NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

RED CELL ANALYSIS FOR MOBILE NETWORKED CONTROL SYSTEMS

by

Larry W. Wigington

June 2021

Thesis Advisor:	Ruriko Yoshida
Co-Advisor:	Douglas P. Horner
Second Reader:	Samuel E. Buttrey

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE June 2021	3. REPORT TYPE AND DATES COVERED Master's thesis

4. TITLE AND SUBTITLE RED CELL ANALYSIS FOR MOBILE NETWORKED CONTROL SYSTEMS	5. FUNDING NUMBERS
6. AUTHOR(S) Larry W. Wigington	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	12b. DISTRIBUTION CODE A

13. ABSTRACT (maximum 200 words)

In the near future, networked unmanned autonomous systems will increasingly be employed to support ground force operations. Approaches to collaborative control can find near-optimal position recommendations that optimize over system parameters such as sensing and communication to increase mission effectiveness. However, over time these recommendations can create predictable paths that may provide leading indications of the force's operational intent. Using time series forecasting methods and deep neural networks, this thesis conducts an adversarial assessment of unmanned mobile networked control systems. In the first scenario, the path of the team's ground motion predicted by the model follows the initially planned but not executed path. In a second scenario, the model achieves a maximum path error rate of only 75 meters. In both cases, this methodology correctly identifies the direction and distance the team would travel and even identified points where the team changed direction, allowing the autonomous red cell analysis to discern the ground force's intent. These results indicate that automated red cell analysis is a potentially valuable component in planning and executing unmanned mobile networked control systems supporting expeditionary ground teams. It provides near real-time feedback on the unmanned agents' paths to determine if course adjustments can reduce operational intent predictability.

14. SUBJECT TERMS UAV, machine learning, adversarial machine learning, AI vs. AI, NCS, control systems, counter-reconnaissance, expeditionary advanced base operations, multi-domain operations	15. NUMBER OF PAGES 97
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

RED CELL ANALYSIS FOR MOBILE NETWORKED CONTROL SYSTEMS

Larry W. Wigington
Major, United States Marine Corps
BS, Troy University, 2009
M, Systems Analysis, Naval Postgraduate School, 2017

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
June 2021

Approved by: Ruriko Yoshida
Advisor

Douglas P. Horner
Co-Advisor

Samuel E. Buttrey
Second Reader

W. Matthew Carlyle
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

In the near future, networked unmanned autonomous systems will increasingly be employed to support ground force operations. Approaches to collaborative control can find near-optimal position recommendations that optimize over system parameters such as sensing and communication to increase mission effectiveness. However, over time these recommendations can create predictable paths that may provide leading indications of the force's operational intent. Using time series forecasting methods and deep neural networks, this thesis conducts an adversarial assessment of unmanned mobile networked control systems. In the first scenario, the path of the team's ground motion predicted by the model follows the initially planned but not executed path. In a second scenario, the model achieves a maximum path error rate of only 75 meters. In both cases, this methodology correctly identifies the direction and distance the team would travel and even identified points where the team changed direction, allowing the autonomous red cell analysis to discern the ground force's intent. These results indicate that automated red cell analysis is a potentially valuable component in planning and executing unmanned mobile networked control systems supporting expeditionary ground teams. It provides near real-time feedback on the unmanned agents' paths to determine if course adjustments can reduce operational intent predictability.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

Adam	Adaptive Movement Estimation
AR	Auto-Regressive
ARIMA	Auto-Regressive Integrated Moving Average
CAVR	Center for Autonomous Vehicle Research
CENETIX	Center for Network Innovation and Experimentation
CRUSER	Consortium for Robotics and Unmanned Systems Education and Research
COMTHIRDFLT	Commander, US Third Fleet
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
D1	Dissimilarity Matrix 1
D2	Dissimilarity Matrix 2
D3	Dissimilarity Matrix 3
DDG	Guided Missile Destroyer
DNN	Deep Neural Network
DOD	Department of Defense
DS	Distributed Submodularity
DTN	Delay-Disruption Tolerant Network
GFT	Ground Force Triangulation
HPC	High Performance Computers
IDS	Multi-Thread Experiment (MTX) Infiltration Data Set

IPB	Intelligence Preparation of the Battlefield
ISR	Intelligence, Surveillance, and Reconnaissance
JIFX	Joint Interagency Field Experimentation Program
LSTM	Long Short-Term Memory
LTI	Linear Time-Invariant
MA	Moving Average
MAPE	Mean Absolute Percentage Error
MASE	Mean Absolute Scaled Error
MIT	Massachusetts Institute of Technology
MLR	Multiple Linear Regression
MTX	Multi-Thread Experiment
NED	Northing Easting Down
NIWC	Naval Information Warfare Systems Command
NNETAR	Neural Network Auto-Regressive Time Series Model
NPS	Naval Postgraduate School
NSW	Naval Special Warfare
NCS	Networked Control System
RLM	Robust Linear Models
ROS	Robot Operating Systems
RNN	Recurrent Neural Network
SCI	San Clemente Island
SELU	Scaled Exponential Linear Units

SNN	Shallow Neural Network
UxV	Unmanned Vehicle
UAV	Unmanned Aerial Vehicle
UDS	Unseen Data Set
USV	Unmanned Surface Vehicle
UUV	Unmanned Underwater Vehicle

THIS PAGE INTENTIONALLY LEFT BLANK

Executive Summary

This thesis uses prior Naval Postgraduate School (NPS) work with Unmanned Vehicle (UxV) Networked Control System (NCS) and develops a novel framework to determine if multiple, distributed heterogeneous unmanned systems that support an expeditionary ground force behave in a manner that can provide operational intent of the ground force it supports. A NCS consists of heterogeneous agents, which include unmanned and manned assets, where each node has communications, sensing, and mobility capabilities and constraints. Recent research in this area of collaborative control has produced in methodologies that can find near-optimal position recommendations that optimize over system parameters, such as sensing and communication, to increase mission effectiveness, ensuring a ground force has sufficient communications capabilities or Intelligence, Surveillance, and Reconnaissance (ISR) of its target. However, over time these recommendations can create predictable paths that may provide leading indications of the force's operational intent. If an opposing force can track the unmanned systems in the UxV NCS would they be able to ascertain information such as where the ground force is or where the ground force is going?

Determining if these unmanned systems' behavior telegraphs operational intent requires several assumptions; paramount is the ability for the adversary to observe and collect position data about the UxV NCS. Therefore, this thesis begins by examining a worst-case. In this scenario, the adversary has perfect information about previous force disposition, system configurations, and access to detailed training data containing time-stamped geolocation information linking the ground force to the UxV NCS.

The next assumption required in this research is that opposing forces will use Artificial Intelligence and Machine Learning Techniques to discover operational patterns. This thesis analyzes the "worst-case scenario" data set from a data science approach; first, by applying unsupervised learning methods to learn interesting relationships or identify clusters present within the data set. Using the insights from the unsupervised learning methods, the next set of analyses focuses on supervised learning methods to determine if the grid coordinates or position data of the ground force can be inferred from the positions of the UxV network. And although these methods are developed from the "worst-case scenario" this research does not assume an adversary will know if there is a ground force present in the vicinity of

the autonomous agents. Rather, the communications constraints and methods learned from this analysis will provide the ability to focus an adversarial search and detect approach and potentially provide advanced warning of a target.

A more likely scenario is that an adversary will have access to similar training data that contains ground force and UxV positions and will have already inferred the relationships between a UxV NCS and a ground force. Therefore, the adversary is likely able to observe the UxV NCS but not the ground force, as in the "worst-case scenario." Taking the hidden ground force location into account, a novel approach is presented. The automated red cell analysis methodology generally follows a Bayesian Hierarchical Model, using a series of time series regression models to forecast UxV positions into the future and then, using those forecasted locations, triangulate the ground forces future path. These models are developed using the "worst-case scenario" data and then applied to the "more likely scenario" data.

The time series neural network regression model correctly identified the ground force's operational intent in both scenarios. The ground force's predicted path deviated from the actual path by an average of only 39 meters. The Automated Red Cell methodology is a novel framework that quantifies how autonomous agents may telegraph or predict a ground force's operational intent. The implications of this methodology and analysis are far-reaching as the Department of Defense (DOD) begins to focus on competing with near-peer adversaries in the Pacific, and the Marine Corps identifies the need for reconnaissance and counter-reconnaissance capabilities when conducting operations within the "weapons engagement zone."

Acknowledgments

I would like to thank my advisor, Dr. Ruriko Yoshida, for her guidance and expert advice throughout this research. Dr. Yoshida allowed me to take this research everywhere I could and explore any aspect that seemed promising, all while keeping me on track. The time we spent talking through hard problems proved to be invaluable.

I would also like to thank my co-advisor, Dr. Douglas Horner, for his expertise in autonomous systems and practical approach problems. Dr. Horner's mentorship and guidance not only focused this research but invigorated my interest in robotics and autonomous systems and helped me get to creating an operational model of my framework.

Finally, I would like to thank my wife, Tiffinni, and our girls for their unwavering support and encouragement during my time at Naval Postgraduate School. Tiffinni has heard more about machine learning and drones than she would care to admit, but she has been my constant sounding board for ideas and is not afraid to tell me when they are good or bad. She keeps me grounded and focused; without her, none of my successes would be possible.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1: Introduction

Future Department of Defense (DOD) missions will leverage multiple, distributed, heterogeneous, unmanned systems to support a human component (Defense Science Board 2012). These unmanned systems will increasingly be employed in cooperation with one another through communication networks to increase mission effectiveness, in part, through their collective ability to efficiently identify objectives based on selective criteria and disseminate information across a network of systems. Research conducted at Naval Postgraduate School (NPS) by Wachlin (2018) and Lowry (2020) focused on a centralized and distributed adaptive submodular optimization approach to find near-optimal, near-real-time, position solutions for agents in the Networked Control System (NCS). Exploratory analysis of the data generated from this system, detailed in Chapter 3, reveals that the path of the ground team and the autonomous unmanned agents that make up the NCS are predictable. An adversary can detect clear patterns of motion, especially once it can be determined that multiple autonomous agents are part of the same network and working in concert.

1.1 Problem Statement

An exploratory analysis of the data generated from an NCS reveals that the behavior of autonomous unmanned agents may provide leading indications of a ground force's path and intent. This thesis investigates an adversary's ability to predict the paths of an observable autonomous multi-vehicle network and its associated ground force. Existing control systems for collaborating unmanned systems seek to calculate positions or trajectories that maximize the utility of the system, and recent work by Wachlin (2018) and Lowry (2020) can find near-optimal network configurations by optimizing the network's sensing potential and communication robustness. The optimized network allows the network control system to make decisions and share information to ensure that the failure of one or more nodes will not significantly impact the network (Lowry 2020). However, the resulting network creates predictable paths between configuration points. The paths and network configurations can determine the path and target of the ground forces supported by the autonomous vehicle network. This thesis does not predict autonomous agent behavior from a control systems

approach; rather, it predicts autonomous agent behavior using a more generalized approach through data science and statistical machine learning tools.

1.2 Case Scenario

To test this framework, this thesis analyzes data from a real-world experiment where a network of autonomous vehicles supported a ground force. In November of 2017, NPS designed and conducted a Multi-Thread Experiment (MTX) series on San Clemente Island (SCI), which brought together faculty and students from across NPS including, Center for Autonomous Vehicle Research (CAVR), Center for Network Innovation and Experimentation (CENETIX), Consortium for Robotics and Unmanned Systems Education and Research (CRUSER), and Joint Interagency Field Experimentation Program (JIFX), as well as fleet sponsors such as Naval Special Warfare (NSW), Commander, US Third Fleet (COMTHIRDFLT) and Naval Information Warfare Systems Command (NIWC). The experiment provided a platform to test the collaboration of an autonomous network of Unmanned Vehicle (UxV) with manned assets. The scenario was built around a NSW team tasked with landing on SCI and traversing the island to conduct a direct action mission on a known target. The NSW team was supported by "manned and unmanned assets operating together to provide Intelligence, Surveillance, and Reconnaissance (ISR) support, transportation, and battle-space awareness" (Lowry 2020, p. 2). The MTX generated a good data set to test the hypothesis that NCS generates predictable paths that can provide an adversary indications of a ground force's operational intent. Additionally, the MTX scenario provides the opportunity to conduct a more robust analysis since the NCS dynamically positions itself to support the ground force, whereas providing a different ground force path to the NCS generates new autonomous agent configurations and behavior. However, the data from the MTX does present some limitations; specifically, that the exercise is being conducted on an island, which channelizes the possible paths a ground force may take.

1.3 Proposed Analytical Framework

This research proposes a framework to conduct an adversarial, or red cell, analysis of the detectable network using machine learning techniques to predict each autonomous vehicle's path and the ground force's path and target. Additionally, this thesis establishes the foundation to automate this capability, which could be later incorporated into the NCS

methodology to disguise the ground force's intent. This thesis begins by analyzing the NCS-generated data from all aspects of data science, including supervised learning and unsupervised learning. The automated analytical framework detailed in this thesis consists of two independent models and a hierarchical model. There are two data sets used to support this analysis, where each data set consists of five UxV, a Guided Missile Destroyer (DDG), and an NSW Team's coordinate pairs with associated time stamp values for several thousand observations. The goal of this analysis is inferring relationships between the UxVs and NSW team and predicting future NSW team behaviors. The UxV Forecast Model conducts univariate time series analysis of the *x*- and *y*- coordinates, where the *x* and *y* values are generated for a localized grid coordinate system in a Northing Easting Down (NED) configuration and the point (0, 0) is based at lat 33.03191N, lon -118.60428W. This model builds twelve independent models and then forecasts each model a certain number of steps into the future. The output of the UxV Forecast Model creates the predicted locations of all components of the NCS. The Ground Force Triangulation (GFT) model uses numeric regression techniques to calculate the ground force's location as a function of UxV positions. The trained GFT model is then used in the time series regression model; the outputs of the GFT model are compared to the validation data to determine if the model is sufficient. The time series regression model combines the predictions from the UxV forecast model with the optimized model from the GFT model to generate the ground force's predicted locations, which can then be compared against their actual positions. A detailed overview of the methodology and models are contained in Chapter 3 and Figure 3.11 provides a general overview of the data and how it is analyzed in this thesis.

In order to investigate how this approach could be used as part of an integrated approach with the UxV NCS, this thesis explores three different implementations of this analytical framework; the micro-computer, portable/laptop computers, and high-performance computer. The resulting analytical framework should be implementable in an autonomous system that can make decisions near real-time, on micro-computers, or edge-computing devices, such as the Raspberry Pi or other micro-computing platforms capable of controlling a single autonomous vehicle. Automated Red Cell Methodology on micro-computers relies on algorithms that are not computationally complex and that can be accomplished with very little computing power, such as multiple linear regression techniques combined with simple time series analysis methods. In addition to being implemented on micro-computers, this

analytical framework was primarily researched using a portable laptop computer.

In an NCS, the automated red cell will not need to be conducted on the edge devices. Instead, it can be conducted by a centralized control station, which will likely have at least the processing power of a portable laptop computer. This portable laptop implementation provided the most insight into the NCSs and provided the best all-around performance concerning run-time and accuracy of predictions.

This analytical framework can also integrate recurrent neural networks, long-short term memory, and deep neural networks into the time series analysis model. However, these models exceed the computational capabilities of portable laptop computers and require the processing power of dedicated workstations or high-performance computers.

This chapter states that future DOD missions will require a combination of manned and unmanned systems to support a ground force and that these systems are controlled via an NCS. These control systems create clear patterns of motion that, if observed, can create leading indications of the ground force's operational intent. Data from NPS' 2017 MTX series is analyzed from an adversary's perspective to determine if the autonomous agent behavior generated by these systems provide leading indications of the ground force's intent and then proposes the framework to conduct this adversarial, or red cell, analysis.

In Chapter 2, the concepts related to the NCS, submodularity, neural networks, and Red Teaming are reviewed. Chapter 3 will provide an overview of the methodologies used throughout this analysis and introduce the various time series regression models developed by this analysis. This research will explore several analytical methods to determine if machine learning algorithms can predict the target of an NSW team, their path, and the paths of the UxV swarms. Chapter 4 details the resulting models generated as part of this framework as well as the numerical results from applying the analytical framework to the new, unseen data set. Because an automated red cell analysis may be applied to various computing platforms, Chapter 5 will also address the frameworks and initial implications of applying this analytical framework to microcomputers, mobile computing platforms, and high-performance computers. Finally, Chapter 6 will discuss the implications and future work that should arise from this analysis. This chapter will also discuss how the analytical framework is generalizable and applies to fields other than communications networks.

CHAPTER 2:
Background

Sections 2.1 and 2.1.1 introduce and detail the topics of NCS and submodularity as they are directly related to the control of UxVs. Section 2.2 introduces the applications of statistical machine learning and neural networks in the areas of trajectory prediction and position triangulation. Finally, Section 2.3 will review the red cell and red team concept and their application within an automated framework.

2.1 Networked Control Systems

Consistent with DOD planning documents—such as the Defense Science Board's report on Autonomy (2012), the Marine Corps' Force Design 2030 (2021) and the Navy's NAVPLAN 2021 (2021)—it is assumed that a UxV NCS will be used for future expeditionary warfare missions. Furthermore, it is assumed that there will be centralized or distributed software used to control the system. This thesis uses simulation results from a distributed system control implementation. This section reviews important aspects of NCS that were researched and implemented by Wachlin (2018) and Lowry (2020) since this research is an investigation of the data generated from their work.

An NCS can be "described as a system composed of multiple discrete entities with control loops that share feedback and control signals over a shared communications network" (Zhang et al. 2016, p. 1740). These systems are either static or dynamic, where a static NCS's agents do not change state; they are either a sensor or a part of the communications network. The work conducted by Wachlin (2018) and Lowry (2020) focus primarily on a dynamic NCS implementation, which contains one or more agents whose state varies. The NCS described and researched in this thesis are "characterized by independent decision-making agents, locally sensed information, and limited communication abilities working towards cooperative control" (Lowry 2020, p. 5).

The NCS investigated in this thesis are modeled as a Linear-Time Invariant system where the system model, measurement model, and overall controllability, observability, and communications are determined through a system graph representation. Research conducted

on formation control focuses on centralized and distributed approaches. Centralized approaches can reach a global consensus easier, while distributed approaches are more robust in their approaches to control, in that if the centralized system's main node fails, the entire system becomes uncontrolled (Wachlin 2018).

2.1.1 Centralized Networked Control Systems

In a centralized approach, the autonomous agents "actively control their position as prescribed by a centralized control algorithm that provides a specific location (and velocity) to actuate toward" (Wachlin 2018, p. 37). Often, these systems do not require interaction between the agents in the system since a centralized unit controls them. Wachlin (2018) takes advantage of this and develops a methodology to control an NCS composed of heterogeneous agents which ensures "adequate controllability, observability, and robustness of a Linear Time-Invariant (LTI) system." Critical aspects of this method are the use of a graph-based framework and submodular utility function. This methodology creates a graph-based network of vehicles where each of the UxVs is a node, and their communications connections are the arcs. The graph framework developed by Wachlin is used better to quantify the resiliency and robustness of the network. Submodularity is the concept that allows the NCS to generate its near-optimal position recommendations. (Wachlin 2018, p. 41) cites Krause and Golovin (2014) in describing submodularity as "a property of set functions commonly referred to as as the property of diminishing returns; that is adding a node to a subset of B, will produce a larger utility gain than adding the node to B."

Graph-Based Framework

Using a graph representation provides several advantages. First, through the use of the Laplacian matrix, it permits determining controllability and observability of the system (Mesbahi and Egerstedt 2010). Second, it permits analysis related to resiliency, robustness and connectivity of communications. Wachlin investigated several options to determine a single metric for assessing all aspects of the network and recommended *effective graph resistance* (R) as the desired metric (Kooij 2013).

Submodular Utility Function

Submodularity is a very particular property of set functions that, in certain models, certain optimization algorithms can take advantage of, providing optimal solutions within proven bounds. A thorough discussion of submodular set functions and their applications in optimization and function maximization can be found in Krause and Golovin (2014). A *set function* $f : 2^V \Rightarrow \mathbb{R}$, where 2^V is the power set of a set V, is a function that assigns a real value $f(S)$ to every subset $S \subseteq V$. The key property of certain set functions leveraged in Lowry (2020) is called submodularity and it is defined as,

Submodularity: A set function $f : 2^V \Rightarrow \mathbb{R}$ is submodular for any A and B, with $A \subseteq B \subseteq V$, and any $s \notin B$,

$$f(A \cup s) - f(A) \geqslant f(B \cup s) - f(B).$$

The submodular set functions allow the algorithms used in the NCS to find near-optimal position recommendations. The centralized control approach to NCS is interested in finding a set of node positions that will maximize the utility function $f(s)$. Wachlin (2018, p. 44) presents the methodology and analysis used to select the submodular utility function used to evaluate the NCS model. The resulting utility function J is a convex combination of two functions that quantify sensing ability (f_s) and communications robustness (f_r), where the associated α values are non-negative coefficients that weigh their respective functions as

$$\begin{aligned} J(S) &= \alpha_s f_s(S) + \alpha_r f_r(S) \\ \text{s.t. } & \alpha_s + \alpha_r = 1 \\ & \alpha_r, \alpha_s \geqslant 0. \end{aligned} \tag{2.1}$$

Sensing Subfunction

The sensing subfunction calculates the network's ability to sense its environment and provide coverage of an area of interest, $f_s(S)$, from Equation 2.1. The sensing subfunction allows a user to identify a sensing benefit for a given target. Wachlin (2018, p. 45) provides

the following example for a sensing subfunction:

> If we place node i at location j, we say that it provides a sensing benefit of $\Theta_{i,j}$, where $\Theta \in \mathbb{R}^{Nxp}$. If each node is assigned to the location with the largest benefit, the total value is the set function
>
> $$f_s(S) = \sum_{i=1}^{N} \max_{j \in S} \Theta_{i,j}.$$

This sensing subfunction allows the network to either be explicitly biased towards certain points of interest or self-identify points of interest by using feature recognition software and also allows the network to identify locations where certain nodes cannot physically go (Wachlin 2018).

Robustness Subfunction

The robustness subfunction uses effective graph resistance—discussed in Section 2.1.1—to quantify the robustness of the network. Wachlin (2018, pp. 47–48) further expands on the robustness subfunction, stating

> Since the graph becomes more robust as the effective resistance decreases, we manipulate the resistance to form a new metric Ω that conforms to the maximization problem
>
> $$f_r(S) = \sum_{i=1}^{N} \max_{j \in S} \Omega_{i,j}$$
>
> where $\Omega_{i,j} = 1 - \text{norm}(R_{i,j})$. Here, $R_{i,j}$ is the effective resistance if a node i is added to the network at location j.

Calculating the robustness metric in this way allows for maximization of the network's robustness by optimizing the placement of nodes and using communications strength to calculate the communications-based edge weights.

2.1.2 Distributed Networked Control Systems

The preceding versions of submodular optimization rely on a centralized controller to compute and plan the positions of all nodes within the NCS. Within the DOD there are several issues with centralized control that make the entire network vulnerable to enemy actions. In addition to the susceptibility of equipment failures, in an adversarial environment, should the location (or nature) of the centralized controller be discovered by an adversary, it could target the central control to disrupt or disable the NCS. Lowry (2020) presents a framework that uses distributed submodular optimization to control the NCS. In this framework, each node shares the computing burden while removing the vulnerability of only having a centralized planner. Lowry (2020) goes on to present the applicability and limitations of the approaches that most closely address the decentralized NCS control problem and develops an algorithm for distributed submodularity.

According to Lowry (2020), the goal of distributed submodularity method is to find a near-optimal position set S from V possible discrete locations that maximizes the function $f(S)$. The Distributed Submodularity (DS) method relies on a connected network with no disconnected nodes, where each node must possess knowledge of the operating area. Lowry (2020) presents a simple, novel algorithm for the DS method that optimizes node placement on a deterministic map of which all agents have complete knowledge. The DS algorithm proceeds with each node repeating a three-step process which re-positions the NCS using the following phases:

1. **Local Search**: Each node concurrently and greedily evaluates a local area for its optimal position. It assumes all other nodes in the NCS remain in place. The node then finds the location corresponding to the maximum increase in total utility that it can effect. This phase accounts for both the dynamic constraints on the vehicles and any overlapping sensor coverage that would result from a potential move.
2. **Information Sharing**: All nodes then broadcast two pieces of information: the maximum contribution to total NCS utility, and the location it will re-position to if its contribution is greater than all other nodes.
3. **Evaluation and Update**: The node which can effect the greatest change in total utility is designated and re-positions itself, while all others update the position vector $\mathbf{X}$. Each node then calculates the new value for J, the utility function found in equation 2.1, and ΔJ, the change in the utility function. This phase repeats until $\Delta J \leq 0$,

indicating the near-optimal topology has been reached.

2.2 Neural Networks

This section presents some of the background and fundamental information behind the machine learning and neural networks used in this thesis. Machine Learning is a sub-field of computer science that describes solving problems by gathering a relevant dataset and an algorithm using the dataset to build a statistical model (Burkov 2019). James et al. (2013) state that statistical and machine learning problems generally fall into three categories: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning methods are statistical tools that start with n observations of a given set of p features $X_1, X_2, \ldots, X_p$, where each set of features has a response Y. Supervised learning aims to predict Y using the values of $X_1, X_2, \ldots, X_p$. In contrast unsupervised learning only has n observations of a set of features $X_1, X_2, \ldots, X_p$ and no associated response Y. The goal of unsupervised learning is not prediction but rather the discovery of interesting observations or relationships present among $X_1, X_2, \ldots, X_p$. Although reinforcement learning may have eventual applications to this type of research, these methods are not applied in this thesis.

"Artificial neural networks are popular machine learning techniques that simulate the mechanism of learning in a biological organism," by mimicking the neurons in a brain (Aggarwal 2018, p. 1). Aggarwal (2018) states that "an artificial neural network computes a function of the inputs by propagating the computed values from the input neuron to the output neuron(s) and using the weights as intermediate parameters" and that learning occurs by changing the weights connecting the neurons (p. 2). Aggarwal (2018) provides in-depth discussion, definition, and overviews of the many aspects of neural networks, but states very simply that "neural networks are built as higher-level abstractions of the classical models that are commonly used in machine learning" (p. 2). Neural networks also have the added property of often being more accurate than classical models when sufficient data and computational power are available.

Time Series analysis does not use many of the traditional models and analysis methods since the traditional methods require data to be independent observations. "Neural architectures are inherently designed for multidimensional data in which the attributes are largely independent of one another; however, certain data types, such as time series data, contain

sequential dependencies among the attributes" (Aggarwal 2018, p. 271). In a Recurrent Neural Network (RNN), there is a "one-to-one correspondence between the layers in the network and the specific positions in the sequence," that allow for the processing of data where the order of the data in the sequence is relevant and important to the model, such as time series data. (Aggarwal 2018, p. 273). There are two requirements for processing sequences or time series data, including the "ability to receive and process inputs in the same order as they are present in the sequence and that the treatment of inputs at each time-stamp in a similar manner in relation to the previous history of inputs" (Aggarwal 2018, p. 273). Aggarwal (2018) further claims that RNN are *Turing complete* algorithms, which indicates the algorithm is capable of simulating any other algorithm if it has enough training data and computational resources. Figure 2.1 provides a general overview of the different Machine Learning paradigms, their model outputs, and common each uses.

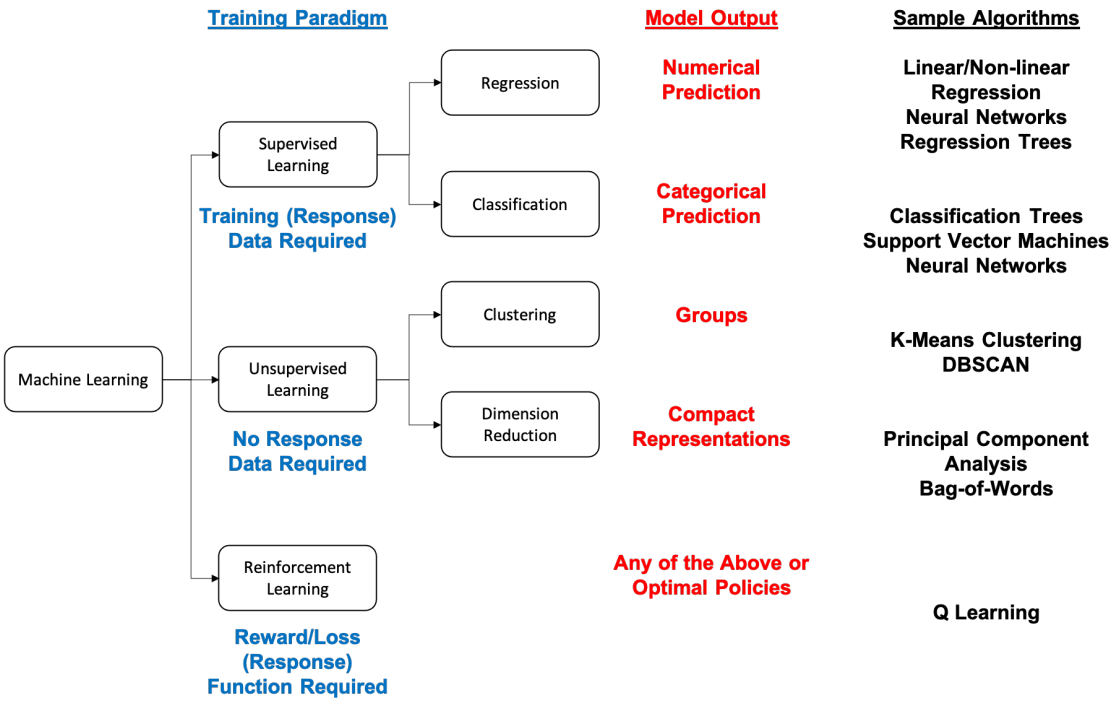

Figure 2.1. Overview of Machine Learning Applications. Source: Yoshida (2020).

2.2.1 Trajectory Prediction

Recent innovations in the automobile industry have given rise to self-driving car technology. These emerging technologies have identified a new problem; how these cars predict and react to pedestrian actions. It naturally follows that "predicting the movement of dynamic objects is a central problem for autonomous agents, and anticipation by prediction is required for smooth and safe path planning in a changing environment" (Mangalam et al. 2020, p. 1). There has been increased research into forecasting pedestrians' trajectories and their behaviors, including observed motion trajectories for future trajectory prediction and the use of scene and social information (Mangalam et al. 2020). The method of forecasting pedestrian trajectories is not used in this research; however, many of the aspects of predicting the behaviors and trajectories of the components of a NCS are similar. First, the components of a NCS, like pedestrians, have some understanding of their long-term desired destination, and second, the UxV plans a trajectory to reach its next desired position or sub-goal. Mangalam et al. (2020) propose that the trajectory of a pedestrian can be predicted given the past t_p steps, as a sequence of coordinates, and by assuming that all humans in the scene act in a cooperative manner and also respect social norms. Their methodology considers a pedestrian's possible sub-goal for the current sequence, and then, jointly considering the past locations of all pedestrians present, they estimate their new endpoints. Their implementation uses multi-layer perceptrons with ReLU non-linearity, trained end-to-end with a predefined loss function using an ADAM optimizer with a batch size of 512 and a learning rate of 3×10^{-4} for all experiments. Mangalam et al. (2020) note their methodology identifies that "pedestrians, having a predilection towards their destination, exert their will towards it; hence predicting the last observed way-point allows for a lower prediction error than way-points in the middle" (Mangalam et al. 2020, p. 13) claim that their methodology produces a diverse prediction set, which is conditioned on each pedestrian's inferred endpoints.

Although this thesis follows a similar methodology, the specific formulas to achieve NSW path predictions are different.

2.2.2 Position Triangulation

Position triangulation is not a new concept; it is often shown in movies and television shows when someone is missing. Police contact the missing person's cell phone provider and request that the missing person's cell phone be pinged. The provider uses the last known

signal location and distances to multiple known cell phone towers, and the police have a good idea where to search for the missing person. Research conducted by Shareef et al. (2007) explores the use of neural networks to solve localization problems and state that "localization is used in location-aware applications such as navigation and autonomous robotic movement to position a moving object on a coordinate system" (Shareef et al. 2007, p. 1). Triangulation is an analytical localization method where three or more distance measurements from known points are used to identify an unknown position. However, the accuracy and calibration of sensors can cause the measurements to become noisy or fluctuate, making localization more difficult. Shareef et al. (2007) propose that deep learning, or as they call them multi-layer perceptrons, and RNN are capable of conducting localization even with noise and are used to investigate various neural networks implementations to gauge their ability to triangulate a sensor using Massachusetts Institute of Technology (MIT) Cricket sensors, which are "rarely constant and fluctuate often" (p. 3). Cricket is a location architecture developed by MIT that allows sensors, laptops, handheld computers, and other devices to know about their current location. The experiment takes the distances from each of the MITCricket sensors to the mobile node as an input to the neural network. The output of the neural network is the estimated location of the mobile node. Shareef et al. (2007) developed Shallow Neural Network (SNN) and RNN models and compared their accuracy against other localization methods. The SNN consisted of two layers with nine nodes in the first layer and two nodes in the output layer, and the RNN followed a structure nearly identical to that of the SNN, "except that the outputs of the first layer were fed back to each of the nodes in the first layer as inputs" (Shareef et al. 2007, p.4). The results of this experiment show that both networks have a high percentage of error; however, the error was less than 10 cm. Additionally, the performance of the SNN and RNN are nearly identical, as detailed in Table 2.1.

Table 2.1. Comparison of Localization Error (cm). Adapted from Shareef et al. (2007).

Method	Dist Error	RMSE	Net RMSE
SNN	5.726	(5.905, 4.693)	7.543
RNN	5.738	(5.936, 4.710)	7.576

This use of neural networks for localization is further explored in Chapter 3 and will be a central theme in this thesis.

2.3 Red Cell/Red Team

Red cells and red teams provide an objective, or pragmatic, opinion to a commander about the commander's plans and how an adversary may respond to them. More specifically, "Red teams are established by an enterprise to challenge aspects to that very enterprise's plans, programs, or assumptions and provide a wider and deeper understanding of potential adversary options and behavior that can expose potential vulnerabilities in our strategies, postures, plans, programs, and concepts" (Defense Science Board Task Force 2003, p. 2). The Defense Science Board Task Force (2003) contends that in some cases, the red cell will emulate an adversary and offer critiques and "alternatives to the enterprise's assumptions, strategies, plans, concepts, programs, projects, and processes" (p. 2). Red teaming is an important part of military planning and an approach that has been integrated into the various military planning processes. However, the concept of red teaming does not appear to have been incorporated into previous work conducted on distributed network control systems.

The lack of incorporated red cell analysis in a distributed network control system is not surprising; autonomous networked systems do not operate randomly, systems of control and levels of predictability are required to guarantee stability, performance, and fulfillment of all constraints (Farina and Misiano 2018). In addition to controlling the individual components of an NCS, the control system must be able to make predictions state of the system to meet constraints of more than one subsystem at the same time, including "collision avoidance between pairs of vehicles or minimal/maximal total energy production requirements" (Farina and Misiano 2018, p. 1413).

While red teaming has not been introduced to network control science, it has been studied, applied, and even automated in other aspects of the DOD and industry, for example, cyber-security. Numerous "academic researchers have spent time and resources developing tools and techniques to identify vulnerabilities found in software applications and cyber-physical systems," because "it is reasonable to conduct a thorough vulnerability assessment of a small network manually but that it becomes prohibitively cumbersome to assess a large and complex network due to time, effort, and skill requirements" (Plot 2019, p. 10). Because of this need throughout industry and the DOD, many automated tools have been created to conduct red team analysis on computer networks. These frameworks and tools are integrated into the computer networks and devices not connected directly to the internet to reduce

vulnerabilities within their respective frameworks.

The next chapter provides a brief overview of the data science methods used in this analysis and how each of the specific areas was applied to develop this overall methodology.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Methodology

The ultimate goal of automating red cell analysis is to reduce the enemy's ability to determine the operational intent of a ground force, given the ground force is employing an NCS of autonomous vehicles. This methodology can be thought of as an internal process conducted by an NCS that looks at the paths of the entire network, naively attempts to predict the paths of the autonomous agents based solely on past locations, and uses the naive predictions as the inputs to a neural network triangulation model to predict the ground force's path. To accomplish this analysis, this chapter will analyze the MTX Infiltration Data Set (IDS), which comes from the Infiltration Phase of the 2017 MTX and is used in the development of the Automated Red Cell Methodology.

Section 3.1 introduces the data set used in the methodology development, including how the data was generated, its format, and a description of each variable. Section 3.2 provides an overview of the data science tools in the exploration of the IDS and will also provide insights gained from each method. Section 3.3 introduces the different types of neural networks that can be applied to this data set and provides the specific implementation of the Deep Neural Network (DNN) developed as a result of the analysis from Section 3.2. Section 3.4 develops the Hierarchical Model and provides the analytical results from its implementation.

3.1 The MTX Infiltration Data Set

The MTX data set is analyzed from the perspective of an adversary who has maliciously obtained the MTX data sets with the intent of learning the capabilities of an NCS and the DOD's tactics, techniques, and procedures. The IDS was obtained from the research conducted by Lowry (2020) concerning distributed submodularity. Lowry (2020) took real-world data from MTX scenario and introduced a distributed submodular framework to the NCS. His research generated three separate data sets that correspond to the three phases of the MTX: Intelligence Preparation of the Battlefield (IPB), insert, and infiltration. This chapter uses Lowry (2020)'s infiltration data set, consisting of 7,005 discrete observations in time, where each observation represents a single second. Each observation in the data

set contains the two-dimensional (X/Y) position coordinates for a NSW team, one DDG, three Unmanned Aerial Vehicle (UAV), two Unmanned Surface Vehicle (USV), and two Unmanned Underwater Vehicle (UUV). Table 3.1 shows the data used in this analysis, and although two UUVs were present in the exercise and data, these vehicles will likely not be observable in application and are therefore excluded from this analysis.

Table 3.1. MTX Infiltration Data Set

Data Name	Num Obs	Description
UAV1.X	7,005	x-coordinate of UAV 1
UAV1.Y	7,005	y-coordinate of UAV 1
UAV2.X	7,005	x-coordinate of UAV 2
UAV2.Y	7,005	y-coordinate of UAV 2
UAV3.X	7,005	x-coordinate of UAV 3
UAV3.Y	7,005	y-coordinate of UAV 3
USV1.X	7,005	x-coordinate of USV 1
USV1.Y	7,005	y-coordinate of USV 1
USV2.X	7,005	x-coordinate of USV 2
USV2.Y	7,005	y-coordinate of USV 2
DDG.X	7,005	x-coordinate of DDG
DDG.Y	7,005	y-coordinate of DDG
NSW.X	7,005	x-coordinate of NSW Team
NSW.Y	7,005	y-coordinate of NSW Team

3.2 A Data Science Approach

The Infiltration Data Set provided by Lowry (2020) was examined from a potential adversary's perspective. The analysis sought to discover any relationships that existed among the individual UxVs, the UxVs and the ground force, or the entire network of UxVs, ground force, and DDG. Any relationships discovered could then be developed into a comprehensive model that would allow the adversary to predict the UxV path, ground force's path, or other points of interest. This initial analysis uses unsupervised learning, supervised learning, and machine learning methods to infer information about how the NCS, UxVs, and the ground force interact. This section will analyze the data using unsupervised learning methods and use their insights to help develop the models used in the supervised learning methods. Table

3.2 provides a quick summary of the machine learning methods used in this thesis, their general applications, and the algorithms they use.

Table 3.2. Overview of Applied Machine Learning Algorithms

Paradigm	Model Output	Algorithms
Unsupervised Learning	Clustering	K-Means Hierarchical DBSCAN
Supervised Learning	Regression	ARIMA NNETAR Linear Regression Neural Networks

3.2.1 Unsupervised Learning

Because no relationships or meaningful insights were known about the data before analyzing the IDS, the analysis of the IDS begins by using unsupervised learning methods, including hierarchical clustering, k-means clustering, and Density-Based Spatial Clustering of Applications with Noise (DBSCAN), on all fourteen elements of the data set. James et al. (2013, p. 373) describe unsupervised learning as tools used when there are only observable features, $X_1, X_2, ..., X_p$, and no response variable. They further state that unsupervised learning tries to discover interesting things about the measurement of the features, including identifying groups or subgroups of variables or relationships between observations.

Although other approaches are certainly possible, as a proof of concept this analysis elected to conduct cluster analysis with a randomized 80/20 split, treating the 7,005 observations from the IDS as independent observations. The insights from the unsupervised learning methods were used to help target the supervised learning tools used in later analysis. Additionally, beginning with this type of exploratory analysis allows for a better understanding of the relationships a machine learning algorithm may identify or use.

Hierarchical Clustering

Hierarchical Clustering methods were applied to the IDS to determine how dissimilar each variable was from the others. A hierarchical clustering dendrogram is a tree-based

representation of the observations' clustering. It is obtained through a simple algorithm that measures the dissimilarity between a pair of observations or groups of observations (James et al. 2013).

Table 3.3. A Summary of the Hierarchical Clustering Linkages Method. Source: James et al. (2013).

Linkage	*Description*
Complete	Maximal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *largest* of these dissimilarities.
Single	Minimal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the smallest of these dissimilarities. Single linkage can result in extended, trailing clusters in which single observations are added one-at-a-time.
Average	Mean intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the average of these dissimilarities.

The dissimilarity function from the TSClust package in R (R Core Team 2021) was used to calculate the pairwise dissimilarity between the UxVs and NSW paths (Montero and Vilar 2014). Three separate dissimilarity matrices were produced to reflect the observations of the network and applicability of various manned and unmanned assets for future predictive analysis. Dissimilarity Matrix 1 (D1) consisted of the dissimilarity matrix with nearly all the time series data. Dissimilarity Matrix 2 (D2) consists of the dissimilarity matrix excluding the DDG time series and helps visualize the hierarchical cluster dendrograms since the DDG data created a large vertical separation between the DDG and all other components of the NCS. Dissimilarity Matrix 3 (D3) consists of the dissimilarity matrix, which excludes the NSW data, allowing for the illustration of apparent clusters when the NSW paths are unknown.

Hierarchical clustering dendrograms are based on different linkage methods. The different linkage methods define how dissimilarity between clusters is calculated. Table 3.3 provides

a summary of the linkages used in this analysis.

The different methods produced some variation in the dendrogram; however, using the Complete Method provides the most insight into the data. Figure 3.1 is a Hierarchical Cluster Dendrogram of the IDS which shows that the variable UAV3.Y has the closest relationship to NSW.Y and that variables UAV1.X has the closest relationship to NSW.X. Additionally, Figure 3.1 highlights one of the first non-intuitive insights into the IDS, that the DDG's position does not appear to cluster with any of the other attributes of the IDS. The NCS methodology used by Wachlin (2018) and Lowry (2020) require the communications network to be fully connected at all times, indicating that at least one of the components from the NCS should cluster with the DDG positions.

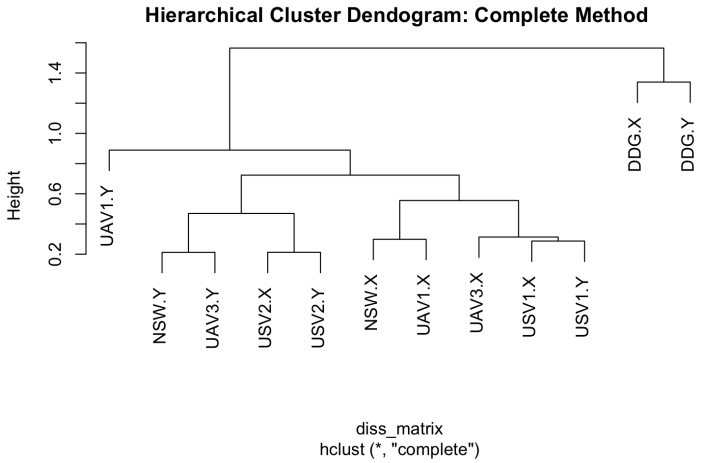

Figure 3.1. Hierarchical Cluster Dendrogram on Infiltration Data Set

Density-Based Spatial Clustering of Applications with Noise

The DBSCAN algorithm is used initially to determine if there are any clusters or noise in the IDS (Ester et al. 1996). Given a set of points in some space, DBSCAN groups together a point with many nearby neighbors and marks any points that lie alone in low-density regions as outliers.

DBSCAN accomplishes this with two simple input parameters; the *Epsilon-Neighborhood*,N_{Eps}, of a point, which is simply the radius of a local neighborhood around a point, and the *Min Points*, k, which is the minimum number of points that must be in the neighborhood for the point to be considered a "core point." Points which are within the

N_{Eps} of a core point, but who themselves do not have the required number of *Min Points*, k, are designated as border points. Figure 3.2 provides a simple depiction of the DBSCAN method.

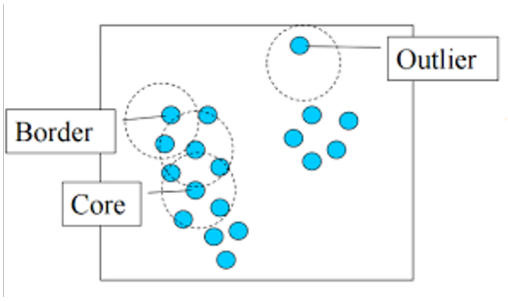

Figure 3.2. General Overview of DBSCAN Method (Buttrey 2021).

Using the DBSCAN package in R (Hahsler et al. 2019), the N_{Eps} was calculated with $k = 2$, which resulted in the best epsilon, Eps, value of 0.3. Figure 3.3 highlights the results from using the DBSCAN algorithm on the principal components of the data. This implementation of DBSCAN with $k = 2$ and $eps = 0.3$ created 4 distinct clusters:

1. NSW.X with UAV1.X
2. NSW.Y with USV1.X, USV1.Y, UAV1.Y, UAV3.X, and UAV3.Y
3. USV2.X with USV2.Y
4. DDG.X with DDG.Y and UAV1.Y

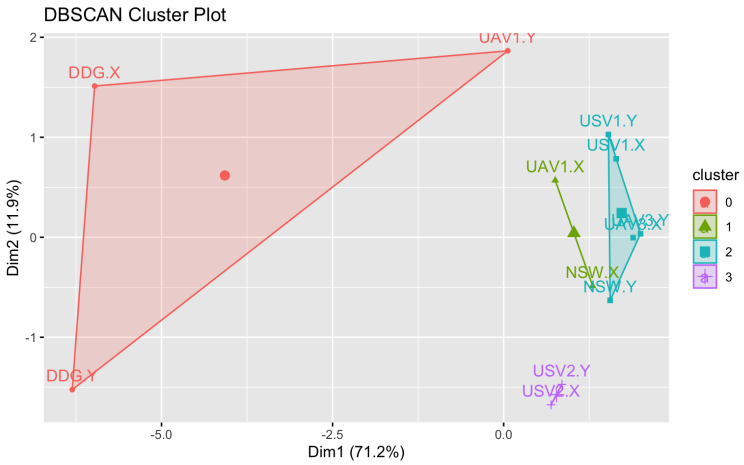

Figure 3.3. DBSCAN Cluster Analysis on Infiltration Data Set

K-Means Clustering

The IDS was also evaluated using k-means clustering to determine if there were useful or meaningful clusters in the data. The goal of k-means clustering is "to partition the observations into K distinct clusters, such that the total within-cluster variation, summed over all K clusters, is as small as possible" (James et al. 2013, p. 387).

For k-means clustering, the optimal k-value from both within-clusters sum of squares and the silhouette method was considered. Although both methods indicated k=2 was the best option, the clusters were not insightful; however, adjusting the k-value to 4 allowed for further definition of clusters. With k=4, the resulting cluster for both NSW.X and NSW.Y with USV2.X and USV2.Y. By performing cluster analysis across all predictors, the paths of UAV 1, UAV1.X and UAV1.Y, and UAV 3, UAV3.X and UAV3.Y, were determined to be most relevant in predicting the NSW team's location.

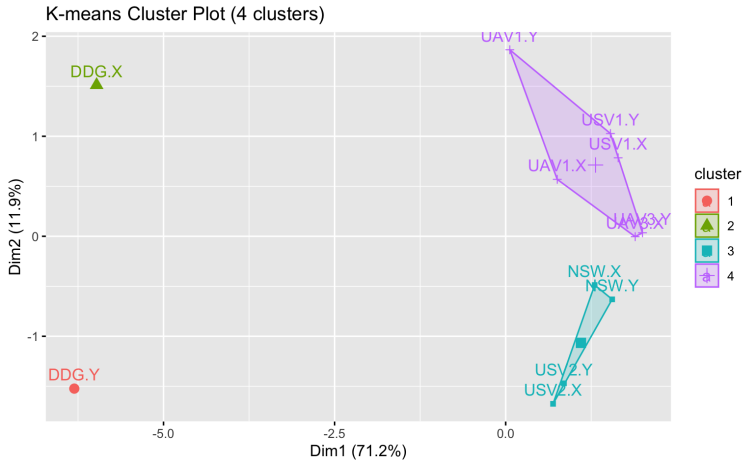

Figure 3.4. K-Means Cluster Analysis on Infiltration Data Set

Figure 3.5 contains two charts, one for the NSW.X variable and the other for NSW.Y, that provide an overview of the results of the cluster analysis. Each of the charts in Figure 3.5 is read in the same manner. Columns represent a variable from the IDS and their relationships to the variable NSW.X or NSW.Y. Chart rows indicate the cluster method used to analyze the relationship. A cell highlighted in green indicates the predictor variable having a relationship with the NSW.X or NSW.Y variable, according to the specific cluster algorithm. Grey cells indicate that the cluster algorithm does not indicate a clustering relationship with the response variable. For example, in the first row of the NSW.X chart

in Figure 3.5, using the Ward (hierarchical) cluster method UAV1.X clusters with NSW.X. Additionally, no other predictor variables cluster with NSW.X using the Ward (hierarchical) cluster method. This single snapshot of the cluster analysis highlights that most of the methods have similar results, NSW.X is most related to UAV1.X, and NSW.Y is most related to UAV3.X, UAV3.Y, USV1.X, and USV1.Y.

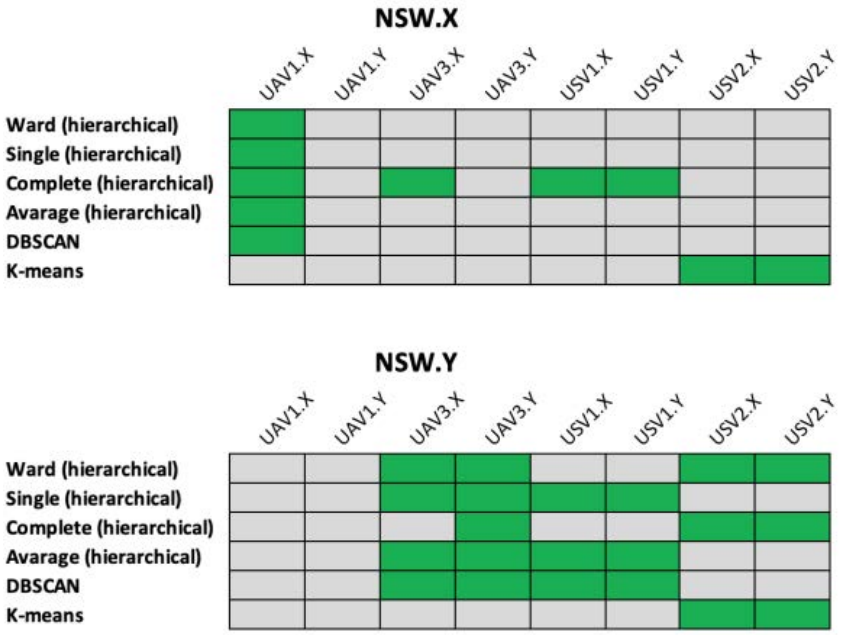

Figure 3.5. Cluster Results for all Methods used on Infiltration Data Set

3.2.2 Supervised Learning

Using the insights from the unsupervised learning methods, the next set of analyses conducted in this thesis focus on supervised learning methods to determine if the values of NSW.X and NSW.Y can be inferred from the other twelve features of the IDS. In supervised learning, a set of observations,$X_1, X_2, ..., X_p$, has an associated response parameter(s), Y_i. The goal of supervised learning is to "fit a model that relates the response to the predictors, to accurately predict the response of future observations, *prediction*, or better understanding the relationship between the response and the predictors, *inference*" (James et al. 2013).

There are many approaches to supervised learning, but this thesis investigates the applicability of various time series methodologies which analyze UxV paths and various regression

techniques to infer the relationship between all of the predictor variables and the position of the NSW team, NSW.X and NSW.Y.

Time Series Analysis

The IDS is simply the coordinate-pairs for five UxVs, one DDG, and the NSW team over 7,005 equally-spaced, time-dependent observations. Because these observations are time-dependent, the order of the data matters, limiting the types of analysis that can be conducted on the data set. Time series data requires special analytical techniques since sequential observations are often correlated and "previous observations of the response variable are often the most important predictive feature for the forecast" (Huddleston and Brown 2018, ch. 7). Time series analysis methods "account for the fact that data points taken over time may have an internal structure such as auto-correlation, trend, or seasonal variation that should be accounted for" (NIST and SEMATECH 2013, ch 6.4).

This thesis investigates the UxV data as univariate time series data, looking at UAV1.X, UAV1.Y, UAV2.X, UAV2.Y, UAV3.X, UAV3.Y, USV1.X, USV1.Y, USV2.X, USV2.Y, DDG.X, and DDG.Y individually, without respect to any possible correlation between predictors. Figure A.1, contained in Appendix A, provides a correlation plot for the data contained in the IDS. This analysis shows that nearly every predictor is positively correlated with each other and further confirms the cluster analysis from Section 3.2.1.

This approach ignores the possibility that a single UxV's coordinates, such as UAV1.X and UAV1.Y, are interdependent. Furthermore, this approach also ignores the fact that the nature of the NCS guarantees that the entire UxV network is interdependent. This approach does provide a better understanding of how each component of a UxV behaved and if that behavior was similar across the NCS. Various time series analysis methods were investigated; however, only two methods were relevant to the IDS. Table 3.4 provides a list of the components normally present in a time series; the IDS only exhibits trend on some of the individual predictors and does not exhibit seasonality or cycle, which caused significant computational errors for the other methods.

Auto-Regressive Integrated Moving Average (ARIMA) is a method of fitting time series data where Auto-Regressive (AR) indicates the response variable is regressed on its own lagged values, Moving Average (MA) is the mean of a given number of recent observations,

Table 3.4. Time Series Components. Source: Yoshida (2020).

Component	Definition
Trend	Long-term (not necessarily linear) increases or decreases in the data; the long-term component of change
Seasonality	Increase and decreases in data with a fixed/known period; often related to weather patterns
Cycle	Data exhibits rises and falls that are not of fixed periods
Noise	The remaining variance in the data after accounting for other components

and Integrated (I) values have been replaced with the differences between their values and the previous values.

Neural Network Auto-Regressive Time Series Model (NNETAR) are feed-forward neural networks fitted with a lagged response variable and a single hidden layer. NNETAR and ARIMA have some similarities; however, NNETAR models have non-linear functions (Hyndman and Khandakar 2008).

Figure 3.6 provides an initial overview of the behavior for six of the predictors across the 5,604 observations from training set of IDS. For all images in 3.6 the x-axis represents time in one-second intervals and the y-axis represents the coordinate value in meters from the origin of the local grid coordinate system developed by Lowry (2020). Over time, UAV1.X, UAV1.Y, UAV3.X, and UAV3.Y possess a decreasing trend, and UAV2.X and UAV2.Y appear constant, but have minor deviations that are not perceptible when maintaining a constant scale across all six variables. Additionally, none of the plots in Figure 3.6 show seasonality or cycles.

Because of the lack of seasonality and cycles, each coordinate of the UxV's data was fitted with a non-seasonal ARIMA model. The `auto.arima()` and `nnetar()` functions, found in the forecast package from R, were applied to calculate the best fitting parameters (Hyndman and Khandakar 2008). Once both models were complete, they were used to forecast each of

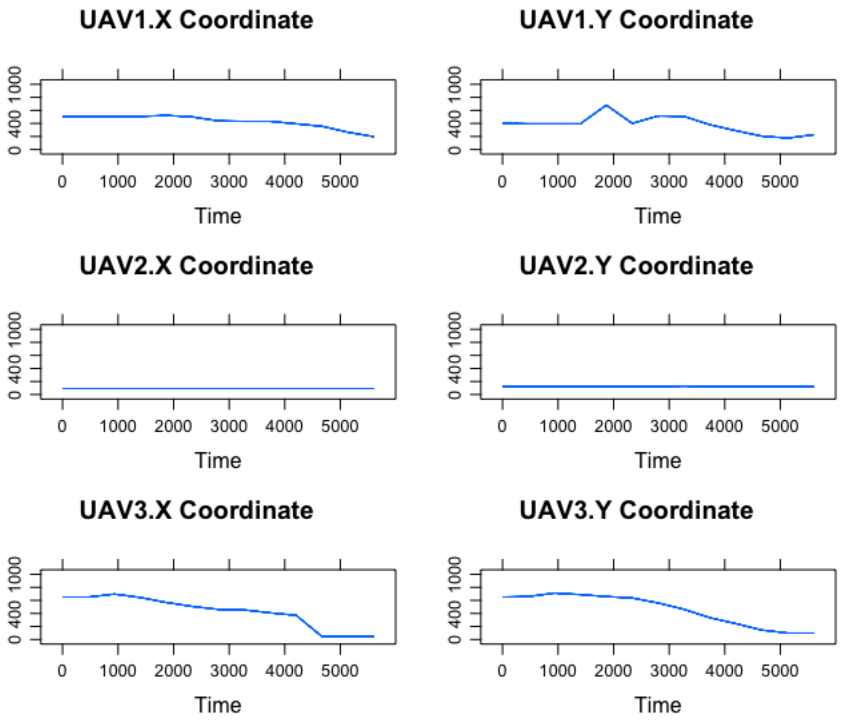

Figure 3.6. Univariate Time Series Analysis of UAV Coordinates

the UxVs coordinates 1,401-time steps, approximately 24 minutes, into the future and then compared to their test set values. Algorithm 1 provides a simple overview of the steps taken to evaluate each model.

Algorithm 1: Time Series Modeling and Forecast Overview

Result: Forecast of time series x-steps into the future
Step 1: Split data into appropriate training/test split;
Step 2: Decompose data to look for time series components;
Step 3: Build model with appropriate time series method/function;
Step 4: Use time series forecast model to forecast x-steps into the future;
Step 5: Compare forecast values to test data set;
Step 6: Calculate performance metrics.

A naive model sets the forecast value to the last observation, whereas a seasonal naive model uses the forecast from the previously observed seasonal period. This data science approach seeks a model that has better predictive power than the naive model (Yoshida 2020). The use of a naive model was investigated initially; however, no intuitive or reasonable model

could be assumed for the IDS.

If a naive model for predicting UxV paths existed, the primary method for evaluating this forecast model's performance would be comparing the Mean Absolute Percentage Error (MAPE) and Mean Absolute Scaled Error (MASE) values of the naive model against the MAPE and MASE values of the forecast model. Therefore, the primary statistic used to evaluate the time series forecast model's performance is the MAPE. While this initial analysis focuses on more traditional performance metrics, Section 3.4 offers a more intuitive performance metric, path deviation, measured with Euclidean Distance in meters. MAPE is a measure of prediction accuracy used in forecast models, where accuracy is measured by the ratio of forecast error divided by the observation. A MAPE closer to zero is good, where a MAPE greater than or equal to 1 indicates 100% or greater forecast error. The MAPE has many desirable qualities, however most important to this analysis is that it is scale-free. Equation 3.1 gives the formula for calculating the MAPE, with N is the sample size, Y_t representing the observed value of Y at time t and F_t representing the predicted value of Y at time t as

$$MAPE = \frac{\sum_{t=1}^{N} (\frac{|Y_t - F_t|}{Y_t})}{N}. \tag{3.1}$$

Figure 3.7 shows the forecasts generated by the ARIMA model for UAV 1; the forecasts for UAV 2 and UAV 3 are not included here because they match the actual values precisely. In both UAV 2 and UAV 3's case, this perfect prediction was the result of the UAVs being stationary for the last hundred observations the model was trained on as well as the entirety of test data set.

Table 3.5 shows the performance metrics for UAV 1 and UAV 3 across both the ARIMA and NNETAR models. An initial review of Table 3.5 shows the MAPE values for the ARIMA model is lower across all components of the IDS, except UAV3.X. However, because the NNETAR model is designed to work with non-linear functions its results are likely to be more robust. Additionally, the NNETAR model does not require parameters to be stationary.

Although this initial analysis of the time series aspect of the UxV data appears to favor the ARIMA models, Section 3.3 investigates more complex models such as RNN and Long

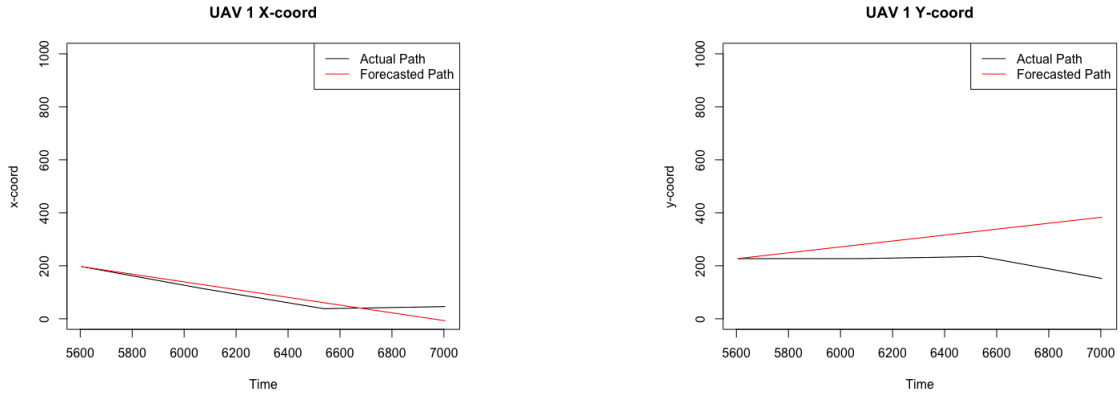

Figure 3.7. UAV1.X and UAV1.Y Predicted Values Compared to Actual Values

Short-Term Memory (LSTM) networks, as well as re-assessing the performance of the ARIMA and NNETAR models when they are used in the time series regression model.

Table 3.5. UxV Time Series Model Performance

Model	UAV1.X	UAV1.Y	UAV2.X	UAV2.Y	UAV3.X	UAV3.Y
ARIMA MAPE	0.285	0.084	0.000	0.004	2.081	0.008
NNETAR MAPE	2.490	0.098	0.000	0.016	0.215	0.332

Prediction Methods

Analyzing and predicting the path of the NSW team, at first glance, is a time series problem. The NSW team's location changes over time and, similar to the UxV positions, if the NSW team's position is observable it can likely be predicted using time series forecasting methods. However, in a real operation, the location of the NSW team is likely not as observable as the UxVs supporting their operation. In addition to the UxV data, the IDS contains the coordinate pairs, NSW.X and NSW.Y, for the NSW team throughout the infiltration phase of the MTX. In an attempt to learn any relationships that may be present between the NSW team and the UxVs, a different approach was taken to analyzing and predicting the path of the ground force.

Cluster analysis from Section 3.2.1 reveals a relationship between UAV1.X and NSW.X as

well as a relationship among UAV3.X, UAV3.Y, and NSW.Y values. This research explores various regression methods to predict the values of NSW.X based on the values of UAV1.X, UAV3.X, USV1.X and NSW.Y based on UAV1.Y, UAV3.Y, and USV1.Y. The combination of NSW.X and NSW.Y makes the coordinate pair for the NSW team's location.

The regression methods used in this section include Multiple Linear Regression (MLR) and Robust Linear Models (RLM). MLR relies on the assumption that the data comes from independent observations, with Normally-distributed errors having equal variance across all observations, to estimate the β coefficients by finding the values of β which minimize the residual sum of square errors. RLM are not affected when the data violates these assumptions and instead fit the models using "iterated re-weighted least squares" functions (Venables and Ripley 2002). Both models followed the simple formula contained in equations 3.2 and 3.3. These regression models follow an approach similar to that of the time series forecast modeling conducted in Section 3.2.2: each of these models only tries to predict a single UxV coordinate instead of trying to model both the x- and y-coordinates simultaneously. The linear regression equations used in this thesis are

$$NSW.X = \beta_{11} \cdot UAV1.X + \beta_{12} \cdot UAV3.X + \beta_{13} \cdot USV1.X, \tag{3.2}$$

and

$$NSW.Y = \beta_{21} \cdot UAV1.Y + \beta_{22} \cdot UAV3.Y + \beta_{23} \cdot USV1.Y, \tag{3.3}$$

where $\beta_{ij} \in \mathbb{R}$, for $i = 1, 2$ and for $i = 1, 2, 3$, are parameters for each model, namely, coefficients of predictors.

Ignoring the correlation between observations and predictors allows for a simple analysis of the relationships between the individual coordinate components. More complex models, discussed in Section 3.3, investigate the interdependence of all predictors. Figure 3.8 shows the diagnostic plots for the linear regression model for NSW.X. The first image in Figure 3.8 is the Residual vs. Fitted plot which shows that the values of NSW.X are not linear. The Normal Q-Q plot indicates that most of the errors appear to be normally distributed; however, there is significant deviation to this in the extreme tails. The final plot in Figure 3.8 is the Scale-Location plot which shows that the square root of the standardized residuals are not evenly distributed This initial analysis of the linear models reveals that the normality

assumptions do not hold, but is used to establish a baseline prediction for later, more complex, models to be compared against. In this initial case, the MLR model performed the best for predictive accuracy of the methods investigated.

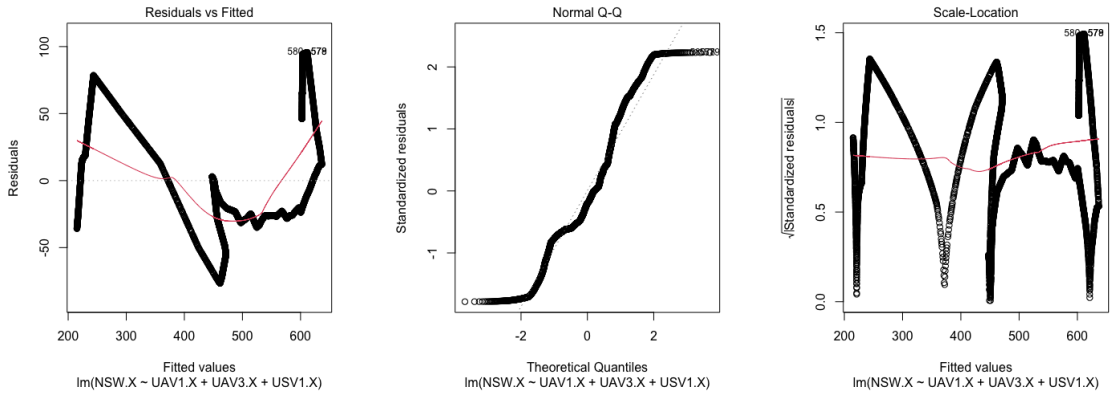

Figure 3.8. Linear Model Diagnostic Plots—NSW.X

Figure 3.9 compares the values of NSW.X and NSW.Y, using the MLR model, to the actual values of NSW.X and NSW.Y. In both predictions, the initial values have an error of approximately 400m; however, the general trend of the data appears to be correctly identified, with the error in the later values of the prediction space decreasing.

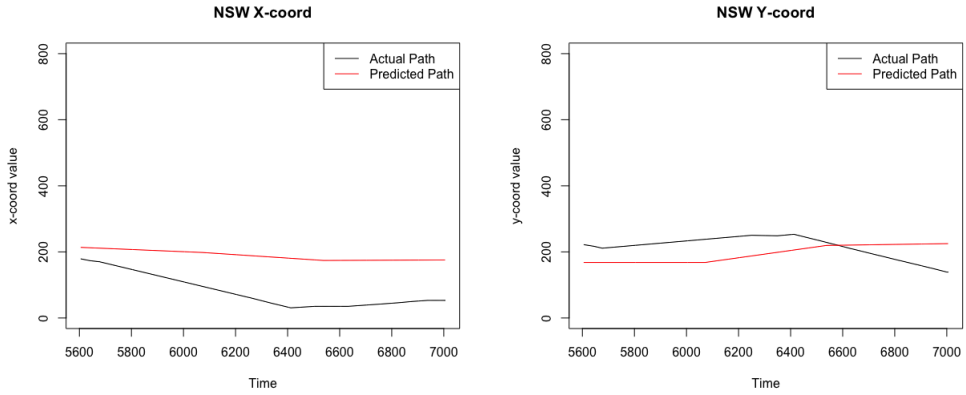

Figure 3.9. Performance of MLR on Predicting NSW Coordinates

A more statistical approach to evaluating the performance of these models is to compare their MAPE. Table 3.6 compares the statistical results from the models used to predict NSW.X and NSW.Y against their actual coordinates. The values in the table are the MAPE

values measured as percentage error, where a MAPE of 0 is a perfect prediction and a MAPE of 1 indicates the predicted value has an error of 100-percent. For example, if the actual value is 100 and the predicted value is 100, the percentage error is 0, whereas if the predicted value is 0 or 200, the percentage error is 1.

Table 3.6. Ground Force Triangulation (Numeric Regression) Performance Matrix

Model	NSW.X	NSW.Y
MLR MAPE	0.0966	0.0704
RLM MAPE	0.0959	0.0696

Table 3.6 shows that the MLR and RLM models perform similarly.

3.3 Neural Networks

The simple time series analysis methods and linear regression models investigated in Section 3.2 provide some of the background and justification for more in-depth analysis and modeling of the UxV time series data and the NSW triangulation problem. This section investigates various neural networks in the aforementioned time series forecast models and linear regression models.

RNN do not immediately appear to be viable options for modeling time series data in automated red cell analysis. The automated red cell requires near-real-time model training and forecasting for the UxV time series data. RNN models are capable of deeper insight into the data as well as multi-variate time series analysis, including addressing correlation and variable inter-dependence. However, the amount of time required to train these models makes their use in an NCS infeasible. DNN have not traditionally been used to solve localization problems; however, this analysis finds that DNN can provide insights into the relationship between the NCS and the supported ground force.

Recurrent Neural Network

RNN and LSTM models provide frameworks to deal with sequential data, such as time series data and were briefly investigated by this research. While these methods are likely to produce

more accurate models with better forecasts, their computational complexity extends beyond what is capable by edge-computing devices or portable computers, including personal laptops, desktops, or portable military servers.

"Recurrent neural networks are designed for sequential data like text sentences, time series, and other discrete sequences, where the input is of the form $\bar{x_1}, ...\bar{x_n}$, where $\bar{x_t}$ is a d-dimensional point received at the time-stamp t" and "allows for the input $\bar{x_t}$ to interact directly with the hidden state created from the inputs at previous time stamps" (Aggarwal 2018, pp. 38-39). Although neural networks are more computationally intensive and not as well understood as the methods discussed in Section 3.2.2, RNN have the added benefit of being able to "learn far more complex models than those obtained with traditional time-series modeling" (Aggarwal 2018, p. 273). RNN models can better model the entire dataset from a multi-variate time series analysis perspective and address the correlation and interdependence present among predictors.

Deep Neural Network

Deep Neural Networks are extremely powerful algorithms, capable of identifying hidden—at least to a human—relationships in data. The use of DNN has increased over the years, with the increase in the availability of large data sets and computational resources. However, Shareef et al. (2007) state that localization problems, such as tracking autonomous robotic movements, have not traditionally been solved with neural networks. This section expands the investigation of the linear regression models used for localization of a ground force, discussed in Section 3.2.2, to neural networks. Neural network models can identify any correlation or interdependence between the$x$- and y-components of a single UxV and any correlation or interdependence among multiple UxVs and the NSW team.

DNN models simplify the modeling of individual UxV interactions and how they can relate to the NSW team's location. The DNN discussed in this section follows a similar scheme to that of the MLR model, except that this model takes all twelve predictors, the$x$- and y-coordinates of all UxVs and the coordinates of the DDG, as the input parameters and uses NSW.X and NSW.Y as the response variables. This approach takes into account any correlation or interdependence that may be present between the individual UxV coordinates or across all UxVs.

Developing the DNN for this research was an iterative process; however, the general framework began with having a single layer for each UxV and starting with the number of nodes equal to the total number of interactions between all twelve components of the time series data. This framework became extremely large, with more than 4,000 nodes in the first layer, and so it was scaled back. The final DNN model sought to balance computational efficiency with performance, containing only five dense layers and starting with 128 nodes. Various activation functions were investigated; however, Table 3.7 describes the final model, which exclusively uses the Scaled Exponential Linear Units (SELU) activation function (Allaire and Chollet 2020). The SELU activation function was chosen since there is no problem with vanishing gradients and it learned faster and had better accuracy than other activation functions investigated. Additionally, the model was compiled with the MAPE loss function.

Table 3.7. Ground Force Triangulation Deep Neural Network

Layer	Output Shape	Activation Function	Num Param
Dense 1	(None, 1, 128)	SELU	1664
Dense 2	(None, 1, 64)	SELU	8256
Dense 3	(None, 1, 32)	SELU	2080
Dense 4	(None, 1, 16)	SELU	528
Dense 5	(None, 1, 2)	SELU	34
Total Params:			12,562

Once a DNN model was built, it was trained on the first 90% of the original training data set; with the remaining 10% from the training data set was used as the validation set. The model was trained with 10 epochs. Although more epochs were investigated, 10 epochs achieved a nice balance between computation efficiency and accuracy without over-fitting. Figure 3.10 provides a simple illustration of how the training of the DNN performed.

With the training of the DNN complete, the original UxV test data was applied to the DNN, and the resulting predicted coordinate pairs were compared to the NSW team's actual coordinates. The model's performance was calculated using the MAPE. The results of this analysis are combined with the results from the previous section and are contained in Table 3.8. Upon initial comparison of the models, the DNN appears to be the worst performer; however, it is the only model which takes into account correlation and interaction across all predictors. Section 3.4 combines these models and presents a time series regression

framework for predicting ground force paths.

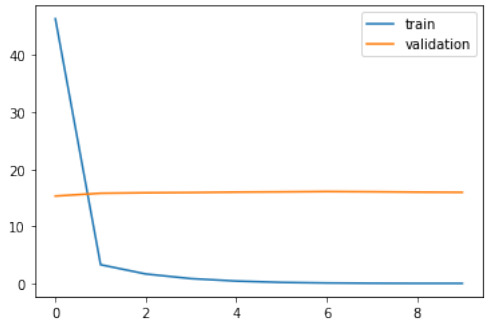

Figure 3.10. DNN Training Performance—Loss Function MAPE

Table 3.8. Ground Force Triangulation, w/ Neural Network, Performance Matrix

Model	NSW.X	NSW.Y
MLR MAPE	0.0966	0.0704
RLM MAPE	0.0959	0.0696
DNN MAPE	3.064	0.2303

3.4 Time Series Regression Models

This section presents a novel framework and model for predicting the location of a ground force supported by an NCS. This framework generally follows a Bayesian hierarchical framework and uses time series regression models to combine the forecast x- and y-coordinates from the UxV forecast models and uses them as the input parameters to a regression model, in this case, the GFT model.

The time series regression model combines the UxV time series forecast model and the GFT model into a singular framework that predicts the future path of a ground force. Figure 3.11 provides a general overview of the data and how it is analyzed. The yellow bar contains the observations of the twelve UxVs over a known time period, which are assumed to be observable by sensors or systems. As time is increased the positions of the UxVs changes; this change is highlighted by the solid horizontal arrows. The red cell model will use time series models to analyze these changes in an attempt to forecast the UxVs future locations,

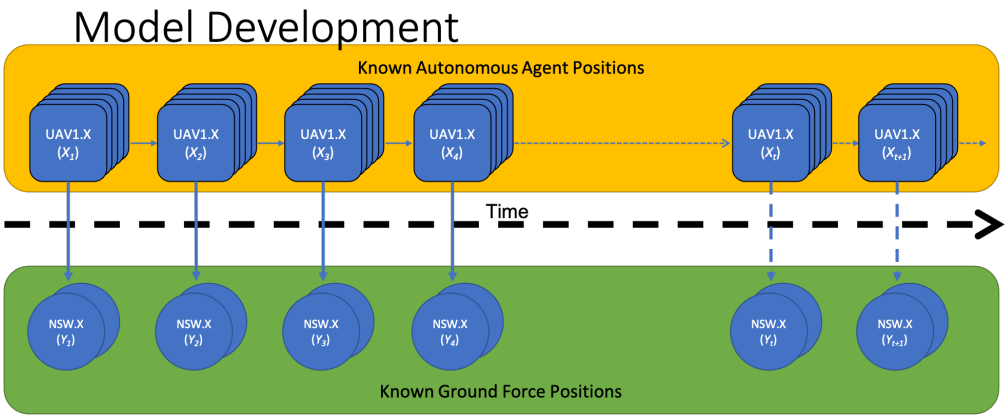

Figure 3.11. Overview of Time Series Regression Model

indicated by the dashed horizontal line. The green bar contains the values of the ground force's positions and while these values are known in the initial analysis, this information is not likely to be known in application. Since the twelve predictors and response variables are known, the red cell model builds a regression model that relates the twelve UxV coordinates to the two NSW coordinates, represented by the solid vertical line. Finally, the red cell model uses the forecasted values of the UxVs as the input to the regression model to predict the future locations of the NSW team, as indicated by the dashed vertical lines.

This approach is not specific to a single implementation of the model and different machine learning algorithms may be applied. In addition to being more robust than the simpler models from previous sections, the time series regression model considers the fact that an adversary is not likely to observe a ground force with the same frequency or certainty as sensors that can detect UxVs. Additionally, an adversary will not have *a priori* knowledge of which UxV most closely relates to the ground force's coordinates, making simpler forms of regression nearly impossible. The time series regression model sets out to solve the problem of uncertainty in past and current ground force positions.

The time series linear regression model discussed in Section 3.4.1 takes observable data, builds a time series forecast model and simultaneously builds a regression model, trains both models on past data observations, and uses the predicted values from the time series forecast model as input parameters for the trained regression model to provide a predicted path for a ground force. Section 3.4.2 uses univariate autoregressive neural networks to

produce the time series forecast model and a deep neural network for the regression model. This type of framework allows for multiple implementations of an Automated Red Cell Analysis; two of these implementations are detailed below.

3.4.1 Time Series Linear Regression Model

The time series linear regression Model formulation follows a simple approach that takes advantage of the more basic models' speed and computational efficiency. This model begins by conducting a univariate time series analysis of the training data set.

UxV Forecast Model: ARIMA

Continuing from the time series forecast modeling conducted in Section 3.2.2, the time series regression model builds twelve ARIMA models. However, instead of using the 5,604 observations of each predictor from the training set, the time series regression models uses fewer observations from the data. Since each observation in the IDS is equal to a single second, the methodology works best when data is sampled at regular intervals. In building this model, sample rates between 1 second and 180 seconds were investigated to simulate not having continuous data from sensors. When the sample rates were low, the predicted location of UAV 2 and UAV3 were nearly perfect; however, this perfect prediction is an artifact of the two UAVs becoming stationary in the last 100 observations of the training set. Additionally, the predicted path of UAV 1 becomes more accurate as sampling is increased. This increased performance is likely because the model The model achieves optimal performance when the data is sampled at 120-second intervals; previous time series forecast models also sampled the data at these rates but found that as sample rates increased, there appeared to be a negative impact on the model's performance. However, in the development of the Time Series Regression models, this increased variability provided more accurate predictions for the ground force's predicted path.

The training data set was then reduced, taking observations every 120 seconds. This reduced training set used 37 observations to build 12 univariate ARIMA models and predict 12 observations, or 24 minutes, into the future. The left panel of Figure 3.12 shows that the predicted path of UAV 1 appears to be very close to the actual path, and UAV 3 does not appear to be accurate in its prediction. In the case of UAV 2, the pink/yellow dot indicates a near-perfect prediction.

Ground Force Triangulation: Linear Regression

Building the GFT linear regression model does not have to occur after the time series forecast model as neither model requires input or outputs from the other. This implementation of the GFT uses the multiple linear regression method explored earlier in this Section 3.2.2. This GFT starts by taking the training data set and using the first 80% of the observations as the training set and the last 20% of the observations as the validation set and then applies the MLR model found in equations 3.2 and 3.3. The overall performance of the GFT model seems to perform well and achieves MAPE values of 0.1368 for NSW.X and 0.09017 for NSW.Y. However, these results are the overall performance metrics on missing data and not the performance of predicted locations.

Time Series Linear Regression Model Realized

The third step in building a time series linear regression model is taking the predicted values of the UxVs from the UxV/ARIMA Forecast Model and applying them as the input parameters for the GFT Linear Regression model. The output of the time series linear regression Model are the predicted values of NSW.X and NSW.Y, or the NSW team's predicted path, 12 observations into the future. This visual comparison of the model output against the actual values helps to better frame which performance metrics can help convey the accuracy and performance of the model. The right panel of Figure 3.12 shows the overall performance of the time series linear regression model as well as the performance statistics, achieving a MAPE of 0.8044 for NSW.X and a 0.2023 for NSW.Y, but reviewing these figures presents a more interesting performance metric, path deviation.

While the traditional metrics discussed provide statistical relevance to the model, path deviation provides a more intuitive metric. Because the model is trying to predict the path of autonomous vehicles and a ground force, understanding the prediction's deviation from the actual path may be of more use to a commander than traditional statistics.

In the case of the time series linear regression model, the predicted NSW team's path has a mean path deviation of 696 meters and a final position prediction deviation of only 326 meters, both measured using Euclidean distance.

Algorithm 2, contained in Appendix A, details the steps taken to implement the initial design of this model. Chapter 4 applies a slightly adjusted version of the algorithm when it

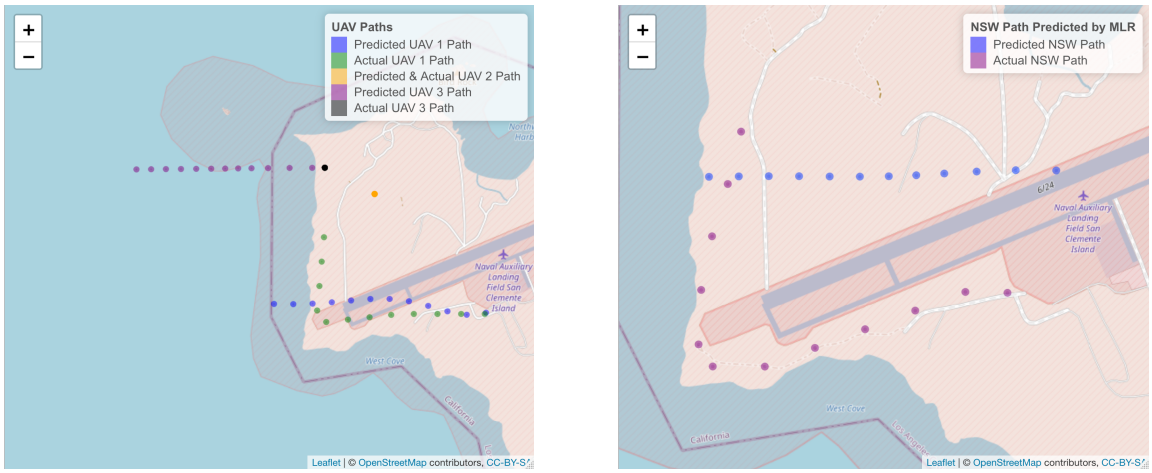

Figure 3.12. Time Series Linear Regression Model Results

is applied to a new data set.

3.4.2 Time Series Neural Network Regression Model

Time series neural network regression model follows the same framework as time series linear regression model; however, this implementation is more complex and takes into account the various correlations and inter-dependencies that may be present in the NCS.

UxV Forecast Model: Neural Net Auto-Regressive

The time series forecast model used in time series neural network regression model follows the NNETAR framework found in the forecast package within R (Hyndman and Khandakar 2008). NNETAR is a feed-forward neural network with a single layer which is fitted with lagged values of the response and is fitted across 20 epochs (Hyndman and Khandakar 2008).

Similar to time series linear regression model, time series neural network regression model benefits from sampling the data at 120 second intervals and also conducts its time series forecast modeling in a univariate fashion. The reduced training set uses 37 observations to build 12 univariate NNETAR models and predict their values 12 observations, or 24 minutes, into the future. Figure 3.13 highlights the predicted paths of UAV 1 and UAV 3, although this model calculates the paths of all UxV in the NCS.

Table 3.9. Time Series Neural Network Regression Model: UAV 1 and UAV 3 Results

Metric	UAV1.X	UAV1.Y	UAV3.X	UAV3.Y
MAPE	0.2852	0.0839	2.0812	0.0082
Path Deviation	1,420m		332m	

Appendix A contains a detailed list of this model's performance metrics.

Ground Force Triangulation: Neural Networks

Time series neural network regression model uses the GFT Neural Network model from Section 3.3. This GFT Neural Network contains five dense layers, each using the SELU activation function, compiled with a MAPE loss function, and using the Adaptive Movement Estimation (Adam) optimizer. However, the time series neural network regression model differs from the other models in that the GFT Neural Network is not trained on a sample of the training data but is instead trained on the first 90% of the training set and the remaining 10 % is used for the validation set, using all 5,604 observations.

Time Series Neural Network Regression Model Realized

The final step in this time series regression is to combine the UxV Forecast NNETAR model and the GFT Neural Network models into a cohesive model. The time series neural network regression model achieves this by taking the 12 univariate NNETAR predictions and feeding them into the GFT model. The outputs from time series neural network regression model are the predicted$x$- and y-coordinates for the NSW team 12 time-steps, or 24 minutes, into the future. Figure 3.13 provides a simple graphic of the predictive accuracy of the time series regression model. The left-side picture of Figure 3.13 helps to visualize the performance of the UxV NNETAR model, and the right-side picture provides the detail of the NSW team's actual location compared to the predicted location from time series neural network regression model. However, a closer look at this figure shows the performance of the time series neural network regression model appears to be quite poor; in fact, the performance is much poorer than the time series linear regression model. The performance statistics for the$x$- and y-coordinates also support the poor performance of the time series neural network regression model, with a MAPE of 1.5792 for predicting the$x$-coordinate of the NSW team

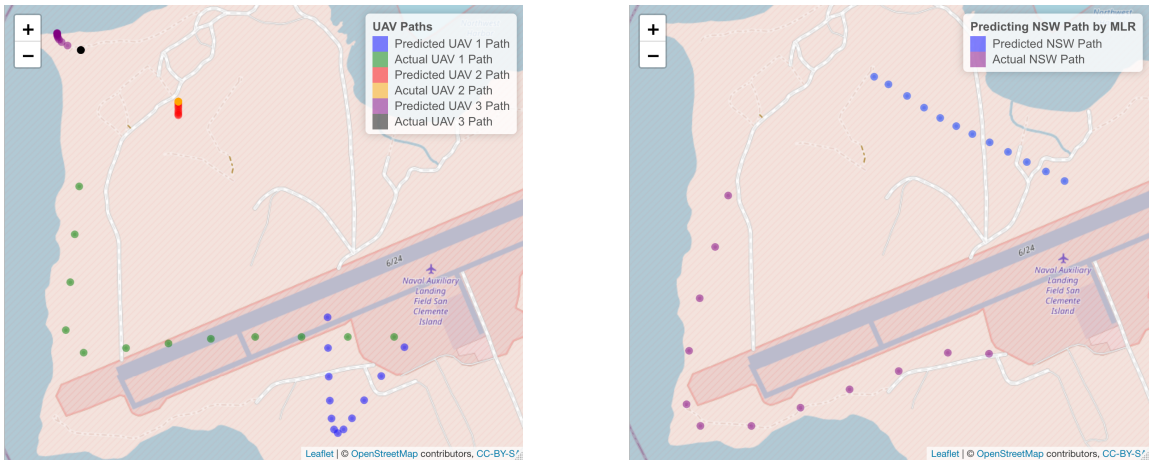

Figure 3.13. Time Series Neural Network Regression Model Results

and a MAPE of 0.453 for predicting the y-coordinate of the NSW team. However, the metric that best describes the accuracy of the prediction is mean path deviation. time series neural network regression model has a mean NSW path deviation of 2,940 meters, nearly five times worse than the mean path deviation from the time series linear regression model.

3.4.3 Analysis of Time Series Regression Models

This chapter explored the many aspects of the IDS in an attempt to learn about any relationships across the NCS that the enemy may exploit. Additionally, this chapter has established the foundation and rationale for applying a time series regression model. In comparing the two resulting time series regression models, the time series linear regression model is the better model for simplicity, accuracy, computational efficiency, and overall run time. These performance characteristics do not address which method best reveals the operational intent of the UxVs and the NSW team. Determining operational intent is more complex than simply making an accurate prediction; it requires qualitative analysis and an understanding of the NSW team's objective to identify which method works better. It is through this qualitative analysis that this research determines that the time series neural network regression model is more accurate in describing the expected behaviors of the entire UxV network and NSW team behavior.

Time series linear regression model provides good statistical predictions for individual UxV coordinates but does not appear to be a good estimator of the general intent of a up on the

intent of the systems. In the case of the time series linear regression model, the predicted paths of UAV 1, UAV 3, and the NSW team are simply straight lines, and this prediction does not necessarily agree with how each UAV behaved before. The time series neural network regression model, however, does a better job of describing the expected behaviors of the entire NCS and NSW team. For UAV 1, the time series neural network regression model predicts the UAV will make a sharp turn north around the seventh prediction, or 14 minutes into the future, and UAV 1 does make a sharp turn north at the eigth time-step, or 16 minutes into the test data set. For UAV 3, the time series neural network regression model predicts that the UAV remains mostly still, which is the exact actions taken by UAV 3. Moreover, for the NSW team, the time series neural network regression model predicts the team is likely to move north-west, and while the NSW team's path was not north-west, its objective was. Another detail discovered through the qualitative analysis is that if the predicted NSW path is adjusted to the correct starting position, the last observation from the training set, the resulting path matches the NSW team's originally planned route towards the objective and ends on the team's objective. The NSW team was diverted around the runway for safety reasons.

3.5 Summary

This chapter uses data science methods to analyze the 2017 MTX data from an adversary's perspective to develop a ground force prediction model based solely on observable data. The time series linear regression Model and time series neural network regression model have qualities that make them desirable, the linear regression model achieves very good statistical metrics and the neural network model appears to predict general system intent. However, for automated red cell analysis to be feasible, the models described in this chapter need to generalize and perform well on different data sets. So, while time series linear regression Model has the better performance statistics on the IDS, time series neural network regression model appears to be better at predicting general behaviors. Therefore, Chapter 4 applies both models to a new unseen data set to evaluate the models' ability to generalize.

CHAPTER 4:
Application, Results and Analysis

This chapter takes the two time series regression models developed in Chapter 3 and applies it to the new, unseen data set. The purpose of applying the two models to a new data set is to determine if the analytical framework and models can generalize to other situations and data sets or if the results from Chapter 3 are unique to that analysis and data set.

To achieve this end, Section 4.1 introduces a new data set to be analyzed. With the data introduced, Section 4.2 explains the implementations of the time series regression models. Section 4.3 defines two measures of effectiveness relevant to the time series regression models and Section 4.4 provides detailed analysis of each model's performance.

4.1 The Unseen Data Set

The Unseen Data Set (UDS) is analyzed from the perspective of an adversary who observes autonomous vehicles and a US Navy ship off its coast. The adversary previously obtained the MTX data sets and has two models which may be able to predict a ground force's path, if one is present.

The UDS was obtained from the research conducted by Lowry (2020) concerning distributed submodularity. Lowry (2020) modified the data set from Chapter 3 to create a new ground force path and objective and simulated UxV positions. Figure 4.1 provides a comparison of the NSW paths from the two data sets. In Figure 4.1, the blue lines represent the path of the NSW team during the training phase of the model, and while there are some similarities between the two data sets, the predicted portions of the NSW path are very different in direction, distance, and general behavior. Table 4.1 describes the data found in the UDS, which consists of 5,175 discrete observations in time, where each observation represents a single second and contains the two-dimensional (X/Y) positions coordinates for a NSW team, one DDG, three UAVs, USVs, and two UUVs. Although UUVs are included in the data set, they are not a part of this analysis as they would not be observable to sensors. This new, UDS has a different ground force objective and path, and therefore, the distributed submodular framework creates a different network configuration for the NCS.

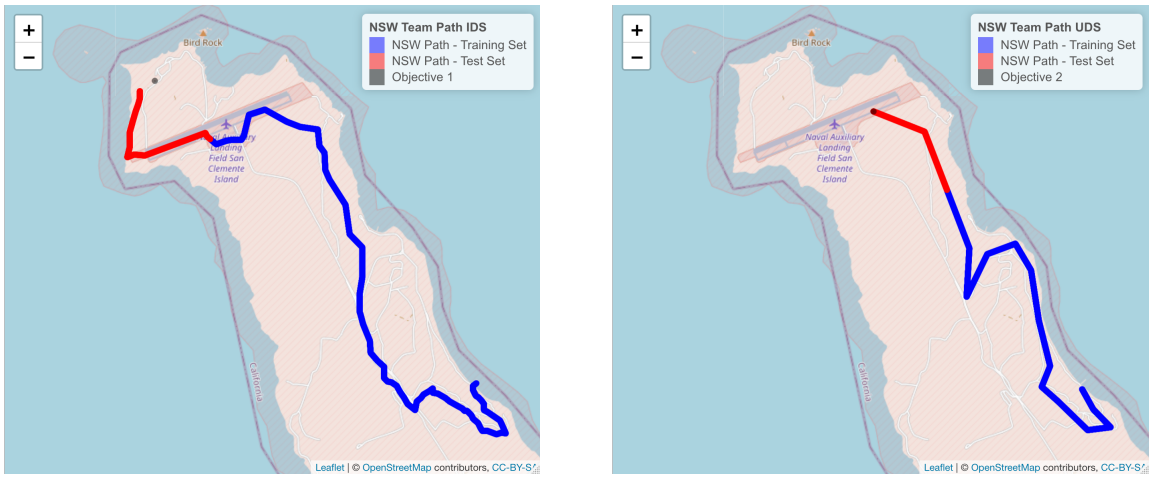

Figure 4.1. Comparison of NSW Paths—IDS vs. UDS

The main benefit of evaluating the Time Series Regression models on the UDS is that the data is generated using the same NCS framework, which results in the data format matching the data format used in the IDS. The UDS following the same format as the IDS allows for easier analysis of the models since the only parameter that had to be changed was the name of the data file to read in. Moreover, the model's results have identical naming of the variables, charts, statistics. Variable names throughout this Chapter will match those used in Chapter 3; however, these variables will only reference the output from the model's application to the UDS.

4.2 Time Series Regression Models Applied

This section details how the two time series regression models are applied to the UDS. Because this chapter assumes the adversary will not be capable of observing the ground force at regular time intervals, the application of both models requires the reuse of each model's respective GFT model. Although the models discussed here assume the ground force is hidden to an adversary's sensors, this does not preclude the adversary from occasionally receiving reports of the ground force's location and updating their prior probabilities, which could greatly increase the accuracy of the models. Figure 4.2 provides a general overview of how the model is applied when the ground force's location is unknown.

Since the UDS follows the same data structure as the IDS, there were no data manipulation

Table 4.1. Overview of Simulated Unseen Data Set

Data Name	Num Obs	Description
UAV1.X	5,175	x-coordinate of UAV 1
UAV1.Y	5,175	y-coordinate of UAV 1
UAV2.X	5,175	x-coordinate of UAV 2
UAV2.Y	5,175	y-coordinate of UAV 2
UAV3.X	5,175	x-coordinate of UAV 3
UAV3.Y	5,175	y-coordinate of UAV 3
USV1.X	5,175	x-coordinate of UAV 1
USV1.Y	5,175	y-coordinate of USV 1
USV2.X	5,175	x-coordinate of USV 2
USV2.Y	5,175	y-coordinate of USV 2
DDG.X	5,175	x-coordinate of DDG
DDG.Y	5,175	y-coordinate of DDG
NSW.X	5,175	x-coordinate of NSW Team
NSW.Y	5,175	y-coordinate of NSW Team

or formatting changes required to conduct the analysis in this chapter. The UDS is divided into a training set and a testing set, with 4,140 observations in the training set and 1,035 observations in the testing set.

Figure 4.2 provides an updated overview of the data and how it is analyzed in this chapter. The yellow bar contains the observations of the twelve UxVs over a known time period, which are assumed to be observable by sensors or systems. As time is increased the positions of the UxVs changes; this change is highlighted by the solid horizontal arrows. The red cell model will use time series models to analyze these changes in an attempt to forecast the UxVs future locations, indicated by the dashed horizontal line. The orange bar contains the future—unknown—values of the UxVs. The green bar contains the future—unknown—values of the ground force's positions. Finally, this application of red cell analysis uses the forecasted values of the UxVs as the input to the regression model, via transfer learning, to predict the future locations of the NSW team, as indicated by the dashed vertical lines.

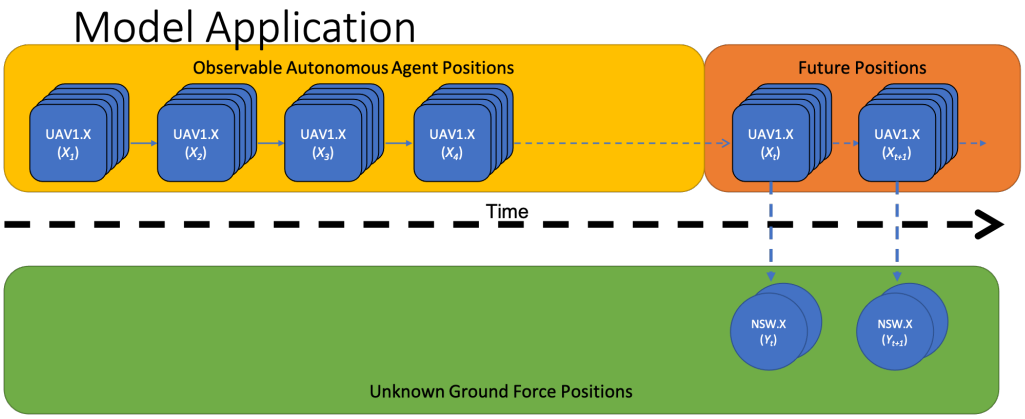

Figure 4.2. Time Series Regression Model—Applied

4.2.1 Time Series Linear Regression Model

The Time Series Linear Regression Model builds twelve univariate ARIMA models. Following the same methodology established in Chapter 3, the time series regression models sample the data every 120 seconds. The outputs from the twelve univariate ARIMA models are combined and used as the input values to the linear regression model contained in Equations 3.2 and 3.3,

$$NSW.X = \beta_{11} \cdot UAV1.X + \beta_{12} \cdot UAV3.X + \beta_{13} \cdot USV1.X,$$

and

$$NSW.Y = \beta_{21} \cdot UAV1.Y + \beta_{22} \cdot UAV3.Y + \beta_{23} \cdot USV1.Y.$$

Section 4.3 and Section 4.4 provide detailed results and analysis of the applied Time Series Linear Regression Model.

4.2.2 Time Series Neural Network Model

The Time Series Neural Network Model builds twelve univariate NNETAR models, and follows the NNETAR framework discussed in Section 3.4.2 and found in the forecast package within R (Hyndman and Khandakar 2008). Following the same methodology established in Chapter 3 and in Section 4.2.1, the Time Series Neural Network Regression Model samples the data every 120 seconds. The output from the twelve univariate NNETAR models are

combined and used as the input values to trained GFT Neural Network Model, described in Section 3.3. Table 3.7 provides the detailed description of the GFT Neural Network. Section 4.3 and Section 4.4 provide detailed results and analysis of the applied Time Series Neural Network Regression Model.

4.3 Methods of Evaluation

The overall performance of the ensemble models will use the metrics discussed in Chapter 3, MAPE and Path Deviation. These metrics allow for different aspects of analysis. The first of these is statistical accuracy, which uses statistical tools to measure how well a given model predicts the individual values of the predictors, UAV1.X, UAV1.Y, UAV2.X, UAV2.Y, UAV3.X, UAV3.Y, USV1.X, USV1.Y, USV2.X, USV2.Y, DDG.X, and DDG.Y, and the response variables, NSW.X and NSW.Y. The second looks at the data as coordinate pairs and compares the predicted coordinate pairs to the actual coordinate pairs, using Euclidean distance measured in meters.

4.3.1 Statistical Accuracy

Statistical accuracy was instrumental in developing the ensemble methodology, and these metrics is to be applied here. The purpose of these statistically-based metrics will provide insight into the model's accuracy and should continue to be used to compare other forecasted results. Because this thesis is the first step in Automating Red Cell Analysis for Mobile Networked Control Systems, follow-on work, discussed in Chapter 6, should focus on using statistical metrics, along with classification algorithms, to create a feedback loop that can tell a ground force commander how much operational intent is being telegraphed by the NCS.

Chapter 3, and this Chapter, rely on the MAPE as its primary statistical tool. MAPE is a measure of prediction accuracy used in forecast models, where accuracy is measured by the ratio of forecast error to observation. However, this thesis establishes the framework for using multiple univariate ARIMA models as the naive model for predicting individual UxV coordinates, which can then be compared against other model's predictions. The MASE is complementary to the MAPE, in that it is scale-free, but also provides a metric for comparing two different models to the actual value simultaneously and is a good sense of

predictive power. Equation 4.1 provides the formula for calculating the MASE, where Y_t represents the actual value of the t^{th} observation, F_t is the value predicted by model F, G_t is the prediction from the naive model, and N is the number of observations:

$$MASE = \frac{\sum_{t=1}^{N}(|\frac{Y_t - F_t}{Y_t - G_t}|)}{N}. \tag{4.1}$$

A MASE with a value of less than one indicates that the new model—in the equation above, this is model F—has better predictive power than the naive model. In contrast, a MASE equal to 1 indicates the naive model and new model have the same predictive power. A MASE greater than one indicates the naive model has more predictive power than the new forecast model.

4.3.2 Accuracy of Path Prediction

While the statistical metrics provide insight into how well each model can predict a single value of a component of the data, path deviation is a much more intuitive metric when talking about predicting the path of one or more UxVs or the NSW team. Path deviation provides a measure of how accurate a predicted coordinate pair is compared to the actual coordinate pair. For example, if an object is at position (1,1), but the model predicts the coordinate pair value is (4,5), the associated MAPE values for the x- and y-coordinates is 3 and 4, respectively. A more intuitive measure of accuracy is the Euclidean distance between the actual point and the predicted point. In this case, the path deviation for our example problem is five distance units. Figure 4.3 provides a simple example of path deviation.

4.4 Analysis of Time Series Regression Models

Because the UDS is designed to test if the ensemble method is generalizable, the data was not evaluated in the same manner as the IDS. The analysis does not look at the data using unsupervised learning methods or other statistical tools. Instead, it blindly applies the two models to the UDS and evaluates performance based on the metrics discussed in Section 4.3.

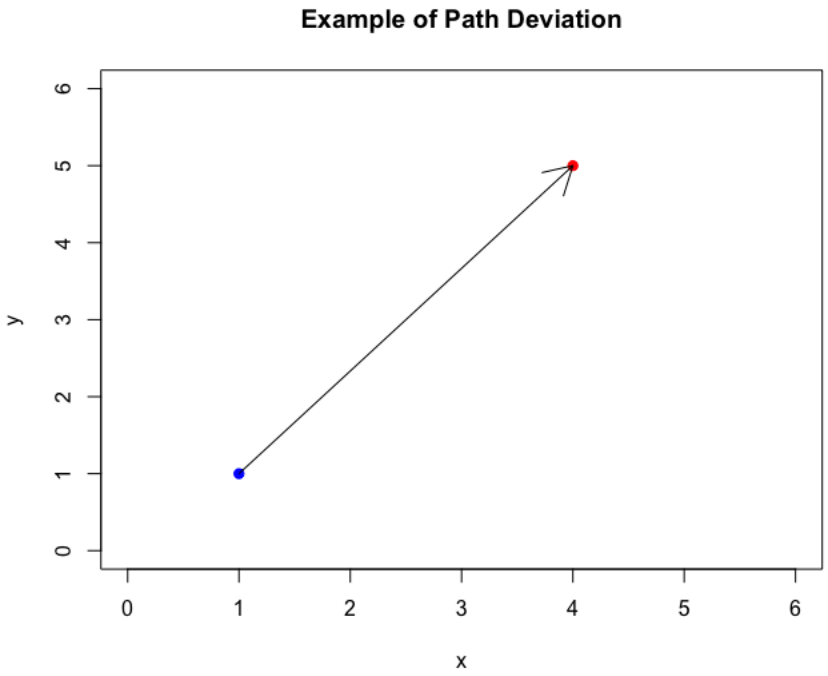

Figure 4.3. Example of Path Deviation Metric

4.4.1 Time Series Linear Regression Model

In Chapter 3, time was taken to evaluate the various parameters for correlation, clustering, and trend to craft the specific formulas for the ground force triangulation model. This section simply applies the Time Series Linear Regression Model to the UDS and provides the analysis of its outputs.

From the statistical perspective, the application of the Time Series Linear Regression Model is not impressive. Table 4.2 provides Time Series Linear Regression Model's performance matrix, and while some of the models have good statistical prediction metrics, such as UAV2.X, UAV2.Y, and UAV3.X, the forecasted values for the other predictors have very high MAPE scores.

The graphical outputs for the Time Series Linear Regression Model are contained in Figure 4.4 and help highlight the second metric, path deviation. Figure 4.4 provides a stark contrast to the performance Time Series Linear Regression Model had on the IDS.

Table 4.2. Time Series Linear Regression Model: UxV Forecast Performance—Applied

Metric	UAV1.X	UAV1.Y	UAV2.X	UAV2.Y	UAV3.X	UAV3.Y
MAPE	0.1428	0.2931	0	0	0.0516	0.2642

UxV Forecast Model Analysis

The left-hand image of Figure 4.4 compares the predicted path of UAV1, the actual path of UAV1, the predicted path of UAV3, and the actual path of UAV3. The predicted path of UAV1 is a straight line, and while this prediction is in the same general direction that UAV1 takes, it fails to predict any changes in direction that UAV1 may take and overshoots the target. The initial path deviation for the first four minutes' worth of predictions is less than 150 meters; however, because the prediction fails to identify any changes in direction or to reach the objective, it achieves an average path deviation of 684 meters. The predicted path of UAV3 does appear to pick up on some of the planned movements of UAV3; however, in terms of path prediction, it has a similar performance of UAV1, having the initial 6-minutes' worth of predictions within 161 meters, but as the prediction space increased, the path deviation increases to an average path deviation of 631 meters.

Ground Force Triangulation Analysis

The right-hand image of Figure 4.4 compares the predicted path of the NSW team to the actual path of the NSW team. This graph clearly shows that the predicted path of the NSW is in the ocean. The prediction begins being more than 500 meters off and only worsens, achieving an average path deviation of more than 1000 meters. In addition to having inaccurate predictions for UxV behavior, the associated MAPE for the NSW.X is 0.4796, and NSW.Y is 0.2207.

Time Series Linear Regression Model Analysis

We see that the Time Series Linear Regression Model is not able to determine the NSW's operational intent. While there are many possibilities as to why this could occur, it is important to recall that Time Series Linear Regression Model was developed deliberately based on cluster analysis from the IDS. The multiple linear regression model applied to this

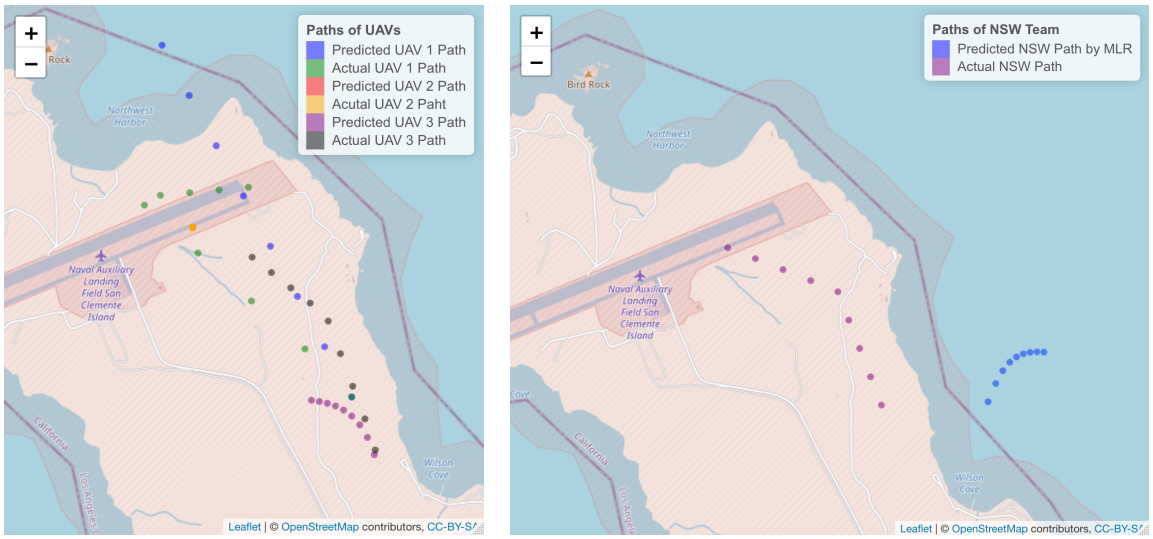

Figure 4.4. Time Series Linear Regression Model Results—Unseen Data Set

Table 4.3. Time Series Neural Network Regression Model Time-Series Performance—Applied

Metric	UAV1.X	UAV1.Y	UAV2.X	UAV2.Y	UAV3.X	UAV3.Y
MAPE	0.1302	0.2677	0	0	0.0935	0.0331

data set was trained on the data from the IDS and the β coefficients that best fit the IDS clearly do not fit the UDS. Since the NCS is capable of dynamically selecting and repositioning UxVs based on which is most related to another, in terms of sensing and communication, it is likely that the combination of predictors developed in Chapter 3 changed importance and behavior, and that a different combination of predictors would produce better results.

4.4.2 Time Series Neural Network Regression Model

The Time Series Neural Network Regression Model is expected to more robust in its performance since the ground force triangulation model is a DNN. This section will follow a structure similar to that of Section 4.4.1 in that it will simply apply the Time Series Neural Network Regression Model to the UDS and provide an analysis of its outputs.

The statistical analysis of Time Series Neural Network Regression Model's performance appears to be much better than Time Series Linear Regression Model.

UxV Forecast Model Analysis

The left-hand image of Figure 4.5 compares the predicted paths of UAV1, UAV2, and UAV3 against their actual paths. An initial visual comparison of the predictions against their actual paths seems to be less than impressive. However, the model does appear to determine that UAV1 will move in a more westerly direction and that UAV3 will continue north with a slight adjustment to the west. In terms of the statistical measures of accuracy, the model achieves low MAPE values for UAV2.X, UAV2.Y, and UAV3.X, and UAV3.Y However, the model does not appear to accurately predict the values of UAV1.X and UAV1.Y, since both have MAPE values of greater than 0.1. UAV1 and UAV3 begin with very accurate predictions, with the mean path deviation for the first 6 minutes being 55 meters and 26 meters, respectively. However, this level of performance quickly disappears in both cases as the prediction horizon increases, with UAV1 having a mean path deviation of 611 meters over a 24-minute prediction horizon and UAV3 having a mean path deviation of 346 meters over the same prediction horizon.

Ground Force Triangulation Analysis

The right-hand image of Figure 4.5 compares the NSW team's predicted and actual paths. What is most evident in this image is that the predicted path and the actual path are nearly identical. Over the 24-minute prediction horizon, the mean path deviation is only 39 meters; what is more impressive is that the largest path deviation occurs in the middle of the prediction space and is only 75 meters. The predicted NSW locations that occur after this mid-point converge back towards the NSW team's actual path. In addition to having very accurate path predictions for NSW behavior, the associated MAPE for the NSW.X is 0.0117 and NSW.Y is 0.0112.

Time Series Neural Network Regression Model Analysis

In Chapter 3, the time series neural network regression model had poor statistical results but did provide insight into what the NSW team planned on doing or operational intent. As shown in this section, the time series neural network regression model continues to highlight the NSW's operational intent and even achieves good performance metrics. In addition to NSW.X and NSW.Y having very low MAPE and path deviation values, the time series neural network regression model even identifies when the NSW team will change

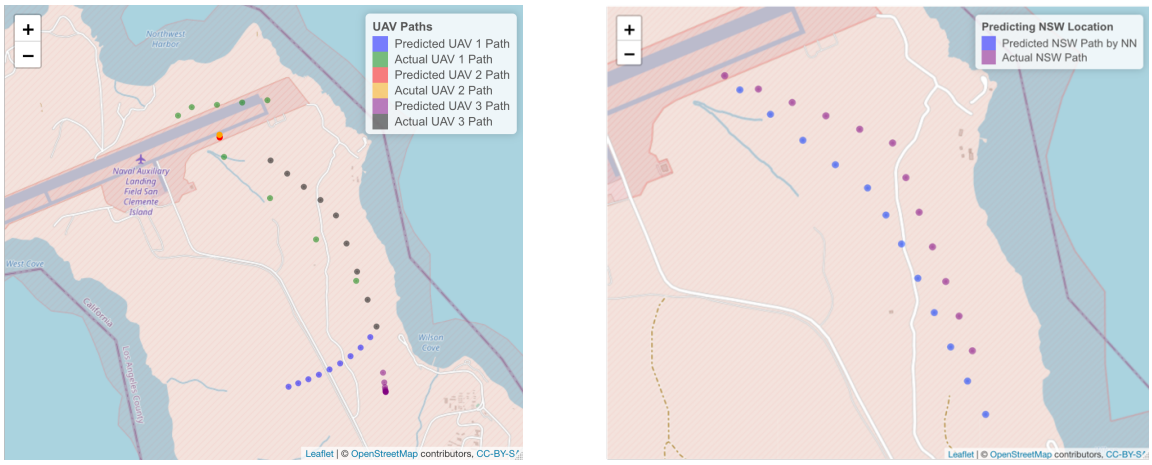

Figure 4.5. Time Series Neural Network Regression Model Results — Unseen Data Set

direction and gets the path adjustment nearly exactly right.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Implementation Analysis

Automated Red Cell analysis is designed to determine if unmanned systems' behavior telegraphs the operational intent of the ground force it supports. There are various implementations of NCS, as well as new forms of control being developed. Lowry (2020) recommends the use of a Delay-Disruption Tolerant Network (DTN) to loosen the communications constraint requirements of an NCS. DTN is a computer networking model that allows nodes from a system to be disconnected for a specified period of time (Tzinis 2020). Research into DTN originally focused on planetary communications systems that allowed NASA to communicate and control satellites over extreme distances. More recent research from Rohrer and Xie (2013)—conducted at NPS—has extended DTN uses to vehicular networking.

Therefore, to properly incorporate Automated Red Cell analysis into an NCS, the methodology needs to scale and be implementable on a variety of computing platforms. The NCS examined in this thesis uses Distributed Submodularity to identify the best positions for each UxV; however, this approach still maintains the requirement that the NCS be fully connected. This requirement allows Automated Red Cell analysis to take advantage of this connectedness and implement its models on more powerful computing devices. Future implementations of NCS that employ DTN relax the requirement to have a fully connected network, allowing autonomous agents to be disconnected from the system for a specified period of time. The disconnected autonomous agents will not be able to leverage the connected network to implement Automated Red Cell analysis on a dedicated machine and will instead rely on the on-board computing power of the UxV.

In addition to the computational requirement of Automated Red Cell Analysis, as NCS methodologies transition from unmanned-centralized control to autonomous or semi-autonomous control, additional computational power will be required on-board individual UxVs to handle the computationally complex algorithms required for autonomous control. The specific UxVs used in the MTX were the ScanEagle UAV, SeaFox USV, and REMUS 100 UUV. The amount of space, power, and weight that can be added to each

of these UxVs varies, and therefore, the methods for implementing Automated Red Cell analysis must vary. This chapter provides possible implementations across three computing devices; microcomputers or edge computing devices, mobile computing platforms, and high-performance computers.

This thesis does not investigate the use of High-Performance Computing since High Performance Computers (HPC) are not currently components of the NCS. However, if future implementations of NCS do contain HPC capabilities, additional complexity can be added to the methodology.

5.1 Microcomputer/Edge Computing

There is not a single agreed-on definition for "edge computing"; however, edge computing can be thought of as conducting computation or analysis on an individual node of a connected network. There are many examples of edge computing devices, but the most common is the smartphone. An iPhone or Android phone can conduct complex tasks across its connected network and conducts some tasks and analysis on the device itself. Some of the benefits of conducting computation and analysis at an edge device include privacy and security as well as reliability (Merenda et al. 2020).

An example of a micro-computer is a Raspberry Pi 4, pictured in Figure 5.1. A Raspberry Pi is a low-cost, credit card-sized computer that was designed for teaching computer science and robotics, but because of its low cost, versatility, and modular design is popular among electronic hobbyists (Pi 2021). Current versions of the Raspberry Pi 4 Model B contain a quad-core ARM processor and can be purchased with 2, 4, or 8 gigabytes of RAM and contain various USB and GPIO interfaces (Pi 2021).

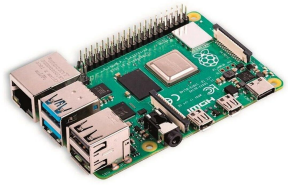

Figure 5.1. Raspberry Pi 4 (Edge Computing Device). Source: RaspberryPi.org (2021).

The Raspberry Pi 4, and many of the other micro-computing platforms on the market today,

possesses the required hardware and computational capacity to control a single UxV and conduct Automated Red Cell Analysis. However, a single board may not be capable of conducting both actions simultaneously.

Of the three UxVs contained as part of the NCS, the component with the most constraints is likely to be UAV. The ScanEagle UAV does not have a native, on-board computer capable of autonomous control or Autonomous Red Cell analysis; however, ScanEagle does have a payload capacity of approximately 11 pounds, which includes any sensors, batteries, or additional cargo, which can support a micro-computing device (Insitu 2021). The NPS CAVR lab has already addressed this shortcoming and integrated two micro-computers into its ScanEagle platforms. These micro-computers provide the ScanEagle the additional computational resources needed to conduct autonomous control of the UAV and may also support native automated red cell analysis. Therefore, the framework and models developed in this Thesis must be capable of operating on computers with fewever computational resources.

A version of Automated Red Cell analysis was implemented on-board a Raspberry Pi 4 Model B to see how well the methodology, in its current state, functions when given a live feed of data. This section documents some of the issues with this implementation and process for developing this test environment; however, it does not cover details or provides quantitative analysis of this output. A current limitation to deploying Automated Red Cell Analysis to micro-computers is processor architecture; many machine learning packages will not function on the ARM-based processors, limiting the choices of algorithms that can be implemented. While a micro-computer can run the ROS and python programming languages, many machine learning applications are not compatible with ARM-based processors.

Time Series

Some of the models built and discussed in this thesis take advantage of specific packages or functions resident within R. While micro-computers are capable of operating statistical software such as R, if these systems are employed in coordination with UxVs, they may be limited to a single-versatile programming language, such as Python. Robot Operating Systems (ROS) is an open-source robotics middleware suite written in C++ and Python, which allows for easier integration between the UxV software, NCS, and Automated Red

Cell. Limiting the time series models to python-based algorithms precludes the use of the auto.arima or nnetar functions discussed previously; however, the pmdarima offers a comparable implementation. (Smith et al. 2017).

Ground Force Triangulation

The multiple linear regression models, discussed as part of the Time Series Linear Regression Model, can run on a micro-computer since they can be solved using simple matrix algebra. However, the more complex implementations of the DNN require more work to be implemented on-board a micro-computer.

TensorFlow and Keras can run, based on processor speed and RAM, limited machine learning algorithms on an ARM-based micro-computer. While the DNN built in this thesis is likely beyond the scope of what can be trained on a micro-computer, an already trained DNN can be used to make predictions on-board a UxV through transfer learning.

5.2 Mobile Computing Platform

The models and methodology presented by this thesis were built on a personal laptop, with a 2.6 GHz 6-Core Intel Core i7 processor, 16 GB 2667 MHz DDR4 RAM, and an Intel UHD Graphics 630 1536 MB graphics processor. Moreover, the approach to model and methodology development was intentional; since current constructs of NCS require a fully connected network to operate, a control station would likely have similar processing power. Table 5.1 details the processing times for each stage of the models. The Deep Neural Network used in this thesis was thoroughly discussed in Section 3.3 and is not discussed in this section.

Table 5.1. Mobile Computing—Model Processing Times (seconds)

	1 Second Interval		120 Second Intervals	
Model	Training Time	Prediction Time	Training Time	Prediction Time
auto.arima	1.787	0.049	1.013	0.041
nnetar	33.845	15.079	0.378	0.345
mlr	0.007	0.004	0.010	0.003
dnn	230.841	0.369	N/A	0.100

A key component to building the Automated Red Cell methodology was finding a time series forecast model capable of taking in several minutes' worth of raw observations and providing a near-real-time forecast for X-number of observations into the future. Because the time-series model must take in dynamic data and predict future locations without prior knowledge of the future, it can be considered an online optimization problem. The desired methodology must be able to take in the dynamic data, build an appropriate time-series model, train the model using the collected data, and forecast future values, all in near-real-time. This thesis focused on using R's auto.arima and nnetar functions from the forecast package Hyndman and Khandakar (2008) since these functions could train the model and make forecasts in less than 2 seconds. The auto.arima and nnetar functions could train the model in very similar times when the training data set was sampled at rates greater than every 60 seconds. However, when the training data were sampled at rates less than 60 seconds between observations, there was a substantial difference in the time required to train the nnetar model.

Later iterations of Automated Red Cell Analysis may see more accurate time-series model predictions if multivariate LSTM neural networks are incorporated into the methodology. The use of RNN and LSTM were initially explored; however, because of the amount of time required to train the LSTM model, these methods were not feasible solutions for implementing Automated Red Cell analysis into an NCS.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 6:
Conclusion

This chapter discusses the findings of this thesis, Section 6.1, and identifies areas where this research can be implemented and further work conducted, Section 6.2.

6.1 Discussion

This thesis continues work from Wachlin (2018) and Lowry (2020) regarding a graph-theoretic framework for real-time control of a network of UxVs and Distributed Submodularity for finding near-optimal position recommendations for an NCS agents in real-time.

Automated Red Cell analysis provides a framework and methodology to determine if an NCS supporting a human component or ground force is telegraphing or predicting the ground force's operational intent. Automated Red Cell analysis is not held to a single model but is flexible and can be applied to nearly any autonomous system. In the case of the NCS implementation researched in this thesis, the Automated Red Cell model needs to predict the future locations of the ground force but is limited in its ability to sense or see the ground force at any given time. Therefore the model must use data that is correlated to the ground force, and using data acquired from past exercises, builds a model which relates the position of the ground force to the UxVs supporting them. The overall model consists of a Time Series Forecast Model to forecast the future UxV positions, a trained Ground-Force Triangulation Model, and a time series regression model which combines the forecasted values from the Time-Series Forecast Model as the inputs to the GFT model to predict the path of the ground force X number of steps in the future.

The Automated Red Cell methodology is a novel framework that quantifies how autonomous agents may telegraph or predict a ground force's operational intent. The implications of this methodology and analysis are far-reaching as the DOD begins to focus on competing with near-peer adversaries in the Pacific, and the Marine Corps identifies the need for reconnaissance and counter-reconnaissance capabilities when conducting operations within the "Weapons' Engagement Zone."

While this thesis presents a novel application of time series regression models, there are limitations to this research. These limitations do not impact the results of this analysis; however, it is prudent they are acknowledged and discussed. The DNN developed and trained in Chapter 3 and then applied in Chapter 4 is only trained on 5,604 observations. While the number of observations is small—when compared to the training required for accurate convolutional neural networks—there were no other data sets available to continue training the model. Additionally, the NSW paths from both the IDS and UDS are similar, which is briefly discussed in Chapter 4. Figure 4.1 highlights the NSW path from both data sets and highlights the training set observations in blue and the testing set observations in red. In both cases the NSW team maneuvers along the east coast of SCI, in nearly identical positions. Following insertion, the team's path follows several roadways, moving north towards Naval Auxiliary Landing Field. And while the NSW team's path during training set the IDS is significantly different than its path in the UDS, there are enough similarities in the general ground force behavior that could be mitigated with more data sets.

6.2 Future Work

A natural extension to this research is to apply the methodology in a real-world test opportunity. Implementing the Automated Red Cell methodology into a live NCS and operating it at the Joint Interagency Field Experimentation Program will provide a stable environment to test the robustness of the ground force triangulation model as well as the generalizability of the Automated Red Cell methodology.

The frameworks and models discussed in this thesis provide the foundation for incorporating a feedback loop, and potentially deception, into an NCS. Figure 6.1 proposes what a possible feedback loop may look like. The NCS generates path recommendations and feeds the recommended paths to the automated red cell module. The NCS system operator or ground force commander provides a time horizon and level of certainty, and possibly path deviation, to the red cell module. The red cell analysis module analyzes the path recommendations and if, over the given time horizon, the predicted ground force path is within the predefined level of certainty, the automated red cell analysis module notifies the human in the loop or the NCS generates a new configuration and paths. When the automated red cell analysis indicates that the UxV NCS has high predictability or clearly identifies a ground force's operational intent, the NCS can notify a ground force commander who can then incorporate

deception and show that the deception plan is difficult to track.

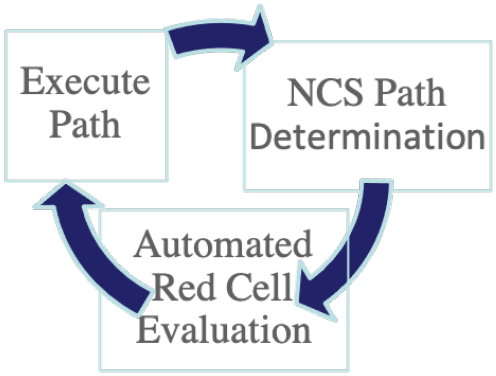

Figure 6.1. Automated Red Cell Analysis Feedback Loop

This thesis takes a data science approach to conducting red cell analysis of a mobile NCS; however, this type of analysis is not, and should not, be limited to use in just NSW UxV tactical operations. The red cell analysis methodology investigated in this thesis has the potential to change many aspects of military operational planning, including operational and strategic level planning and wargaming. The United States Marine Corps is in the process of drastic force design changes in order to focus on "fulfilling the role as the nation's naval expeditionary force-in-readiness" (United States Marine Corps 2021, p. 2). USMC Force Design 2030 highlights the Marine Corps's need for forward deployed naval expeditionary forces, sensory networks to detect adversary targets and command and control systems that remain functional as forces move, communicate and act, as well as the need to conduct reconnaissance and counter-reconnaissance (United States Marine Corps 2021). Automating red cell analysis and further developing it to work in and with other systems will be instrumental in accelerating reconnaissance constructs Marines will need as the "Stand-in force."

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A:
Time Series Regression Model Development

A.1 Assumptions and Correlations

This appendix provides a more detailed information on some of the underlying assumptions from the data and explanation of the models used and includes the step-by-step directions for implementing each. This thesis assumes the UxV position data can be analyzed and modeled using univariate time series forecasting methods; however, it also acknowledges the because the data comes from a network of systems designed to operate in a connected way that this data is correlated. Figure A.1 provides the correlation plot of each variable from the IDS and highlights two important facts:

1. UxV coordinates are correlated with other UxVs, and
2. The x- and y-coordinates of each UxV are highly correlated with each other.

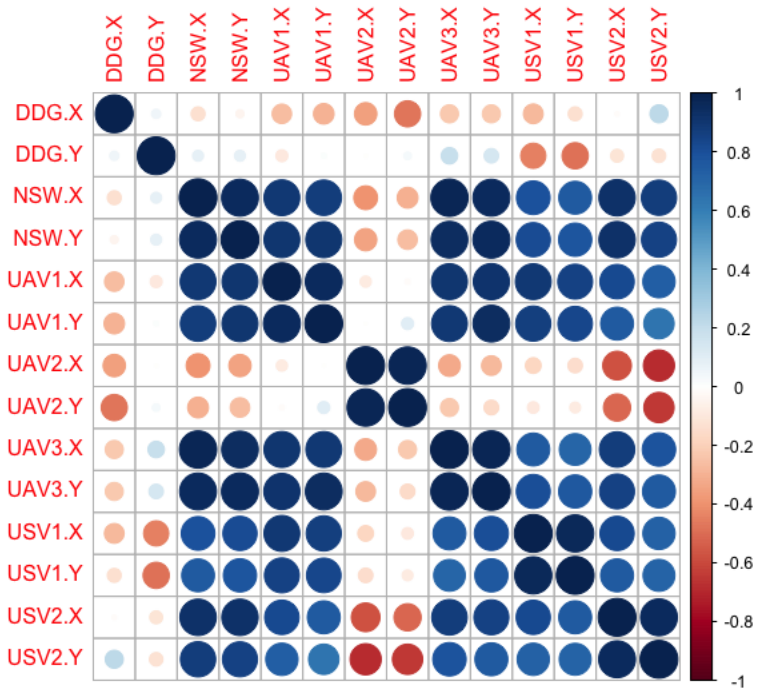

Figure A.1. IDS Correlation Plot

A.2 Time Series Linear Regression Model

The algorithms to develop the models used in this thesis are contained in Algorithms 2 and 3.

Algorithm 2: Time Series Linear Regression Model Formulation

Result: Predicted Ground Force Path using Linear Regression
Data Preprocessing;
Instruction 1: Conduct 80/20 training-test split of the entire data set—note, this split is NOT random and creates the subset of the data both models will be trained on as well as a subset of the data which this model will predict.
Time Series Forecast Model;
Instruction 1: Build univariate ARIMA model for the x- and y-component of each UxV;
Instruction 2: Forecast x- and y-coordinate predictions desired number of steps into the future;
Ground Force Triangulation Model;
Instruction 1: Using insights from unsupervised learning, create two linear regression models, one for each of the ground force's coordinates;
Instruction 2: Conduct a randomized 80/20 training/validation split on the 'training' data subset;
Instruction 3: Train the linear regression models using the training subset;
Instruction 4: Use the validation subset to test the accuracy of the linear regression model;
Instruction 5: Save the optimized linear regression model weights.;
Combined Time Series Linear Regression Model;
Instruction 1: Recall forecasted values from Time Series Forecast Model;
Instruction 2: Use forecasted values as input parameters to the optimized linear regression models;
Instruction 3: Compare predicted values of x- and y-coordinates of ground force to values from the "test" set;

A.3 Time Series Neural Network Model

Algorithm 3: Time Series Neural Network Regression Model Formulation

Result: Predicted Ground Force Path using Neural Network

Data Preprocessing;

Instruction 1: Conduct 80/20 training-test split of the entire data set—note, this split is NOT random and creates the subset of the data both models will be trained on as well as a subset of the data which this model will predict;

Time Series Forecast Model;

Instruction 1: Build univariate ARIMA model for the x- and y-component of each UxV;

Instruction 2: Forecast x- and y-coordinate predictions desired number of steps into the future;

Ground Force Triangulation Model;

Instruction 1: Build neural network, with all observable predictors as the inputs and the x- and y-coordinates of the ground force as the response variables;

Instruction 2: Conduct a randomized 80/20 training/validation split on the 'training' data subset;

Instruction 3: Train the deep neural network using the training subset;

Instruction 4: Use the validation subset to test the accuracy of the linear regression model;

Instruction 5: Save the trained neural network weights;

Combined Time Series Linear Regression Model;

Instruction 1: Recall forecasted values from Time Series Forecast Model;

Instruction 2: Using transfer learning, apply forecasted UxV values as the input parameters to the trained neural network;

Instruction 3: Compare predicted values of x- and y-coordinates of ground force to values from the "test" set;

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B:
Time Series Regression Model Application Results

This appendix provides a detailed report of the path deviation results when applying the Time Series Linear Regression and Time Series Neural Network models to the UDS.

Table B.1. Ensemble Model 1 Performance Matrix: Path Deviation (meters)

Time Step	UAV1	UAV2	UAV3	NSW
1	0.00001	8.526513e-14	37.39296	523.1826
2	156.4547	1.421085e-13	143.77240	637.6912
3	367.0061	1.989520e-13	302.02125	786.7997
4	577.5716	2.557954e-13	488.30615	963.3151
5	788.1411	3.126388e-13	695.30847	1160.1788
6	584.7435	3.694822e-13	816.13527	1252.1253
7	747.2414	4.263256e-13	934.12732	1364.4299
8	1201.4729	4.831691e-13	1062.16692	149.2.3049
9	1731.3081	5.400125e-13	1196.77864	1631.2357
mean	683.7711	3.126388e-13	630.6677	1090.140e+02

Table B.2. Ensemble Model 2 Performance Matrix: Path Deviation (meters)

Time Step	UAV1	UAV2	UAV3	NSW
1	8.6975	5.74E-11	4.1021	66.298
2	20.7765	1.15E-10	11.6985	42.2972
3	137.9459	1.72E-10	62.3256	24.3954
4	288.8922	2.30E-10	124.468	8.904
5	440.1368	2.87E-10	189.4864	6.4612
6	591.4511	3.44E-10	256.0677	21.7607
7	742.7933	4.02E-10	325.6235	33.2111
8	597.7482	4.59E-10	407.2015	39.2831
9	628.9607	5.17E-10	512.4748	47.1182
10	908.4265	5.74E-10	629.2061	53.38
11	1283.0601	6.32E-10	752.3557	60.0634
12	1690.7414	6.89E-10	879.4062	75.9563
mean	611.64	3.73177e-10	346.25	39.9

List of References

Aggarwal C (2018) *Neural Networks and Deep Learning* (Springer International Publishing).

Allaire J, Chollet F (2020) *keras: R Interface to 'Keras'*. URL https://CRAN.R-project.org/package=keras, r package version 2.3.0.0.

Burkov A (2019) *The hundred-page machine learning book, vol. 1* (Andriy Burkov).

Buttrey S (2021) Clustering. Lecture Slides, Statistical Machine Learning, Naval Postgraduate School,February, 2021, Monterey, CA.

Defense Science Board (2012) The role of autonomy in dod systems. Technical report, Defense Science Board, Washington, DC, https://fas.org/irp/agency/dod/dsb/autonomy.pdf.

Defense Science Board Task Force (2003) The role and status of DOD red teaming activities. Technical report, Defense Science Board, Washington, DC, https://fas.org/irp/agency/dod/dsb/redteam.pdf.

Department of Navy (2021) Cno navplan 2021. Technical Report NAVPLAN-2021, United States Navy, Washington, DC, https://media.defense.gov/2021/Jan/11/2002562551/-1/-1/1/CNO%20NAVPLAN%202021%20-%20FINAL.PDF.

Ester M, Kriegel HP, Sander J, Xiaowei X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, URL https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf.

Farina M, Misiano S (2018) Stochastic distributed predictive tracking control for networks of autonomous systems with coupling constraints. *IEEE transactions on control of network systems* 5(3):1412–1423, ISSN 2325-5870.

Hahsler M, Piekenbrock M, Doran D (2019) dbscan: Fast Density-Based clustering with R. *J. Stat. Softw.* 91(1), ISSN 1548-7660, URL http://dx.doi.org/10.18637/jss.v091.i01.

Huddleston SH, Brown GG (2018) *Chapter 7: Machine Learning (INFORMS Analytics Body of Knowledge)* (Wiley).

Hyndman RJ, Khandakar Y (2008) Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software* 26(3):1–22, URL https://www.jstatsoft.org/article/view/v027i03.

Insitu ABC (2021) Scan eagle. Https://www.insitu.com/products/scaneagle.

James G, Witten D, Hastie T, Tibshirani R (2013) *An Introduction to Statistical Learning* (Springer Science+Business Media, New York).

Kooij WER (2013) Graph measures and network robustness. *arXiv.org* .

Krause A, Golovin D (2014) *Submodular Function Maximization*, 71–104 (Cambridge University Press), URL http://dx.doi.org/10.1017/CBO9781139177801.004.

Lowry B (2020) *Distributed Submodular Optimization for a UXV Networked Control System*. Master's thesis, Department of Mechanical and Aerospace Engineering, Naval Postgraduate School, Monterey, CA.

Mangalam K, Girase H, Agarwal S, Lee KH, Adeli E, Malik J, Gaidon A (2020) It is not the journey but the destination: Endpoint conditioned trajectory prediction. *Proceedings of the European Conference on Computer Vision (ECCV)*.

Merenda M, Porcaro C, Iero D (2020) Edge machine learning for ai-enabled iot devices: A review. *Sensors* 20(9), ISSN 1424-8220, URL http://dx.doi.org/10.3390/s20092533.

Mesbahi M, Egerstedt M (2010) *Graph Theoretic Methods in Multiagent Networks* (Princeton University Press), stu - student edition edition.

Montero P, Vilar JA (2014) TSclust: An R package for time series clustering. *Journal of Statistical Software* 62(1):1–43, URL http://www.jstatsoft.org/v62/i01/.

NIST, SEMATECH (2013) *e-Handbook of Statistical Methods* (NIST), https://doi.org/10.18434/M32189.

Pi R (2021) Raspberry pi 4 model b specifications. URL https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/.

Plot JA (2019) *Red Team in a Box (RTIB): Developing Automated Tools to Identify, Assess, and Expose Cybersecurity Vulnerabilities in Department of the Navy Systems*. Master's thesis, Department of Computer Science; Naval Postgraduate School, Monterey, CA.

R Core Team (2021) *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, URL https://www.R-project.org/.

Rohrer JP, Xie GG (2013) Dtn hybrid networks for vehicular communications. *IEEE 2nd International Conference on Connected Vehicles* .

Shareef A, Zhu Y, Musavi M (2007) Localization using neural networks in wireless sensor networks. *MOBILWARE'08: Proceedings of the 1st international conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications*, 1–7 (ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering)), ISBN 978-1-59593-984-5.

Smith TG, et al. (2017) pmdarima: Arima estimators for Python. URL http://www.alkaline-ml.com/pmdarima, [Online; accessed 5 May 2021].

Tzinis I (2020) Delay/disruption tolerant networking. Https://www.nasa.gov/directorates/heo/scan/engineering/technology/disruption_tolerant_networking.

United States Marine Corps (2021) Force design 2030 - april 2021 update. Technical Report Force Design 2030-updated 2021, United States Marine Corps, Washington, DC, https://www.marines.mil/Portals/1/Docs/2021%20Force%20Design%20Annual%20Update.pdf.

Venables WN, Ripley BD (2002) *Modern Applied Statistics with S* (Springer, New York), fourth edition, URL https://www.stats.ox.ac.uk/pub/MASS4/, iSBN 0-387-95457-0.

Wachlin N (2018) *Robust Time-Varying Formation Control with Adaptive Submodulairty*. Master's thesis, Department of Mechanical and Aerospace Engineering, Naval Postgraduate School, Monterey, CA.

Yoshida R (2020) Data science modeling overview. Lecture Slides, Advanced Data Analytics, Naval Postgraduate School, September, 2020, Monterey, CA.

Zhang X, Han Q, Yu X (2016) Survey on recent advances in networked control systems. *IEEE Transactions on Industrial Informatics* 12(5):1740–1752, URL http://dx.doi.org/10.1109/TII.2015.2506545.

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California