NPS-MAE-21-001

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

NPS CRUSER MTX AFTER ACTION REPORT 2017-2020

by

Horner, Douglas

October 7, 2021

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

NAVAL POSTGRADUATE SCHOOL
Monterey, California 93943-5000

Ann E. Rondeau President	Dr. Scott Gartner Provost

Approved for public release; distribution is unlimited

This report was prepared by:

Horner, Douglas

Reviewed by:	Released by:
Garth Hobson Chairman of Mechanical and Aerospace Engineering	Jeffrey D. Paduan Dean of Research

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704–0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202–4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
7–10–2021	Technical Report	2017-01-01—2020-12-31

4. TITLE AND SUBTITLE

NPS CRUSER MTX After Action Report 2017-2020

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Horner, Douglas

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER

NPS-MAE-21-001

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Department of the Navy

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES

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

14. ABSTRACT

The Naval Postgraduate School (NPS) Consortium for Robotics and Unmanned Systems Education and Research (CRUSER) initiative Multi-Thread Experimentation (MTX) encompasses multiple research objectives with an overarching goal of developing a general purpose UxV Networked Control System (NCS). The report includes efforts over the last four years. It started with an initial, concept demonstration conducted in San Clemente Island, CA from November 01-17, 2017 with (4) Unmanned Aerial Vehicles (UAVs), (2) Unmanned Surface Vehicles (USVs) and (2) Unmanned Underwater Vehicles (UUVs) supporting a Naval Special Warfare (NSW) direct action mission scenario with command and control and simulated on-call distributed fire support from a COMTHIRDFLT ship. The report describes the results and the potential and future challenges associated with fielding a UxV NCS for Department of Defense mission objectives.

15. SUBJECT TERMS

Network Control System, Robotics, Wireless Networking, Artificial Intelligence. Unmanned Underwater Vehicles, Unmanned Aerial Vehicles, Unmanned Surface Vehicles

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Douglas Horner
Unclassified	Unclassified	Unclassified	UU	127	19b. TELEPHONE NUMBER *(include area code)* 831-656-3821

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

1 Introduction **3**
1.1 System Architecture 5
1.2 Technology Triad 6
1.3 Unmanned Systems Overview 7

2 Communications **11**
2.1 Mesh Radio Wireless Network 11
2.2 MTX Communications Network 11
2.3 Administrative Network 13
2.4 Network Data Collection and Analysis 13

3 Experimentation and Research Initiatives **17**
3.1 Research initiatives 18
3.2 High-level autonomy for NCS optimization 18
3.3 UxV NCS communications optimization 18
3.4 Optimal UAV trajectories supporting road network interdiction 18
3.5 Mobile mesh network performance and analysis 19
3.6 Cross-domain identification of road networks with Domain-Adapted Convolutional Neural Networks (DANNs) 19
3.7 A novel approach for training CNN for 3D objects 19
3.8 Team leaders and participants 19

4 Adaptive Submodularity for a UxV Network Control System **21**
4.1 Multi-objective optimization 24
4.2 Centralized submodularity 26

4.3 Distributed submodularity 29
4.4 Pareto optimality 30
4.5 Adaptive distributed submodularity 31

5 UxV NCS Communications Optimization **35**
5.1 Background 35
5.2 Methodology 35
5.3 MTX San Clemente Island Tests 35
5.4 Camp Roberts, CA Tests 36
5.5 Yuma Proving Ground Tests 36
5.6 Optimal spatial estimation 38
5.7 Information theoretic path planning 38

6 Optimal UAV Trajectories to Support Road Network Interdiction **45**
6.1 Optimal search 45
6.2 Detection model: camera sensor 45
6.3 Searcher model: UAV dynamics 47
6.4 Target model: Opposing forces on road network 47
6.5 Optimal control problem 48
6.6 Results 49

7 Mesh Network Performance and Analysis **55**
7.1 Introduction 55
7.2 Overview of MTX networking tasks 55
7.3 CENETIX networking team 55
7.4 Operations 56
7.5 Implementation and link integration 57
7.6 Network setup and configuration findings 57
7.7 Network operations findings 58
7.8 Unmanned systems formation networking findings 59
7.9 Captured data analysis 60
7.10 Conclusion 62

8 Cross-Domain Identification of Road Networks Using Domain Adversarial Neural Networks **65**
8.1 Overview 65
8.2 DANN background 66
8.3 DANN 68
8.4 Experimental Design 70
8.5 5-Phase process 73
8.6 Results 75

9 A Novel Approach for Training CNN for 3D Objects **83**
9.1 Model overview 83
9.2 Sensor platform configuration 84
9.3 Phase one: Automated dataset creation and training 87
9.4 Phase two: Context discovery 91
9.5 Phase three: Real-time pointcloud classification 91
9.6 Model conclusions 92
9.7 Results 92
9.8 Conclusions 96

10 Conclusions and Recommendations **99**

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

Figure 1.1 The Networked Control System (NCS) model stack. 6

Figure 1.2 The Technology Triad - Robotics, Internetworking and AI/ML impact all levels of the NCS model stack. It also highlights that NCS design and information flow is significantly impacted by the computation available on each of the platforms. 7

Figure 1.3 CAVR ROS Architecture 8

Figure 2.1 Overview of the communications network during the San Clemente Island portion of the MTX experimentation. 12

Figure 2.2 Three components of the Persistent System Wave Relay system used on San Clemente Island. a) the MPU5 carried by the SEAL tactical element b) the embedded module installed on the Insitu Scan Eagle UAV, c) The Quad Radio Router installed on the USV SeaFoxs. 13

Figure 2.3 The secondary controller hardware including the central database, mesh relay network card, secondary controller and main controller for the mobile unmanned system. 14

Figure 2.4 The image represents a snapshot of the visual display of networked information being sent to the central database. This display is available for each of the mobile agents within the NCS. 15

Figure 4.1 An example NCS graph with the nodes representing unmanned and manned systems and the edges representing the signal strength between valid communication paths. 22

Figure 4.2 A superimposed grid on top of a map of the northern half of San Clemente Island, CA. It highlights the complexity of the problem for determining how to place each of the unmanned systems into the grid. It has been identified as an NP-Hard complexity problem 25

Figure 4.3 The sensing utility $\Phi_{i,j}$ assigned to UAV i for all locations $j = 1, 2, \ldots, p$. (a) the satellite imagery is analyzed to extract (b) points of interest such as roads, and (c) an utility is assigned to each discretized point in accordance with a ranging function. 26

Figure 4.4 The sensing utility $\Phi_{i,j}$ assigned to UUV i for all locations $j = 1, 2, \ldots, p$. (a) the satellite imagery is analyzed to (b) eliminate the land, and (c) an utility is assigned to each discretized point in accordance with a ranging function. 27

Figure 4.5 Topological changes are shown for a single mission phase. These changes are induced by varying the utility subfunction weights within the optimization function. In (a) $\alpha_s = 0$ and $\alpha_r = 1$. In (b) $\alpha_s = 1$ and $\alpha_r = 0$. In (c) $\alpha_s = 1$ and $\alpha_r = 1$. In (d) $\alpha_s = 3$ and $\alpha_r = 1$. 28

Figure 4.6 (a) The network is shown with the most recently-placed node highlighted in red. This node was placed at the point maximizing the utility function. (b) The total utility shown as an intensity map. (c) An intensity map of the sensing utility $\Phi_{i,j}$ shown for all locations j. (d) A heat map of the robustness utility $\Omega_{i,j}$ shown for all locations j. 29

Figure 4.7 The optimal network configuration is shown for a simple grid containing 130 discretized points (the image shown is a subset of the grid containing 108 grid boxes). It requires 3.7 x 10^{10} evaluations of the utility function to determine the optimal configuration with utility $J(S^*) = 0.9895$. . . 30

Figure 4.8 The near-optimal network configuration is shown for a small grid containing 130 discretized points. This only required 650 evaluations of the utility function to determine the optimal configuration with utility $J(S^*) = 0.9820$. 31

Figure 4.11 Two different Pareto-optimal NCS topologies. Locations near the target objective (red triangle) and NSW team (purple asterisk) have higher sensing utility values. 31

Figure 4.9 The Distributed Submodularity Algorithm 32

Figure 4.10 Example of a Pareto front. 33

Figure 4.12 Adaptive Submodularity approach to distributed submodularity for a UxV NCS. On the left-hand side is a time sequence of optimal solutions for the combination of communications and sensing for the NCS. On the top right side is the result of the policy in terms of the NCS simulation based on a epsilon constraint of the pareto front supporting a notional SEAL direct action mission. The bottom right shows the initial components associated with the quality of experience metrics for commander's intent. 34

Figure 5.1 Camp Roberts SNR data for ScanEagle link to Node 1 (ECEF). 37

Figure 5.2 CampRoberts SNR data for ScanEagle link to Node 2 (ECEF). 38

Figure 5.3 Yuma SNR data for ScanEagle link to Node 1 (ECEF). 39

Figure 5.4 Yuma SNR data for ScanEagle link to Node 2 (ECEF). 40

Figure 5.5 Yuma SNR data vs. distance from each ground node. 41

Figure 5.6 Yuma Estimate of SNR Fields. 42

Figure 5.7 Kriging estimate of SNR field intersection. 43

Figure 5.8 Kriging estimate of SNR field intersection. 44

Figure 6.1 Main components of the optimal search problem. 46

Figure 6.2 Example Detection Rate Function: Poisson Scan Model 47

Figure 6.3 The steps involved in formulating the optimal search problem, beginning with a digital map of the operating area (a), extraction of the road network (b), and assigning *a priori* probabilities to opposing forces' starting locations (c). 50

Figure 6.4 Minimizing risk from opposing forces by solving an optimal control problem to compute UAV search trajectories (a), flying the UAV's sensor over the road network to minimize risk of non-detection (b), and choosing a path for the SEAL team which avoids high-risk areas (c). 50

Figure 6.5 Optimal search trajectory flown by the ScanEagle UAV in support of a SEAL team's ingress from its insertion point in the south to its objective in the north. 51

Figure 6.6 Optimal search trajectories flown by the ScanEagle UAV in four phases, from yellow, to green, to magenta, to light blue. The UAV supported a SEAL team's ingress from its insertion point in the south to its objective in the north. 53

Figure 7.1 Network Operations Center (NOC): Experimentation Phase Plan Example 56

Figure 7.2 Network Operations Center (NOC): Experimentation Phase Plan Example 57

Figure 7.3 MTX Distributed Data Logger 58

Figure 7.4 SNMP Plug-in Agents Activity Monitor 59

Figure 7.5 Solar Winds Network Management System and Wave Relay HTTP-interface 60

Figure 7.6 Observer's Notepad collaborative log tool 60

Figure 7.7 Ship Joining Network 61

Figure 7.8 ScanEagle Extending Network to Ship 61

Figure 7.9 Supporting the Mission Area Unit Networking 62

Figure 7.10 MTX Distributed Data Logger: Low Bandwidth Chatting Room 63

Figure 7.11 Throughput-In distribution 64

Figure 7.12 Input-Throughput vs. Output-Throughput 64

Figure 8.1 Pictured is the general data flow and processing steps, colored by phase. The domains and datasets are created in phase I. The satellite and UAS camera CNNs are trained in phases II and III. The upper and lower performance bounds are established in phase IV and the DANN is trained and performance is evaluated in phase V. 66

Figure 8.2 Several CNNs are trained in phases II and III. At the top, is the satellite CNN where the left side X represents the training set and the right side Y represents the validation set used to train them. The next level is the same for the UAS CNN. In phase IV, the models from phases II and III will be evaluated against the X/Y "outside" each model(UAS testset) to determine lower and upper performance bounds. In phase V, DANNs are similarly trained and evaluated against the UAS test set to determine DANN performance 67

Figure 8.3 An overview of the Camp Roberts area from which satellite data was collected. Several distinct satellite collection passes that compose the image mosaic are visible. Clips for the source dataset were pulled from the red-bounded region. 70

Figure 8.4 71

Figure 8.5 This a sampling of the “road" class from the 160x160 satellite domain dataset. It shows how broad the domain definition for “road" is and how challenging it could be with some of these images for the classifier. . . 76

Figure 8.6 CNN training and validation plots for the simple, custom and transfer learning architectures on the 120x120 satellite data. 79

Figure 8.7 CNN training and validation plots for the simple, custom and transfer learning architectures on the 160x160 satellite data. 80

Figure 8.8 CNN training and validation plots for the simple, custom and transfer learning architectures on the 40x40 UAS video data. 80

Figure 8.9 DANN plots for source training and validation accuracy, target validation accuracy, and domain training and validation accuracy for the “simple" architecture. 80

Figure 8.10 DANN plots for source training and validation accuracy, target validation accuracy, and domain training and validation accuracy for the custom architecture. 81

Figure 9.1 Flowchart of the Three-Phase Point Cloud Classification Model. Phase one is the automated dataset creation. Phase two is context discovery. Phase three is pointcloud classification. 84

Figure 9.2 3D printed housing and mount for the lidar and cameras. Inside the mount are a 200W inverter, 12V battery and Velodyne lidar interface box. 85

Figure 9.3 The sensor mount with the Veloryne lidar and four USB Logitech cameras. 85

Figure 9.4 Horizontal Field of View (FOV) of the camera and LIDAR sensing platform. Only the the front camera was used for testing. 86

Figure 9.5 Combined 32.32 degree Vertical FOV of the camera and LIDAR systems. 87

Figure 9.6 Example of the combined data capture with the 3 images from the cameras and the corresponding LIDAR pointcloud scene. 88

Figure 9.7 The LIDAR and camera hardware to software flowchart. 89

Figure 9.8 The phase one processes: Pointcloud segmentation, 3D-2D Correlation, 2D classification and confidence-level thresholding. 90

Figure 9.9 Example of a Raw LIDAR pointcloud of an excavator. 90

Figure 9.10 Transformed excavator to a scaled, translated and voxelized representation. 90

Figure 9.11 Map view of routes for Neighborhood 1 and Neighborhood 2 data collections. 93

Figure 9.12 Visualization of "Scene Context" database containing relationship weights between objects in a specific physical environment. 94

Figure 9.13 Performance on "Neighborhood 1" dataset in producing correctly labeled pointcloud segments. 95

Figure 9.14 Occlusion Example. The tree pointcloud segment labeled as a car due to foreground of the image crop. 96

Figure 9.15 Performance on "Neighborhood 2" dataset in producing correctly labeled pointcloud segments. 97

List of Tables

Table 8.1 Definitions of road and trail classes. All of these classes are considered "road" class for training. Everything else is classified as "not road". . . 74

Table 8.2 Definitions of road and trail classes. All of these classes are considered "road" class for training. Everything else is classified as "not road". . . 75

Table 8.3 Distribution of generated satellite datasets. 75

Table 8.4 Distribution of generated UAS video datasets. 76

Table 8.5 Maximum accuracy of the three evaluated CNN architectures during training and validation and the total number of epochs after early stopping or after the maximum set 600 epochs was reached. 76

Table 8.6 Target domain only CNN accuracy. Accuracy of the three evaluated CNN architectures during training and validation. 77

Table 8.7 Evaluation on 120x120 trained CNN. Accuracy of the highest performing models evaluated on the test sets.Source on source accuracy include for reference only. 77

Table 8.8 Evaluation on 160x160 trained CNN. Accuracy of the highest performing models evaluated on the test sets.Source on source accuracy include for reference only. 78

Table 8.9 Evaluation on 120x120/40x40 trained DANNs on the 40x40 dataset. Accuracy of the highest performing architectures evaluated on the test sets. 78

Table 8.10 Evaluation on 160x160 trained CNN. Accuracy of the highest performing models evaluated on the test sets.Source on source accuracy include for reference only. 79

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

ADCP Acoustic Doppler Current Profiler

AI/ML Artificial Intelligence/Machine Learning

API Application Programming Interface

C2 Command and Communication

C4ISR Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance

CAVR Center for Autonomous Vehicle Research

CENETIX Center for Network Innovation and Experimentation

CNN Convolutional Neural Network

COT Cursor On Target

CRUSER Consortium for Robotics and Unmanned Systems Education and Research

DA Domain Adaptation

DANN Domain Adapted Neural Network

ECEF Earth Centered Earth Fixed

DON Difference of Normals

GCS Ground Control System

GenOC Generalized Optimal Control

GRL Gradient Reversal Layer

GPS Global Positioning System

GSD Ground Sample Distance

HITL Human In The Loop

HOTL Human On The Loop

HRI Human/Robot Interface

IPB Intelligence Preparation of the Battlespace

INS Inertial Navigation System

KL Kullbeck Leibler

KLD Kullbeck Leibler Divergence

LOS Line of Sight

LTI Linear Time Invariant

MANET Mobile Ad-hoc Network

MCTS Monte Carlo Tree Search

MIMO Multiple Input Multiple Output
MoPs Measures of Performance
MOUT Military Operations in Urban Terrain
MSL Mean Sea Level
MTX Multi-Threaded Experimentation
NAWC Naval Air Warfare Center
NIWC Naval Information Warfare Center
NN Neural Network
NOC Network Operation Center
NPS Naval Postgraduate School
NAVSPECWAR Naval Special Warfare
NEDU Navy Experimental Diving Unit
NSW Naval Special Warfare
NCS Networked Control System
NGA National Geospatial-Intelligence Agency
OSE Optimal Spatial Estimation
OSI Open Systems Interconnection
PCAP Packet Capture
POMCP Partially Observable Monte Carlo Processing
POMDP Partially Observable Markov Decision Process
RCP Remote Control Protocol
RF Radio Frequency
ROS Robot Operating System
RTT Round Trip Time
SA Situational Awareness
SCI San Clemente Island
SISO Single Input Single Output
SLAM Simultaneous Localization and Mapping
SNMP Single Network Management Protocol
SNR Signal Noise Ratio
SRT Special Reconnaissance Team
SOCOM Special Operations Command
UAV Unmanned Aerial Vehicle
UDP User Defined Protocol

USV Unmanned Surface Vehicle

UGV Unmanned Ground Vehicle

UxV Unmanned [Aerial, Ground, Surface, Underwater] Vehicle

WV02 WorldView-2 Satellite

XML Extensible Markup Language

YPG Yuma Proving Ground

THIS PAGE INTENTIONALLY LEFT BLANK

Abstract

The Naval Postgraduate School (NPS) Consortium for Robotics and Unmanned Systems Education and Research (CRUSER) initiative Multi-Thread Experimentation (MTX) encompasses multiple research objectives with an overarching goal of developing a general purpose UxV Networked Control System (NCS). The report includes efforts over the last four years. It started with an initial, concept demonstration conducted in San Clemente Island, CA from November 01-17, 2017 with (4) Unmanned Aerial Vehicles (UAVs), (2) Unmanned Surface Vehicles (USVs) and (2) Unmanned Underwater Vehicles (UUVs) supporting a Naval Special Warfare (NSW) direct action mission scenario with command and control and simulated on-call distributed fire support from a COMTHIRDFLT ship. The report describes the results and the potential and future challenges associated with fielding a UxV NCS for Department of Defense mission objectives.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

Mobile unmanned systems provide an unparalleled ability to collect information and offset numerical disadvantages on the battlefield while simultaneously reducing risk to personnel. These advantages were made clear during the recent Nagorno-Karabakh war between Azerbaijan and Armenia. Azerbaijani forces, who had lost a previous war to Armenia, were able to use unmanned aerial systems to quickly establish air supremacy through precision fires to enable ground maneuver. The scale and speed of these operations may be a harbinger of future strategies for top peer adversaries.

The CNO's NAVPLAN 2021 [1] recognizes this reality and reliance on unmanned systems as a fundamental component of Naval power projection...

"Unmanned platforms play a vital role in our future fleet. Successfully integrating unmanned platforms—under, on, and above the sea—gives our commanders better options to fight and win in contested spaces. They will expand our intelligence, surveillance, and reconnaissance advantage, add depth to our missile magazines, and provide additional means to keep our distributed force provisioned. Furthermore, moving toward smaller platforms improves our offensive punch while also providing affordable solutions to grow the Navy. Through analysis, simulations, prototyping, and demonstrations, we will systematically field and operate systems that possess the endurance and resilience to operate with infrequent human interaction. By the end of this decade, our Sailors must have a high degree of confidence and skill operating alongside proven unmanned platforms at sea".

Within this new reality, a networked team of collaborative unmanned systems offers to further increase mission effectiveness through sharing information and responsibilities to improve operational knowledge, speed and precision. Our work at Naval Postgraduate School (NPS), starting in 2017, has focused on research and experimentation for a collaborative pairing of unmanned aerial, surface, ground and undersea platforms - a UxV Networked Control System.

Consistent with the Navy's recently released "UNMANNED" campaign plan [2], the NPS CRUSER-funded initiative, Multi-Threaded Experimentation (MTX), sought to develop an initial mobile UxV Networked Control System (NCS). The NCS is a distributed system of mobile agents (manned and unmanned systems) where sensing, guidance, navigation, control and communication information are exchanged through a wireless network to support mission objectives. From a system control perspective, the addition of the wireless network adds complexity through the introduction of time-varying delays, imperfect information exchange and information loss [1].

[1] For a video overview of the MTX experimentation of San Clemente Island, see `https://www.youtube.com/watch?v=o2mTAzyZdPo`

A heterogeneous mix of unmanned systems includes different modalities for communication, sensing and navigation. This adds to the diversity and robustness of the NCS. Communications can be transmitted through multiple modalities including acoustic, radio and optical devices. Each have performance characteristics that impact overall network performance. Of additional consideration is the positioning of agents to optimize these communication links; this includes agents that provide a relay between domains (an example is a USV that can act as a mobile communications gateway buoy between acoustic and RF transmission paths).

Sensing devices are frequently specific to the physical environment they operate in. An NCS might include sonar, radar, lidar, and cameras. It can produce a large amount of data. In order not to overload the limited network bandwidth between the mobile agents, processing the sensor data to filter out superfluous information greatly reduces the network load. With respect to the navigation and control of the UxVs, each has different characteristics including on-station time, maneuverability, velocity, detectability and coverage rate. These vehicles can work in collaboration to achieve better results, this includes localization and multi-sensor detection and classification.

Overall the mobile UxV NCS supports traditional military mission areas that include: collaborative search, collaborative coverage, distributed fires, hybrid control and formation control but a central tenet of the approach is that the system diversity enhances robustness and creates an approach useful for a wider sets of military missions.

Some of the most critical general research questions relative to the UxV NCS include the following:

1. ***How is the system controlled?*** Among considerations for control of the system is whether it is a centralized, distributed or a combination. An additional consideration is the role of humans in system control. Options include Human-in-the-Loop (HITL) where all decisions involve human input and Human-on-the-Loop (HOTL) where the system is capable of acting autonomously but humans observe the system and can inject command and control (C2) when desired.
2. ***How is the system optimized?*** Of interest is how to place unmanned systems to support ground operations. Examples include optimization with respect to capabilities (e.g. communications and sensing), mission objectives, duration, robustness and flexibility.
3. ***What are the attributes and vulnerabilities of these systems?*** An example of a vulnerability might be the ability of opposing forces to figure out mission objectives or unit maneuvers.
4. ***What is the role of autonomy in the development of the system?*** A military UxV NCS will require human decision-making for the foreseeable future. Given that there will be an Artificial Intelligence/Machine Learning (AI/ML) system autonomy component for positioning the unmanned systems, a significant issue will be the design of the interface between the control autonomy and human decision-makers that permits transparency, flexibility and control.

1.1 System Architecture

Figure (1.1) shows the system architecture. It emphasizes a modular approach that is analogous to the Open Systems Interconnection (OSI) model [3]. It consists of the following layers:

1. **Layer 1: Mobile Agents** - The manned and unmanned systems that make up the network. Each node is distinguished by its unique parameters associated with its capabilities. This can include, but is not limited by the node's mobility, energy, speed, sensing, communications and computational processing.
2. **Layer 2: Communications** - The ability to communicate is critical for controllability and observability of NCS. It is not limited to a single communications infrastructure (e.g. wireless communications) but can be composed of multiple modalities. For MTX, there were both RF and acoustic communications. The RF network was represented by a Persistent Systems mesh relay radio (`http://www.persistentsystems.com`). The mesh relay includes a routing discovery service software component that automatically determines routes for message traffic between agents. An important component of the communications is the ability to use connection metrics associated with the communication medium for ensuring system controllability. For example, through an API, it is possible to measure the channel statistics such as Signal to Noise Ratio (SNR) between radios. This can be used for optimally positioning systems within the network to ensure control and sensory data can be reliably transmitted through the network.
3. **Layer 3: Information** - The information layer is an abstraction that represents the protocols and content that is transmitted through the network. The information layer includes UxV state, sensor data and interconnectivity data. The state data gives the position, orientation (and their derivatives) and material condition of the agents. The interconnectivity data (from the communications API) is used for control and navigation. It is the paths that are viable locally within the network. This provides a necessary input for determining appropriate control strategies to accomplish tasks such as creating a robust communications path between nodes. The information layer supports autonomous inferencing associated with AI/ML approaches especially with respect to multi-sensor fusion.
4. **Layer 4: Control and Navigation** - This is the ability of the system to achieve mission objectives under the constraints of communication, sensing, energy, time and vehicle dynamics. Control of the system can be centralized and/or distributed. The ability to have both centralized and distributed control can potentially add flexibility to handle multiple mission scenarios. It includes trajectory or path planning for all system nodes including recommendations for manned systems.
5. **Layer 5: Human/Robot Interface (HRI)** - HRI involves both the potential for a human to control the system and the ability of the NCS to provide timely information to users. A key design emphasis is the ability to control a greater number of agents with fewer people.
6. **Layer 6: Cyber-Security** - Protecting all layers is a Cyber-Security model. This would include application security, information security, network security, algorithmic security, disaster recovery and end-user compliance.

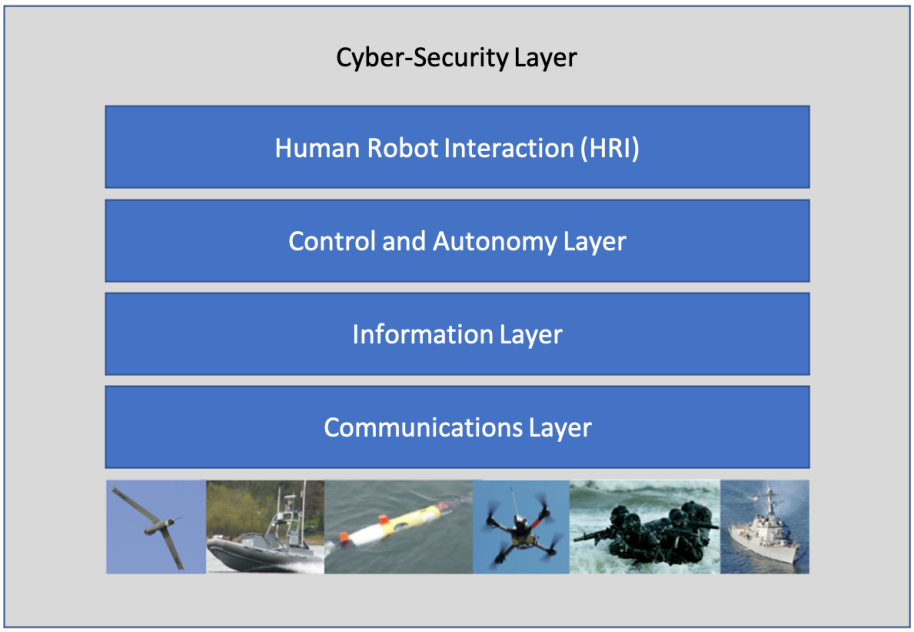

Figure 1.1: The Networked Control System (NCS) model stack.

1.2 Technology Triad

The push towards greater levels of autonomy revolve around on technology triad of Robotics, Internet-networking and AI/ML (figure 1.2). Each impacts all of the system architecture layers described above. Critical issues in Robotics for a UxV NCS include: 1). Automating of logistics and maintenance for deploying and recovering many unmanned systems rapidly. 2). Reducing the forces necessary for operating and maintaining the unmanned systems. Increasing the “Tooth-to-tail" ratio improves mission effectiveness through greater operational flexibility and reducing force protection requirements.

Salient issues for AI/ML revolve around trust. They include: transparency/explainability, ethical considerations, system controllability, robustness and reliability [4], [5]. Issues for internet-networking include development of a general flexible communications model to support maximizing system utility, development of mesh radio algorithms for dynamic packet route planning and development of service-oriented architecture to prioritize network flow depending on packet importance.

In the center of the triad is computation. It highlights the impact of computation on the system performance. The type of unmanned platform influences the amount of onboard computational available. Smaller UAVs and UUVs tend to have limited onboard computation while USVs and UGVs may have more flexibility for including additional computers. Offloading processing and positioning agents accordingly may need to be explicitly considered within the design and employment of the system.

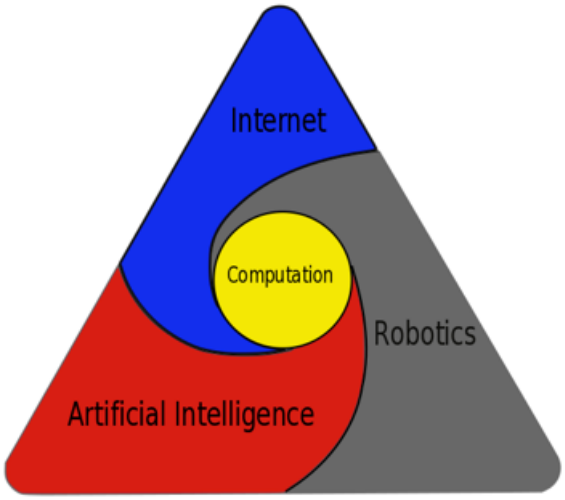

Figure 1.2: The Technology Triad - Robotics, Internetworking and AI/ML impact all levels of the NCS model stack. It also highlights that NCS design and information flow is significantly impacted by the computation available on each of the platforms.

1.3 Unmanned Systems Overview

The collection of unmanned systems for MTX were almost entirely provided by the NPS Center for Autonomous Vehicle Research (CAVR) (`https://nps.edu/web/cavr`). All had a WiFi capability and the UUVs were the only vehicles that didn't have the Persistent Systems Mesh Relay radios. They passed information while undersea using acoustic modems. The NCS unmanned systems included the following:

1. **(2) NPS REMUS UUVs** - These Hydroid systems are specialized REMUS 100 UUVs that include the following sensing and navigation features: Navigation grade Inertial Navigation System (INS), upward and downward looking Acoustic Doppler Current Profiler (ADCP), 900/1800 KHz side scan sonar, Blueview 450 forward looking sonar and 2.25 GHz micro-bathymetry sonar, WHOI acoustic micro-modem, GPS and WiFi.
2. **(2) NPS SeaFox USVs** - The systems were built by Northwind Marine. The first hull acted as a mobile communications relay buoy and featured the ability to insert an acoustic modem into the water for collecting and disseminating acoustic data from the UUVs through the UAVs to users. The second hull was used for inserting the SEALs and had a mounted radar for detecting surface traffic. The SeaFoxs have water jet engines that run using JP-5 and are capable of greater than 40 knots.
3. **(2) NPS ScanEagles** - Built by Insitu these UAVs were operated by a combination of NPS and NAVSPECWAR Special Reconnaissance Team ONE. The UAVs have been modified to include a NAWC China Lake built Power Control Board (PCB), and an NPS secondary controller CPU. This permits the ability to collect state information and distribute the data through the mesh radios.
4. **(1) Shield AI Quadrotor** - The quadrotor was used for searching inside a building for

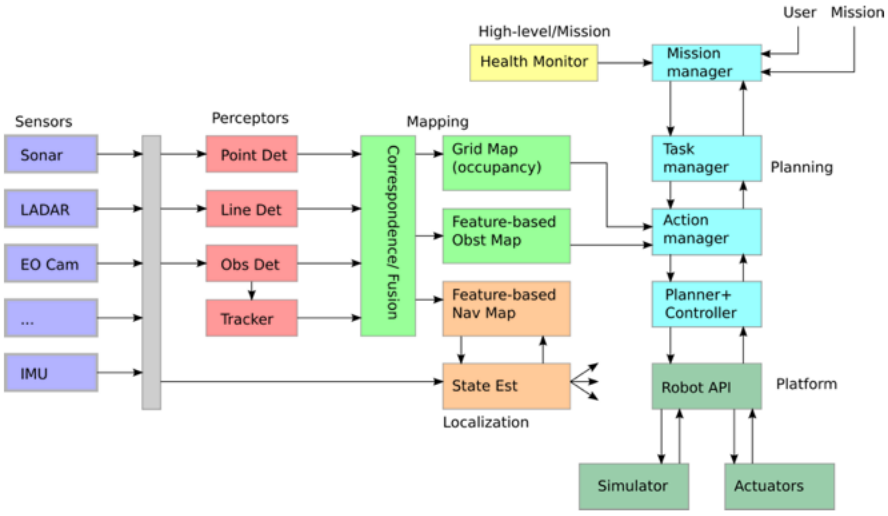

Figure 1.3: CAVR ROS Architecture

the detection of the notional radiological equipment that was the objective of the NSW mission scenario. The ability to navigate without GPS using just the camera and a rudimentary dead reckoning algorithm is known as SLAM - Simultaneous Localization and Mapping and is critical for GPS degraded or denied environments such as indoors, space and undersea.

A key design consideration for all NPS UxV systems was the implementation of a secondary controller architecture. All NPS systems are military grade systems. They either are or have been in the military inventory for operational forces. The secondary controller is an installed computer that, through a communications Application Programmers Interface (API), permits sending overriding commands to control the unmanned system and receiving state information from the main controller. In general, this permits separating higher level autonomy (e.g. route planning) from low level commands (e.g. controlling the system actuators to follow a path). This approach assumes the manufacturer provides the "low-level" control and communication API. This can greatly speed up the development process and is considered an integral component for expansion of the multi-agent system regardless of the unmanned system manufacturer. An example of this approach is the Hydroid REMUS RECON or Remote Control Protocol (RCP) API.

Figure (1.3) shows the overall software architecture of the CAVR Secondary Controller Architecture. It is built using a Linux Operating System (Ubuntu 18.04) and the Robotic Operating System (ROS) Melodic Morenia as the software middleware (`https://www.ros.org/`). Each box represents a ROS node. Information flows from left to right. The color of the box reflects the functionality of the ROS node. Data is collected from various sensors (purple). Perceptors process the sensor data (pink) and provide this information into mapping components (green). This information is used as input into path planning and state estimation (orange).

There are processes for managing the navigation of the vehicle (light blue). This includes a mission manager, task manager, planners and controllers. There is also a health monitor (light yellow). Finally there are processes that include the sending and receiving information from/to the main controller and for simulation and actuators (dark green).

The remaining document describes in detail the research and experimentation associated with MTX. This includes the San Clemente Island (SCI) demonstration but also includes testing conducted with NPS ScanEagle at the Yuma Proving Grounds (YPG) and Camp Roberts, CA, UUV and USV testing in Monterey Bay, CA and recent developments associated with the overall NCS system approach. The experimentation “threads" include: High-level autonomy for NCS optimization, UxV communications optimization, optimal UAV trajectories to support road network interdiction, mobile mesh network performance and analysis, cross-domain identification of road networks using domain-adapted convolutional neural networks and automated creation of labeled pointcloud datasets in support of machine learning-based perception.

The next chapter focuses on communications. Viewing the UxV NCS as a single system with manned and unmanned agents as the subcomponents, communications can be abstractly viewed as “virtual springs" of a mechanical system. Each spring between a pair of communicating nodes is a representation of the signal strength where a fully extended “spring" represents minimal communications strengths between the nodes. This is useful for considerations of system observability and controllability.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Communications

Figure (2.1) is an overview of the UxV NCS network architecture while at SCI. It includes both operational and administrative components. The operational network included the mobile agents that could communicate through either RF or acoustic modems. The administrative network was a combination of mobile and fixed network nodes that were designed for experimentation safety and coordination and used as a backup in case the operational network was unavailable (i.e. the Scan Eagle weren't in the air).

2.1 Mesh Radio Wireless Network

The backbone of the wireless network was the persistent systems mesh radios known as Wave Relay (`https://www.persistentsystems.com/`). They were located on the ScanEagle UAVs, the SeaFox USVs, C2 and administration and safety personnel. At the heart of the mesh radios is a software component that facilitates a Mobile Ad-hoc Network (MANET) for enabling a more flexible model of wireless communications through self-forming and self-healing. Shown in figure 2.2, the following radios were used:

1. MPU-4 and 5 Handheld radios
   `https://www.persistentsystems.com/site/wp-content/themes/persistensystems/pdf/mpu5/mpu5_spec_sheet.pdf`
2. Embedded Module
   `https://www.persistentsystems.com/embedded-module/`
3. Auto-Tracking Antenna System
   `https://www.persistentsystems.com/tracking-system-overview/`
4. Integrated Antenna Series
   `https://www.persistentsystems.com/integrated-antenna/`
5. Quad Radio Router
   `http://www.persistentsystems.com/pdf/Quad_SpecSheet.pdf`

2.2 MTX Communications Network

Starting at the top left of figure 2.1, the operational network consisted of the 2 NPS CAVR Seafoxes with the 2.4GHz Quad Router radios. These were connected to secondary controllers running the CAVR ROS software on a Ubuntu 18.04 operating system including the Center for Network Innovation and Experimentation (CENETIX) SNMP agent. The Seafoxes were required to deploy with a support boat and they were allotted two IP addresses. Omnidirectional, high gain (12dBi) antennas were mounted on the aft mast.

Moving clockwise, the SEAL element was assigned 5 MPU4 or 5 handheld radios. The MPU4 had a single omnidirection (2-3dBi) stub antenna and the MPU5 had three of these anten-

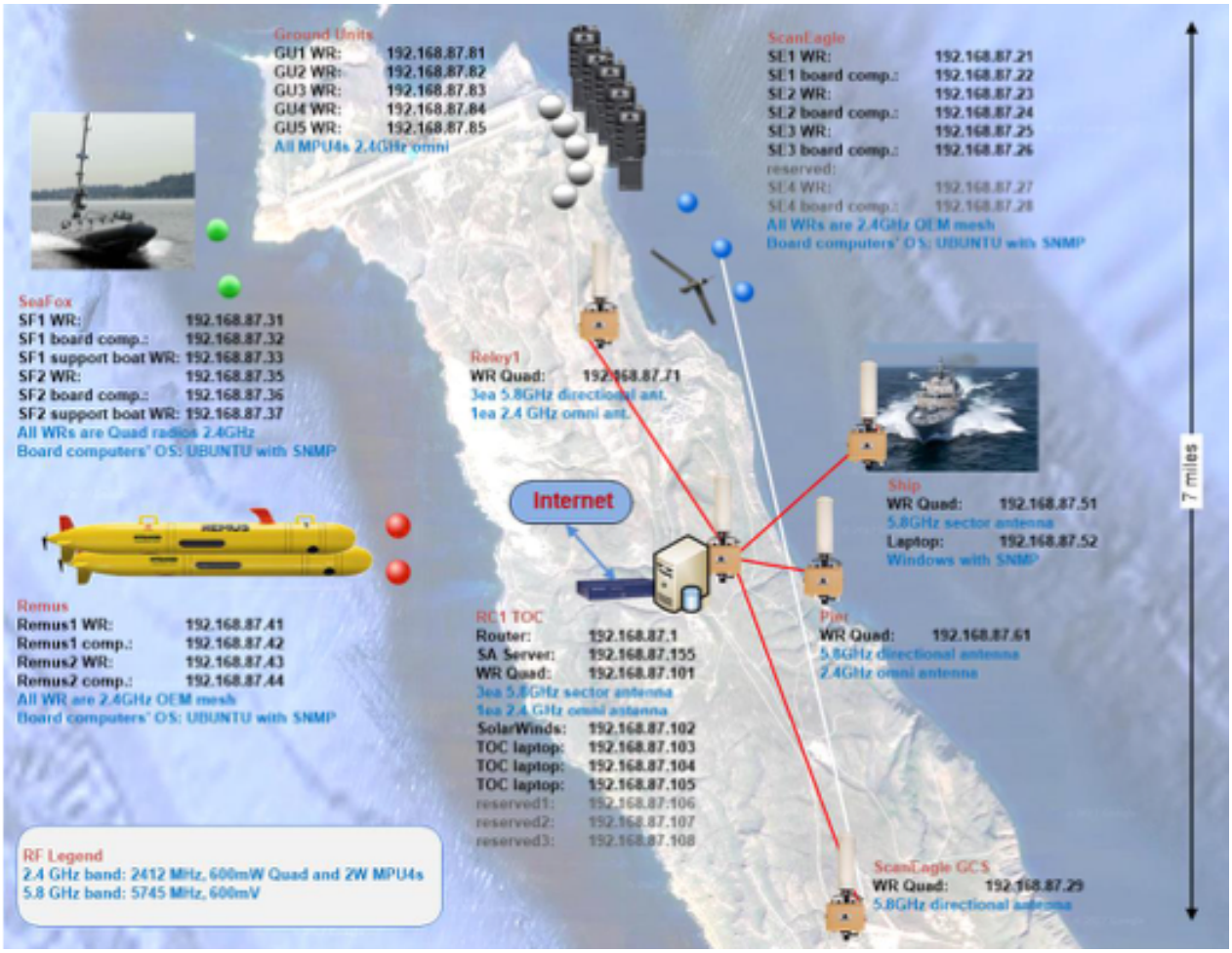

Figure 2.1: Overview of the communications network during the San Clemente Island portion of the MTX experimentation.

nas. There was a significant difference in performance between the Single Input Single Output (SISO) MPU4 radios and the Multiple Input Multiple Output (MIMO) MPU5 radios.

The 3 NPS CAVR Scan Eagle UAVs had the secondary controller architecture shown in figure (2.3). It included the embedded Wave Relay radio and an Odroid RX1 as the secondary controller CPU which was connected to the main ScanEagle CPU for sending overriding control commands and receiving state information. Additionally a Raspberry Pi CPU was the dedicated SNMP agent for sending and receiving quality of service metrics across the network. A Shield AI quadrotor was used as part of the SEAL direct action mission exercise. It was used to conduct an initial clearing of a building. Video collected by the quadrotor was transmitted in real time into the WiFi network. The quadrotor was not using the Wave Relay but was still able to make itself available as an NCS network node.

A COMTHIRDFLT ship was part of the experimentation as the expeditionary command and

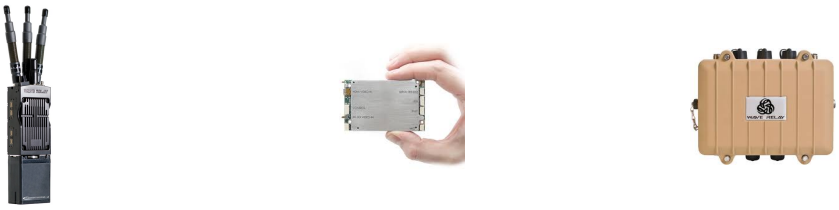

Figure 2.2: Three components of the Persistent System Wave Relay system used on San Clemente Island. a) the MPU5 carried by the SEAL tactical element b) the embedded module installed on the Insitu Scan Eagle UAV, c) The Quad Radio Router installed on the USV SeaFoxs.

control (C2). Onboard was a 5.8GHz Wave Relay radio with both a high-gain directional (or sector) and omnidirectional antenna. The NPS CAVR REMUS UUVs were the only unmanned systems not running the Wave Relay radios. They had an existing WiFi capability that was able to connect into the network while on the surface. WiFi is not available while the vehicles are underway undersea so information was transmitted via acoustic modems to the USVs which then acted as a mobile buoy to collect and transmit data into the WiFi network. This will be more fully described later in the report.

2.3 Administrative Network

Critically important to the safety and logistics experiment was the administrative network. This is seen in the center of the diagram. Operations were overseen from a SCI facility known as RC1. From there, WiFi connections with fixed directional antennas were set between a) the Scan Eagle Ground Control Site (GCS) was where the UAVs were launched and recovered b) the Naval Information Warfare Center (NIWC) Dive Locker, where UUV and USV operations were conducted from and c) a midway point between the SEAL exercise objective and RC1. All of these fixed nodes had the Wave Relay Quads with various high gain antennas and either 2.4 or 5.8 GHz radios.

2.4 Network Data Collection and Analysis

Key to the effort was the collection and dissemination of information across the network. This included network performance. To do so, software was developed and installed as part of the secondary controller for each of the mobile agents. It was developed by the NPS CENETIX lab (`https://nps.edu/web/cenetix`) and called the Simple Network Management Protocol (SNMP) agent. It included the following capabilities:

1. Measure SNR between one agent and all (one-hop) mobile agent neighbors.
2. Measure Throughput In, Throughput Out, Percent Packet Loss, Round Trip Time (RTT) between one agent and all (one-hop) mobile agent neighbors.
3. Transmission of state information of the mobile agent (at 1 Hz) using a Cursor On Target (COTs) Extensible Markup Language (XML) message format sent via User Defined Protocol (UDP).

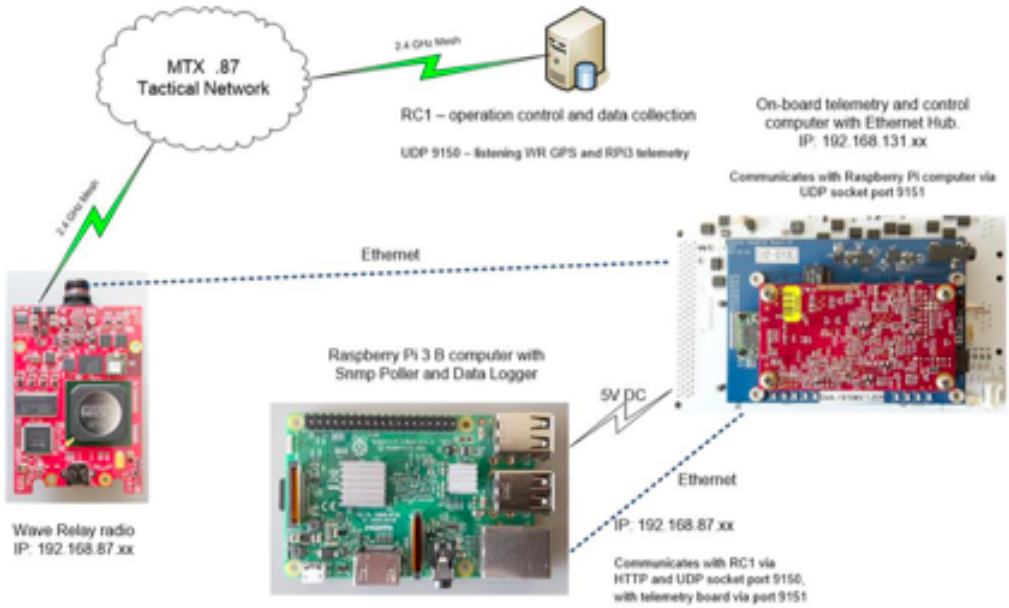

Figure 2.3: The secondary controller hardware including the central database, mesh relay network card, secondary controller and main controller for the mobile unmanned system.

The UDP state message was received by a central database which collected network statistics for near real-time assessment and post mission analysis. similar to figure (2.3) the hardware architecture implemented for each of the mobile agents included the Persistent Systems mesh relay radio, various types of PC-104 stacks for operating the secondary controller and the SNMP agent. Figure (2.4) shows a screen capture of one of the ScanEagle data loggers.

Chapter seven provides an analysis of the network performance. A relatively unique aspect of the UxV NCS approach is using the radio performance of the overall system to influence how it is controlled. Chapter six discusses our approach for communications optimization complete with additional signal data collected during experimentation with NPS CAVR ScanEagles at YPG in Arizona.

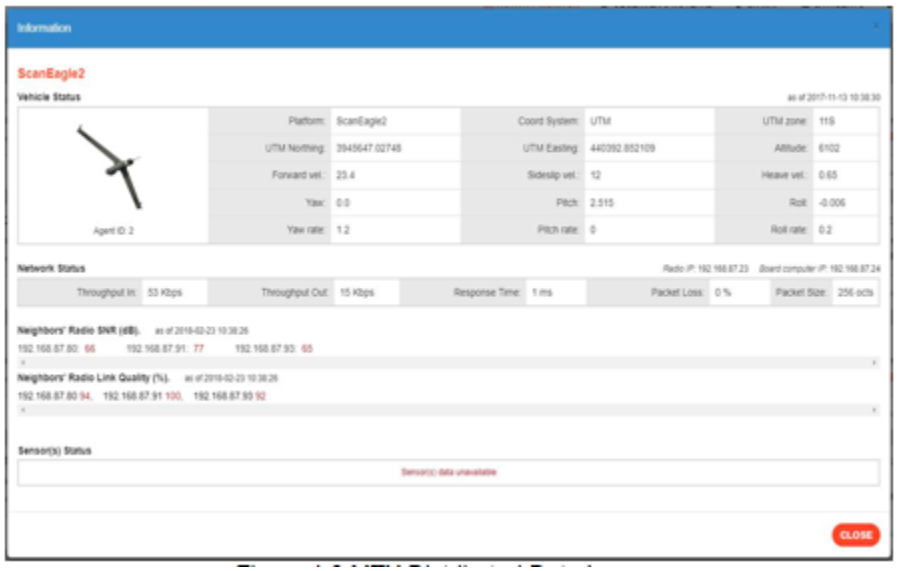

Figure 2.4: The image represents a snapshot of the visual display of networked information being sent to the central database. This display is available for each of the mobile agents within the NCS.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Experimentation and Research Initiatives

The MTX design emphasized a single operational scenario that could be used as realistic motivation for several research and experimentation “threads". Mobile UxV networks are potentially viable for a wide variety of military applications. For combining aerial, surface and underwater vehicles, perhaps no other mission area is as relevant as naval expeditionary warfare - which involves extending maritime operations ashore. Given that approximately 40% of the world’s population live within 100 kilometers of an ocean, a strong US naval expeditionary capability will continue to be critically important to the nation’s defense.

With this in mind, a small scale, relatively simple scenario was designed. It consisted of a NSW direct action exercise to recover a radiological device inside a building at the military operations in urban terrain (MOUT) site, on the northern portion of SCI. Key to the operational design is evaluating the potential of a UxV NCS to act as an autonomous force multiplier. If ground forces can be effectively supported by the UxV NCS with services like defensive overwatch, facilitating Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance (C4ISR) and distributed offensive fire support, it increases combat effectiveness and potentially decreases the operational and logistics footprint to support forward operations.

Mission planning and exercise execution of the experiment included the following phases:

1. **Intelligence preparation of the battlespace (IPB)** - In this phase of the operation, the UUVs conducted undersea bathymetric mapping to help determine the location to insert the NSW element. A SeaFox USV was used to receive the acoustic modem data from the REMUS vehicles and transmit the messages into the WiFi mesh network.
2. **Insertion** - The NSW element is autonomously inserted using the SeaFox USV. ScanEagles fly overhead and use the video cameras as sensory input for inserting at a location ashore that is clear of opposing forces. For both the IPB and insertion phases the SeaFox had a mounted radar for autonomous detection of surface craft.
3. **Infiltration** - The NSW element moves via ground to the objective. ScanEagles are overhead ensuring that selected ground routes are free from opposing forces (Note: there were no opposing forces during this experiment) and maintaining overhead video feed of the objective. The USVs are available for creating a communications link between the NSW element and at sea C2.
4. **Actions at the objective** - The NSW deploys the Shield AI quadrotor to conduct reconnaissance of the target area including the inside of the target building. This is followed by a clearing of the building and recovery of the (notional) radiological material. ScanEagles provide a video feed from above and provide a communications link to C2.
5. **Exfiltration** - The NSW element moves to the extraction site and awaits pickup by the USV. Again the ScanEagles provide overwatch and a communications bridge to C2 while the USV moves into position for rendezvous.

6. **Extraction** - The USV picks up the NSW element and the ScanEagles provide overwatch and act as a communications bridge.

Each of these phases have communications and sensing requirements that are different. This means that the topology (the spatial position and distance between the network agents) is changing over time. An important future consideration is not only optimizing the NCS as conditions change but also the path planning for each of the mobile agents as they transition from different temporal topologies.

3.1 Research initiatives

There were a total of six NPS faculty/student driven research initiatives. The objectives were principally selected based upon two criteria: (1) Address a fundamental aspect of command, control, sensing and communications associated with the mobile UxV NCS. (2) Select research initiatives that fit into the mission scenario. Below is a short description of each research initiative.

3.2 High-level autonomy for NCS optimization

The goal of this thread was to develop an high-level NCS controller for optimal, time-varying formations in near-real time. The controller provides agent position recommendations by assessing the current state of the system and providing timely solutions. It takes into consideration the vehicle dynamics, available sensor systems, communications and mission objectives to provide recommendations.

3.3 UxV NCS communications optimization

Communications can occur using different physical modalities. For this experimentation, there were examples of both acoustic and RF communications. The physical position and orientation of each of the agents influence the capabilities of the wireless network. Based on changing priorities, an important consideration is how to position agents to maximize the communications network effectiveness. Communications optimization can be accomplished with respect to a number of parameters including SNR, throughput in, throughput out, RTT and bit rate error. For this experimentation, the NPS researchers focused on building a SNR data model to better understand the degradation of SNR as a function of distance between multiple ground stations and a UAV.

3.4 Optimal UAV trajectories supporting road network interdiction

The objective was to generate optimal search and patrol patterns for the ScanEagles to search for opposing forces for the infiltration and exfiltration phases. The goal was to be able to provide trajectories for the ScanEagles taking into account road topography and mission timing.

3.5 Mobile mesh network performance and analysis

A critical component of the experimentation included a set of prioritized objectives to evaluate the field network performance at SCI. The methodology was to deliberately build out the network and conduct incremental testing. It started with the fixed (backbone) network and then slowly added in components of the mobile unmanned systems. It included the following steps:

1. Determine connectivity between the various nodes of the backbone network.
2. Put a single ScanEagle in the air and test radio connectivity between UAV and the Network Operation Center.
3. Test override control through the UAV secondary controller.
4. Gradually put multiple vehicles into the wireless network to test connectivity.
5. Fly multiple UAVs simultaneously to determine the communication bridging capabilities and limitations.
6. Conduct final testing of entire system for mission scenario.
7. Conduct mission scenario with the full NCS system, SEAL element and C3F ship support.

3.6 Cross-domain identification of road networks with Domain-Adapted Convolutional Neural Networks (DANNs)

Developing AI/ML techniques that enable processing sensor information at the edge of the network will be critical to managing the limited bandwidth of the wireless network. This research investigates the possibility of using satellite imagery to train a Convolutional Neural Network (CNN) to detect roads from a UAS video feed. The ability to successfully train a CNN from dissimilar data creates an important capability when forces have not previously been able to collect sensory data from the operational environment.

3.7 A novel approach for training CNN for 3D objects

Similar to the previous research. This initiative is designed to train a 3D CNN (using lidar data) from a 2D CNN classifier. From a ground vehicle, 2D camera images and 3D lidar point clouds are simultaneously collected. The 3D data is spatially segmented and correlated to a cropped 2D image. Each 2D image crop is then fed through a classifier to assign a label to the 3D pointcloud segment. This work aims to develop the next generation robotic mapping tools to enable mobile systems to develop path planning in more realistic environments.

3.8 Team leaders and participants

During the three weeks at SCI, MTX was conducted with all unmanned systems in participation and the full administrative and operations network implemented. This was a significant accomplishment. The ScanEagles were flown through the oversight of NPS Faculty Associate Aurelio Monarrez and a team that consisted of NAVSPECWAR Special Reconnaissance Team (SRT) ONE, Insitu and Special Operation Command (SOCOM) Air Boss. The Seafox USVs where successfully deployed off the coast by Dr. Sean Kragelund with NIWC support boats. Undersea bathymetric surveys by the NPS REMUS UUVs were conducted by representatives

from the Naval Experimental Diving Unit (NEDU). The Shield AI Quadrotor was deployed by the SEAL squad. The administrative mesh network was setup, maintained and monitored by the NPS CENETIX lab through the leadership of Dr. Alex Bordesky and Eugene Bourkarov. Critical to the overall success of the MTX experiment was NPS Faculty Associate Keri Williams. Her tireless efforts in coordination of facilities, organizations and logistics were deeply appreciated by all. Finally, this experimentation wouldn't have been possible without the leadership of Dr. Ray Buettner. As head of CRUSER, he provided valuable guidance, ensured adequate resources were made available and actively engaged Navy leadership about the experiment.

CHAPTER 4:
Adaptive Submodularity for a UxV Network Control System

This chapter is a summary of the thesis work of ENS Noah Wachlin and LT Brian Lowry [6], [7].

Fundamental to the control of a collaborative mobile network is to position the individual agents to maximize utility or mission effectiveness. Tenets associated with the design of the system include:

1. Making system control recommendations for placing agents (network nodes) in near-real time.
2. Quantifying and modeling system parameters like vehicle dynamics, sensing, communications and energy for development of an optimization function.
3. Providing sufficient system autonomy while simultaneously permitting human control when desired. This is called HOTL control.
4. Developing control methodologies that recognize the inherent challenges in warfare by maximizing system robustness and flexibility.
5. Designing and enabling the system autonomy, distributed sensing and information processing for rapid situational awareness.
6. Integrating cyber-security considerations as fundamental to the architectural design.

Our approach models the NCS as a graphical Linear Time Invariant (LTI) system that seeks near-real time position solutions (converted into control commands) for the agents based on maximizing a multiple objective optimization function using adaptive submodularity.

4.0.1 Problem statement

The UxV NCS is represented by the following equations:

$$\dot{\mathbf{x}} = f(\mathbf{x},\, \mathbf{u},\, \mathbf{m},\, \mathbf{z}, t) \tag{4.1}$$

$$\mathbf{x}(t) = [x_1(t),\, x_2(t),\, \ldots,\, x_N(t)]^T \tag{4.2}$$

$$\mathbf{u}(t) = [u_1(t),\, u_2(t),\, \ldots,\, u_N(t)]^T \tag{4.3}$$

$$\mathbf{m}(t) = [m_1(t),\, m_2(t),\, \ldots,\, m_N(t)]^T \tag{4.4}$$

$$\mathbf{z}(t) = [z_1(t),\, z_2(t),\, \ldots,\, z_N(t)]^T \tag{4.5}$$

$$\tag{4.6}$$

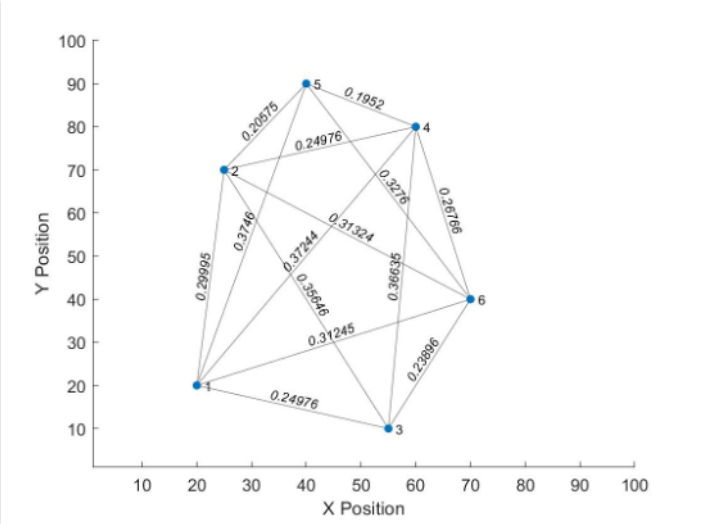

Figure 4.1: An example NCS graph with the nodes representing unmanned and manned systems and the edges representing the signal strength between valid communication paths.

where $\mathbf{x}(t)$, is a position vector, $\mathbf{u}(t)$ is a control input, $\mathbf{m}(t)$ is the map representation and $\mathbf{z}(t)$ is the sensory data all for the n-th vehicle in the NCS, with the initial conditions $(\mathbf{x}(t_0), \mathbf{u}(t_0), \mathbf{m}(t_0), \mathbf{t}(t_0)$, on the time interval $t = [t_0, T]$. The centralized control problem is to find a control vector $\mathbf{u}(t)$ that maximizes a vector quantity objective function (J) and the distributed control problem through a consensus building process that seeks to maximize a vector of objectives functions.

$$J_1(\mathbf{x_1}, \mathbf{u_1}, \mathbf{m_1}, t) \rightarrow max \tag{4.7}$$

$$\ldots \tag{4.8}$$

$$J_n(\mathbf{x_n}, \mathbf{u_n}, \mathbf{m_n}, t) \rightarrow max \tag{4.9}$$

Consider a UxV NCS consisting of N agents. We represent this NCS as an undirected graph (see figure (4.1)), $\mathbb{G} = (\mathbb{V}, \mathbb{E})$, which consists of a non-empty set of vertices $\mathbb{V} = \{1, 2, \ldots, n\}$ and an edge set $\mathbb{E} \subseteq \mathbb{V} \times \mathbb{V}$ where an edge in the graph is a pair of un-ordered nodes. The set $\mathbb{V}$ can include both unmanned and manned mobile platforms. Values assigned to an edge in the graph represents a fractional communications strength, $\mathbb{E}_{i,j} = (0, 1]$, where the maximum upper bound represents the highest quality of communications. The determination of the edge value is based on a communications function $\mathbb{E}_{i,j} = \mathbb{C}(\mathbf{x}_i, rp_i, \mathbf{x}_j, rp_j)$, where rp_n represents the radio parameters of the nth agent.

The graph Laplacian $\mathbf{L}(\mathbb{G})$, or simply $\mathbf{L}$ represents the connectivity for a multi-agent collaborative NCS [8]. The off-diagonal elements of $\mathbf{L}$ represent the interaction strength between agent i and j. We assume that $\mathbb{G}$ has a spanning tree, which implies that the system is stablizable

(see [9] for a proof). In our control we will also use the eigenvalues of $\mathbf{L}$, $\lambda_i, i \in 1,2,\ldots,N$. Note if G has a spanning tree, $\lambda_1 = 0$.

The dynamics of each agent $n = 1,2,\ldots,N$ is represented with double integrator particle dynamics, given by:

$$\dot{\mathbf{r}}_i(t) = \mathbf{v}_i(t) \tag{4.10}$$
$$\dot{\mathbf{v}}_i(t) = \mathbf{b}_i\mathbf{u}_i(t) \tag{4.11}$$

where $\mathbf{r}_i(t) = [x_i(t),\ y_i(t)]^{\mathrm{T}}$ and $\mathbf{v}_i(t) = [u_i(t),\ v_i(t)]^{\mathrm{T}}$ represent the two-dimensional position and velocity vectors respectively, and $\mathbf{b}_i\mathbf{u}_i(t) \in \mathbb{R}^2$ relates the control input to the acceleration. We represent the individual agent dynamics in state-space as a Linear Time Invariant (LTI) system, where the column vector $\mathbf{y}$ represents sensor measurements, the matrix $\mathbf{A}$ is the transition matrix, and the matrix $\mathbf{C}$ is the output matrix representing the linear relation between the state vector and the measurement vector.

$$\dot{\mathbf{x}}_i(t) = \mathbf{A}\mathbf{x}_i(t) + \mathbf{B}\mathbf{u}_i(t) \tag{4.12}$$
$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) \tag{4.13}$$

Normally we assume that the dynamics are consistent in the horizontal plane, but this is not the case for a heterogeneous mix of unmanned aerial, surface, ground and underwater vehicles. In this case we represent the dynamics of the NCS as

$$\dot{\mathbf{x}} = \mathbf{I}_N \otimes \mathbf{A}\mathbf{x} + \mathbf{I}_N \otimes \mathbf{B}\mathbf{u}, \tag{4.14}$$

where $\otimes$ indicates the Kronecker product, $\mathbf{x}(t) = [\mathbf{x}_1(t),\ \mathbf{x}_2(t),\ \ldots,\ \mathbf{x}_N(t)]^{\mathrm{T}}$,
and $\mathbf{u}(t) = [\mathbf{u}_1(t),\ \mathbf{u}_2(t),\ \ldots,\ \mathbf{u}_N(t)]^{\mathrm{T}}$.

We define the time-varying formation as $\mathbf{h}(t) = [\mathbf{h}_1(t),\ \mathbf{h}_2(t),\ \ldots,\mathbf{h}_N(t)]^{\mathrm{T}}$. Here $\mathbf{h}_i(t)$ is the piece-wise continuously differentiable vector, $\mathbf{h}_i(t) = [\mathbf{h}_{i,\mathbf{r}},\ \mathbf{h}_{i,\mathbf{v}}]^{\mathrm{T}}$ which indicates the position and velocity assigned to agent i in the formation. New formation configurations $\hat{\mathbf{h}}(t)$ are computed at discrete intervals separated by some Δt. The continuous function $\mathbf{h}(t)$ is then defined as a linear interpolation between the configurations $\hat{\mathbf{h}}(t)$ and $\hat{\mathbf{h}}(t+\Delta t)$.

From here we will simplify our notation by indicating $\mathbf{x}_i(t)$ with $\mathbf{x}_i$, $\mathbf{u}_i(t)$ with $\mathbf{u}_i$, $\mathbf{h}_i(t)$ with $\mathbf{h}_i$

etc. For the the system defined by Equation 4.12, the proposed the control law is:

$$\mathbf{u}_i = \mathbf{K}_1[\mathbf{x}_i - \mathbf{h}_i] + \mathbf{K}_2 \sum_{j=1}^{N} \left[w_{ij} \left(\mathbf{x}_j - \mathbf{h}_j \right) - (\mathbf{x}_i - \mathbf{h}_i) \right] + \dot{\mathbf{h}}_{i,\mathbf{v}}. \tag{4.15}$$

In this control law, the feedback gains $\mathbf{K}_1$ and $\mathbf{K}_2$ are used to shape the response of the time-varying formation center and the response of individual agents respectively. For homogeneous nodes, following [10] the complete system can then be defined as:

$$\dot{\mathbf{x}} = [\mathbf{I}_N \otimes (\mathbf{A} + \mathbf{B}\mathbf{K}_1) - \mathbf{L} \otimes (\mathbf{B}\mathbf{K}_2)]\mathbf{x} - [\mathbf{I}_N \otimes (\mathbf{B}\mathbf{K}_1) - \mathbf{L} \otimes (\mathbf{B}\mathbf{K}_2)]\mathbf{h} - [\mathbf{I}_N \otimes \mathbf{B}]\dot{\mathbf{h}}_{\mathbf{v}}. \tag{4.16}$$

4.1 Multi-objective optimization

The goal of the mobile NCS is to support manned, ground teams through the use of mobile sensing platforms for situational awareness. Commensurate with sensing is the ability to transmit information to users. The wireless communications network is dependent on the positioning of agents to facilitate network performance. Communication objectives could include maximizing information flow across a designated path or providing network robustness through the ability to maintain communication, even in the presence of agent loss. A third common objective is the ability to minimize total energy loss (or equivalently maximizing total system duration). The impact is to reduce system energy consumption minimizing total travel time and selective use of sensors.

In a bounded environment partitioned into cells of a designated resolution (figure 4.2), each agent can be positioned in any cell within the bounded area. Assuming for a moment that between two planning iterations all agents can reach any cell, there are a total of 2^V different possibilities. This has been long recognized as an non-deterministic polynomial-time hardness (NP-Hard) problem [11].

An approach for solving this problem is known as submodular optimization [12]. If a problem has the property of submodularity one can use a greedy selection process for determining positions for the system's agents [13]. This approach is useful because 1) It reduces the complexity of calculating solutions from NP-Hard to polynomial time. This has the critical benefit of being able to provide solutions in near-real time. 2) It has known convergence properties ensuring near-optimal solutions. In practice, the solutions are very close to optimal.

Define a *set function* $f : 2^V \rightarrow \mathbb{R}$ as a function that assigns a subset $S \subseteq V$ to some value $J(S)$ [12]. The UxV NCS is composed of N agents. The *ground set* V is the set of all possible locations where nodes may be positioned. If S is the set of N positions where the agents are placed, then the NCS with position set S is then assigned some utility $J(S)$.

Submodularity is a property of set functions commonly referred to as the property of diminish-

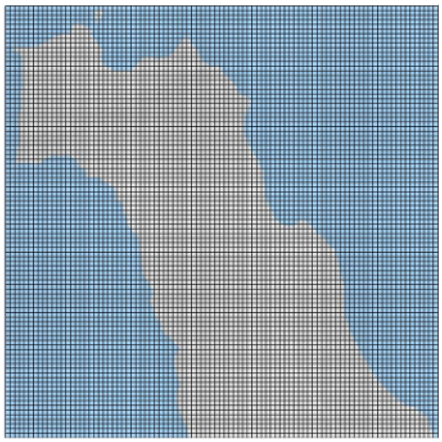

Figure 4.2: A superimposed grid on top of a map of the northern half of San Clemente Island, CA. It highlights the complexity of the problem for determining how to place each of the unmanned systems into the grid. It has been identified as an NP-Hard complexity problem

ing returns. It can be expressed in several ways, but we define it as follows:

Definition 4.1.1 (Submodularity). A set function $f : 2^{\mathbb{V}} \to \mathscr{R}$ is submodular if for any set A and B, with $A \subseteq B \subseteq V$, and any $s \notin B$,

$$f(A \cup s) + f(A) \geq f(B \cup s) - f(B). \tag{4.17}$$

The adding of element s to sets A and B has a larger impact on the smaller set. An additional important property for submodularity is that it is preserved for a linear combination of submodular functions. For submodular set functions $f_1, \ldots, f_n : 2^{\mathbb{V}} \to \mathbb{R}$ with nonnegative coefficients $\alpha_1, \ldots, \alpha_n$, the general optimization is given by:

$$f(S) = \sum_{i=1}^{n} \alpha_i f_i(S) \tag{4.18}$$

From this we can derive two important conclusions. First, if the utility function is nonnegative monotone submodular, it asymptotically approaches the optimal solution using a greedy algorithm that can be executed in polynomial time. Second, as the number of nodes increase, the approximation approaches the optimal solution based on the relation

$$f(S^*) \geq (1 - e^{-K/N}) \tag{4.19}$$

for positive integers N and K. As the number of nodes (K) increase the approximation approaches the optimal solution $f(S^*)$. The submodular optimization function f(S) is composed

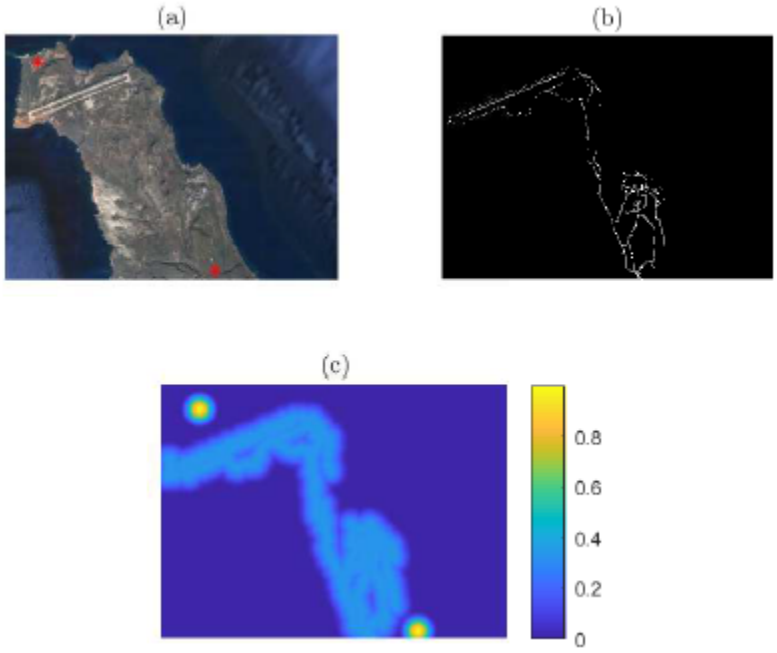

Figure 4.3: The sensing utility $\Phi_{i,j}$ assigned to UAV i for all locations $j = 1,2,\ldots,p$. (a) the satellite imagery is analyzed to extract (b) points of interest such as roads, and (c) an utility is assigned to each discretized point in accordance with a ranging function.

of components which quantifies the sensing ability $f_s(S)$ and the communication robustness of the network $f_r(S)$

$$f(S) = \alpha_s f_s(S) + \alpha_r f_r(S) \tag{4.20}$$

where α_s and α_r are nonnegative coefficients. The vehicle dynamic constraints are included within the utility function. It is done so in two ways: First, a boundary constraint is enforced by setting the values of the utility function to zero for locations that would place a node in violation of the physical constraints. Examples for the UAV and UUV a given in figures 4.3 and 4.4. Additionally, mobility constraints are placed within the optimization such that the agents can reach locations within their maximum velocity and heading rates.

4.2 Centralized submodularity

Our initial approach was a centralized solution. This has the liability that if the main calculating node is destroyed the system is uncontrolled. The main advantage is it's simplicity for calculating control commands. The following are simulation results using a centralized approach.

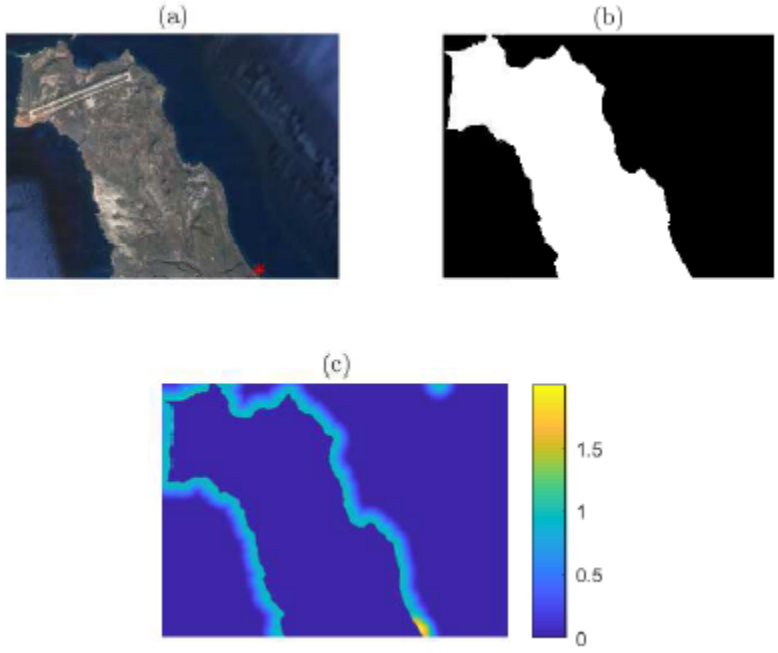

Figure 4.4: The sensing utility $\Phi_{i,j}$ assigned to UUV i for all locations $j = 1, 2, \ldots, p$. (a) the satellite imagery is analyzed to (b) eliminate the land, and (c) an utility is assigned to each discretized point in accordance with a ranging function.

4.2.1 Results and analysis

The intent of the optimization is an allocation of the system agents that effectively handle the mission parameters. The formation it generates must appear *logical*. This can be a difficult metric to quantify, but it should provide coverage over areas of interest with respect to communications and sensing. One of the ways to generate different network topologies is through the varying of the sensing and communications robustness gains (α_s and α_r).

In figure 4.5(a), all of the emphasis is placed on maximizing robust communications with respect to the virtual leaders (i.e., $\alpha_s = 0$ and $\alpha_r = 1$). This causes the agents to cluster in a formation that maximizes the number of communications links. Figure 4.5(b) demonstrates the polar opposite case, where $\alpha_s = 1$ and $\alpha_r = 0$. Here, the nodes are placed to simply maximize the each agent's ability to sense the environment. When equal emphasis is placed on each subfunction $\alpha_s = 1$ and $\alpha_r = 1$, the formation shown in figure 4.5(c) results.

The equal weighing of each utility subfunction did not provide sufficient coverage of the island. In figure 4.5(d) shows the case where $\alpha_s = 3$ and $\alpha_r = 1$. This seems to be a potentially appropriate balance of the competing objectives to sense the environment and communicate robustly. In future research, the goal will be to select optimal *policies*. An example of a policy, could be a specific set of subfunction weights. Rather than tuning the parameters based on observations, an adaptive submodularity approach could be used to adjust these weights based on a probabilistic

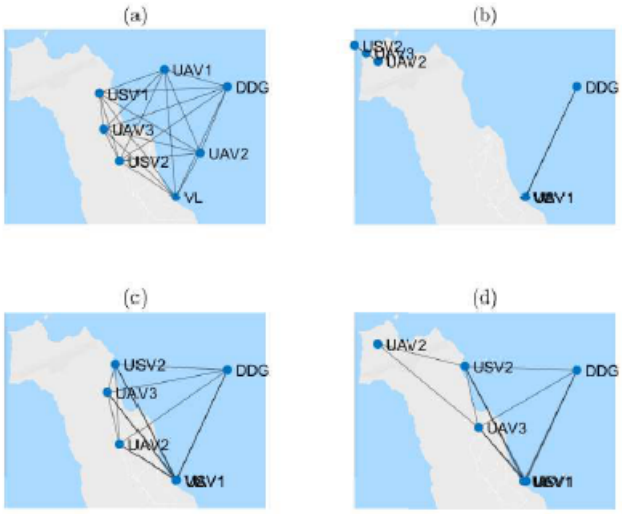

Figure 4.5: Topological changes are shown for a single mission phase. These changes are induced by varying the utility subfunction weights within the optimization function. In (a) $\alpha_s = 0$ and $\alpha_r = 1$. In (b) $\alpha_s = 1$ and $\alpha_r = 0$. In (c) $\alpha_s = 1$ and $\alpha_r = 1$. In (d) $\alpha_s = 3$ and $\alpha_r = 1$.

model of the world states and measurements.

For the various topologies of heterogeneous networks, we placed nodes according to the somewhat arbitrary rule of placing the most constrained (either slowest or having the smallest operating zone) first. The hypothesis was that this would result in better formations and that the order of placing nodes would significantly affect the network topology. Although more research is necessary, it appears this is not the case and that selecting different orders of UxVs impacted the network formation minimally. Figure 4.6 show a formation with all heterogeneous agents placed.

4.2.2 Submodular comparison

The selection of the submodular approach was due to the computation efficiency to potentially solve optimization problems in near real time. This section compares the submodular optimization approach with a brute force testing of all possibilities. Due to the sheer complexity of computing the absolute maximum value of the utility function, we limited the problem to finding the maximum utility for the placement of five nodes in a grid with only 130 grid points (in all other cases we used a grid with 1000 points).

A brute force algorithm was designed to exhaustively compute all possible configurations of the network. The optimal formation computed using the brute force algorithm required over 3.7 x 10^{10} computations of the utility function. This formation is shown in figure 4.7 with the grid overlaid on the map.

In comparison, the greedy algorithm only required 650 utility function evaluations and produced a very similar formation. However, the most telling result is the fact that $J(S^*) = 0.9895$, while

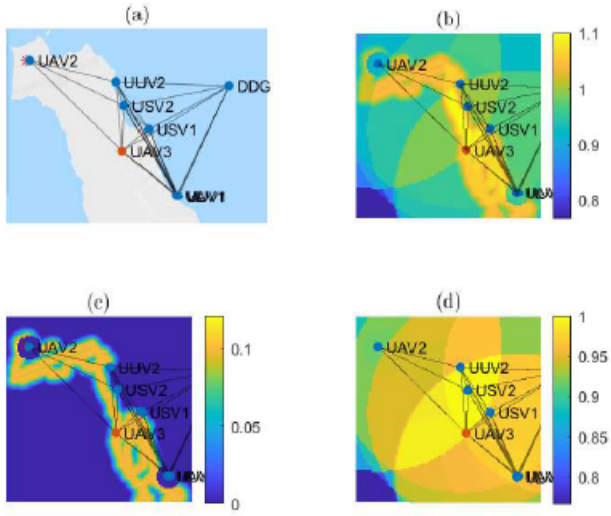

Figure 4.6: (a) The network is shown with the most recently-placed node highlighted in red. This node was placed at the point maximizing the utility function. (b) The total utility shown as an intensity map. (c) An intensity map of the sensing utility $\Phi_{i,j}$ shown for all locations j. (d) A heat map of the robustness utility $\Omega_{i,j}$ shown for all locations j.

the greedy algorithm found a formation with a nearly identical utility $J(S^*) = 0.9820$. This formation and grid is overlaid on the map of SCI in figure 4.7. Overall, we consider this a remarkable success and proof of concept.

4.3 Distributed submodularity

The previous section focused on a centralized approach. It is vulnerable if the centralized node responsible for computing solutions is destroyed. Alternatively, a distributed approach calculates the local optimization functions onboard each agent and through a consensus building approach determines how to position each of the network agents.

Our method of distributed submodular maximization is shown in figure 4.9. the method proceeds with each node repeating a three-step process. Each node in the NCS keeps track of the vector $\mathbf{X}$ containing the positions of all nodes, as well as the current total utility $\mathbf{J}$ and the change in $\mathbf{J}$ from the last round. This process to reposition the NCS then proceeds in the following phases:

1. **Local search**: (Lines 4-8) Each node concurrently and greedily evaluates a local area v for its optimal position. This is evaluated using the utility function described in Equation 4.20, assuming 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 affect. 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**: (Lines 9-10) All nodes then broadcast two pieces of information:

Figure 4.7: The optimal network configuration is shown for a simple grid containing 130 discretized points (the image shown is a subset of the grid containing 108 grid boxes). It requires 3.7 x 10^{10} evaluations of the utility function to determine the optimal configuration with utility $J(S^*) = 0.9895$.

the maximum contribution to total NCS utility, and the location it would reposition to if its contribution is greater than all other nodes.

3. **Evaluation and update** (Lines 11-19) The node which can affect the greatest change in total utility is designated and repositions itself, while all others update the position vector $\mathbf{X}$. Each node then calculates the new $\mathbf{J}$ and $\Delta\mathbf{J}$. The process repeats until $\Delta\mathbf{J} \leq 0$, indicating the near optimal topology has been reached.

4.4 Pareto optimality

As mentioned earlier, a key aspect of the approach is how to determine the gains that influence prioritization of the subfunctions of communications and sensing. It turns out that the multi-objective optimization function has multiple sets of gains that maximize the utility function. Figure 4.10 shows an example of a Pareto Front for the NCS at a given time, found using the Epsilon-Constraint method [14]. The solutions in blue are from maximizing the sensing objective function while constraining communications robustness, and the solutions in red are from the opposite - constraining sensing while maximizing communication robustness.

While each of the solutions are optimal, they each result in distinctly unique policies. Figure 4.11a shows the NCS positioned by maximizing sensing utility when constraining communications robustness to greater than 0.625, while the NCS figure 4.11b was positioned by constraining sensing utility to greater than 0.58 and maximizing communications robustness.

Figure 4.8: The near-optimal network configuration is shown for a small grid containing 130 discretized points. This only required 650 evaluations of the utility function to determine the optimal configuration with utility $J(S^*) = 0.9820$.

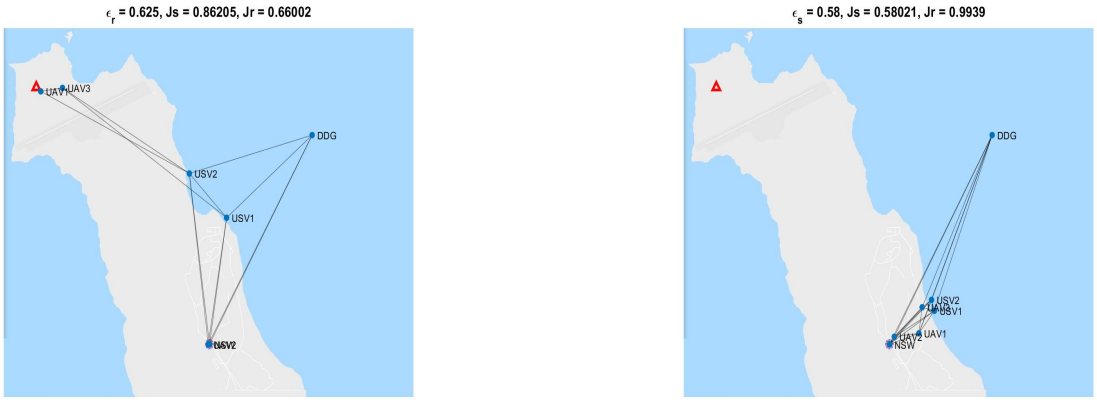

Figure 4.11: Two different Pareto-optimal NCS topologies. Locations near the target objective (red triangle) and NSW team (purple asterisk) have higher sensing utility values.

4.5 Adaptive distributed submodularity

A primary goal of the approach is to ensure that ground forces can easily control the NCS. Given the complex nature of combat, the methods of control need to be flexible. There are times when greater control and oversight is desired (i.e. during an initial ISR phase when exposure to opposing forces is minimal) and other times when the embedded autonomy handles the majority of decisions (i.e. during actions at the objective).

Our approach is to provide the control flexibility through linking a simple set of human commands that can significantly impact how the system performs. This is accomplished through a

Algorithm 1: Distributed Submodularity (DS) Method

Input: NCS size n, Current NCS topology $\mathbf{X}_o$, Sensing potential field $\mathbf{F}$, utility subfunction f_s and f_r, Local search area $v \subseteq \mathbf{V}$, Sensing and Communications Robustness utility weights α_s and α_r, NCS dynamics and communications constraints

Output: New NCS topology $\mathbf{X}_f$

$J = \alpha_s f_s(\mathbf{X}_o) + \alpha_r f_r(\mathbf{X}_o)$; // Initial NCS Utility

$\Delta J = J$; // Initialize

while $\Delta J > 0$ **do** // Each node $i = 1 : n$ performs concurrently

  for *each* $x \subseteq v$ *centered on* $X(i)$ **do** // from the perspective of node i

    $J_{NCS}(\mathbf{X}|X(i) = x)$ // NCS utility if node i positions at x

  end

  $[J_{max,i}, x_i] = max(J_{NCS})$; // Max utility, position for node i

  $\Delta J_i = J_{max,i} - J$;

  Broadcast $\Delta J_i, x_i$ to other nodes;

  Receive $\Delta J_j, x_j$ from all other nodes;

  Let k be the node that satisfies $max(\Delta J)$;

  if $i = k$ **then**

    Reposition, set $\mathbf{X}(i) = x_i$ // Update own position

  else

    Set $\mathbf{X}(k) = x_k$ // Update position of node that moved

  end

  $J_{new} = \alpha_s f_s(\mathbf{X}) + \alpha_r f_r(\mathbf{X})$; // Update NCS utility

  $\Delta J = J_{new} - J$;

  $J = J_{new}$;

end

$\mathbf{X}_f = \mathbf{X}$; // Final Optimal NCS Toplogy

Figure 4.9: The Distributed Submodularity Algorithm

limited set of commands which reflect command objective goals. These are considered system behaviors and initially include maximizing information flow, system duration, response time, disguise of intent and survey designated objective.

As described in the last section, the pareto optimal front is a combinatorial selection of optimization gains which results in an equivalent evaluation of the optimization function. An AI/ML technique known as Partially Observable Monte Carlo Processing (POMCP) evaluates the pareto optimal gains through the perspective of the above command objective goals. One of the pareto optimal solutions is then selected and used by the UxV NCS agents as the gain inputs

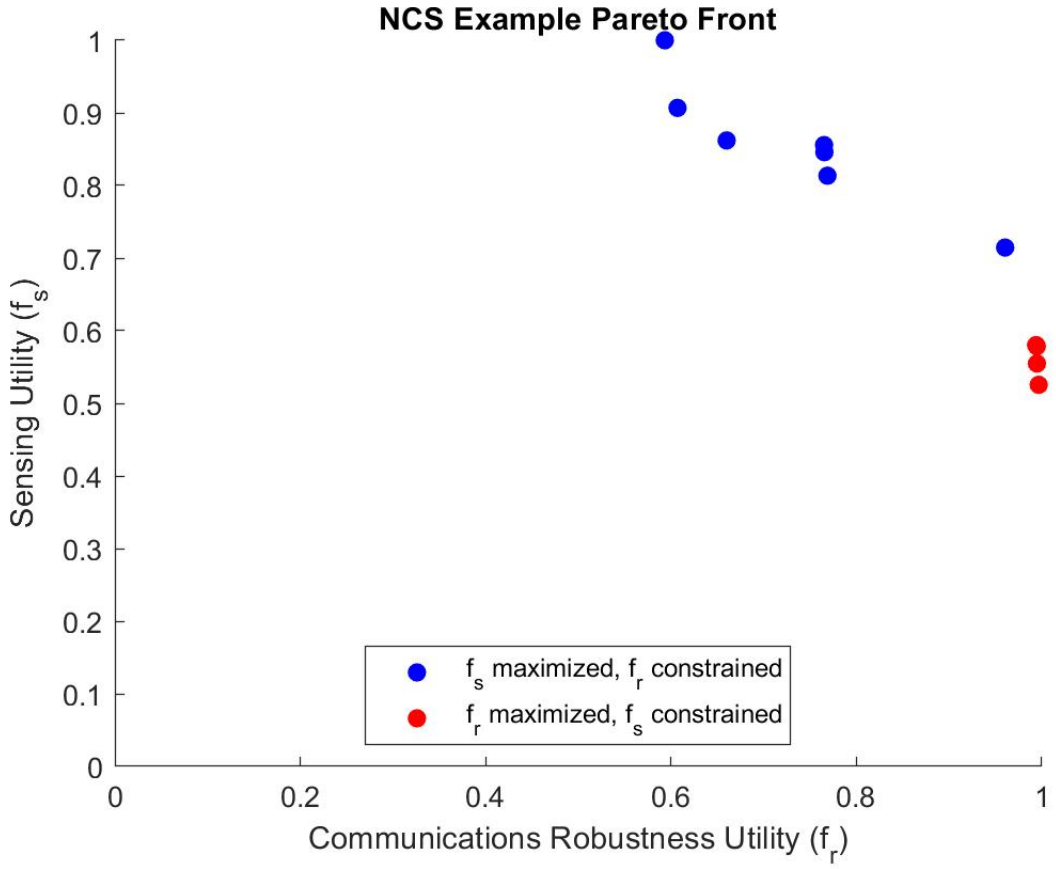

Figure 4.10: Example of a Pareto front.

for the submodular optimization function.

POMCP is a technique for efficiently searching through a dense configuration space [15]. It has been made popular, in part, from being a key component of the software architecture for AlphaGo success in defeating the Go world champion [16]. POMCP is a combination of a Partially Observable Markov Decision Process (POMDP) and a Monte Carlo Tree Search (MCTS). Figure 4.12 captures an example the planning process.

On the left-hand side of the diagram is a depiction of a time sequence of pareto front optimal solutions. On the top right-hand side, using the indicated epsilon constraint with respect to either communications (f_s) or sensing (f_c) shows the resulting behavior of these gains assigned to the distributed submodular network control systems. At the bottom right is the behavior metrics used for evaluating system performance.

In summary, this chapter presented a general methodology for placing agents optimally as a part of a UxV NCS. At the lower level a submodular optimization function for communications and sensing is used for determining where place agents to best support the ground team. The opti-

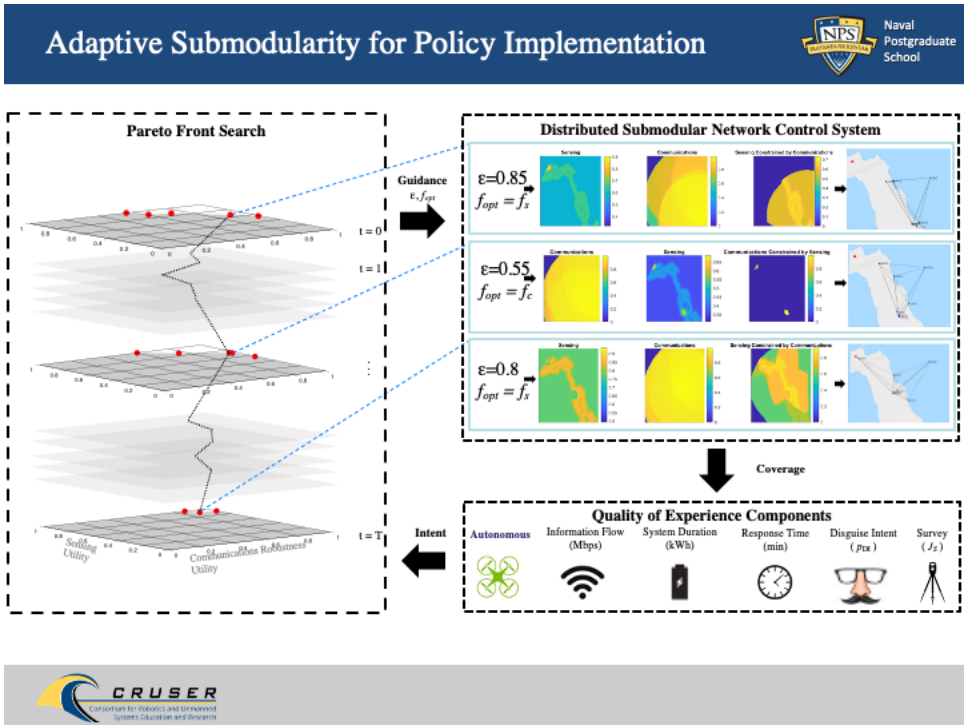

Figure 4.12: Adaptive Submodularity approach to distributed submodularity for a UxV NCS. On the left-hand side is a time sequence of optimal solutions for the combination of communications and sensing for the NCS. On the top right side is the result of the policy in terms of the NCS simulation based on a epsilon constraint of the pareto front supporting a notional SEAL direct action mission. The bottom right shows the initial components associated with the quality of experience metrics for commander's intent.

mization function has gains for weighing the influence of the parameters and there are several sets of gains that maximize the objective function. This pareto front set of solutions is used as input into a POMCP which evaluates them with respect to a separate set of metrics associated with the desired overall behavior of the system. This set of behaviors facilitates HOTL control of the UxV NCS.

CHAPTER 5:
UxV NCS Communications Optimization

The chapter is a summary of the thesis work of ENS Ben Keegan [17].

This research initiative explored autonomously positioning UAVs as wireless nodes in near-optimal locations to form robust, reliable communication links between static or slow-moving nodes, on land or at sea. The approach uses Optimal Spatial Estimation (OSE) to build a communications map relative to the static ground nodes. This map would then be used to find a local SNR extremum as an optimal loitering point for the UAV.

5.1 Background

In the last chapter, a component of the optimization for the positioning of networked nodes dealt with communications. The ability to dynamically consider communications strength between pairs of agents enables the UxV NCS to improve robustness and flexibility of the overall system, in part, by moving the agents to positions that ensure communications. Many radios include an API which permits understanding the quality of the communications link between various networked agents. Together with an understanding of the power and antenna design of the radio it can be used for a holistic approach to support temporally dynamic communication requirements.

5.2 Methodology

The following simple setup was used for both simulation and experimentation. Two static, wireless ground nodes were placed a significant distance apart on relatively flat land. Each of the nodes had omni-directional antennas with a 2 watt power output. The ScanEagle had either an omni-direction stub antenna on the underside of the fuselage or an antenna integrated into the vertical winglet. Each of the wireless radios were the Persistent System Mesh radios. The goal of the aircraft is to find the best location to facilitate communication between the two ground nodes. We would expect that to be at the midway point between the ground radios. For experimental testing the ground radios had laptop computers connected with ethernet cables and were recording signal statistics using the SNMP software described in chapter two. The ScanEagle had the secondary controller CPU which recorded the data via the SNMP software.

5.3 MTX San Clemente Island Tests

Initial preparatory experimentation was conducted at Camp Roberts, CA. But for both Camp Roberts and MTX at SCI the Persistent System MPU4 SISO radios did not perform at a minimum level to permit optimization. At SCI, data collection was additionally hampered by the fact that the minimum permitted aircraft altitude was 1,830 meters (6,000 feet), Furthermore, with radio antenna located on the belly of the aircraft it may have negatively impacted signal strength. Nevertheless, lessons learned from these experiments were instrumental in planning

follow-on experiments. This included the decision to use more capable MPU5 radios and locate the onboard radios antenna in the ScanEagle's right winglet instead of on its belly. Recall that the MPU5 radios (shown in figure 2.2) are MIMO and have three omni-directional antennas.

5.4 Camp Roberts, CA Tests

A second set of flights were conducted at Camp Roberts, CA using the MPU5 radios with a winglet antenna design. One wireless ground node was positioned at the ScanEagle launch site, while a second node was positioned approximately three kilometers away behind several hills which obstructed line of sight (LOS) communications. The ScanEagle was flown in specified patterns at target altitudes of 610, 1220, and 1830 meters MSL (2,000, 4,000, and 6,000 feet respectively), the last altitude was the maximum permissible flight ceiling of our designated airspace. The SNMP polling agent successfully collected SNR data on both ground nodes and on the ScanEagle.

The ScanEagle was flown in specified patterns at different altitudes above the center point between the two ground nodes. The patterns specified by the ground control operator were shaped like flower petals based on the assumption that they would provide the greatest amount of coverage. The patterns also ensured the UAV flew straight and level for the majority of the flight, to minimize the effect of bank angle. The center point was of particular interest because it was expected to coincide with the peak of the intersection of the SNR fields generated by both ground nodes. As the UAV was flown along its specified trajectory, SNR data for its communications links to each ground node was collected. The collected data for each link is presented in figures 5.1 and 5.2. It is displayed in Earth Centered, Earth-Fixed (ECEF) coordinates, where the SNR measured at each location is represented by the color bar shown.

These plots illustrate how the majority of SNR data collected for both communications links was measured between 20 dB and 40 dB. This suggests that this test did not explore the full range and capability of the MPU5 ground nodes (i.e. the ground nodes were placed too close to one another to sufficiently capture the degradation of SNR with distance for each node). Unfortunately, this test was the last scheduled experiment of the day, which did not afford the ability to repeat this experiment with the ground nodes spaced further apart. Ultimately, due to the limited range of this test, it was difficult to ascertain trends in the SNR data as a function of the UAV's flight altitude.

5.5 Yuma Proving Ground Tests

During the flight tests at YPG, in the ScanEagle executed specified flight patterns similar to the ones at Camp Roberts. These patterns were flown at different altitudes above both the center point between two ground nodes and directly above the ground node positioned at the launch site. Patterns included simple orbits, lawnmower tracks, and radial stars to ensure a wide coverage area and minimize the impact of aircraft orientation on measured SNR. The flat desert terrain presented negligible environmental obstructions to LOS wireless communication. Once again, the center point was of particular interest because we expected that the peak of

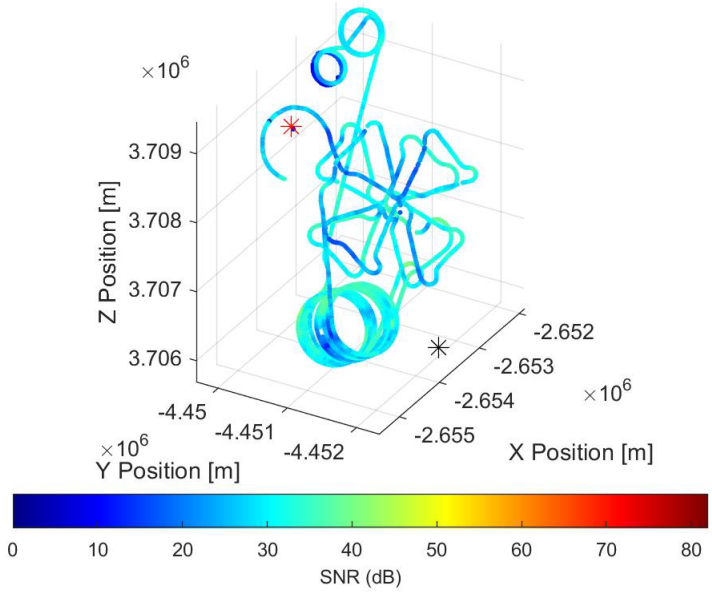

Figure 5.1: Camp Roberts SNR data for ScanEagle link to Node 1 (ECEF).

the intersection of the SNR fields generated by the ground nodes would be located there. As the UAV flew along its specified trajectory, SNR measurements were collected relative to each ground node. The collected data for each link is shown in figures 5.3 and 5.4.

As shown by the colorbar, a much greater range of SNR values was collected compared to the SNR data collected at Camp Roberts, spanning from 0 dB to over 45 dB, which was deemed to be an acceptable and realistic range. The test was planned to further explore the range and capability of the MPU5 ground nodes, so these nodes were positioned 7,000 meters apart, more than double the link distance tested in Camp Roberts. As shown in figures 5.3 and 5.4, there is a noticeable trend in the data for each node. As the UAV flew closer to each node, the SNR values associated with their respective communications links increased, and they decreased as the UAV flew further away. This trend supports our initial hypothesis that it is possible to use the SNR as a sensor measurement for optimally position the UAV as a communications relay.

The SNR data for each node, as measured by the ScanEagle, is plotted as a function of distance in figure 5.5. A wide range of data is shown across the 7,000-meter range as well as data collected at greater ranges for Ground Node 1 from the longer, westerly legs of the star patterns flown above the launch site. There is a clear and noticeable trend in SNR degradation as a function of increasing distance with some outliers for Ground Node 1. Based on the distance at which the outliers are located, it seems likely that these SNR spikes were captured prior to take-off or post-flight when the SNMP polling agent for Ground Node 1 was running and the UAV was on the ground.

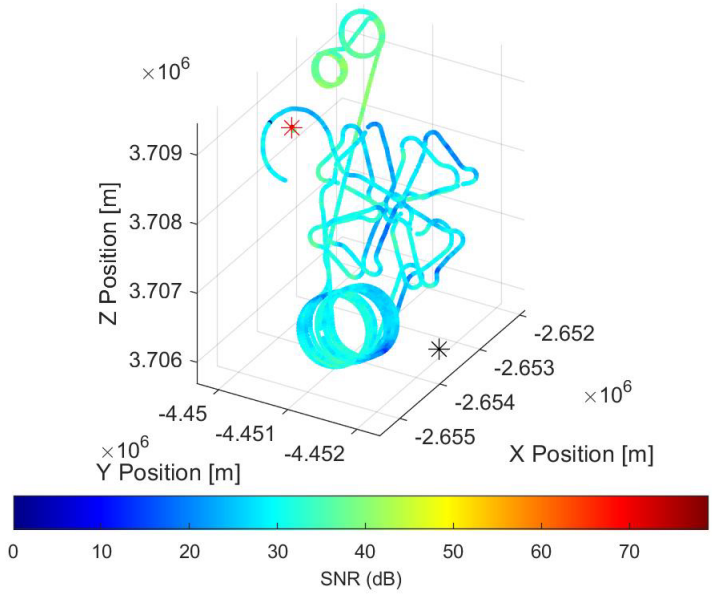

Figure 5.2: CampRoberts SNR data for ScanEagle link to Node 2 (ECEF).

5.6 Optimal spatial estimation

OSE involves the building of a spatial-temporal model of a Gaussian Random Field that can be used to describe the SNR degradation of ground-based radio. For simulations conducted for this chapter, a 10,000-meter by 10,000-meter field is discretized into 200-meter by 200-meter grid cells. Each cell represents a Gaussian distribution with a mean SNR value and an associated variance value. By using accumulated measurements, a spatial model can be built to predict the value of SNR as a function of both space and time.

In our application, we seek to enable a UAV to estimate the SNR fields of two ground nodes as a function of position and orientation, based upon actual, noisy SNR measurements. Our approach leverages a geostatistical, nonlinear, OSE technique also known as Kriging, to estimate the SNR field with spatially autocorrelated data. Key to the Kriging process is to understand the relationship between measurements. This is known as semivariance analysis and produces a functional relationship of the dissimilarity of measurement over increasing distance. This is used as a scaling factor in the interpolation process.

Figure 5.6 shows the results of the Kriging process with the YPG data set and figure 5.7 shows the Kriging estimate of the SNR field intersection.

5.7 Information theoretic path planning

The UAV needs a path to fly to build the SNR models and fly to the optimal position for communications between ground nodes. The path must enable the collection of measurements that will facilitate efficient model updates and reduce uncertainty associated with the SNR map. This can

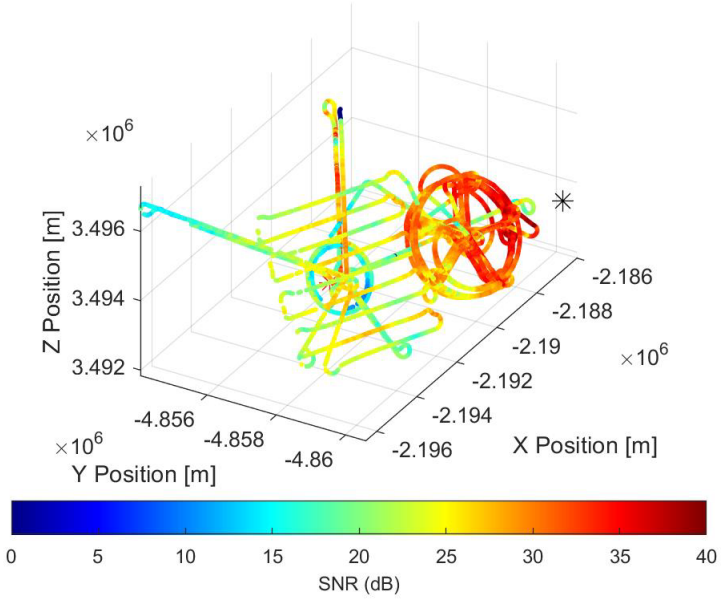

Figure 5.3: Yuma SNR data for ScanEagle link to Node 1 (ECEF).

be achieved by maximizing the rate of a given information gain metric between a current map and a predicted future map together with an energy metric based on the change of the SNR with respect to each of the ground nodes. The information gain metric is based on a Kullbeck-Leiber (KL) entropy measure.

The KL metric is defined the distance between probability distributions. The computed distance can also be used to measure inefficiency between an assumed probabilistic model, $q(x)$, and what is discovered through measurements to be an actual probability, $p(x)$, for a given random variable.

For this problem, we make the assumption that the SNR values for each ground node, computed at each discretized grid location, are single, independent random variables. Additionally, we assume that $q(x)$ describes the most current assumed variance models for the SNR maps generated by two ground nodes, and $p(x)$ describes the actual, updated models based on accumulated measurements. Assuming a) that the initial distribution of potential values x for random variable X is given by $q(x)$; and b) that the actual distribution, based on measurements, is $p(x)$; then the distance between these two distributions, the Kullback-Leibler divergence (KLD), is defined as

$$D_{KL}(p(x)\|q(x)) = \sum_{x\in X} p(x) log \frac{p(x)}{q(x)} \tag{5.1}$$

This definition can then be expanded to determine the KLD between two univariate Gaussian

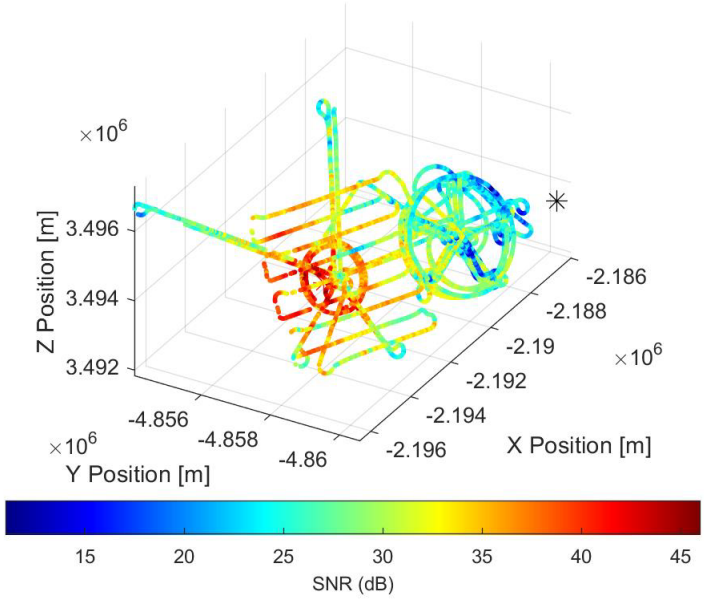

Figure 5.4: Yuma SNR data for ScanEagle link to Node 2 (ECEF).

distributions, each defined by a mean, μ , and a variance, σ , where $q(x) = N_{\mu_2,\sigma_2}$ and $p(x) = N_{\mu_1,\sigma_1}$. It is now defined as:

$$D_{KL}(p(x)\|q(x)) = log(\frac{\sigma_2}{\sigma_1}) + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2} \tag{5.2}$$

This value quantifies the difference between our initial assumption of the SNR distribution and the distribution calculated from accumulated measurements of the random variable. By accumulating SNR measurements for each ground node, and updating their assumed probability distribution, each measurement has a quantifiable value in terms of the information gain it provides about the random variable. Therefore, by evaluating the KLD between the current assumed Gaussian process model and an updated Gaussian process model based on predicted measurements computed using a measurement model of the random variable, the predicted information gain along each candidate UAV trajectory can be evaluated. In order to optimally explore the map and reduce its uncertainty, the candidate trajectory with the greatest predicted information gain can then be chosen.

In addition to determining the trajectory with the greatest information gain, a second objective is to determine which candidate trajectory affords the greatest change in energy. The energy metric in this path planning problem is based on electromagnetic energy associated with SNR. Specifically, the energy value along a candidate trajectory is computed as the absolute change in SNR magnitude with respect to path length for the intersection of the two SNR fields. The intersection of the two SNR fields, Intersection E , is simply computed as

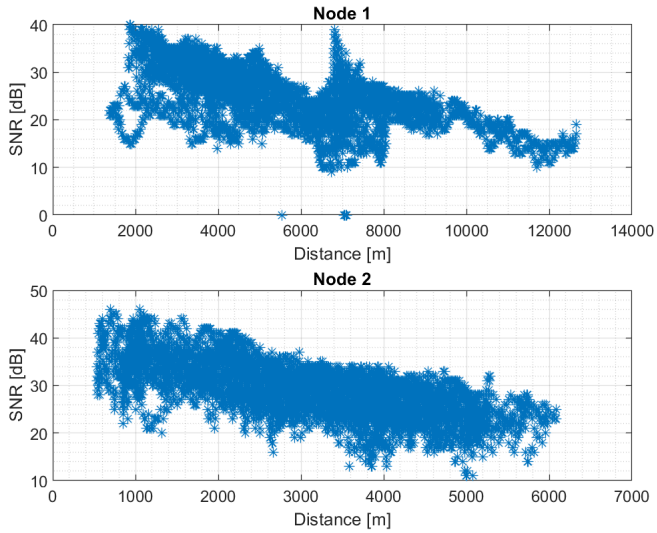

Figure 5.5: Yuma SNR data vs. distance from each ground node.

$$E_{intersection} = min(SNR_{Field1}, SNR_{Field2}) \tag{5.3}$$

The energy gain computed along a particular segment of a trajectory is then computed as:

$$|\frac{dE_{Intersection}}{dx}| = |\frac{E_2 - E_1}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}}| \tag{5.4}$$

where E_1 and E_2 are SNR values for the intersection of two SNR fields computed at two discretized grid locations along a particular segment of a candidate trajectory, and $|\frac{dE_{Intersection}}{dx}|$ is the absolute change in SNR magnitude for the intersection over the course of that trajectory segment. When examining the SNR fields generated by two ground nodes, the net energy value computed from both fields is the greatest in the intersecting region where the slope of the combined change in SNR over distance for both nodes is greatest. This is also realized by examining the intersection of the two fields, whose slope is greatest around its peak. Therefore, when this metric is combined with information gain, the UAV will not only choose the path that will afford the greatest amount of information gain, but it will also be inclined to choose the path that guides it towards the intersecting region of the ground nodes' SNR fields. Discovering this region is one of the steady-state goals for this path planner, once exploration has sufficiently reduced the uncertainty in the underlying SNR models.

Considering both information gain and energy change as well as feasibility with regards to UAV dynamics, the resulting information-theoretic objective function used to evaluate candidate trajectories is defined as:

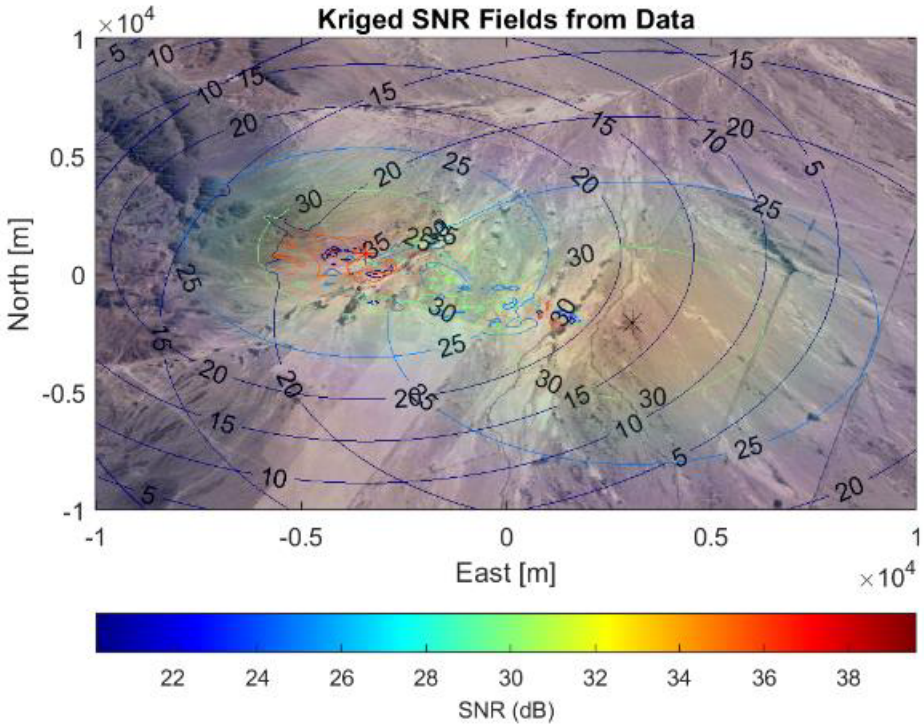

Figure 5.6: Yuma Estimate of SNR Fields.

$$J = \sum_{t=0}^{t=t_f} k_{IG}(D_{KL,Node1} + D_{KL,Node2}) + k_E \left| \frac{dE}{dx} \right|$$

where k_{IG} and k_E are gains associated with the information gain and change in energy along a candidate trajectory segment. The variables $D_{KL,Node1}$ and $D_{KL,Node2}$ are the predicted information gains for each SNR field based on predicted measurements taken while traveling between two discretized grid locations along a particular segment of the candidate trajectory. These grid locations are used to compute updated SNR maps for each node along each candidate trajectory segment. The values computed for energy and information gain are then summed along the entire length of each candidate trajectory to compute the objective function **J** for each one. The candidate trajectory with the greatest objective value is selected for execution. Figure 5.8 gives an example path based on the Yuma dataset.

In summary, this chapter presented a methodology for optimization of communications. Initial experimentation at Camp Roberts and at SCI was unsuccessful due to the limited radio signal strength associated with the Persistent System MPU4 SISO radios. After updating the radios to MPU5 MIMO radios, subsequent testing at Camp Roberts and YPG validated the approach. Using the data set at YPG, an information-theoretic path planning algorithm was designed for

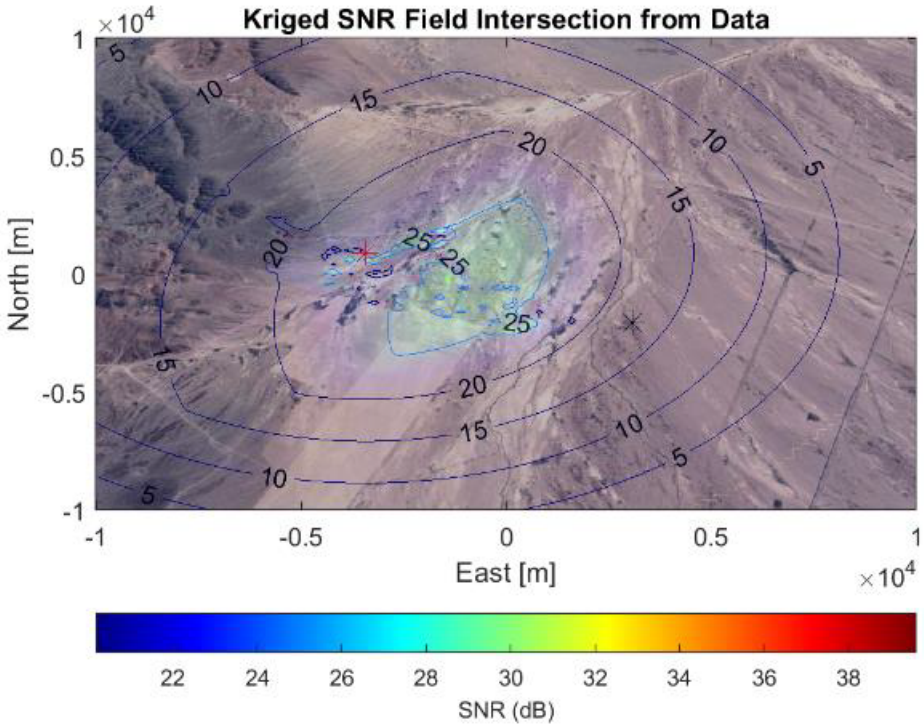

Figure 5.7: Kriging estimate of SNR field intersection.

dynamically generating a flight path that would find and loiter about the optimal communications position for two static ground nodes. The next steps to the research would be to expand on the number of vehicles in the network and include mobile and static nodes as part of the optimization.

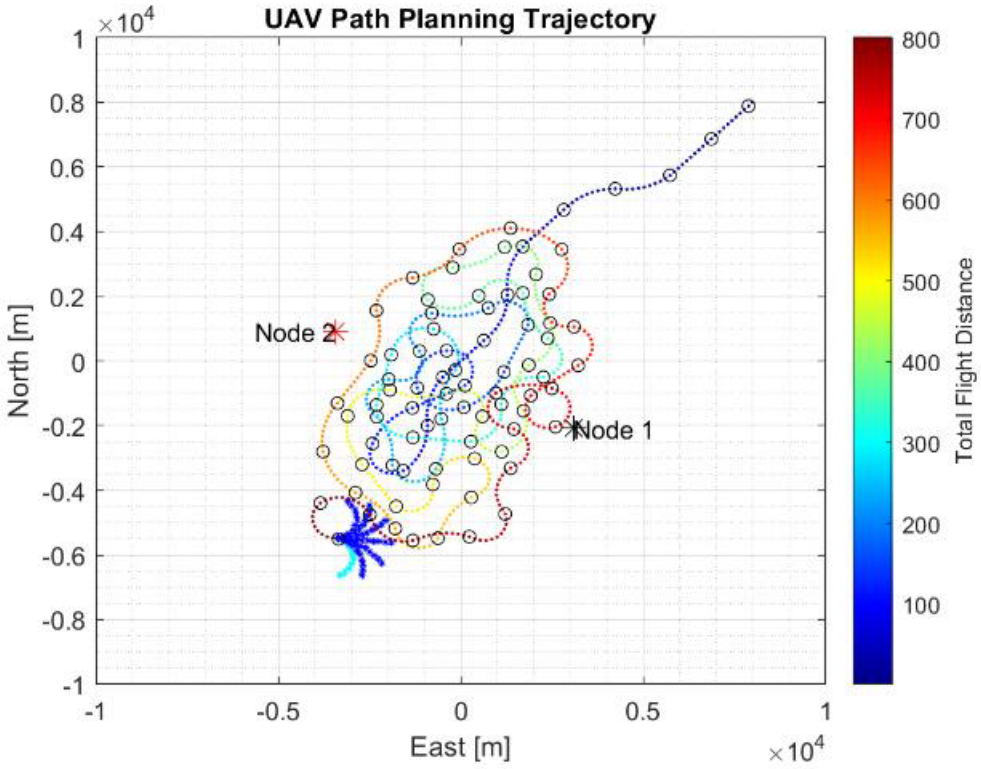

Figure 5.8: Kriging estimate of SNR field intersection.

CHAPTER 6:
Optimal UAV Trajectories to Support Road Network Interdiction

This initiative addresses how to design UAV trajectories that maximize the probability of detecting targets constrained to travel along a road network. The goal is to support friendly forces traveling to an overland objective through timely detection of opposing forces with available aerial search assets. This capability will allow friendly forces to select routes that avoid contact with the enemy, or in the worst case to alert them of a potential attack.

6.1 Optimal search

This takes the form of an optimal search problem, which has been studied extensively in applied mathematics, operations research and robotics. These problems have three major components (6.1).

1. A sensor/detection model for the probability of detecting a given target as a function of allocated search effort.
2. A searcher model describing how search effort can be allocated in a given scenario.
3. A target model which captures prior knowledge about the number, motion, and location of potential targets.

In the MTX scenario, one or more UAVs equipped with video cameras are deployed to detect opposing forces while a SEAL team transits to its mission objective. For this problem, a detection model was developed for the sensor's field of view, based on camera gimbal angles and zoom settings and the given UAV altitudes. Meanwhile, a searcher model described the sensor coverage traced by each UAV trajectory, subject to ScanEagle dynamic constraints. Finally, a target model encompassed the opposing forces' unknown starting locations, their initial probability distributions, and assumed speed profiles, with all motion constrained by the topography of the island's road network. The optimal control problem then solved for UAV control inputs and output trajectories which maximized the expected probability of detecting opposing forces before they could interdict the SEAL team. Typical control inputs are heading or turn rate commands, however this experiment required output trajectories to be discretized into a waypoint listing for ScanEagle to follow.

6.2 Detection model: camera sensor

This research initiative focused primarily on time-limited motion planning to optimally search a road network with UAVs. Therefore, even though the Hood Technology Alticam payload flown by the ScanEagle UAVs at MTX is quite sophisticated (`http://www.alticamvision.com/doc/Alticam_09EO1.pdf`) we utilized a simple detection model based on the camera's field of view for a fixed down-angle, zoom setting, and UAV altitude. Specifically, we

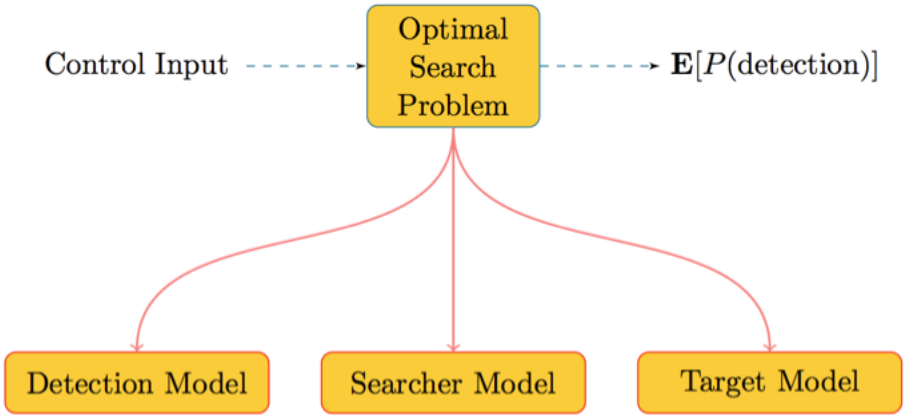

Figure 6.1: Main components of the optimal search problem.

used a Poisson scan model [18], [19], [20], [21] to produce a probabilistic detection rate for the UAV camera as described in [22], [23]. This detection model depends on 1) a target's position $y(t)$ within the camera field of view corresponding to a UAV's location $x(t)$, and 2) the amount of time the camera remains focused on the target:

$$r(x(t),y(t)) = \lambda \Phi \left(\frac{-\|x(t)-y(t)\|^2}{\sigma} \right) \tag{6.1}$$

where λ is the Poisson scan rate, Φ is the cumulative normal distribution function, σ provides a tunable variance parameter, and $\|\cdot\|$ is the Euclidean norm. The numerator of the argument models a decrease in effectiveness with increasing distance from the sensor field of view. Given this detection rate function, we can compute the probability that a searcher located at $\mathbf{x}(t) \in \mathbb{R}^{n_x}$ fails to detect a target located at $\mathbf{y}(t) \in \mathbb{R}^{n_y}$ during the time interval $[0,t]$ by using the well-known exponential non-detection law:

$$P_{ND}(t) = e^{-\int_0^t r(x(\tau),y(\tau))d\tau} \tag{6.2}$$

where $r(x(t),y(t))$ is the detection rate function [23, 24]

For MTX, this sensor model was tuned to have an effective detection radius of approximately 500 meters, with positive detections occurring every 10 seconds on average (figure 6.2). This is a very conservative estimate designed to simulate an automated process whereby computer vision algorithms detect and localize targets by analyzing each frame of the camera's video feed. This data would then be shared with other agents of the NCS, improving their ability to respond to dynamic events on the ground. While not implemented during the experiment, this capability is an area of active research for future MTX experimentation.

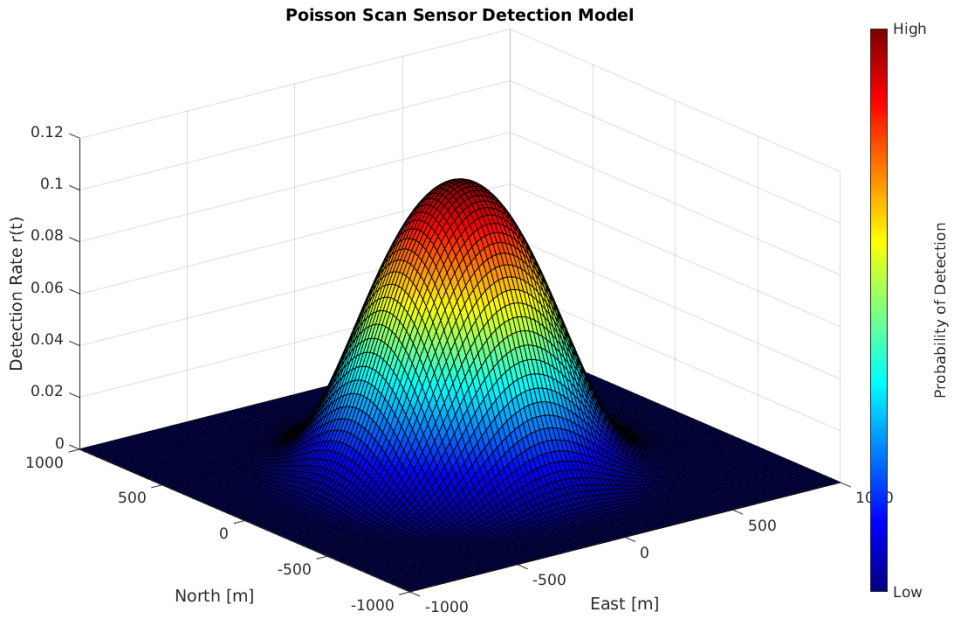

Figure 6.2: Example Detection Rate Function: Poisson Scan Model

6.3 Searcher model: UAV dynamics

The motion of the ScanEagle UAV can be described by a differential equation comprised of vehicle states $x(t) \in \mathbb{R}^{n_x}$ that evolve from an initial condition $x(0)$ under the influence of control inputs $u(t) \in \mathbb{R}^{n_u}$:

$$\dot{x}(t) = f(x(t), u(t), t), \quad x(0) = x_0 \tag{6.3}$$

This equation represents the dynamic constraints placed on the searcher model in our optimal control problem. In addition, each ScanEagle was confined to operate at a specific altitude under the airspace restrictions in place during this experiment. As a result, we assumed that UAVs moved on a two-dimensional plane at their designated altitude, with constant velocity of 25 meters per second and constrained by maximum turn rate limits of 10 degrees per second.

6.4 Target model: Opposing forces on road network

Opposing forces are the targets of this optimal search problem. These targets are modeled as dynamic agents that use a network of roadways to interdict the SEAL team under the following assumptions:

- Each opposing agent begins its motion from an uncertain starting location, with an assumed prior probability distribution.

- Opposing agents utilize ground vehicles that must travel on the established roadways of San Clemente Island.
- These vehicles change speed only as needed to achieve the highest velocity permitted along each road, which is constrained by each road's curvature.
- All agent trajectories terminate at the SEAL team's insertion point.

The first assumption formalizes the initial condition of each opposing agent as an uncertain parameter in our optimal search problem. The other three assumptions govern the motion of each agent from its initial to final condition, constrained by ground vehicle dynamics and the existing road network. Therefore, all potential target locations can be parameterized by the set of possible initial conditions $\omega \in \Omega \subset \mathbb{R}^{n_\omega}$. While an initial condition can be located anywhere along one of the island roadways, a prior probability density function $\phi(\omega)$ is used to represent any *a priori* information or "intel" we might have about the island. For example, high-traffic roads near airports, harbors, or troop barracks are more likely to contain opposing forces than more remote roads. The set of possible target trajectories are generated by sampling from the probability distribution of agent initial conditions and simulating ground vehicle motion along the road network until the agent reaches its destination. For this experiment, we utilized an algorithm described in [25] to extract a digital representation of each roadway from Google Maps and compute its curvature constraints for wheeled vehicles.

6.5 Optimal control problem

Since all target locations are conditioned on their uncertain starting locations, their "detection probabilities are conditional as well" [24]. We wish to minimize the expected value, over all possible starting locations, of the conditional probability of *not* detecting the opposing forces. For a finite time horizon T, we can define the following cost function J:

$$J = \mathbb{E}\{P_{ND}\} = \int_\Omega P_{ND}(T)\phi(\omega)d\omega = \int_\Omega e^{-\int_0^T r(x(\tau),y(\tau))d\tau} \tag{6.4}$$

Having defined a cost function and models for sensor detection rate, searcher dynamics, and target motion, we can now formulate this search problem as a nonstandard, or generalized optimal control (GenOC) problem [24].

$$\begin{aligned} \underset{\vec{u}(t)}{\text{minimize}} \quad & J = \mathbb{E}[P_{ND}] = \int_\Omega P_{ND}(T)\phi(\omega)d\omega = \int_\Omega e^{-\int_0^T r(x(\tau),y(\tau))d\tau} \\ \text{subject to} \quad & \dot{x} = f(x,u,t), \quad x(0) = x_0, \quad g_1(u) \le 0 \\ & \dot{y} = h(y,v,t), \quad y(0) = \omega, \quad g_2(v) \le 0 \end{aligned}$$

where the constraints on the states and controls for the searchers, $f(x,u,t)$ and $g_1(u)$, and targets, $h(y,v,t)$ and $g_2(v)$, have been described previously. The optimization is somewhat novel due to the additional integral over parameter space. A general approach has been developed for

discretizing the parameter space Ω using a set of M nodes $\{\omega_i\}_{i=1}^{M}$ and associated quadrature weights $\{\alpha_i\}_{i=1}^{M}$ to approximate the cost function as a finite sum [24]:

$$J^M = \sum_{i=1}^{M} \alpha_i [P_{ND}(t)] \phi(\omega_i) \tag{6.5}$$

This produces a family of standard optimal control problems which can be solved using numerical methods [26].

6.6 Results

MTX provided an opportunity to apply the NPS generalized optimal control (GenOC) framework in support of a tactically-relevant NSW mission scenario. Optimal search trajectories were generated for ScanEagle UAVs to surveil an island road network based on 1) the uncertain motion of opposing forces and 2) the intended progress of a SEAL team towards its mission objective. Figures 6.3 and 6.4 illustrate the steps in the optimal search process:

1. Problem Formulation (figure 6.3)
    - Obtain a digital map of the operating area on Scan Clemente Island (figure 6.3a).
    - Extract a digital representation of the area's traversable roadways (figure 6.3b).
    - Using available *a priori* information, apply an initial probability distribution for opposing forces' possible starting locations on the road network. The color map in figure 6.3c indicates the likelihood of an opposing agent's vehicle beginning the simulation at a given road location.
2. Optimal Search Execution: (figure 6.4)
    - Solve a generalized optimal control problem to compute sensor-based search trajectories for ScanEagle UAVs (figure 6.4a). Due to the computational effort required for current numerical algorithms, this step was not implemented onboard the ScanEagle secondary controller. Claire Walton generated optimal search trajectories for one- or two-UAV missions each night, and provided ScanEagle operators with waypoint mission files the next day.
    - ScanEagle UAVs fly discretized search trajectories (encoded as waypoint mission files) to minimize the residual risk of failing to detect opposing forces. The color map in figure 6.4a depicts this *a priori* risk for each section of the road network before the UAV search begins. Figure 6.4b illustrates this risk after the UAV search has been completed.
    - Search assets coordinate and respond to the SEAL team's movements along a path (light blue dashed line) toward its objective (red "X"). Figure 6.4c illustrates how a SEAL team can exploit the Network Control System's UAV surveillance assets to plan a route from an insertion point to an objective that minimizes risk from opposing forces. Note that this step did not take place in real-time during the MTX experiment; instead, this capability was simulated by executing a sequence of four

Figure 6.3: The steps involved in formulating the optimal search problem, beginning with a digital map of the operating area (a), extraction of the road network (b), and assigning *a priori* probabilities to opposing forces' starting locations (c).

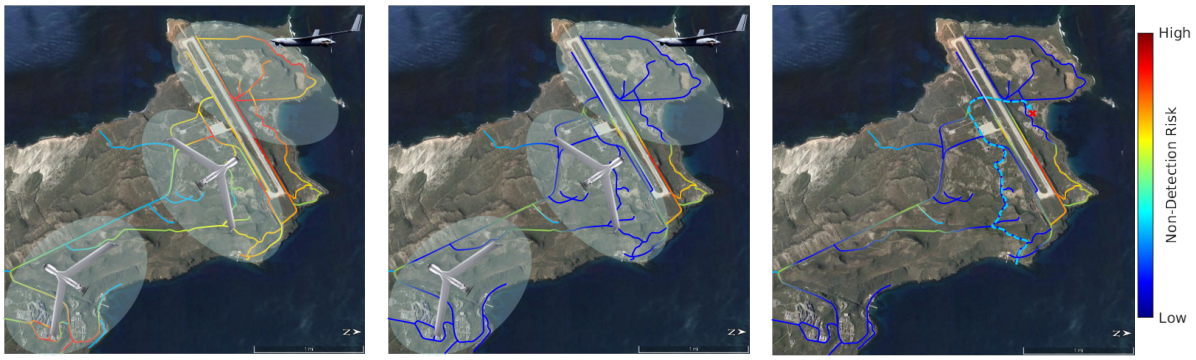

Figure 6.4: Minimizing risk from opposing forces by solving an optimal control problem to compute UAV search trajectories (a), flying the UAV's sensor over the road network to minimize risk of non-detection (b), and choosing a path for the SEAL team which avoids high-risk areas (c).

search phases to inspect the roads just ahead of the SEAL team's advance.

This process was applied to compute and fly optimal ScanEagle UAV search trajectories on two successive days during MTX. Due to aircraft availability, both demonstrations involved only one ScanEagle UAV. For the first demonstration, Figure (6.5 shows the optimal trajectory in light blue, represented as discrete waypoints, for searching the portion of the roadway shown in red. The UAV location at mission start is indicated with a green star. The search trajectory begins from a loitering orbit offshore and proceeds to cover the roadway with the UAV's camera sensor in order to minimize risk of target non-detection in the time available. Note that a manual process was utilized to transcribe algorithm-generated waypoints into the mission file delivered to ScanEagle operators. Unfortunately, a typo in the latitude field for one of the waypoints produced an erroneous excursion shown at (3.7 km East, 4.3 km North) in this figure.

The second demonstration repeated the previous scenario, but conducted the optimal search in four distinct phases. Figure 6.6 shows UAV trajectories for each phase, depicted in order from yellow, to green, to magenta, to light blue. This approach was designed to simulate a UAV road

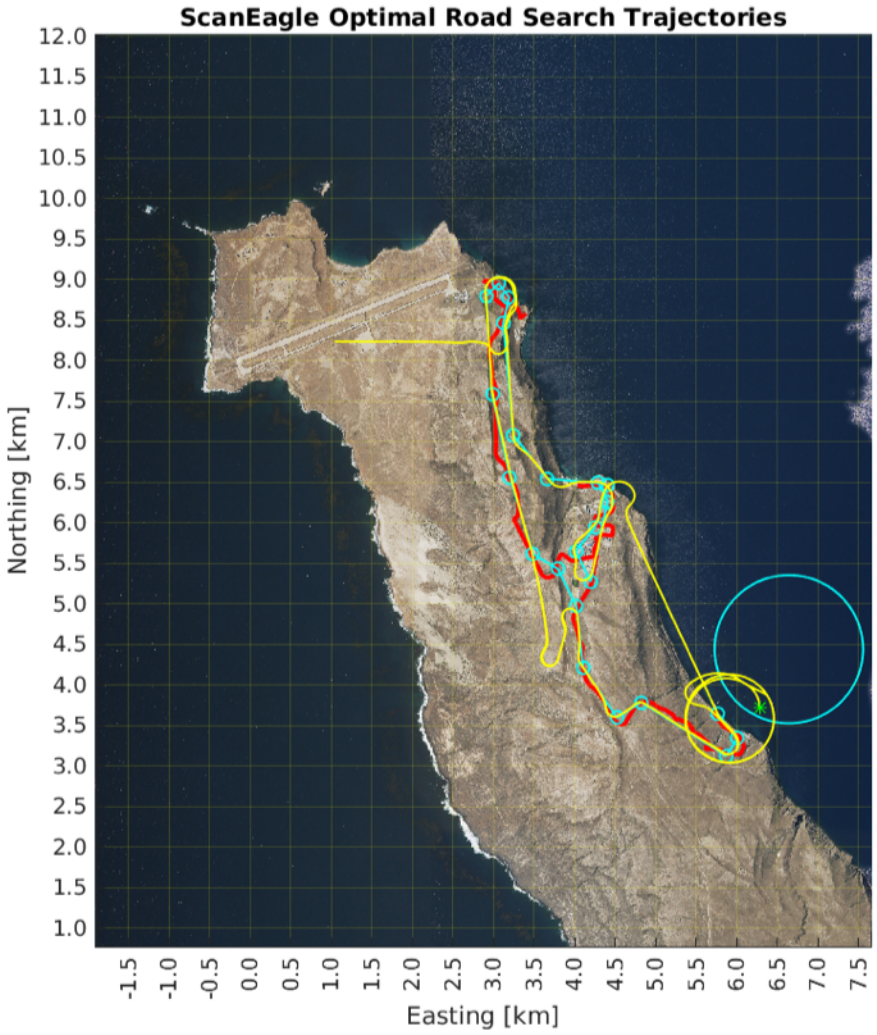

Figure 6.5: Optimal search trajectory flown by the ScanEagle UAV in support of a SEAL team's ingress from its insertion point in the south to its objective in the north.

search conducted in coordination with the SEAL team's actual movements. Conceptually, the UAV would search all roads approaching a major intersection prior to the SEAL team's arrival at that intersection. In this way, the SEAL team could select a path toward its objective which had already been searched and found free of opposing forces. As the SEAL team proceeded to the next phase of its mission, the UAV advanced to optimally search the roads ahead of the next intersection. Admittedly, this demonstration relied on pre-computed UAV trajectories in a carefully scripted scenario. Future research and experimentation will improve upon this initial effort by porting the NPS GenOC solution framework onto the ScanEagle secondary controller. Solving optimal search problems in near real-time will allow UAVs to plan and re-plan their

trajectories in response to the changing surveillance needs of networked ground forces.

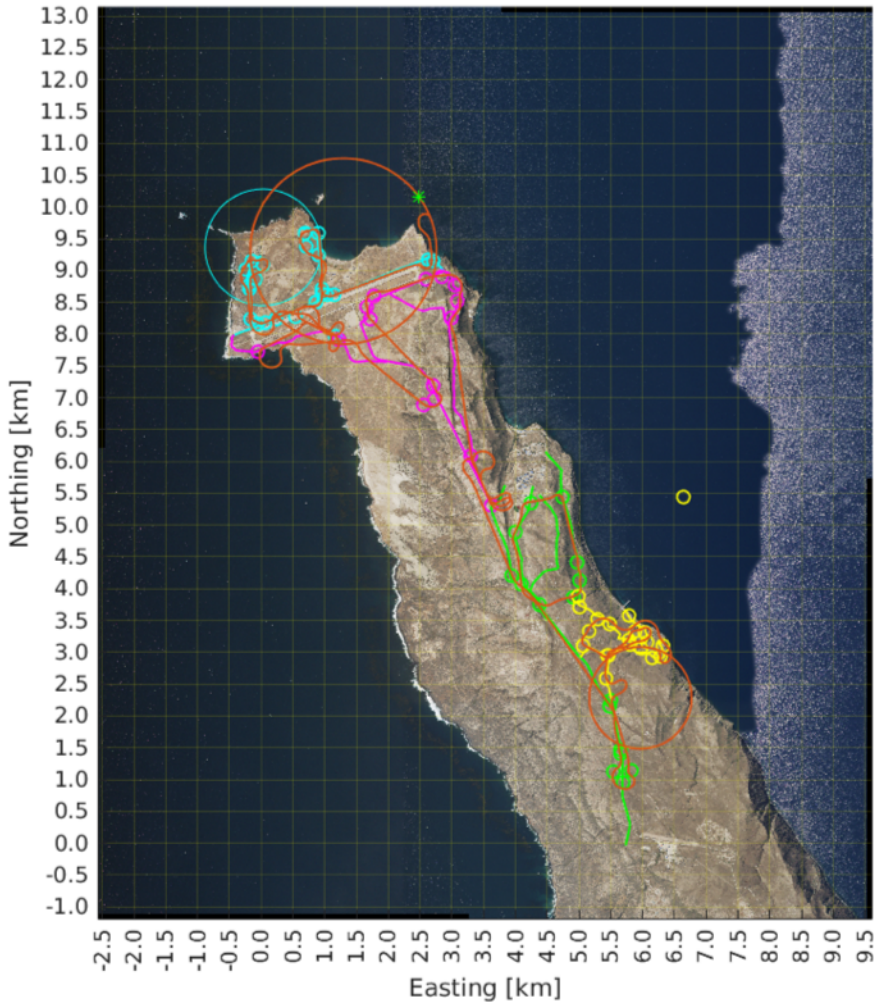

Figure 6.6: Optimal search trajectories flown by the ScanEagle UAV in four phases, from yellow, to green, to magenta, to light blue. The UAV supported a SEAL team's ingress from its insertion point in the south to its objective in the north.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 7:
Mesh Network Performance and Analysis

The following performance and analysis of the mesh network is an abridged version of the report originally created by COL Steve Mullins (ret.) and the NPS CENETIX lab.

7.1 Introduction

For MTX, the primary goal for deploying and operating the self-forming, mesh network was to bring together unmanned aerial, surface and underwater vehicles as a collaborative networked system for supporting a greater combined situational awareness in an austere environment. Figure 2.1 summarizes the experiment network setup and the backbone mesh across SCI. The main experiment was conducted from November 4, 2017 to November 13, 2017.

7.2 Overview of MTX networking tasks

To conduct this experiment, many tasks had to be completed before the experiment trials could begin. This included:

1. Definition of the operation
2. Development of a novel plug-in control network for monitoring and managing formation mesh
3. Development of SNMP agents (a novel hardware and software plug-in) for managing the UAV-USV nodes.
4. Design and setup of the mesh network to support operations and experimentation.
5. Design and setup of the Network Operations Center (NOC), including the integration of the new Activity Monitor and a set of the Network Management Systems

Another set of tasks were conducted during the MTX mission execution:

1. Network behavior data capture
2. Conducting network performance tests
3. Capturing network performance patterns

After the experiment, the following tasks were conducted:

1. Transfer of captured data for post processing analysis
2. Detecting network behavior patterns
3. Training ML algorithms and analyzing the results

7.3 CENETIX networking team

MTX Networking Lead: Dr Alex Bordetsky, Professor, Director, CENETIX

Research Team:

Short Title	Evaluate Capability to Support Full Mission Profile
Phase	Test XII (Monday, 13 Nov 2017)
Operational Level Problem	Employ a Network Control System of unmanned and manned nodes in support of an operational mission. Scenario will contain Offensive and Defensive areas of interest. • The Offensive Area of interest – Ensure that the mission movement routes and objective are under nearly constant surveillance and that through the distributed sensor network, if necessary, strike operations can be conducted against the designated target • The Defensive Area of Interest – Ensure that forces assaulting the target are protected against opposing forces.
Experiment Objective	The collective objective is the NSW/C3F/Network Control System themed experiment. FMP Phases: 1. ISR (UxV Intel Prep of the Battlespace) 2. Insertion (USV maritime) 3. Infiltration (transit to objective) 4. ORP (TUAV/actions on objective while at rally point) 5. OBJ (actions on objective) 6. Exfiltration (transit to extraction point) 7. Extraction (maritime)
Technical Objective	Examine technical solutions to autonomous vehicle support of tactical mission.
Research Questions	*1. How well does mesh NCS support UxVs' ISR/IPB?* *2. How well does mesh NCS support USV tactical team autonomous insertion?* *3. How well does mesh NCS support UAV surveillance for tactical team infiltration?* *4. How well does mesh NCS support tactical team TUAV surveillance at ORP?* *5. How well does mesh NCS support tactical team operations on the objective?* *6. How well does mesh NCS support UAV surveillance for tactical team exfiltration?* *7. How well does mesh NCS support tactical team extraction?*
Constraints	1. Weather conditions 2. Intervening terrain 3. Airspace limitations 4. Sea/kelp state

Figure 7.1: Network Operations Center (NOC): Experimentation Phase Plan Example

- Eugene Bourakov, Senior Researcher
- COL Steve Mullins (ret), Operations
- Maj Michel Cybulski, USMC, Network Operations
- LT Ryan Clapper, USN, Network Operations
- LT Josh Dennis, USN, Ship link operation
- LT Ryan Waddington, NOC Interfaces
- LT Beverly Crawford, USN, Network Operations
- LT Inna Stukova, USN, Network Operations
- Carsten Glose, Guest Researcher, Data Analysis/AI
- Jung Hun Ryu, Guest Researcher, Simulation

7.4 Operations

The scenario assumed that threat forces would try to take in a nuclear material into the littoral areas outside main cities in areas that have unprotected coastline and desirably also have limited infrastructure in terms of commercial communication networks. The mission scenarios included 5 phases from the IPB phase to the extraction phase. In each phase, an experimentation plan was developed that includes actions associated with the operational problem, experiment objective, technical objective, research questions and constraints. Figure 7.1 shows an example.

Research questions were translated into technical questions depending of the type of asset and technologies used. The questions were then broken into Measures of Performance (MoPs) for

1. *How well does mesh NCS support UxVs' ISR/IPB?*	
MoPs	**Data Collector**
a) Consistent connectivity into mesh NCS	Bourakov
b) Physical (SNR) layer through Activity Monitor and Solarwinds monitor c) IP (routing neighbors) layer through Wave Relay control view	Sipes/Crawford Stukova
2. *How well does mesh NCS support USV autonomous insertion of tactical team*	
MoPs	**Data Collector**
a) Consistent connectivity into mesh NCS	Bourakov
b) Physical (SNR) layer through activity monitor and Solarwinds monitor c) IP (routing neighbors) layer through Wave Relay control view	Sipes/Crawford Stukova
3. *How well does mesh NCS support UAV surveillance for tactical team infiltration?*	
MoPs	**Data Collector**
a) Consistent connectivity into mesh NCS	Bourakov
b) Physical (SNR) layer through Activity Monitor and Solarwinds monitor c) IP (routing neighbors) layer through Wave Relay control view	Sipes/Crawford Stukova
4. *How well does mesh NCS support tactical team TUAV surveillance at ORP?*	
MoPs	**Data Collector**
d) Consistent connectivity into mesh NCS	Bourakov
e) Physical (SNR) layer through Activity Monitor and Solarwinds monitor f) IP (routing neighbors) layer through Wave Relay control view	Sipes/Crawford Stukova
5. *How well does mesh NCS support tactical team operations on the objective?*	
MoPs	**Data Collector**
d) Consistent connectivity into mesh NCS	Bourakov
e) Physical (SNR) layer through activity monitor and Solarwinds monitor f) IP (routing neighbors) layer through Wave Relay control view	Sipes/Crawford Stukova

Figure 7.2: Network Operations Center (NOC): Experimentation Phase Plan Example

monitoring, collection and analysis (figure 7.2).

7.5 Implementation and link integration

Because it was important for the participants to share the situation awareness (SA) in real-time in the experiment, an SA server was implemented and used. Each participant connecting to the server could get SA information and each asset reported its status data to the server periodically. Data was collected by SNMP Agent specifically created for this project utilizing Node.js framework. Since Node.js is a platform-independent environment, SNMP Agent was able to run on Windows and Linux OS. For this particular experiment SNMP Agent was running on Raspberry PI 3 and Odroid microcomputers (Linux OS) added to the UAV and USV payloads. These agents recorded this data and also device-specific non-SNMP data such as the GPS position. The recorded data was uploaded to the central database after the experiment. It provided a unique repository of online and offline network performance data, a critical resource for subsequent replay of experiment and the post-processing analysis. In addition to this automated recorded data, significant events, interesting discoveries and relevant information that could not be collected automatically were recorded in textual form manually into the central database.

7.6 Network setup and configuration findings

One of the major network setup and deployment findings appeared to be the critical role of robust backbone, depicted in figure 2.1 by the RC1, Nots Pier Relay, VC3 Relay, VC3, and Ridge Relay nodes. By being initially placed for the experiment control, the set of backbone

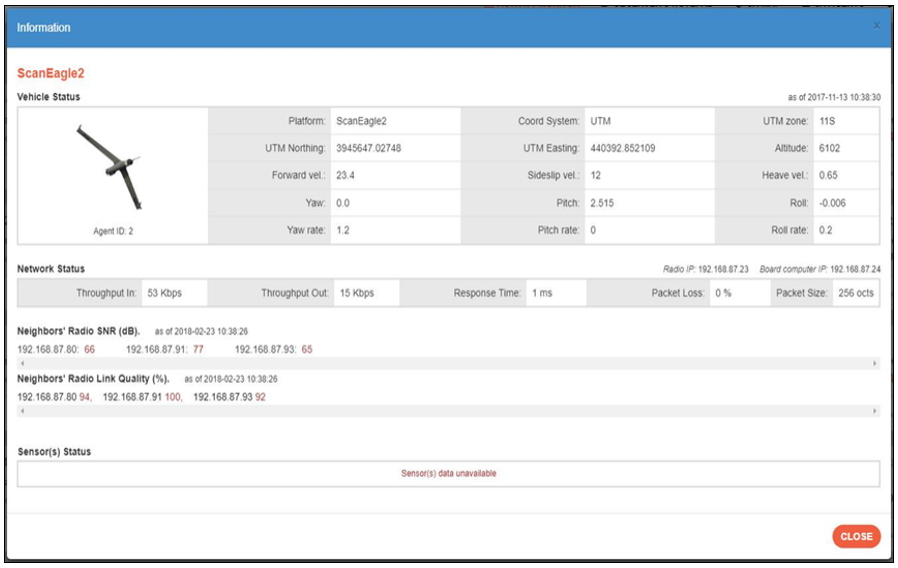

Figure 7.3: MTX Distributed Data Logger

nodes revealed a critical role of slowly moving or fixed mesh networking multi-hop architecture for moving and sharing information flows across highly mobile UAV, USV, UUV, and Blue Team unit clusters.

The other important findings include:

1. The need to use directional antennas between the critical relays, such as Ridge Relay, VC3 Relay, NOTS Pier and Main Objective areas.
2. The need to enable human re-routing of mesh routes across the backbone, subject to changing end-to-end network performance. This feature isn't allowed by the existing WR MPU-4 mesh radios. We expect substantial progress to be achieved in that direction by applying the Software Defined Radios to the mix.

7.7 Network operations findings

From the network operations perspective, the new Activity Monitor (fig. 7.4) as well as standard network management software (Solar Winds) was used. In certain instances, the proprietary management interface of nodes (such as the Wave Relay Management Interface) were directly accessed (shown in right corner at the bottom of fig. 7.5). Concurrently, the important observations were manually noted in Observer's Notepad collaborative log tool (fig. 7.6).

In the network design the stationary ground system was originally set for primarily situational awareness work. However, this stationary system emerged as the primary path for both ground and mobile units for the entirety of the experiment. The backbone became the core of the operational environment.

Another issue was the signal strength between several nodes and comparing how omni-directional

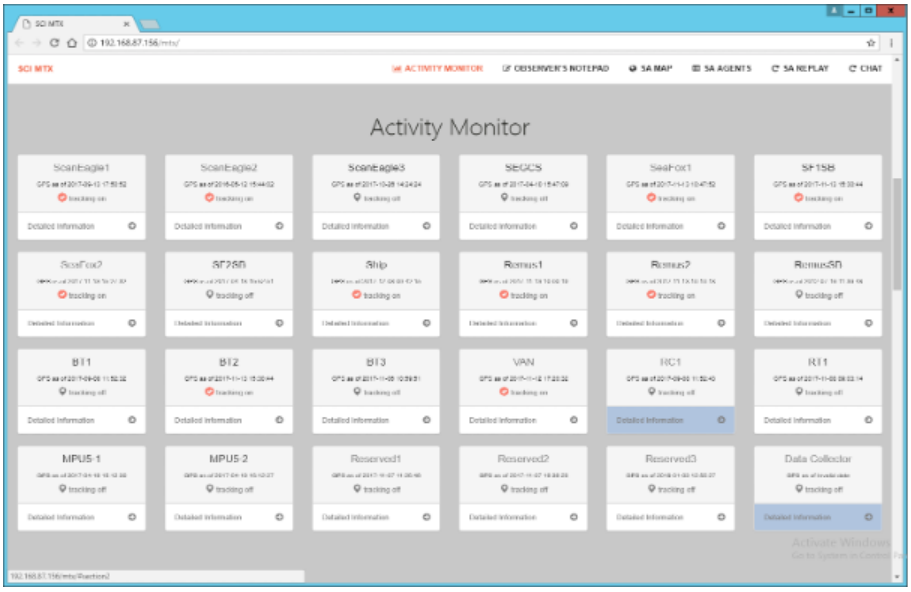

Figure 7.4: SNMP Plug-in Agents Activity Monitor

or directional antennas would improve the situation. It was noted that the signal between Ridge Relay and Frank2 was about seven megabits per second. By contrast, from Ridge Relay to the NOC it was barely one megabit. When using a directional antenna, the readings got slightly better. Sometimes a Ridge Relay improved the situation; however, a big improvement couldn't be seen. A stable link between the NOC and the ship could not be reached in the first two days of the experiment.

Some nodes such as the ScanEagle were connected but the performance was so low that it was never really used as an important relay. We then switched to static routing from Ridge Relay to Frank2 on the third day, which improved the situation, but static routing wasn't meeting our expectations. Anecdotally, it seemed to be that the routing algorithm of the Wave Relay nodes was using a shortest path algorithm, where path length is measured in terms of greatest signal degradation.

7.8 Unmanned systems formation networking findings

An instantaneous feedback on network performance, which was achieved through the NOC crew rapid response to the events, allowed the MTX team to optimize positioning and maneuvering the UxS MANET nodes. Figure 7.7 shows the success achieved in the ship-to-shore communication directly through in-land relay to as far as 27 miles offshore.

Correspondingly, figure 7.8 illustrates successful ship-to-shore ScanEagle relay maneuvering in providing the reachback to mission area unit.

Figure 7.9 illustrates optimizing effect of the ScanEagle grid pattern flying technique to enable stable communication with the blue force unit in the mission area.

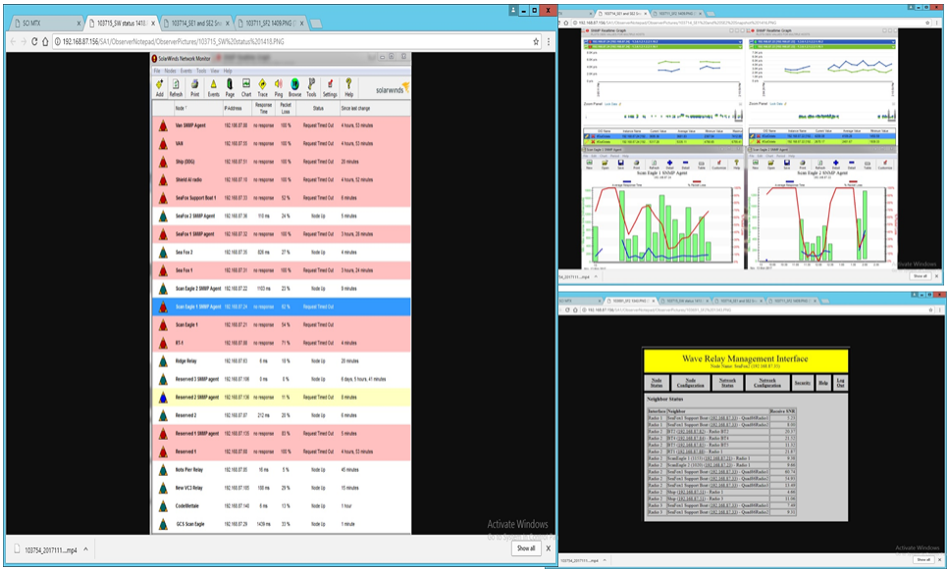

Figure 7.5: Solar Winds Network Management System and Wave Relay HTTP-interface

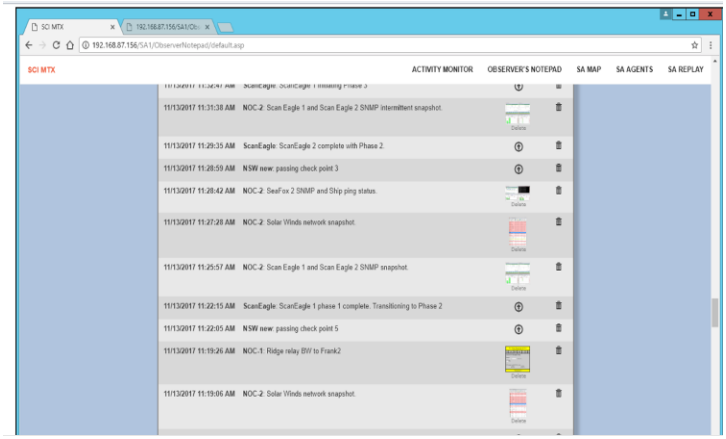

Figure 7.6: Observer's Notepad collaborative log tool

One other important finding of the MTX network operation appeared to be in the need of adopting the application flow to the intermittent behavior of the mission mesh network by scaling down the application flow between the capitol shop and the rest of the network. Shortly after the experiment the new version of bandwidth degradation resilient chat tool was created to address the finding. Figure 7.10 illustrates the result.

7.9 Captured data analysis

To ensure a high-quality dataset for analysis, we post-processed the recorded data and filtered out invalid or faulty data items. After this, we performed some basic statistical analysis to learn more about the nature of the data, find interesting properties or even identify features and/or a feature set to use as a basis for a machine learning system. Lastly, we used several machine

Figure 7.7: Ship Joining Network

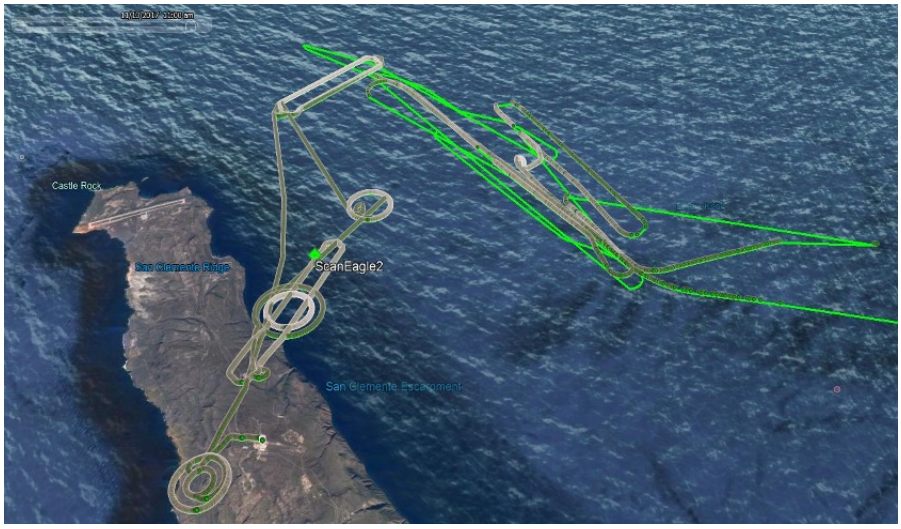

Figure 7.8: ScanEagle Extending Network to Ship

learning algorithms like rule-based, lazy learning, tree-based and support vector machines with this dataset to investigate whether machine learning can be used in this environment for predictive network performance.

We found strong statistical regularities in the recorded network data of the observed mesh network designed to support a tactical operation mission. These regular patterns are sufficient to predict relevant network management decision features related to unmanned system operation, subject to changing network performance and configuration conditions.

Figure 7.9: Supporting the Mission Area Unit Networking

Many classic machine learning algorithms master this learning problem. Except for Naïve Bayes, Logistic Regression, Support Vector Machines (SVM) and Ripper, performance does not differ between the learning algorithms very much. As 5% of the variance is lost by the Principle Component Analysis (PCA) transformation, we were surprised that the best learning algorithms could even archive a higher classification rate. We believe that our findings give some cause to expect that distributed autonomous network management systems for unmanned systems in tactical networks are in the realm of feasibility. On the contrary, the data also shows clearly a much higher degree of variance than it is seen in other network data. We assume that these irregularities are the special feature of tactical networks.

7.10 Conclusion

Overall, the numerous results accomplished during configuration and operation allowed the MTX team to move forward with optimizing patterns for flying and sailing the MTX UAV-USV-UUV formation. They demonstrated good potential of self-forming mesh network to support such operations and highlighted the major networking challenges to address in the next evolution of MTX experiment. One example of such evolution is to migrate to UAV-enabled backbone, enabled by tethered UAVs or/and UGV platform. This unconventional task as well as multi-platform mesh option, including the software-defined radios, is now on the drawing board for the next experiment planning.

Figure 7.10: MTX Distributed Data Logger: Low Bandwidth Chatting Room

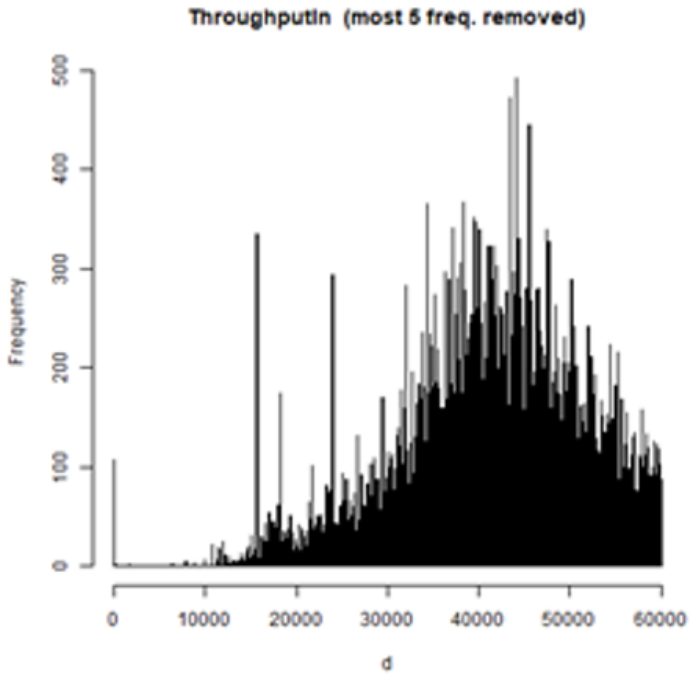

Figure 7.11: Throughput-In distribution

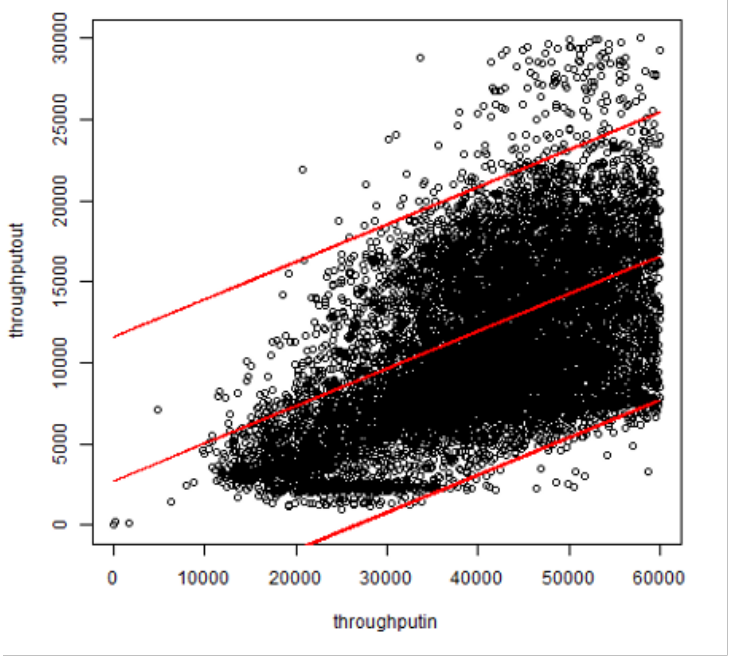

Figure 7.12: Input-Throughput vs. Output-Throughput

CHAPTER 8:
Cross-Domain Identification of Road Networks Using Domain Adversarial Neural Networks

This chapter is a summary of the thesis work of MAJ Teal Peterson [27].

CNNs have revolutionized sensor processing. The ability to identify features from image-based sensor systems permits greater perception and autonomy for unmanned systems. Still these CNNs need to be trained. They require prior data that typically perform better when the data is representative of the environment the vehicle is operating in. For the military this presents a conundrum. Frequently the vehicles are operating in new areas where prior data is not available.

This initiative seeks to change that paradigm by attempting to train neural networks with a dissimilar dataset such that the trained CNN is then used with a different sensor system to detect features. The technique to do this is known as Domain Adversarial Neural Networks (DANNs). Domain Adaptation (DA) is a method for leveraging the labeled data from one or more domains to train a discriminative classifier or predictor that will be evaluated against another domain that has little or no labeled data available. [28], [29].

Our application is the use of satellite imagery to train a CNN and then the use the CNN to detect roads from a ScanEagle UAV camera. The methodology is attractive, in part, because of the plethora of satellite data that could be used for training a CNN for feature detection. It is challenging because the difference in sensing platforms produce images that have different characteristics (e.g. range, resolution, warping from lens design, camera angle, scene occlusion). It is also useful since the trained CNN feature detector can be used for position estimation in GPS denied environments using techniques in SLAM.

8.1 Overview

The research consisted of five phases:

1. Define the source and target domains and build the datasets that will be used to train all CNNs and DANNs.
2. Train the source domain CNN on the generated source domain dataset.
3. Train the target domain CNN on the generated target domain dataset.
4. Evaluate the trained networks on the target domain test data.
5. Create a domain adapted CNN from the best performing CNNs to evaluate potential performance improvements provided by DANNs to this application.

Figures 8.1 and 8.2 illustrate the process. The first phase used existing satellite imagery and road vector data to extract road images within the Camp Roberts test range for the source domain. The primary challenge of building this dataset was ensuring accurate geo-rectified satellite im-

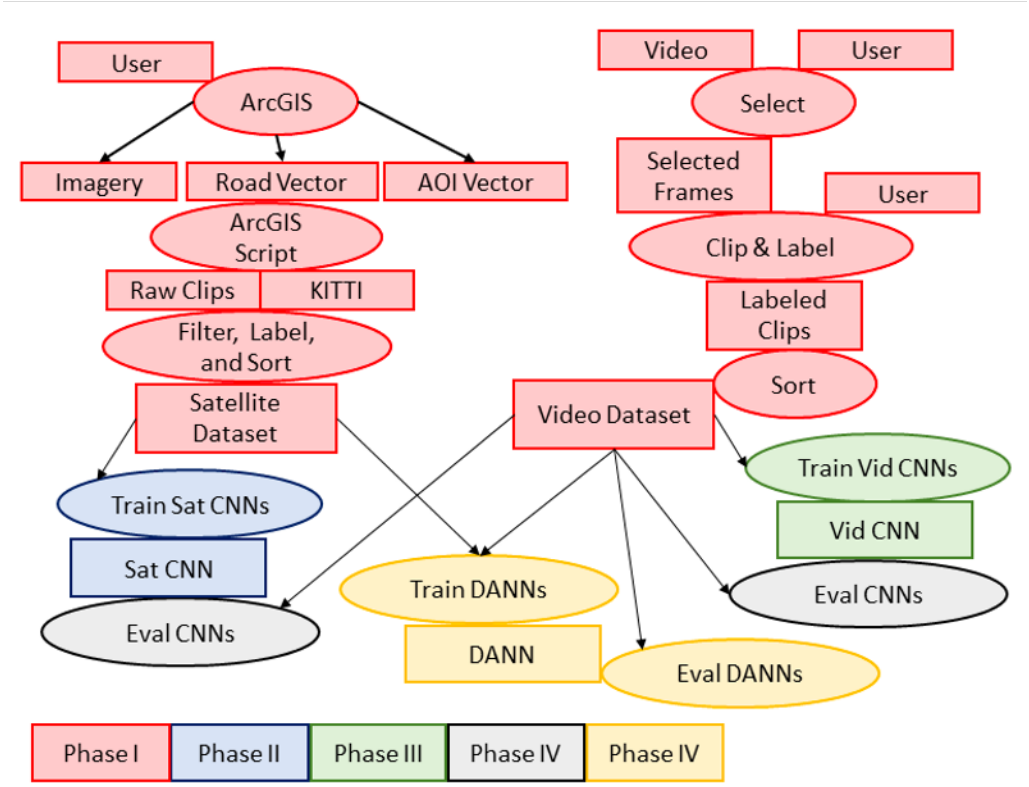

Figure 8.1: Pictured is the general data flow and processing steps, colored by phase. The domains and datasets are created in phase I. The satellite and UAS camera CNNs are trained in phases II and III. The upper and lower performance bounds are established in phase IV and the DANN is trained and performance is evaluated in phase V.

agery and road vector data. Satellite data is used to train a UAS-based CNN. Our target domain used footage from ScanEagle flights in the area. Footage was collected at 5000 and 3500 feet mean sea level (MSL) with the camera orientation facing straight down.

The second phase involved training the source domain CNNs using the generated source domain dataset. Training included three CNNs of varying type and structure. In the third phase, we trained target domain CNNs on the generated target domain dataset, also created in phase one. The fourth phase evaluated the trained networks on the target domain test data. During this fourth phase, the theoretical performance limit for DA was determined. The fifth phase created a domain adapted CNN from the best performing CNNs to evaluate potential performance improvements provided by DANNs to the application.

8.2 DANN background

A domain is a labeling function $f : X \rightarrow [0,1]$ paired with a matching distribution of data ($\mathscr{D}$), given as a set of inputs (X) to the function $\langle \mathscr{D}, f \rangle$. An example is a specific email user and their determination that an email is spam or not spam. Source and Target domains will be represented

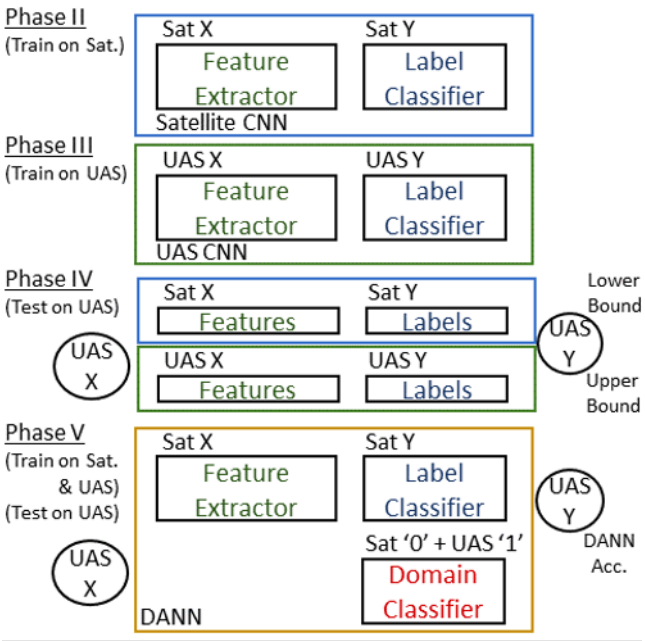

Figure 8.2: Several CNNs are trained in phases II and III. At the top, is the satellite CNN where the left side X represents the training set and the right side Y represents the validation set used to train them. The next level is the same for the UAS CNN. In phase IV, the models from phases II and III will be evaluated against the X/Y "outside" each model(UAS testset) to determine lower and upper performance bounds. In phase V, DANNs are similarly trained and evaluated against the UAS test set to determine DANN performance

by $\mathscr{D}_S$ and $\mathscr{D}_T$ respectively.

DA is a method for leveraging the labeled data of one or more domains (source domains) to train a discriminative classifier or predictor that will be evaluated against another domain (target domain, shifted from the original domain), that has little or no labeled data available. This process limits, or eliminates, the need to label enough training data to create a model for the target domain.

The performance limits of DA is bound by the performance of a source only, target only, or equally-weighted set of training data. In other words, we cannot expect a hypothesis, created through DA to perform better than a source domain trained classifier classifying source domain data, or a target domain trained classifier classifying target domain data.

Understanding the distance, or divergence, between domains in DA is an important part of understanding the performance of DA hypotheses. Single domain hypotheses provide a base for performance comparisons. As a target domain begins to diverge further from the source domain, we can and do expect the performance of that source-trained hypothesis to degrade when applied to that target domain. $\mathscr{H}$-divergence is a way to measure the divergence between the source and target domains.

8.2.1 $\mathscr{H}$-divergence

$\mathscr{H}$-Divergence is a measure devised by Ben-David [29], for situations where an estimate of divergence must be measured from an unlabeled, finite set of samples from each domain. It is defined as follows:

Definition 8.2.1 ($\mathscr{H}$-Divergence)**.** Given two domains distributions $\mathscr{D}_S^X$ and $\mathscr{D}_T^X$ over X, and hypothesis class $\mathscr{H}$, the $\mathscr{H}$-Divergence between $\mathscr{D}_S^X$ and $\mathscr{D}_T^X$ is:

$$d_{\mathscr{H}}(\mathscr{D}_S^X, \mathscr{D}_T^X) = 2 \sup_{\eta \in \mathscr{H}} \left| \Pr_{X \sim \mathscr{D}_S^X}[\eta(x) = 1] - \Pr_{X \sim \mathscr{D}_T^X}[\eta(x) = 1] \right|$$

$\mathscr{H}$-Divergence is the distance between source and target probability distributions of positively labeled inputs. The value varies between 0, representing identical source/target domains and 2 where the source and target domains are completely divergent.

8.3 DANN

DANNs, are a modification to existing deep learning frameworks that implements a trade-off between error on the source domain and a generalization of features used by that framework. By using this modification, the required deep feature mapping, label prediction, and domain "convergence" can be completed using existing methods for deep learning in one training process. Under the hood, DANNs perform label prediction just like any other feedforward NN. Specifically, raw inputs are fed into a network of perceptrons that generate, throughback-propagation, the deep features that optimize the performance of the final label predictor. This optimizes the performance of the label predictor as the first half of the aforementioned trade-off.

The unique addition to the DANN architecture is a domain classifier which operates in parallel to the standard label prediction process. As deep features are defined and fed into a label predictor, these deep features are also fed into this domain classifier. Ganin et al., established that reducing $\mathscr{H}$-Divergence reduces target error [28]. This domain classifier essentially determines for the DANN how much $\mathscr{H}$-Divergence is present across the deep features induced while training the label predictor. Throughback-propagation and a Gradient Reversal Layer (GRL) feedback allows the DANN to "forget", or un-weight, the domain specific deep features which cause the source and target domains to diverge.

It is important to note how DANNs, create a trade-off between source domain accuracy and a reduction in induced domain divergence. The performance of any NN is dependent on its ability to identify features, within its intended domain, that allow its predictor to separate the examples provided to it. DANNs improve their performance on their target domain, by forcing the model to "unlearn" the deep features used to classify the source domain only. Since the DANN label predictor has fewer deep features to rely on for labeling the source domain, we can expect a reduction in performance on the source domain.

While seemingly complex, DANN implementation can be boiled down to a domain classifier and a "minus one" (internal to the GRL). The GRL is placed between the domain classifier and the deep features common to it and the label predictor. During the forward pass of data, features remain unchanged heading into the domain classifier, as the loss from this forward pass is back-propagated, the gradient is reversed by a negative hyperparameter set prior to training. This GRL is the DANN component that directly computes the $\mathscr{H}$-Divergence and implements the reduction in $\mathscr{H}$-Divergence. It reduces the weights of features that allow for better domain classification. In other words, this implementation is essentially a regularizer that penalizes divergence of deep features from each domain. This regularizer is defined as:

$$R(\theta_f, \theta_d) = -\max_{u,z} \left[\frac{1}{n} \sum_{i=1}^{n} \mathscr{L}_d^i(\theta_f, \theta_d) + \frac{1}{n'} \sum_{i=1}^{N} \mathscr{L}_d^i(\theta_f, \theta_d) \right] \tag{8.1}$$

where θ_f is the set of deep features common to the gradient regularizer and the label predictor, and θ_d are the weights of the gradient regularizer itself.

8.3.1 DANN optimization functions

DANNs are optimized by adjusting loss to support the objectives. As seen in the equations below, a DANN's overall loss is the combined loss from label prediction and the negative weighted loss from the gradient classifier:

$$E(\theta_f, \theta_y, \theta_d)) = \frac{1}{n} \sum_{i=1}^{n} \mathscr{L}_d^i(\theta_f, \theta_d) - \lambda \left(\frac{1}{n} \sum_{i=1}^{n} \mathscr{L}_d^i(\theta_f, \theta_d) + \frac{1}{n} \sum_{i=n+1}^{N} \mathscr{L}_d^i(\theta_f, \theta_d) \right) \tag{8.2}$$

where θ_y is the set of weights for the label predictor. Gradient descent can progress normally with the following updates

$$\theta_f - \theta_f - \mu \frac{\partial \mathscr{L}_y^i}{\partial \theta_f^i} - \mu\lambda \frac{\partial \mathscr{L}_d^i}{\partial \theta_f^i}$$

$$\theta_y \to \theta_y - \mu \frac{\partial \mathscr{L}_y^i}{\partial \theta_y^i}$$

$$\theta_d \to \theta_d - \mu \frac{\partial \mathscr{L}_d^i}{\partial \theta_d^i}$$

where the learning rate is denoted as μ and the GRL regularization weight is denoted as λ:

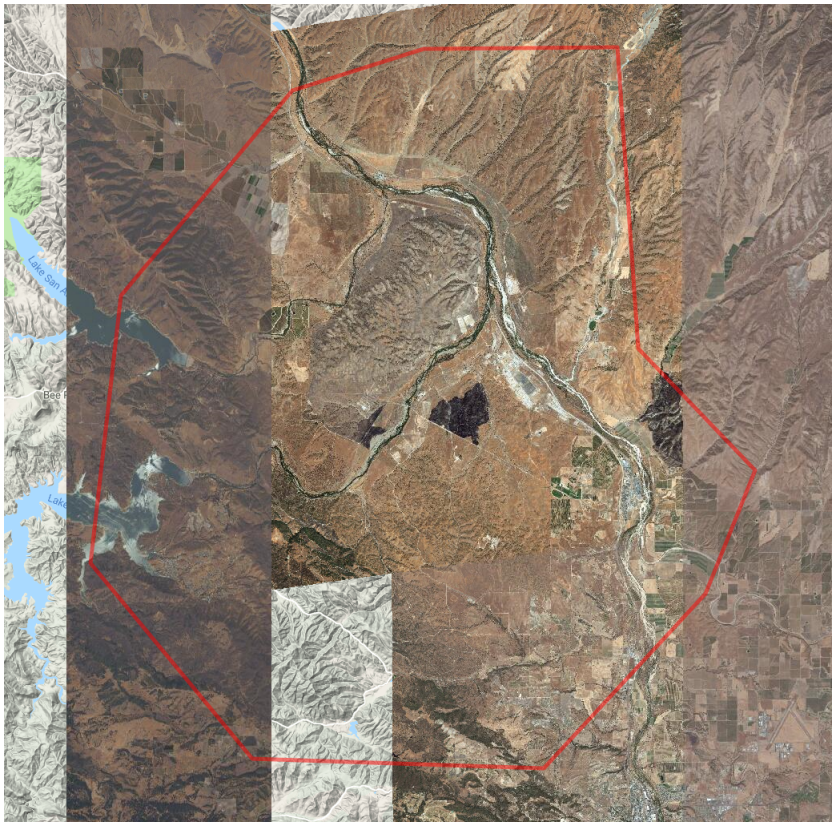

Figure 8.3: An overview of the Camp Roberts area from which satellite data was collected. Several distinct satellite collection passes that compose the image mosaic are visible. Clips for the source dataset were pulled from the red-bounded region.

8.4 Experimental Design

The DANN datasets were collected from Camp Roberts, CA where the NPS frequently conducts UAV flights. Covering an area of approximately 43,000 acres, it is one of the largest Army training installations in the US. It also covers a large variety of ecological zones and vegetation as seen in satellite imagery, in person surveys, and Berkeley's Wieslander Vegetation Type Mapping database [30], [31], [32]. Specifically, this area is a mix of grass range land, scrub, and trees. In addition to normal vegetation, large tracts of the Camp Roberts training area appears burnt, changing the appearance of normally golden brown grassland to a mix black, grey, and white ash (likely due to controlled burns as part of a responsible range management program). See a satellite overview of the region in figure 8.3.

A number of road types were visible in the satellite and UAV video. Paved roads measure approximately 3-12 meters wide with a light to dark grey appearance. They typically have clear boundaries with the road shoulder and contrast well with the surrounding terrain. Dirt and gravel roads are common throughout both datasets. Dirt roads are generally 2-3 meters wide and light brown to bright grey in appearance. Their shoulders are less well-defined, but

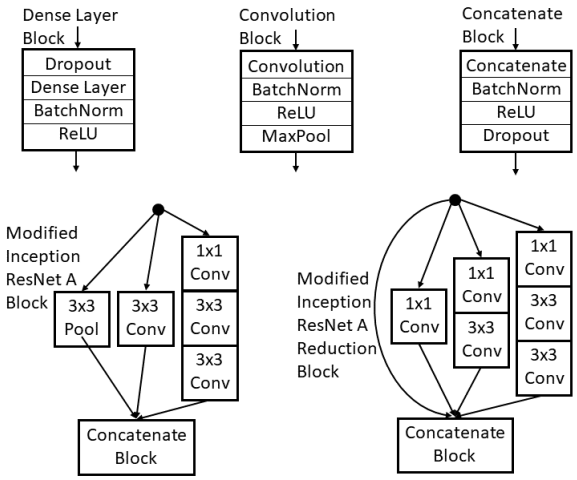

Figure 8.4:

still contrast well with their surroundings. "jeep trails" are present as well, appearing as two discernible, parallel tracks through open areas. These trails are generally 2-3 meters wide are generally lighter than the surrounding terrain, and often meander or loop back on themselves. “game trails” are also visible. These are narrow single track paths through open/grassy areas.

While paved and unpaved roads are easily discernible, even through more dense forested/treed areas, jeep and game trails can require a substantial amount of analyst interpretation to identify and track. This ambiguity in track and class can lead to error and realistically should exist as their own class in the generated datasets (i.e., Road/Trail/NotRoad vs Road/NotRoad).

8.4.1 Experimentation

The goal of the experiment was to quantify the benefits of DANNs, relative to NNs that are not domain adapted, on the specific task of predicting the class of items (e.g. roads) in a target dataset when no labeled target data is available for training. Lower and upper performance is bounded. At its lowest, the classifier will do no worse than a NN trained on satellite data, classifying ScanEagle data (no DA). At its best, our classification task will perform no better than a NN trained on ScanEagle data, classifying ScanEagle data.

8.4.2 Data

There were three primary sources of data for this research: satellite imagery, road vectors, and UAS video. Raw satellite imagery and road vectors will be used to generate the labeled source domain dataset for identifying the lower bounds of DANN performance and later training the DANN. UAS video will be used to generate the unlabeled target domain dataset (labeled for validating performance). This dataset will be used to identify the upper bounds of DANN performance and later for training the DANN.

Satellite imagery
All satellite data was acquired through a company called DigitalGlobe. DigitalGlobe is a company that owns and operates a constellation of five high-resolution, panchromatic to super-spectral satellites. They provide satellite imagery to a variety of customers including the United States Government (USG), telecommunications, automotive, and members of the global defense and intelligence community [33], [34]. Satellite data for this research was downloaded from DigitalGlobe's EnhancedView Web Hosting Service (EV-WHS) through a DoD subscription. EV-WHS is a web-based application that provides the ability to perform basic data analysis, viewing, query, and construction of satellite datasets [35].

The imagery came from the DigitalGlobe's WorldView-2 (WV02) Satellite. WV02 has been in orbit since October 2009 and is currently one of five satellites used by DigitalGlobe. It is able to collect multispectral imagery across 8 spectral bands (400-450 nm, 450-510 nm, 510-580 nm, 585-625 nm, 630-690nm, 705-745 nm, 770-895 nm, and 860-1040 nm) as well as panchromatic imagery (450-800 nm). Resolution for multispectral imagery from WV02 is as good as 1.85 meter Ground Sample Distance (GSD). Panchromatic imagery from WV02 is available at a much higher resolution: 0.46 meter GSD at nadir. Resolutions are slightly lower off-nadir.

The color satellite imagery for Camp Roberts was constructed as an imagery mosaic of data from five different WV02 collection passes. Cloud cover on each date of collection was 4% or less and an average of 16 degrees off nadir. All imagery has a GSD of 0.5 meters or less. Imagery used only contains three channels: red, green, and blue. Road data was manually generated using ArcGIS Pro and stored as linear vectors in an ArcGIS shapefile.

8.4.3 Annotated road data
Initially, road vector data was downloaded for use from National Geospatial-Intelligence Agency (NGA). Road vector data from NGA was originally intended as a source of labels for the satellite imagery downloaded from EV-WHS but was discarded due to issues with the data being out of date and trouble matching map projections between the road vector data and the satellite data. The road vector data for Camp Roberts was created manually using ArcGIS Pro.

8.4.4 UAV video footage
UAS video footage was acquired from a ScanEagle flown by NPS CAVR over Camp Roberts. It was controlled manually by a drone operator from the UAS GCS. The ScanEagle did have a secondary controller on-board that ran the open-source ROS. ROS managed the collection and transmission of data. Data from the collection flight was stored in a ROS bag file and video was stored in a packet capture (PCAP) file prior to being extracted for use.

The ScanEagle UAS is a product of Insitu. The aircraft is a small, modular UAV with and a rear-mounted pusher propeller. Depending on its specific configuration, the ScanEagle is about 1.5 meters long with a 3 meter wing span. It can carry a variety of payloads with a max takeoff

weight of 18 kilograms (12 kilogram empty weight). The ScanEagle system includes a GCS, launch system, and retrieval system. It can be configured for both sea or land operations [33].

The ScanEagle standard camera payload (Sony EX780) provides color video with a 640x480 resolution at 30 frames per second and a 1.8 to 45 degree FOV. This camera is mounted to an inertially stabilized turret in the aircraft nose that can point to the sides, down, front, and back. Video is transmitted to the ground receiver through an analog L-band link.

UAS video footage was collected from an NPS CAVR ScanEagle on 14 September 2018. This collection flight generated just short of an hours worth of 640x480 pixel footage at 30 frames per second. During video collection, the ScanEagle flew in rough, parallel tracks across Camp Roberts at 3,500 and 5,000 feet MSL (approximately 2,500 and 4,000 feet AGL). By comparing road width in pixels between satellite imagery and ScanEagle video, we can estimate the ScanEagle video to be at approximately 2 meter GSD as compared to the satellite imagery's 0.5 meter GSD.

Since video from the ScanEagle is transmitted in analog, quality varies throughout the flight depending on distance from the GCS, angle of bank, and signal interference. During turns, features often blur, but regain sharpness once the ScanEagle stabilizes. Throughout most of the video, the ScanEagle Video Sensor is pointed nadir to low-oblique with occasional transits to high-oblique during turns.

Sampling method
For training the CNNs and DANNs, the larger satellite mosaic and UAS video feed was broken into labeled clips with dimensions that were factors of the ScanEagle 640x480 video dimensions. Two satellite clip sets were made with clip size 120x120 and 160x160. One UAS video clip set was manually created with clip size 40x40. All four datasets were broken into training, validation, and test sets using a 50/25/25 percentage split. Clips were randomly selected, without replacement, from the original pool.

8.5 5-Phase process

As described previously, the research was divided into five phases. For defining domains and building datasets (Phase I) it was important to have a clear definition of road. In our case, there were 4 domain class definitions. They are shown in table 8.1. All of these classifications were considered a road. Another consideration included removing image variability by selecting source with nadir to low-oblique camera angles. Additionally, source imagery was collected during low cloud cover for image uniformity and collecting datasets at similar times of year so the overall environment and visibility were similar.

The ScanEagle video was extracted from saved PCAP files using Wireshark. The Python ImageIO package and OpenCV was used to review and save individual frames from this video. Frames are saved only if they displayed a unique scene, are nadir to low-oblique, are free of

Label	Description	Width (m)
Paved Road	Light to dark grey appearance. Often has visible, painted center-lines and shoulders, clear boundaries and good contrast with surrounding terrain	3-12
Unpaved Road	Light brown to bright grey in appearance. Shoulders are less well-defined, but still contrast with surroundings.	2-3
Jeep Trail	Appear as 2 discernible, parallel tracks. These trails are generally lighter than the surrounding terrain, but in certain areas blend well so that they are identifiable only by textural differences.	2-3
Game/Foot Trail	These are narrow single track paths that are only identifiable where they contrast with the surrounding terrain.	0.5-1

Table 8.1: Definitions of road and trail classes. All of these classes are considered "road" class for training. Everything else is classified as "not road".

major video transmission errors, and relatively blur free. These frames are then broken into overlapping clips and labeled clip-by-clip as "road" or "not road."

Both datasets were then sampled randomly, without replacement, to be included in a training, validation, or test set in a 50/25/25 split. Clips were then saved in a file structure required for TensorFlow's ImageDataGenerator class flow_from_directory method. Test sets remained reserved for evaluating model performance after all hyperparameter tuning had been completed.

For the building and training the source-domain-only CNNs (Phase II), Three, non-DANN CNN architectures were built, trained and tuned on the satellite data. They are described in table 8.2. Training was completed using the data and tuned based on the resultant model's evaluation against the validation set.

Phase IV evaluated the CNNs on the source and target data and the empirical bounds for DANN performance were determined. The target domain (aerial) test dataset was evaluated on non-DANN source-only trained CNNs and a target-only trained CNNs. The results of these tests will provide the lower and upper bounds of performance for the DANNs to be trained in Phase V. In this phase we continued to evaluate performance differences between the 120x120, 160x160 and 40x40 clip sizes. The target input size for each model was 80x80.

Phase V determined the actual performance of the DANNs. The two best CNNs from Phase IV were modified and included a domain classifier with its GRL. These DANNs were trained using the source-domain training data with labels to train the classifier. Both the source-domain training data and the target-domain training data were used to train the domain-classifier.

Model	Description
Simple 6-layer CNN	Three convolutional blocks followed by two dense layer blocks.
Custom, abbreviated Inception/ResNetV2 CNN	High modified model based on Inception/ResNetV2. Composed of an initial stem designed for capturing base features with complexity reduction. Followed by one modified Inception/ResNet Block-A and Reduction Block-A. Terminates with more complexity reduction and our standard label classifier.
CNN with frozen Inception/ResNetV2 base	Contains and Inception/ResNetV2 feature extractor with frozen weights followed by two dense layer blocks.

Table 8.2: Definitions of road and trail classes. All of these classes are considered "road" class for training. Everything else is classified as "not road".

For evaluating performance of the DANNs during training, and for hyper-parameter tuning, validation sets were used. The label classifier was evaluated against the target-domain validation set and labels. The domain classifier was evaluated against both the source- and target-domain data and generated labels ("[1,0]" for source "[0,1]" for target). As in previous phases, the various combinations of generated clip sizes was used for training to determine optimal size and mixing appropriate for overcoming the resolution disparity between domains.

The effectiveness of DANNs was measured by the distance closed between our empirical lower and upper bounds towards the upper bound. If this improvement is marginal regardless of architecture, then we can know that DANNs may not be the best form of DA for adapting satellite trained CNNs to unlabeled UAS data.

8.6 Results

Tables 8.3 and 8.4 show the distribution of satellite imagery and UAV video datasets. The creation of the road vector data through ArcGIS and manually extracting, creating and labeling the 40x40 clips from video was laborious.

Dimensions	Stride	Road/NoRoad	Train	Validate	Test	Processing Time
160x160	80	37.7/62.4	168576	84288	84289	96h 15m
120x120	60	30.5/69.5	299419	149710	147910	213h 07m

Table 8.3: Distribution of generated satellite datasets.

In Phase II the 3 CNN model architectures were trained and tuned using the training and validation source-domain only datasets. Shown in table 8.5, after adjustments to structure and hyperparameters, all three models performed well on the 120x120 and 160x160 datasets.

Dimensions	Stride	Road/NoRoad	Train	Validate	Test	Processing Time
40x40	20	33.0/67.0	60720	30361	30361	9h 09m

Table 8.4: Distribution of generated UAS video datasets.

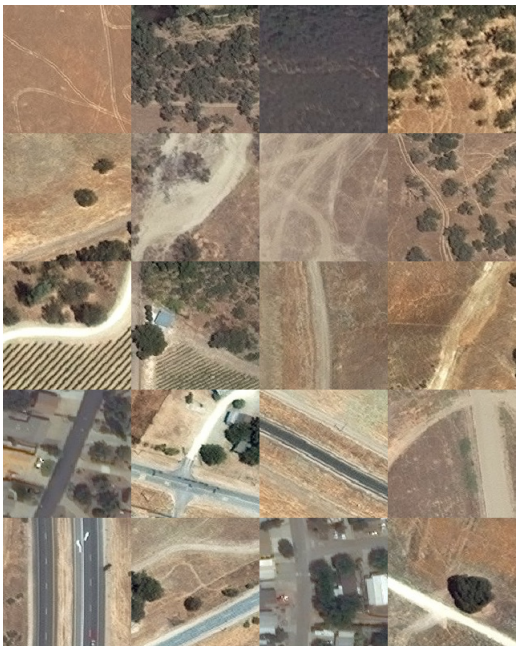

Figure 8.5: This a sampling of the “road" class from the 160x160 satellite domain dataset. It shows how broad the domain definition for “road" is and how challenging it could be with some of these images for the classifier.

Architecture	Dataset	Train	Validate	Time	Stop Epoch
Simple 5-layer					
	120x120	92.47%	91.14%	116h 38m	600
	160x160	89.37%	91.06%	29h 06m	600
Custom Inc/ResV2-like					
	120x120	82.78%	84.54%	62h 24m	457
	160x160	83.52%	81.64%	59h 05m	600
Frozen Inc/ResV2 Base					
	160x160	99.88%	90.00%	291h 08m	600

Table 8.5: Maximum accuracy of the three evaluated CNN architectures during training and validation and the total number of epochs after early stopping or after the maximum set 600 epochs was reached.

The Adam optimizer provided the best results in all cases. Batch sizes were set to 64 with a learning rate of $1x10E-6$ and momentum set to 0.9. Dropout was set to 0.5 and L2 regularization was introduced and set to 0.2. Early stopping was used to prevent unnecessary training

time in most cases, though set at 50 due to a mid-training plateau for the custom CNN architecture. Training was extended only when it was clear that training stopped too early: i.e., a clear upward, but oscillating, trend in performance. An absolute maximum of 600 epochs was used.

Phase III was essentially a repeat of Phase II, but using the target-domain training and validation datasets instead. Hyperparameters from Phase II were used without modification. The highest accuracy achieved for each model/dataset is presented in table 8.6.

Architecture	Dataset	Train	Validate	Time	Stop Epoch
Simple 5-layer					
	40x40	91.23%	87.50%	10h 05m	600
Custom Inc/ResV2-like					
	40x40	83.16%	83.98%	7h 13m	600
Frozen Inc/ResV2 Base					
	40x40	99.66%	86.50%	105h 56m	600

Table 8.6: Target domain only CNN accuracy. Accuracy of the three evaluated CNN architectures during training and validation.

Phase IV established the lower and upper bounds for the performance of Phase V DANNs. All trained models were re-evaluated against both source and target domain test datasets. Results are presented in tables 8.7, 8.8. The performance of models trained on the 40x40 UAS video data is included in those tables.

Architecture	Trained on	Tested on	Accuracy	Evaluation Time
Simple 5-layer				
	120x120	120x120	91.08%	1m 56s
	120x120	40x40	50.66%	0m 08s
	40x40	40x40	87.40%	0m 17s
Custom Inc/ResV2-like				
	120x120	120x120	84.55%	5m 43s
	120x120	40x40	63.26%	1m 02s
	40x40	40x40	87.40%	0h 17m
Frozen Inc/ResV2 Base				
	120x120	120x120	84.55%	5m 43s
	120x120	40x40	63.26%	1m 02s
	40x40	40x40	86.35%	1m 38s

Table 8.7: Evaluation on 120x120 trained CNN. Accuracy of the highest performing models evaluated on the test sets.Source on source accuracy include for reference only.

For Phase V, The best architectures, training time considered, were the simple CNN and custom CNN architectures. Due to this, the DANN versions of these were used for the satellite/UAS

Architecture	Trained on	Tested on	Accuracy	Evaluation Time
Simple 5-layer				
	160x160	160x160	89.47%	1m 26s
	160x160	40x40	53.71%	0m 22s
	40x40	40x40	87.40%	0m 17s
Custom Inc/ResV2-like				
	160x160	160x160	83.53%	3m 25s
	160x160	40x40	54.95%	1m 15s
	40x40	40x40	83.99%	1h 02m
Frozen Inc/ResV2 Base				
	160x160	160x160	90.04%	9m 19s
	160x160	40x40	67.81%	1m 39s
	40x40	40x40	86.35%	1m 38s

Table 8.8: Evaluation on 160x160 trained CNN. Accuracy of the highest performing models evaluated on the test sets.Source on source accuracy include for reference only.

DA task. The transfer learning models proved so complex that they took more time than administratively available for this research. Results are shown in tables 8.9 and 8.10.

Architecture	Trained on	Tested on	Accuracy	Evaluation Time
Simple 5-layer				
	120x120	40x40	50.66%	0m 8s
	120x120/40x40	120x120	84.27%	2m 41s
	120x120/40x40	40x40	87.40%	0m 17s
	40x40	40x40	87.40%	0m 17s
Custom Inc/ResV2-like				
	120x120	40x40	63.26%	1m 2s
	120x120/40x40	120x120	81.32%	6m 52s
	120x120/40x40	40x40	69.40%	1m 15s
	40x40	40x40	83.99%	1m 2s

Table 8.9: Evaluation on 120x120/40x40 trained DANNs on the 40x40 dataset. Accuracy of the highest performing architectures evaluated on the test sets.

In summary, DANNs offer a framework for understanding when a CNN trained in one environment can be used in another. This is important for the generalization of CNNs. In this chapter, the two domains were imagery from remote sensing satellites and video from a UAS. This is a potentially useful application since there is an abundance of satellite imagery and frequently a paucity of UAS imagery in remote areas of operation.

We showed that DANNs do provide an increase in cross-domain labelling accuracy in a remote

Architecture	Trained on	Tested on	Accuracy	Evaluation Time
Simple 5-layer				
	160x160	160x160	89.47%	1m 26s
	160x160	40x40	53.71%	0m 22s
	40x40	40x40	87.40%	0m 17s
Custom Inc/ResV2-like				
	160x160	160x160	83.53%	3m 25s
	160x160	40x40	54.95%	1m 15s
	40x40	40x40	83.99%	1h 02m
Frozen Inc/ResV2 Base				
	160x160	160x160	90.04%	9m 19s
	160x160	40x40	67.81%	1m 39s
	40x40	40x40	86.35%	1m 38s

Table 8.10: Evaluation on 160x160 trained CNN. Accuracy of the highest performing models evaluated on the test sets.Source on source accuracy include for reference only.

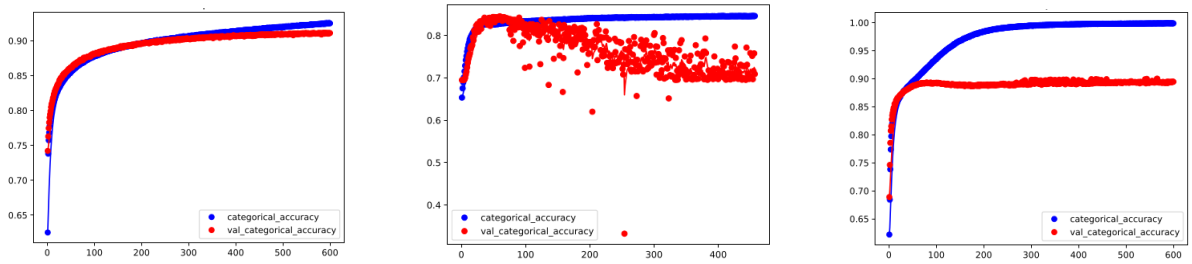

Figure 8.6: CNN training and validation plots for the simple, custom and transfer learning architectures on the 120x120 satellite data.

sensing setting when target-domain labels are “unavailable". Increases in performance, however, were modest at best; even when using architectures that capture more of the source and target domain complexity. We believe this modest level of improvement had more to do with curation of our datasets than the actual structure of the models used.

As a whole, we can say that a DANN modified CNN does help a CNN trained on satellite data to perform better on the cross-domain classification task on aerial video clips. Since our DANNs did not completely cross the gap from lower to upper empirical performance bounds another form of DA is worth exploring as are other architectures and data pre-processing techniques. The performance we experienced is likely due to a number of limitations that were identified during experimentation that would warrant further investigation.

An important observation was how well the DANNs performance adhered to theory. All trained DANNs performed within the lower and upper bounds identified in Phase IV of our work. These bounds are useful given the ambiguity to be expected in training a DANN without a labeled

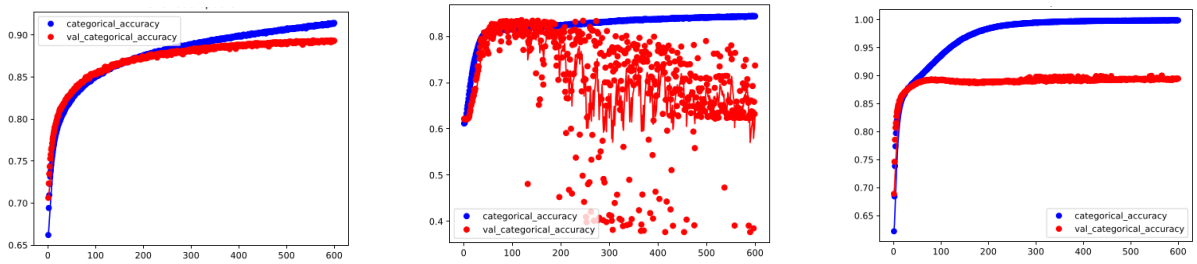

Figure 8.7: CNN training and validation plots for the simple, custom and transfer learning architectures on the 160x160 satellite data.

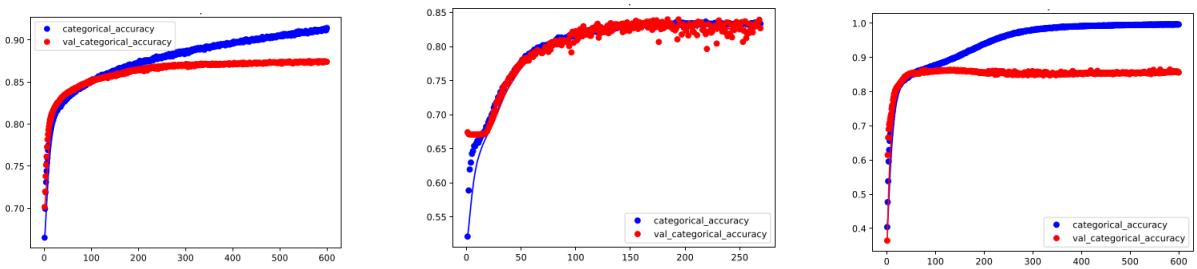

Figure 8.8: CNN training and validation plots for the simple, custom and transfer learning architectures on the 40x40 UAS video data.

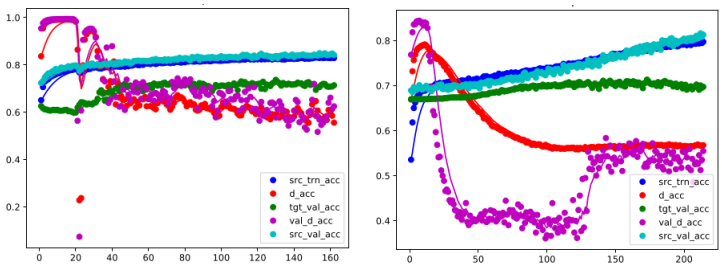

Figure 8.9: DANN plots for source training and validation accuracy, target validation accuracy, and domain training and validation accuracy for the “simple" architecture.

target dataset to validate performance against.These bounds are also useful while evaluating the type and form of data preparation or transformation that is best for training. In our work, we experienced the best DANN performance when using the 120x120 source satellite clips and 40x40 target UAS clips resized to 80x80 for input. This was expected as the 120x120 and 40x40 clips demonstrated the lowest domain divergence.

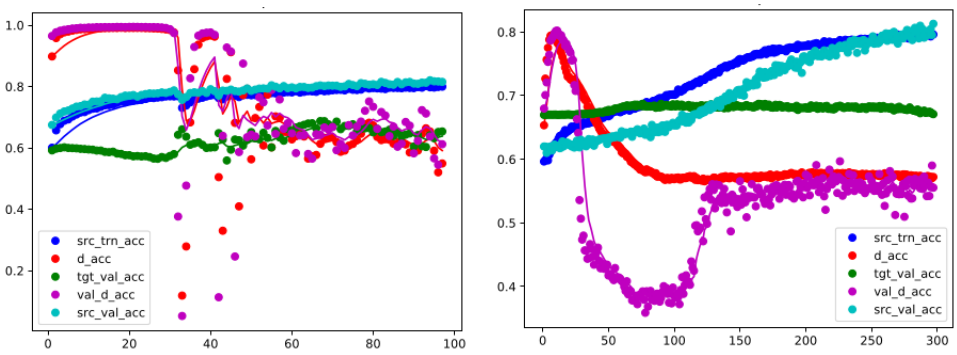

Figure 8.10: DANN plots for source training and validation accuracy, target validation accuracy, and domain training and validation accuracy for the custom architecture.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 9:
A Novel Approach for Training CNN for 3D Objects

This chapter is a summary of the thesis work of Andrew Watson [36].

Similar to the previous chapter, this work presents a methodology to train a CNN with dissimilar data. In this case, the goal is to train a CNN for detection and classification of 3D objects from lidar. The novel approach simultaneously collects camera and lidar data and uses the detection of the objects from the camera to label the clustered lidar 3D data as an object. Again, similar to the last approach, this work provides a methodology to avoid the laborious step of collecting and hand-labeling data but the main goal is to develop a CNN that uses real-time lidar data as input and outputs a label accurately classifying the 3D object.

What follows is a three-phase model for pointcloud classification. The model first collects synchronized 2D wide-angle camera images and 3D lidar pointclouds and depth clusters each lidar frame to spatially segment the scene. Next, it correlates each resultant pointcloud segment to a cropped 2D image and processes each 2D image crop through a CNN to assign a classification to the image crop as well is the segmented pointcloud. Another aspect of the approach is a method of scene discovery to boost pointcloud classification performance. It explores a method to scrape regional geo-tagged media for processing through an object-detection neural network. A data mapping of object-type spatial relations is developed for a specific environment and apply these relationships as weights in order to boost pointcloud classifier performance.

9.1 Model overview

The three-phase model is shown in figure 9.1. Phase one is the automated dataset creation. It begins with the collection of synchronized lidar and RGB wide-angle camera data and leverages a 2D image classifier, coupled with the RGB camera frames to create labels that are then used for the segmented pointclouds from the synchronized lidar data. Phase two is context discovery. It is a technique to improve classification by producing a database keyed to a physical environment and contains assessed likelihoods of a spatial object pairing for that environment. These pre-calculated likelihoods are applied as weights to the phase three classification. The phase two database is built by processing geo-tagged media from the designated operating environment.

Phase three is the 3D pointcloud classification. There were three approaches used to boost classifier performance: temporal weighting, absolute weighing and relational weighting. Temporal weighting leverages previous-frame segment classifications to adjust weights of current-frame segment classifications. Absolute weighting applies the absolute hit-counts from the context discovery database as weights to the pointcloud classifier outputs. Relational weighting does intra-frame analysis and applies a weight to prospective classifications in a given scene, based on the prevalence of object relationships in the context discovery database.

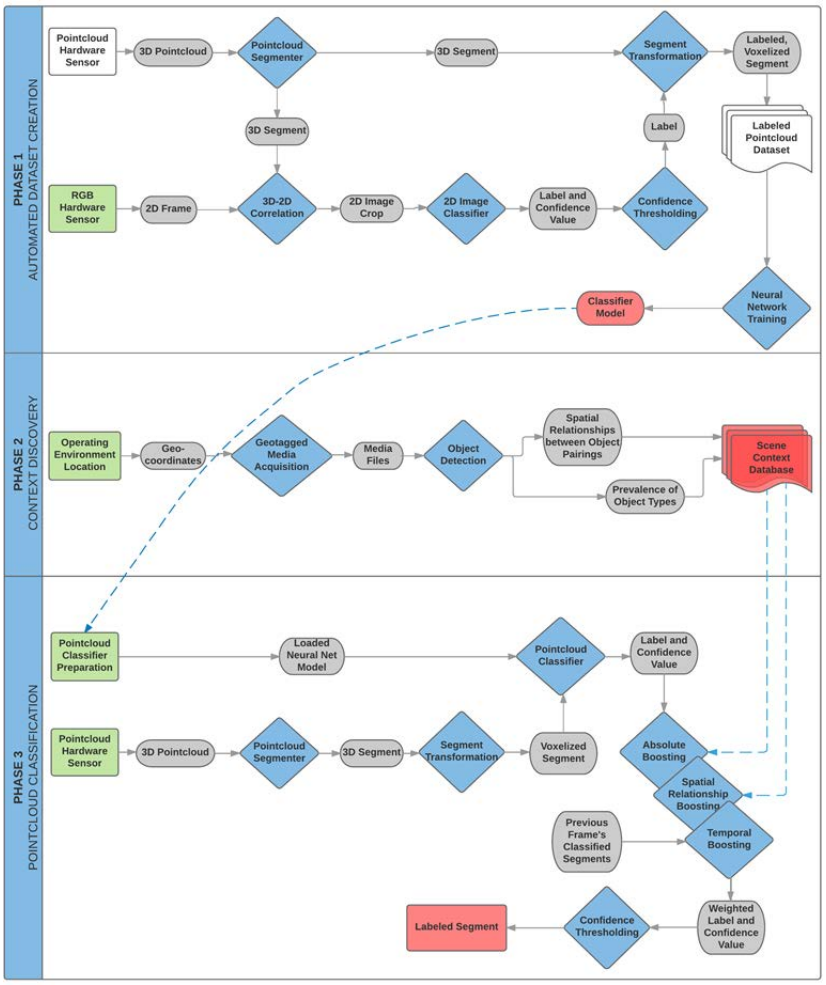

Figure 9.1: Flowchart of the Three-Phase Point Cloud Classification Model. Phase one is the automated dataset creation. Phase two is context discovery. Phase three is pointcloud classification.

9.2 Sensor platform configuration

A tandem lidar and multi-camera module was designed for the geo-synchronized collection of 3D pointcloud and 2D imagery data. Figure 9.2 shows the CAD designed 3D-printed housing for the mobile connection mount that was used to hold the Velodyne HDL-32e scanning laser (lidar), three Logitech c920 HD cameras, Velodyne lidar interface box, 12V battery and 200W inverter. This was mounted on the top of a pickup truck and driven around the objective area. Figure 9.3 shows the complete setup, with three cameras oriented at 90-degree offsets on the horizontal plane and the Velodyne lidar in the center.

Data collection was conducted in and around the NPS campus in Monterey, California and on SCI, California. All data was collected with the tandem lidar and camera collection rig. The data was collected while driving at an estimated maximum speed of 30 mph.

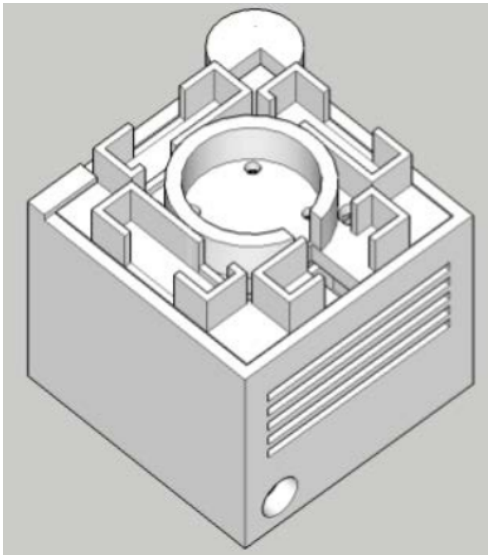

Figure 9.2: 3D printed housing and mount for the lidar and cameras. Inside the mount are a 200W inverter, 12V battery and Velodyne lidar interface box.

Figure 9.3: The sensor mount with the Veloryne lidar and four USB Logitech cameras.

9.2.1 Lidar hardware

All pointcloud data was collected from a Velodyne HDL-32e lidar, a sensor capable of producing 700,000 3D points per second [37]. Data from the lidar was produced at a rate of 10Hz and was collected at a minimum range of 0.9 meters and maximum range of 130 meters.

9.2.2 Camera hardware

2D camera data was collected from three Logitech c920 HD cameras oriented 90 degrees apart. As the scanning lidar provides a 360-degree view of scene, the data processing pipeline could feasibly accommodate a full 360 degree 2D view as well. However, data processing issues limited the collection to only three simultaneous camera feeds. Our tandem collection setup further limits the horizontal and vertical field of view (FOV) available to the pipeline. The c920's horizontal FOV is approximately 70.42 degrees [50] leaving a nearly 20-degree gap between adjacent c920 cameras that is not collected. Figure 9.4 depicts the effective horizontal FOV of our collection setup.

The c920's vertical FOV is approximately 43.30 degrees and the LIDAR's vertical FOV is 40 degrees, but the LIDAR's vertical FOV is angled downward and ranges from +10 to -30 degrees. The overlap between these two sensors provides an effective vertical FOV of approximately 32.32 degrees (see figure 9.5). Our pipeline addresses the inconsistent FOV capabilities of the two sensors by ignoring segments containing points that map to pixel locations beyond the camera's horizontal or vertical FOV. This approach has the effect of missing opportunities to

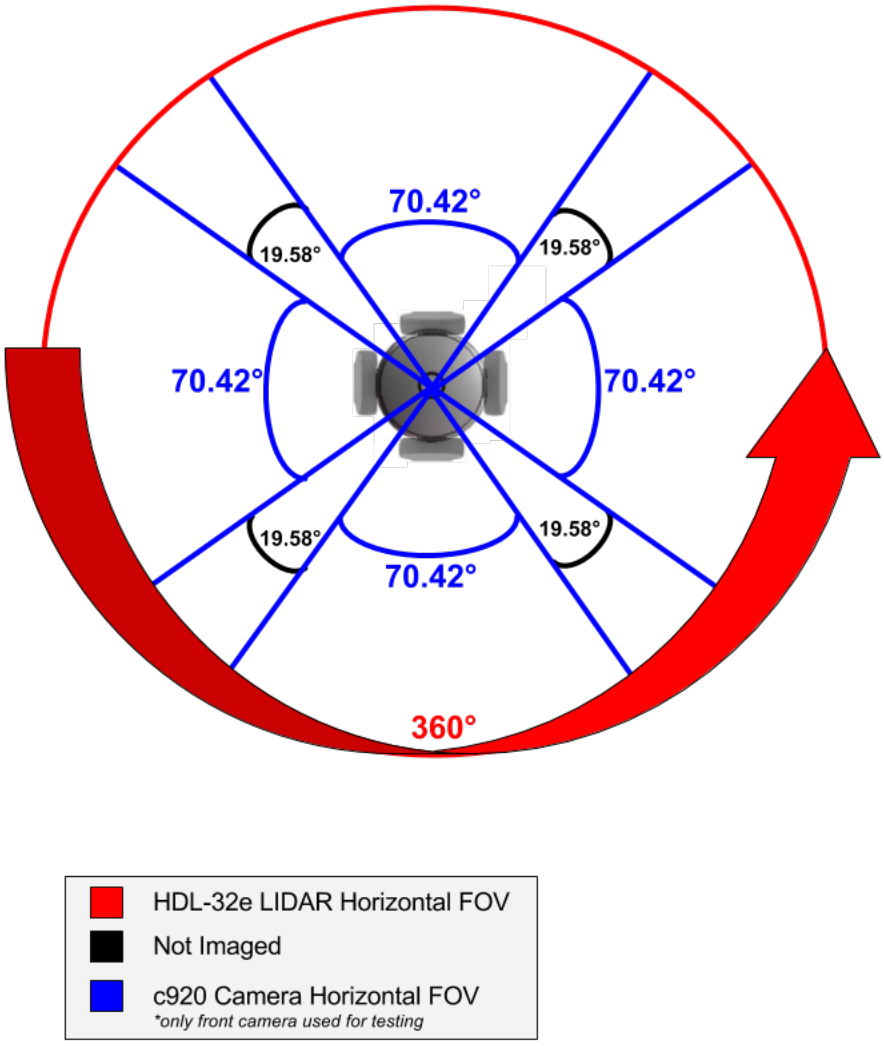

Figure 9.4: Horizontal Field of View (FOV) of the camera and LIDAR sensing platform. Only the the front camera was used for testing.

process segments as they transition out of the camera's view.

Figure 9.6 shows an example of the raw data collected. The pipeline loaded a manufacturer-provided calibration file for the LIDAR and did not correct for lens distortion on the camera system.

9.2.3 Software

ROS provided the software infrastructure for processing and storing our LIDAR and camera data in a synchronized manner. ROS is a publish-subscribe message passing architecture designed for robotics and features a multitude of integrations for computer vision and pointcloud processing [38]. In addition to the core ROS infrastructure provided in all ROS applications,

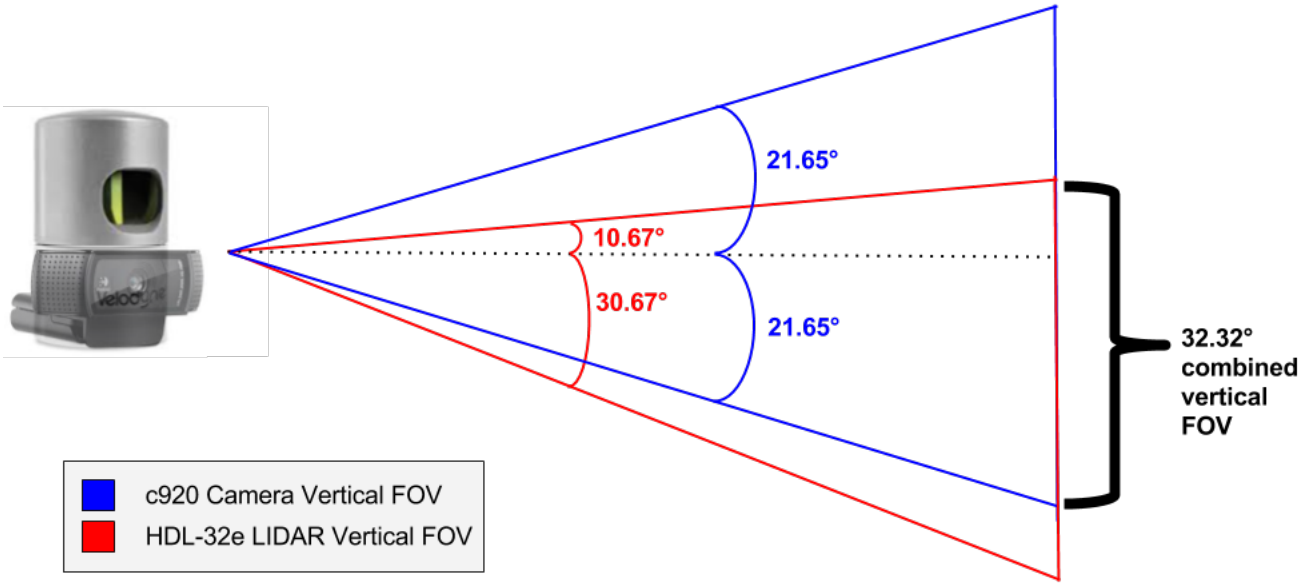

Figure 9.5: Combined 32.32 degree Vertical FOV of the camera and LIDAR systems.

two additional ROS nodes were employed - usb_cam and velodyne_pointcloud. ROS nodes are parallel processes that communicate with a central ROS master node and typically pass messages to each other. Three instances of the usb_cam node were run during data collection to process three simultaneous streams of camera frames, with each video stream encoded with MJPEG compression at a resolution of 1920 x 1080 at 30Hz.

The Velodyne pointcloud ROS node ingested raw UDP data packets from the LIDAR and published the data as ROS PointCloud2 messages. Raw packets were also preserved in the data collection. The archiving for the synchronized LIDAR and camera data was done via ROS bag files. Bag files store specific ROS topics to disk as time series data and, in the case of the topics published for the LIDAR and three camera streams, the bag file archives grow at a rate of 2 gigabytes per minute. All bag file datasets have been archived and are available upon request. Figure 9.7 visually depicts the process.

9.3 Phase one: Automated dataset creation and training

Figure 9.8, pictorially shows the phase one data collection process for collecting synchronized LIDAR and camera datasets. It consists of 5 steps: pointcloud segmentation, 3D-2D correlation, 2D classification, confidence-level filtering and segment transformation. The output of phase one produces labeled, high confidence pointcloud segments without human intervention during the annotation process.

Figure 9.6: Example of the combined data capture with the 3 images from the cameras and the corresponding LIDAR pointcloud scene.

9.3.1 Pointcloud segmentation

Pointcloud segmentation divides the pointcloud scene into individual cluster of points with each cluster or segment representing a distinct object in the scene. Two segmentation approaches were evaluated: Difference of Normals (DON) and Depth Clustering segmentation. DON segmentation combines surface normal calculation with Euclidean clustering. It is computationally intensive. Depth clustering creates 2D range images from LIDAR-derived depth values and has shown to produce similar quality results while boasting a 100x speedup over Euclidean clustering methods [39].

9.3.2 3D-2D correlation

This process finds the 2D corresponding image segment that is associated with the pointcloud segment. It is done by calculating a pixel-per-degree ratio from the horizontal and vertical field-of-view angular ranges for the RGB sensor, along with the pixel dimensions of the RGB frames. Next, each 3D coordinate in the pointcloud segment is correlated to a pixel location on the RGB frame by 1) selecting the 3D coordinate on the LIDAR pointcloud, 2) calculating the 3D angular offsets and 3) calculating the 2D pixel location on the frame. This is done for each 3D point in the pointcloud segment and results in a 2D image crop of the same physical object as pointcloud segment from the LIDAR.

9.3.3 2D classification

The next step is to determine an English language label for the pointcloud segment. To accomplish this, the 2D image crop is provided as input into a neural network image classifier. The NN classifier outputs a list of possible classifications (e.g. stop sign) and then puts a confidence

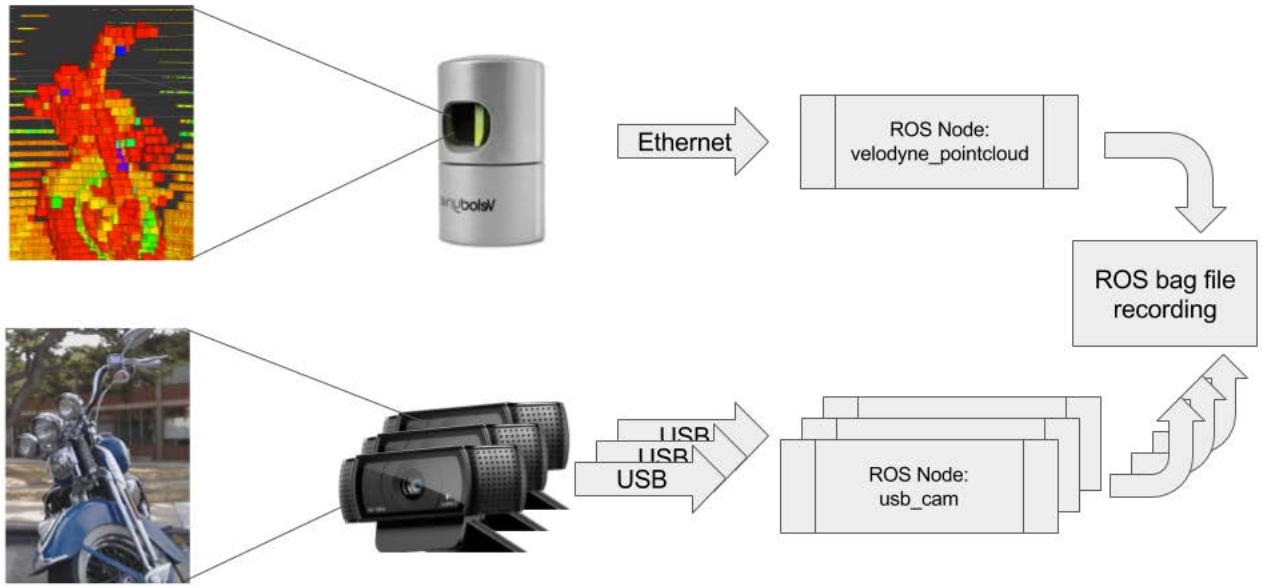

Figure 9.7: The LIDAR and camera hardware to software flowchart.

value associated with each classification. The classifications are ranked using the confidence levels where the highest ranking, known as the Top-1 result is selected.

9.3.4 Confidence-Level thresholding

Without any additional filtering the Top-1 results can still represent a low-confidence identification. Confidence-level thresholding discards low-confidence Top-1 labels by filtering based on a minimum, user-defined threshold. The classified image is now assigned to the 3D pointcloud segment. These labeled segments can now be used in training neural networks for classifying 3D objects but additional transformations are necessary for a representation appropriate for input into a neural network.

9.3.5 Segment transformation

Pointcloud segment transformation is necessary because NN models require uniformly sized training set records. For example, images for 2D classifier might be resized to a fixed pixel height and width. Pointclouds require the same fixed-sized inputs but this can be challenging representing the same object at different distances. The technique that is used is to employ volumetric pixels (voxels) to transform the raw pointcloud data in a labeled segment prior to training.

Voxel grids, coupled with 3D scaling operations allow pointcloud segments representing various physical objects to be transformed into fixed-sized NN training inputs (e.g. 20 x 20 x20 voxel

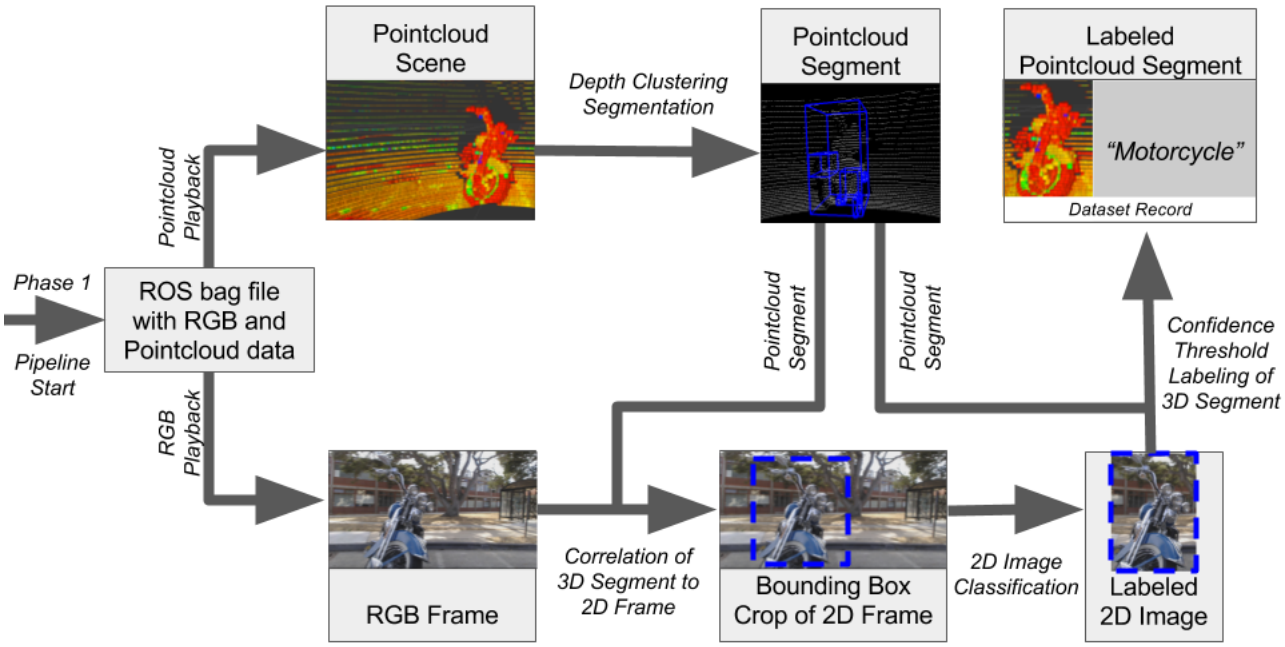

Figure 9.8: The phase one processes: Pointcloud segmentation, 3D-2D Correlation, 2D classification and confidence-level thresholding.

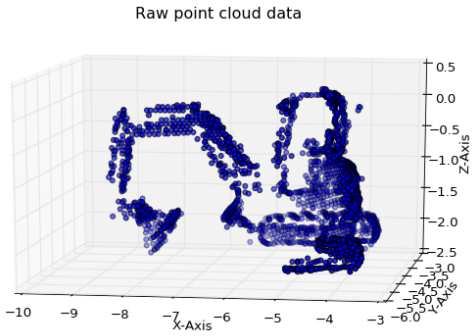

Figure 9.9: Example of a Raw LIDAR pointcloud of an excavator.

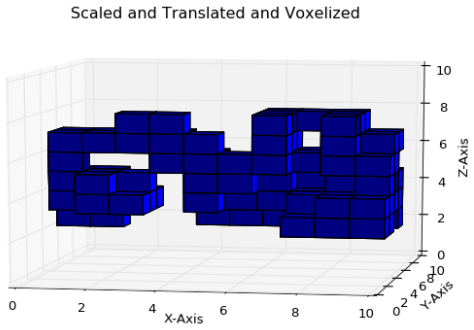

Figure 9.10: Transformed excavator to a scaled, translated and voxelized representation.

grid). Figures 9.9 and 9.10 show the transformation process of converting a raw pointcloud segment to a voxelized representation of uniform volume [40].

9.3.6 Neural network training

Using industry best practice standards for NN training. 80% of the dataset is used as a training set and the remaining 20% is used as a test set to validate the model for accuracy and overfitting.

9.4 Phase two: Context discovery

Scene recognition can improve the performance of pointcloud classifiers by taking into account the environmental context to weight classification results. A database containing the likelihood of an object appearing in an environment as well as each object's likelihood of being located with another object can be used to boost the confidence value of classifications.

To avoid the same resource-intensive hand labeling datasets as was done in phase one, this phase automates the context discovery process by performing object detection on internet geo-tagged media and populating a database containing object-type relationships for a given location. By pre-calculating geographically-localized scene context relationships to object classification, it can improve performance of the pointcloud classifiers.

To accomplish this, phase two has three components: Acquisition of geo-tagged media, object detection and populating the scene context database. This first component requires foreknowledge of the intended operating environment. By searching the internet based on the position (e.g. latitude and longitude) as well as keyword searches, APIs exist to conduct metadata-based media searching for extracting targeted geo-tagged imagery and videos. These sources are then mined for relationship information using object detection tools.

Processing each geo-tagged frame through the object detector identifies objects that appear together in a frame. These indicate a spatial relationship between those object classes. The final step is populating a relational database. The probability of an identified relation between pairings of geo-tagged media is simply determined by the number of hits received by the object type. This is stored using a $Nx(N+1)$ relational database containing a table and column for each object type and the count for each object class.

9.5 Phase three: Real-time pointcloud classification

The culmination of the process is a real-time pointcloud classifier. Once trained, there is no requirement for the camera. This also means that the classifier (based on the lidar sensor) can work either in daytime or nighttime, which is a major advantage for military operations. Phase three components include classifier preparation, pointcloud collection, pointcloud segmentation, data transformation, classification by NN, boosting and confidence thresholding.

Classifier preparation loads the now-trained, phase one NN model and the relational database into the machine learning toolset's classifier. Pointcloud collection mimics the phase one process but without the camera 2D data collection. As pointcloud data is collected by the lidar, it is segmented and voxelized to prepare it for NN classification.

Pointcloud segmentation is similar to the process described in phase one but now has a real-time requirement. This limits the options for pointcloud segmentation to those capable of achieving near real-time performance. In general this requirement has an impact of imposing a size restriction on the segmented objects that corresponds to the horizontal and vertical fields of view.

Data transformation operations are the same as phase one. It results in fixed-sized, voxelized, pointcloud segment representation. Each pointcloud segment is now processed by the NN pointcloud classifier model trained during phase one. NN classification are generally dramatically faster than training operations. Each transformed pointcloud segment is processed by the classifier and returns a ranked list of identifications with confidence values.

As discussed earlier, it is possible to overcome inaccuracies in pointcloud segment classification results through boosting. The previous discussion of boosting was limited to evaluation of each segment independently. We can also boost performance by considering temporal weighting between a sequence of pointcloud frames. Additionally, boosting performance is improved by leveraging the relational database created in phase two. The final step is a thresholding based on a confidence level.

9.6 Model conclusions

In summary, the three phase model provides a comprehensive, novel framework for pointcloud classification. The methodology starts with automating the creation of training data for a pointcloud classifier (phase one), maps out spatial relationships between object pairings in a designated operating environment (phase two) and integrates training data and scene context into the real-time pointcloud classifer (phase three).

9.7 Results

System testing was conducted in the Oak Hill neighborhood in Monterey, CA and San Clemente Island, CA. For security reasons only the Monterey results will be included. Phase one implementation performance was on two data sets shown in figure 9.11. The “Neighborhood 1" collection is from the path highlighted in yellow and the “Neighborhood 2" collection is annotated in red.

Accuracy for phase one is defined in three ways: First, for each of the labeled pointcloud segments outputted by the pipeline, the resultant label was manually compared to the original cropped image and assessed for accuracy in labeling. This was necessary to determine whether a classification error was caused by Inception-v3. Second, each pointcloud segment was compared to the corresponding 2D crop. This was necessary to detect whether there was a synchronization error between the lidar and camera frames or to determine whether a foreground object in the 2D image crop inappropriately produced a label for the 3D pointcloud segment in the background (occlusion error).

No attempt was made to evaluate the performance of the phase two context discovery. The implementation built the relational database through bulk video scraping from YouTube object detection through the YOLO9000 NN, and the creation of a customized schema to create a SQL scene context database. The database contained 9418 tables, one for each of the YOLO9000's classes. Figure 9.12 shows a visualization of a particular 90 x 90 subset of the database and

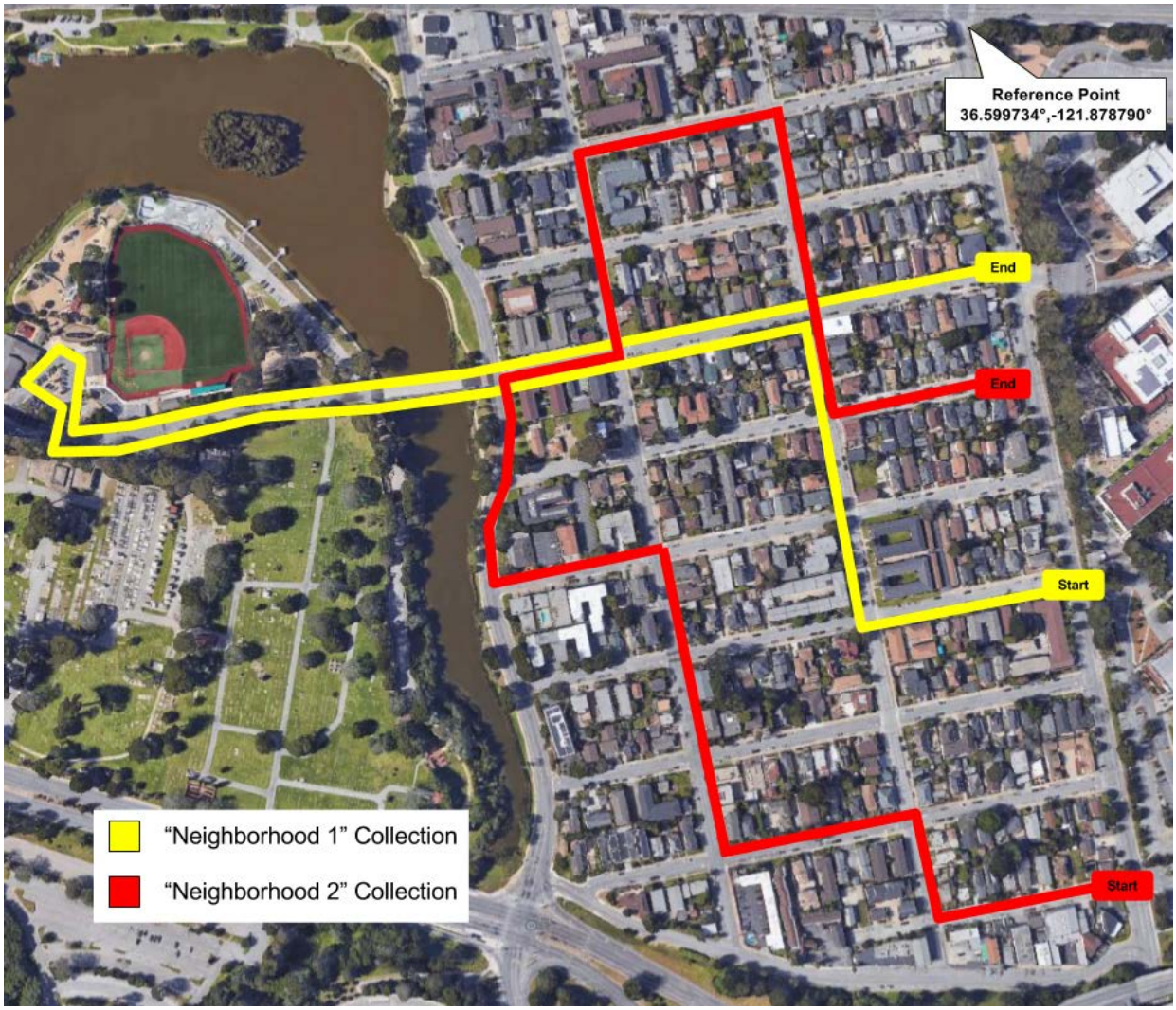

Figure 9.11: Map view of routes for Neighborhood 1 and Neighborhood 2 data collections.

shows each object's type absolute hit count as the dominant diagonal values, with spatial relationships between object types depicted as intersecting values of the graph's 3D grid.

Note that the relationships discovered as part of the phase two implementation required no human intervention. beyond the selection of a bounded physical environment.

9.7.1 Phase one performance

The Neighborhood 1 dataset was processed through the automated pipeline and created a total of 41 labeled pointcloud segments from a collection duration of 346 seconds. Based on the following criteria, we judged the pipeline to have produced 31 correctly labeled pointcloud segments and ten incorrectly labeled pointcloud segments, representing a 75.6% accuracy (Figure 9.13).

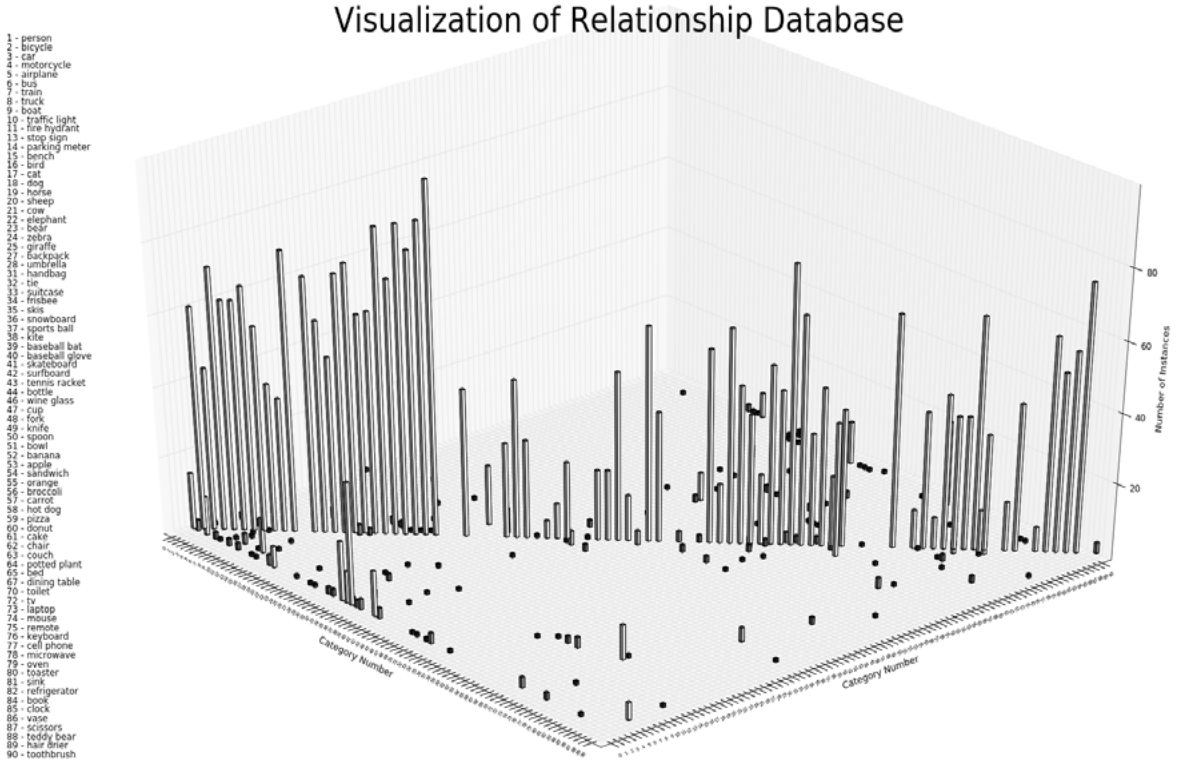

Figure 9.12: Visualization of "Scene Context" database containing relationship weights between objects in a specific physical environment.

The ten incorrectly labeled segments suffered from incorrect image classification, synchronization issues and occlusion errors. The two incorrect image classification results were not necessarily due to poor accuracy in Inception-v3's classification but, instead, were due to extremely small image crops being fed into Inception-v3. For context, our implementation neglected to do any filtering of small image crops. This can be addressed in future revisions.

The primary source of error was occlusion, with six of the 41 results exhibiting this trait. In all six cases, the cropped frame contained multiple objects and Inception-v3 correctly classified an unintended object, one which did not match the corresponding segmented pointcloud. As the data was collected in a suburban environment, it was common for trees or vehicles to enter the foreground of cropped images, raising the potential for Inception-v3 to classify that object instead. Interestingly, in some cases, the reverse occurred as well, with Inception-v3 correctly classifying a distant background object instead of the intended foreground object. To mitigate this behavior, a more exacting approach to creating image crops would be required to avoid including unintended background or foreground objects. Figure 9.14 shows an example of occlusions where the tree pointcloud segment is erroneously labeled as a car.

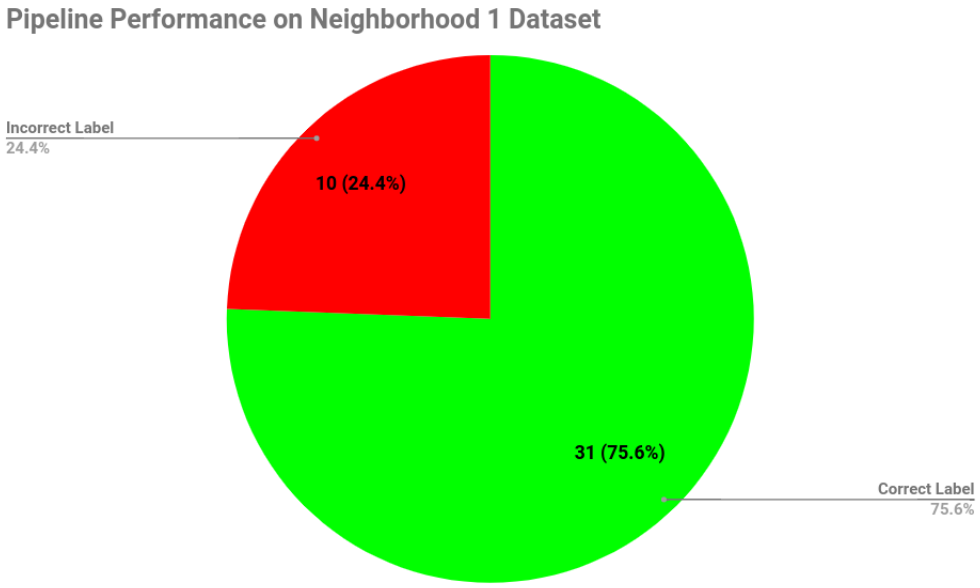

Figure 9.13: Performance on "Neighborhood 1" dataset in producing correctly labeled pointcloud segments.

9.7.2 Phase two performance

The pipeline was subsequently run on the “Neighborhood 2” dataset, which measured 317 seconds in length, and achieved greater accuracy (see figure 9.15). This second test produced a total of 35 labeled segments with 31 evaluated as “correct” and four as “incorrect.”

The pipeline performed better on this dataset due to the higher incidence of vehicles compared to Neighborhood 1, as the pipeline is able to consistently produce vehicle labels. Two of the four incorrect labels suffered from synchronization errors due to the image crop not containing the appropriate object, and the remaining two incorrect labels stemmed from poor segmentation and classification of a background object instead of the intended foreground object (occlusion).

The aggregate performance of the pipeline, when combining the results of both datasets Neighborhood 1 and Neighborhood 2, reached the level of 81.6% accuracy. With the two collection’s combined duration of 663 seconds, the pipeline created a correctly labeled segment every 10.7 seconds and an incorrectly labeled segment every 47.4 seconds. The diversity of the resultant labeled dataset was extremely limited due to time-imposed practical constraints. The area processed by our pipeline was directly in front of the collection vehicle and the collection was culled to 10 meters of maximum depth with a 70-degree horizontal field of view. This culling largely limited the ground-truth diversity of object types available for classification to a mixture of street objects, such as parked vehicles, street signs, motorcycles, and landscaping. Most of the physical space processed by the pipeline contained open road. This produced a dataset heavily biased towards our limited collection environment, with Neighborhood 1’s results comprised of over 50% vehicles. Moreover, 100% of Neighborhood 2’s correctly labeled results

Figure 9.14: Occlusion Example. The tree pointcloud segment labeled as a car due to foreground of the image crop.

were from vehicles. Our implementation requires testing in different operating environments to assess its effectiveness in producing diverse datasets.

9.8 Conclusions

This chapter described a novel approach to 3D object detection. Synchronized 2D imagery and 3D lidar data was collected where standardized neural networks detected and classified features in the image space and through a correlation process labeled the corresponding 3D segmented lidar data. This was used to train a 3D NN classifier. This approach has a couple of additional major benefits. It permits training the 3D classifier without the laborious step of hand labeling datasets. Second, the lidar with the trained 3D NN can operate at night. This is a significant advantage over standard camera systems. While this approach to 3D object detection showed promising initial results, it requires additional research to show the advantages of over established dataset creation techniques.

Within the context of the MTX objectives, this research highlights ways in which data collected

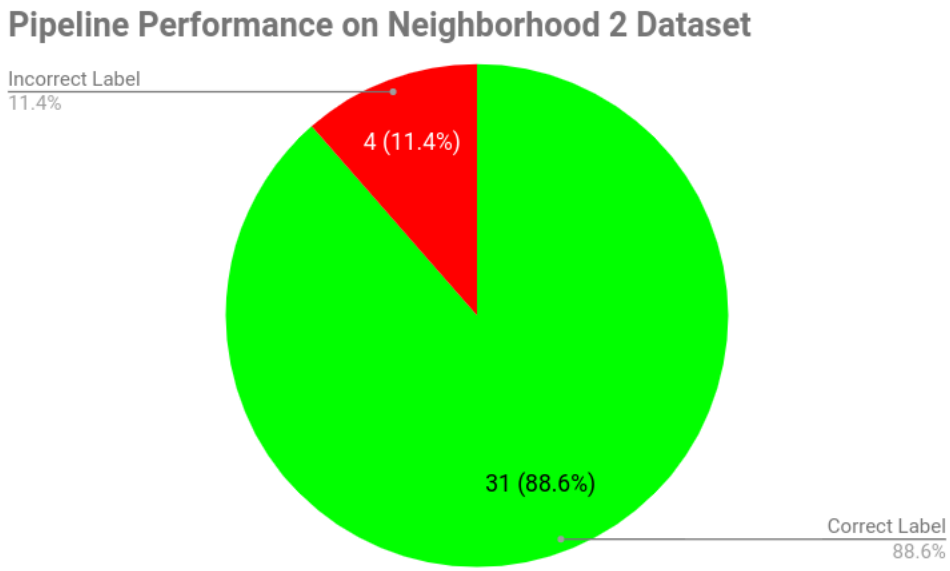

Figure 9.15: Performance on "Neighborhood 2" dataset in producing correctly labeled pointcloud segments.

by mobile vehicles can be collected and parsed at the network's edge so that rather than sending dense sensor streams across the wireless networks, compact data streams with location descriptions of detected objects can be transmitted instead. This improves the performance of the NCS system is several important ways. It decreases reliance on the network for situational awareness and permits faster reactions by system agents (since the confirmation of the detected object doesn't have to be verified by a more distance processor). Additionally by not having as much data to transmit, the mobile nodes can increase coverage since wireless SNR can be reduced since data volume isn't as great.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 10:
Conclusions and Recommendations

MTX was a first of its kind for NPS and possibly for the Navy. It was designed with a focus on a single research concept (an expeditionary UxV NCS) but included a number of distinct research initiatives designed to support the concept. For the inaugural event, the expeditionary operational scenario was a NAVSPECWAR DA mission where the UxV NCS was used in to support the SEALs in all mission phases. MTX was strongly supported by both operational commands (principally NAVSPECWAR and COMTHIRDFLT) and a Navy lab (NIWC).

Viewed as a single system composed of individual subcomponents, the UxV NCS can provide a kind of overwatch for expeditionary forces. It can collect and disseminate ISR information for improved situational awareness, provide direct and indirect fire support, provide transportation and resupply and provide flexible communications. Through this support, it can reduce the logistics tail that is associated with combat operations.

In figure 1.1, the UxV NCS model described fundamental components that comprise the system. They include the manned and unmanned systems, the communications layer, the information layer, the control and autonomy layer, the HRI layer and the cyber-security layer.

In after action discussions, the NSW SEAL element participating in MTX recognized the potential of the UxV NCS to support expeditionary warfare. While there is still a lot of work to be done, the core idea of having an overwatch that could reduce logistics support while simultaneously providing a variety of different *services* was validated.

With MTX, we were able to conduct research and experimentation on four of the six layers. At the lowest level, we were able to simultaneously deploy aerial, surface and underwater unmanned vehicles to support the DA exercise. The communications layer was demonstrated through both wireless radio and acoustic networks and multihop transmission between nodes that, in one instance, demonstrated telemetry and sensor data being sent from the UUV through the USV (acting a mobile acoustic to radio router), through the UAV to the CENETIX data server.

At the information layer, the wireless network transmitted COTs XML messages from all the unmanned systems. These messages, at a minimum, delivered vehicle state at 1 Hz. Camera and sonar sensor data was also transmitted through the network. On all the unmanned systems SNMP agents collected and disseminated signal statistics into the network. The control and autonomy layer were represented by several research initiatives that include optimal trajectory calculation, near-real time, near-optimal, positioning of the system agents and experimentation into optimal positioning of agents taking into consideration signal statistics of the wireless network. Not represented in the experimentation was the HRI layer or considerations for cyber-security.

MTX was a challenging endeavor. Deploying the unmanned systems on SCI for the first time, developing the hardware and software for the wireless network, constructing, deploying and recovering the administrative wireless network, making logistic coordination for sea and land ranges and conducting coordination with naval particpiating commands (NIWC, COMTHIRD-FLT, NSW, NEDU, NPS) required an outstanding effort by all participants.

Maybe the most important lesson learned in MTX is that for the UxV NCS to fulfill its promise, serious consideration is required for minimizing the logistics associated with the unmanned systems. Maybe the most difficult is the deployment, recovery and maintenance of unmanned systems. As an example, the ScanEagle UAS deployment and recovery system consisted of a pneumatic launch and a Skyhook recovery system. This is expensive and requires significant support to transport and operate. The FLARES system is a quadrotor-based alternative that would greatly reduce the costs associated with deploying and recovering the UAS.

For the UUV, the challenge is to deploy and recover the system rapidly since the slower speeds requires energy and time to put the platform into the designated operating area. Methods to speed up this process increase the utility of the system. Possibilities include aerial insertion and undersea docking. For USVs like the SeaFox, a major advantage of the platform is the duration. On a single fuel load, it can stay loitering in position over days and even weeks. It can also handle heavy equipment including computers and sensors. A disadvantage is the platform can be relatively easily detected by opposing forces. Having a USV that could temporarily submerge to avoid contact could be potentially advantageous.

Simulation is an important component of testing, validation and training of UxV NCS. Future iterations of MTX would more thoroughly take advantage of simulation for testing control algorithms, specially to confirm performance of a NCS with significantly greater number of UxVs. Additionally simulation can be used simultaneously with the fielded system to validate performance. A key part of a next MTX would be reacting to opposing forces. This could be done initially through simulation.

Bibliography

[1] Michael Gilday. Navplan 2021, 2017.

[2] Thomas W Harker, MM Gilday, and David H Berger. Department of navy unmanned campaign framework. Technical report, DEPARTMENT OF THE NAVY WASHINGTON DC, 2021.

[3] John D Day and Hubert Zimmermann. The osi reference model. *Proceedings of the IEEE*, 71(12):1334–1340, 1983.

[4] Eric Schmidt, Bob Work, Safra Catz, Steve Chien, Chris Darby, Kenneth Ford, Jose-Marie Griffiths, Eric Horvitz, Andrew Jassy, William Mark, et al. National security commission on artificial intelligence (ai). Technical report, National Security Commission on Artificial Intelligence, 2021.

[5] European Commission. White paper on artificial intelligence: A european approach to excellence and trust, 2020.

[6] Noah Wachlin. Robust time-varying formation control with adaptive submodularity, 2018.

[7] Bryan Lowry. Distributed submodular optimization for a uxv networked control system, 2020.

[8] Wendy Ellens and Robert E Kooij. Graph measures and network robustness. *arXiv preprint arXiv:1311.5064*, 2013.

[9] Gerardo Lafferriere, J Caughman, and A Williams. Graph theoretic methods in the stability of vehicle formations. In *Proceedings of the 2004 American Control Conference*, volume 4, pages 3729–3734. IEEE, 2004.

[10] Xiwang Dong, Bocheng Yu, Zongying Shi, and Yisheng Zhong. Time-varying formation control for unmanned aerial vehicles: Theories and applications. *IEEE Transactions on Control Systems Technology*, 23(1):340–348, 2014.

[11] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

[12] Andreas Krause and Daniel Golovin. Submodular function maximization. *Tractability*, 3:71–104, 2014.

[13] Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, 42:427–486, 2011.

[14] YV HAIMES YV, Leon S Lasdon, and DA Da Wismer. On a bicriterion formation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man and Cybernetics*, (3):296–297, 1971.

[15] David Silver and Joel Veness. Monte-carlo planning in large pomdps. Neural Information Processing Systems, 2010.

[16] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[17] Benjamin P. Keegan. Uav position optimization for wireless communications, 2018.

[18] Alan R Washburn. *Search and detection*. Institute for Operations Research and the Management Sciences, 2002.

[19] Daniel H Wagner and W Charles Mylander. *Naval operations analysis*. Naval Institute Press, 1999.

[20] Paul C Etter. *Underwater acoustic modeling and simulation*. CRC press, 2018.

[21] Sean Kragelund, Claire Walton, and Isaac Kaminer. Sensor-based motion planning for autonomous vehicle teams. In *OCEANS 2016 MTS/IEEE Monterey*, pages 1–8. IEEE, 2016.

[22] Claire Walton, Chris Phelps, Qi Gong, and Isaac Kaminer. A numerical algorithm for optimal control of systems with parameter uncertainty. *IFAC-PapersOnLine*, 49(18):468–475, 2016.

[23] Claire L Walton, Qi Gong, Isaac Kaminer, and Johannes O Royset. Optimal motion planning for searching for uncertain targets. *IFAC Proceedings Volumes*, 47(3):8977–8982, 2014.

[24] Claire Walton, Sean Kragelund, and Isaac Kaminer. The application of 'optimal search'to marine mapping. In *OCEANS 2016 MTS/IEEE Monterey*, pages 1–6. IEEE, 2016.

[25] Chin B Ang. Study of a terrain-based motion estimation model to predict the position of a moving target to enhance weapon probability of kill. Technical report, Naval Postgraduate School Monterey United States, 2017.

[26] Qi Gong, Wei Kang, and I Michael Ross. A pseudospectral method for the optimal control of constrained feedback linearizable systems. *IEEE transactions on automatic control*, 51(7):1115–1129, 2006.

[27] Teal A. Peterson. Cross-domain identification of road networks using domain-adapted convolutional neural networks, 2020.

[28] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

[29] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1):151–175, 2010.

[30] Maggi Kelly, Barbara Allen-Diaz, and Norma Kobzina. Wieslandervegetation, 2019.

[31] Maggi Kelly, Barbara Allen-Diaz, and Norma Kobzina. Digitization of a historic dataset: the wieslander california vegetation type mapping project. *Madroño*, 52(3):191–201, 2005.

[32] Maggi Kelly, Ken-ichi Ueda, and Barbara Allen-Diaz. Considerations for ecological reconstruction of historic vegetation: Analysis of the spatial uncertainties in the california vegetation type map dataset. *Plant Ecology*, 194(1):37–49, 2008.

[33] DigitalGlobe. Aboutdigitalglobe, 2019.

[34] DigitalGlobe. Thedigitalglobeconstellation, 2019.

[35] DigitalGlobe. Aboutmydigitalglobe, 2019.

[36] Andrew K. Watson. Automated creation of labeled pointcloud datasets in support of machine learning-based perception, 2017.

[37] Logitech c920 and c910 fields of view for RGBDtoolkit. `http://velodynelidar.com/hdl-32e.html`, 2017. [Online; accessed 30-October-2017].

[38] ROS Integration with other libraries. `http://www.ros.org/integration/`, 2017. [Online; accessed 4-December-2017].

[39] Igor Bogoslavskyi and Cyrill Stachniss. Fast range image-based segmentation of sparse 3d laser scans for online operation. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 163–169. IEEE, 2016.

[40] Mark De Deuge, Alastair Quadros, Calvin Hung, and Bertrand Douillard. Unsupervised feature learning for classification of outdoor 3d scans. In *Australasian Conference on Robitics and Automation*, volume 2, page 1, 2013.

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