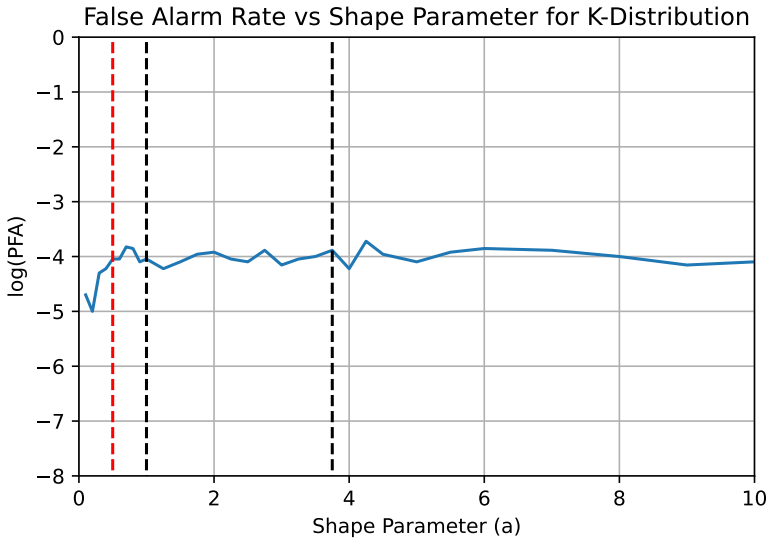


Figure 6.12: False Alarm Rate for K-Distribution Threshold Agents.

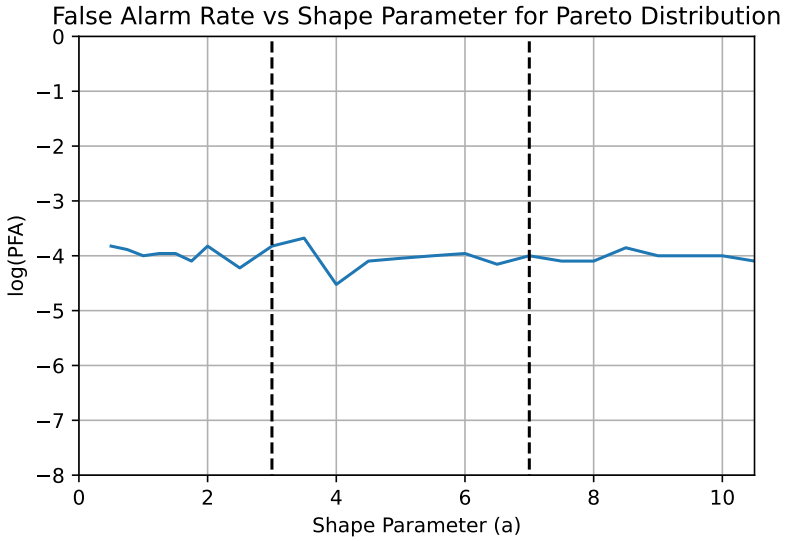


Figure 6.13: False Alarm Rate for Pareto Distribution Threshold Agents.

down-sampled magnitude ordered statistics $\mathbf{z}_{ML}$ from the support samples $\mathbf{z}_K$. The GLRT adaptive detector was passed the cell under test $\mathbf{z}_{CUT}$ and the inverse of the estimated clutter covariance matrix $\mathbf{S}^{-1}$. 3) The Discriminator ANN classifier utilized

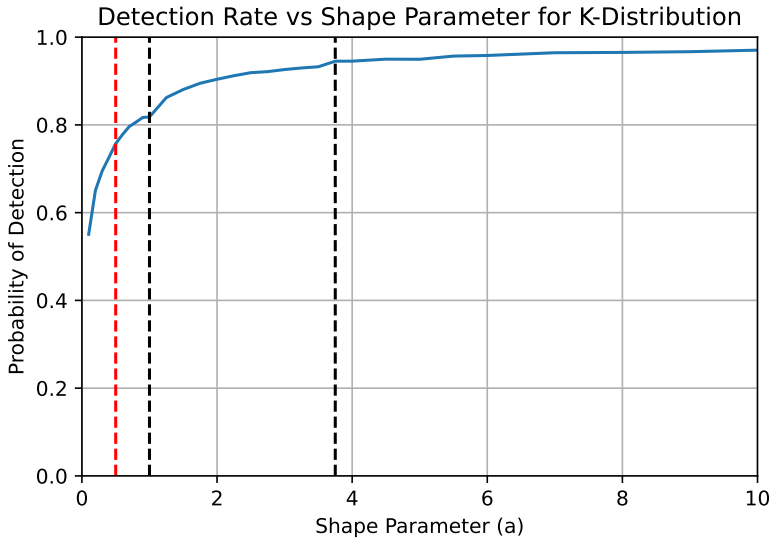


Figure 6.14: Detection Probability for K-Distribution Threshold Agents at SIR = 35dB.

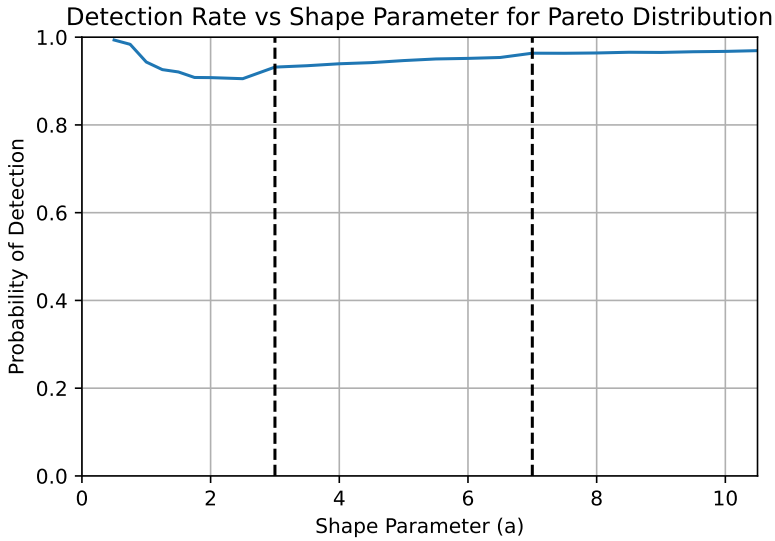


Figure 6.15: Detection Probability for Pareto Distribution Threshold Agents at SIR = 35dB.

$\mathbf{z}_{ML}$ to determine both the distribution and distribution region that the clutter within the support samples most likely belonged. 4) The selected class decided if the threshold was calculated directly for the Gaussian case or which finely tuned Threshold selector ANN was used to determine the optimal threshold for a given probability of false alarm P_{FA}. 5) Finally this Threshold along with the data from step 2 was used to conduct the hypothesis test with the GLRT and determine if a target is present or not.

For the final testing a new dataset was generated with samples containing simulated distribution from any of the three distribution and their shape parameter regions. All clutter data used for training and testing of the proposed system was generated with a one-lag correlation coefficient of $\rho = 0.9$. The system was tested using $500,000$ test samples (with associated sample support) spread across the full region of distributions and shape parameters of interest. $200,000$ of these samples were generated using the K-distribution clutter model, $200,000$ were generated using the Pareto model, and the remaining $100,000$ were generated with the Gaussian model. For each the K- and Pareto distributed clutter data, 100 shape parameters were randomly selected (between 0.5 and 10 for K and between 0.5 and 10.5 for Pareto) and each shape parameter was used to generate $2,000$ samples. These $500,000$ samples were first generated as clutter only (H_0) samples and were used to evaluate the false alarm rate of the detection algorithm. Those same samples were then used to generate (H_1) samples by adding a target as in Equation (6.2) to determine the P_d for the detector over different distributions and shape parameters. For this work, a static steering vector was used for all H_1 test samples with $\mathbf{p}_i = \exp^{-j2\pi(i-1)0.2}$. α was set to ensure the desired SIR for the CUT using [54]:

$$SIR = \alpha^2 \times \mathbf{p}^H \mathbf{M}^{-1} \mathbf{p} \tag{6.9}$$

This process used to select shape parameter covered the range that our system was trained but were uniformly randomly selected and none perfectly matched training

data. This distinction was chosen specifically to test that our system could generalize well and did not over-fit to specific labeled training data. It should also be noted that the region of K-distribution shape parameters used for this testing was limited slightly from what the Threshold Selectors were trained on. Specifically, both the Gaussian GLRT and the ensemble system struggled with the K-distributed clutter samples generated with extremely low shape parameter values ($a = 0.1 - 0.5$). This clutter region is known to be extremely challenging for detection algorithms, and so was ultimately evaluated separately.

The second set of testing data was generated in an effort to explore how well the detector performed with a distribution that it was not trained with to explore its general response. This data was a set of simulated returns containing clutter with a lognormal distribution. The meta-cognitive detector was not trained on lognormal data prior to running these tests, making this a good test of how well the approach generalizes to unknown distributions. The clutter was generated as a set of correlated, complex lognormal random variables. As with the other clutter distributions used for this research, a correlation coefficient of $\rho = 0.9$. Figure 6.16 shows the histogram of the magnitude of the generated lognormal clutter coupled with the PDF of the distribution. This distribution does not fall within the category of SIRVs, but closely resembles a heavy tailed distribution similar to the low-shape parameter K distribution. A total of $100,000$ samples were generated for this generalization testing.

In the third stage of ensemble testing, the system was tested in the most heavy-tailed clutter region considered here: the aforementioned low-shape parameter K-distribution. $500,000$ samples were used in this evaluation, $100,000$ generated from each of the following five shape parameters: 0.1, 0.2, 0.3, 0.4, and 0.5. Once again, the same clutter data was used to generate both H_0 and H_1 test samples for determining the P_{FA} and P_d (respectively) of the algorithm. Performance comparison was conducted using both the Gaussian GLRT and the K_1 GLRT.

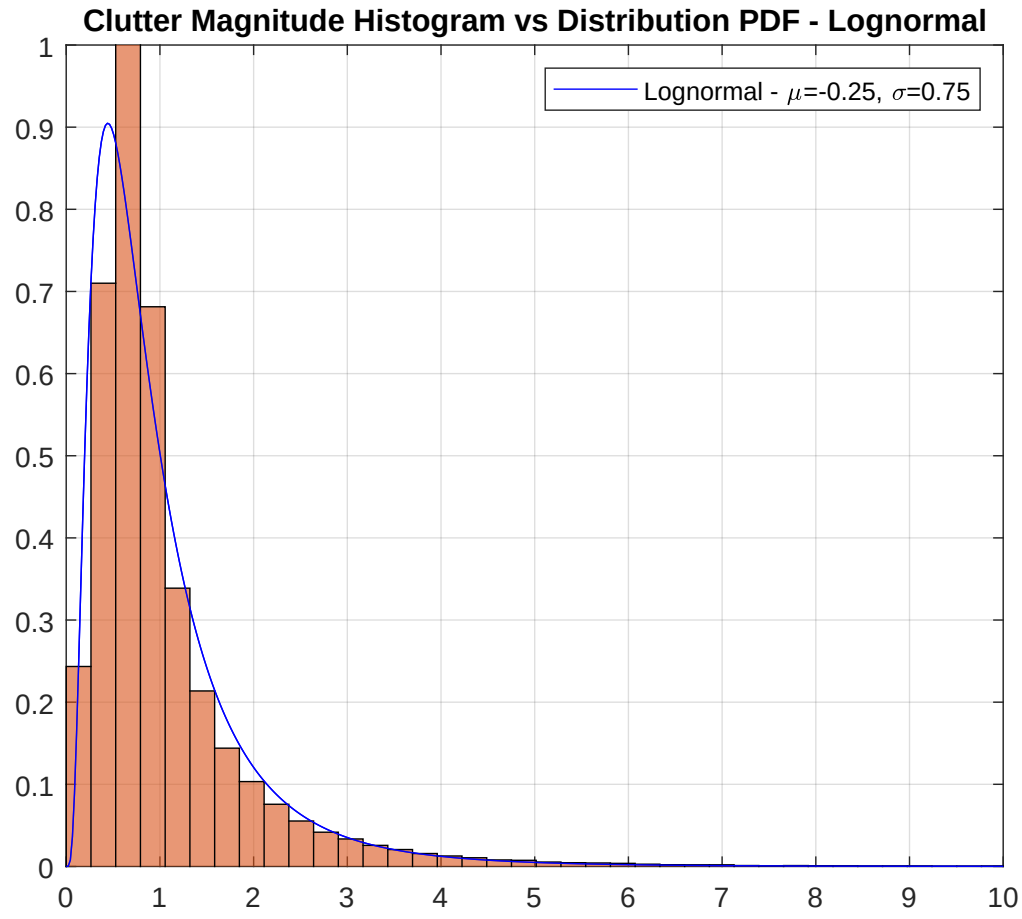


Figure 6.16: Distribution PDF vs Data Histogram - Lognormal Clutter.

In order to fairly test our method we chose to test with other comparable methods. I will note here that I focused on the integration of the full system in python and debugging and testing of the full system. The primary author, Dr. Stringer, implemented the comparison methods including adaptive detectors and a purely-ML based detection system. We used the same data for the comparison methods as used to test our meta-cognitive system. The Gaussian GLRT and the AMF (see section 6.2.2) and a deep learning ML method outlined in two works [130] and [131] and the specific class K1-Threshold selector were used for comparison. We chose these comparison

methods with the goal to addressing our system with fair comparison to both classical adaptive detection methods and a modern ML method that focused on using ML to conduct detection directly. Finally, the K1 region was the most non-Gaussian and heavy-tailed region and provided the highest threshold values. This selector acted as an upper limit for the distributions we tested.

The deep learning method used for comparison presented in [131] is shown from that work in figure 6.17. This method used matched filtered return signals as input to a single neural network. The output of this neural network was then compared to secondary neural network to determine the threshold for a final hypothesis test.

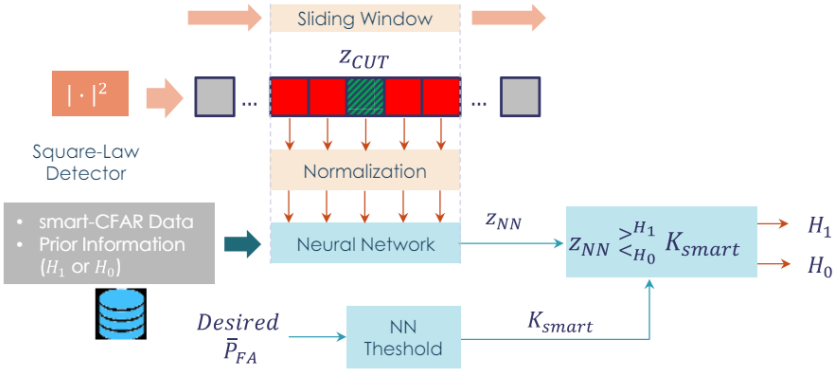


Figure 6.17: Direct NN Method for Comparison.

The following results is for first testing case with $500,000$ samples as well as a final column including the second set with the log-normal distribution. This data set was used to test our meta-cognitive detector, the base line Gaussian GLRT, and traditional AMF, the K_1 GLRT as an upperbound on threshold, and finally the comparison deep learning method (DL Detector). This initial testing had a target probability of false alarm $P_{FA} = 10^{-4}$. These results are shown in tables 6.3 and 6.4.

These initial results are for a single P_{FA} and before exploring the results across a sweep of P_{FA} the meta-cognitive approach showed impressive performance. Importantly, the meta-cognitive detector closely matched the Gaussian GLRT for the Gaus-

Detector	**Simulation Count**	**False Alarms**	**Detections**	$\mathbf{P_{FA}}$	$\mathbf{P_d}$
$Gaussian$					
Meta-Cognitive Detector	$100,000$	9	$98,615$	$\mathbf{0.000090}$	$\mathbf{0.986150}$
Gaussian GLRT	$100,000$	10	$98,692$	0.000100	0.986920
AMF	$100,000$	7	$98,645$	0.000070	0.986450
K_1 GLRT	$100,000$	0	$72,236$	0.000000	0.722360
DL Detector	$100,000$	10	$99,759$	0.000100	0.997590
$K(0.5-10)$					
Meta-Cognitive Detector	$200,000$	21	$186,996$	$\mathbf{0.000105}$	$\mathbf{0.934980}$
Gaussian GLRT	$200,000$	162	$194,636$	0.000810	0.973180
AMF	$200,000$	841	$197,664$	0.004205	0.988220
K_1 GLRT	$200,000$	3	$127,548$	0.000015	0.637740
DL Detector	$200,000$	280	$198,975$	0.001200	0.994875
$Pareto(0.5-10.5)$					
Meta-Cognitive Detector	$200,000$	23	$187,825$	$\mathbf{0.000115}$	$\mathbf{0.939125}$
Gaussian GLRT	$200,000$	132	$194,355$	0.000660	0.971775
AMF	$200,000$	1756	$198,013$	0.008780	0.990065
K_1 GLRT	$200,000$	1	$112,982$	0.000005	0.564910
DL Detector	$200,000$	$1,004$	$198,027$	0.005020	0.990135

Table 6.3: Verification Test Results - Original Gaussian, K , and Pareto

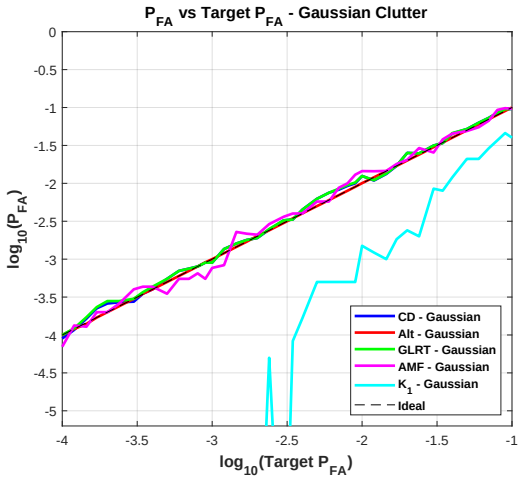
Detector	**Simulation Count**	**False Alarms**	**Detections**	$\mathbf{P_{FA}}$	$\mathbf{P_d}$
Full Test - $Gaussian + K(0.5-10) + Pareto(0.5-10.5)$					
Meta-Cognitive Detector	$500,000$	53	$473,436$	$\mathbf{0.000106}$	$\mathbf{0.946872}$
Gaussian GLRT	$500,000$	304	$487,683$	0.000608	0.975366
AMF	$500,000$	2604	$494,292$	0.005208	0.988584
K_1 GLRT	$500,000$	4	$312,766$	0.000008	0.625532
DL Detector	$500,000$	$1,294$	$496,761$	0.002588	0.993522
$Lognormal$					
Meta-Cognitive Detector	$100,000$	52	$91,693$	0.000520	0.916930
Gaussian GLRT	$100,000$	378	$95,890$	0.003780	0.958900
AMF	$100,000$	4398	$99,640$	0.043980	0.996400
K_1 GLRT	$100,000$	14	$87,914$	0.000140	0.879140
DL Detector	$100,000$	$1,390$	$98,765$	0.013900	0.987650

Table 6.4: Verification Test Results - Full Range of Clutter Distributions and Additional Lognormal Case

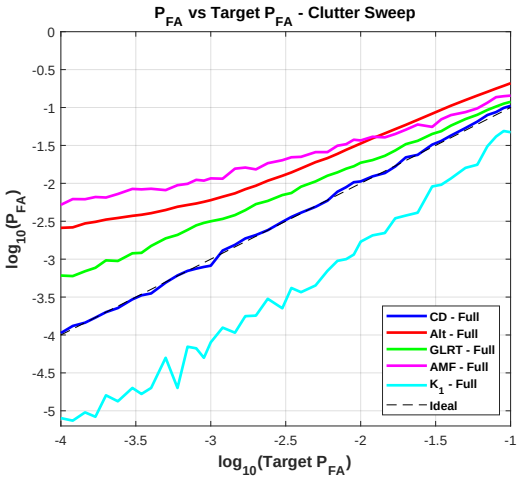
sian region. This shows the effectiveness of the discriminator and establishes that the system does not lose performance for the idealized Gaussian case. This is expected if the discriminator was accurate in identifying Gaussian clutter as the MC detector used the Gaussian GLRT. The K-distributed, Pareto, and full testing sections are the most impressive results. The MC detector maintained a CFAR performance across all of the proposed regions while the other comparative methods did not. This confirms the performance goal of added flexibility to the detector given changing distributions and those distributions shape parameter.

Although these results are impressive we wanted to confirm that it was consistent across P_{FA}. Therefore, additional testing was performed to generate receiver operating characteristics (ROC) curves for each of the detectors. The Gaussian-only and Gaussian/K/Pareto sweep data sets were each run through the detectors over a range of target P_{FA}s from 10^{-4} to 10^{-1}. A total of 40 different P_{FA} values were tested with a linear increase in the logarithmic scale. This test was conducted with constant SIR values of 35 dB and 10 dB. The P_D and P_{FA} for each detector were recorded and plotted against the target P_{FA} in Figures 6.18-6.20. The proposed detector is labeled as CD, the alternative detector of [130, 131] as Alt. This testing again demonstrates the superior performance of the proposed meta-cognitive detection algorithm.

When analyzing the ROC curves it is important to establish that the primary design goal is for CFAR performance. Therefore the closer that a detector performs to the ideal line ($P_{FA} = P_{fa,target}$) in the P_{FA} plots (Figure 6.18) the better. These P_{FA} plot show that the impressive performance of the CD detector is consistent across a sweep of target P_{FA}. The comparative methods, DL approach, GLRT, and AMF all perform well for the Gaussian case 6.18a but for the full sweep 6.18b these methods suffer as they are unable to be flexible for the range of distributions and shape parameter present. The P_D ROC curves are calculated for two different signal to interference ratio (SIR) values. As P_D is not the primary design parameter it will be inversely

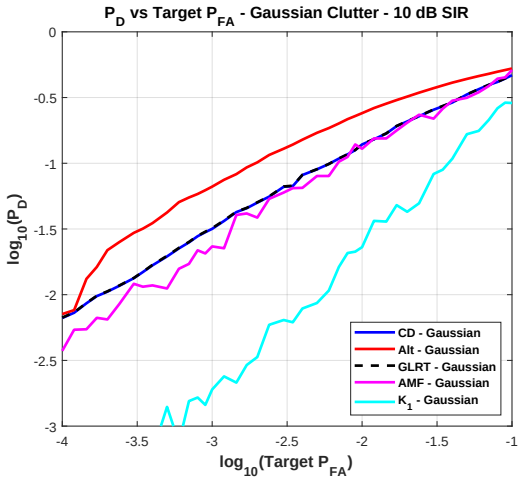


(a) ROC Curve - P_{FA} vs Target P_{FA} - Gaussian Clutter.

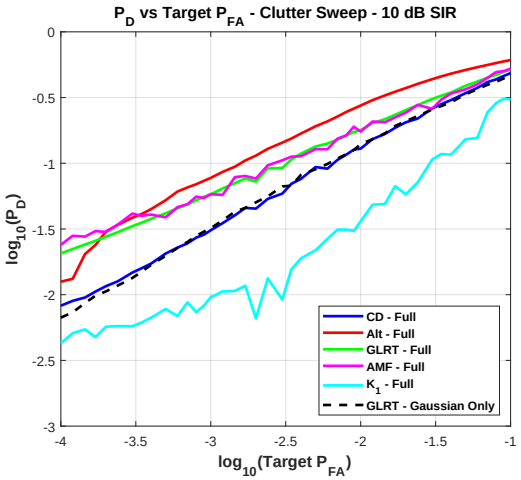


(b) ROC Curve - P_{FA} vs Target P_{FA} - Clutter Sweep.

Figure 6.18: ROC Curves Showing the Measured P_{FA} vs the Target P_{FA}.

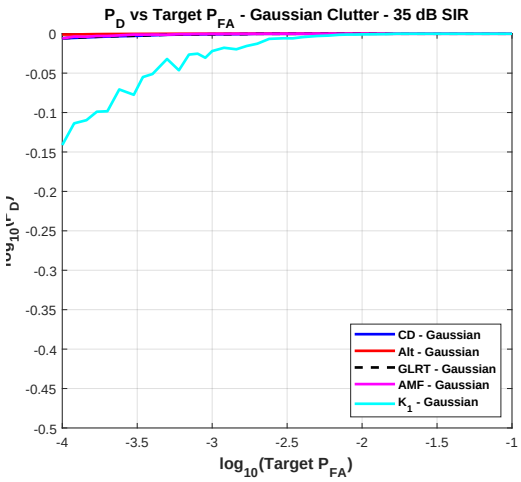


(a) ROC Curve - P_D vs Target P_{FA} - Gaussian Clutter.

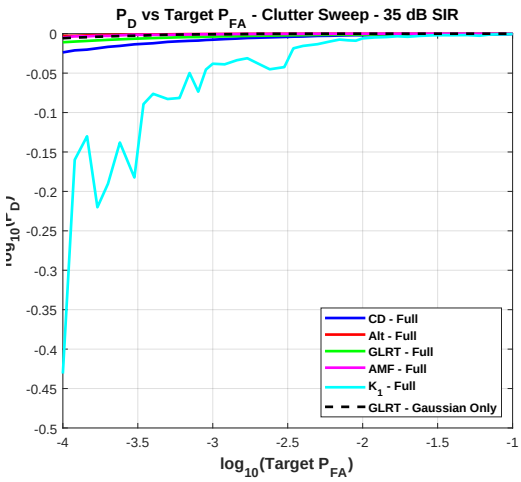


(b) ROC Curve - P_D vs Target P_{FA} - Clutter Sweep.

Figure 6.19: ROC Curves Showing the Measured P_D vs the Target P_{FA} with 10 dB SIR.



(a) ROC Curve - P_D vs Target P_{FA} - Gaussian Clutter.



(b) ROC Curve - P_D vs Target P_{FA} - Clutter Sweep.

Figure 6.20: ROC Curves Showing the Measured P_D vs the Target P_{FA} with 35 dB SIR.

related to the P_{FA} performance. Additionally, we see that the P_D increases with higher SIR for all methods.

As the the K_1 GLRT expects extremely heavy tailed clutter, it naturally sets a high threshold and therefore has a lower P_{FA} than desired. The loss of sensitivity causes corresponding lower probability of detection, as shown in Figure 6.19. Even more interesting is the observation that for both the low SIR and high SIR cases the P_D for the proposed detector demonstrates no variation between the Gaussian-only and full distribution sweep cases. Therefore, the meta-cognition allows the proposed system to maintain desired performance across the variety of clutter types it was trained with. Further, if Gaussian clutter is present, there is no penalty for using the proposed approach relative to an ideal detector.

The second state of testing focused on testing the meta-cognitive detector and the comparison methods to a new distribution that the MC detector was not trained with.

The goal of this testing was to see how well the discriminator and threshold selectors could perform with a related distribution but not one specifically seen during training. This testing was performed on a set of simulated returns containing clutter with a lognormal distribution. The meta-cognitive detector was not trained on lognormal data prior to running these tests, making this a good test of how well the approach generalizes to unknown distributions. The clutter was generated as a set of correlated, complex lognormal random variables.

A test set of $100,000$ samples was generated and run through each detector twice; once with a target signal added to each sample and once with no added target. This process was repeated over a range of P_{FA} values ranging from 10^{-4} to 10^{-1}. The ROC curves for this testing are shown in Figures 6.21 and 6.22. It is worth noting that the proposed approach has degraded P_{FA} performance in this scenario. However, from examination of Figure 6.21 it is much closer to the ideal than the Gaussian GLRT, AMF, and the alternative DL approach. Only the K_1 threshold selector and GLRT pair has better performance. This makes sense, given that the lognormal distribution is heavy-tailed and most closely matched the low shape region of the K-distribution. This tells us that the discriminator struggled to consistently classify the lognormal distribution as similar to the K_1 region. Regardless, our CD method still out performed the other comparison methods. Note that in Figure 6.22 the performance is plotted against the *target* P_{FA}, not the *actual* P_{FA}. Therefore, while the AMF, GLRT, and alternative DL method will have better sensitivity for a *desired* probability of false alarm, their actual probability of false alarm will be the much higher value shown in Figure 6.21.

The final stage of tested was to break down the low K region $(0.1 - 0.5)$ as initial testing showed that this region had problematic performance for every method including the focused K_1 GLRT. The results for the heavy-tailed clutter region of the K-distribution with extremely low shape parameter is shown in Tables 6.5 and 6.6. As

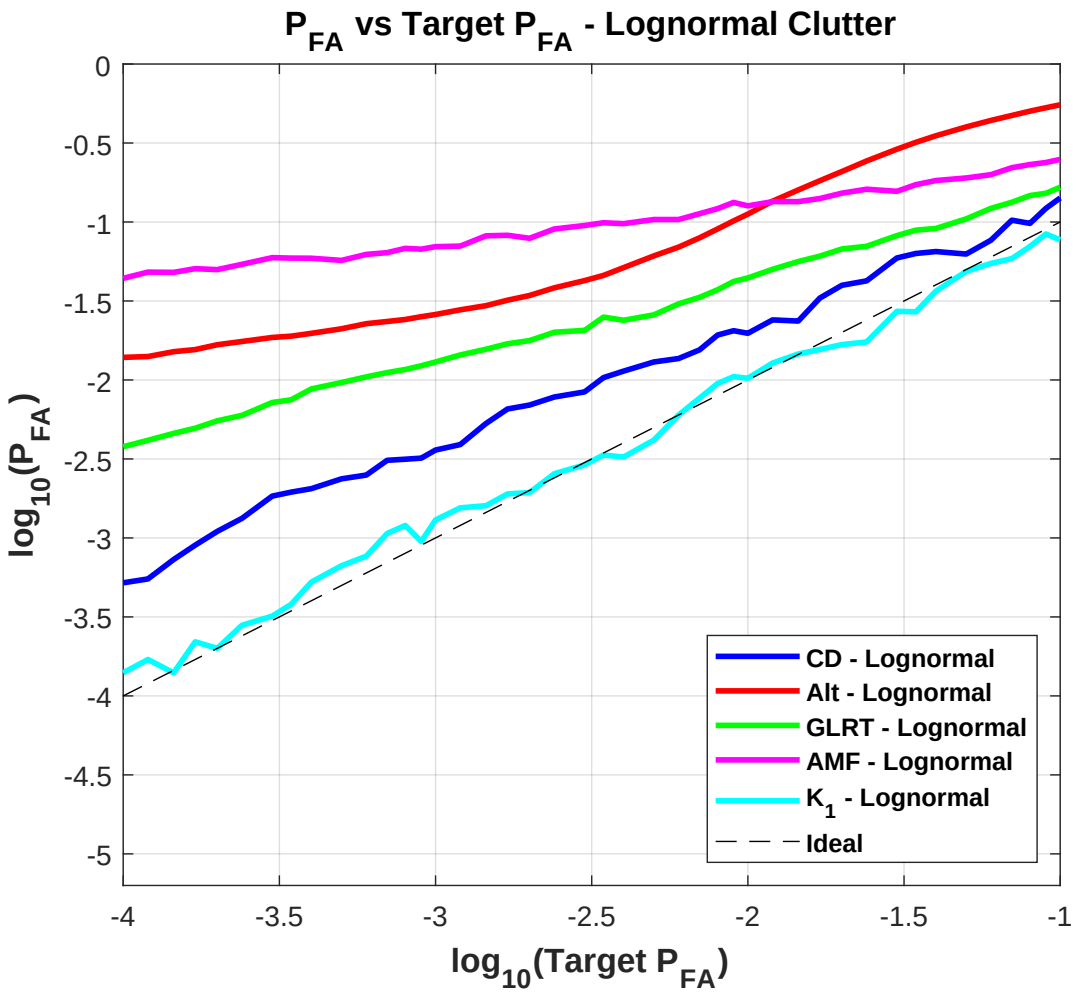
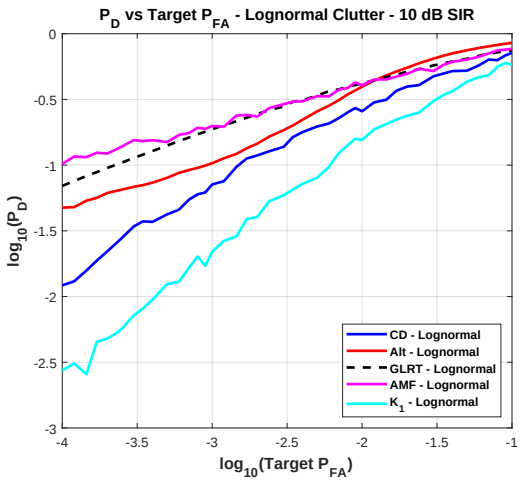
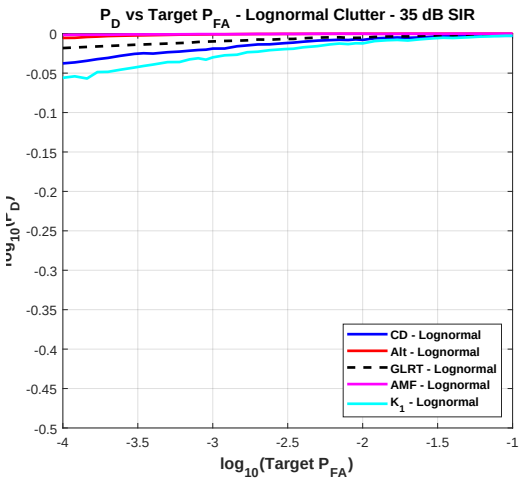


Figure 6.21: ROC Curve - P_{FA} vs Target P_{FA} for Lognormal Clutter.

the shape parameters approach 0 the ability for the Meta-Cognitive Detector to maintain target P_{fa} degrades, but it still out performs the Gaussian GLRT. The targeted filter K_1 is able to maintain the target P_{fa}. Considering the positive performance of the K_1 filter but the degradation of the Meta-Cognition Detector it is likely that the Discriminator has issues correctly classifying these low shape parameter considering it was trained to identify the average shape parameter ($a = .55$) in the region of K_1. Given that this region has high volatility, training the Discriminator with an additional class in this low region could result in better selection and full meta-cognitive results



(a) ROC Curve - P_D vs Target P_{FA} for Lognormal Clutter with 10 dB SIR.



(b) ROC Curve - P_D vs Target P_{FA} for Lognormal Clutter with 35 dB SIR.

Figure 6.22: ROC Curves Showing the Measured P_D vs the Target P_{FA} for Lognormal Clutter.

that more closely match the K_1 GLRT results.

This research and exploration into an adaptive detection challenge was both novel and impactful in the detection domain as well as foundational in establishing multiple core principles to this dissertation's focus multi-agent/objective radar tasks like detection. Adaptive detection research has developed multiple ways to detect in different distributions beyond the default assumption of Gaussian clutter. But it requires a prior knowledge about the distribution and more specifically an estimation of the characteristic parameters for that distribution. (i.e. shape and scale) This results in adaptive detectors that are performant for a known region, but the core trade off of hypothesis testing is that an incorrect threshold will either violate a CFAR design requirement or impact the P_D. An ideal adaptive detector would maintain CFAR and have a consistent P_{FA} as close to the CFAR target P_{FA} as possible. This meta-cognitive detector shows results that accomplish this ideal goal for three distributions.

Detector	**Simulation Number**	**False Alarms**	**Detections**	$\mathbf{P_{fa}}$	$\mathbf{P_d}$
$K(0.1-10)$					
Meta-Cognitive Detector	$200,000$	220	$185,276$	$\mathbf{0.001100}$	$\mathbf{0.926380}$
Gaussian GLRT	$200,000$	782	$194,478$	0.003910	0.972390
K_1 GLRT	$200,000$	1	$126,488$	0.000005	0.632440
$K(0.5)$					
Meta-Cognitive Detector	$100,000$	10	$75,520$	$\mathbf{0.000100}$	$\mathbf{0.755200}$
Gaussian GLRT	$100,000$	$1,277$	$95,054$	0.012770	0.950540
K_1 GLRT	$100,000$	10	$75,504$	0.000100	0.755040
$K(0.4)$					
Meta-Cognitive Detector	$100,000$	12	$72,840$	$\mathbf{0.000120}$	$\mathbf{0.728400}$
Gaussian GLRT	$100,000$	$2,040$	$94,896$	0.020400	0.948960
K_1 GLRT	$100,000$	9	$72,807$	0.000090	0.728070
$K(0.3)$					
Meta-Cognitive Detector	$100,000$	43	$69,407$	$\mathbf{0.000430}$	$\mathbf{0.694070}$
Gaussian GLRT	$100,000$	$3,502$	$94,837$	0.035020	0.948370
K_1 GLRT	$100,000$	10	$69,031$	0.000100	0.690310

Table 6.5: Verification Test Results - Extremely Heavy-Tailed K-Distributed Clutter

Detector	**Simulation Number**	**False Alarms**	**Detections**	$\mathbf{P_{fa}}$	$\mathbf{P_d}$
$K(0.2)$					
Meta-Cognitive Detector	$100,000$	981	$68,772$	$\mathbf{0.009810}$	$\mathbf{0.687720}$
Gaussian GLRT	$100,000$	$7,358$	$95,110$	0.073580	0.951100
K_1 GLRT	$100,000$	8	$64,859$	0.000080	0.648590
$K(0.1)$					
Meta-Cognitive Detector	$100,000$	$13,995$	$83,488$	$\mathbf{0.139950}$	$\mathbf{0.834880}$
Gaussian GLRT	$100,000$	$19,290$	$96,204$	0.192900	0.962040
K_1 GLRT	$100,000$	4	$54,825$	0.000040	0.548250

Table 6.6: Verification Test Results - Extremely Heavy-Tailed K-Distributed Clutter - Continued

The detector did train on simulated data for those distributions but when tested it was able to identify the distribution and its shape region and then select an appropriate threshold. The result is an impressive detector that can handle a changing clutter environment. Additionally, the design is modular and expandable allowing for additional distributions and other detector methods to be used. The core idea was Dr. Stringer's work, but I personally explored, designed, and tested the novel discriminator. This discriminator was instrumental in the success of the adaptive detector as it provided the a priori distribution and shape classification needed for the finely-tuned methods to effectively perform. I will note that the integration testing and debugging of the system was a collaborative effort between myself and Dr. Stringer.

One challenge that we explored was the low shape, heavy-tailed region of the K-distribution. The discriminator was trained to the average shape ($a = .55$) in that region ($a = [.1 - 1]$). This proved to be too generalized for the heavy-tailed region as the discriminator classified $a = [.5 - 1]$ but as the shape approached 0 it change greatly. If the regions near 0 had more granularity and were identified and classified as their own regions then the discriminator could have been trained to identify them more accurately. This led to a follow-on effort (discussed in section 6.4) that developed a divergence based method to determine distribution regions instead of selection through expert analysis of JD divergence heat maps and threshold sweeps.

This was the first research I conducted for this dissertation. The Discriminator selected the best distribution region given its situational awareness of the environment. This work also highlighted the power and importance of modularity. The structure of this detector allowed for other distributions and regions to be added in the future through a retraining effort. Additionally, the system did not need to use the GLRT as the hypothesis testing method, our ML threshold selectors, or our ML discriminator. Each of those components needs to accomplish an action, but could be replaced with a different method or a human user. This work resulted in a follow-on effort to introduce

scalability and contributed to my RRM research.

6.4 Dynamic Decentralized Expansion

In the previous sections 6.1 to 6.3, I discussed the collaborative work on a MC adaptive detection framework that we presented in Stringer *et al.* [23]. Following this effort my research partner, Dr. Stringer, and I each decided to tackle two extensions to the MC design. Together we discussed areas of improvement and identified the challenge of low sample support detection and a more application focused approach to the MC detector. Dr. Stringer led the effort to investigate methods to generate the inverse CCM directly by utilizing generative ML. I supported his work with ideation, testing data generation, peer review and paper writing/editing but they were the primary architect and contributor. The initial work was presented at the radar conference 2024 in Denver [129] and additional testing and development was later presented in Dr. Stringer's dissertation [132].

I led the second expansion to the MC design with support from my colleague, Timothy Sharp. This expansion focused on the application side of the MC detector. The primary challenge space was to address the fact that the original design required the effort of domain experts (myself and Dr. Stringer) to design the architecture and determine the distribution regions. Additionally the process for building the original system included a total of seven ANNs that were each trained and then integrated together. Although the system worked it was not efficient or scalable and would require large amounts of monitoring to train and verify each of the important task and tactics based subsystems. The following sections address the work we completed to address these challenges and will describe the micro-service application I developed.

6.4.1 Dynamic Sizing Method

This expansion to the original MC detector first address the process for determining the regions for distributions. The previous work determined the regions with expert analysis of GLRT threshhold values given a sweep of SIRV shape parameter for a distribution. In the previous work I conducted this sweep for the K-distribution and the Pareto distribution for shape values $a = 0.5$ to 10 and $a = 1$ to 10.5 respectively. Additionally the halving algorithm implemented in 1 was tuned to a $P_{FA} = 10^{-4}$. As an additional form of verification I conducted the statistical divergence with the Jensen-Shannon divergence as described in 6.2.3. This method proved to be effective at identifying unique statistical regions given shape parameter for SIRVs resulting in ML models that could classify unique regions and train finely-tuned threshold selectors. However, this method was neither scalable or held a quantitative foundation that could be trusted and repeated. This inspired me to explore a divergence-based measure to determine some n number of regions that had the greatest statistical uniqueness.

Given that I utilized divergence methods in the original MC work, I decided to dig into the mathematical representations and theorems related to SIRVs. I began with the Jensen-Shannon divergence method due to its symmetric nature. While exploring mathematical derivation for the the JS divergence and the SIRV form $\mathbf{z} = \sqrt{\tau}\mathbf{x}$ we realized that the resulting derivation became overly complicated and the Gaussian speckle x could not be separated from the characteristic function easily. This led to an exploration of the KL divergence the JS divergence uses. The KL divergence is not symmetrical but an aspect of the Gamma family of SIRV is that as the shape parameter approaches infinity the SIRV will become more Gaussian. It was this observable feature in our work that led to analyzing the theorems and properties of SIRV and the KL divergence if the comparison distribution Q is Gaussian. This work resulted in a region measure that utilized the KL divergence value for SIRV distributions with respect to a Gaussian distribution to determine the optimal splitting point for distribution

regions.

This new measurement method using the KL-divergence with SIRVs is explained using the following assumptions and theorems:

1. Given a SIRV with the form $\mathbf{z} = \sqrt{\tau}\mathbf{x}$, with $\sqrt{\tau}$ representing the texture and the zero mean Gaussian vector $\mathbf{x}$ as the speckle.

2. The corresponding PDF for the gamma family of SIRVs takes the form: $r_Z(\mathbf{z}) = r_\tau(\tau(a, b, \rho))r_X(\mathbf{x})$ where a is the shape parameter, b is the scale parameter, ρ is the correlation coefficient.

3. According to Theorem 2, there exists a linear transformation $p_Z = r_Z\mathbf{A} + \mu$ such that $p_X(\mathbf{x})$ = $r_X(\mathbf{x})$ and $p_\tau(\tau)$ has zero mean and a standard deviation of 1. It can be assumed that $p_\tau(\tau)$ and $p_X(\mathbf{x})$ are independent.

4. A PDF, $q_z(z) = q_\tau(\tau)q_X(\mathbf{x})$ can be defined such that $q_\tau(\tau) = \mathcal{N}(0,\, 1)$ and $q_X(\mathbf{x}) = p_X(\mathbf{x})$.

5. Equation (2.41) then simplifies to:

$$\begin{aligned} &= D_{KL}(p_\tau(\tau)\|\mathcal{N}(0,\, 1)) + D_{KL}(p_X(\mathbf{x})\|p_X(\mathbf{x})) \\ &= D_{KL}(p_\tau(\tau)\|\mathcal{N}(0,\, 1)) \end{aligned} \tag{6.10}$$

6. It is established in detection literature that the texture and speckle elements vary on different time scales - speckle varies within the CPI while texture is assumed to remain constant within the CPI

7. As a result, each pulse is effectively defined as the combination of a Gaussian SIRV and a constant sampled from another SIRV.

8. When the Gamma distribution is defined by a shape parameter, a, it is well known that as $a \to \infty$, the Gamma distribution flattens out and effectively becomes uniform with μ and σ going to infinity. Note that in the special case

where μ and σ are held constant while $a \to \infty$, then the Gamma distribution becomes $\sim \mathcal{N}(\mu,\, \sigma^2)$.

9. Given this behavior, if we take the KL divergence of $\Gamma(a \to \infty)$ with respect to a uniform distribution (as in (2.41)), this will tend towards a constant value.

10. The transformation of any non-zero constant will also be a constant.

11. For the Gamma family of SIRVs the characteristic functions are characterized by linear transformations of the Gamma distribution.

12. As a result, $p_\tau(\tau(a \to \infty)) \to \mathcal{N}(0,\, 1) = q_\tau(\tau)$.

13. Therefore as $a \to \infty$ for the Gamma family of SIRVs then the KLD will approach a minimum constant. $p_\tau(\tau(a \to \infty)) \to \mathbf{U}$

14. Additionally, the KLD for this family of SIRV will be monotonically decreasing as $p_\tau(\tau(a \to \infty)) \to q_\tau(\tau)$.

15. Given this behavior we can define some ϵ divergence corresponding to a γ confidence in statistical uniqueness of a given Gamma family SIRV for a range of shape parameter. This ϵ will correspond to a unique region δ_n. See Figure 6.23

Algorithm 2 Dynamic Splitting

Given: (a_{min}, a_{max}, n)

$D_G(a) = D_{KL}(p_{\vec{y}}(a) \| p_{\vec{x}})$

$\epsilon = (D_G(a_{min}) - D_G(a_{max}))/n$

$\delta_{1,min} = a_{min}$

for $i \in \{1, 2, 3, ...n\}$ **do**

$\delta_{i,max} = D_G^{-1}(D_G(\delta_{i,min}) - \epsilon)$

$\delta_{i+1,min} = \delta_{i,max}$

end for

Algorithm 2 represents my method for determining the δ_n regions for a given range

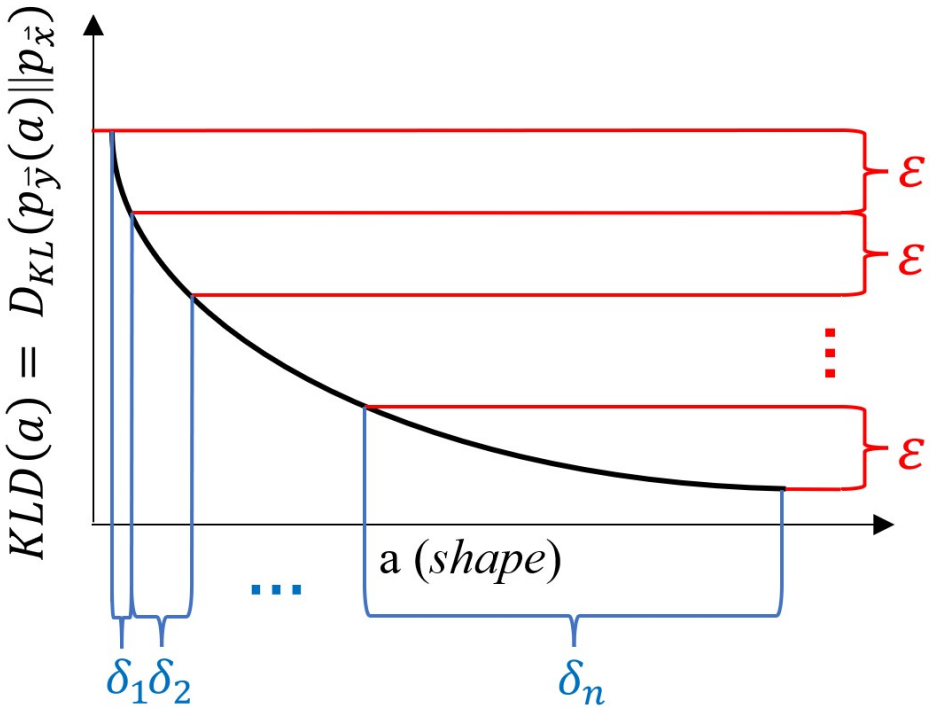


Figure 6.23: Regions Given ϵ Range for KLD Shape Parameter

of shape for a SIRV utilized by our system. The KLD with respect to a SIRVs speckle is represented with $D_G(a) = D_{KL}(p_{\vec{y}}(a)\|p_{\vec{x}})$. The use of the inverse of this KLD calculation ($D_G^{-1}(\cdot)$) in practice is accomplished empirically. Given the target KLD, as an increment of ϵ, the corresponding shape is calculated within a tolerance. This process uses a support file defined in section 6.4.2.

The above theorems and logical progression establish a foundation for using Algorithm 2 to determining the splitting locations that provide n total regions with equal range, ϵ in KLD and therefore the equal but maximum statistical uniqueness for those regions assuming that the KLD is an appropriate measure of statistical uniqueness in this case. I would make the argument that it is sufficient given the relationship of Gamma family SIRVs as their shape parameter approached infinity and the Gaussian speckle component. Additionally, our original MC design showed effective perfor-

mance using the JSD which is a modification on the KLD.

6.4.2 Microservice Architecture

The previous section focused on the need to determine and n regions in a given SIRV distribution range for the MC system to be dynamic in scaling and algorithmically self-forming. The second need for a application that can be generically utilized also needs an architecture that could apply to different detection requirements and needs. To do this I worked with my research collegue, Timothy Sharp, to build and implement a micro-service architecture. Blinowski *et al.* in [133] expands upon the importance of microservice scalability and Hasselbring *et al.* [134] makes a strong case for its agility and reliability. This architecture would used Algorithm 2 to determine optimal distribution regions and then automatically train the associated discriminator and threshold selectors for a new MC system. My implementation used the containerization software Docker to build and deploy our containerized microservice architecture. I designed, optimized and translated the internal code for the majority of the microservice architecture. This included generation of the testing and data support files. The containerization of the ML, full system testing, and supporting translation was performed by Timothy Sharp.

Individual pieces of the MC system were containerized using Docker, creating self contained applications. Figure 6.24 shows the layout of the containerized system. The baseline system configuration contains three main levels when initialized, a discriminator container, an arbitrary number of threshold selection containers, and a GLRT container. These containers pass data through internally exposed network ports. The system also ingests data into the overall system by exposing three external network sockets. These are needed to input the radar data into the system, starting with the pre-processed ordered statistics being sent to the discriminator. The ANN discriminator uses the ordered statistics to select one of the threshold selection containers

and forwards the ordered statistics to that container through the correct port. One of these threshold selection containers receive the ordered statistics and passes it through a threshold selector ANN. The threshold selection container then sends the output of the selector ANN to the GLRT container. The GLRT container then computes the detection determination with the threshold selector output and the covariance matrix input from the Data Source on port 65310. This detection determination is then returned as a message on the output port 65300.

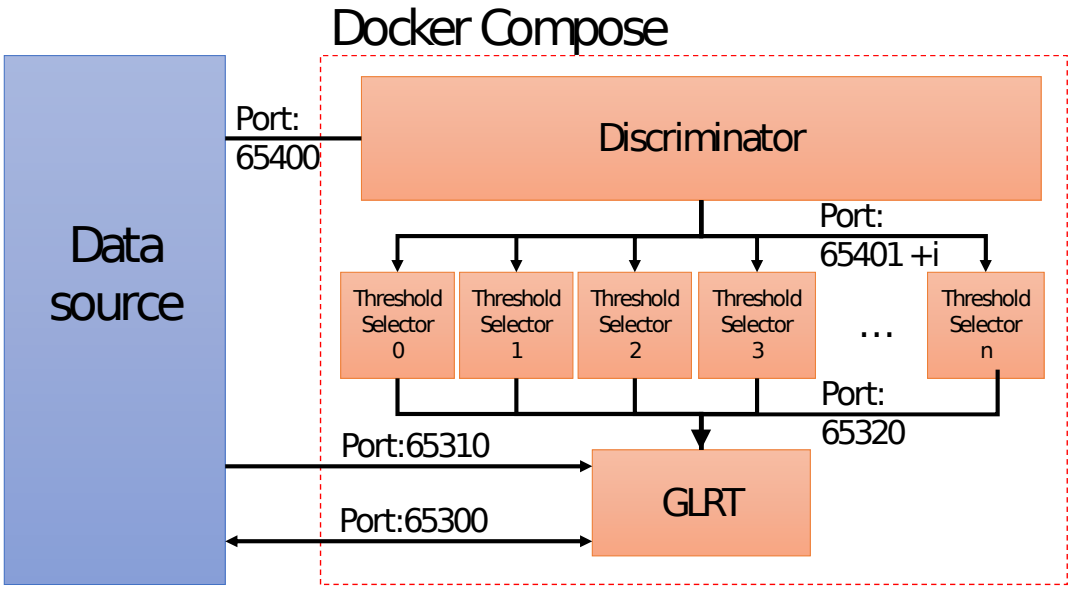


Figure 6.24: Layout of the Containerized Programs.

Figure 6.25 shows the containerized microservice architecture used for training. This architecture was divided into a sequence of jobs, the first of which was the splitting algorithm. The splitting algorithm uses algorithm 2 from the dynamic sizing method above to generate the region bounds for each parameterized distribution.

In order to optimize the training process and allow for dynamic regions, three support files were pre-populated and used by the system. The first support file is used by the splitting algorithm and contains the values of D_G (KLD with respect to a SIRV and it's Gaussian speckle) for the different distributions. These files contained dis-

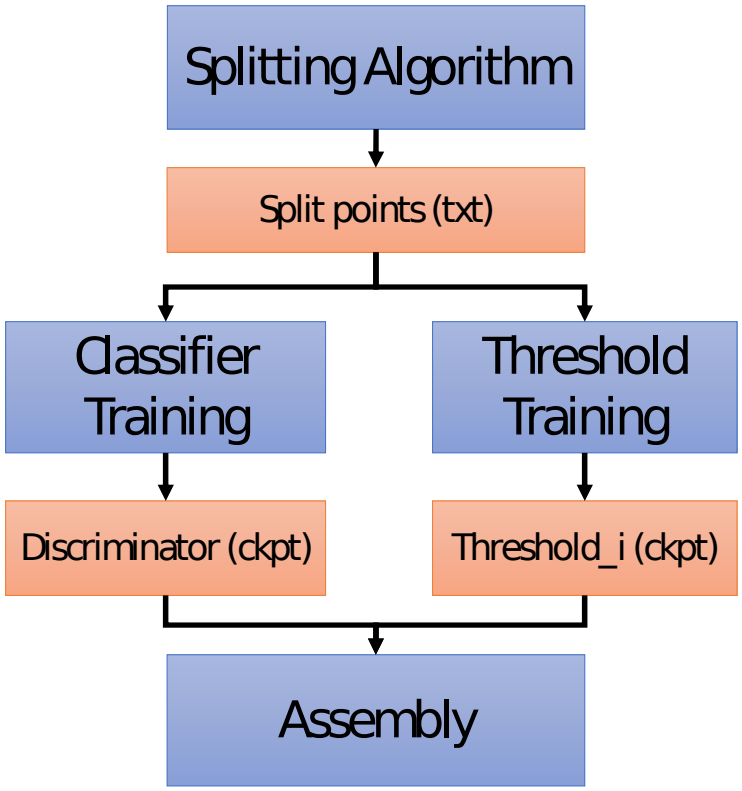


Figure 6.25: Flow Diagram of the Training Process.

crete empirical calculations of D_G for a sweep of shape parameter in K-distribution and Pareto. The second database file also contained empirically calculated threshold values for different shape parameters using the halving algorithm 1 from the original MC method [23]. The method presented here deviated from the original by using a stopping margin of 5% and a data size of $20/P_{fa,target}$. This decision was made to reduce the excessive computation required, but proved to have impacts on results by increasing noise. This file was generated for a sweep of shape parameters from K-distribution$= [.1, 10]$ and Pareto$= [1.05, 10]$ with a sweep delta $\Delta_a = [.001, .01, .1]$ for general regions of $[a \leq 2, 2 < a \leq 4, 4 < a]$ in shape. This file took extensive processing to generate empirically, but once generated it was referenced by the dynamic system to train custom threshold ANNs for any sub-regions given the splitting

algorithm. The final support file is a database file with i.i.d. training samples for a sweep of shape parameters with the same delta and range as the threshold label file.

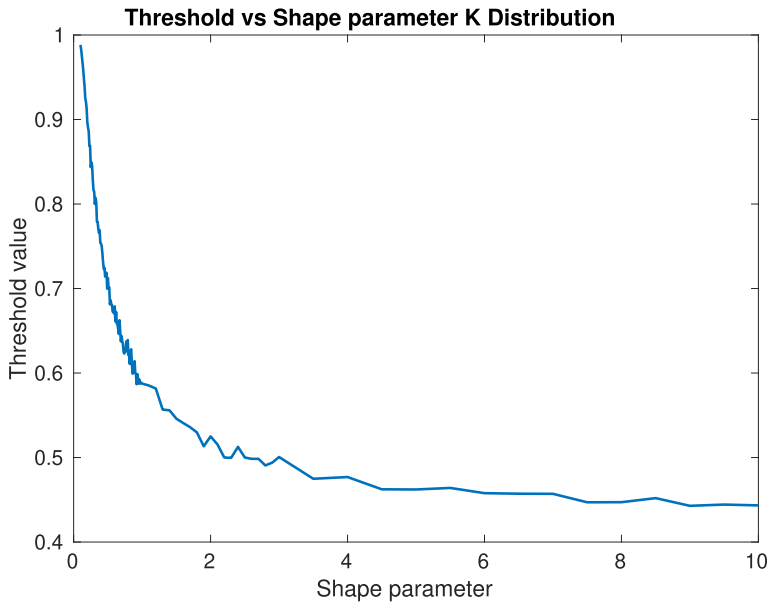


Figure 6.26: Threshold vs. Shape Support File (K-distribution)

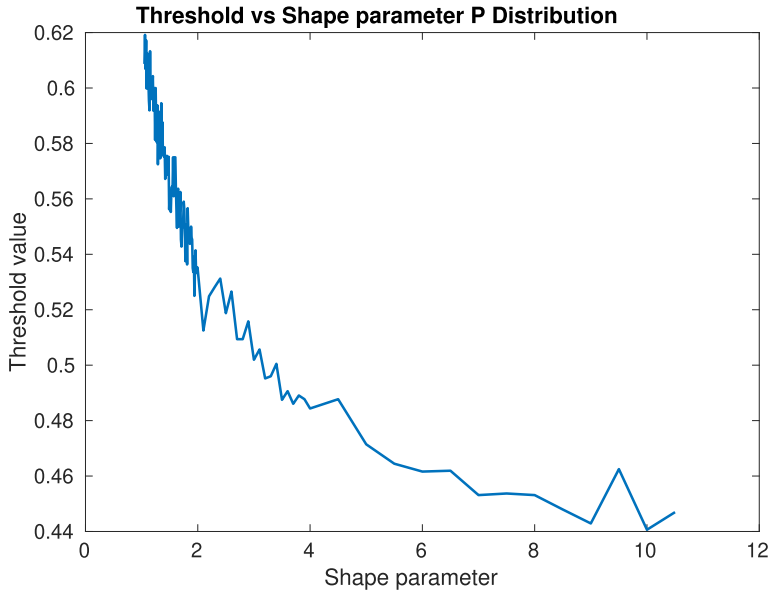


Figure 6.27: Threshold vs. Shape Support File (Pareto)

After the splitting container determines the region boundaries, it saves its results

in a text file. Next there are two training containers that run in parallel, one for the discriminator and one for all the threshold selectors. The discriminator container uses the database support file to build a uniformly distributed database. Appropriate labels are created from the splitting region input. The threshold training container trains each threshold setter in series, since these train much faster then the discriminator. This container builds a database containing training data and label pairs for each threshold ANN. To do this, the shape truth values are matched between the database and threshold label support files. After the two containers finish training all the models, a final assembly container copies all the necessary python scripts, models, and Docker files into a single directory and compresses it into an archive file. This archive file is the output of the training job, and contains the entire system in a runnable state. This results in the entire system being trained and able to process data by changing the input configuration file in two commands.

In general the dynamic system runs in a similar data flow as the original system but includes a few optimization and application based improvements.

6.5 Dynamic Expansion Results

Testing the system was optimized with the generation of testing data files that span the expected distributions and shape regions. A python script was created to iterate through the testing database file, sending data in to the system one at a time and recording the results. The testing database file contained 10^6 data samples, each with 64 length ordered statistics data, an approximated covariance matrix, and one 16 length cell under test. The ordered statistics and the covariance matrix were derived from 32 support sample returns, each of length 16, to conform with the data from the original MC paper. The 10^6 samples were generated across the Gaussian, K, and Pareto distributions, with 20%, 40%, 40% of the total data respectively. Of the $4 \cdot 10^5$ K and Pareto data samples, a weighted approach was taken across the shape parame-

ters, with more samples generated in the lower region. This was done as the splitting algorithm will split more heavily in regions with a higher threshold as shown in Figures 6.26 and 6.27, and would generate more threshold selector models in this region. The rejection method used the threshold curve and scaled the values with a uniform offset of $.25$. The result was a PDF representative of the range for the threshold values but included a minimum number of samples for all values of shape parameter.

In order to test the dynamic system we utilized the testing database with a spread of 10^6 samples described above. We tested three dynamic split configurations of 3 regions, 5 regions, and 10 regions in both K-distribution and Pareto. These were tested against the original MC system (Section 6.2) splitting points. The data included samples under test with and without targets (10^6 each), and the probability of false alarm and detection were calculated. They are presented in Tables 6.7 and 6.8. The systems were trained to target a $P_{FA} = 10^{-4}$ and the targets were generated with a $SINR = 3.3dB$.

System Size	**System** P_{FA}	**Gaussian** P_{FA}	**K-Dist** P_{FA}	**Pareto** P_{FA}
MC	$1.59 \cdot 10^{-4}$	$8.67 \cdot 10^{-5}$	$1.54 \cdot 10^{-4}$	$2.01 \cdot 10^{-4}$
K3P3	$2.03 \cdot 10^{-4}$	$1.05 \cdot 10^{-4}$	$2.3 \cdot 10^{-4}$	$2.25 \cdot 10^{-4}$
K5P5	$1.72 \cdot 10^{-4}$	$8.8 \cdot 10^{-5}$	$1.88 \cdot 10^{-4}$	$1.99 \cdot 10^{-4}$
K10P10	$1.79 \cdot 10^{-4}$	$9.0 \cdot 10^{-5}$	$2.25 \cdot 10^{-4}$	$1.78 \cdot 10^{-4}$

Table 6.7: P_{FA} for Dynamic Sizing Configurations

These results represent a single trained system for the K3P3 and the K10P10 configurations, five instances of the MC system, and ten instances of the K5P5 system. Additional iterations were trained and tested to explore the robustness of the process. Both the MC and the K5P5 systems had roughly half of the systems produce a $P_{FA} \approx 10^{-3}$ (MC: 2 out of 5 and K5P5: 5 out of 10). Upon further investigation it was found that the discriminator ANNs converged in a local loss region and did not

System Size	System P_D	Gaussian P_D	K-Dist P_D	Pareto P_D
MC	0.0052	0.0063	0.0044	0.0053
K3P3	0.0063	0.0070	0.0060	0.0062
K5P5	0.0055	0.0063	0.0050	0.0055
K10P10	0.0056	0.0076	0.0051	0.0051

Table 6.8: P_D for Dynamic Sizing Configurations SINR = 3.3dB

converge completely. This is a common issue with machine learning systems and can be mitigated by tuning the stopping parameters (max epoch count, patience, stopping delta) or by training multiple discriminator ANNs in parallel and selecting the optimal version based off of confusion matrix results. The values presented in tables 6.7 and 6.8 are the average of the fully converged runs.

The values for P_{FA} were consistently over the target P_{FA} by a factor of 1.5 to 2.5. This is not ideal given the target $P_{FA} = 10^{-4}$ but the original system was optimized and over 20 variations of models where trained and analyzed before the final models were selected. Additionally, the threshold support file used in the dynamic system shown in figure 6.23 was very noisy. This was due to the file being generated using the Algorithm 1 but with a stopping margin of 5% and PFA calculation data size of $20/P_{fa,target}$. Parameter optimization, robustness testing during training, and higher resolution support files should improve performance and will be explored.

In order to explore the results of the splitting algorithm in section 6.4.1, we present P_{FA}, and P_D for all tested systems in table 6.9.

These regions show the ϵ based selection and the δ compression of regions in the lower shape region due to the greater difference in D_G. Even though the P_{FA} for the different system configurations are all beyond the target, they are similar in value and should improve with tuning and robustness. This similarity in P_{FA} and P_D is a positive outcome for the goal of the system to be dynamic and self-assembling. The

System Size	**K-Dist Region (a)**	**K-dist P_{FA} (10^{-4})**	**K-dist P_D (10^{-3})**	**Pareto Region (a)**	**Pareto P_{FA} (10^{-4})**	**Pareto P_D (10^{-3})**
MC	1.00-3.50	1.69	3.1	2.00-3.50	2.0	4.4
	3.50-10.0	1.37	6.1	3.50-6.50	1.95	5.7
				6.50-10.5	2.11	6.7
K3P3	1.00-1.29	2.69	4.5	2.00-2.50	3.06	6.7
	1.29-1.99	3.33	5.5	2.50-3.44	2.07	6.0
	1.99-10.0	1.93	6.4	3.44-10.5	2.05	6.1
K5P5	1.00-1.17	2.14	3.1	2.00-2.30	2.03	5.5
	1.17-1.39	2.40	3.8	2.30-2.64	2.11	4.8
	1.39-1.77	2.27	4.1	2.64-3.17	2.07	5.0
	1.77-2.61	2.15	5.0	3.17-4.29	2.05	5.3
	2.61-10.0	1.80	5.7	4.29-10.5	1.94	5.9
K10P10	1.00-1.08	3.66	4.3	2.00-2.14	3.21	7.1
	1.08-1.17	4.29	3.9	2.14-2.30	0.0	6.2
	1.17-1.26	5.43	5.1	2.30-2.44	5.03	4.3
	1.26-1.39	2.69	4.3	2.44-2.64	3.18	5.3
	1.39-1.54	3.10	4.9	2.64-2.89	2.20	4.4
	1.54-1.77	2.43	4.5	2.89-3.17	2.70	3.8
	1.77-2.09	3.53	5.4	3.17-3.62	1.74	4.3
	2.09-2.61	2.26	5.1	3.62-4.29	1.25	4.1
	2.61-3.76	2.13	4.1	4.29-5.65	1.38	4.4
	3.76-10.0	1.30	5.7	5.65-10.5	1.85	5.8

Table 6.9: P_{FA} and P_D for K and Pareto Sub-regions

application based improvements mean that this dynamic MC system can be generated, trained, and tested for different applications with different distribution expectations. Additionally, it builds a modular structure that can incorporate new expected distributions, different adaptive detectors, and inverse covariance matrix estimation methods. The architecture is modular and adaptable as a software architecture and is now self-assembling.

6.6 Conclusions

In this chapter I have described my ideation, investigation, and contribution to a novel meta-cognitive (MC) adaptive detection algorithm. This novel system combines traditional adaptive detection methods with AI/ML to operate with CFAR like performance across multiple non-Gaussian clutter distributions. A traditional adaptive detector can be tunes to perform well for different distributions and distribution regions. But If the interference distribution does not match the assumptions used to tune the detector then performance degrades. My contribution to this work falls into three main regions: distribution analysis, region classification, and a dynamically scalable application of the MC detector. This effort was instrumental in developing a novel detector that can identify the distribution region and perform binary hypothesis detection with CFAR behavior.

The first contribution to this work was my understanding of SIRV distributions and methods to measure differences in distributions. In addition to the Gaussian distribution; I explored the principles and characteristics of SIRV distributions (Pareto, K-distribution, etc.). I utilized the Kullback-Leiber and Jensen-Shannon divergence methods to analyze these distributions and define distribution regions. I used these distribution regions to design, train, and test a deep neural network classifier. In order to make the classifier train I also analyzed the complex IQ radar data and devised a pre-processing step to translate the interference information into down sampled, uni-

formly spaced, ordered statistics of the data. This classifier was able to identify the distribution and its region so that a set of finely tuned regression deep neural networks could determine the exact threshold for the interference information. The remaining components of the MC detector where designed by my colleagues but I contributed to the integration of the systems with the distribution and classifier novelty.

The results for this effort showed that the MC detector can maintain a CFAR performance across three distributions (Gaussian, Pareto, and K-Distribution). SIRV are characterized by the shape a and scale b parameters with shape dominating changes in the probability density function. Therefore the shape parameter was used as the dependent variable to determine distribution regions. The MC detector was trained with and tested for a sweep of shape parameter for K-distribution and Pareto. But also showed generalized performance for the log-normal distribution which it was not trained with. Testing showed that the response of the MC detector worked for different target P_{fa} and input power settings. These results showed that the system was inherently scalable to other distribution as long as the training data covered distribution regions with enough similarity to the new distribution. A key feature of the MC detector is that it is not necessary to identify the distribution, shape, and scale but just to classify the distribution region. These distributions have overlapping PDFs and will result in CFAR detection if properly categorized. This work was compared to other ML based detection methods and showed its resilience to changes in data as well as accuracy for CFAR behavior.

I provided the sole novelty to expand on this initial design by addressing a practical application challenge. A current challenge with advanced techniques and especially AI/ML is that they are inaccessible for adoption to existing systems unless an expert customizes the tool for an application. This dynamic scaling effort was motivated to create a self forming and automatic implementation of this MC detector. This work developed a proof using the KL-divergence to determine n distribution regions across

multiple distribution. Then I built an automatic application that could analyze IQ data then build and train all of the ML models to create a custom MC detector given the data and number of regions desired. This work provides novelty for the application and fielding of this detector as opposed to the basic research of creating the detector.

There are multiple areas of potential future work to support both the MC detector and the dynamic application. The MC detection algorithm should be tested on mixed clutter data sets. I hypothesize that the distribution region implementation will generalize well to mixed clutter as the resulting PDF and covariance matrix for mixed clutter should fall into another clutter region. Ideally if the clutter covariance matrix and threshold are accurate then the detection algorithm does not need an exact classification of the clutter distribution. Regardless, this data scenario should be tested. Additionally the MC detection algorithm is ready to be tested on high-fidelity data from tools like RFview and implemented on live systems. The current results are promising and the structure theoretically generalizes well, but testing is required to confirm. Effort was reference in this chapter regarding continued work with generative ML to generate the clutter covariance matrix with reduced sample support [129]. This work is vital to augment this capability and will be continued.

The dynamic sizing work was successful as an initial exploration, but suffered from resolution issues. Future work will focus on determining the appropriate data volume and training required to overcome the resolution issues. The current implementation needs the desired number of distribution regions as an input. Ideally this should be related to performance. Future work would focus on deriving a connection between the number a regions and an allowable δ error regarding the CFAR performance of the auto generated MC detector.

The contributions in this work show the value of a multi-objective approach like meta-cognition where ML augments traditional RSP methods for flexible performance. Additionally, the dynamic self forming system based on statistical divergence

provides a path from theory to application in real systems. This enables a more generalizable and scalable version of the already performant detection algorithm. The domain of radar detection is greatly benefited by these methods. The use of finely tuned AI/ML models provides flexibility and increases confidence in usage of black-box AI/ML. My work shows novelty with the determination of distribution regions, the MC adaptive detector with CFAR performance, and the scalable self-forming implementation. This provides a method to field the detector to different radar systems that have personal requirements and interference expectations.

The work presented in this chapter built and tested an adaptive detection system with CFAR performance with non-Gaussian clutter. It is core to my research goal to explore AI/ML applications with multi-agent/objective structures for radar tasks. The meta-cognitive structure provides a multi-agent solution with AI/ML flexibility. This research has greatly improved my understanding of adaptive detection, multi-agent hierarchical structures, and distribution characteristic and divergence measures. A core interest for this work was the need to build scalable systems that can generalize to domain challenges. These contributions were peer reviewed and evaluated by the community through a journal paper [23].

Chapter 7

Conclusions and Future Work

7.1 Summary of Work

This dissertation explored the challenge of multi-agent collaboration to accomplish radar tasks in C2 scenarios. The field of collaborative game theory presents mechanism to maximize the collaborative utility of coordinating agents instead of agents maximizing their personal utility. The future of C2 scenarios will have ever increasing complexity and interacting agents. This driving challenge led to the work in the dissertation that explores both novel methods to address radar task during C2 scenarios and to coordinate multi-agent performance. As such, the methods need to perform well in the radar space, coordinate with multiple agents, and need to scale well as agent swarms increase.

In Chapter 1, the challenges of multi-agent coordination to accomplish radar tasks was introduced. The current climate of C2 operations was explained and the need for scalable collaborative systems was presented. This chapter concluded with a summary of the contributions in this work.

Chapter 2 provided an overview of radar concepts. The initial background laid the foundation for the radar simulations used in this work. It was then followed by a discussion of detection methods, SIRV distributions, statistical divergence methods,

and RRM tasks. The RRM tasks of track confirmation and search were expanded on as they are focal topics for two of the chapters.

An introduction to ML basics was presented in chapter 3. ML methods are used for both the MC detection algorithm and the track confirmation work in the contributing chapters. This segued into a thorough discussion of reinforcement learning, which is a major tool explored in this work. This is followed by a short discussion of the C2 domain. This chapter concludes with a discussion of game theory as a primary discipline for multi-agent collaboration.

Chapter 4 presented the first and most explored topic of multi-agent track confirmation. This work presents a difficult C2 scenario with an evasive target and multiple radar platforms with limited resources. Additional realistic difficulty is added with a clutter ridge with greatly reduced P_D as the relative velocity of targets approach zero and the presence of false alarms. I explain my reasoning and design for three low information tracking methods including probabilistic heuristics and RL. This scenario is tested with two platforms with a focus on collaborative methods for coordinated success. Collaborative game theory is leveraged to coordinate agents for improved performance. This work was tested using three increasing fidelity simulations and showed the value of approaching radar tasks collaboratively.

Continuing with research into scalable multi-agent radar tasks, Chapter 5 described the design and implementation of a collaborative utility representation. This representation combined platform location and estimated P_D and C2 knowledge of the surveillance space to establish a baseline utility. The information gained from illuminations was recorded and subtracted from the utility layers. A dynamic beta distribution method was used to represent the growth of uncertainty in prior illuminations over time. This method cause agents to revisit areas of importance more often then others and balance exploration of new space and exploitation of high priority locations. The representation could share information between agents to further increase

collaborative utility. Additionally, the representation and methods are tested with the platforms in motion to further show the value of this real-time search method.

In chapter 6, I present a meta-cognitive adaptive detector algorithm and my contributions to probability distribution analysis, an ML classifier, and a dynamic self-forming application of the detector. This MC detector is a multi-level solution that utilizes the Gaussian GLRT with two levels of ML deep neural networks to maintain CFAR-like performance across multiple non-gaussian interference sources. I present our process for clutter modeling, processing of complex data into forms usable by ML, the design and training of the distribution region classifier and the finely tuned threshold selectors, and the extensive testing showcasing its CFAR-like performance. As a follow-on to this work, this chapter presents a dynamic self-forming application that uses the KL-divergence to determine appropriate distribution regions and programmatically builds the MC structure and automatically trains and tests the ML components. The result is a customized version of this MC detector based off data sources and a potential user's needs.

Finally, Appendix A is included which details the process for the radar simulation work that was used to test the RRM task methods in this dissertation. Considering the focus of this dissertation included multi-agent scenarios focused on RRM tasks, it was necessary for a simulation to represent a C2 challenge but to also have a realistic radar simulation for the illumination and detection process connected to radar actions. The work in this appendix is not particularly novel but is necessary to establish credibility that the methods tested here could scale to real radar systems. The appendix represents three different increasing fidelity simulation methods. The first estimates P_D based off the radar range equation for a simulated phased array radar. The second is a medium-fidelity system that generates support samples with clutter and noise for each range and velocity gate and then performs detection with the GLRT. If a target would be present within the beam width and in gate then a steering vector is used to include the

target information in the sample under test. The clutter profile used includes a clutter ridge where probably detection drops greatly as relative velocity approaches zero. Additionally the GLRT detection method includes false alarms which have impacts on the work in this dissertation. The final simulation generates a pulse-Doppler transmit waveform; simulates noise, clutter, and targets of interest; and then performs range-Doppler processing followed by the cell averaging CFAR detection method. Chapters 4 and 5 heavily use these simulators and the results are presented with the discussion of the radar simulation's impact on method performance.

7.2 Conclusions

This dissertation presents a design approach focused on developing scalable multi-agent solutions for radar tasks. The driving challenge for this exploration is founded in C2 scenarios where individual radar resources are limited but an autonomous swarm of platforms is available. As part of this work I provide algorithms for use in the radar tasks of track confirmation, optimal surveillance, and adaptive detection was scalable coordination for multiple agents. All of these radar tasks have established methods but are mostly focused on maximizing the performance of singular radars. This work explores the challenges and required innovation for scalable multi-agent scenarios. Additionally, some of these tasks are trivialized with the assumption that radar resources are abundant compared to the evasiveness of targets of interest. This work looks at the C2 challenges where those assumptions don't hold and individual radar platforms have limited resources in contested scenarios dealing with highly evasive targets. These extreme C2 challenges are mitigated with the availability of multiple autonomous platforms that can and should collaborate.

ML models are used to provide flexibility alongside high accuracy RSP methods to improve performance over a wide range of scenarios. RL is explored to provide a tracking strategy in a low information scenario with highly evasive targets. Game

Theory is used as the primary mechanism for multi-agent collaboration. Finally, all of these methods are designed with consideration for their scalability into larger swarms as well as modular design principles to enable human-in-the-loop augmentation by future users. The resulting methods and collaborative augmentation show improved performance within these C2 challenges.

In order to address the C2 challenges, a simulator that coordinated the environment and multiple agents was designed and implemented. In order to increase confidence in the methods proposed and tested, a realistic radar detection simulation was implemented to map agent actions to realistic measurements from the environment. Over the years of this research the radar detection methods increased in fidelity. These simulations began with a probability of detection estimation using the radar range equation, following with a simulated adapter detection method, and concluding with a full pulsed-Doppler simulation and range-Doppler processing. The scenario and radar measurement simulations were a necessary design step in order to efficiently represent realistic C2 challenges and test the viability of agent actions and collaboration methods.

The first contributions address the C2 challenge of track confirmation for a highly evasive target while having limited radar resources. Traditionally, when the radar system has abundant resources, track confirmation is simply a re-illumination of a previous detection to confirm that it is not a false alarm. When resources are limited and a target is evasive, this strategy will no longer illuminate the previous detection and a confirmation cannot be made. This scenario is similar to a low-information tracking challenge. This work explored two probabilistic heuristic methods using a naive Gaussian and a Gaussian augmented velocity predictor. These methods were compared with RL trained agents which made track action selections in a highly uncertain state. Tracking methods alone struggle to perform track confirmation in this extreme scenario, but the introduction of collaborative GT enabled two track con-

firmation agents to collaborate and greatly increase their probability of successfully establishing a track. The novel leader-follower consensus game theory method improved scenario success for all methods by a factor of $\approx 20\%$ and the Gaussian velocity tracker in combination with GT collaboration improved success by $\approx 25\%$. This work not only showcase the viability a multi-agent collaboration but proposed a consensus based game theoretic method to optimize area coverage for multiple agents. This scenario included a relatively high number of false alarms which added additional difficulty for the collaborative agents and showcase the necessity of generalized methods with low assumptions when dealing with scenarios of high uncertainty.

The second C2 challenge focused on the explore and exploit problem of surveillance of important space with variable priority locations. A common assumption in the RRM search task is that all space is of equal priority. The C2 scenario in this work provides two challenges to this assumption. First, the surveillance agents have intelligence information regarding geographic locations of higher priority for surveillance. The second is that multiple platforms are surveying the area together but have limited resources; therefore, they must maximize their collaborative utility as opposed to their personal utility. The novel contribution presented here is a utility-based framework to represent known information that would prioritize locations of surveillance. First the location and estimated probability of detection for each platform is used to bias their actions toward illuminations with higher odds of target detection. Second the geographical C2 information is included as areas of higher priority for surveillance. In this domain of knowledge aided search, their emerges are challenging exploration and exploitation problem where an agent needs to explore new space but also should revisit high priority locations as uncertainty increases. This work presents a novel beta-distribution method which encourages revisiting high priority locations sooner as uncertainty in those areas can be tolerated less. Additionally, this utility representation enables collaborative platforms to share their action histories and optimize their

collaborative utility. This method scales linearly as platform count increases. Finally this method and the comparison methods are tested with stationary platforms as well as platforms in motion. The utility search representation improved the number of targets found by over 8% and found them an average of $300m$ before the other methods. When the platforms moved, this improvement was increased to 10% and $400m$ showing its optimal response in scenarios with motion.

The final work, provided in this dissertation, is a MC adaptive detection algorithm. This algorithm was designed as a two-stage ML hierarchy that classifies the distribution region of the interference in the data and then passes that to a finely tuned threshold setting model to be used with existing radar adapter detectors (GLRT). The higher level model is a ML model that is trained to classify distribution regions across SIRV distributions. The next level uses the same radar interference information to determine a threshold for a binary hypothesis test detection determination using the Gaussian GLRT. This method was able to maintain CFAR-like performance across a wide range of distributions. Additionally, there was no observable loss when operating in the default Gaussian case. This novel MC detector had CFAR performance with a P_{fa} error compared to the target P_{fa} of $\pm 0.1dB$ across three clutter distribution while the comparisons suffered by ± 5 to $10dB$ error. A primary limiting factor for this MC detector is the need for expert determination of distribution regions and the intensive design and training of ML models. To address this problem, this work presents a self-forming dynamic application which utilizes input data and desired parameters from the user to automatically build, train, and test a custom version of the MC detector.

As the C2 domain continues to grow in complexity and scale of operations, the exploration of multi-agent coordination and scalable systems is paramount to success. The approach and methods in this work provide contributions to the fields of command and control, the edge case scenarios of multiple RRM tasks, coordination of multi-agent systems to maximize collaborative utility, and scalable systems with a modular

design philosophy. Addressing challenges with increasing complexity and a need for coordination helps bridge the gap between methods showing promise by optimizing individual performance and the reality of integrating those systems into large-scale scenarios. These multi-agent methods are also applicable beyond the C2 and RRM tasks given distributed autonomy is a growing field as the volume of autonomous systems increases. Additional research into the challenges of expanding single agent methods into multi-agent scenarios has increasing importance as autonomous systems become more integrated across technology domains.

7.3 Future Work

C2 is a domain with many challenges which utilizes radar as its primary method to observe and orient agents in the environment. Additionally, the current direction of autonomous systems needs to incorporate multi-agent collaboration and scalability to maintain high agility performance. Advancements in ML and RL show great promise to support both the fields of RSP and coordinated autonomy. Considerable work remains to be done to address the challenges in the C2 domain and to explore the viability of different RRM task and multi-agent collaboration methods. The following will expand on a few major areas of future research and work to build upon this research or were revealed during my exploratory process.

The first major area of future work would be improvements in the C2 scenario simulator. Given the complexity of establishing the scenarios, this initial work had only two platforms for the track confirmation work and the maximum of four platforms for the utility search work. A major expansion area is the scaling of these scenarios. Ideally this would look at tens if not hundreds of agents coordinating tasks together. In order to accomplish this scaling, there are a few changes that would be needed for some of the methods. For the game theoretic collaboration used in the track confirmation, there were only two platforms and they either collaborated or they did not. If

you had a scenario with 20 agents you would not want all 20 of them to work together to confirm the track of a single target. The swarm should have different sub-tasks. A major area a future work would be implementations of locality and incorporating a controller or scheduling mechanism. This dissertation is building toward multi-agent collaboration for full RRM scheduling but the scope had to be paired down. The GT methods of confidence and negotiation is a concept that would work well for a collaborative scheduler. A confident structure has each agent identify the importance of the task it wants to do then the agents can negotiate whether or not their personal tasks hold more value than entering into a collaborative action with another agent. It would be useful to explore both the coordinated scheduling of multiple agents, the impact of locality regarding how many agents should form coalitions, and the use of confidence-negotiation mechanisms to coordinate that collaboration.

The track confirmation work explored the viability of RL models for multi-agent scenarios. Although multiple architectures and models were explored, the RL was trained as a singular agent and approached an upper limit on performance. This decision was made to accommodate modularity concerns and enable a human-in-the-loop option. But in order to improve performance, an area of future work is more advanced reinforcement learning methods. Among these include training agents within a dynamic environment including multiple other agents and using the game theoretic concept of price of anarchy to develop a custom reward function. Price of anarchy reflects the impact of selfish actions upon the success of the multi-agent coalition and fits well to RL reward design. Additionally, the field of MARL has shown great promise in the robotics domain. MARL was avoided because it integrates all of the agents into a monolithic deep learning structure, but if the performance gains are great enough it could be worth the loss in modularity. Finally in the field of GT, only the consensus algorithm and utility maximization were fully explored. The concepts of price of anarchy, regret management, and confidence-bargaining methods are all promising ways

to coordinate multi-agent systems. The focus would be to identify the right type of collaborative GT for different scenarios.

The work with utility search presented a simple objective point representation of prioritized areas of surveillance. Additional work should be done in mapping C2 scenarios to a utility representation. This could include creating areas that are expected to generate targets of interest as well as represent changing strategic priorities in addition to geographic priorities. An additional aspect of search that I am currently researching is the track recovery step. This is the situation where a track has been lost, but a radar system would like to search locally to try and recover the track. I've already been working on a multivariate Gaussian expansion based on the Kalman filter family of trackers to represent the increase in uncertainty and the prioritized search regions based on time since lost track.

This work tested the impacts of platform motion upon both the representation and the mechanisms for search action selection. But the testing criteria only had platforms moving in a counter-clockwise direction and at a constant velocity. Variable motion patterns would be beneficial research as they more accurately reflect real scenarios. It would be valuable to explore the impact of platforms in close proximity and the importance of utility maximization. Both the track confirmation and the utility search methods would benefit from being combined with an intelligent scheduler. This could explore the prioritization between task agents as well as how each agent would respond the variable time since it was last scheduled.

There are multiple areas a viable future work when it comes to the MC detector and the self-forming dynamic application. Even though the MC detector was tested for a sweep of clutter distributions each individual test had homogeneous clutter in the support samples. The distribution region implementation should generalize well even to mixed clutter scenarios assuming that the resulting mixed PDF would fall into another clutter region. But this assumption needs to be tested with mixed clutter

data. Extensive work has already been done by my colleague to use generative ML to generate a clutter covariance matrix from reduced sample support [129]. Continued work in that field is planned and currently underway. Finally, the MC detector should be tested on high-fidelity or live capture data.

The dynamic-sizing self-forming application was successful but suffered from issues and resolution. This was due to an attempt to minimize training time with less data and label granularity, but it resulted in performance reduction to maintain CFAR-like results. Testing and exploration needs to be done to determine the relationship between CFAR-like performance, data volume, and training time. Additionally, the number of distribution regions was made as a user input but it is unlikely that a user would know how many regions they need for optimal performance. An area of future work that is already underway is determining the link between the number of distribution regions and the detector CFAR performance. Ideally, the user should provide some allowable error δ and then the system should auto-generate the appropriate number of distribution regions to stay within that error.

Appendix A

Search and Evade Game Simulation

A.1 Introduction

The work in this dissertation explored collaborative multi-agent methods for radar tasks. In order to test the effectiveness of both individual agents and the game theoretic methods it was necessary to build a simulated environment as opposed to traditional data. The simulation was a combination of game theoretic methods to devise a relevant command and control (C2) challenge while maintaining a realistic simulation for radar platforms. These scenarios and the backend radar simulation played a vital role in three publications as well as additional medium-fidelity testing for completeness of this dissertation. As such there were creations of the C2 scenario and three iterations of the radar simulation with increasing fidelity. This simulation was an important testing ground for the contributions of this dissertation and therefore is not included in the dissertation chapters. But it is important that both the scenarios meet the C2 needs as well as the radar simulation fidelity being appropriate in order to trust the results from the contributions. The C2 scenarios played an integral part in the methods used in the core contribution chapters but the radar simulations are methods presented in literature and not particularly novel. Therefore they are included in this appendix and explained in detail in order to establish the credibility of the results in this dissertation.

The following sections will describe the three different radar simulation methods used in this work. The first is a probability of detection only method that utilizes signal to noise ratio based off the radar range equation [20]. The second simulation utilizes the GLRT adaptive detector and generates support samples with realistic clutter for multiple range and velocity gates. If a target is present, then a steering vector is used with a calculated SNR to add the target signal to the gate [21, 22]. The final medium fidelity simulation method generates a LFM transmit signal then formulates the return signal with noise, clutter, and potential targets to a simulated CPI of IQ data. If a target is present its information is added to the fast-time slow-time IQ data. Finally this data undergoes range-Doppler processing and a cell averaging (CA) CFAR detection method is utilized to determine detections in individual range and velocity bins.

A.2 Original Range Equation Simulation

The first radar simulation is of a digital array radar whose parameters along with the range of a potential target are used to calculate a SNR value. Given the SNR, a probability of detection was calculated and compared with a random variable to determine detections. If a target was detected by a platform using this method, then the range and velocity information was given to the platform agent for action selection and potential collaboration. The simulation did not consider false alarms or simulate real noise or clutter, but simply assumed a constant noise power. The following is a detailed explanation of how the simulation was implemented. The equations and parameters here were programmed in Python and used within a gym environment for the C2 scenario. This version of the simulation was featured in the paper "Collaborative game theory and reinforcement learning improvements for radar tracking," at the 2023 International Radar Convention [20].

Each radar simulation was simulated as a digital array radar with parameters presented in Table A.1.

Parameter	**Value**	**Parameter**	**Value**
Array elements	16×16	Center frequency	3 GHz
Sample rate	100 Mhz	Bandwidth	30 MHz
Pulse width	1 ms	Pulse repetition frequency	5 khz
Number of pulses	8	$3dB$ Beamwidth	$\theta_a = 2^o$

Table A.1: Detector Simulation Parameters

This simulator was based on a simulation written by my colleague, Shane Flandermeyer. He provided a simulator that generated beams based on the above parameters. The simulation would load in a target location and then calculate the corresponding radar return power if the target was present. I modified the radar simulator to allow for beamsplitting/spoiling.

Using this simulation I designed the radar platforms with a dynamic action space. Each radar was capable of transmitting one, two, or four $\theta_a = 2^o$ beams. I made this design decision in order create a relatively consistent area of detection for the platforms as the looked a close locations and far locations. This created a trade-off for the radar between range and angular coverage. This meant that if the radar surveyed close regions it could spoil the beam for more effective beam width and if searching a far space it would have a full power focused pencil beam. Table A.2 show the centering of the beams when spoiled and the designed ranges where power lowered enough to justify the beam spoiling. Figure 4.1 is an example of this spoiling as Radar 1 is using two beams and Radar 2 is using one beam.

Although the simulation for the radar utilized a power and azimuth angle input to direct beam energy I chose to simplify the actions for the agents by having agents select a coordinate to focus energy. Then as a post processing step I calculated the azimuth and beam splitting mode/transmit power needed to that a target at that location with a radar cross section of $10m^2$ would be detected with a probability of 95%. A

probability of detection (P_d) was calculated for the associated detector based on the signal-to-noise ratio (SNR) of the return signal. To simplify the game and more effectively evaluate the impact of introducing different GT strategies, the P_d for any target falling outside of a detector's $3dB$ beamwidth was set to 0. To develop a baseline game representation of a search and evade challenge, false alarms were not considered. To simplify the selection of the number of beams, the probability of detection was modeled as a decaying exponential

$$P_d = 1 - \alpha * e^{\beta * SNR} \tag{A.1}$$

For this initial work, $\alpha = -4.69$ and $\beta = -.536$ were selected to cause detection to fall below 1% at $\approx 110\%$ of the range at the $P_d = 95\%$ action point and drive the beam splitting behavior according to Table A.2

This probability of detection equation is non-standard and designed to target a game with interesting win rates. In order to set a baseline and test the effectiveness of GT methods, games need to be neither unwinnable or too easy for the agents or the GT method will not present impacts outside of the variance within a game representation. My extended simulation in section A.3 implements probabilities of false alarm and utilizes a GLRT and clutter processing to determine detections and false alarms. These changes result in a simulated game that does not perfectly reflect reality, but represents realistic radar beam geometry and a SNR decaying probability of detection.

A final modification to action selection is the capability to split beams when a lower range is selected. This was utilized to be both representative of a radar system as well as to not bias actions toward longer range action selection benefiting from beam spreading. The derived boresight azimuth action is θ_a, the $3dB$ beam width is θ_{3dB}, and the power for the split beam is the same that would attain a $P_d = .95$ for the original θ_a

Beams	Azimuth targets	Range(from detector) to split
Single	θ_a	$R > 10400m$
Two	$\theta_a + \frac{\theta_{3dB}}{2}$, $\theta_a - \frac{\theta_{3dB}}{2}$	$6400m > R > 10400m$
Four	$\theta_a + \frac{3\theta_{3dB}}{2}$, $\theta_a + \frac{\theta_{3dB}}{2}$ $\theta_a - \frac{\theta_{3dB}}{2}$, $\theta_a - \frac{3\theta_{3dB}}{2}$	$R < 6400m$

Table A.2: Condition for Radar Beam Splitting

A.3 GLRT Dynamic Simulation

This simulation update implemented a realistic noise and clutter profile model as the backend for the detectors. This added additional challenge to the players as they had to not only deal with the issue of degrading probability of detection P_D but also with a chance of false alarm detections at a constant probability of false alarm P_{FA}. Each detector was simulated as a digital array radar with parameters presented in Table A.3.

Parameter	Value	Parameter	Value
Array elements	16×16	Center frequency	2 GHz
Sample rate	100 Mhz	Bandwidth	30 MHz
Pulse width	1 ms	PRF	8.5 khz
Max Power	60 W	Beamwidth	$\theta_a = 1^o$
Number of pulses	16	Sample Support	64
P_{FA}	10^{-5}		
Range Resolution	500 m	Max Range	17630 m
Velocity Resolution	40 m/s	Max Velocity	318 m/s

Table A.3: Detector Simulation Parameters

The radar is capable of transmitting one, two, or four $\theta_a = 1^o$ beams. The beams

trade-off range with angular coverage, enabling the radar to more quickly search an area at short ranges with a spoiled transmit beam, or focus a pencil beam for a long range search. The detector action was implemented by selecting Cartesian coordinates (in x and y). This Cartesian coordinate is converted to an azimuth and a range for use for the radar signal processing. This beam is split into range and velocity bins that are simulated with noise and a realistic clutter profile according to [31].

If the target falls into one of the range/velocity bins then a signal with a radar cross-section of 10 was added into the sample. Each bin was individually processed through a GLRT detector with 64 additional support samples. This means each bin has a chance of a true detection or a false alarm. To deal with the possibility of multiple detection an algorithmic association method was implemented to choose the best detection based on the previous detection location. This associated detection is then given to the players to inform their next action.

A.3.1 Beam Splitting

Beams	**Azimuth targets**	**Split range (m)**	**Cutoff range (m)**
Single	θ_a	$R > 15100$	20000
Two	$\theta_a + \frac{\theta_{3dB}}{2}$, $\theta_a - \frac{\theta_{3dB}}{2}$	$15100 > R > 12350$	15500
Four	$\theta_a + \frac{3\theta_{3dB}}{2}$, $\theta_a + \frac{\theta_{3dB}}{2}$ $\theta_a - \frac{\theta_{3dB}}{2}$, $\theta_a - \frac{3\theta_{3dB}}{2}$	$R < 12350$	14000

Table A.4: Condition for Radar Beam Splitting - GLRT

Each radar detector had a maximum power output of 60 Watts. This parameter was tuned to allow the detectors to have a good probability of detection across the entire map. As a trade off we allowed the detectors to split their beams. This was modeled by

the detector emitting multiple beams with less power. Three increments were allowed as shown in Table A.4. As the number of beams doubled, the power emitted per beam was equivalently halved, resulting in a constant total emitted power. The probability of detection was profiled for each power level to determine the splitting ranges. These splitting cutoffs were chosen such that a probability of detection greater than 90% is achieved at any look point while maximizing the width due to beam splitting. To save on computational costs the beam was not processed at ranges after the probability of detection dropped below 50%.

A.3.2 Clutter Simulation

Each radar beam was simulated individually using space-time adaptive processing methods. The number of pulses $N = 16$ and a sample support of $K = 64$ are used to simulate and process the radar signals. The length N radar return vector is modeled as

$$z = N_s + C_s + S_s \tag{A.2}$$

where C_s is the clutter component, N_s is the noise component, and S_s is the signal component. A noise Figure is calculated from the radar parameters and uses standard temperature to calculate a noise power of $\approx 1.006\dot{1}0^{-12}$. This noise power is used to create $K + 1$ Gaussian noise samples. The samples are uncorrelated random vectors of length N. This Gaussian noise was used in each bin. The beam is then split into 35 range bins and 16 velocity bins ranging from $0m$ to $17,630m$ and $-300m/s$ to $300m/s$ respectively. Only the range bins that landed within the game area were processed. This allowed us to ignore any range bin within the radar blind range since the detectors are placed at least $3000m$ from the game.

Next, the simulator loops over each range bin, and generates a clutter signal based on the clutter model shown in [31]. We used a root mean squared (RMS) clutter velocity of $\sigma = 20m/s$ with a clutter power of:

$$P_c = \frac{P_{tx} P_n 1006^2}{r^2} \tag{A.3}$$

Where P_{tx} is the transmit power of the beam, P_n is the noise power, and r is the distance of the center of the range bin from the detector. The constant in the noise power was tuned so that the clutter played an important role across the full game. It can be noted that when combining the constant with the max transmit power and range variables, the clutter power is greater than the noise power up until a range of $\approx 7800m$. These parameters were then used in Equation 2.23 from the background section 2.2.6.

$$\tilde{r}(n) = P_c e^{-8(\frac{\pi n T_p \sigma}{\lambda})^2} \tag{A.4}$$

Where T_p is the pulse repetition interval, and λ is the wavelength. The $N \times N$ Clutter Covariance matrix is then constructed, with each component being calculated with 2.23, where n is the distance from the diagonal of the matrix. This covariance matrix is then used to generate $K + 1$ correlated Gaussian random vectors of length $N = 16$.

A.3.3 Signal Detection

After generating the noise and clutter the simulator loops over each velocity bin. If the target is within the beam its velocity projection onto the direction of the beam is calculated. This is done because realistic radar can only resolve the radial velocity through the Doppler shift in its signal. It gains no information about the angular velocity of the target. It is then assigned the velocity bin that is closest to its radial projection. For empty bins, the signal component is set to zero. Otherwise, an SNR is calculated based on the loop gain, radar cross-section, range, and the angle from the beam center. Using the noise and clutter power, this is converted into the $SINR$ which is then used to generate the signal sample as follows

$$S_s = \alpha p = \sqrt{\frac{SINR}{pM^{-1}p^T}} \tag{A.5}$$

Where M is the true combined covariance matrix of the clutter and noise, and p is the temporal steering vector for the current velocity bin. The Doppler frequency is calculated with the target's true velocity and then added to only the first of the $K+1$ clutter and noise samples. This first sample is the sample under test (SUT) used in the GLRT.

The GLRT requires an estimation of the noise and clutter covariance matrix. This is what the last K samples are used for. The estimated covariance matrix, Doppler steering vector centered at the current velocity bin, and the SUT are then passed to the GLRT detector. Using the GLRT's analytical solution for Gaussian samples, a threshold is selected and compared against the test statistic to make a detection determination. This is done for each velocity bin across all the range bins. The performance of the radar simulation and detection system was tested in the game. The beams achieved the set probability of false alarm rates when tested without targets. Next, the target was placed at specific ranges and radial speeds within the beam, and the probability of detection was measured. This testing is seen in Figure A.1

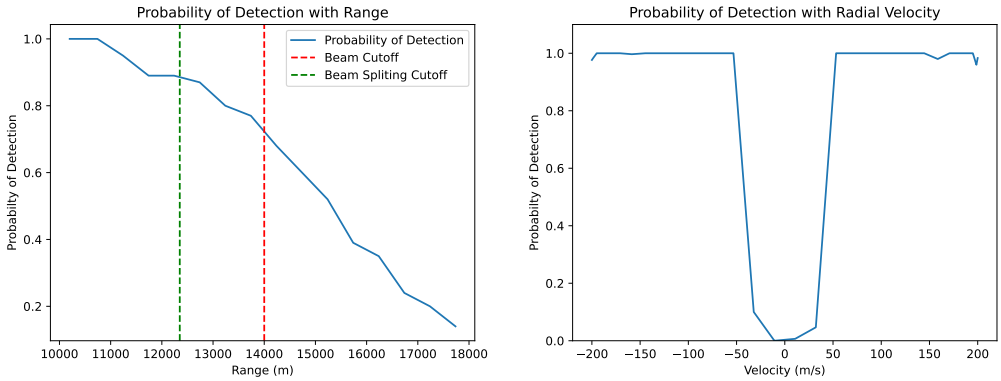


Figure A.1: Probability of Detection of a 15W Beam with Range and Velocity

The detector then collates the detections from each of its beams through an association algorithm. This association algorithm is done so the players of the game do not have to handle a dynamic list of detections. The association algorithm takes in the list of detections and attempts to find one detection that is the most reasonable to be the target based on the previous successful detection. The current heuristic used minimizes the sum of the L2 norm of the position and velocity between the detections. The logic behind this is that detections closer to the assumed last location of the target are more likely to be a correct detection than a false alarm. In addition, a detection that has a similar velocity is more likely as the target would have had to turn $180°$ to reverse its velocity.

A.4 Range-Doppler and Cell Averaging CFAR Simulation

This final higher fidelity simulation update utilized by the task confirmation and utility search methods was implemented by simulating a CPI of return signals. This CPI of return data used a simulated transmit signal and imposed realistic noise, clutter, platform motion, and potential target of interest information upon the transmit samples. This CPI of data is then range-Doppler processed using a matched filter and an FFT to generate a range-Doppler plot. I implemented a CA-CFAR algorithm in order to determine a threshold and perform the detection hypothesis test on each of the range and Doppler bins. Finally, association logic is used to summarize a list of possible detections. The motivation behind implementing this medium-fidelity simulation was to test published work with a radar simulation that better represented a true pulsed-Doppler radar acting within a C2 scenario. This implementation and the subsequent results for the track confirmation and utility search increases confidence in the proposed methods when they are implemented on a live system. The following section will explain the parameters selected for this simulation and the implementation to complete the action-observation loop for the agents with realistic radar simulation.

A.4.1 Radar Parameters

The following Table A.5 is a list of radar parameters for a digital phased array. These parameters are utilized throughout the following discussion of the CPI data simulation and the range-Doppler processing and detection.

Parameter	**Value**	**Parameter**	**Value**
Array elements	16×16	Center frequency ($F_c = \frac{1}{\lambda}$)	2 GHz
Sample Rate ($F_s = \frac{1}{T_s}$)	6 Mhz	Bandwidth (β)	3 MHz
Transmit Pulse Width (T_p)	5 μ s	PRF ($PRF = \frac{1}{PRI}$)	10 khz
Max Power (P_{tx})	40 W	TX/RX Gain (G)	100
Number of pulses (M)	128	Chirp Rate ($\gamma = \frac{\beta}{T_p}$)	$6 \cdot 10^{11}$ Hz/s
P_{FA}	10^{-8}	Range Resolution	49.97 m
Min Range	3000 m	Max Range	17000 m
Velocity Resolution	5.85 m/s	Max Velocity	$\pm$ 300 m/s
System Temp (T)	290 K	Noise Figure (F)	4 dB
Clutter Velocity (V_c)	5 m/s	Volume Reflectivity (η)	100 m^6/m^3
CA-CFAR Guard ($[G_R, G_D]$)	[1,1]	CA-CFAR Training ($[T_R, T_D]$)	[3,3]

Table A.5: Detector Simulation Parameters - Range-Doppler

Each of these parameters is used in the following equations for the formulation of the CPI data matrix, range-Doppler processing, and final detection. Additionally, the beam splitting was updated to reflect the new transmit power settings. This information is shown in Table A.6

Beams	**Azimuth targets**	**Split range (m)**	**Cutoff range (m)**
Single	θ_a	$R > 11000$	20000
Two	$\theta_a + \frac{\theta_{3dB}}{2}$, $\theta_a - \frac{\theta_{3dB}}{2}$	$9000 > R > 11000$	11000
Four	$\theta_a + \frac{3\theta_{3dB}}{2}$, $\theta_a + \frac{\theta_{3dB}}{2}$ $\theta_a - \frac{\theta_{3dB}}{2}$, $\theta_a - \frac{3\theta_{3dB}}{2}$	$R < 9000$	11000

Table A.6: Condition for Radar Beam Splitting - Range Doppler Processing

A.4.2 CPI Data Simulation

The simulation of the CPI data follows very closely with the pulsed-Doppler signal formation radar background (Section 2.2 and will reference it frequently. Although a transmit signal is characterized by both a carrier signal and a time-varying signature envelope, the processing in this section will focus on the envelope only. In this work, I utilized a triangular LFM transmit signal based on the parameters presented above. The first step to developing the return signal CPI information is to build a fast-time slow-time matrix of transmit data that will be modified based off of the noise, clutter, platform motion, and potential targets of interest. The sampling rate which along with a minimum and maximum range for range processing is used to determine the number of fast time points in the range dimension. The number of samples (M) will determine the slow Time dimension and are from $[-M/2, M/2-1]$ in order to center the samples at zero. The transmit pulse width and the sampling rate will determine the number of points ($N = \frac{T_p}{T_s}$)in the transmit signal. The transmit amplitude envelope is then convolved with a handing window (Equation (2.7)). The convolution process increases the number of points in the transmit signal. This new number of points is then used to calculate the effective bandwidth for the convolved transmit signal. Finally, the transmit amplitude window is combined with the phase information in

order to generate the final transmit signal (2.9).

$$s(t) = a(t)\exp(j2\pi(\frac{-\beta}{2}t + \frac{\beta}{2T_p}t^2) + \theta_0)$$

An FFT and FFT shift is used to attain a frequency domain version of the transmit signal. To support frequency domain processing a frequency variable ranging from $[-PRF/2, PRF/2]$ of the same size as the fast time dimension is generated.

The following will explain the process of embedding a potential target of interest, adding thermal GWN, and including clutter in a simulated return signal given the transmit signal. The simulation will check the current illumination location against all of the target positions in order to determine if the target is within the 3dB beam width of the radar illumination. If the target of interest is present then the first step is to apply a time-shift in the frequency domain in order to encode the range information in the return signal. Is it important to note that this time shift information includes the velocity information as the time (τ) is shifted across the samples this builds the entire CPI in the frequency domain for the fast-time dimension. This is followed by a Doppler-shift and appropriate alpha scaling also in the frequency domain. At this point all of the information is encoded for a target of interest. If multiple targets of interest are within the 3db beam then this process is repeated until all targets have been added to the return signal. The equations for this process are found in section 2.2.3 and included here:

$$\tau(m) = \frac{2(R - v_r \cdot m \cdot pri)}{c}$$

$$F_D = \frac{2v_r}{\lambda_0}$$

$$\alpha = \sqrt{\frac{P_r}{P_t}} = \sqrt{\frac{G^2\lambda^2\sigma}{(4\pi)^3R^4}}$$

$$y(t) = \alpha a(t - \tau(m))\exp(j\theta(t - \tau(m))\exp(-j2\pi(F_0 + F_D)\tau(m) + j2\pi F_D t)$$

The process is explained step by step in the aforementioned background section. An example of a return signal with a target of interest at range of 8400 and relative velocity of 102 m/s is shown in Figure A.2. Additionally, Figure A.2 shows what a final ranged Doppler processing would look like if only the target of Interest was present.

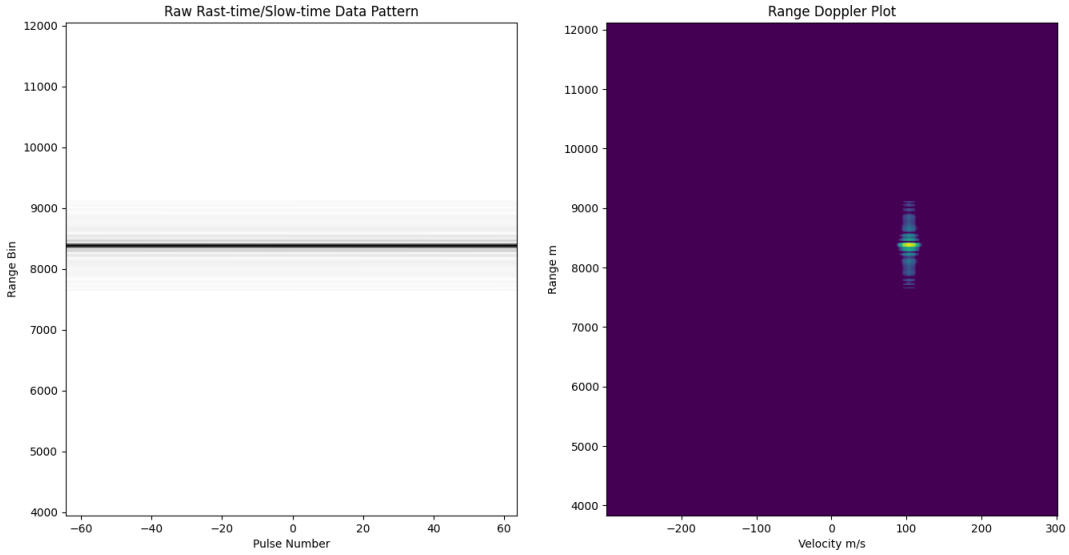


Figure A.2: CPI and Range-Doppler Processing for Target of Interest

The GWN is generated using the following equations and method based off the noise power.

$$P_n = k_b \beta F T_s$$
$$n(t,m) = \sqrt{P_n/2}\mathcal{N}[0,1](t,m) + j\mathcal{N}[0,1](t,m)$$

The simulation included code for generating clutter power for volumetric clutter as well as area clutter that is both pulse-limited and beam-limited. The work in this dissertation focused on two dimensional C2 scenarios and therefore volumetric clutter was used as the primary mechanism for determining clutter power. The equations used to determine this volumetric clutter power are explained in Section 2.2.6 and are included here:

$$V_c = \alpha \rho R^2 \theta_{az} \theta_{el}$$
$$\eta = \frac{\sigma}{V_c}$$
$$P_{vc}(R_0) = \frac{P_t G^2 \lambda^2 \eta \rho \theta_{az} \theta_{el}}{(4\pi)^3 R_0^2 L_s L_a(R_0)}$$

The clutter power was then used to generate the clutter covariance matrix for a clutter ridge implementation by [31]. The final received signal is simply the sum of the potential target of interest return, the noise, and the gaussian clutter . A final range-Doppler processing image of this noise and clutter is shown in Figure A.3.

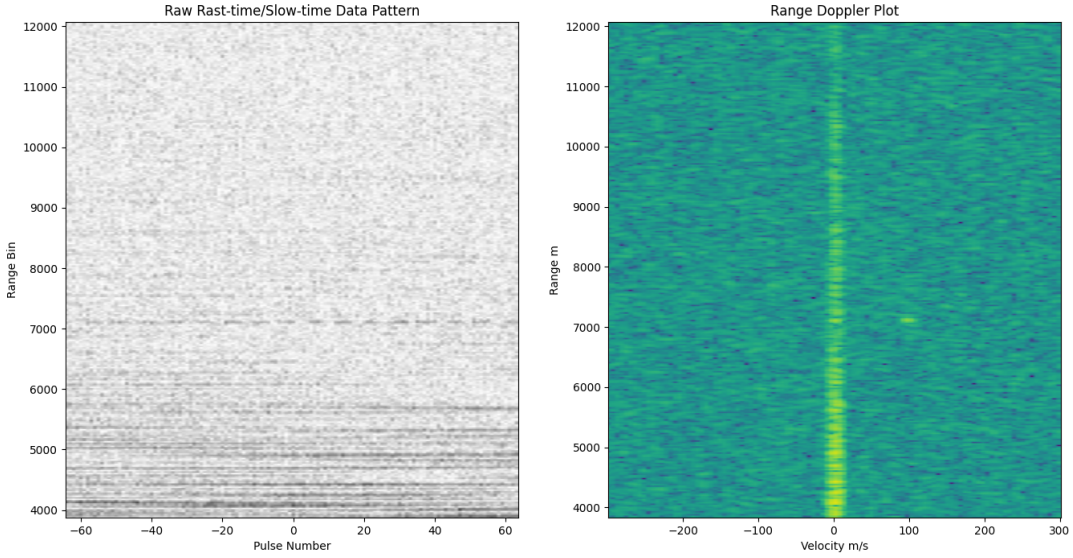


Figure A.3: CPI and Range-Doppler Processing for Target, Noise, and Clutter

The clutter and the noise makes the target less obvious in the CPI of data, but the range-Doppler processing causes the Doppler component to reveal the target of interest.

The final step that is utilized in the utility search implementation is to include a phase shift of the entire CPI data given the platform velocity this is accomplished with the following equation:

$$y(t,m)_{shift} = y(t,m) \cdot e^{(j4\pi t \frac{v_{plt,rel}}{\lambda_0})} \tag{A.6}$$

This simulated a return signal then always includes thermal GWN and gaussian clutter and could also include targets of interest and a potential shift given platform motion. This fast-time, slow-time CPI of data is used in the following section for range-Doppler processing.

A.4.3 Range-Doppler Processing and Detection

The first step of the range Doppler processing is to apply the matched filter to every range column. This process is explained in detail in Section 2.2.5. By utilizing the transmit signal, the targets of interests have their SNR maximized as they convolve with the time reversed conjugate with high corelation compared to the noise. Before Doppler processing, I chose to apply a Doppler taper using a Chebyshev window with a 100 dB side-lobe setting. This was accomplished with a python window package.

The Doppler processing is performed with an FFT that is done across the slow-time dimension following the match filtering of the CPI data. This process is explained in the radar background Section 2.2.7. Once completed the match filter and Doppler processing FFT generate the range-Doppler plot. This range-Doppler plot can then be used with a detection algorithm. The detection algorithm used in this simulator was the CA-CFAR algorithm. This detection method is explained in detail in section 2.3.2. Figure A.4 shows a full visualization of both range-Doppler plot and the results of the CA-CFAR algorithm.

A final box association step is performed in order to collapse multiple detection bins near each other into a single detection and provided as a detection list with the range and relative velocity information from the CA-CFAR detection. This process can easily include false alarms. Due to the thinness of the clutter ridge and the resolution from the radar parameters multiple false alarms are present in the clutter ridge. Therefore, the center-clutter ridge is ignored for two times the standard deviation of the clutter velocity. The final associated list of the detections, whether they are false

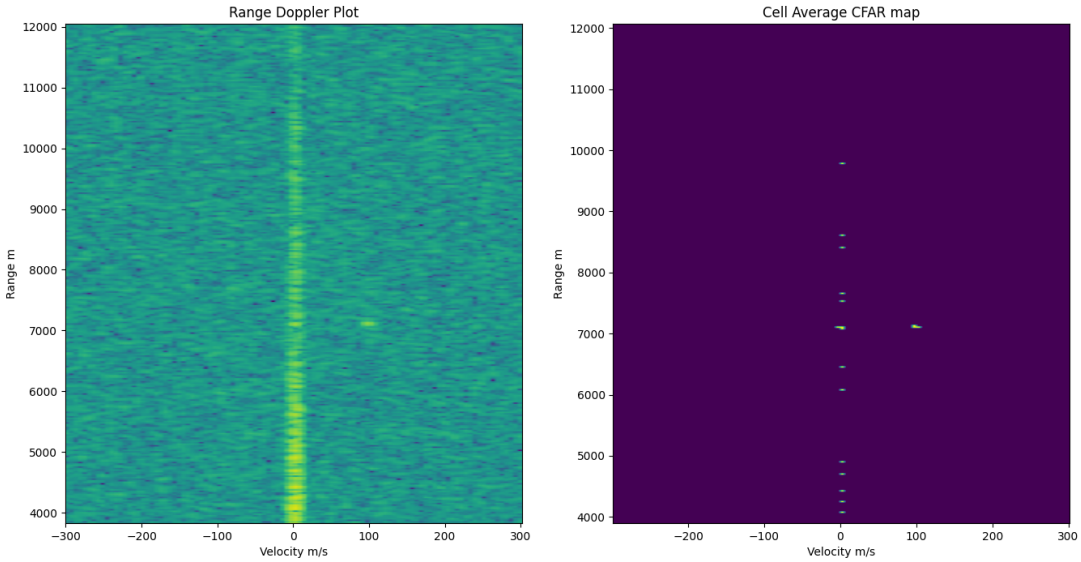


Figure A.4: Range-Doppler Plot and CA-CFAR for Target, Noise, and Clutter

alarms or targets of interest, is returned to the agents as a list of possible detections including range and relative velocity information. This information is what is used by both the scenario as well as the agents within that scenario to continue their C2 operations and to test the different methods included in this dissertation.

References

[1] C. A. McCann and R. A. Pigeau, "Clarifying the concepts of control and of command," 1999.

[2] B. Brehmer, "Command and control as design," *IEEE Aerospace and Electronic Systems Magazine*, no. 182, 2010.

[3] J. R. Boyd, "The essence of winning and losing," 1996.

[4] S. Z. Gurbuz, H. D. Griffiths, A. Charlish, M. Rangaswamy, M. S. Greco, and K. Bell, "An overview of cognitive radar: Past, present, and future," *IEEE Aerospace and Electronic Systems Magazine*, vol. 34, no. 12, pp. 6–18, 2019.

[5] J. R. Guerci, "Cognitive radar: A knowledge-aided fully adaptive approach," in *2010 IEEE Radar Conference*. IEEE, 2010, pp. 1365–1370.

[6] S. Haykin, "Cognitive radar: a way of the future," *IEEE Signal Processing Magazine*, vol. 23, no. 1, pp. 30–40, 2006.

[7] X. Song, P. Willett, S. Zhou, and P. B. Luh, "The mimo radar and jammer games," *IEEE Transactions on Signal Processing*, vol. 60, no. 2, pp. 687–699, 2012.

[8] A. Deligiannis, A. Panoui, S. Lambotharan, and J. A. Chambers, "Game-theoretic power allocation and the nash equilibrium analysis for a multistatic

mimo radar network," *IEEE Transactions on Signal Processing*, vol. 65, no. 24, pp. 6397–6408, 2017.

[9] K. Han and A. Nehorai, "Jointly optimal design for mimo radar frequency-hopping waveforms using game theory," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 2, pp. 809–820, 2016.

[10] D. J. Bachmann, R. J. Evans, and B. Moran, "Game theoretic analysis of adaptive radar jamming," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 47, no. 2, pp. 1081–1100, 2011.

[11] Z. E. Fuchs and J. Metcalf, "Equilibrium radar-target interactions in an atr scenario: A differential game," in *2018 IEEE Radar Conference (RadarConf18)*, 2018, pp. 1228–1233.

[12] B. Kang, V. Krishnamnurthy, K. Pattanayak, S. Gogineni, and M. Rangaswamy, "Smart interference signal design to a cognitive radar," in *2023 IEEE Radar Conference (RadarConf23)*, 2023, pp. 1–6.

[13] A. Deligiannis, S. Lambotharan, and J. A. Chambers, "Game theoretic analysis for mimo radars with multiple targets," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 6, pp. 2760–2774, 2016.

[14] J. R. Marden, G. Arslan, and J. S. Shamma, "Cooperative control and potential games," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 6, pp. 1393–1407, 2009.

[15] S. Haykin, "Radar vision," in *IEEE International Conference on Radar*, 1990, pp. 585–588.

[16] M. C. Wicks, "Sensors as robots," in *2006 International Waveform Diversity & Design Conference*, 2006, pp. 1–7.

[17] Z. C. Lipton, "The mythos of model interpretability," *Commun. ACM*, vol. 61, no. 10, p. 36–43, sep 2018.

[18] Y. Yang and J. Wang, "An overview of multi-agent reinforcement learning from game theoretical perspective," *https://arxiv.org/abs/2011.00583*, 2021.

[19] Z. Ning and L. Xie, "A survey on multi-agent reinforcement learning and its application," *Journal of Automation and Intelligence*, vol. 3, no. 2, pp. 73–91, 2024.

[20] G. Dolinger, A. Stringer, T. Sharp, J. Karch, J. G. Metcalf, and A. Bowersox, "Collaborative game theory and reinforcement learning improvements for radar tracking," in *2023 IEEE International Radar Conference (RADAR)*, 2023, pp. 1–6.

[21] G. Dolinger, T. Sharp, B. Lavender, A. Stringer, J. Karch, A. Bowersox, and J. Metcalf, "Multi-agent track confirmation utilising reinforcement learning and game theoretics," *IET Radar, Sonar & Navigation*, vol. 19, no. 1, p. e12694, 2025.

[22] G. Dolinger, T. Sharp, A. Stringer, J. Karch, A. Vakil, J. G. Metcalf, and A. Bowersox, "Utility representation for collaborative surveillance," in *2025 IEEE International Radar Conference (RADAR) [accepted but proceeding not yet published]*, 2025, pp. 1–6.

[23] A. Stringer, G. Dolinger, D. Hogue, L. Schley, and J. Metcalf, "A meta-cognitive approach to adaptive radar detection," in *IEEE Transactions on Aerospace and Electronic Systems (In Review)*. IEEE, 2023, pp. 1–12.

[24] M. Richards, *Fundamentals of Radar Signal Processing, Second Edition*. McGraw-Hill Education, 2014.

[25] F. A. Butt and M. Jalil, "An overview of electronic warfare in radar systems,"

in *2013 The International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE)*, 2013, pp. 213–217.

[26] S. D. Blunt and E. L. Mokole, "Overview of radar waveform diversity," *IEEE Aerospace and Electronic Systems Magazine*, vol. 31, no. 11, pp. 2–42, 2016.

[27] A. W. Doerry, "Noise and noise figure for radar receivers," Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), Tech. Rep., 2016.

[28] M. S. Greco and S. Watts, "Chapter 11 - radar clutter modeling and analysis," in *Academic Press Library in Signal Processing: Volume 2*, ser. Academic Press Library in Signal Processing, N. D. Sidiropoulos, F. Gini, R. Chellappa, and S. Theodoridis, Eds. Elsevier, 2014, vol. 2, pp. 513–594.

[29] M. Long, *Radar Reflectivity of Land and Sea*, ser. Artech House radar library. Artech House, 2001.

[30] D. Barton, *Radar System Analysis and Modeling*, ser. Artech House radar library. Artech House, 2004.

[31] A. Hassanien, B. Himed, J. Metcalf, and B. D. Rigling, "Moving target detection in spatially heterogeneous clutter using distributed mimo radar," in *2018 International Conference on Radar (RADAR)*, 2018, pp. 1–6.

[32] M. A. Richards, "Exact and approximate detection probability formulas," in *Fundamentals of Radar Signal Processing*. McGraw-Hill Education, 2018, p. 5.

[33] M. A. Richards, "Alternative forms of albersheim's equation," in *Fundamentals of Radar Signal Processing*. McGraw-Hill Education, 2014, p. 5.

[34] M. Skolnik, *Introduction to Radar Systems*, 3rd ed. McGraw Hill, 2002.

[35] J. Ward, ""space-time adaptive processing for airborne radar"," in *IEE Col-*

loquium on Space-Time Adaptive Processing (Ref. No. 1998/241), 1998, pp. 2/1–2/6.

[36] M. Rangaswamy, D. Weiner, and A. Ozturk, "Non-gaussian random vector identification using spherically invariant random processes," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 29, no. 1, pp. 111–124, 1993.

[37] M. Rangaswamy, D. Weiner, and A. Ozturk, "Computer generation of correlated non-gaussian radar clutter," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 31, no. 1, pp. 106–116, 1995.

[38] M. Rangaswamy, D. Weiner, and A. Ozturk, "Computer generation of correlated non-gaussian radar clutter," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 31, no. 1, pp. 106–116, 1995.

[39] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[40] P. Bratley, B. L. Fox, and L. E. Schrage, *A guide to simulation (2nd ed.)*. Berlin, Heidelberg: Springer-Verlag, 1987.

[41] K. J. Sangston, F. Gini, and M. S. Greco, "Coherent radar target detection in heavy-tailed compound-gaussian clutter," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, no. 1, pp. 64–77, 2012.

[42] P. Stinco, M. Greco, and F. Gini, "Adaptive detection in compound-gaussian clutter with inverse-gamma texture," in *Proceedings of 2011 IEEE CIE International Conference on Radar*, vol. 1, 2011, pp. 434–437.

[43] K. Sangston and K. Gerlach, "Coherent detection of radar targets in a non-gaussian background," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 30, no. 2, pp. 330–340, 1994.

[44] A. Balleri, A. Nehorai, and J. Wang, "Maximum likelihood estimation for

compound-gaussian clutter with inverse gamma texture," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 43, no. 2, pp. 775–779, 2007.

[45] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79 – 86, 1951.

[46] D. Endres and J. Schindelin, "A new metric for probability distributions," *IEEE Transactions on Information Theory*, vol. 49, no. 7, pp. 1858–1860, 2003.

[47] L. Bombrun and Y. Berthoumieu, "Multivariate texture retrieval using the kullback-leibler divergence between bivariate generalized gamma times an uniform distribution," in *2012 19th IEEE International Conference on Image Processing*, 2012, pp. 2413–2416.

[48] L. Bombrun, Y. Berthoumieu, N.-E. Lasmar, and G. Verdoolaege, "Multivariate texture retrieval using the geodesic distance between elliptically distributed random variables," in *2011 18th IEEE International Conference on Image Processing*, 2011, pp. 3637–3640.

[49] N.-E. Lasmar and Y. Berthoumieu, "Multivariate statistical modeling for texture analysis using wavelet transforms," in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2010, pp. 790–793.

[50] F. Robey, D. Fuhrmann, E. Kelly, and R. Nitzberg, "A cfar adaptive matched filter detector," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 28, no. 1, pp. 208–216, 1992.

[51] J. Roman, M. Rangaswamy, D. Davis, Q. Zhang, B. Himed, and J. Michels, "Parametric adaptive matched filter for airborne radar applications," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 36, no. 2, pp. 677–692, 2000.

[52] E. Conte, M. Lops, and G. Ricci, "Asymptotically optimum radar detection in

compound-gaussian clutter," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 31, no. 2, pp. 617–625, 1995.

[53] A. De Maio, "Rao test for adaptive detection in gaussian interference with unknown covariance matrix," *IEEE Transactions on Signal Processing*, vol. 55, no. 7, pp. 3577–3584, 2007.

[54] C. Richmond, "Performance of a class of adaptive detection algorithms in nonhomogeneous environments," *IEEE Transactions on Signal Processing*, vol. 48, no. 5, pp. 1248–1262, 2000.

[55] N. Pulsone and M. Zatman, "A computationally efficient two-step implementation of the glrt," *IEEE Transactions on Signal Processing*, vol. 48, no. 3, pp. 609–616, 2000.

[56] E. Kelly, "An adaptive detection algorithm," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-22, no. 2, pp. 115–127, 1986.

[57] E. Kelly, "Performance of an adaptive detection algorithm; rejection of unwanted signals," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 25, no. 2, pp. 122–133, 1989.

[58] E. Conte and G. Ricci, "Sensitivity study of glrt detection in compound-gaussian clutter," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 34, no. 1, pp. 308–316, 1998.

[59] P.-J. Chung and K. M. Wong, "A full generalized likelihood ratio test for source detection," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 2445–2448.

[60] P. Wang, H. Li, and B. Himed, "A simplified parametric glrt for stap detection," in *2009 IEEE Radar Conference*, 2009, pp. 1–5.

[61] D. P. Bruyere and N. A. Goodman, "Adaptive detection and diversity order in

multistatic radar," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 44, no. 4, pp. 1615–1623, 2008.

[62] K. J. Sohn, H. Li, and B. Himed, "Parametric glrt for multichannel adaptive signal detection," *IEEE Transactions on Signal Processing*, vol. 55, no. 11, pp. 5351–5360, 2007.

[63] U. S. A. Force, "Air force doctrine publication 3-60, targeting," November 2021, last Published: 12 Nov 21.

[64] M. Walsh, L. Menthe, E. Geist, E. Hastings, J. Kerrigan, J. Léveillé, Joshua, Margolis, and N. Martin, "Exploring the feasibility and utility of machine learning-assisted command and control: Volume 1, findings and recommendations," 2021.

[65] J. Schubert, J. Brynielsson, M. Nilsson, and P. Svenmarck, "Artificial intelligence for decision support in command and control systems," 11 2018.

[66] S. Ghosh, R. Rajkumar, J. Hansen, and J. Lehoczky, "Scalable resource allocation for multi-processor qos optimization," in *23rd International Conference on Distributed Computing Systems, 2003. Proceedings.*, 2003, pp. 174–183.

[67] J. Hansen, S. Ghosh, R. Rajkumar, and J. Lehoczky, "Resource management of highly configurable tasks," in *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.*, 2004, pp. 116–.

[68] A. Charlish, K. Woodbridge, and H. Griffiths, "Phased array radar resource management using continuous double auction," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, no. 3, pp. 2212–2224, 2015.

[69] P. Chavali and A. Nehorai, "Scheduling and power allocation in a cognitive radar network for multiple-target tracking," *IEEE Transactions on Signal Processing*, vol. 60, no. 2, pp. 715–729, 2012.

[70] J. Yan, H. Liu, B. Jiu, B. Chen, Z. Liu, and Z. Bao, "Simultaneous multibeam resource allocation scheme for multiple target tracking," *IEEE Transactions on Signal Processing*, vol. 63, no. 12, pp. 3110–3122, 2015.

[71] J. Yan, H. Liu, W. Pu, H. Liu, Z. Liu, and Z. Bao, "Joint threshold adjustment and power allocation for cognitive target tracking in asynchronous radar network," *IEEE Transactions on Signal Processing*, vol. 65, no. 12, pp. 3094–3106, 2017.

[72] J. Yan, W. Pu, S. Zhou, H. Liu, and M. S. Greco, "Optimal resource allocation for asynchronous multiple targets tracking in heterogeneous radar networks," *IEEE Transactions on Signal Processing*, vol. 68, pp. 4055–4068, 2020.

[73] W. Yi, Y. Yuan, R. Hoseinnezhad, and L. Kong, "Resource scheduling for distributed multi-target tracking in netted colocated mimo radar systems," *IEEE Transactions on Signal Processing*, vol. 68, pp. 1602–1617, 2020.

[74] P. W. Moo and Z. Ding, "Chapter 2 - overview of rrm techniques," in *Adaptive Radar Resource Management*, P. W. Moo and Z. Ding, Eds. Academic Press, 2015, pp. 9–36.

[75] T. McGee and J. Hedrick, "Guaranteed strategies to search for mobile evaders in the plane," in *2006 American Control Conference*, 2006, pp. 6 pp.–.

[76] F. Hoffmann and A. Charlish, "A resource allocation model for the radar search function," in *2014 International Radar Conference*, Oct 2014, pp. 1–6.

[77] M. Byrne, K. White, and J. Williams, "Scheduling multifunction radar for search and tracking," in *2015 18th International Conference on Information Fusion (Fusion)*, 2015, pp. 945–952.

[78] H. Chen, D. Shen, G. Chen, E. P. Blasch, and K. Pham, "Tracking evasive

objects via a search allocation game," in *Proceedings of the 2010 American Control Conference*, 2010, pp. 6981–6986.

[79] R. A. Romero and N. A. Goodman, "Adaptive beamsteering for search-and-track application with cognitive radar network," in *2011 IEEE RadarCon (RADAR)*, 2011, pp. 1091–1095.

[80] B. Carlson, E. Evans, and S. Wilson, "Search radar detection and track with the hough transform. i. system concept," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 30, no. 1, pp. 102–108, 1994.

[81] Y. Briheche, F. Barbaresco, F. Bennis, and D. Chablat, "Optimisation of radar search patterns in localised clutter and terrain masking under direction-specific scan update rates constraints," *IET Radar, Sonar & Navigation*, vol. 12, no. 12, pp. 1429–1436, 2018.

[82] J. Lu, H. Xiao, Z. Xi, and M. Zhang, "Cued search algorithm with uncertain detection performance for phased array radars," *Journal of Systems Engineering and Electronics*, vol. 24, no. 6, pp. 938–945, 2013.

[83] Y. Wan, C. Sun, H. Fu, B. Zhang, and W. Fan, "Adaptive optimization method of radar search resources based on genetic algorithm," in *2022 3rd China International SAR Symposium (CISS)*, 2022, pp. 1–5.

[84] O. Thakoor, J. Garg, and R. Nagi, "Multiagent uav routing: A game theory analysis with tight price of anarchy bounds," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 1, pp. 100–116, 2020.

[85] B. S. Weir and T. M. Sokol, "Radar coordination and resource management in a distributed sensor network using emergent control," in *Defense Transformation and Net-Centric Systems 2009*, R. Suresh, Ed., vol. 7350, International Society for Optics and Photonics. SPIE, 2009, p. 73500I.

[86] T. Kirubarajan and Y. Bar-Shalom, "Probabilistic data association techniques for target tracking in clutter," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 536–557, 2004.

[87] Y. Bar-Shalom, T. Kirubarajan, and X. Lin, "Probabilistic data association techniques for target tracking with applications to sonar, radar and eo sensors," *IEEE Aerospace and Electronic Systems Magazine*, vol. 20, no. 8, pp. 37–56, 2005.

[88] D. Musicki and R. Evans, "Joint integrated probabilistic data association: Jipda," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 40, no. 3, pp. 1093–1099, 2004.

[89] B. Habtemariam, R. Tharmarasa, T. Thayaparan, M. Mallick, and T. Kirubarajan, "A multiple-detection joint probabilistic data association filter," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 3, pp. 461–471, 2013.

[90] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-dujaili, Y. Duan, O. Al-Shamma, J. I. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: concepts, cnn architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, 2021.

[91] A. Shrestha and A. Mahmood, "Review of deep learning algorithms and architectures," *IEEE Access*, vol. 7, pp. 53 040–53 065, 2019.

[92] M. Soori, B. Arezoo, and R. Dastres, "Artificial intelligence, machine learning and deep learning in advanced robotics, a review," *Cognitive Robotics*, vol. 3, pp. 54–70, 2023.

[93] A. Aggarwal, M. Mittal, and G. Battineni, "Generative adversarial network: An overview of theory and applications," *International Journal of Information Management Data Insights*, vol. 1, no. 1, p. 100004, 2021.

[94] H. Naveed, A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Akhtar, N. Barnes, and A. Mian, "A comprehensive overview of large language models," 2024.

[95] A. Dimitriu, T. V. Michaletzky, V. Remeli, and V. R. Tihanyi, "A reinforcement learning approach to military simulations in command: Modern operations," *IEEE Access*, vol. 12, pp. 77 501–77 513, 2024.

[96] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, 1943.

[97] M. L. Minsky and S. A. Papert, *Perceptrons*. MIT Press, 1969.

[98] D. Rumelhart, G. Hinton, and R. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[99] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, *An Introduction to Deep Reinforcement Learning*, 2018, vol. 11, no. 3-4.

[100] A. Plaat, W. Kosters, and M. Preuss, "High-accuracy model-based reinforcement learning, a survey," *Artificial Intelligence Review*, vol. 56, no. 9, 2023.

[101] F. AlMahamid and K. Grolinger, "Reinforcement learning algorithms: An overview and classification," in *2021 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, 2021, pp. 1–7.

[102] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," 2015.

[103] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," 2016.

[104] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[105] G. T. Capraro and M. C. Wicks, "Metacognition for waveform diverse radar," in *2012 International Waveform Diversity & Design Conference (WDD)*. IEEE, 2012, pp. 348–351.

[106] H. T. Samson Lasaulce, *Game Theory and Learning for Wireless Networks: Fundamentals and Applications*. San Diego: Elsevier Science, 2011.

[107] J. Nash, "Non-cooperative games," *Annals of Mathematics*, vol. 54, no. 2, pp. 286–295, 1951.

[108] G. Bacci, S. Lasaulce, W. Saad, and L. Sanguinetti, "Game theory for signal processing in networks," 2015.

[109] J. Park and Y. Sung, "On the pareto-optimal beam structure and design for multi-user mimo interference channels," *IEEE Transactions on Signal Processing*, vol. 61, no. 23, pp. 5932–5946, 2013.

[110] O. Thakoor, J. Garg, and R. Nagi, "Multiagent uav routing: A game theory analysis with tight price of anarchy bounds," *IEEE transactions on automation science and engineering*, vol. 17, no. 1, pp. 100–116, 2020.

[111] C. Saraydar, N. Mandayam, and D. Goodman, "Efficient power control via pricing in wireless data networks," *IEEE Transactions on Communications*, vol. 50, no. 2, pp. 291–303, 2002.

[112] C. Papadimitriou and T. Roughgarden, "Computing correlated equilibria in multi-player games," *Journal of the ACM*, vol. 55, no. 3, pp. 1–29, 2008.

[113] O. Compte and P. Jehiel, "The coalitional nash bargaining solution," *Econometrica*, vol. 78, no. 5, pp. 1593–1623, 2010.

[114] G. H. G. H. Golub, *Matrix computations / Gene H. Golub, Charles F. Van Loan.*, 3rd ed., ser. Johns Hopkins studies in the mathematical sciences. Baltimore, Md.: Johns Hopkins University Press, 1996.

[115] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[116] S. Hart and A. Mas-Colell, "A simple adaptive procedure leading to correlated equilibrium," *Econometrica*, vol. 68, no. 5, pp. 1127–1150, 2000.

[117] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[118] D. Calitoiu, "New search algorithm for randomly located objects: A non-cooperative agent based approach," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6.

[119] D. Song, X. Wang, A. Dai, H. Ma, and D. Wei, "Cooperative search strategy of multiple uavs based on matrix game," in *2022 19th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, 2022, pp. 609–614.

[120] J. G. Baylog and T. A. Wettergren, "A roc-based approach for developing optimal strategies in uuv search planning," *IEEE Journal of Oceanic Engineering*, vol. 43, no. 4, pp. 843–855, 2018.

[121] C. Gao and Z. Zhao, "An optimal allocation approach of cooperative search capability based on game theory," in *The 2014 2nd International Conference on Systems and Informatics (ICSAI 2014)*, 2014, pp. 63–67.

[122] H. Zhao, H. Chen, F. Yang, N. Liu, H. Deng, H. Cai, S. Wang, D. Yin, and M. Du, "Explainability for large language models: A survey," *ACM Trans. Intell. Syst. Technol.*, vol. 15, no. 2, feb 2024.

[123] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[124] M. Khodarahmi and V. Maihami, "A review on kalman filter models," *Archives of Computational Methods in Engineering*, vol. 30, no. 1, 2023.

[125] D. L. Libby and M. R. Novick, "Multivariate generalized beta distributions with applications to utility assessment," *Journal of Educational Statistics*, vol. 7, no. 4, pp. 271–294, 1982.

[126] Y. Xiang, M. Akcakaya, S. Sen, and A. Nehorai, "Point detection, learning, and adaptation," *Circuits, Systems, and Signal Processing*, vol. 40, p. 233–261, 2021.

[127] D. Mata-Moya, N. del Rey-Maestre, V. M. Peláez-Sánchez, M.-P. Jarabo-Amores, and J. M. de Nicolás, "Mlp-cfar for improving coherent radar detectors robustness in variable scenarios," *Expert Systems with Applications*, vol. 42, no. 11, pp. 4878–4891, 2015.

[128] M. Rangaswamy and J. Michels, "A parametric multichannel detection algorithm for correlated non-gaussian random processes," in *Proceedings of the 1997 IEEE National Radar Conference*, 1997, pp. 349–354.

[129] A. Stringer, T. Sharp, G. Dolinger, S. Howell, J. Karch, J. G. Metcalf, and A. Bowersox, "Application of generative machine learning for adaptive detection with limited sample support," in *2024 IEEE Radar Conference (RadarConf24)*, 2024, pp. 1–6.

[130] J. Akhtar and K. E. Olsen, "A neural network target detector with partial ca-cfar supervised training," in *2018 International Conference on Radar (RADAR)*, 2018, pp. 1–6.

[131] M. V. i. Carretero, R. I. A. Harmanny, and R. P. Trommel, "Smart-cfar, a ma-

chine learning approach to floating level detection in radar," in *2019 16th European Radar Conference (EuRAD)*, 2019, pp. 161–164.

[132] A. Stringer, "First principles machine learning in radar: Augmenting signal processing techniques with machine learning for detection, tracking, and navigation," PhD thesis, University of Oklahoma, Norman, OK, August 2024, available at https://shareok.org/items/9c6bb8a2-ee5c-496f-ba26-ec2d2970d6db.

[133] G. Blinowski, A. Ojdowska, and A. Przybyłek, "Monolithic vs. microservice architecture: A performance and scalability evaluation," *IEEE Access*, vol. 10, pp. 20 357–20 374, 2022.

[134] W. Hasselbring and G. Steinacker, "Microservice architectures for scalability, agility and reliability in e-commerce," in *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*, 2017, pp. 243–246.