VERIFICATION OF AUTONOMOUS SYSTEMS: DEVELOPMENTAL TEST AND EVALUATION OF AN AUTONOMOUS UAS SWARMING ALGORITHM COMBINING SIMULATION, FORMULATION AND LIVE FLIGHT

THESIS

Corey A. Combs, Capt, USAF

AFIT-ENV-MS-19-M-168

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DISTRIBUTION STATEMENT A
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views expressed in this document are those of the author and do not reflect the official policy or position of the United States Air Force, the United States Department of Defense or the United States Government. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

AFIT-ENV-MS-19-M-168

VERIFICATION OF AUTONOMOUS SYSTEMS: DEVELOPMENTAL TEST AND EVALUATION OF AN AUTONOMOUS UAS SWARMING ALGORITHM COMBINING SIMULATION, FORMULATION, AND LIVE FLIGHT

THESIS

Presented to the Faculty

Department of Systems Engineering and Management

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Systems Engineering

Corey A. Combs, BS

Capt, USAF

March 2019

DISTRIBUTION STATEMENT A
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AFIT-ENV-MS-19-M-168

VERIFICATION OF AUTONOMOUS SYSTEMS: DEVELOPMENTAL TEST AND EVALUATION OF AN AUTONOMOUS UAS SWARMING ALGORITHM COMBINING SIMULATION, FORMULATION, AND LIVE FLIGHT

THESIS

Corey A. Combs, BS
Capt, USAF

Committee Membership:

John M. Colombi, Ph.D.
Chair

Lt Col Amy Cox, Ph.D.
Member

David Jacques, Ph.D.
Member

AFIT-ENV-MS-19-M-168

Abstract

This research was driven by the increase of autonomous systems in the current millennium and the challenging nature of testing and evaluating their performance. A review of the current literature revealed proposed methods for verifying autonomous systems, but few implementations. It exposed several gaps in the current verification and validation methods and suggested goals for filling them. Through the use of modeling, software in the loop (SITL), and flight test, this research verified an autonomous swarming algorithm for unmanned aerial systems (UAS) and validated an exemplar of a testing framework.

Thirteen sets of three-vehicle swarm data produced over two days of flight testing provided a baseline algorithm analysis. During these tests, vehicle separation distances deviated an average of 5.61 meters from the ideal state, with separation distance violations $< 6.39\%$ of the time. The swarm achieved a 0.27 m average deviation and 0.43% violation in the best cases. Average packet loss between vehicles was 4.94% at a 5 Hz update rate, with an optimal communication lag < 0.04 seconds.

The multi-faceted empirical analysis created through the pairing of qualitative and quantitative analysis provided a complete understanding of vehicle behavior. This analysis also identified various areas of improvement for the algorithm and testing framework. The outcomes of this research formed a baseline testing continuum that is adaptable to a variety of follow-on investigations into formal verification of autonomous systems.

AFIT-ENV-MS-19-M-168

This thesis is dedicated to all who have helped me through this journey. To my wife, for staying by my side since day one. Having learned I would be moving across the country when we started dating, you uprooted your life to be beside me as I pursued this opportunity. I thank you for putting up with my many late nights and for being my rock during stressful times. There was no greater day than the day you became my wife and I look forward to our future together. To my parents, you have always been a guiding light in my life. Your love, guidance, and encouragement since I was young have made me into who I am today. I cannot thank you enough for everything you have done for me and hope to be the same example for my children some day. To my friends and fellow students, thank you for the memories we have created during the past months. It was important to remember to step away from work to still experience life and its many blessings and I thank you for the time we've spent together. I hope to cross paths with you in the future as we continue our careers. This has been an amazing opportunity and learning experience for all of us.

Acknowledgements

The highest of appreciation is given to Dr. John Colombi for his guidance, assistance, rigor, and expertise for the duration of this research effort. Many mentally stimulating discussions provided valuable insight and generated many ideas to pursue. Thanks are also given to Dr. David Jacques and Lt Col Amy Cox for their assistance in preparation for flight testing and their challenging questions during this endeavour. Special acknowledgement is due to Mr. Jeremy Gray for his assistance in response to my many coding questions. Additional thanks to the ANT Center Lab technicians and all involved in executing flight tests at the Camp Atterbury UAS Range. Their expertise in the realm of unmanned systems was invaluable, and without their help, successful flight testing of this algorithm would not have been possible.

Corey A. Combs

Table of Contents

Page

Abstract iv

Acknowledgements vi

List of Figures ix

List of Tables xii

List of Abbreviations xiii

I. Introduction 1

1.1 General Issue 1
1.2 Scope 4
1.3 Research Objectives and Questions 5
1.4 Methodology 6
1.5 Assumptions and Limitations 7
1.6 Preview 7

II. Literature Review 8

2.1 Overview 8
2.2 Utility of UAS Swarms 8
2.3 Algorithm Review 10
2.4 Autonomous Systems Testing 13
2.5 Summary 19

III. Methodology 20

3.1 Overview 20
3.2 Test Objectives, Metrics, & Data 20
3.3 Preliminary SITL Verification 21
3.4 Algorithm Analysis and Modification 26
3.5 Flight Test Methodology 37
3.6 Summary 46

IV. Analysis and Results 48

4.1 Overview 48
4.2 SITL Test Results 48
4.3 Flight Test Overview 52
4.4 Data Analysis Procedures 58
4.5 Flight Test Data 60
Final Observations in Quantitative Analysis 73

Page

4.6 Summary Flight Test & Communication Data 76
4.7 Summary 81

V. Conclusion 83

5.1 Overview 83
5.2 Significance of Findings 83
What improvements, if any, are required to improve the swarming architecture for testing? 84
What role does quantitative and qualitative assessment have in the verification of autonomous systems? 85
What measure(s) or metric(s) can be used to evaluate swarm behavior? 86
How well do these measure(s) or metric(s) quantify swarm behavior and algorithm performance? 86
5.3 Heuristics, Lessons Learned, and Recommendations 86

Appendix A. SITL Configuration Guide 94

Appendix B. Algorithm and Analysis Code Repository 100

Appendix C. Flight Testing 5028 for TRB/SRB 127

Appendix D. Flight Test After Action Report 165

Appendix E. Additional Figures 169

Bibliography 170

List of Figures

Figure Page

1 Block diagram of the Original Swarming Architecture found in Allen (2018) 13

2 SysML block definition diagram of the Original SITL Architecture 15

3 SysML interal block diagram of the Original SITL Communication Protocols 16

4 UgCS visualization environment with 3 simulated vehicles and telemetry. 23

5 SysML bdd of the New SITL Architecture 24

6 SysML ibd of the New SITL Communication Protocols 25

7 High level view of SITL communication structure 25

8 Separation and Flight Floor Rules from Allen (2018) 26

9 Separation Rule with varying r_{min} values for $b = 0.25$ and $v_{max} = 2m/s$ 31

10 Separation Rule with varying b values for $r_{min} = 5m$ and $v_{max} = 2m/s$ 31

11 Separation Rule with varying v_{max} values for $r_{min} = 5m$ and $b = 0.25$ 32

12 Reynolds+ rule set with $b = 0.25$, $r_{min} = 5m$, $f_{min} = 4m$ and $v_{max} = 2m/s$ 33

13 2D Stable State with $b = 0.25$, $r_{min} = 5m$, $f_{min} = 4m$ and $v_{max} = 2m/s$ 34

14 3D Stable State with $b = 0.25$, $r_{min} = 5m$, $f_{min} = 4m$ and $v_{max} = 2m/s$ 34

15 Ubuntu SITL interface showing a snapshot of the 'rplusONE' algorithm with full debug outputs and the Ardupilot simulation on the right. 36

16 Illustration of data flow among vehicles in the swarm. 37

Figure Page

17 UAS runway at Camp Atterbury. North is up. 38

18 LCM spy interface used on node 4 to monitor traffic. 40

19 LCM spy individual channel live view with graphs. 40

20 Screenshot of the planned flight path for the lead vehicle. 44

21 AFIT Flight Test Facility. 46

22 Flight 1, 2D flight paths with the lead vehicle in 'ALT_HOLD'. 53

23 Flight 3, 2D flight path of first three vehicle swarming flight. 55

24 Flight 3, Vehicle-Vehicle distance during the first three vehicle swarming flight. 56

25 Flight 8, 2D flight path showing unresponsive behavior of Vehicle 2. 57

26 UgCS telemetry player window showing three vehicle tracks during the same time period. 59

27 Flight 6, Vehicle paths with the lead vehicle under autonomous and manual control. 62

28 Flight 6, Vehicle Velocity vs Commanded Velocity. $v_{max} = 2m/s$, $r_{min} = 5m$, $b = 0.25$, $u = 5Hz$. 62

29 Flight 6, Vehicle 2 Velocity vs Commanded Velocity in NED. $v_{max} = 2m/s$, $r_{min} = 5m$, $b = 0.5$, $u = 5Hz$. 63

30 Fourier Transform of Vehicle 2's Commanded Velocity during Flight 6. $v_{max} = 2m/s$, $r_{min} = 5m$, $b = 0.5$, $u = 5Hz$. 64

31 Flight 6, Vehicle-Vehicle Distances with the lead vehicle in 'ALT_HOLD'. 65

32 Flight 15, Plot of vehicle paths under autonomous control. 66

33 Flight 15, Vehicle-Vehicle distances with $v_{max} = 3m/s, r_{min} = 5, b = 0.25, u = 5Hz$. 67

Figure Page

34 Flight 15-SITL, Plot of vehicle paths in SITL under autonomous control. 68

35 Flight 15-SITL, Vehicle-Vehicle distances with $v_{max} = 3m/s, r_{min} = 5, b = 0.25, u = 5Hz$. 68

36 Flight 12, Vehicle-Vehicle distances with $v_{max} = 3m/s, r_{min} = 10, b = 0.25, u = 5Hz$. 69

37 Flight 12, Plot of vehicle paths with $v_{max} = 3m/s, r_{min} = 10, b = 0.25, u = 5Hz$. 70

38 Flight 15, Vehicle-Vehicle distances with $v_{max} = 3m/s, r_{min} = 5, b = 0.25, u = 5Hz$. 71

39 Flight 7, Vehicle 2 commanded velocity magnitude versus distances with $v_{max} = 2m/s, r_{min} = 10, b = 0.25, u = 5Hz$. 72

40 Flight 7, Vehicle 3 commanded velocity magnitude versus distances with $v_{max} = 2m/s, r_{min} = 10, b = 0.25, u = 5Hz$. 73

41 Flight 12, Sample of time delay between commanded and realized velocity magnitude. 74

42 Flight 9, 2D plot with $v_{max} = 2m/s, r_{min} = 5, b = 0.5, u = 5Hz$ and Vehicle 2 unresponsive. 75

43 Flight 8, Vehicle half-median distance with $v_{max} = 2m/s, r_{min} = 5, b = 0.5, u = 5Hz$ and Vehicle 2 unresponsive. 75

44 Flight 8, Vehicle 2 Commanded vs Vehicle Velocity. 76

45 LCM log start example with Vehicle 2 being the first to broadcast a message. 78

List of Tables

Table Page

1 Table of *'argparse'* parameters in 'rplusONE' code 35

2 Algorithm behavior by mode adapted from Allen (2018). 39

3 Summary of planned missions for flight testing. The value of $f_{min} = 10$ for all flights. 42

4 Summary of executed missions during flight testing. The value of $f_{min} = 10$ for all flights. 61

5 Summary of successful autonomous missions during flight testing. 65

6 Summary of average, ideal vehicle-vehicle spacing deviations and r_{min} violation percentage for successful autonomous flights. The value of $f_{min} = 10$ for all flights. 77

7 Summary of vehicle-vehicle spacing and r_{min} violation percentage for successful autonomous flights. The value of $f_{min} = 10$ for all flights. 77

8 Summary of average half-median or distance to centroid deviation for successful autonomous flights. The value of $f_{min} = 10$ for all flights. 78

9 Summary of time delay between send and receive of LCM messages. All logs at 5 Hz update. 79

10 Summary of packet transmission loss percentage between vehicles. 80

11 Summary of algorithm loop frequency. 80

12 Summary of Metrics Used. 87

13 Summary of Lessons Learned and Future Recommendations. 93

List of Abbreviations

Abbreviation		Page
UAS	Unmanned Aerial Systems	1
DoD	Department of Defense	1
RDT&E	Research, Development, Test, and Evaluation	1
S&T	Science and Technology	1
COI	Community of Interest	1
ATEVV	Autonomy COI Test and Evaluation, Verification and Validation	2
V&V	Verification and Validation	2
T&E	Test and Evaluation	2
DT&E	Developmental Test and Evaluation	2
OT&E	Operational Test and Evaluation	2
AFIT	Air Force Institute of Technology	3
WPAFB	Wright Patterson Air Force Base	3
SE	Systems Engineering	3
SITL	Software in the Loop	4
SysML	Systems Modeling Language	4
STAT COE	Scientific Test and Analysis Techniques Center of Excellence	5
DOE	Design of Experiments	6
RFID	radio frequency identification	9
DJI	Da-Jiang Innovations	9
ISR	Intelligence, Surveillance, and Reconnaissance	10
LOCUST	Low-Cost UAV Swarming Technology	10

Abbreviation		Page
NED	North-East-Down	11
GPS	global positioning systems	11
RTK	real time kinematic	11
BBB	Beagle Bone Black	12
BATMAN	Better Approach To Mobile Ad-hoc Networking	12
LCM	Lightweight Communications Marshalling	12
COTS	commercial off-the-shelf	12
HIL	hardware-in-the-loop	13
GCS	Ground Control Station	15
bdd	block definition diagram	15
ibd	internal block diagram	15
UAVs	Unmanned Aerial Vehicles	17
DOE	Design of Experiments	17
m	meters	18
m/s	meters per second	18
s	second	18
MFR	Military Flight Release	22
UgCS	Universal ground Control Software	22
UDP	User Datagram Protocol	23
TCP	Transmission Control Protocol	23
VM's	Virtual Machines	24
AGL	above ground level	27
CRM	Crew Resource Management	38
LiPo	Lithium Polymer	39

Abbreviation		Page
mAh	milliamp hour	39
USB	Universal Serial Port	39
ssh	secure shell	39
TRB	Technical Review Board	45
SRB	Safety Review Board	45
AAR	After Action Report	45
3DR	3D Robotics	45
mph	miles per hour	52
RTK	real-time kinematic	90

VERIFICATION OF AUTONOMOUS SYSTEMS:
DEVELOPMENTAL TEST AND EVALUATION OF AN
AUTONOMOUS UAS SWARMING ALGORITHM COMBINING
SIMULATION, FORMULATION, AND LIVE FLIGHT

I. Introduction

1.1 General Issue

Across the globe, automated and autonomous systems have become ever-present in our society. From the use of multirotor vehicles in agriculture and geography (Huuskonen and Oksanen, 2018), to self-driving cars, and the military use of Unmanned Aerial Systems (UAS), the rapid pace of technological growth in the sector has caught the interest of private and public organizations alike. Countless universities and businesses have begun research efforts to develop new architectures, behavior algorithms, and navigation techniques, but the Department of Defense (DoD) continues to be one of the larger investors with at least $6.97B requested for procurement, Research, Development, Test, and Evaluation (RDT&E), and construction in 2018 (Gettinger, 2017). This request is a reflection of the 2011 memo written by then Secretary of Defense Robert Gates which designated Autonomy as one of seven Science and Technology (S&T) investment areas for the upcoming FY 13-17 Program Objective Memorandum (Ahner and Parson, 2016). As a result of this memo, the Autonomy Community of Interest (COI) was formed, along with six other teams corresponding to the seven investment areas (Clark et al., 2015).

In May 2015, the Autonomy COI Test and Evaluation, Verification and Validation (ATEVV) Working Group published a Technology Investment Strategy which highlighted challenges, gaps, and goals for the autonomy community in 2015-2018. Automation and autonomy definitions in the context used by the ATEVV are as follows:

- "Automation: The system functions with no/little human operator involvement; however, the system performance is limited to the specific actions it has been designed to do. Typically these are well-defined tasks that have predetermined responses (i.e., simple rule-based responses).

- Autonomy: The system has a set of intelligence-based capabilities that allows it to respond to situations that were not pre-programmed or anticipated (i.e., decision-based responses) prior to system deployment. Autonomous systems have a degree of self-government and self-directed behavior (with the humans proxy for decisions)." (Clark et al., 2015)

Four challenges previously identified in FY13 by the ATEVV include state-space explosion, unpredictable environments, emergent behavior, and human-machine communication. These are complimented by six gaps in the current Verification and Validation (V&V) structure; including lack of modeling, design, and interface standards and lack of autonomy Test and Evaluation (T&E) capabilities. In order to address these challenges and gaps, five goals were developed to guide RDT&E efforts in the DoD. This thesis will focus on two goals directly (Clark et al., 2015):

1. "Evidence-Based Design and Implementation"

2. "Cumulative Evidence through RDT&E, Developmental Test and Evaluation (DT&E), & Operational Test and Evaluation (OT&E)"

Since the publishing of the initial memo, Air Force Institute of Technology (AFIT), partnered with other organizations at Wright Patterson Air Force Base (WPAFB) and bases across the United States, has invested in the development, analysis, and test of autonomous systems. Students and faculty in the Systems Engineering (SE), Aerospace, and Electrical Engineering programs have authored multiple theses and papers exploring these areas. Some examples include a Unified Command and Control Architecture (Gray, 2015), Flocking Operations with Formation Drag Reduction (Lambach, 2014), Autonomous Rule Weighting in Flocks (Kaiser, 2014), and most recently a Swarm Architecture using an Ad Hoc Mesh Network (Allen, 2018). These efforts have made advances in developing autonomous capabilities and establishing baselines for future work, to include T&E. However, a centralized effort explicitly focused on verification through DT&E of an autonomous system has not yet been accomplished.

In order to develop a DT&E strategy, one must understand that it does not just occur once in the system's life cycle, but is an ongoing process. "It verifies a system is built correctly and meets the technical requirements..." and "informs the systems engineering process and acquisition decisions, to help manage the design and programmatic risks, and to evaluate the combat capability of the system and its ability to provide information to the users" (Cortes, 2014a). However, unique challenges arise when dealing with autonomous systems, as most guidance in the DoD is focused on hardware requirements and not the autonomy software (Defense Science Board, 2012). To combat this, modeling and simulation in combination with real-world testing has been suggested "to explore thousands of test cases, statistically measuring system performance against the desired standard"... and "ensure that the modeled and real-world behavior match" (David and Nielsen, 2016). This research

aims to establish an autonomous system test strategy and provide empirical data to validate its effectiveness.

1.2 Scope

The focus of this work extends the swarm architecture baseline developed by Allen (2018). Allen employed multiple X-8 multirotor aircraft communicating over a Mesh Ad Hoc Network; the aircraft leverage control laws based on natural flocking to inform their flight. Allen validated the development work with extensive Software in the Loop (SITL) testing and limited flight testing. The underlying algorithm of the swarm leveraged established research on bio-mimicry (Reynolds, 1987); providing rules of separation, flight floor, cohesion, and alignment. The relative weights of these rules were managed using a bucket priority scheme, which acted as an arbitrator on the total velocity vectors sent to the vehicle.

Under algorithm control, each vehicle could be considered an 'automated' agent based on the definitions. However, when analyzing swarm behavior, solutions for individual agent behavior are less tractable as additional agents, and environmental conditions get introduced. This swarm has self-governance and self-directed behavior when no human input is received. It provides a means to experiment with various system states and attempt to replicate a biological phenomenon.

Systems Modeling Language (SysML) was used in the creation of a DT&E program to include SITL and flight testing. This testing program did not aim to be all-inclusive nor an exhaustive endeavor, but focused on the ATEVV goal #2 of recording, aggregating, and reusing T&E data through an SE approach (Clark et al., 2015). By establishing a tested baseline algorithm, future research may introduce new rules and expand the testing framework to include fault injection or other environmental factors. SITL testing was used to verify the accuracy of flight tests and

swarm behavior in an open environment. Lastly, this thesis incorporated insight from the OSD Scientific Test and Analysis Techniques Center of Excellence (STAT COE), to address challenges published in a 2016 Workshop Report on Test and Evaluation of Autonomous Systems (Ahner and Parson, 2016).

1.3 Research Objectives and Questions

In addition to the two goals previously identified, four of the eight challenges presented in (Ahner and Parson, 2016) served as focuses for this research. These challenges were; requirements and measures, design for test, test adequacy and integration, and testing continuum. Combining these goals and challenges resulted in the overarching objective of verification of an autonomous UAS swarming algorithm through developmental testing. A further breakdown follows:

1. Determine the suitability of the current algorithm for testing, determining and implementing changes to facilitate testing if necessary.

2. Use simulation and flight testing to cumulatively gather qualitative and quantitative data to evaluate swarm performance against chosen metrics and establish a baseline performance level.

3. Develop heuristics using the gathered data to establish a foundation for future autonomous system testing methods.

The above culminated in the completion of SITL and open-environment flight testing of the algorithm in order to answer the following research questions:

1. *How can the swarming architecture be improved for testing?*

2. *What role does quantitative and qualitative assessment have in the verification of autonomous systems?*

3. *What measure(s) or metric(s) can be used to evaluate swarm behavior?*

4. *How well do these measure(s) or metric(s) quantify swarm behavior and algorithm performance?*

1.4 Methodology

A significant effort of this research included improving and implementing existing methods to establish a testing program for an autonomous system. These methods were adapted to fit the specific needs of the testing program while maintaining flexibility for future modification. The primary testing method chosen to evaluate the algorithm was SITL. Use of SITL testing was prevalent throughout the literature as a developmental tool to verify software before flight testing hardware configurations.

Chapter III outlines the original and updated configuration of the SITL environment and its relation to the hardware configuration. Once established, the updated SITL architecture allowed evaluation of the principles and logic of the original algorithm. Results of these early tests were used to modify the algorithm and create a stable baseline for flight testing.

Flight test planning incorporated best practices from the literature and progressed from simple to more complex swarm states. Organization of the test plan joined qualitative analysis from SITL testing with a Design of Experiments (DOE) factorial design. This plan culminated in two successful days of flight testing at Camp Atterbury Army Reserve Base in Indiana with up to three-vehicles in a swarm. Chapter III contains a full explanation of the test plan, procedures, and data collection methods used for these flight tests.

1.5 Assumptions and Limitations

When considering the applicability of this research, it is essential to understand the limitations of the methods used to analyze the autonomous system's behavior. SITL testing was limited to three vehicles without environmental influences, vehicle degradation, battery life, or communications bandwidth limitations. The original hardware configuration from Allen remained unchanged for this research.

Revisions to the algorithm did not alter the fundamental rule set through the introduction of controls theory. Good weather throughout open-air flight tests provided limited environmental influences on swarm performance. Additionally, flight tests used a conservative set of parameters and limited flight envelope as this was the first live-flight demonstration of this algorithm. The small data set collected during flight tests limited the statistical power of results.

1.6 Preview

Chapter I provided background to the topic area, introduced the problem, discussed methods for testing, and outlined the limitations of the research. Chapter II discusses the fundamental idea of the swarm behavior, its potential application to military or other uses, past and current research in the area of multi-vehicle swarming, and an overview of the architecture developed by Allen. The testing methodologies used for this research, architecture models, algorithm modifications, flight testing concepts, and testing framework are explained in Chapter III. Chapter IV provides analysis of the swarm architecture from multiple sources to include open-air flight testing, SITL testing, and other test analysis. Comparisons among these various sources are made, with consideration to internal and external factors affecting the results. Chapter V summarizes conclusions and highlights significant findings within the data as well as heuristics, lessons learned, and recommendations for future research.

II. Literature Review

2.1 Overview

This chapter aims to familiarize the reader with current and past advances in swarming architecture development. Similar swarm algorithms, testing methods, and their results will be highlighted alongside potential applications for such a swarm. Explanation of the baseline algorithm used for this research will familiarize the reader with the concepts being implemented without diverging from the test and evaluation focus of the latter chapters. Challenges facing testers of autonomous systems in industry and the DoD will be addressed to set a focus for the following discussion on designing for test. Current and previous methods of autonomous systems testing will be explored to provide a foundation for the testing methodology in development and suggest metrics for evaluation.

2.2 Utility of UAS Swarms

The availability of UAS have dramatically increased in the past decade, both in private and commercial use. In 2016, Goldman Sachs estimated a total of $100 billion may be spent on drones between 2016 and 2020. Of this huge sum, "the commercial segment would be the fastest-growing, notably in construction (accounting for $11.2B), agriculture ($5.9B), insurance ($1.4B), and infrastructure inspection ($1.1B)" (Standage, 2017). This prediction does not seem far fetched, having noted that $6.97B of the 2018 DoD budget is dedicated to UAS and one can only predict that this number will increase in future budgets (Gettinger, 2017). Setting aside the application of UAS in the DoD for a moment, it is important to explore some of the other commercial industries in the aforementioned article to understand the impor-

tance of testing these systems and what applicability a swarm architecture could have in each.

Beginning with agriculture, the relatively low cost of consumer grade UAS equipped with high resolution cameras has allowed farmers to become more knowledgeable about their fields in multiple ways. Perhaps the most obvious is that aerial systems provide immediate imagery data that can be interpreted by the farmers to monitor crop health and watering effectiveness without the need to travel through, or around their fields. Research has demonstrated the potential for aerial mapping and multispectral imagery to give farmers near real-time status on their crops (Hovhannisyan et al., 2018). This was possible through software tools such as Pix4Dmapper, which can assemble imagery using position information stored in the images and create a 3D model to overlay on maps in software such as Google Earth. These same principles, combined with manual soil sample collection, led to the creation of top soil maps for precision agriculture in Huuskonen and Oksanen (2018).

A multitude of opportunities for UAS usage in civil engineering and construction have been proposed, ranging from transportation planning and flood monitoring (Liu et al., 2014), to radio frequency identification (RFID) part tracking (Hubbard et al., 2015). Other examples include building image previews, thermal scanning, remote job walks, and interior missions (Small, 2018). Rakha and Gorodetsky (2018) performed a thorough review of prior research using UAS for visual and thermal inspections of buildings, and completed their own case study using similar techniques. Using a solo Da-Jiang Innovations (DJI) UAS with a FLIR thermal camera and post-processing software, they were able to create a 3D thermal model of a small building. Regarding interior missions, small UAS were used to inspect damage to the Fukushima Daiichi nuclear power plant following the magnitude-9 earthquake and resulting tsunami in March of 2011 (Liu et al., 2014). In many of these cases, a decentralized, collaborative

swarm could offer the potential for faster task completion and a wider range of imaging sensors, as many smaller UAS are limited in payload capacity and flight duration.

Lastly, the use of UAS in the DoD and other government agencies has a long history of research. In a Sandia Report prepared for the US Department of Energy, Feddema et al. (2004) explored airborne and maritime uses for cooperative behaviors with UAS. The history of swarming, behaviors and metrics, mission types, and a basic framework were all outlined in this report. Specifically, "Intelligence, Surveillance, and Reconnaissance (ISR), communications, swarming attacks, deception operations, airfield denial, distributed sensing, and logistics" are missions that are later classified as behaviors for a cooperative swarm, and are the most relevant in regard to this research and military applications. In particular, the US Navy has demonstrated a swarm methodology known as Low-Cost UAV Swarming Technology (LOCUST) to potentially be used in an offensive capacity (Smalley, 2015). In 2018 Raytheon was awarded a contract for $29M to further develop and field this system based on its performance in tests started in 2016 (Today, 2018). This one example reflects the funding and usage rate increases of systems under 55 pounds for military operations around the globe since 2008 (UAS Task Force: Airspace Integration Integrated Product Team, 2011). Additionally, the Department of Homeland Security, Department of State, and other agencies have added UAS to their inventories for missions at home and abroad (Defense Science Board, 2012).

2.3 Algorithm Review

The original control algorithm designed by Allen (2018) was designed for multirotor vehicles, based on three Reynolds rules of separation, alignment, and cohesion published in Reynolds (1987). A fourth flight deck rule was implemented to avoid ground collisions and a fifth rule of a communication range was proposed, but not imple-

mented. This algorithm differed from prior demonstrations involving Reynolds rules, as it used a "bucket method" to aggregate velocity magnitudes instead of weighting each rule. In this schema, the velocity magnitudes for each rule were aggregated in a priority order up to a maximum as set by the user, similar to Reynolds' 'boids' model. Once the "bucket" was full, any additional rules and magnitudes were discarded for that iteration of the algorithm. The rules were prioritized based on a perceived value to safety of flight resulting in the following order; separation, minimum altitude, alignment (or velocity matching), and cohesion.

The velocity vector calculations were developed via a heuristic trial-and-error method through simulations and were not "optimized for any specific behaviors" (Allen, 2018). These vectors were calculated in the North-East-Down (NED) coordinate system using a common local tangent plane. This method required the conversion of global position data for each vehicle to a relative distance in meters, from a common 'home' coordinate for the entire swarm. This was determined to be better suited for swarming behaviors than global position data in latitude, longitude, and altitude. An additional Python function was developed to execute these calculations using methods found in Drake (2002). The effectiveness of this choice was not analyzed; however, the limitations of global positioning systems (GPS) were addressed, focusing mainly the ±2-3 m accuracy. A proposed mitigation of real time kinematic (RTK) differential GPS was suggested, but not implemented.

Decentralized control was chosen to allow the swarm to be independent of a ground station and permit individual members of the swarm to control their position and velocity based on the data being received from other members. This control structure required continual information sharing among vehicles, driving the need to implement a mesh ad-hoc network where each vehicle is a "node" of the network. This allows expansion of the swarm via node hops if distances between two members is too great

for a direct line of communication. Implementing the mesh network on the Linux-based Beagle Bone Black (BBB) computers required the addition of Better Approach To Mobile Ad-hoc Networking (BATMAN) Advanced, an open-source layer-2 routing protocol, since the computers did not have native support for mesh networks. In order to pass the required data among vehicles over this network, Lightweight Communications Marshalling (LCM) was used to publish simple numeric arrays of position and velocity data at a user-determined frequency to a common routing address. Further discussion of network protocols and configurations can be found in Chapter II & III of Allen (2018).

One of the underlying themes of the swarming algorithm implementation was the use of commercial off-the-shelf (COTS) components to demonstrate such a concept for low cost. Prior knowledge and research by students and faculty at AFIT were leveraged to determine which hardware and software to use in its development, with the end result being X-8 Multicopters with Pixhawk autopilots performing vehicle control functions. This differs from other swarm demonstrations such as Vásárhelyi et al. (2014) in which MikroKopter Basicset L4-MEs were utilized and Hauert et al. (2011) where fixed-wing aircraft were used. While there are pros and cons to each autopilot, the Pixhawk was used primarily due to its open-source SITL component, which could be used to test all software before hardware implementation.

Additionally, the Pixhawk contained the gyros, magnetometer, pressure sensor, and processor that Vásárhelyi et al. (2014) had to build onto a custom extension board (PX4, 2019). However, this custom board allowed them to also include their 2.4 GHz radio and micro-computer in an integrated package with their sensors. The rules of repulsion, alignment, cohesion, and formation flight in this demonstration were similar to Allen, but were implemented via short-range, medium-range, and global position constraints. Allen was able to accomplish a similar vehicle payload configuration,

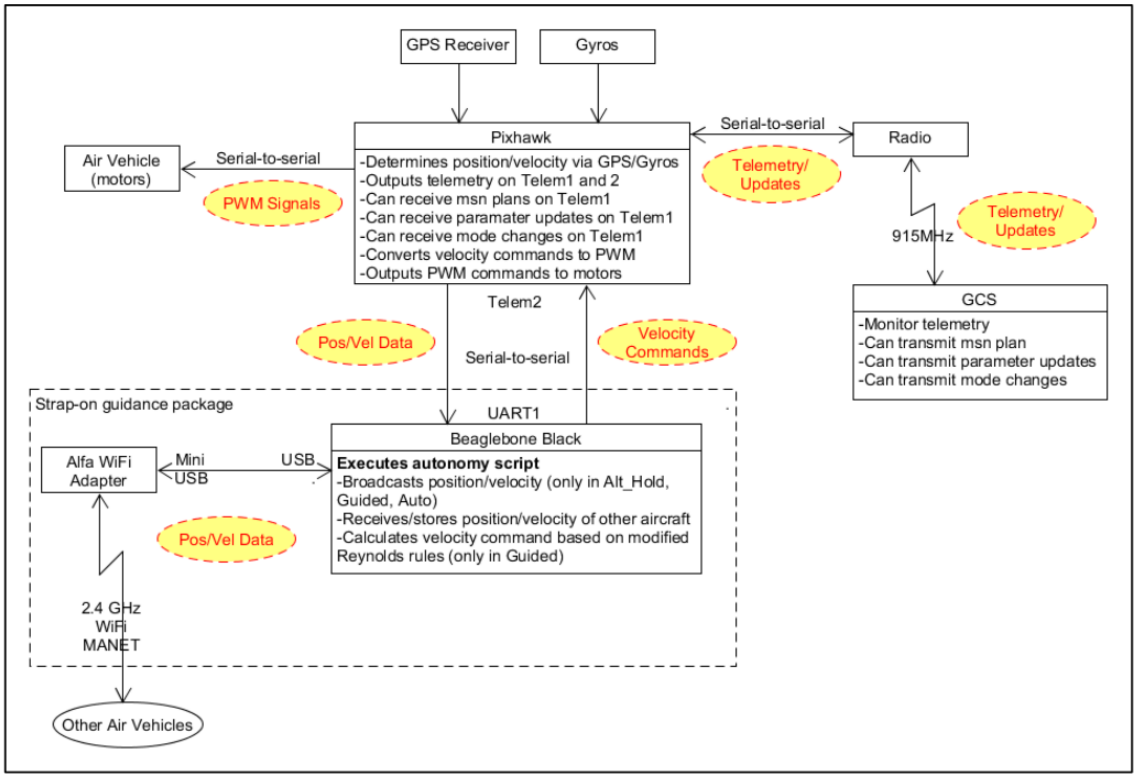


Figure 1. Block diagram of the Original Swarming Architecture found in Allen (2018)

but used a BBB microcomputer with a separate 2.4 GHz Wi-Fi transmitter, focusing most of the integration efforts within the Python control script. A block diagram of the overall architecture from Allen is show in Figure 1. Additional aspects of the control algorithm are discussed throughout the remainder of this thesis as analysis and changes are performed.

2.4 Autonomous Systems Testing

Various methods for testing autonomous systems have been utilized across this field of research, with a large majority using either hardware-in-the-loop (HIL) or SITL to test their algorithms before flying. "Simulation has benefits that include reduced competition for scarce resources, no risk of harm to personnel or equipment, the ability to add as yet undeveloped capabilities to subsystems, and the ability

to perform repeated tests" (Feddema et al., 2004). There are many examples in the literature of researchers using or developing tools, some open source (Odelga et al., 2016; Adiprawita et al., 2007) and others proprietary (Vásárhelyi et al., 2014; Balakirsky et al., 2005), to accomplish early development and testing of autonomous algorithms and control laws before flight testing.

During a rapid UAS development, Adiprawita et al. (2007) utilized HIL to perform 150 hours of comparable real-world testing on a control algorithm. Researchers at the Naval Postgraduate School used HIL to test an autonomous landing system for UAS returning to ships at sea (Kaminer et al., 2004). Hauert et al. (2011) utilized simulation experiments for development and to determine which parameters to test when transitioning to flight testing. The results of each were then qualitatively analyzed to determine the differences experienced in real life. Other work has demonstrated the ability to perform HIL testing with multiple vehicles to evaluate communications in addition to the control algorithms. Using Gazebo, Odelga et al. (2016) were able to perform experiments with three multirotor vehicles, including wireless communication among the simulated vehicles.

Anderson et al. (2008) was able to utilize both SITL and HIL testing through the use of Cloud Cap Technology's Piccolo II. This autopilot supports both types of testing and allows users to account for shortfalls found in each. As found in their research, "the SIL simulator is very useful for low order testing and training..." however, "it will not detect any errors in the hardware, so problems such as software bugs or communications errors will not be revealed with the SIL simulator." Thus, HIL was a valuable testing environment for their integration efforts to supplement SITL testing. In the development of the algorithm currently being studied, Allen utilized Ardupilot SITL testing as it was readily available and had been used previously. This is equivalent to the Pixhawk autopilot software, but lacks the physics models of

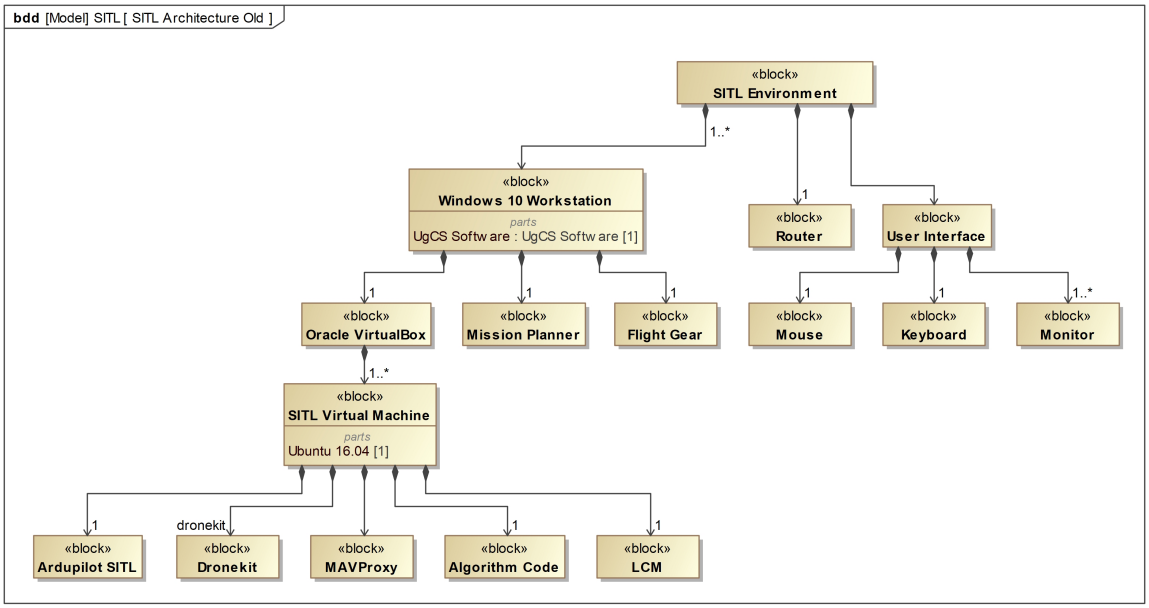


Figure 2. SysML block definition diagram of the Original SITL Architecture

the multirotor aircraft. For visualization purposes he used a Flight Gear multiplayer server to display the members of the swarm in a 3D environment. To control the leader of the swarm, Mission Planner was used as a Ground Control Station (GCS) software to interface with the autopilot. A full block definition diagram (bdd) of the software testing framework is shown in Figure 2 with the various communication protocols presented in the internal block diagram (ibd) of Figure 3.

Test Design.

Challenges of testing systems have always existed and the advent of autonomous systems is no different. In 2012, the Defense Science Board issued a report on the role of autonomy within the DoD, highlighting explicit challenges facing the T&E community. These included, "the ability to evaluate emerging autonomous systems safety, suitability and performance, as well as human interaction with autonomous systems" (Defense Science Board, 2012). Additionally, prediction of decision making and behavior within the operating environment was cited as a necessary part of

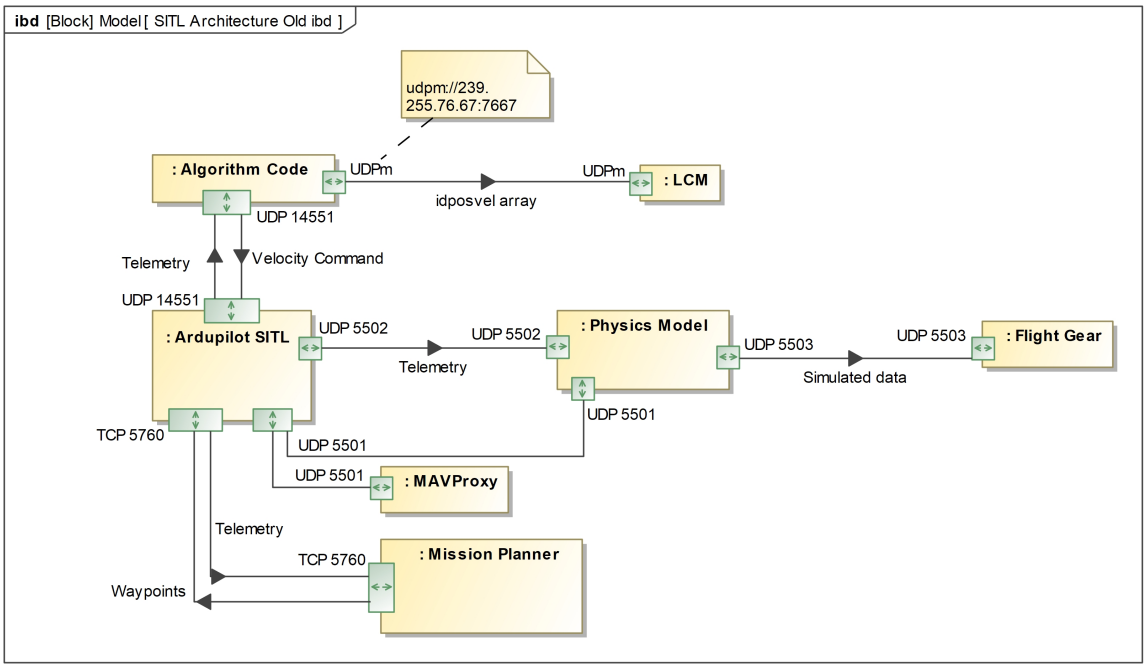


Figure 3. SysML interal block diagram of the Original SITL Communication Protocols

evaluating systems. In 2015, the Autonomy Community of Interest (COI) published their Autonomy Investment Strategy which outlined five goals for T&E of autonomous systems. As previously mentioned, two of these goals were of interest to this research, "evidence based design and implementation" and "cumulative evidence through..." all phases of testing.

Following these reports, the STAT COE held a workshop and published their findings in Ahner and Parson (2016). The group refined the challenges facing autonomous system testing and identified gaps in current testing practices that needed to be addressed. Four of the eight challenges were used to develop the objectives and research questions for this thesis. They were; requirements and measures, designing for test, test adequacy and integration, and testing continuum. Specific gaps from these challenges included quantifying decision making success, a standard framework or architecture, assessing swarm dynamics, and a pseudo test data management system. Addressing these challenges and gaps also reflects a response to a 2017 com-

mittee review of AFIT's SE program in which it was recommended to "accelerate the exploration of autonomous functions..." and develop "airworthiness for swarming Unmanned Aerial Vehicles (UAVs)" (Colombi, 2017).

To better design the testing framework for this research, the 2014 STAT COE report *Best Practices for Highly Effective Test Design; Part 2* was referenced (Cortes, 2014b). Seven best practices are suggested, beginning with a "clear test objective," which highlights the need for all involved parties to understand the who, what, where, when, why, and how's of the test. This should also produce an understanding of how the data will be evaluated and what steps need to be taken in designing collection methods into the tests. Determining the testable space, designing a test structure, and randomizing the test execution are elements from Design of Experiments (DOE) and are necessary to create a statistically relevant experiment. The resulting data sets are then evaluated using statistics and can be expanded depending on the initial test structure, such as a fractional factorial design (Montgomery, 2017). Finally, the testers must inform decision makers with organized results, test processes, designs, and the performance of the system being tested (Cortes, 2014b).

Testing Metrics.

Previous demonstrations with swarming algorithms have shown a variety of testing metrics to use in analyzing the swarming algorithm. Feddema et al. (2004) suggests two main metrics for swarm analysis based on a mathematical framework, "residual error in the global performance index..." and a "communication sample period" to analyze the time to reach a goal state. Multiple examples of global performance index equations are found in section III of Feddema's report, but are usually dependent on the swarm type and purpose. Additionally, "robustness to failure, required on-board computing, and required sensing" are suggested metrics to be investigated.

An early example of testing multiple cooperative unmanned vehicles is found in Gray (2015). During the test and verification phase of this unified command and control architecture thesis, several metrics were used to evaluate formation flocking performance with ground and air vehicles. Distance error and system latency were used for quantitative data analysis and were designed into a well structured test plan. This provided defendable results upon analysis across the various tests performed, with the most relevant being two UAS in a leader-follower configuration with position errors ranging from 3-5 meters (m). In their implementation of a multi-vehicle swarm architecture Vásárhelyi et al. (2014) used velocity correlation to indicate stability of the swarm. They also evaluated distance between swarm members and communication time lag among vehicles. The experimental swarm used an impressive 10 members at fixed altitudes, distances of 6-10 m, a velocity maximum of 4 meters per second (m/s), with a calculated 1.5 second (s) average time lag.

Hauert et al. (2011) performed successful experiments of a fixed wing swarm in a closed communication range and chose to measure turn rates and distance changes between aircraft. Multiple tests were completed in simulation and reality which resulted in a coherent swarming behavior with a communication range of 300 m and maximum turn rates of 0.7 radians per second. Again using fixed wing UAS, Kaiser studied the effects of dynamic weighting of swarming rules in a 2D simulation (Kaiser, 2014). This differed from the aforementioned examples as the focus was on swarm effectiveness and not performance of individual members. A utility function was created which included values such as: time to find a target, percentage of runs when a target was found, attrition rates, average coverage, and near misses among members. The result of this research showed optimizing the boundary, divergence, and wander rules improved the utility function results an average of 10%.

During the single flight test completed by Allen of the original algorithm, the primary focus was placed on individual vehicles to validate the modified Reynolds rules. The first flight test was completed with 3 vehicles; one in position hold, the second under autonomous control, and a third broadcasting position data while stationary on the ground. As with Virágh et al. (2014), position was recorded on each vehicle to determine distances between vehicles, with an average separation just over 8 m. Vehicle velocity and velocity rule commands from the algorithm were also recorded, showing rules triggering when expected and higher priority rules outweighing those with lower priority. In general, the vehicle under autonomous control appeared to behave as expected, but unfortunate testing conditions led to a crash of one of the vehicles and no further data was collected. The metrics to be used in the current evaluation of this modified algorithm will be discussed in Chapter III.

2.5 Summary

Concluding this chapter, the reader should have an understanding of multiple concepts related to this research. The past and present utility of UAS swarms highlighted the importance of continuing research in this area, followed by a fundamental review of the algorithm developed by Allen. Discussion of previous UAS swarm testing, test design recommendations, and testing metrics establish a foundation of knowledge which will be used to develop the methodology of Chapter III.

III. Methodology

3.1 Overview

Chapter III forms the foundation of methods used in this research and discusses the development of the test plans for a UAS swarm algorithm. The main focus was not on the development of the algorithm, but verification through developmental testing utilizing simulation, formulation, and live flight. Outlined first are the testing objectives, metrics, and data requirements. Next, a review of the SITL environment previously used highlights challenges faced and recounts changes made during early phases of familiarization. Following this are several significant modifications resulting from design issues discovered during preliminary SITL verification of the original algorithm. Architecture diagrams throughout the chapter aid the reader in understanding the previous and updated algorithm and testing environments. The chapter concludes with test design and planning for execution of the experiments in SITL and live-flight.

3.2 Test Objectives, Metrics, & Data

The first stage of test design was developing the test objectives, metrics, and required data sets to reflect the research objectives and questions, using best practices from Cortes (2014b). These remain constant and serve as references throughout the next Chapters.

1. Overarching Test Objectives
   (a) Identify deficiencies in the algorithm's design for test and testing environment.
   (b) Verify the performance of the algorithm using simulation, formulation, and live flight testing.
   (c) Observe swarming responsiveness and abnormalities during verification.

(d) Determine the usefulness of quantitative and qualitative assessment of the system.

2. Metrics

   (a) Swarm "goodness," or average deviation from the ideal vehicle-vehicle distance (m) based on vehicle positioning.

   (b) Percent violation of Reynolds+ rule set based on relative vehicle-vehicle positioning.

   (c) Communication time delay of LCM messages.

   (d) Packet loss of LCM messages.

3. Required Data

   (a) Vehicle position in 3D space using the common local tangent plane NED coordinates at the specified update rate.

   (b) Vehicle velocity vectors in the same coordinate frame and update rate.

   (c) Reynolds+ rule outputs for each algorithm loop.

   (d) Redundant logging of all transmitted and received LCM messages across all vehicles with time stamps.

3.3 Preliminary SITL Verification

Qualitative analysis of the existing SITL configuration was performed to meet Objective 1a; determine its adequacy, make any required changes, and establish a baseline for future testing and development. The original testing framework used Flight Gear visualization as explained in Chapter II above (see Figure 2) and Appendix D of Allen (2018), which proved to be difficult to configure from scratch, and provided only a basic visualization of three vehicles in space, limited in geographic location by the software. Additionally, it lacked telemetry data for the simulated vehicles and did not provide a vehicle ID to know each vehicle's unique location. Although this visualization was not required to collect quantitative data for analysis of the algorithm's behavior, it provided immediate and intuitive feedback for qualitative analysis.

Other limitations of the previous testing framework were a result of Mission Planner, the GCS software used for lead vehicle path planning. The most significant limitation of this software related to swarm testing was its inability to support more than a single vehicle connection at one time, both in SITL and real-world testing. Although it boasted a robust parameter list, easy to manage connection interface, and detailed tuning page for vehicles, those features were not required since vehicle tuning was not a focus of this research. However, this is not to say that these parameters would not affect the responsiveness or behavior of individual vehicles.

An additional limitation induced by Mission Planner was the number of required personnel. The current Military Flight Release (MFR) required one safety pilot, one GCS operator, and one observer per air vehicle when operated at WPAFB (Gonzalez, 2014). In total, a three-vehicle swarm required twelve personnel to execute a flight test with these restrictions.

Before the difficulties encountered in configuring the old visualization environment, a new GCS software Universal ground Control Software (UgCS) was explored. An unpublished study was performed regarding operator performance when using this new system, resulting in detailed knowledge of how this system integrated with the Ardupilot SITL environment (Combs, 2018). The primary benefit observed over Mission Planner was the ability to connect to multiple vehicles simultaneously in SITL and flight testing. Additionally, an intuitive 3D world model and easy-to-view telemetry data for any connected vehicle as shown in Figure 4 allowed for improved visualization in the testing framework. Planning a mission for virtual and live flight testing was also straightforward after some familiarization with the software, as observed by members of the study mentioned above and test team members.

The only difficulty encountered with the software was configuring connections to the multiple simulated autopilots. Unlike Mission Planner's connection interface,

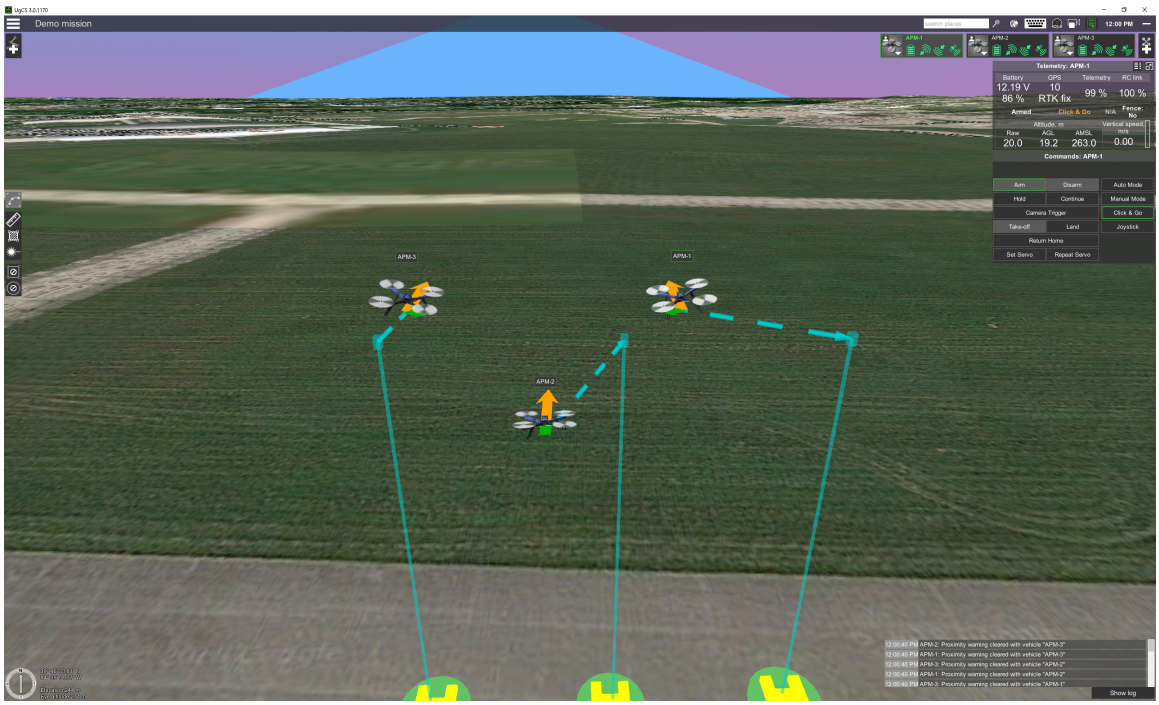


Figure 4. UgCS visualization environment with 3 simulated vehicles and telemetry.

UgCS uses an "Ardupilot VSM" configuration file to look for specific incoming connections (Projects, 2015). After various trials with different connection types (User Datagram Protocol (UDP) & Transmission Control Protocol (TCP)) and specifying proper identification numbers to each simulated autopilot, all three aircraft were connected and displayed in UgCS as seen in Figure 4. Minor drawbacks observed included the inability to alter tuning parameters on the autopilot and configure connections on the main window. Since visualization was the primary concern in SITL testing and tuning of the vehicles was not required during flight testing, UgCS was chosen to replace the original FlightGear multiplayer server and Mission Planner software for visualization and GCS operations.

The final SITL testing environment modeled in Figure 5 was composed of a Windows 10 workstation hardwired to an Ethernet router running several software packages. Algorithm development and analysis in the Python coding language used PyCharm for Windows. UgCS was used for visualization and GCS operations as men-

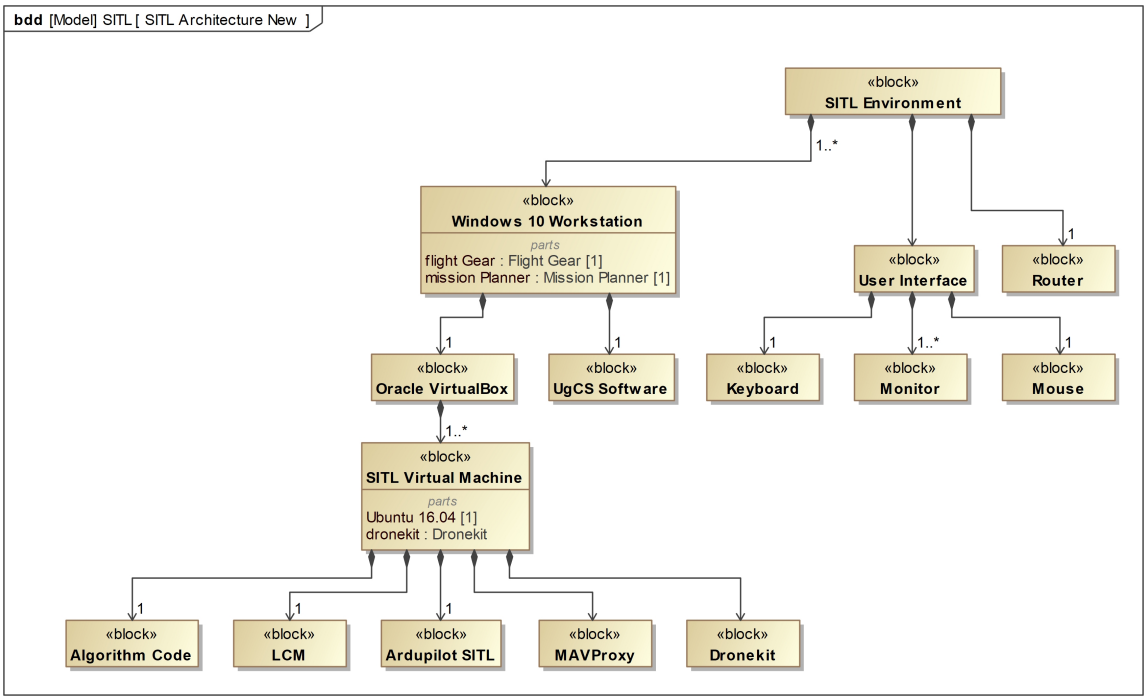


Figure 5. SysML bdd of the New SITL Architecture

tioned previously. Oracle VirtualBox was used to run three Virtual Machines (VM's) with the Ubuntu 16.04 operating system installed. Hosted on each of the VMs was Ardupilot SITL, Java, LCM, and other software packages found in the SITL setup guide in Appendix A. Upon configuring the first VM, clones were made so that they were identical in most aspects, except for individual Ardupilot SITL parameters (i.e. "SYSID_THISMAV").

Executing tests in this environment required an initial configuration of connections between Ardupilot SITL and UgCS, and between the three VMs via LCM. Details of the configuration process are found in Appendix A and modeled in Figure 6 for an individual vehicle. Figure 7 represents the entire SITL framework. Initial verification tests did not focus on data collection, but scripts were in place to enable LCM logger to record all messages broadcast by the algorithm. These messages were then replayed via the LCM logplayer or converted to MATLAB matrices for data analysis using Python (see Appendix B for conversion code). In order to have redundant data and

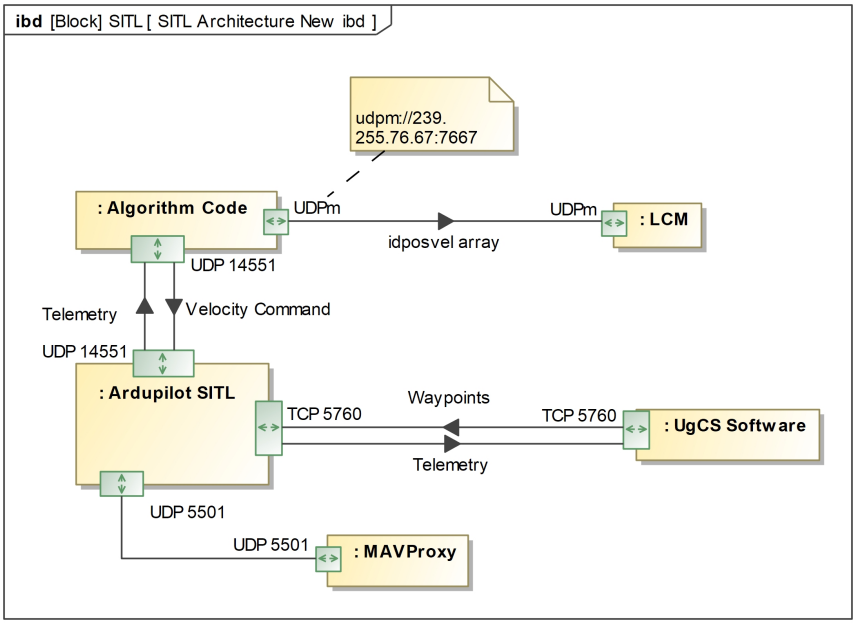


Figure 6. SysML ibd of the New SITL Communication Protocols

observe any network delay in LCM traffic, the script was designed to run on each VM. Appendix A contains the full guide to configuring and running simulations using this new testing framework.

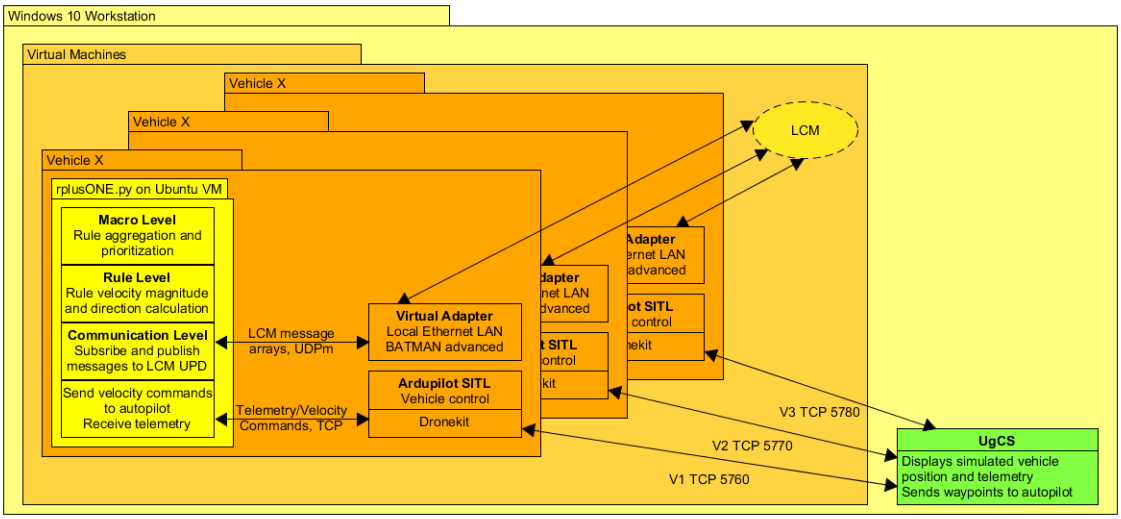


Figure 7. High level view of SITL communication structure

3.4 Algorithm Analysis and Modification

The following describes refinements made to the algorithm to resolve issues that could hamper future testing efforts and address Test Objective 1a and Required Data 3a, b, & c. As mentioned previously in Chapter II, the trial-and-error method of velocity vector equation generation led to two findings in preliminary SITL evaluation. First, multiple code versions for various testing environments and separation distances contained different control laws for Rule 1 (separation) and Rule 2 (flight floor). It was decided to only evaluate the control laws for the "cage" version, as this was assumed to be previously flight tested based on results presented in Allen (2018). The flight floor found in this code version was 4 m and not the 15 m found in other versions intended for open-air flights.

When using this version of the algorithm, abnormal behavior existed when a single vehicle was in "GUIDED" mode. In this mode the vehicle should have maintained altitude at its current position; however, the vehicle descended and became trapped under an altitude of approximately 10 m. Based on telemetry data from UgCS and the inability of this vehicle to gain any altitude in the vertical direction, further evaluation of the control laws and logic was necessary. As shown in Figure 8 the

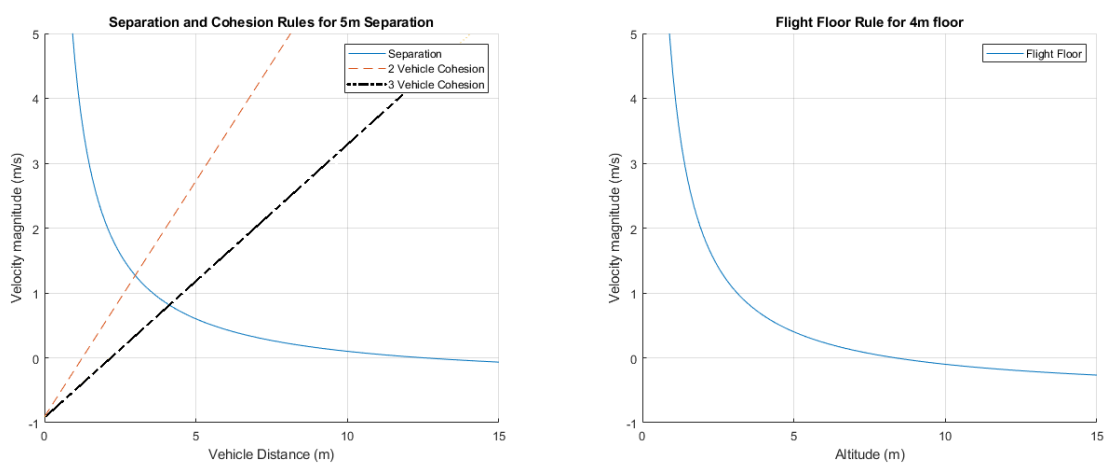


Figure 8. Separation and Flight Floor Rules from Allen (2018)

velocity magnitude for separation and cohesion rules was plotted versus the inter-vehicle distance vector magnitude using MATLAB. The following activation logic for each rule was inferred from the algorithm:

1. The separation rule only activated at distances <5 m between vehicles.
2. The flight floor rule only activated if the magnitude of the separation velocity vector did not fill the maximum bucket velocity of 2 m/s and the current altitude was <4 m above ground level (AGL).
3. The alignment rule activated if the combined magnitude of the above rules was less than the velocity bucket maximum.
4. The cohesion rule activated if the combined magnitude of prior rules was less than the velocity bucket maximum and the swarm had two or more members.

Assuming separation and cohesion were the only rules active, the expected vehicle-vehicle distance in a two-member swarm would have been approximately 3 m separation as seen in Figure 8 since the magnitudes are opposing. In a three-member swarm, the equilibrium state would result in an equilateral triangle with approximate side lengths of 4.1 m. These two states drew concern since the minimum push distance was 5 m. Vehicles could also pass below the minimum floor altitude before the rule took effect. In the worst case scenario of two vehicles vertically stacked, separation rule activation could force the lower vehicle below the flight floor upwards of 2 m/s, filling the bucket and giving just two seconds for the floor rule to execute and force the vehicle upward. Investigation of the "trapped" behavior proved inconclusive, but discovered a damping coefficient of -0.1 applied to the vertical component of the cohesion velocity vector. This appeared to be an effort to keep the swarm from continually gaining altitude. Without it, if a perturbation pushed one vehicle vertically, the swarm could continue vertically indefinitely due to the velocity matching rule. These observed behaviors, logic concerns, and the overall desire to create a robust testing environment resulted in the following modifications to the algorithm.

The first significant change to the code was from an architectural standpoint, which created a hierarchical framework with multiple levels of calculation. This change decoupled the logic for each Reynolds+ Rule calculation and the priority logic for filling the velocity bucket. At the lower level, each rule executed during every loop of the algorithm based on logic specific to the purpose of the rule. The velocity vectors were then aggregated at a macro level using a simplified priority logic, up to the user-defined maximum, and passed to the autopilot as one velocity command. This change simplified the addition or removal of rules such as communication radius.

Another result of this change was a re-prioritization of the rules in the following order; separation, flight floor, cohesion, and alignment. This was necessary to avoid skipping the cohesion rule if the aggregation of separation and velocity matching exceeded the maximum velocity. Since separation and cohesion are fundamentally in opposite directions, their magnitudes may cancel out, allowing for the alignment rule to still be factored into the final velocity vector, or for the swarm to reach a stable state. Another potential, but unexplored integration is the addition of other velocity-based control laws at the lower level, such as urban navigation techniques found in Rufa and Atkins (2016) and Hyslop and Humbert (2010).

New velocity equations developed and evaluated in MATLAB for the Reynolds+ Rule set addressed the potential for undesirable swarm behavior. The focus was to ensure the equations and resulting logic permitted the rules to activate before vehicles violated the separation and floor boundaries desired by the operator. An exponential function served as the foundation for the following equations as the value could be ensured to always be positive in magnitude. Additionally, proper parameter and coefficient placement allowed for manipulation of the equations (Nykamp, 2019). The following equations represent the velocity vectors calculated by the four rules of separation (1), flight floor (2), cohesion (3), and alignment (4):

$$\vec{\boldsymbol{v}}_{1,i} = \begin{cases} 0 \cdot \vec{\boldsymbol{d}}_{1,i} & \text{if } \|\vec{\boldsymbol{v}_{1,i}}\| < \epsilon \\ \left\| \left(v_{max} \times e^{b(-\left\|\sum_{j=1}^{n} \vec{\boldsymbol{p}}_{i,j}\right\| + r_{min})}\right) \cdot \frac{\vec{\boldsymbol{p}}_{i,j}}{\|\vec{\boldsymbol{p}}_{i,j}\|} \right\| \cdot \vec{\boldsymbol{d}}_{1,i} & \text{if } \epsilon < \|\vec{\boldsymbol{v}}_{1,i}\| < v_{max} \\ v_{max} \cdot \vec{\boldsymbol{d}}_{1,i} & \text{if } \|\vec{\boldsymbol{v}}_{1,i}\| > v_{max} \end{cases} \tag{1}$$

where:

$$\begin{aligned} i &= i^{th} \text{ vehicle in the swarm} \\ \vec{\boldsymbol{d}}_{1,i} &= \frac{\vec{\boldsymbol{p}}_{i,j}}{\|\vec{\boldsymbol{p}}_{i,j}\|} \\ \vec{\boldsymbol{p}}_{i,j} &= \text{vector from the } j^{th} \text{ encroaching swarm member on the } i^{th} \text{ vehicle} \\ v_{max} &= \text{maximum bucket velocity (m/s)} \\ b &= \text{response magnitude parameter} \\ r_{min} &= \text{minimum separation radius around the vehicle (m)} \\ \epsilon &= \text{configurable minimum velocity requirement, default} = 0.3 \text{ m/s} \end{aligned}$$

$$\vec{\boldsymbol{v}}_{2,i} = \begin{cases} \vec{\boldsymbol{0}} \cdot \vec{\boldsymbol{d}}_{2,i} & \text{if } \|\vec{\boldsymbol{v}}_{2,i}\| < \epsilon \\ \left(v_{max} \times e^{b(-z_i + f_{min})}\right) \cdot \vec{\boldsymbol{d}}_{2,i} & \text{if } \epsilon < \|\vec{\boldsymbol{v}}_{2,i}\| < v_{max} \\ v_{max} \cdot \vec{\boldsymbol{d}}_{2,i} & \text{if } \|\vec{\boldsymbol{v}}_{2,i}\| > v_{max} \end{cases} \tag{2}$$

where:

$$\begin{aligned} \vec{\boldsymbol{d}}_{2,i} &= \text{unit vector [0 0 -1]} \\ z_i &= \text{altitude AGL (m) of the } i^{th} \text{ vehicle} \\ f_{min} &= \text{minimum altitude (m) as defined by operator} \end{aligned}$$

$$\vec{\boldsymbol{v}}_{3,i} = \begin{cases} \vec{\boldsymbol{0}} \cdot \vec{\boldsymbol{d}}_{3,i} & \text{if } \|\vec{\boldsymbol{v}}_{3,i}\| < \epsilon \\ \left(v_{max} - \left(v_{max} \times e^{b(-\|\vec{\boldsymbol{p}}_{i,cent}\| + r_{min})}\right)\right) \cdot \vec{\boldsymbol{d}}_{3,i} & \text{if } \epsilon < \|\vec{\boldsymbol{v}}_{3,i}\| < v_{max} \\ v_{max} \cdot \vec{\boldsymbol{d}}_{3,i} & \text{if } \|\vec{\boldsymbol{v}}_{3,i}\| > v_{max} \end{cases} \tag{3}$$

where:

$$\vec{\boldsymbol{d}}_{3,i} = \frac{\vec{\boldsymbol{p}}_{i,cent}}{\|\vec{\boldsymbol{p}}_{i,cent}\|}$$

$$\vec{\boldsymbol{p}}_{i,cent} = \text{vector of the } i^{th} \text{ vehicle to the centroid of the swarm, } \hat{C}$$

$$\hat{C} = \left[\frac{\sum_{i=1}^{n} \hat{N}_i}{n}, \frac{\sum_{i=1}^{n} \hat{E}_i}{n}, \frac{\sum_{i=1}^{n} \hat{D}_i}{n}\right]$$

$$\hat{N}_i = \text{North Local Tangent Plane coordinate of the } i^{th} \text{ vehicle of the swarm}$$

$$\hat{E}_i = \text{East Local Tangent Plane coordinate of the } i^{th} \text{ vehicle of the swarm}$$

$$\hat{D}_i = \text{Down Local Tangent Plane coordinate of the } i^{th} \text{ vehicle of the swarm}$$

$$\vec{\boldsymbol{v}}_{4,i} = \begin{cases} 0 \cdot \vec{\boldsymbol{d}}_{4,i} & \text{if } \|\vec{\boldsymbol{v}}_{\boldsymbol{4,i}}\| < \epsilon \\ \|\vec{\boldsymbol{d}}_{4,i}\| \cdot \vec{\boldsymbol{d}}_{4,i} & \text{if } \epsilon < \|\vec{\boldsymbol{v}}_{4,i}\| < v_{max} \\ v_{max} \cdot \vec{\boldsymbol{d}}_{4,i} & \text{if } \|\vec{\boldsymbol{v}}_{1,i}\| > v_{max} \end{cases} \tag{4}$$

where:

$$\vec{\boldsymbol{v}}_j = \text{velocity of the } j^{th} \text{ member of the swarm}$$

$$\vec{\boldsymbol{d}}_{4,i} = \frac{\sum_{j=1}^{n} \vec{\boldsymbol{v}}_j}{n}$$

These rules were aggregated in priority order with a logic check after each addition so as not to exceed v_{max}. In the event the magnitude of the aggregated vectors exceeded v_{max}, the algorithm would not aggregate the remaining rule vectors. Figures 9 through 12 were developed to represent velocity vector magnitudes for rules 1-3 in varying arrangements. These figures account for maximum velocity magnitude, and the resulting curve appears to plateau at this value. The effect of varying r_{min} between 5, 10 & 15 m, while holding v_{max} and b constant is demonstrated in Figure 9.

Due to the placement of r_{min} in the equation, increases or decreases in r_{min} shift the curve right and left, respectively. Additionally, the velocity magnitude is always positive, thereby eliminating any undesired velocity vectors. Figure 10 demonstrates how b can be utilized as a tuning parameter to increase or decrease the response rate

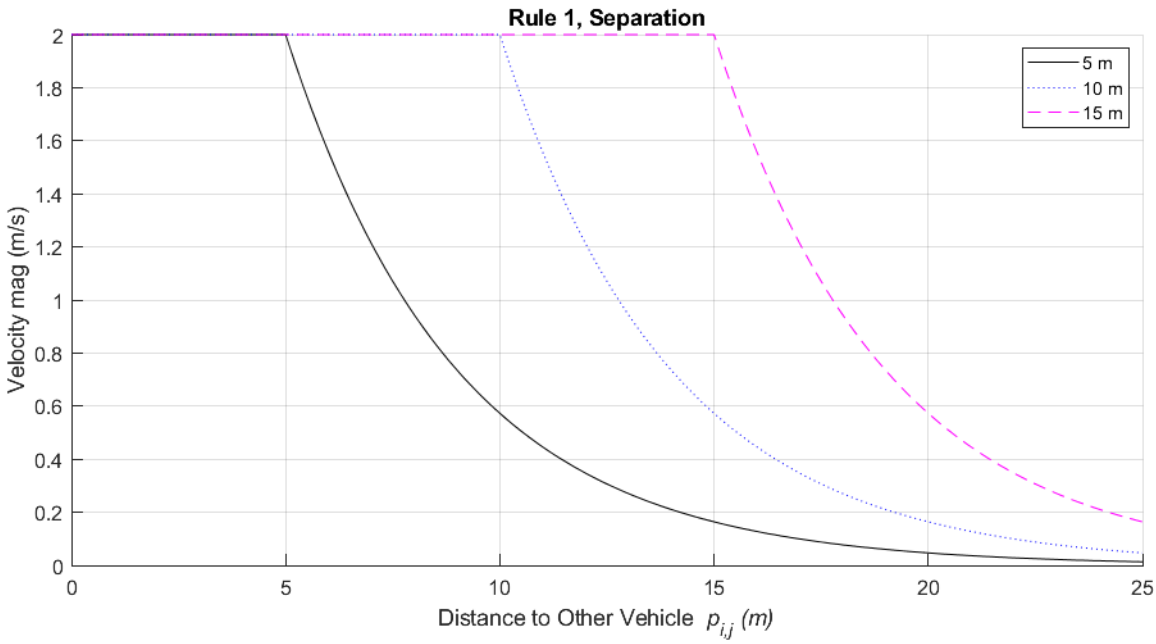


Figure 9. Separation Rule with varying r_{min} values for $b = 0.25$ and $v_{max} = 2m/s$

of the vehicle. As b decreases, the response curve becomes more gradual. The final variation of this rule in Figure 11 shows that an increased or decreased v_{max} alters

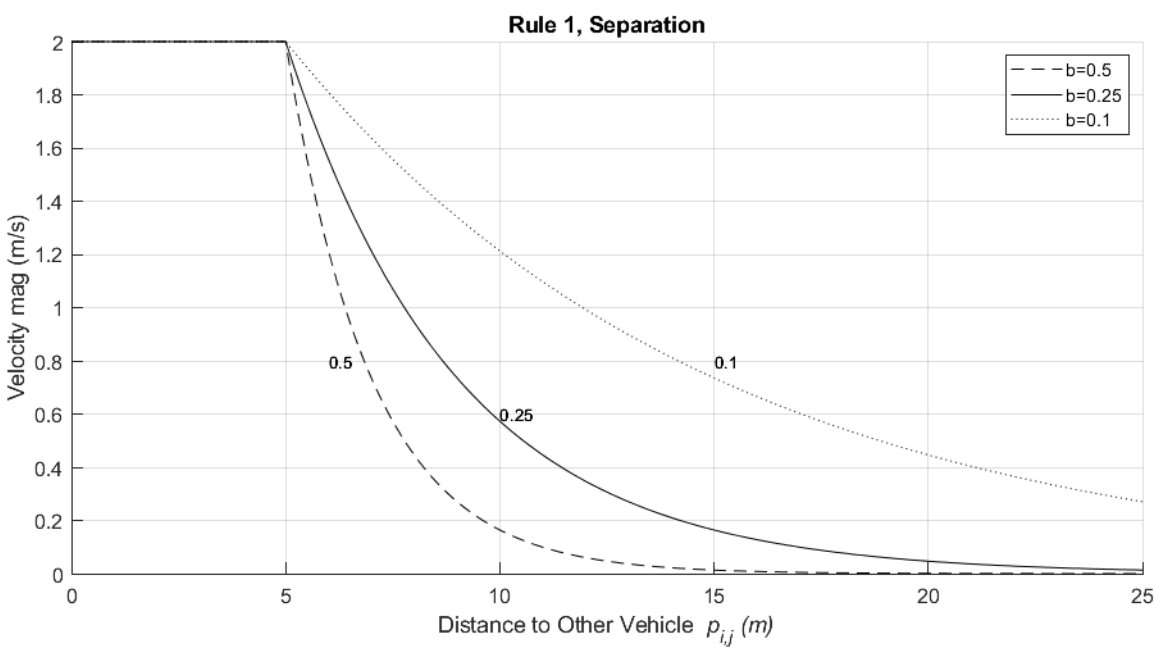


Figure 10. Separation Rule with varying b values for $r_{min} = 5m$ and $v_{max} = 2m/s$

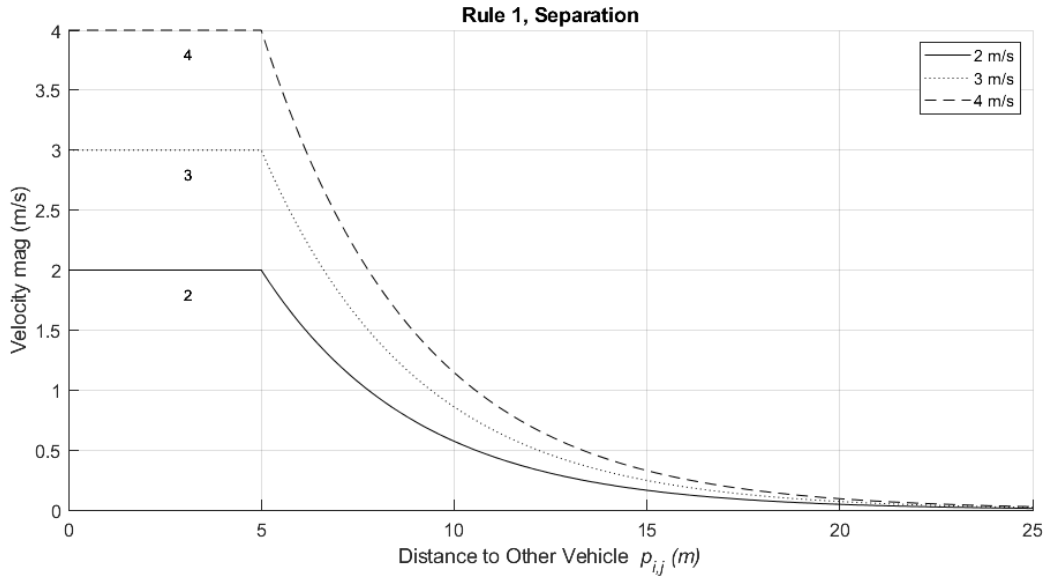


Figure 11. Separation Rule with varying v_{max} values for $r_{min} = 5m$ and $b = 0.25$

the magnitude of the velocity accordingly. The effect of these three parameters was similar in each rule except for cohesion since the magnitude was inverted.

As shown in Figure 12, if the distance was less than the specified minimum for the separation and flight floor curves, the resulting magnitude was always equal to the maximum allowed. If the distance was greater than the specified minimum, the magnitude gradually dropped as distance increased. For cohesion, zero velocity magnitude is allowed within the separation minimum, and the magnitude increases as distance increases. Considering identical r_{min}, v_{max}, and f_{min} values from before, with only separation and cohesion in effect, ideal vehicle-vehicle distance in a two vehicle swarm occurred at 7.77 m with $b = 0.25$. In a three vehicle swarm, the ideal state is as an equilateral triangle with side length 10.61 for the same b. Equation 5 was used to determine the vehicle-vehicle distance in this scenario:

$$v_{max} \times e^{b(-p_{i,j}+r_{min})} = v_{max} - (v_{max} \times e^{b(\frac{-p_{i,cent}+r_{min}}{)})} \tag{5}$$

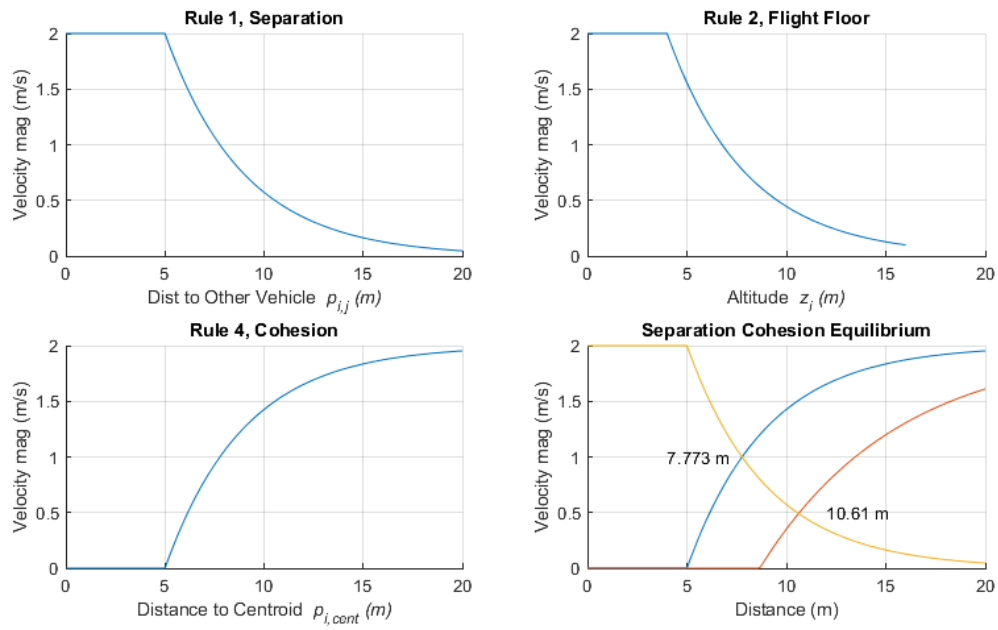


Figure 12. Reynolds+ rule set with $b = 0.25$, $r_{min} = 5m$, $f_{min} = 4m$ and $v_{max} = 2m/s$

The solution was found by substituting $\frac{p_{i,j}}{\sqrt{3}}$ for $p_{i,cent}$ in the right side in addition to the numerical values and solving for $p_{i,j}$. This was done using the mathematical principle that the distance from the vertex to the center of an equilateral triangle of side L is $\frac{L}{\sqrt{3}}$. Figure 13 is a visual representation of the ideal swarm state in 2D, but the plane these points lie upon can be rotated through all three dimensions and remain the same. Another visualization of this is in Figure 14, which shows the ideal vehicle-vehicle distance as a sphere around the center mass, where each sphere intersects the middle of the other two spheres.

A highly beneficial modification made to the algorithm was the introduction of the *'argparse'* function in the Python language (Foundation, 2019). This consolidated code variants into a single version dubbed 'rplusONE,' which was functional in both SITL and hardware configurations. Not only did this reduce version control issues, but it also enabled a smooth transition from lab to flight testing. An additional benefit was the ability to adjust the different parameters within rules through a script file

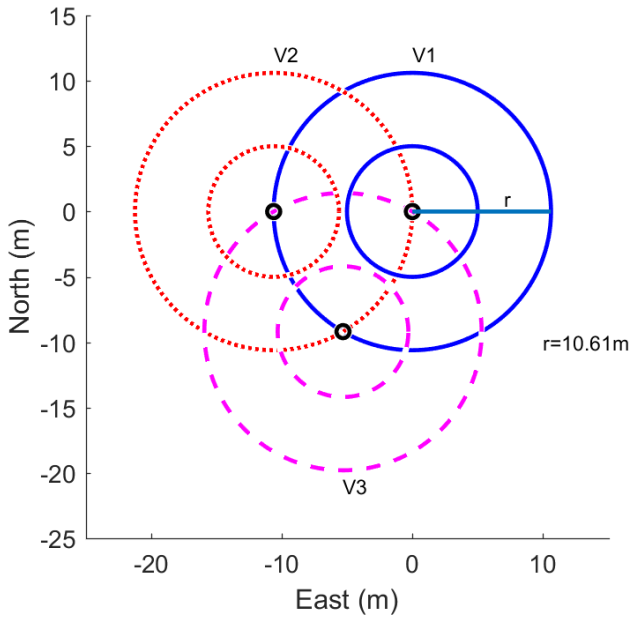


Figure 13. 2D Stable State with $b = 0.25$, $r_{min} = 5m$, $f_{min} = 4m$ and $v_{max} = 2m/s$

during testing without modifying the baseline code before each run. A total of nine arguments, including the different debug options, are shown in Table 1. These were added as passable arguments when initializing the algorithm during testing.

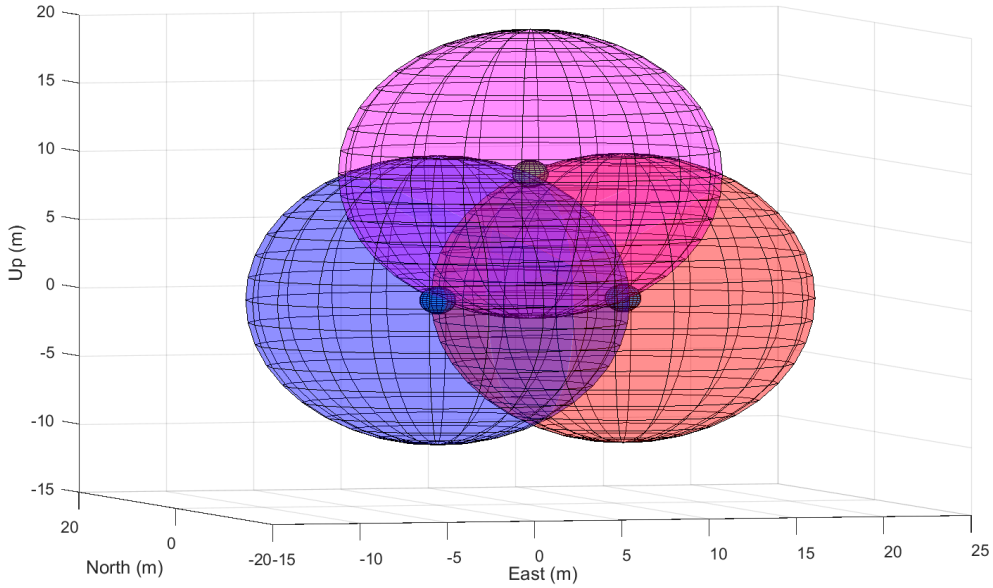


Figure 14. 3D Stable State with $b = 0.25$, $r_{min} = 5m$, $f_{min} = 4m$ and $v_{max} = 2m/s$

Table 1. Table of *'argparse'* parameters in 'rplusONE' code

myID	vehicle ID number for the swarm, i.e. 1, 2, 3...	
-c	connection string, i.e. /dev/tty01, 127.0.0.1:14550	
-s	r_{min} minimum separation radius around the vehicle, default 5 m	
-f	f_{min} minimum altitude defined by operator, default 10 m	
-v	v_{max} or maximum bucket velocity, default 2 m/s	
-u	update frequency for the algorithm in Hz, default 5 Hz	
-b	b response magnitude parameter, default 0.5	
-a	vertical cohesion damping, default -1	
-debug	1	Print Rule 1 magnitude and vector components
	2	Print Rule 2 magnitude and vector components
	3	Print Rule 3 magnitude and vector components
	4	Print Rule 4 magnitude and vector components
	5	Print aggregated Macro level velocity vector
	6	Print LCM message and channel it was received on
	7	Print background and foreground loop with current vehicle position and velocity vectors
	8	Print all of the above debug options on screen
	9	Write all ID, position, velocity, Rule outputs, and Macro level outputs to file

Additional debugging options enabled immediate analysis of individual rules, LCM outputs, and other critical data calculated by the algorithm. A sample of this full debug output is shown in Figure 15. The final debug option added printed all necessary data for analysis to space delimited text files if a backup to LCM logger was required. The passable update frequency parameter was an element of the original algorithm. This parameter must be > 1 Hz as velocity commands expire after a maximum of one second on the Pixhawk. Additional considerations in setting this parameter are the 5 Hz update frequency of the 3DR GPS and potential network bandwidth restrictions.

Slight modifications made to existing inter-vehicle communication methods resulting from the above changes did not alter the fundamental method used by Allen. The LCM system was still used to broadcast an array of ID, position, and velocity data, as well as an array of the velocity commands sent to the autopilot. The first array is below.

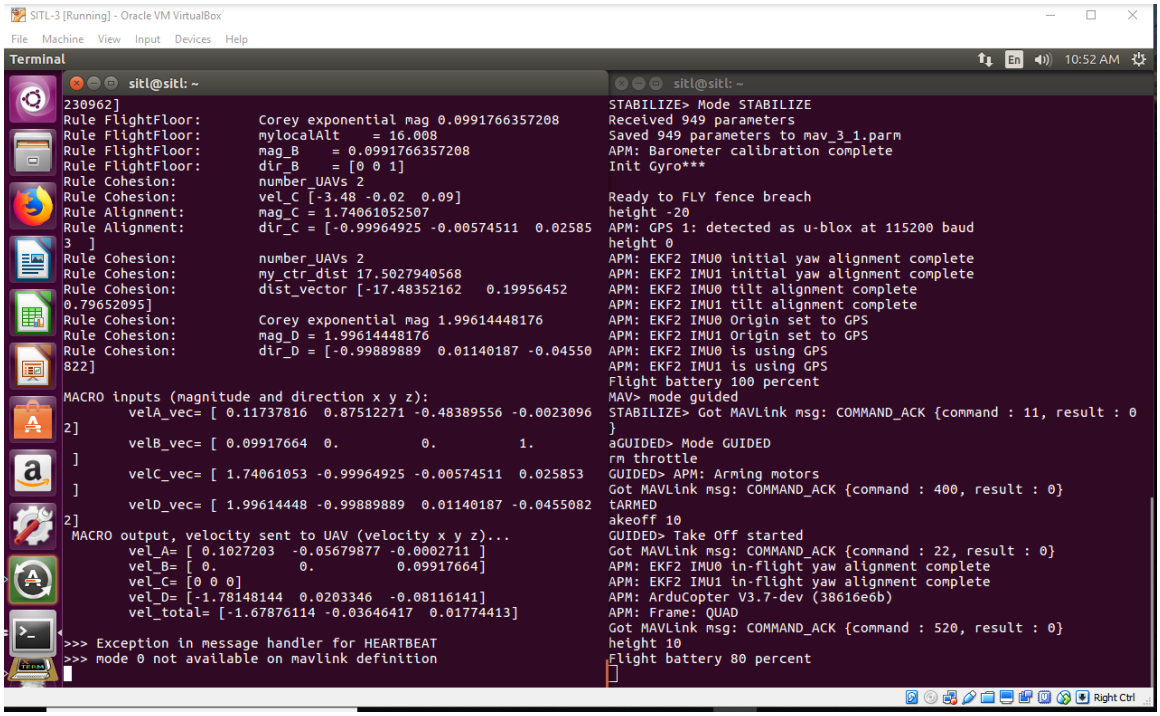


Figure 15. Ubuntu SITL interface showing a snapshot of the 'rplusONE' algorithm with full debug outputs and the Ardupilot simulation on the right.

$$\begin{bmatrix} ID_i & \hat{p_N} & \hat{p_E} & \hat{p_D} & \hat{v_N} & \hat{v_E} & \hat{v_D} \\ ID_{i+1} & \hat{p_N} & \hat{p_E} & \hat{p_D} & \hat{v_N} & \hat{v_E} & \hat{v_D} \\ ID_{i+2} & \hat{p_N} & \hat{p_E} & \hat{p_D} & \hat{v_N} & \hat{v_E} & \hat{v_D} \end{bmatrix}$$

where:

$$i = i^{th} \text{ vehicle in the swarm}$$
$$\hat{p} = \text{position in NED coordinate frame}$$
$$\hat{v} = \text{velocity in NED coordinate frame}$$

Differences between the networking protocol structure of SITL and real-world communications did exist; however, the flow of information was nearly identical and is illustrated in Figure 16. As mentioned, the algorithm calculated in an iterative loop at a frequency determined by the operator. Velocity magnitude and direction for each rule were calculated in the lower-level and passed to the macro level where the values were prioritized and bucketed. After this was complete, the data was passed to the

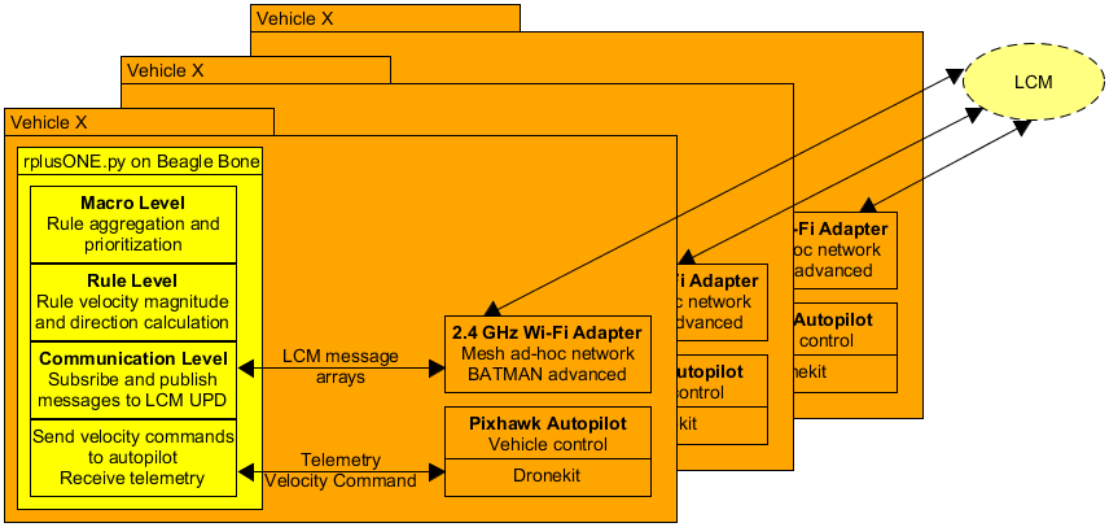


Figure 16. Illustration of data flow among vehicles in the swarm.

vehicle's autopilot and to LCM for other vehicles to retrieve over the mesh ad-hoc network. New data was retrieved from the UDP multicast for the other vehicles in the swarm, and the loop continued.

3.5 Flight Test Methodology

Vehicle Configuration for Flight Tests.

Before exploring the test cases, it is important to understand the testing environment and configuration of the X-8 multirotor vehicles. For flight testing, the AFIT UAS team was able to secure two days of range time at Camp Atterbury Army Base in Edinburgh, Indiana. Figure 17 shows the dedicated UAS runway and adjacent field to the immediate east used for testing. The benefit of testing at Camp Atterbury, was the permissible first-time use of UgCS as a GCS for flight testing multiple vehicles, thus reducing personnel and equipment. As required by the MFR, a GCS operator was assigned to monitor vehicle status and upload mission plans. Additionally, each vehicle had an assigned safety pilot and observer pair for all tests. These

Figure 17. UAS runway at Camp Atterbury. North is up.

were the most critical personnel during testing in the event of unexpected behavior, and they had full authority to recover their vehicle at any time. Before flight testing a Crew Resource Management (CRM) meeting was held to discuss dedicated roles, communication strategies, expected vehicle behavior, and a high-level overview of test cases.

Each safety pilot used a FrSky Taranis X9D 2.4 GHz radio paired with a transceiver for manual control of their vehicle and setting various Pixhawk flight modes during tests. These modes included 'STABILIZE,' 'ALT_HOLD,' 'GUIDED,' and 'AUTO.' In 'STABILIZE' the autopilot stabilized the vehicle, but the safety pilot controlled roll, pitch, yaw, and throttle inputs. The 'ALT_HOLD' mode stabilized and maintained the current altitude of the vehicle while allowing roll, pitch, and yaw control by the pilot. In 'GUIDED' mode, the GCS operator controlled the vehicle via click-and-go waypoints sent to the vehicle. When no waypoints were loaded to the vehicle in this mode, the autopilot maintained the current position and altitude while allowing the pilot to control yaw. Lastly, in 'AUTO' the vehicle navigated all loaded waypoints autonomously in 3D space, keeping the vehicle oriented toward the next waypoint. The selection of these modes was accomplished via a three position switch

on the Taranis depending on the test case. For safety purposes, the first position was always 'STABILIZE,' allowing for manual control during normal launch and recovery, or emergencies.

Each X-8 used a 4-cell Lithium Polymer (LiPo) flight battery of 10,000 milliamp hour (mAh) capacity. Affixed to the top platform was a "guidance package" consisting of the BBB companion computer connected via Universal Serial Port (USB) to an Alfa 2.4 GHz Wi-Fi adapter, and a dedicated Lithium-Polymer (LiPo) battery with a voltage regulator (Allen, 2018). The BBB performed all algorithm calculations, communication on the ad-hoc mesh network, and LCM logging. With the previous changes to the 'rplusONE' algorithm, it was possible to use the same code from SITL testing by passing a new connection string argument. Exchanges of telemetry and velocity commands occurred between the BBB UART1 port and Pixhawk Telem2 port via a serial cable. Algorithm control of the autopilot in each of the autopilot modes is summarized in Table 2.

As was done in Allen (2018), an additional Beagle Bone was used as a ground node on the mesh network to communicate with the airborne nodes. Upon start-up, the BBBs were configured to automatically establish the ad-hoc mesh network and pull the local time from the Pixhawk. This eliminated the need for hard-wired connections to each BBB before each mission and allowed access to each node via secure shell (ssh). This connection served two purposes, with the first being file transfer of updated algorithm code. The second was to execute a shell script to

Table 2. Algorithm behavior by mode adapted from Allen (2018).

Mode	Vehicle	Data Broadcast	Data Received	Autonomous Control
Stabilize	All	No	No	No
Alt_Hold	All	Yes	Yes	No
Guided	All	Yes	Yes	Yes
Auto	All	Yes	Yes	No*
*In terms of autopilot control by the algorithm				

LCM Spy

Channel	Type	Num Msgs	Hz	1/Hz	Jitter	Bandwidth	Undecodable
LCM_SELF_TEST	?? 6c636d2073656c66	6	0.00	Infinity ms	-10.00 ms	0.00 KB/s	6
v1vel	sendvelcomb	34	0.00	Infinity ms	-10.00 ms	0.00 KB/s	0
v2vel	sendvelcomb	1841	6.99	143.00 ms	230.90 ms	1.20 KB/s	0
v3vel	sendvelcomb	1854	5.00	200.19 ms	272.63 ms	0.86 KB/s	0
vehicle1	idposvel	2505	4.00	250.24 ms	601.85 ms	0.25 KB/s	0
vehicle2	idposvel	2534	5.99	166.83 ms	458.88 ms	0.37 KB/s	0
vehicle3	idposvel	2597	5.99	166.83 ms	404.81 ms	0.37 KB/s	0

Figure 18. LCM spy interface used on node 4 to monitor traffic.

initialize LCM-logger and the rplusONE.py file via "screen." This script passed the required $myID$ argument and any of the other variables to be altered for each test.

A new LCM log was created per vehicle by this script with the date and time as the file name, thus satisfying Required Data 3d from the Test Objectives, Metrics, & Data section. The use of "screen" was required to run the algorithm and logger in the background since ssh was terminated to reduce bandwidth on the network as discussed in Allen (2018) and verified via a ground test of the system. After terminating these ssh connections, this node was configured to serve as an LCM traffic monitoring station using LCM-spy as shown in Figures 18 & 19. After each test, ssh was rejoined to terminate the detached screens.

vehicle1

- idposvel		
long	id	1
- double[]	**position[3]**	
double	position[0]	10.369854436382468
double	position[1]	13.80192730373546
double	position[2]	-234.289976637028
- double[]	**velocity[3]**	
double	velocity[0]	1.22
double	velocity[1]	0.69
double	velocity[2]	-0.1
long	*LCM_FINGERPRINT*	*-4445139557512388738*
long	*LCM_FINGERPRINT_BASE*	*7000802258098581439*

Figure 19. LCM spy individual channel live view with graphs.

Test Cases and Documentation.

Every effort was made to align the test cases with the test objectives, metrics, and required data previously outlined while answering the primary research questions. The resulting test plan focused on swarm state exploration, algorithm verification, risk analysis, and data collection during live flight testing. Preliminary trials and validation flights in SITL provided qualitative insight into the expected swarm behavior and data collection process, as well as quantitative data for validation. The following dynamic swarm states were identified for exploration and set as test cases:

1. Single vehicle obeying the flight floor rule
2. Single vehicle following ground-based swarm member
3. Two and three vehicles swarming via "leader-follower" flying with the leader under manual control
4. Two and three vehicles swarming via "leader-follower" flying with the leader under autopilot control, following predetermined waypoints
5. Two and three vehicles swarming with all members under autonomous swarming control to observe near-stable state behavior

Each of these cases was organized into missions of varying levels of difficulty, progressing logically to more complicated testing states as the test team became comfortable with the swarm's behavior. The initial mission plan outlined in Table 3 shows variations in separation distance (r_{min}), maximum velocity (v_{max}), response magnitude (b), and broadcast frequency (u). Early tests used a conservative $r_{min} = 10$ m while the rules were being evaluated to avoid collisions. Later tests reduced this value to $r_{min} = 5$ m allowing for tighter swarming while taking into account relative GPS error of $\sim$ 2-3 m at an update rate of 5 Hz. Maximum velocity varied between 2-3 m/s, with $b = 0.5$ or $b = 0.25$. These were selected based on prior SITL experiments demonstrating conservative swarm behavior. Lastly, update rates of $u = 5$ & 10 Hz

Table 3. Summary of planned missions for flight testing. The value of $f_{min} = 10$ for all flights.

Vehicle			r_{min} (m)	v_{max} (m/s)	b	u (Hz)
1	2	3				
GUIDED	N/A	N/A	10	2	0.5	5
GUIDED	GUIDED*	N/A	10	2	0.5	5
GUIDED	GUIDED	N/A	10	3	0.5	5
ALT_HOLD	GUIDED	N/A	10	2	0.5	5
ALT_HOLD	GUIDED	GUIDED	10	2	0.5	5
AUTO	GUIDED	GUIDED	10	2	0.5	5
			10	2	0.5	10
			10	2	0.25	5
			10	2	0.25	10
			10	3	0.5	5
			10	3	0.5	10
			10	3	0.25	5
			10	3	0.25	10
			5	2	0.5	5
			5	2	0.5	10
			5	2	0.25	5
			5	2	0.25	10
			5	3	0.5	5
			5	3	0.5	10
			5	3	0.25	5
			5	3	0.25	10
GUIDED	GUIDED	GUIDED	10	2	0.5	5
			10	2	0.5	10
			10	3	0.25	5
			10	3	0.25	10
			5	2	0.5	5
			5	2	0.5	10
			5	3	0.25	5
			5	3	0.25	10
*On the ground						

were used to ensure minimum algorithm loops matched GPS update rates and to study the performance of the BBB at higher rates.

The 'AUTO,' 'GUIDED,' 'GUIDED' portion of the test plan incorporated DOE and mimicked a 2^4 factorial design. This did not use randomization due to time

constraints during testing. Some deviations from the mission plan did occur based on swarm performance and input from the test team. These included inferring some behaviors such as when all vehicles were in 'GUIDED' from certain points along the flight plan. Future tests in SITL or live-flight could incorporate these same concepts if more range time is available for test execution.

During all missions f_{min} was set to 10 m as a result of discussions with the safety pilots on a manageable recovery altitude. Another concern of the safety pilots was vertical stacking of the vehicles, with prop wash potentially affecting the lower vehicle. The resulting change was the reintroduction of the vertical damping coefficient (a) on the cohesion rule from the original algorithm. This value was set to $a = -1$ for all flights. SITL verification of this value indicated the vehicles would be limited to a horizontal plane at 10 ± 2 m AGL, forcing the majority of vehicle movement in the north and east components of the velocity vector. Vasarhelyi used a similar concept; however, their algorithm only calculated 2D vectors, and each of the vehicles maintained a predetermined altitude.

Each mission varied slightly but generally followed the same test procedures. Every vehicle was given a new flight battery before each flight, and the payload battery voltage was checked and replaced as needed. The vehicles were then placed in the desired take-off location and powered on allowing the autopilot to obtain GPS lock and a telemetry connection with the GCS. Next, the payload was powered, and upon connection to the mesh network, ssh connections established. Upon connecting, the parameters in the initialization script were altered, and the script was executed to start the algorithm and LCM-logger. Once all vehicles were started the safety pilots took off in 'STABILIZE' one at a time, hovering above the test area. Upon direction from the test director, all pilots switched their vehicle into the assigned mode, and the test began. After the test was complete, pilots recovered their vehicles. Upon

recovery, ssh connections were re-established to kill the background screens, check for data logs, and configure the vehicles for the next test.

For missions where Vehicle 1 was in 'ALT_HOLD,' the lead safety pilot controlled the vehicle and attempted to initiate responses from the follower vehicle(s) by flying toward and away from them, or holding in position. In cases where all vehicles were in 'GUIDED,' occasional switches to and from 'ALT_HOLD' or 'STABILIZE' were be made to re-position the vehicles before observing their tendency to equalize. When Vehicle 1 was in 'AUTO' it followed the waypoint plan shown in Figure 20 at an altitude of 10 m to align it with the follower vehicles' horizontal restriction. Particular areas of interest in this flight plan were the ends of the out-and-back legs where Vehicle 1 flew directly at or between the follower vehicle(s) and the end point where they reached a stable state. The waypoint navigation speed for Vehicle 1 was equal to v_{max} in these cases to avoid out-flying the follower(s).

Although the primary purpose of flight testing was data collection for quantitative analysis, observations from the experienced test team provided supplemental qualitative analysis. Before testing, all members of the team were instructed to call attention

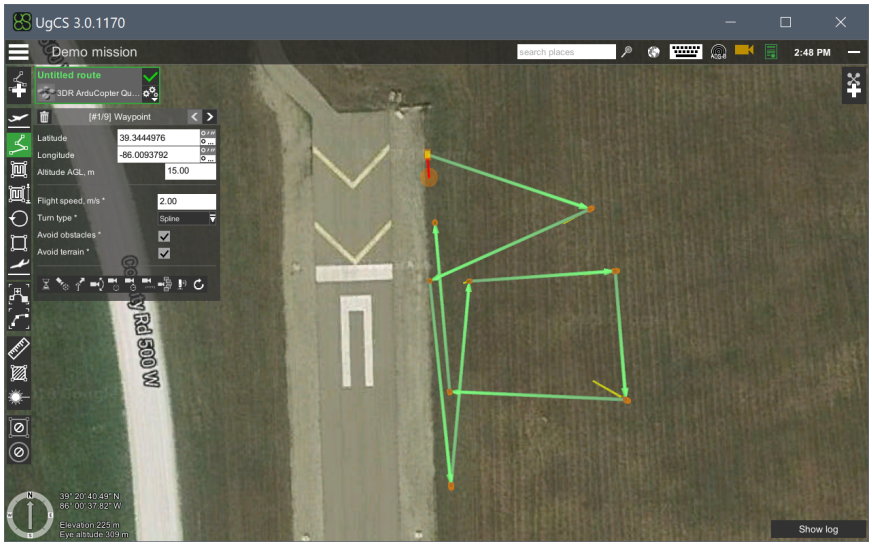


Figure 20. Screenshot of the planned flight path for the lead vehicle.

to unsafe behaviors and make clear their role in data collection. All inputs from the safety pilots, observers, and GCS operator were recorded during testing to aid in this process. Lastly, the expertise of several members on the team was leveraged to troubleshoot mechanical and communication issues.

The test plan was documented per the Operating Instructions outlined for UAS research by AFIT in Badiru (2014). The instructions mandated all tests be submitted and approved by a Technical Review Board (TRB) and a Safety Review Board (SRB). The complete package found in Appendix C was prepared using the template provided and was approved by the TRB/SRB before flight testing. Upon completion of the testing program, a completed After Action Report (AAR) was submitted assessing test execution and providing recommendations for future testing. The full AAR is in Appendix D.

Several final preparations were made before the scheduled flight testing window to ensure smooth test execution. First, each vehicle was paired with a unique safety pilot controller for testing. Afterward, each vehicle was flown individually in the AFIT Flight Test Facility shown in Figure 21 to ensure it was flight worthy. During these flights, proper connection configurations in UgCS were verified on the GCS. Additional vehicle configuration checks were performed in the lab before flight testing to ensure proper fail-safes were in place.

The final action was a ground test with all three vehicles in the ATF, configured as they would be during flight testing. One vehicle was picked up and moved around the ATF while the outputs from the algorithm on the other vehicles were monitored via the ground node. During this test, the communication bandwidth issue of multiple ssh connections was verified, and so only a single vehicle was monitored at a time. Additionally, all three vehicles were connected to the GCS simultaneously via their 3D Robotics (3DR) 915 MHz transceiver pairs. Some vehicle position drift, as well

Figure 21. AFIT Flight Test Facility.

as occasional disconnects from the GCS by Vehicle 3, were observed via this connection. Mesh network communications, LCM messages, and the algorithm performed as expected, and the system was deemed ready for flight tests.

3.6 Summary

In summary, this chapter outlined the development of the testing methodology beginning with the objectives, metrics, and data required to complete flight testing. A thorough analysis of the previous SITL configuration was conducted, and a revised framework was put in place to simplify testing and provide a foundation for future research. This framework was then used to analyze the existing algorithm. Multiple architecture, logic, and rule changes were made to facilitate testing and data collec-

tion, resulting in a stable algorithm version dubbed 'rplusONE' that was used in both SITL and flight testing. Some preliminary analysis was performed on the revised algorithm rule set through the use of MATLAB, providing insight into the expected stable state. A brief overview of the entire system aimed to provide a fundamental understanding of how the entire system interacts; ranging from vehicle hardware configurations and the test environment, to mesh networking and data collection. Finally, a cumulative flight test plan outlined swarm states, DOE inspired mission plans, general testing procedures, and the preliminary test preparations.

IV. Analysis and Results

4.1 Overview

A variety of data and findings are discussed in Chapter IV, beginning with a summary of SITL test results. Following this, observations, revisions, and qualitative analysis of flight testing are discussed. As previously mentioned, a crucial part of conducting flight tests was the collection of qualitative data not just on swarm performance, but also the algorithm, communication, and data collection methods used. Throughout the two days of flight testing, many lessons were learned concerning all of the items mentioned above. This information is presented to the reader to give insight into the quantitative data presented. Following these discussions, a basic overview of the analysis procedure is given, including processing logic and references to the analysis code. Quantitative results from flight testing are then presented using the previously identified metrics.

4.2 SITL Test Results

The primary use of SITL during this research was testing and verifying algorithm performance as modifications were introduced. Following standard software testing procedures, iterative tests were performed on a daily to weekly basis. Due to these rapid iterations, it was superfluous to log every simulation for quantitative analysis. Instead, the visualization within UgCS proved to be a sufficient resource while performing a qualitative analysis of the swarm, ensuring the algorithm was still performing as expected. This extensive familiarization with the algorithm was critical to the testing process, as it gave the tester an intimate knowledge of the algorithm and the expected swarm behavior.

Swarm Behavior and Flight Envelope Exploration.

Many observations made during SITL tests prompted algorithm modifications as previously discussed. However, there is more information to be gained, to include swarming performance not observed during flight testing. One such example was the effect of removing the third vehicle from the swarm when the three vehicles were in equilibrium and 'GUIDED.' In this case, the remaining two vehicles would receive a minor velocity vector impulse and, if near the two-vehicle equilibrium, would continue velocity matching indefinitely in the vector direction. This drift could be interrupted by reintroducing the third vehicle, as the other two would reverse direction and attempt to swarm. However, if the vehicle were removed again, the behavior would resume. To counter this behavior, implementing a communication radius rule could either direct the vehicles off of a barrier toward the center or on a reflective angle as demonstrated in Kaiser (2014). An alternative method is central point gravitation that would always induce a minor velocity toward the home location.

Further experimentation in SITL helped determine which parameters to alter during flight testing and the corresponding values that demonstrated the most influence on swarm behavior. These simulations highlighted potential boundary and endpoint conditions avoided in the design of the test missions for safety purposes. The most significant safety concern was an extremely oscillatory behavior among two or three vehicles when $u = 2$ Hz. While this is still above the minimum rate suggested by Allen of 1 Hz, velocity commands lasting 0.5 s coupled with vehicle momentum caused concern for vehicles nearby. As a result, the minimum for flight testing was $u = 5$ Hz to match the update rate of GPS and ensure safe operation of the swarm.

Parameter Testing.

The second condition observed in SITL before flight testing was the effect of the damping parameter on the vertical component of the cohesion vector. When set to a value of $a = 1$, there was no restriction on vertical movement above the flight floor. This resulted in a less stable swarm with continual 3D movement when all vehicles were in 'GUIDED' similar to the sphere visualization in Figure 14. When the lead vehicle was set to 'AUTO' to fly a waypoint plan with altitude changes above f_{min}, this extra movement was limited but resulted in some vertical stacking of vehicles. If this waypoint plan was limited to an altitude near f_{min}, the followers' vertical movement was minimal. Setting $a = 0.1$ resulted in a more stable swarm in all of these conditions as it scaled the vertical velocity component accordingly. In cases where $0 \leq a < 1$ and all vehicles were in 'GUIDED,' any upward impulse would cause the entire swarm to continue along that vector direction due to velocity matching and no damping.

Conversely, when $0 > a \geq -1$ all of the vehicles were limited in vertical movement to a region above f_{min}. The exact altitude and size of this region was governed by f_{min}, r_{min}, b, a, and v_{max}. This is due to the presence of these variables in the vector calculation for the flight floor and cohesion rules, which are now opposing one another in the vertical components. When all vehicles were in 'GUIDED,' the behavior of the swarm was noticeably more stable than the previous scenarios since the vehicles were mostly limited to 2D movement in the N&E directions. If the lead vehicle was flying waypoints with altitude changes, the follower vehicles were restricted to this region just above f_{min}. Generally, setting $a = -1$ kept the vehicles closer to f_{min} than $a = -0.1$, while any value $0 > a$ eliminated the upward impulse behavior. Ultimately, this analysis guided the decision to implement $a = -1$ during flight tests to mitigate the vertical vehicle stacking concern of the safety pilots.

Testing the implementation of the new swarm responsiveness parameter b, revealed a reasonable upper limit of 0.5 for flight testing, as vehicle responses at higher values appeared extremely impulsive in SITL. On the contrary, $b = 0.1$ appeared to give slow vehicle response due to the more gradual curve as shown in Figure 10. Although not an exact median value, $b = 0.25$ was tested as a midpoint value and provided a more desirable result in the appearance of vehicle response. Full characterization of the range of b values was not desired in this test plan, but the selected values provided insight into the influence it had on vehicle responsiveness. Optimal b values most likely exist across the full spectrum of other parameters which would provide the most stable vehicle response.

Anomalous Behavior.

A final observation during SITL testing was the occurrence of unresponsive vehicles in 'GUIDED.' In this case, the algorithm was broadcasting velocity vectors which were monitored via the debug options and appeared correct, but the autopilot would hold the vehicle in place. An exact cause of this behavior was not determined as a simple reboot of the autopilot and reconnect by the algorithm would resume normal swarming control. This did not present as a persistent issue and appeared to happen randomly.

It should be noted that both Vehicle 2 & 3 would display the message "Exception in message handler for HEARTBEAT mode 0 not available on mavlink definition" in the algorithm window. This only occurred when the simulated vehicle was connected to UgCS and the algorithm simultaneously. When not connected to UgCS this error message was not present, but adequate tests were not performed to check if the algorithm performed as desired or if the vehicle became 'stuck.' The exact cause of this phenomenon remains undiscovered, but potential causes are given in Chapter V.

4.3 Flight Test Overview

Overall, the two days of flight testing were conducted successfully with few issues aside from time constraints, brisk temperatures, and windy conditions. Preparing for flight testing through the TRB/SRB and CRM briefings, SITL tests, and ground tests, in conjunction with an experienced test team, led to a total of 51 sorties and 13 three-vehicle swarm logs. Throughout these sorties, individual vehicle actions and overall swarm behavior represented a successful implementation of the Reynolds+ rule set. Testing completed with no vehicle crashes while under algorithm control.

Test Day One Summary.

Temperatures on the first day were in the mid to high 50's, with winds approximately 15 miles per hour (mph). Previously outlined missions were executed up through a three-vehicle swarm with the leader under manual control in 'ALT_HOLD.' The first test of the flight floor rule was successful, with the vehicle proceeding to gain altitude and hold position above f_{min} after being placed in autonomous mode at ≈ 5 m. A second test with the airborne vehicle following a ground-based vehicle also saw the vehicle obey the flight floor rule and successfully respond to the position changes of the ground vehicle. In all subsequent flight tests, this rule always appeared to be fully functional at $f_{min} = 10$ m.

Next, two vehicles were placed in 'GUIDED' while airborne to demonstrate swarming behavior before progressing to three vehicles. During this test, the vehicles appeared to respond properly, and so the lead vehicle was placed in 'ALT_HOLD' to demonstrate the lead vehicle 'pulling' or 'pushing' the follower vehicle. Upon this successful demonstration, a third vehicle was added to the swarm and flown similarly. The resulting flight tracks for the vehicles are shown as 2D vectors in Figure 22.

At the beginning of the three-vehicle test, each was in 'GUIDED' mode with $v_{max} = 2$ m/s and $r_{min} = 5$ m. From the perspective of the team, the vehicles appeared to attempt equalization; however, the westerly wind created oscillatory behavior and prevented an ideal stable state from being reached. This behavior is represented by the strong concentration of plot lines in the 10-30 m East, 20-30 m North region of Figure 22. After 50 seconds of 'GUIDED' the leader was placed

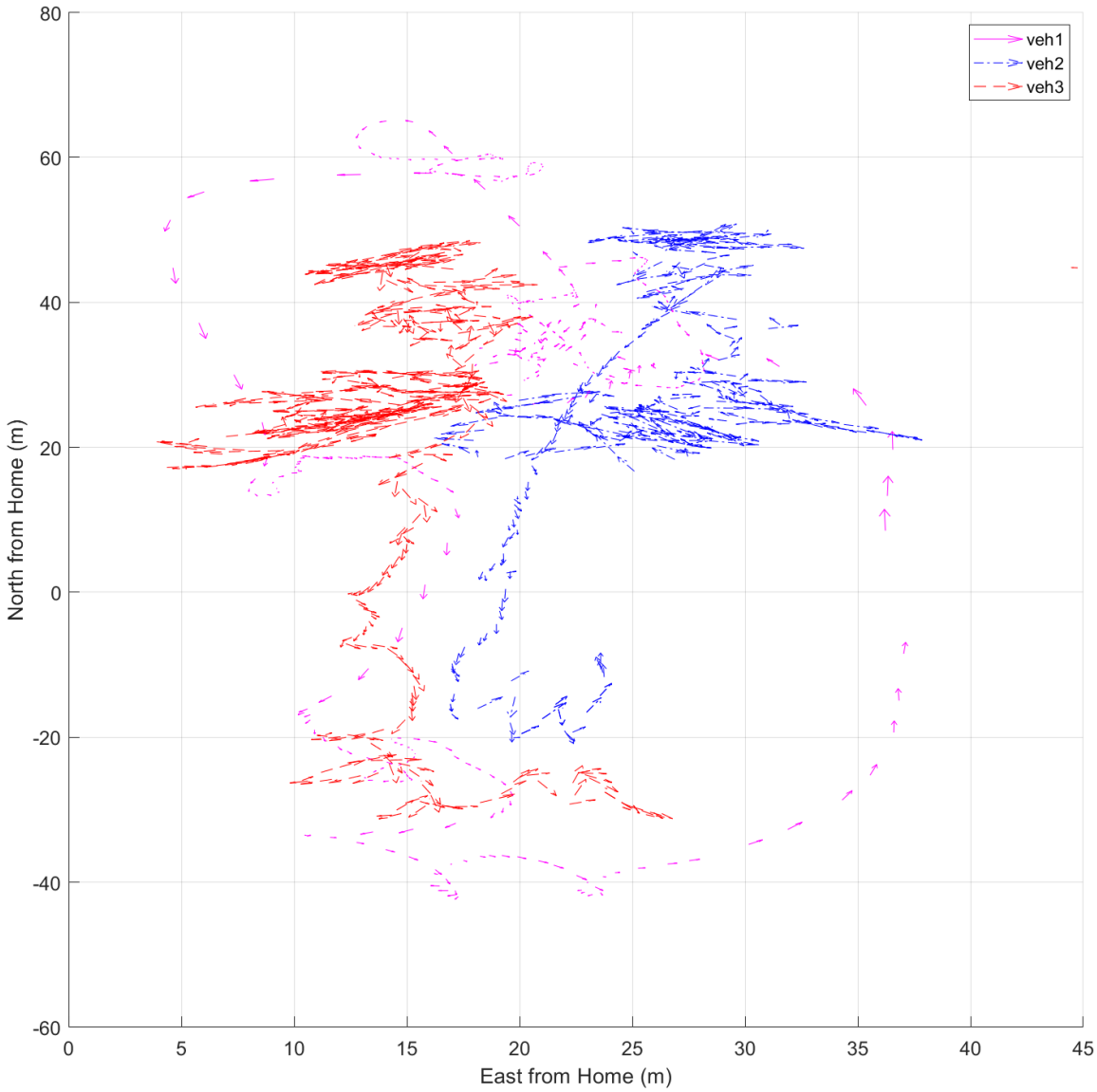


Figure 22. Flight 1, 2D flight paths with the lead vehicle in 'ALT_HOLD'.

in 'ALT_HOLD' to ensure the followers would respond to the lead vehicle and one another simultaneously. Following the path of Vehicle 1 in Figure 22, it first traveled north and paused at (20, 60) where the followers then moved to (10-30, 50) and stabilized. It then moved east and south. During the movement south, the follower flight paths reflect their movement in the same direction. Finally, Vehicle 1 moved east and shortly after the followers reacted, they were placed in 'STABILIZE' and recovered.

In general, this test demonstrated successful following behavior, but the absence of v_{max} on the leader under manual control resulted in delayed reactions and slower movement from the followers restricted by v_{max}. This reinforced the test team's desire to use the autopilot to limit the waypoint navigation speed and altitude of the leader to keep individual vehicle performance equal. The final observations from this flight were via the ground node using LCM spy. The presence of messages from all vehicles provided additional reinforcement that the algorithm was controlling each vehicle. Although some messages appeared as duplicates in short succession, there appeared to be no adverse effects on swarming behavior. Testing concluded after this mission due to an increase in wind and the late time of day.

Test Day Two Summary.

The second day of testing presented calmer conditions with winds $\approx$ 5 mph and warmer temperatures. Testing began with a two-vehicle swarm, with the lead vehicle in 'AUTO' following the predefined flight path. During this test, the follower appeared to perform better than day one due to the velocity restriction on the lead aircraft. After this concluded, a three-vehicle swarm launched represented by the flight paths and waypoints shown in Figure 23. The approximate waypoint positions for Vehicle 1's autonomous route are indicated by '1-9' on the plot.

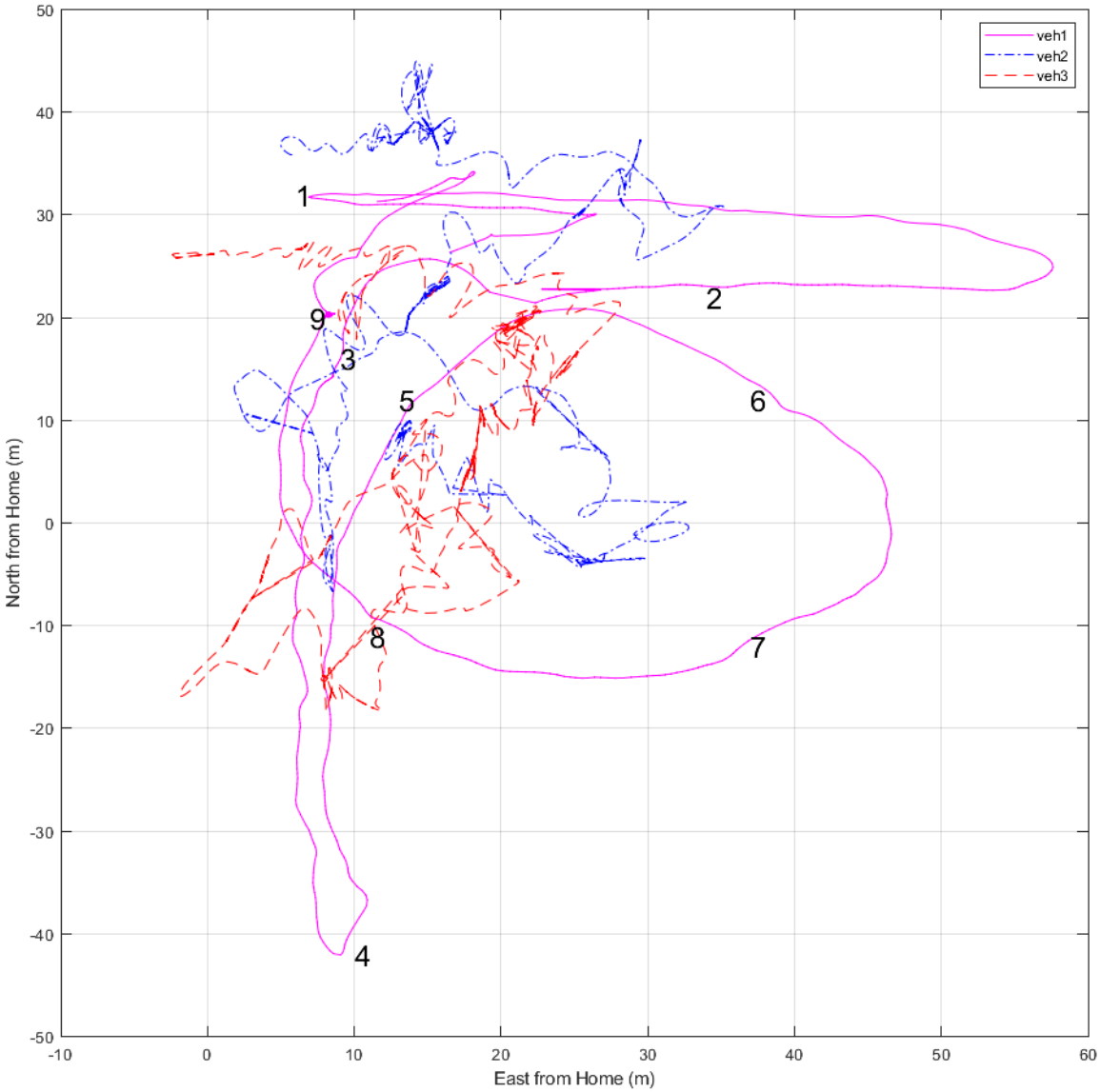


Figure 23. Flight 3, 2D flight path of first three vehicle swarming flight.

This flight gave the appearance of successful three vehicle swarming, with the lead vehicle following the waypoint plan and the followers moving in the same general directions while reacting to the position of one another. Figure 24 illustrates proper adherence to the separation rule for a majority of the flight, with 2 violations of r_{min} at $\approx$ 240 & 310 s. This plot also indicates several large vehicle-vehicle spacing's that occurred between Vehicle 1 & 3 and 1 & 2 at waypoints 2, 4, and the arc from 6-8. The last observation includes two periods from 180-200 & 225-300 s where vehicle-

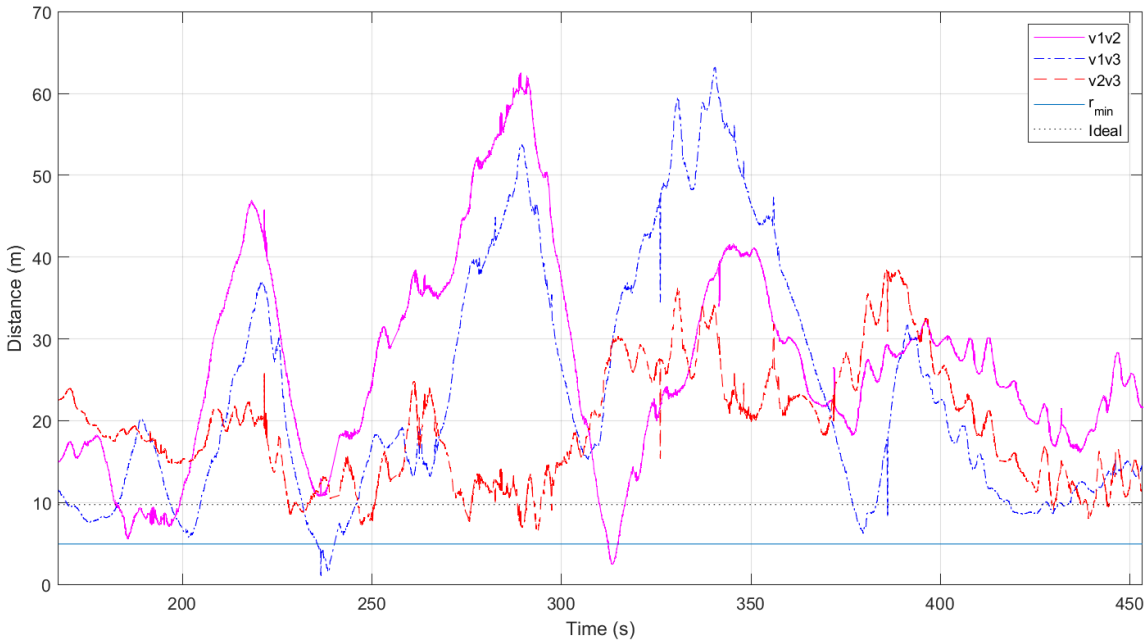


Figure 24. Flight 3, Vehicle-Vehicle distance during the first three vehicle swarming flight.

vehicle spacing remained near the ideal distance. This occurred near waypoint 1 for V1V2, and while Vehicle 1 was approaching waypoint 4 for V2V3.

A final note regarding Vehicle 2 performance during test day two is the emergence of the unresponsive behavior observed in SITL. Figure 25 illustrates this with Vehicle 2 remaining stationary around (11, 43) m. While the test team recognized this phenomenon, the decision was made to continue testing to document swarm performance in this degraded state. Upon doing so, Vehicle 3 appeared to follow Vehicle 1 at an approximate halfway point. More detailed analysis of this behavior is given later.

Qualitative Summary.

The following qualitative observations summarize swarm performance for three-vehicle missions, with a detailed analysis of specific flights addressed in the next section. When $v_{max} = 2$ m/s, the follower vehicles appeared to respond slowly but still exhibited swarming behavior. This slow responsiveness seemed to improve when

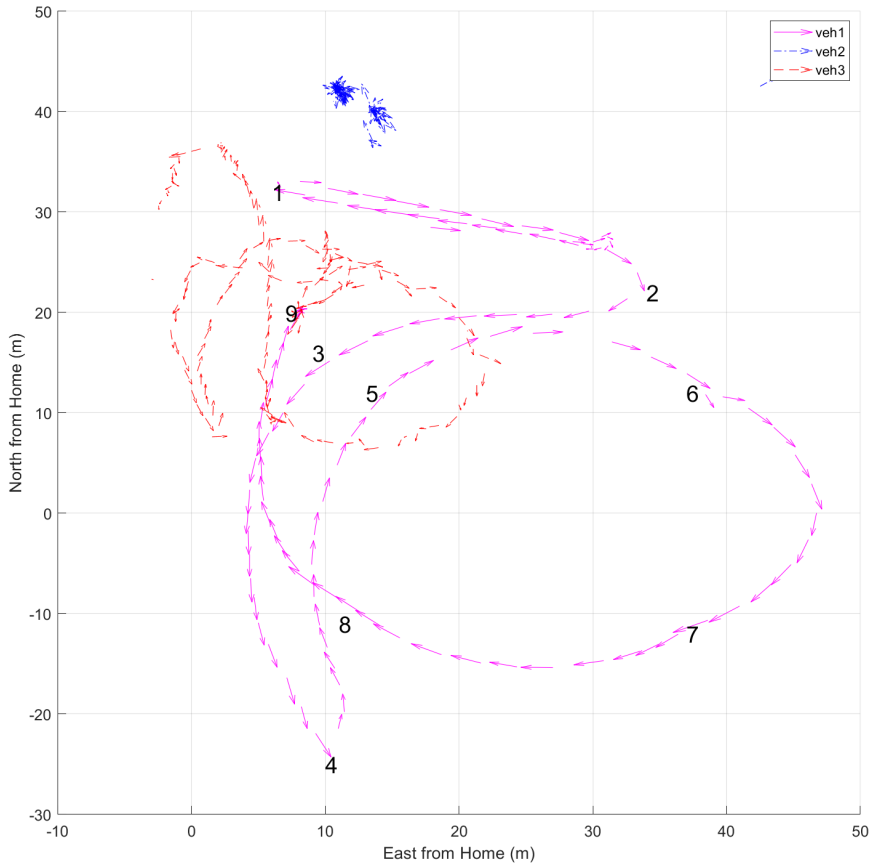


Figure 25. Flight 8, 2D flight path showing unresponsive behavior of Vehicle 2.

r_{min} was reduced from 10 to 5 m and further when v_{max} was increased to 3 m/s. Vehicle responses to input commands and perturbations always appeared smoother when $b = 0.25$. At $b = 0.5$, responses seemed more aggressive as was previously observed in SITL testing. The update frequency was not varied enough during testing to make any qualitative assessment of its impact on swarm behavior.

Test team opinion on the use of UgCS as a GCS during flight tests was favorable, citing its ease of use and 3D interface. Navigating multiple vehicle connections and monitoring pertinent telemetry data was also convenient for the operator and allowed them to keep the safety pilots informed when battery voltages were low or communication issues arose. It also offered the operator visualization of the swarm in 3D space similar to SITL tests. The required setup of the 915 MHz transmitters

plugged into one USB hub drew concern regarding transmitting power, as Vehicle 3 frequently dropped telemetry with the GCS. This was mitigated by powering the hub with a 5V, 2 Amp power adapter and was ultimately solved by replacing the 915 MHz transceiver pair. Although the focus of this research was not aimed at verifying UgCS as a viable GCS for multi-vehicle testing, it proved a valuable asset for reducing personnel and equipment. At a minimum, it is recommended for SITL testing and flight testing when simultaneous monitoring of telemetry, mission planning, and data logging for multiple vehicles is required.

4.4 Data Analysis Procedures

Before briefly explaining the data analysis techniques used, definitions of terminology used in evaluating the swarm. The following terms are found throughout the text:

- 'Ideal' Vehicle-Vehicle Distance: The equilibrium distance between two vehicles calculated using the Reynolds+ rule set as previously demonstrated in Chapter III.
- 'Ideal' Distance to Centroid or Half-Median: The distance from a vehicle to the calculated centroid of the three-vehicle swarm. Adapted from geometric term *median of a triangle* or the "line joining a vertex to the midpoint of the opposite side" (Reference, 2018).
- Commanded Velocity Magnitude: The velocity vector magnitude sent to the autopilot from the algorithm. Equivalent to the Macro level output.
- North, East, from Home: The distance (m) from the home GPS coordinate of each vehicle as set in the algorithm.

After flight testing, the LCM logs were transferred from the BBB to a corresponding Vehicle folder on a Linux computer. The log files were then converted to MATLAB matrix (.mat) files using the 'log_to_mat.py' script. This code was modified slightly from the open source 'libbot 2' GitHub repository to decode and convert all

messages in the LCM log to a matrix file (MIT, 2019). After conversion, all logs were transferred to the main Windows 10 workstation. The new matrix files were then imported into MATLAB as structures for data processing.

The second step of data processing was to plot the 2D flight path of the vehicles using the recorded position data. This ensured the proper grouping of each Vehicle's log by flight number. Recorded telemetry logs in UgCS were replayed for each flight as a supplemental tool for recalling the test flights and identifying specific areas of interest for quantitative analysis. An example of this telemetry replay is shown in Figure 26. It was also discovered that exported UgCS telemetry logs could be converted from their native .tlm format to a .csv format for use in Excel using the standalone UgCS telemetry tool (Engineering, 2018). Autopilot mode changes and the corresponding times were extracted from this data to further aid in processing.

Upon ingesting the matrix files into MATLAB with proper algorithm parameters assigned to each flight, multiple calculations were made. The first determined the

Figure 26. UgCS telemetry player window showing three vehicle tracks during the same time period.

start and end point of three-vehicle swarming, as each vehicle was not placed in or removed from 'GUIDED' at the same time. This trimmed excess data from the front and back of the log, providing an estimated swarming duration. A small calculation to correct outlier values in the altitude data set was performed as some values had improper 'signs,' resulting in spikes when plotting. Next, all three vehicle position and velocity structures were condensed into a table in time order to aid in centroid calculations. The centroid calculation progressed chronologically through Vehicle 1's data points in this table, identifying Vehicle 2 & 3 data points which were within a 0.15 s time tolerance. This resulted in a time-based centroid estimation relative to the data point of Vehicle 1.

Each vehicle's distance (m) to the centroid and vehicle-vehicle distance (m) was calculated in the NED frame using these same data points. These vector components were then used to find the corresponding magnitudes. Deviations from the 'ideal' state for both of these metrics were calculated against the parameters used in that flight. Mean values for all of these data points were found and are summarized in table form. The final calculation was the velocity magnitude of the vehicle as reported by the autopilot. This was compared to the commanded velocities. Many other values could be extracted from the recorded data; however, only those best supporting the identified testing metrics were chosen. A variety of plots were created to illustrate swarm behavior and are selectively included in the next section. Summary statistics and final communication metrics are discussed later.

4.5 Flight Test Data

With an understanding of the test progression and the underlying analysis procedures used, the following analyzes and verifies swarming behavior for several flights using the testing metrics of swarm 'goodness' and rule violations. Various plots of

Table 4. Summary of executed missions during flight testing. The value of $f_{min} = 10$ for all flights.

Vehicle			r_{min} (m)	v_{max} (m/s)	b	u (Hz)
1	2	3				
ALT_HOLD	GUIDED	GUIDED	5	2	0.5	5
			5	2	0.25	5
AUTO	GUIDED	GUIDED	10	2	0.5	5
			10*	2*	0.25*	5*
			10	3	0.25	5
			5	2	0.5	5
			5	2	0.5	10
GUIDED	GUIDED	GUIDED	5	2	0.5	5
			5	2	0.25	5
			5	3	0.25	5
*Completed twice						

data are used as supporting evidence when evaluating the algorithm's performance for the selected test flights. Regions of nominal algorithm performance are identified, as well as areas of unexpected swarm behavior. Table 4 summarizes the parameters used during three vehicle swarming flight test missions.

Flights of Manual Lead Vehicle Manipulation.

The first two flights for analysis are Flights 1 & 6 when Vehicle 1 was placed in 'ALT_HOLD' and manually controlled by the safety pilot. Figures 22 and 27 show the flight paths taken during these missions. The main purpose of these flights was to manipulate the follower vehicles with lead vehicle movements to ensure rule compliance. As mentioned in the Flight Test Overview, these tests provided immediate feedback to the test team that the algorithm was performing as expected. They also served as a familiarization for the safety pilots as they had not previously observed vehicle behavior under algorithm control.

Three-vehicle autonomous swarming was also tested during Flight 6 which is represented by the 170-190 s region in Figure 28. A significant observation of vehicle

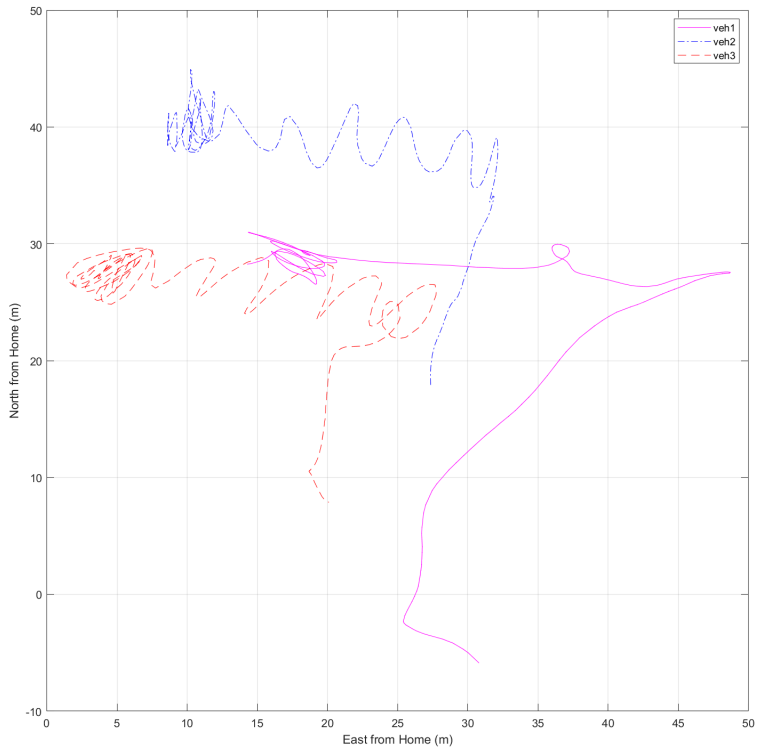


Figure 27. Flight 6, Vehicle paths with the lead vehicle under autonomous and manual control.

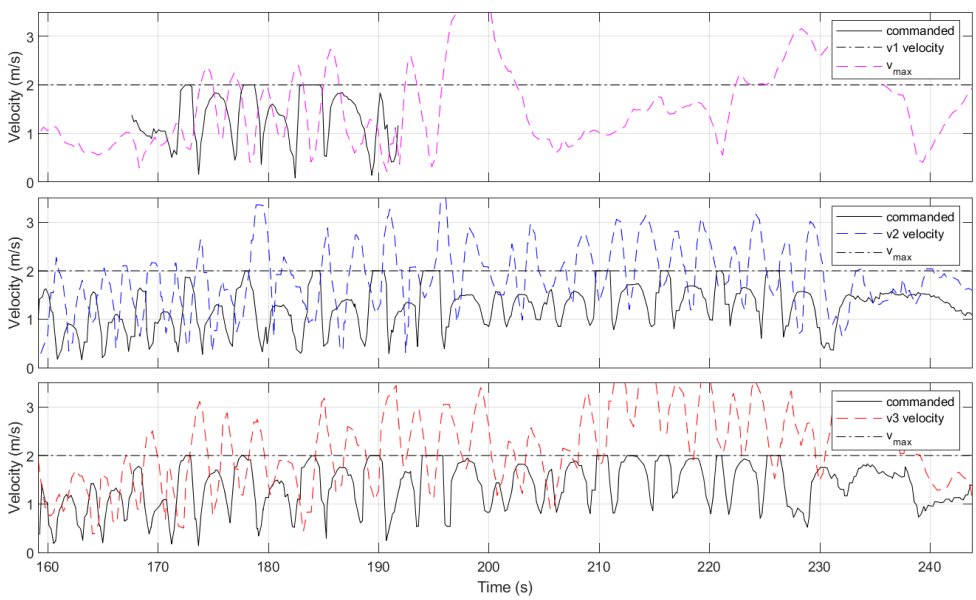


Figure 28. Flight 6, Vehicle Velocity vs Commanded Velocity. $v_{max} = 2m/s$, $r_{min} = 5m$, $b = 0.25$, $u = 5Hz$.

behavior from this plot is the reported velocity exceeding v_{max} under algorithm control. The regions beyond 190 s for Vehicle 1 are pilot induced. While this was unanticipated, it is logical that a vehicle traveling at 1 m/s north with a commanded velocity of 2 m/s in the same direction could exceed 2 m/s. Figure 29 shows the trend between algorithm commanded and actual velocity magnitudes in NED vector components for Flight 6. It should be noted that the presence of the cohesion damping parameter $a = -1$, restricts the magnitude of the vertical velocity commands.

An interesting phenomenon of heavy oscillation occurred during both of these flights previously observed in SITL testing at $u = 2$ Hz. While it was apparent during flight as a position change, the data for Vehicle 2 revealed these oscillations in the commanded North velocity component as well. Performing a Fourier transform on this data, the frequency of the oscillations was revealed in Figure 30 to be ≈ 0.2 Hz. As shown in Figure 27, when all three vehicles were in 'GUIDED' they would

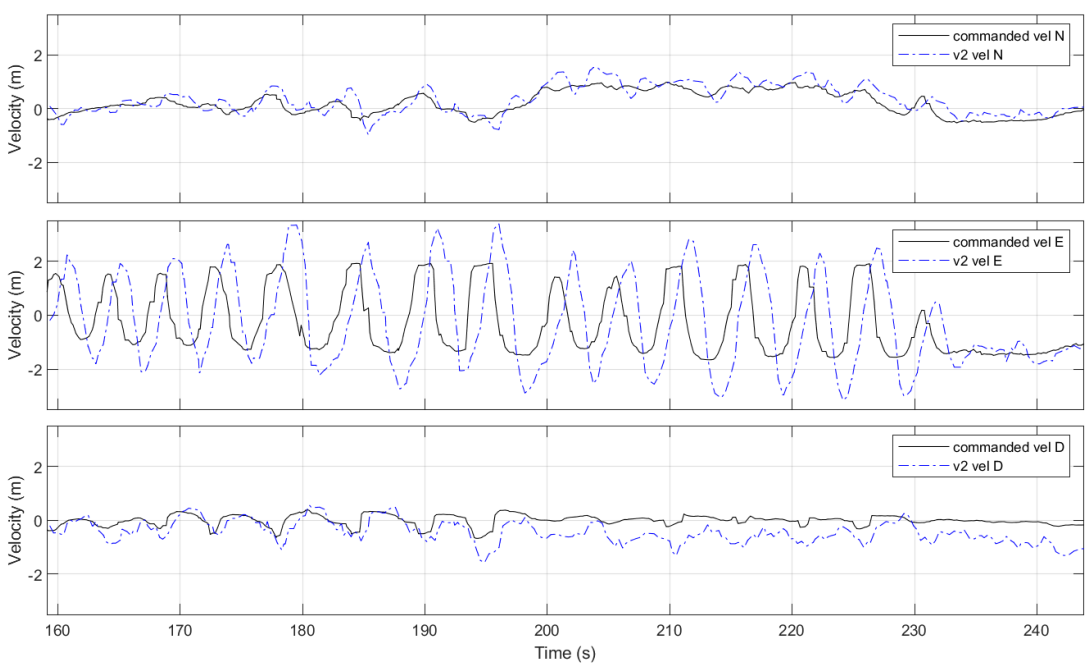


Figure 29. Flight 6, Vehicle 2 Velocity vs Commanded Velocity in NED. $v_{max} = 2m/s$, $r_{min} = 5m$, $b = 0.5$, $u = 5Hz$.

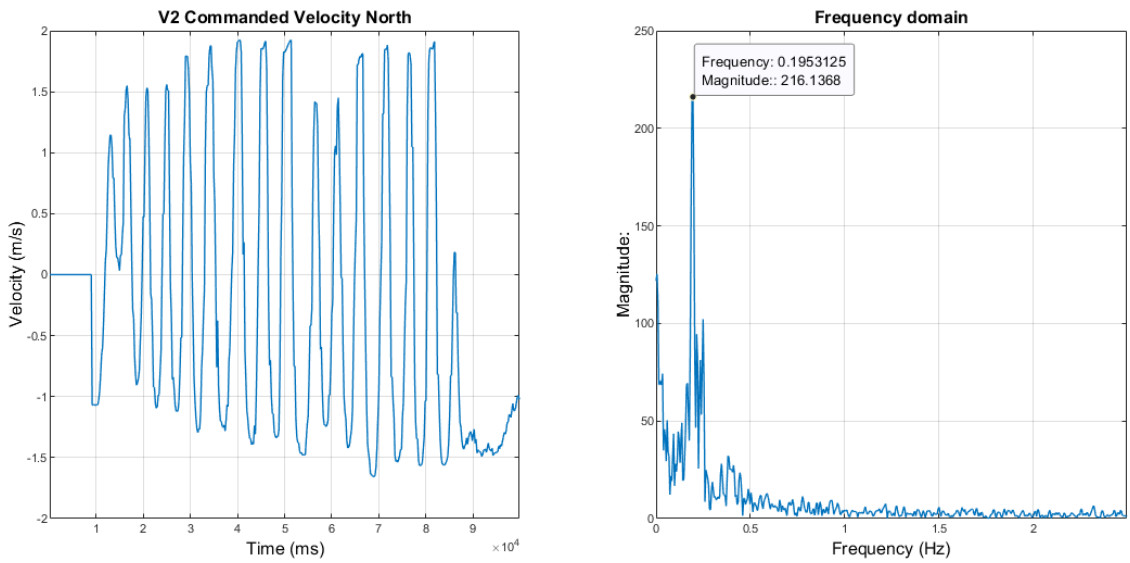


Figure 30. Fourier Transform of Vehicle 2's Commanded Velocity during Flight 6. $v_{max} = 2m/s$**,** $r_{min} = 5m$**,** $b = 0.5$**,** $u = 5Hz$**.**

oscillate in place at this frequency, with position variations of $\approx$ 3 m. Another interesting observation was that Vehicle 2 & 3 continued to oscillate even as Vehicle 1 moved east. When it finally moved south, these oscillations halted. Transformations on similar data from later flights revealed minor concentrations of oscillations at ≈ 0.2 Hz, even in the case of full flight plan completion.

During these large oscillations, the vehicle-vehicle distance remained above r_{min} when under autonomous control as shown in Figure 31. This was considered a success when factoring in the position change amplitudes ranging from 2-4 m between different vehicle pairs. The r_{min} violation at 200 s was a result of manual control of Vehicle 1 and not the algorithm. It is interesting to note the opposing changes in vehicle distances that are present with V1V2 & V1V3. During this time, the V2V3 distance had much smaller variations. This suggests that Vehicles 2 & 3 were almost synchronized in their oscillations while Vehicle 1 was opposite. Average inter-vehicle distance for these flights was of less interest since it was skewed by the safety pilot creating gaps between the vehicles with velocities $>> v_{max}$.

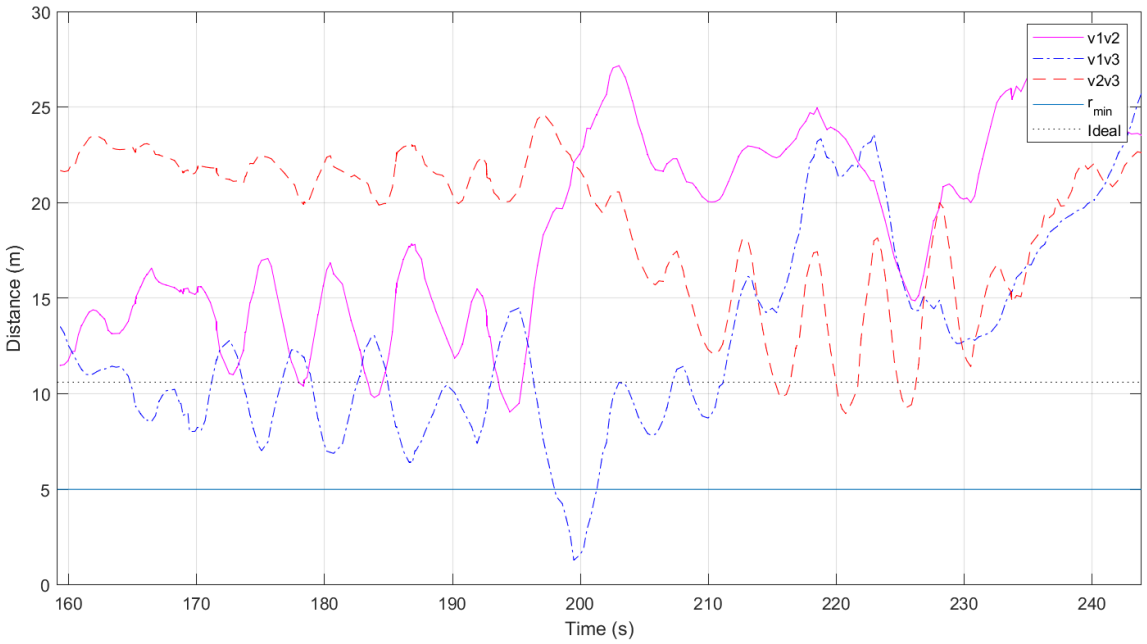


Figure 31. Flight 6, Vehicle-Vehicle Distances with the lead vehicle in 'ALT_HOLD'.

Autonomous Flight Comparison to Simulation.

A total of six successful, three-vehicle, autonomously controlled flights with different parameters were made during testing as summarized in Table 5. A brief analysis of these data sets is presented to highlight algorithm and swarming performance. Figure 32 shows the flight paths of all three vehicles during Flight 15 as it was one of the best representations of swarming behavior. During this flight, vehicles rarely violated r_{min} as shown in Figure 33 with an overall violation of 0.48%. Additionally,

Table 5. Summary of successful autonomous missions during flight testing.

Flight	r_{min} (m)	b	v_{max} (m/s)	f_{min} (m)	Ideal V-V (m)	Ideal Centroid Dist (m)
4	5	0.5	2	10	9.81	5.66
7	10	0.25	2	10	18.26	10.54
10	5	0.25	2	10	10.61	6.12
11	10	0.25	2	10	18.26	10.54
12	10	0.25	3	10	18.26	10.54
15	5	0.25	3	10	10.61	6.12

deviation from the ideal vehicle-vehicle distance between vehicle pairs averaged only 7.81 m.

An approximate comparison to a SITL simulation of this flight using the same plots in Figures 34 & 35 yields several interesting results. It should be noted that the starting position of all three vehicles did differ from live flight testing, and thus the initial distances may have an effect on overall averages and observed behaviors. Additionally, the duration of the two tests differs, with the simulation taking ≈ 140

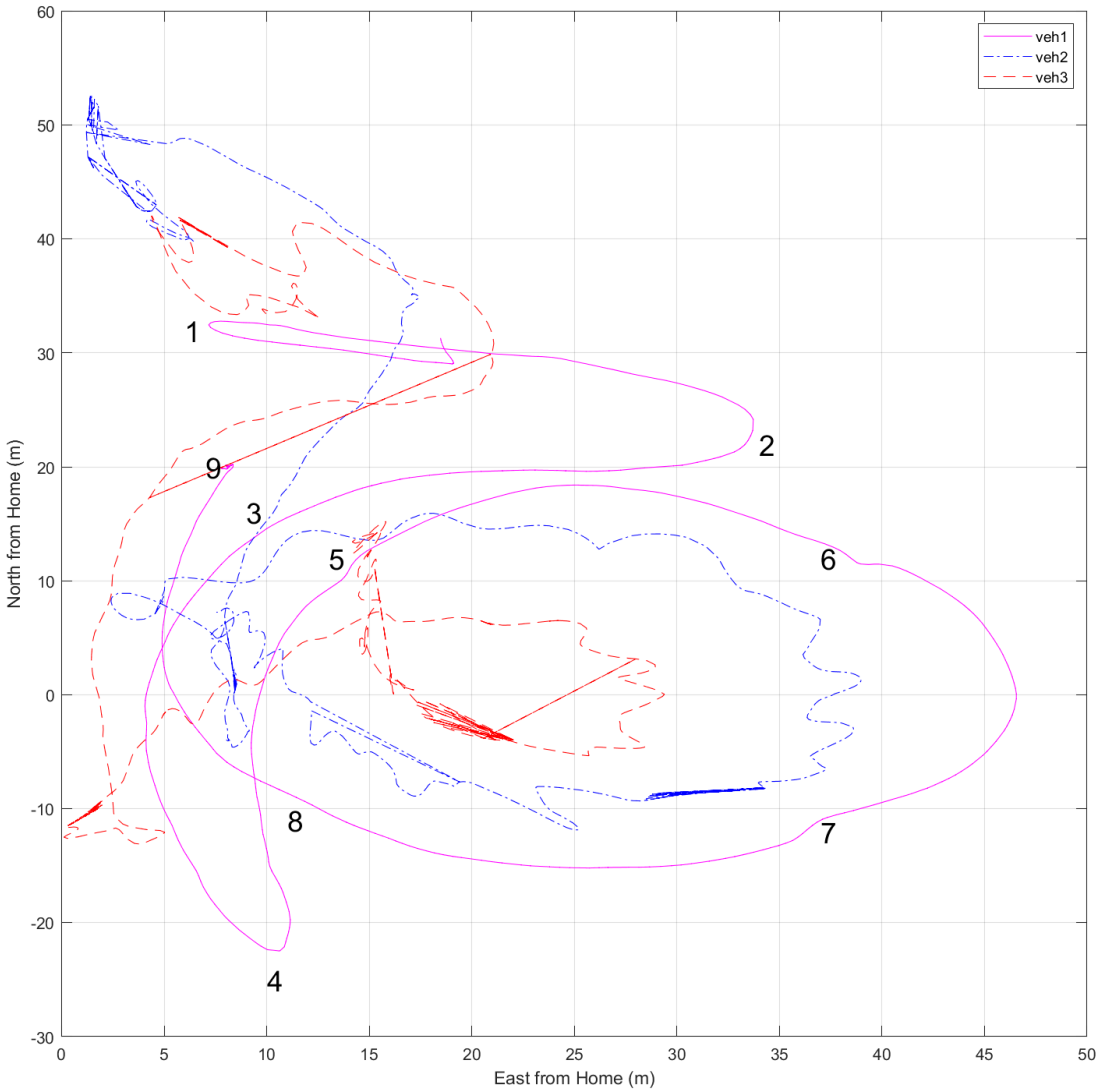


Figure 32. Flight 15, Plot of vehicle paths under autonomous control.

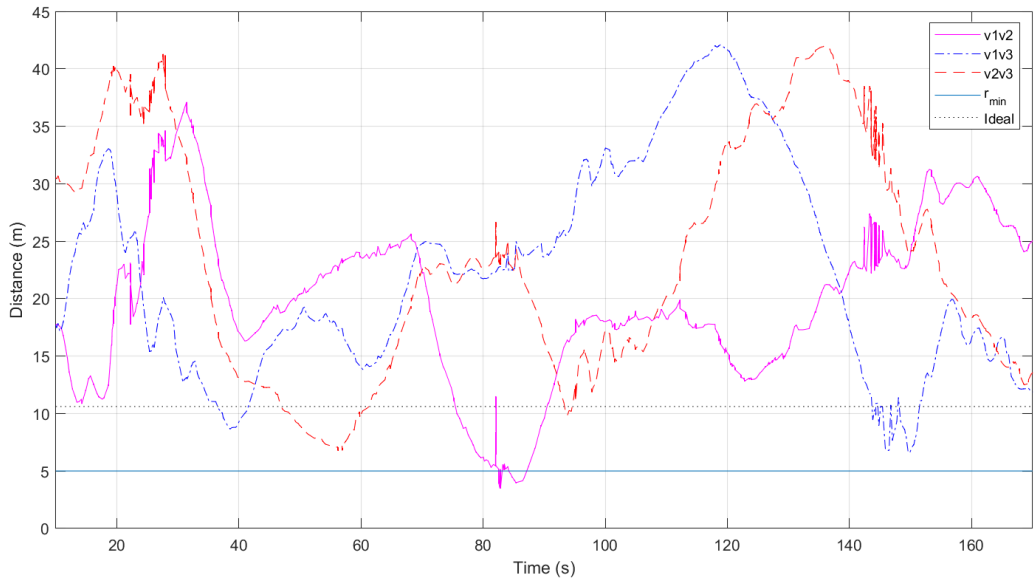


Figure 33. Flight 15, Vehicle-Vehicle distances with $v_{max} = 3m/s, r_{min} = 5, b = 0.25, u = 5Hz$**.**

s to complete waypoints 1-9, while the live flight test took ≈ 160 s. There are, however, similarities in the resulting vehicle-vehicle distances, with the two major troughs occurring when Vehicle 1 passes waypoints 3 & 5, and the major peaks occurring at waypoints 2, 4, & 6-7. Additionally, in Figures 32 & 34 similar paths appear to be taken by Vehicles 2 & 3 while Vehicle 1 completes waypoints 4-9.

The first significant difference is found in Figure 35, where consistent vehicle-vehicle distance oscillations caused by commanded velocities are present through the entire simulated flight. Closer inspection reveals these oscillations occurred at ≈ 0.83 Hz with an amplitude of ≈ 0.25 m. In Figure 32, similar oscillations are still present, but the higher amplitude, $0.2 - 0.25$ Hz oscillations are more apparent and influential on swarming performance. Potential causes for the differences in these results are the exact position data in simulation versus the ±2-3 m error of real-world GPS data and the influence of environmental factors.

Some large vehicle-vehicle distances still occurred in simulation when Vehicle 1 was at the extremities of the flight plan. These are most likely caused by the matching

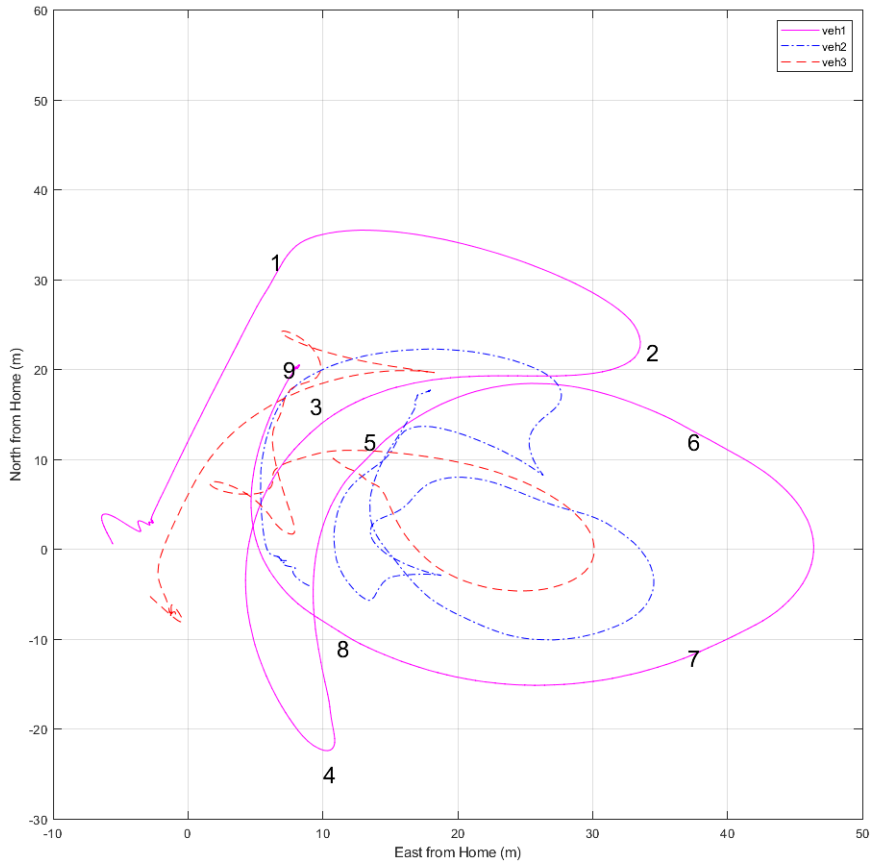


Figure 34. Flight 15-SITL, Plot of vehicle paths in SITL under autonomous control.

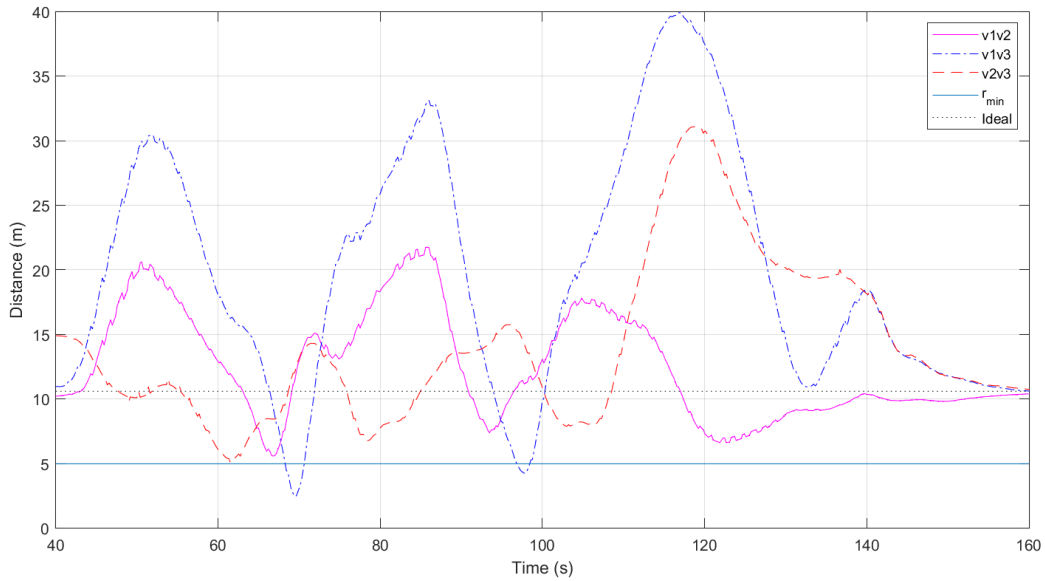


Figure 35. Flight 15-SITL, Vehicle-Vehicle distances with $v_{max} = 3m/s, r_{min} = 5, b = 0.25, u = 5Hz$.

waypoint and v_{max} speed of the leader and follower vehicles. With these differences in mind, relative conclusions regarding swarm performance are made. On average, vehicles violated r_{min} at 0.73% in simulation, which was marginally higher than the 0.48% of live flight. Separation distances for this test flight averaged 14.33 m in the SITL test, while live flight testing averaged 21.04 m. Lastly, the update rate of the algorithm loop in simulation averaged 4.98 Hz compared to 4.61 Hz on the BBB as expected, due to the processing differences in the hardware.

Stable Swarming.

The vehicle-vehicle distances of Flight 12 are provided in Figure 36 to highlight a successful convergence to the stable swarm state when $r_{min} = 10$ m. For this flight Vehicle 1 reached the final waypoint at ≈ 335 s and was allowed to hover in 'AUTO' for the final 27 seconds. Due to the flight paths of the followers shown in Figure 37, the vehicle-vehicle distances were reduced as Vehicle 1 approached the final waypoint.

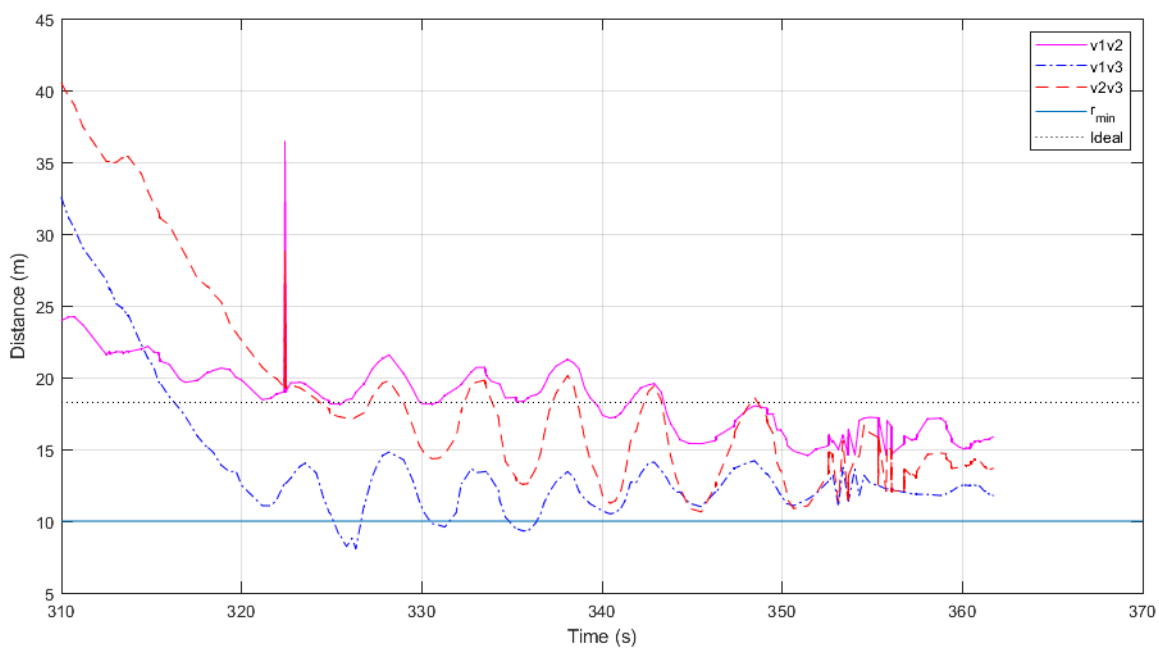


Figure 36. Flight 12, Vehicle-Vehicle distances with $v_{max} = 3m/s, r_{min} = 10, b = 0.25, u = 5Hz$.

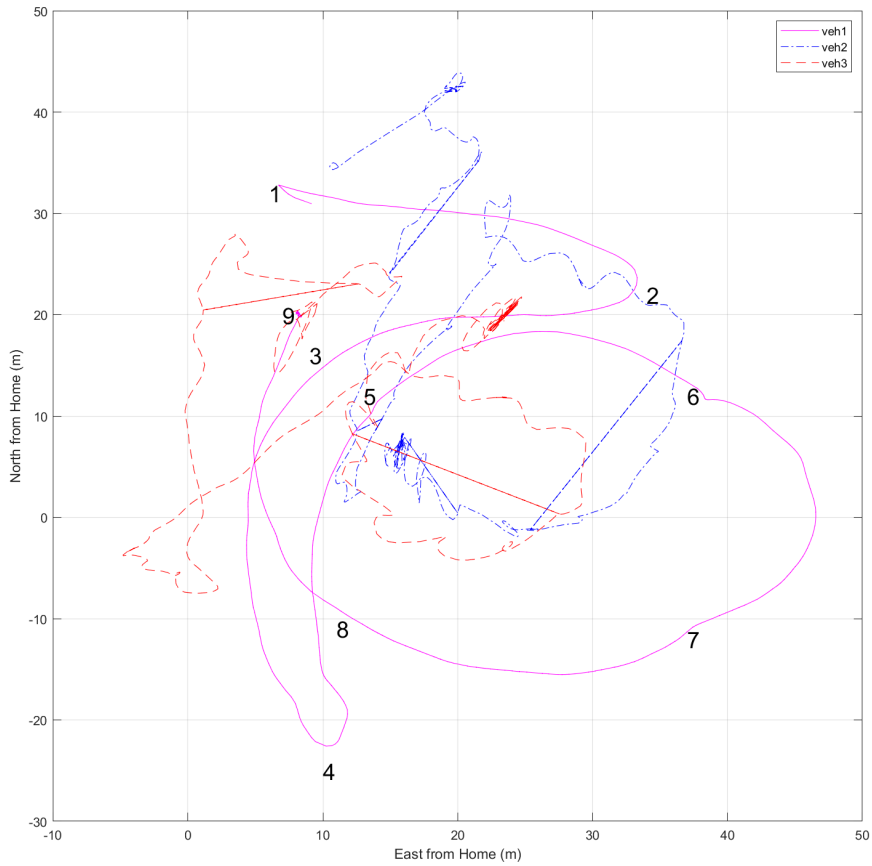


Figure 37. Flight 12, Plot of vehicle paths with $v_{max} = 3m/s, r_{min} = 10, b = 0.25, u = 5Hz$.

While Vehicle 1 held position, the followers' oscillations eventually dampened as they approached an 'ideal' swarm state. This test was performed again at the end of Flight 15 but with $r_{min} = 5$ m. A similar time frame of 60 seconds was used, although the axes were shifted to compensate for the time it took Vehicle 2 to reach the other swarm members.

For this flight, Vehicle 1 arrived at the final waypoint at 155 s and held the position for an additional 55 seconds. In this case, the follower vehicles were not already near the waypoint, and the resulting vehicle-vehicle gap closure is visible. Once Vehicle 3 arrives near the 'ideal' swarm state with Vehicle 1, the distance between the two vehicles appears very stable. However, as Vehicle 2 approaches this state, it exhibits oscillatory behavior until $\approx$ 195 s which is represented in both the V1V2 and V2V3

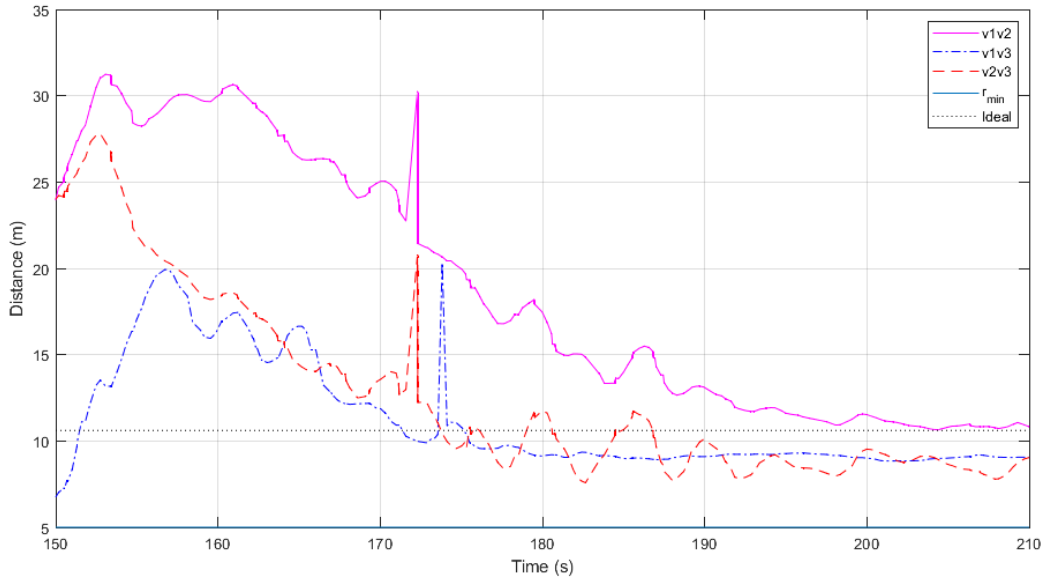


Figure 38. Flight 15, Vehicle-Vehicle distances with $v_{max} = 3m/s, r_{min} = 5, b = 0.25, u = 5Hz$.

distances. A final observation from Figures 36 & 38 is the average vehicle-vehicle distance trending toward ≈ 15 & ≈ 9 m respectively. These values differ from the calculated 'ideals' of 18.26 m and 10.61 m for the same parameters.

Reynolds+ Rule Adherence.

A major component in the verification of this algorithm is concerned with vehicle adherence to the Reynolds+ rule set. By comparing the commanded velocity vector magnitudes with vehicle-vehicle and half-median distances, general conclusions are made about the algorithm's ability to control swarm members. Interpretation of several plots will identify rule violations and in which scenarios various rules have the largest magnitude influence. Figures 39 & 40 provide an example of this comparison from Vehicle 2 & 3 respectively. Looking first at Figure 39, several instances exist where the V2V3 distance approaches r_{min} during the flight. These are matched by correspondingly higher velocity magnitude commands resulting from the separation rule. Other high magnitude commands are a result of the cohesion rule and velocity

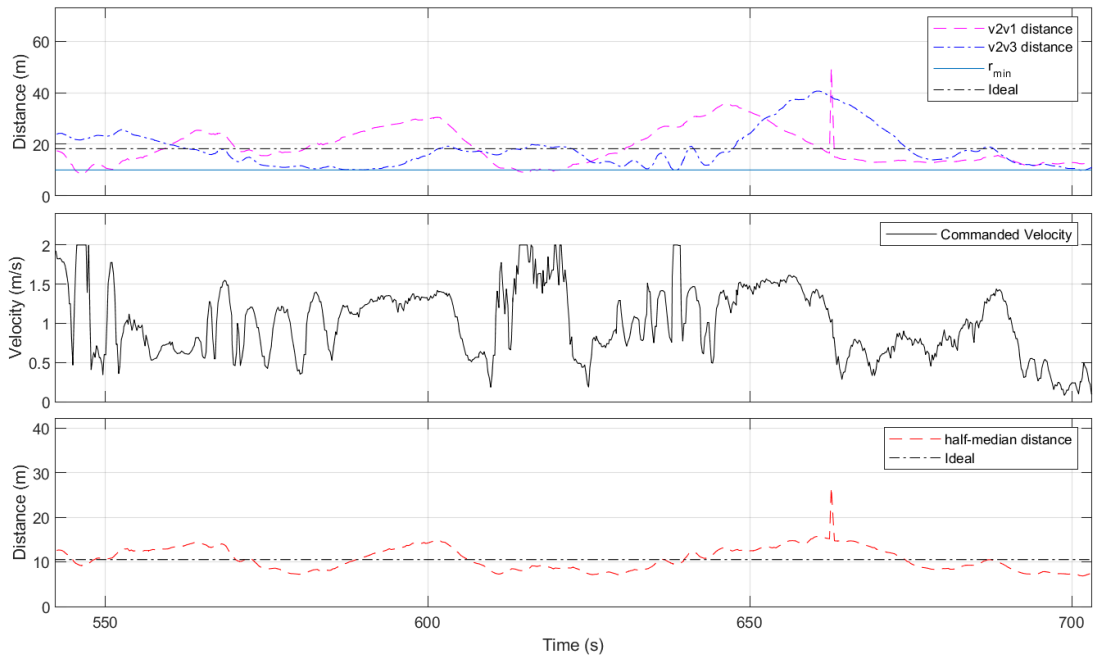


Figure 39. Flight 7, Vehicle 2 commanded velocity magnitude versus distances with $v_{max} = 2m/s, r_{min} = 10, b = 0.25, u = 5Hz$.

alignment when the vehicle-vehicle and half-median distances both increasing during 640-670 s.

Figure 40 shows a similar trend for Vehicle 3 during the same flight. Early portions of the flight (550-570 s) show violations of r_{min} between Vehicle 1 & 3 with only 1 m/s commanded velocities. In this case, it is assumed that other rules were counteracting the separation rule, thus reducing its effective magnitude. During portions of near-ideal values for vehicle-vehicle and half-median distances (600-620 s), the commanded velocities are assumed to be the result of the alignment rule with smaller magnitudes from other rules still present. Near 670 s the separation rule is again observed commanding high velocity magnitudes as Vehicle 3 approaches r_{min} with Vehicle 1. Similar rule adherence can be found in any other vehicle-vehicle or half-median distance figures previously presented.

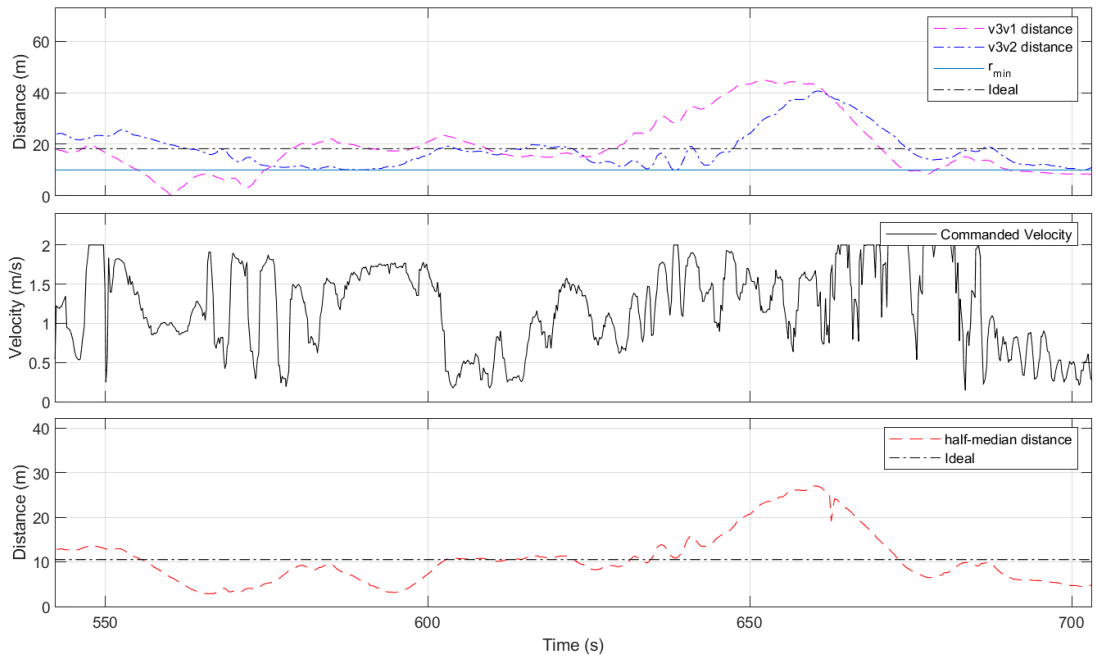


Figure 40. Flight 7, Vehicle 3 commanded velocity magnitude versus distances with $v_{max} = 2m/s, r_{min} = 10, b = 0.25, u = 5Hz$.

Final Observations in Quantitative Analysis.

Two additional analyses not directly related to the testing metrics were performed regarding consistent trends in the results. The first was an estimation of the delay from commanded to achieved velocity presumably due to vehicle momentum. Analysis of different flights revealed similar behavior matching the four peaks in Figure 41. The first piece of information gathered from this plot is the vehicle velocity overshoot of the commanded velocity and $v_{max} = 2$ m/s. In many cases, the algorithm then reacts with a sharp drop in velocity magnitude. The second is the estimated delay from commanded to the achieved velocity of ≈ 1 s for the two sets of peaks. Due to this delay, vehicles inevitably overshoot the intended positions calculated by the algorithm, and the algorithm compensates with a correspondingly high velocity vector in the opposite direction.

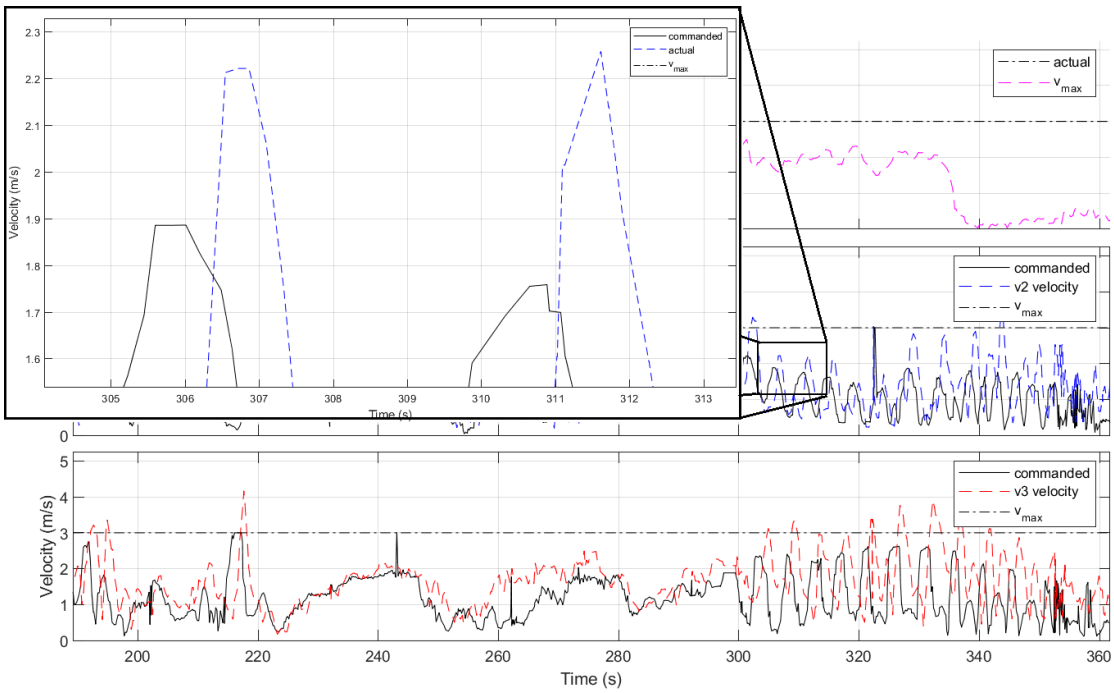


Figure 41. Flight 12, Sample of time delay between commanded and realized velocity magnitude.

The final analysis is regarding Vehicle 2's unresponsiveness to algorithm inputs and its effect on swarm performance. Figures 25 & 42 show 2D plots of this behavior. In these flights, Vehicle 2 is essentially stationary while Vehicle 3's path follows the calculated centroid of the swarm while maintaining its half-median distance. Figure 43 from Flight 8 represents this behavior well since the half-median distance of V3 is much smaller than V1 or V2 during the period of greatest separation. This shows that in the event of an unresponsive vehicle, the algorithm is correctly processing inputs and commanding correct velocity vectors. Figure 44 is provided as justification that the algorithm was sending proper velocity vectors to Vehicle 2's autopilot. Any resulting velocities are a representative of the autopilot holding position against the wind. The straight line from 200-250 s indicates when the vehicle was placed in 'STABILIZE' by the safety pilot which turned off the algorithm and data broadcasting.

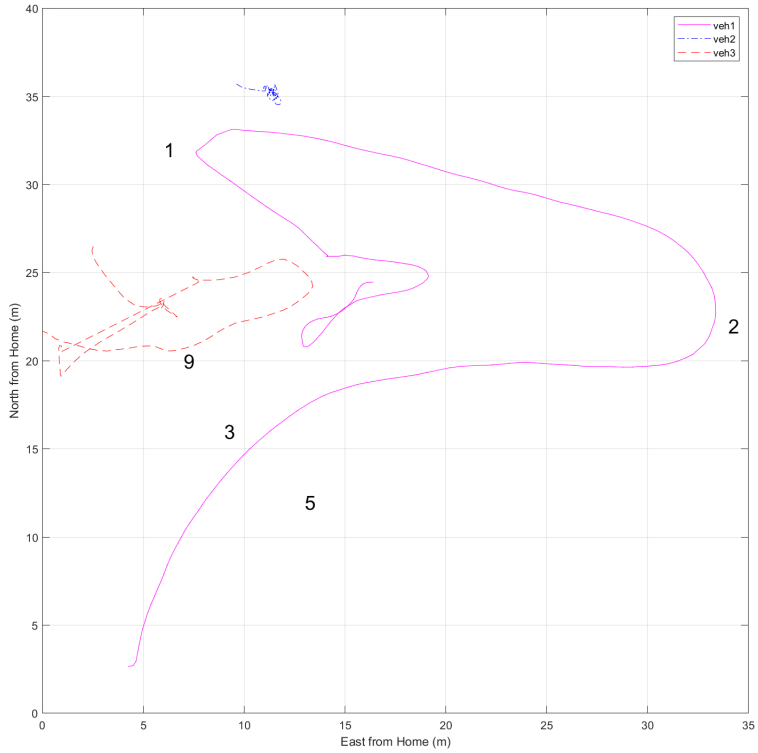


Figure 42. Flight 9, 2D plot with $v_{max} = 2m/s, r_{min} = 5, b = 0.5, u = 5Hz$ and Vehicle 2 unresponsive.

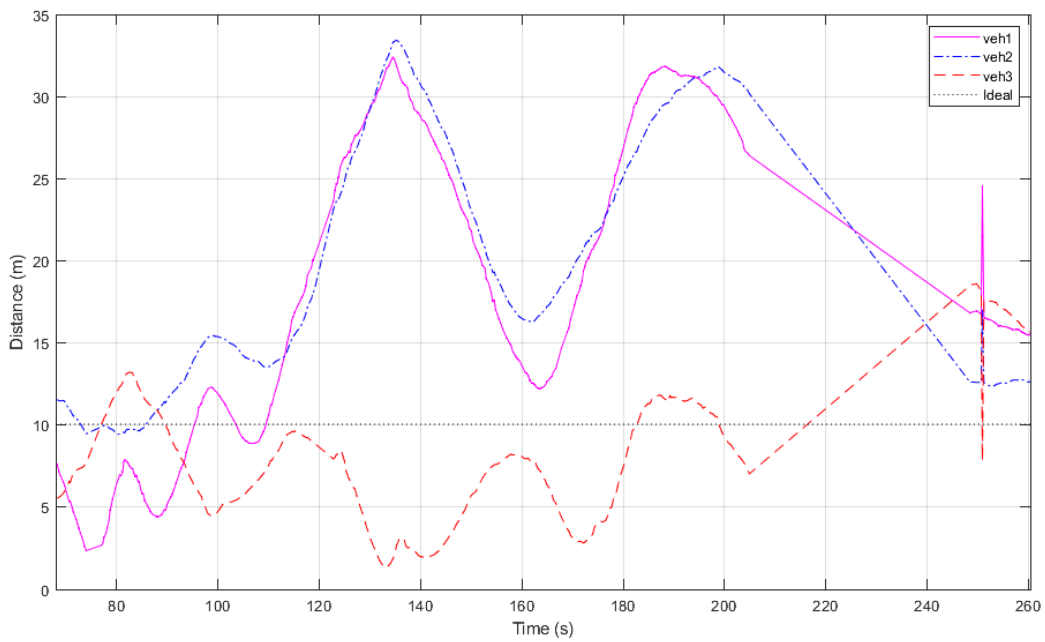


Figure 43. Flight 8, Vehicle half-median distance with $v_{max} = 2m/s, r_{min} = 5, b = 0.5, u = 5Hz$ and Vehicle 2 unresponsive.

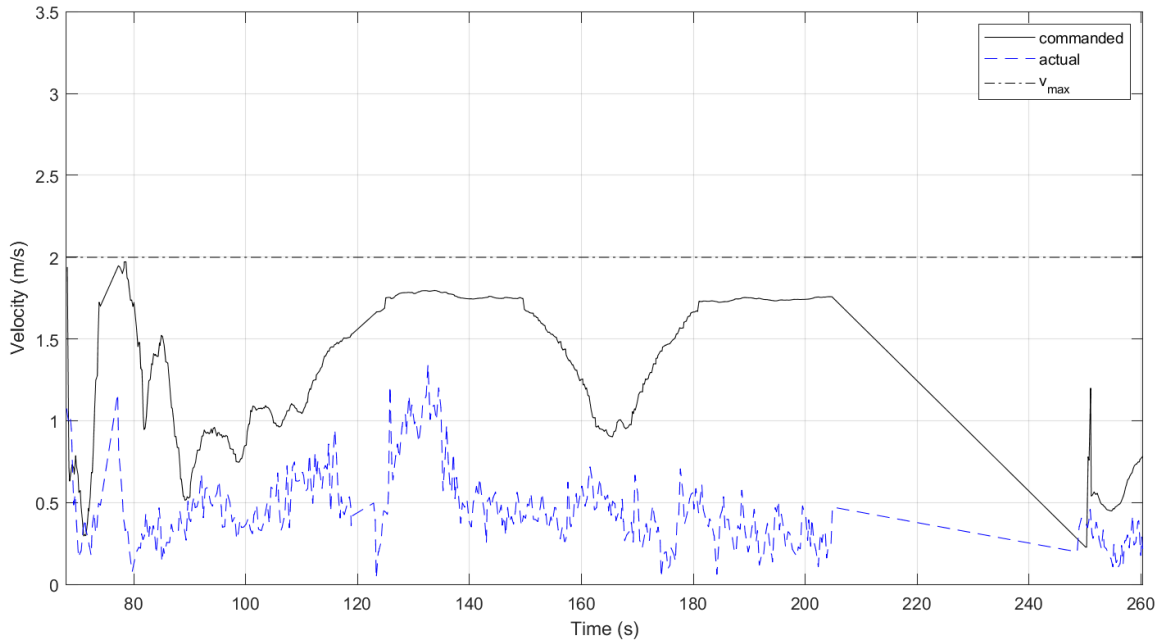


Figure 44. Flight 8, Vehicle 2 Commanded vs Vehicle Velocity.

4.6 Summary Flight Test & Communication Data

To conclude the analysis section, summary statistics for successful autonomous swarming and communication performance parameters are presented. Table 6 summarizes the test metric of average deviation for all autonomous flight which exuded proper flight performance without any previously discussed errors or anomalies. The one negative value indicates the vehicles were marginally inside the ideal distance. Overall, the values represent the successful execution of the rules and a variety of swarming proximity for the given parameters. The most impressive results occurred on Flights 7 & 12, with < 1 m average deviations across the three vehicles and $< 10\%$ r_{min} violations. Overall, the vehicle-vehicle distance deviated from the ideal state by an average of 5.61 m, with r_{min} violations $< 6.39\%$ of the time.

Additionally, the violation percentages during flights 10 & 15 are impressive given the lower average distance deviations. The average vehicle-vehicle spacing is shown in Table 7 for reference. Finally, the average deviation from the ideal distance to the

Table 6. Summary of average, ideal vehicle-vehicle spacing deviations and r_{min} violation percentage for successful autonomous flights. The value of $f_{min} = 10$ for all flights.

Flight	r_{min} (m)	b	v_{max} (m/s)	Ideal (m)	V1V2 (m)	V1V3 (m)	V2V3 (m)	Violation (%)
4	5	0.5	2	9.81	17.27	13.90	9.47	0.77
7	10	0.25	2	18.26	0.43	0.62	-0.25	9.49
10	5	0.25	2	10.61	5.61	7.44	6.25	2.8
11	10	0.25	2	18.26	2.81	6.97	4.09	15.72
12	10	0.25	3	18.26	0.97	0.36	1.54	9.11
15	5	0.25	3	10.61	5.79	8.92	8.73	0.43

centroid for the flights is given in Table 8. Comparing these values to Table 6, similar levels of performance are observed in flights 7 & 12, as expected.

Communication Statistics.

The primary investigations regarding communication are the time delay and packet losses of LCM messages between vehicles. Using log timing and comparing received messages on each channel from all three vehicles, the 'CommDelay.m' script in Appendix B was developed to output summary values. An additional measurement was calculated to determine the average algorithm loop frequency using the main analysis code in Appendix B. After analyzing the flight logs, it was discovered that three Vehicle 2 logs lacked any data for the other vehicles. In several other logs, LCM

Table 7. Summary of vehicle-vehicle spacing and r_{min} violation percentage for successful autonomous flights. The value of $f_{min} = 10$ for all flights.

Flight	r_{min} (m)	b	v_{max} (m/s)	Ideal (m)	V1V2 (m)	V1V3 (m)	V2V3 (m)	Violation (%)
4	5	0.5	2	9.81	27.07	23.71	19.28	0.77
7	10	0.25	2	18.26	18.69	18.88	18.00	9.49
10	5	0.25	2	10.61	16.22	18.04	16.85	2.80
11	10	0.25	2	18.26	21.07	25.22	22.35	15.72
12	10	0.25	3	18.26	19.22	18.62	19.80	9.11
15	5	0.25	3	10.61	16.40	19.52	19.34	0.43

Table 8. Summary of average half-median or distance to centroid deviation for successful autonomous flights. The value of $f_{min} = 10$ for all flights.

Flight	r_{min} (m)	b	v_{max} (m/s)	Ideal (m)	V1 (m)	V2 (m)	V3 (m)
4	5	0.5	2	5.66	10.08	7.83	5.24
7	10	0.25	2	10.54	0.38	0.10	0.03
10	5	0.25	2	6.13	4.18	2.79	3.70
11	10	0.25	2	10.54	2.51	1.92	4.06
12	10	0.25	3	10.54	-0.08	1.18	0.48
15	5	0.25	3	6.13	4.31	3.52	5.36

time stamps were corrupt (i.e. 9.3558e+07) for varying channels. As a result, only 8 logs were analyzed, with the corrupt data channels removed.

Calculated LCM communication delay time is based on the difference in log time between the sending vehicle and receiving vehicle for the same message. Figure 45 shows an example of a log from Vehicle 1 in the case where Vehicle 2 was the first to broadcast a message on the 'vehicle2' channel. This same message was found in

mat1 | mat2 | mat3 | mat1.vehicle1 | mat1.vehicle2 | mat1.vehicle3 | mat2.vehicle2 | mat3.vehicle2

mat1.vehicle1

	1	2	3	4	5	6	7	8	9	10
1	1	30.2769	24.2244	-233.0599	-0.3400	-0.3200	-0.5300	1.3640		
2	1	30.2769	24.2244	-233.0599	-0.3400	-0.3200	-0.5300	1.3683		
3	1	30.1658	24.1813	-233.0199	-0.3200	-0.0200	-0.3300	1.5756		
4	1	30.1658	24.1813	-233.0199	-0.3200	-0.0200	-0.3300	1.5780		

mat1 | mat2 | mat3 | mat1.vehicle1 | mat1.vehicle2 | mat1.vehicle3 | mat2.vehicle2 | mat3.vehicle2

mat1.vehicle2

	1	2	3	4	5	6	7	8	9	10
1	2	37.6157	13.5174	-236.1099	0.7500	0.0400	0.0800	0		
2	2	37.6157	13.5174	-236.1099	0.7500	0.0400	0.0800	0.0016		
3	2	38.2818	13.7415	-236.1899	0.7000	-0.0300	0.0800	0.4132		
4	2	38.2818	13.7415	-236.1899	0.7000	-0.0300	0.0800	0.4149		

mat1 | mat2 | mat3 | mat1.vehicle1 | mat1.vehicle2 | mat1.vehicle3 | mat2.vehicle2 | mat3.vehicle2

mat1.vehicle3

	1	2	3	4	5	6	7	8	9	10
1	3	25.6360	15.8968	-233.3099	0.8200	0.8100	0.3600	2.5598		
2	3	25.6360	15.8968	-233.3099	0.8200	0.8100	0.3600	2.5616		
3	3	25.8358	16.1123	-233.4399	1.0100	0.8400	0.1500	2.7863		
4	3	25.8358	16.1123	-233.4399	1.0100	0.8400	0.1500	2.7880		

Figure 45. LCM log start example with Vehicle 2 being the first to broadcast a message.

Table 9. Summary of time delay between send and receive of LCM messages. All logs at 5 Hz update.

Flight	V2V1 (s)	V3V1 (s)	V1V2 (s)	V3V2 (s)	V1V3 (s)	V2V3 (s)	AVG (s)
5 Hz update							
6	0.0691	0.0707	0.125	0.0929	0.0506	0.0687	0.08
8	0.0345	0.0304	0.0221	0.0314	0.0270	0.0442	0.03
9	0.0119	0.0279	0.0173	0.0342	0.0138	0.0132	0.02
10	7.2135	0.0143	0.031	0.0147	0.0389	7.2173	2.42
12	0.8919	0.1414	0.8891	0.1522	0.8787	0.8917	0.64
15	0.3685	0.3035	3.6641	0.3057	4.9146	0.3672	1.65

Vehicle 2, and the difference in timestamps was recorded. Referencing Table 9, the average time delay of messages throughout each log is provided. The header row identifies the sending vehicle ID first, followed by the receiving vehicle ID. Based on every vehicle's values, the overall average time delay for LCM messages is 0.808 s ($\sigma = 1.63$). However, if outlier values over 1 s are removed from flights where log time discrepancies occurred, the average time delay is improved to 0.190 s ($\sigma = 0.27$).

The average packet transmission drop rate was calculated by subtracting the number of matching messages from the receiving vehicle's log and the sending vehicle's log from the total number of messages sent and then dividing by the total number of messages sent. The results from the 5 Hz update rate logs in Table 10 indicate an overall dropped message average of 4.92%. Analyzing the 10 Hz update rate log, significantly fewer of the total messages broadcast from Vehicle 1 and Vehicle 3 arrived at the other vehicles. A deeper analysis of the log files reveals Vehicle 1 timestamps were progressing faster than Vehicle 2 & 3 as was the case in Flights 10 & 15. One example shows a message sent at 219 s from Vehicle 1's log was recorded in the Vehicle 2 log at 240 s. However, a message sent from Vehicle 2 at 173.5 s arrived at 173.67 s in Vehicle 1's log. It is recommended that further tests explore the exact cause and effect of these inconsistencies and if they are also present in simulation.

Table 10. Summary of packet transmission loss percentage between vehicles.

Flight	V1V2 (%)	V1V3 (%)	V2V1 (%)	V2V3 (%)	V3V1 (%)	V3V2 (%)	AVG (%)
10 Hz update							
4	34.9	34.49	2.18	9.47	20.86	22.97	20.81
5 Hz update							
6	0.46	0.46	2.99	2.99	0.18	0.18	1.21
8	34.01	0.14	11.71	11.71	0	35.72	15.55
9	0	0	0.17	0.69	1.02	0.68	0.43
10	0.5	0	5.47	4.81	0	0	1.80
12	0	0.06	0	0.16	0	0.08	0.05
13	0	0	0	0.17	0	0	0.03
15	5.81	5.93	0	1.1	0	0.78	2.27

Lastly, zero values in the table represent 100% message transmission success, which occurred most frequently between Vehicles 1 & 3.

The final communication-related statistics are the velocity vector output rates, or the algorithm processing rate. These messages were broadcast on LCM from Vehicle X on channel 'vXvel' only for logging purposes. Averages per vehicle found in Table 11 show the ability of the BBB to process the algorithm near the desired update rate, u. In the case of $u = 5$ Hz, the overall average algorithm update rate is 4.60 Hz ($\sigma = 0.54$) across all three vehicles. Individual vehicle average rates across all flights range from less than 3 Hz on a flight where Vehicle 2 was unresponsive, to 4.93 Hz on a very successful flight. No data is available for Vehicle 1 for the two test flights when $u = 10$ Hz as it was in 'AUTO,' but Vehicles 2 & 3 averaged 9.42 & 8.22 Hz, respectively. A closer look at the logs for the 10 Hz flights reveals the limitation of

Table 11. Summary of algorithm loop frequency.

u (Hz)	V1 (Hz)	V2 (Hz)	V3 (Hz)	AVG (Hz)
5	4.62	4.46	4.75	4.60
10	-	9.42	8.22	8.82

a 5 Hz GPS update, as duplicate velocity vectors are sent to the autopilot by the algorithm for lack of new vehicle position data.

Overall, the performance of LCM as a means of broadcasting the required data for the algorithm was sufficient, as reflected by the low packet loss rates across a majority of the flights. The two outlier points during flight 8 of Table 10 occurred on Vehicle 2 during a flight in which it was unresponsive to algorithm commands. Due to issues with data collection, the average time delays summarized in Table 9 are subject to additional offsets from log start time variance among vehicles. Although the exact offsets could not be determined, it is not unreasonable to conclude that log start time offsets of ≤ 0.25 s would occur during the optimal performance. Finally, it can be concluded that the BBB is sufficient for processing the algorithm near the specified update rate, but is not capable of fully reaching 5 Hz. Investigating the use of a faster standalone computer or optimizing algorithm performance may increase the update rates.

4.7 Summary

This chapter was dedicated to presenting a qualitative and quantitative analysis of the accumulated data and knowledge gained through all stages of this research to create a testing continuum baseline for future research. Qualitative results from many iterative tests performed via SITL verified the necessity to heavily utilize simulation in early developmental testing of autonomous systems, as the only primary resource required in this environment is time. Through these tests, boundaries for swarm behavior were identified, and insight was gained into safe, testable parameters for real-world swarming. A synopsis of the flight testing timeline and sorties highlighted the many successes and setbacks encountered during the two days, including feedback on the first field use of UgCS for multi-vehicle monitoring. The majority of this chapter

focused on analyzing the data recorded using the test metrics from Chapter III as well as highlighting unique behaviors during specific missions. Overall, separation rule activation appeared successful in most instances of r_{min} violations, while regions of other rule activation were identifiable using vehicle-vehicle and half-median distances. One of the significant flaws of swarm performance found in data processing was the presence of ≈ 0.2 Hz oscillations of individual vehicles. Summary statistics for the testing metrics were provided, although the small sample size limits the power of the results. Chapter V will highlight the significance of these findings, overall algorithm performance, test effectiveness, a summary of lessons learned, and recommendations for future research considerations.

V. Conclusion

5.1 Overview

This section highlights the work accomplished during this research and the successes of developing a testing framework for an autonomous system. The initial research questions are revisited, providing with conclusions for each one. A cumulative reflection of the experience gained by the researcher discusses heuristics, lessons learned, and recommendations for future research involving this algorithm and other autonomous systems.

5.2 Significance of Findings

This research was successful in meeting the overarching objective of verifying an autonomous UAS swarming algorithm through DT&E. A testing framework utilizing modeling, simulation, and flight test provided adequate data to verify the algorithm against the chosen metrics and identify improvements for future research. Simultaneous validation of this framework aimed to address the broader goals and challenges outlined in Chapter I and to provide a reference for future DT&E of autonomous systems. The resulting testing continuum was specific to this algorithm and paired the validated framework with the empirical data from the test program.

Early verification of the existing algorithm used simulation and included various modifications to the rule set and code architecture. This process familiarized the researcher with the autonomous system under investigation and provided insight into the underlying control laws and expected swarm behaviors. Qualitative data gathered during this stage was later used to develop the test plan, incorporating previously used parameters into missions that covered the desired performance envelope. This test

plan was not all inclusive and tended to be conservative in design since this was the first live, open-air flight test of this autonomous swarming algorithm.

The data from flight tests provided a baseline of swarm performance and indicated that the algorithm was able to accurately control individual vehicles and promote swarming behavior in a variety of conditions. It also highlighted the need to test and further refine the algorithm to improve performance and address various anomalies that occurred during testing. This included the need for relative velocity damping of the commanded responses to avoid the oscillations and additional data outputs to the LCM logs.

Due to the limited number of flights and resulting data from flight tests, a definitive set of 'ideal' parameters was not identified. However, a suggested parameter list for stable swarming based on observations throughout this research, and the qualitative and quantitative results is provided as follows: $r_{min} \geq 5$ m, $v_{max} = 3$ m/s, $f_{min} \geq 10$, $b = 0.25$, $u \geq 5$ Hz, $0 < a \geq -1$. Further experimentation is highly recommended to explore the full state space of this autonomous system and improve the validated testing framework.

What improvements, if any, are required to improve the swarming architecture for testing?.

This question was incorporated to address the "design for test" challenge in Ahner and Parson (2016). A significant portion of Chapter III was spent answering this question through the use of simulation and formulation to verify the original algorithm. To summarize, the first significant change was the incorporation of UgCS in the SITL testing environment for easy visualization of swarm behavior. Second, architectural changes to the algorithm were made to group rule calculations at a lower level and aggregate the results in priority order into a single velocity vector at the macro level.

This change also enables a more straightforward implementation of additional rules in the future. A significant change to the rule equations was implemented to ensure proper rule adherence for the swarm. Lastly, the addition of the *argparse* function was incorporated into the code to consolidate versions, enable more straightforward parameter variation during testing and allow for live streaming of algorithm outputs.

What role does quantitative and qualitative assessment have in the verification of autonomous systems?.

Throughout this research, both quantitative and qualitative assessment played critical roles in verifying algorithm behavior. The use of visualization in SITL testing allowed for qualitative analysis during iterative testing of the algorithm. This provided valuable insight into swarm 'goodness' and behavior in a variety of conditions used in designing the flight test plan. Additionally, this analysis technique was used during flight testing to gauge the performance of the swarm and the success of the mission since quantitative data was not immediately available. Many of the lessons learned resulted from a qualitative reflection on the testing process as a whole.

Quantitative analysis was fundamental in verifying individual vehicle and overall swarm performance during flight testing. The ability to collect, analyze and plot data provided a basis for conclusive results that the algorithm performed as intended. Without quantitative analysis, it is impossible to state this same conclusion defensively. However, this analysis also provided insight into regions of poor algorithm performance and led to many suggested improvements for future revisions of this algorithm. Multiple discussions paired qualitative and quantitative results throughout Chapters III & IV to provide a comprehensive analysis and verification of the autonomous swarm.

What measure(s) or metric(s) can be used to evaluate swarm behavior?.

Chapter III highlighted the test objectives, metrics, and required data chosen to evaluate swarm behavior. The chosen metrics included: swarm "goodness," or average deviation from the ideal vehicle-vehicle distance (m), percent violation of Reynolds+ rule set, the communication time delay of LCM messages, and packet loss of LCM messages. Results using flight test data in Chapter IV provide numeric values for these metrics and also discuss additional observations and conclusions related to other measures of swarm behavior. The additional measures included the half-median distance, commanded and actual vehicle velocities, and other combinations of data that were useful in analyzing swarm behavior.

How well do these measure(s) or metric(s) quantify swarm behavior and algorithm performance?.

The chosen metrics identified in Chapter III were sufficient in quantifying swarm behavior and algorithm performance in some instances. However, visualization of the entire 'picture' of swarm performance was not possible with these metrics alone. Other metrics summarized in Table 12 were incorporated into the analysis to supplement the existing four and provide clarity for individual vehicle and swarm behavior.

5.3 Heuristics, Lessons Learned, and Recommendations

Throughout this research effort, many lessons were learned that were not directly associated with answering the Research Objectives and Questions. To exclude this information and fail to make any recommendations would be remiss in the scholarly duties and responsibilities of conducting research. Further, these are perhaps more important than data analysis for future researchers performing developmental test and evaluation, as they provide insight into a methodology and mindset that would

Table 12. Summary of Metrics Used.

Metric	Added?	Comments
Ideal vehicle to vehicle distance	No	Useful for determining tightness of the swarm relative to ideal state, varied depending on parameters
Average percent violation of r_{min}	No	Identified instances where violations occurred, generally $< 10\%$
Half-median Distance	Yes	Correlates to vehicle-vehicle distance, aided in understanding full swarm dynamics
Velocity magnitude vs. Distances	Yes	Inferences on rule activation were made using this data
Commanded to Achieved Velocity Lag	Yes	Showed differences between SITL and flight tests, effects of vehicle momentum
Oscillation Frequency	Yes	Identified need for damping in the algorithm
Communication time delay	No	Relative time delays between data packets, identified time drift in logs
Packet Loss	No	Overall good data transmission between vehicles for algorithm calculations

otherwise not be experienced except through duplicate efforts. The following paragraphs aim to aid future T&E of this and other autonomous systems and provide recommendations for changes in methodology, architecture, algorithm code, and data analysis. A summary version is presented in Table 13.

Heuristics.

Although this research was very focused on a specific autonomous system, relevant heuristics and best practices were learned throughout this process. First and foremost is the reinforcement of testing early and testing often. Using simulation to verify an autonomous systems' functionality before real-world testing is critical in discovering boundary conditions, potential errors, and unexpected behaviors. The rules used in this algorithm were relatively simple compared to other autonomous systems, and

so qualitative analysis and estimation of the swarm's behavior was perhaps sufficient in the early analysis. However, with more complex systems whose behaviors are less intuitive and harder to predict, the importance of pairing qualitative and quantitative data to evaluate performance increases dramatically.

This flight test demonstration provided only a small window into the complexities and processes of a larger scale DT&E program. However, it reinforced the need for testers to be familiar with the systems being used in experimentation. It also emphasized the importance of developers integrating features to support native testing without requiring the development of additional interfaces or tools by testers. This requires early creation of a system-level test plan to verify its functionality beyond just iterative software testing while identifying the required data needed during the analysis stage. Providing these data requirements will permit "designing for test" from the earliest stages of system development.

SITL Recommendations.

Beginning with the testing framework for SITL, many improvements could be made to aid future DOE based experiments and provide immediate results to the researcher. Foremost is identifying if SITL uses an accurate physics-based model of the system under design and if so, how it differs from the actual vehicle. For this research, the basic SITL Arducopter was simulated for the actual X-8 flight tested. If no model is used, using an open-source physics model software such as JSBSim could be pursued. Exploration of the effects of changing autopilot tuning parameters for the vehicle is another area that could be potentially integrated into the algorithm.

Configuring an automated test environment or standardized testing framework is essential to reduce manual inputs for starting and launching simulated vehicles and the algorithm. If properly implemented, these would provide the opportunity for

custom and repetitive vehicle locations before beginning the algorithm and allow for pre-determined parameters to be passed automatically. Additionally, it may be possible to interface with the Terminal on the virtual machines from a Windows or Linux computer which would allow a central control point for starting tests. Developing an integrated analysis framework that automatically performs data collection and processing from SITL tests would increase efficiency during iterative testing. This could be done via MATLAB directly reading and storing the LCM traffic on the network, importing the log files upon completion, or through other software the researcher prefers.

The aim of such a system would be quick testing of minor code changes to verify against previous versions and to accumulate data during extended experimentation. It could be configured to highlight any anomalies during flight, separation violations, or summary statistics of interest to the researcher. As was previously mentioned in Chapter IV, the individual Reynolds+ rules were not broadcast during flight testing. Ensuring required swarm data is broadcast over LCM or recorded via the added 'debug' option is vital when performing data analysis. It is also recommended to include other data in the messages output by the algorithm, such as the autopilot mode, centroid value, current waypoint, and separation distances. Final considerations for the testing framework include the injection of faults or environmental effects to the autopilot or algorithm to observe behavior in degraded states, inserting small LCM message time delays to better represent wireless transmission, adding representative LCM packet loss, and adding more than three simulated vehicles to the swarm.

Flight Test Recommendations.

The first important lesson is to verify that all required output data is present during ground checks before the first test flight. Configuring this output data to fit

a pre-designed analysis method is recommended. Autopilot configuration should be done carefully to avoid oversights of fail-safes and ensure synchronized timestamps. A smaller way-point path for the lead vehicle could allow for multiple iterations on a single battery. Additionally, a v_{max} higher than the leader way-point navigation speed is recommended to allow gap closure for the follower vehicles. At the beginning of each flight, it is recommended that the lead vehicle holds at the first way-point while the follower vehicles attempt to stabilize in the ideal state before beginning the test. After waypoint navigation, the leader should be placed into 'GUIDED' for 30-60 seconds to collect data on fully autonomous swarming behavior near the ideal state. Establishing and following start-up sequences and resetting the autopilot between flights may help ensure proper data logging and management.

The use of newer or different hardware for flight testing should be explored in future efforts. Updating to the Pixhawk 2 or PX4 autopilot is recommended as these offer faster processing and new Global Navigation Satellite System modules which interface with all satellite-based navigation (i.e. GPS, GLONASS). Further integration to include real-time kinematic (RTK) GPS could allow for lower r_{min} values and increased swarming performance due to the 10 Hz update rate in position data relative to the ground node. Changing from BBB to ODROID computing boards for algorithm execution and logging is recommended for two reasons. First, the ODROID has a faster processor and second, it has an onboard battery to maintain current time, unlike the BBB. This may prevent future log files from being duplicated or saved with incorrect file names.

LCM Tests.

Tests to determine the delay in log start times across vehicles are recommended for future research as LCM logger only assigns a relative time accumulated from log

start. This is defined by when the first message is either broadcast, in the case of being the first vehicle placed in autonomous control, or when the first message is received, in the case of the remaining vehicles. A potential solution to determining this time and fixing alignment issues when data processing is to pass a timestamp per message from the Pixhawk or BBB per respective vehicle. However, the time for each of the vehicles would need to be synchronized to a minimum of milliseconds.

Other tests should be conducted to determine a cause for the high outlier values found in Table 9. Closer inspection of Flights 10 & 15 revealed significant delays in the send and receive times between vehicles and showed the sending vehicle's log timestamps progressing at an abnormally higher rate than the receiving vehicles. A cause of this anomaly was inconclusive, as the inconsistent time progression was gradual throughout the logs. Additionally, it was difficult to determine if these inaccurate log times had any effect on vehicle swarming performance or were merely a result of an LCM logger error, bug, or bottlenecks in BBB processing.

Algorithm Modifications.

Aside from adding additional information to the broadcasting LCM messages in the algorithm, new rules could be developed and integrated such as a communication radius as suggested by Allen, home location gravitation, or mission-focused rules as suggested in Chapter II. Experimenting with different broadcast and code loop rates for different information as done in Vásárhelyi et al. (2014) is also recommended. Methods of dealing with degraded swarm states or dropped swarm members such as predictive positioning could increase swarm performance but would require further analysis. Exploration of a swarm damping parameter through the integration of other vehicle's commanded velocities is also strongly recommended to remove the oscillatory behavior observed in all flight tests.

As discussed previously, occasionally vehicles would be unresponsive to algorithm velocity inputs even though proper values were being sent. Analysis of this issue resulted in two suspected causes; pymavlink issues with multiple telemetry connections and Pixhawk version differences on the autopilots. Potential mitigation to this error on the hardware would be to broadcast the vehicle telemetry to UgCS over the mesh network, but this is limited in range and uses additional network bandwidth. Whether this would negatively affect LCM traffic is unknown. As such, the final recommendation is to test mesh network operation and throughput. By arranging node 1 and 3 to be out of range of one another while both are connected to node 2, simple checks could be performed to ensure data from node 1 successfully hops to node 3. Additional bandwidth checks could be performed for future reference if more swarm members are added to the network.

Table 13. Summary of Lessons Learned and Future Recommendations.

SITL and Automated Testing
- Determine if a physics model is used in Ardupilot SITL. Incorporate more accurate models of the X-8 using JSBSim or similar tools. Explore the effects of altering autopilot tuning parameters on swarm behavior. - Automate the testing framework to reduce manual tasks for launching vehicles and algorithm code. - Integrate automated analysis for testing to verify code changes, calculate flight statistics, or other relevant data. - Introduce fault injection, degraded states, environment effects, and time delays in testing. - Increase number of vehicles in the swarm. - Add new variables to broadcast data in LCM messages such as autopilot mode and centroid calculation to better facilitate data analysis and grooming.
Flight Testing
- Verify required output data is present in log files during ground tests. - Smaller way-point path to increase sorties per battery - Higher v_{max} for follower vehicles to allow gap closure to leader. - Hold leader at way-point 1 before beginning tests to allow follower stability. - Have all in 'GUIDED' at conclusion of way-points to collect three-vehicle swarm stability data. - Establish and follow start-up sequences.
Hardware Changes
- Update to Pixhawk 2 or PX4 autopilot. - Incorporate RTK GPS for closer swarming capability. - Change from BBB to ODROID for faster processing and to avoid clock resets during power-offs.
LCM Tests
- Synchronize log start times for more accurate message delay timing. - Determine why some log times progressed at a higher rate.
Algorithm Changes
- Add or explore new rules for testing such as home location gravitation. - Test for optimal parameter values, i.e. b, r_{min}, u. - Alter code loop rates for different information streams. - Incorporate predictive analysis of other swarm members to aid in degraded states. - Introduce swarm damping parameter to reduce oscillations. - Determine exact cause of unresponsive vehicles.
Other
- Broadcast UgCS telemetry via mesh network. - Test mesh network concept for long ranges. - Determine network bandwidth limitations for additional swarm members.

Appendix A. SITL Configuration Guide

VirtualBox Setup

Tuesday, August 22, 2017 4:38 PM

Install VirtualBox on Windows 10
https://www.virtualbox.org/wiki/Downloads
Create new Virtual Machine with hard disk size of >50 GB if possible and all other default values

Install 32/64 bit Ubuntu 16.04 .iso on VM depending on hardware architecture
https://www.ubuntu.com/download/alternative-downloads

When asked to name the machine, name it SITL, and your username should be "sitl"
Make the password whatever you want. Once it boots I find it nice to turn on automatic login to not require a password when logging in
Under System Settings -> Security and Privacy, Password Settings

---

[Installed Synaptic Package Manager (just to update Firefox)

```
sudo apt-get update
sudo apt-get install synaptic]
```

TO INSTALL SITL on Ubuntu
http://ardupilot.org/dev/docs/setting-up-sitl-on-linux.html
-OR-
https://www.ddmckinnon.com/2015/12/30/idiots-guide-to-dronekit-python-a-journey-to-whoz-chillin/

```
$ git clone git://github.com/ArduPilot/ardupilot.git
$ cd ardupilot
$ git submodule update --init --recursive
$ cd Tools/scripts
$ ./install-prereqs-ubuntu.sh
$ echo 'export PATH=$PATH:$HOME/ardupilot/Tools/autotest' >> ~/.bashrc
$ echo 'export PATH=/usr/lib/ccache:$PATH' >> ~/.bashrc
```

****** be sure to use '>>' and not '>' or you will overwrite your bashrc and be in trouble!******

```
$ . ~/.bashrc
```

INSTALL MAVPROXY

```
$ sudo pip install --upgrade pymavlink MAVProxy
```

INSTALL DRONEKIT

```
$ sudo pip install dronekit
$ sudo pip install dronekit-sitl
```

INSTALL JAVA ON UBUNTU - enables lcm-spy use
https://www.digitalocean.com/community/tutorials/how-to-install-java-with-apt-get-on-ubuntu-16-04

```
$ sudo apt-get update
$ sudo apt-get install default-jdk
$ sudo add-apt-repository ppa:webupd8team/java
$ sudo apt-get update
$ sudo apt-get install oracle-java6-installer
```

INSTALL LCM ON UBUNTU
https://lcm-proj.github.io/build_instructions.html
https://github.com/lcm-proj/lcm/releases
download zip from 2nd link above then in a terminal do the following

Sudo dpkg --list | grep -I jdk
Sudo apt-get purge openjdk-*

```
$ sudo apt-get install build-essential
$ sudo apt-get install libglib2.0-dev
$ sudo apt-get python-dev
$ unzip lcm-X.Y.Z.zip
$ cd lcm-X.Y.Z
$ ./configure  (at this point make sure the JAVA line says enabled)
$ make
$ sudo make install
$ sudo ldconfig
```

At this point it's time to do some file copying.

COPYING FILES
https://askubuntu.com/questions/161759/how-to-access-a-shared-folder-in-virtualbox
On Windows, find the folder you wish to share with the VMs and record the folder path *(currently Lt Combs Thesis)*
In your virtual machine go to Machine -> Settings -> Shared Folders
Click the + folder

Navigate to the folder path on the Windows Host Machine
Give it a name, check Auto-mount, and check Make Permanent
Click ok.
Go to Devices -> insert Guest CD additions
Follow the install that pops up.

```
$ sudo usermod -aG vboxsf userName
$ sudo apt-get install virtualbox-guest-dkms
```

Now restart your VM and the folder should show up when you open the file browser.
You can just copy the contents of the folder "COPY TO UBUNTU" under SITL to your home directory.

Once you have all the files copied to the home directory you will need to do a few things.

1. Move the "locations.txt" and "sim_vehicle.py" files to ardupilot/Tools/autotest and replace the existing files.
    a. These files contain these changes
        i. Added custom locations for 3 AFIT Area B test range coordinates
            1) AFIT1=39.7736089,-84.1030123,244,0
            2) AFIT2=39.7736089,-84.1031123,244,0
            3) AFIT3=39.7736089,-84.1032123,244,0
            4) ATTER1=39.344324,-86.009471,220,0
            5) ATTER2=39.344224,-86.009471,220,0
            6) ATTER3=39.344124,-86.009471,220,0

        ii. Add 14552 to the end of the line 617 with 14550, 14551 to automatically add an extra UDP port
2. Then perform the following code in a terminal

```
$ cd exlcm
$ lcm-gen -p idposvel.lcm
$ lcm-gen -p sendvel.lcm
$ lcm-gen -p sendvelcombine.lcm
$ cd
$ sudo route add 224.3.29.71 dev enp0s3 (enp0s3 may be something different, but is found with ifconfig in the terminal)
$ chmod +x rplusbuildjar.sh
$ ./rplusbuildjar.sh
$ source ~/.bashrc
```

(If you are curious, the rplusbuildjar.sh takes care of building the required java components to use lcm-spy to live view the data being passed via lcm. To run this during simulations open a terminal and type lcm-spy. The file also adds classpaths to the .bashrc file so you don't have to.)

TESTING

Now that everything is installed, you want to make sure it works. Open a terminal and do

```
$ chmod +x drone1.sh
$ ./drone1
```

You should see the SITL start up and everything build. It may take a few minutes the first time around.

Open another terminal with "Ctrl+Shift+t" once the SITL is fully running

```
$ python rplusONEvXX.py 1 -debug 8
```

This starts the python code that should connect to the SITL and begin spitting out a ton of data. Excluding the debug part will remove this extra data

Open a third terminal

```
$ lcm-spy
```

This should open a GUI and output a few lines in the terminal. It should have a line that says "Found 2 LCM types." If not you need to reattempt the LCM install.

At this point, you won't see any LCM traffic because the vehicle is not in Guided. Go to the SITL terminal and type "mode guided" and you should see two channels appear in the lcm-spy GUI. One is a self-test and the other is vehicle1. If you see that, everything is working up to this point. There are additional configuration steps under the UgCS setup to have SITL output via TCP so you can view the vehicle in UgCS.

CLONING

Now that you've set up a machine, shut it down.
In VirtualBox, select the first SITL you created (must be powered off), right click, select clone.
Rename it to SITL2 for example and make sure reinitialize the MAC address is checked.
Perform a full clone, and the rest of the regular values, ensure it is >50GB of space.
Do this as many times as you want, if you want 4 SITLs, go ahead, but ensure the computer can support this many!
Once you have cloned everything, start each additional SITL to do some configuration.

Open a terminal and do an "ifconfig" noting your inet addr for each SITL. You'll need this later.
Then

```
$ chmod +x droneX.sh (replace X with whatever vehicle this SITL will be)
$ ./droneX
```

Once the SITL has started completely, type

```
param set SYSID_THISMAV X
```

Where the "X" is again the number associated with the SITL you are in. This is key to give each SITL a unique ID, otherwise UgCS will not be able to tell them apart and you will have issues.
Type
```
param show SYSID_THISMAV
```
To verify your changes were made

At this point you can Ctrl+C to kill the simulators and move on to UgCS setup or you can verify that LCM and the code work from the TESTING section above.

You should now be configured to have 3 virtual machines that can run the code in a SITL environment!!!

UgCS Setup for SITL

Wednesday, August 15, 2018 1:37 PM

This readme assumes you have at least one VM running Ubuntu with Ardupilot SITL installed as well as MAVProxy
It also assumes you have UgCS downloaded (https://www.ugcs.com/en/page/download)
and activated on a Windows 10 Machine, with maps cached for your locations since some of the LAN routers in the lab usually don't have internet access.

Of note UgCS can be finicky and a pain to work with sometimes, but if you follow this you should be okay.

In order for this to work there are several steps to take.

1. Shutdown all virtual machines.
2. Open VirtualBox and open the settings on your Virtual Machine(s).
3. Go to Networking, change type to bridged, select the ethernet adapter as you cannot bridge through a wireless adapter.
4. If using the SITL workstation in the lab, plug into one of the spare network routers lying around with an ethernet cord or the Verizon Hotspot Router.
5. Start your VM(s).
6. Open a terminal (Ctrl+Alt+T) and type "ifconfig"
7. You should have an adapter called enp0s8 or something similar with an IP of 192.168.X.XXX
8. Note this IP for each VM you have
    a. Mine was 192.168.0.190 for SITL1, 192.168.0.120 for SITL2, 192.168.0.173 for SITL3
9. On the Windows machine go to "C:\Program Files (x86)\UgCS\bin" and find the file "vsm-ardupilot.conf"
10. Open this file and find the line titled "# Connection to Ardupilot SITL"
11. Change this section of the file to mirror the below lines
    a. ***Note: your .address line should match the IP of the VM found above

---
```
# Connection to Ardupilot SITL
connection.tcp_out.1.port = 5763
connection.tcp_out.2.port = 5773
connection.tcp_out.3.port = 5783

# Address of the host running SITL instance
connection.tcp_out.1.address = 192.168.0.190
connection.tcp_out.2.address = 192.168.0.139
connection.tcp_out.3.address = 192.168.0.104

# Vehicle inputs
vehicle.ardupilot.tcp.1.address = 192.168.0.190
vehicle.ardupilot.tcp.1.tcp_port = 5763
vehicle.ardupilot.tcp.2.address = 192.168.0.139
vehicle.ardupilot.tcp.2.tcp_port = 5773
vehicle.ardupilot.tcp.3.address = 192.168.0.104
vehicle.ardupilot.tcp.3.tcp_port = 5783
```

Now you can start each VM, open a terminal and run the "droneX" script.
Once it is running, type
For SITL1 - output add tcp:192.168.X.XXX:5762 (This can be found under Windows cmd prompt with ipconfig, use the VirtualBox Host-Only network)
For SITL2 - output add tcp:192.168.X.XXX:5772
- param set SYSID_THISMAV 2

For SITL3 - output add tcp:192.168.X.XXX:5782
- param set SYSID_THISMAV 3

Then terminate the SITL with a Ctrl+C and restart it. This restart is necessary for UgCS to see it for some reason.
Then start UgCS and add a vehicle in the upper right hand corner. You should see APM-1, APM-2, APM-3. Select them and your simulated APM-X vehicles should be right where you want them!!!

***Note: You should be able to plug in the wireless adapter to the workstation once UgCS is connected to the vehicles if you didn't cache the maps the first time around.

vsm-ardupil
ot

This section should be included somewhere in the configuration file for live flight tests. The specific COM Ports will vary depending on the computer and

should be check one at a time through the device manager when they are plugged in. Windows automatically assigns different port numbers to different devices.

*Device manager can be accessed by right clicking the start button, then select device manager. COM ports are where your devices will show up when plugged in.

```
#USE THIS AREA FOR CONNECTING WITH 3DR RADIOS
# Uncomment this to disable serial port access arbitration across
# different processes.
connection.serial.use_arbiter = no
#connection.serial.exclude.1=
# name matching in Windows is case insensitive
# 57600 is default 3DR radio speed
# 115200 is default Ardupilot USB serial rate
connection.serial.1.name = COM18
connection.serial.1.baud = 57600
connection.serial.2.name = COM22
connection.serial.2.baud = 57600
connection.serial.3.name = COM17
connection.serial.3.baud = 57600
```

vsm-ardupil
ot-3DR

Below is the original vsm-ardupilot file to revert to if you mess things up.

vsm-ardupil
ot-original

RE quote
for UgCS ...

Appendix B. Algorithm and Analysis Code Repository

```
clearvars; clc; close all

% for zz = 16:17 % veh1 flight log
%    clearvars -except zz; close all

zz = 6; % flight log index in Vehicle 1 folder, comment for for loop
j = 1;
k = 1;

% Create structures of file name to reference
Veh1Files = dir('F:Analysis Code/Vehicle1/Matrices/*.mat');
Veh2Files = dir('F:Analysis Code/Vehicle2/Matrices/*.mat');
Veh3Files = dir('F:Analysis Code/Vehicle3/Matrices/*.mat');

% Choose to plot traces or not
plotall = 1; % will ignore/execute all plots set to "1"

plottrace = 0; % 0 is no, 1 is yes
plot3Dpath = 0; % plots path in 3D
plotvelvec = 0; % plots quiver of velocity vectors on path plot
plot2Dpath = 0; % plots path in 2D
plotvelvec2D = 0; % plots 2D of veloctiy vectors

plotcent = 0; % plots distance to centroid for each vehicle
plotvvdist = 0; % plots inter-vehicle distances
% plotvvexceed = 0; % plot deviation from ideal v-v distance

plotvelsub = 0; % plots vehicle velocity vs commanded velocity in subplots
plotv1vel = 0; % plots vehicle velocity vs commanded velocity
plotv2vel = 0; % plots vehicle velocity vs commanded velocity
plotv3vel = 0; % plots vehicle velocity vs commanded velocity

plotvelcmdv1 = 0; % plots velocity magnitude versus distances
plotvelcmdv2 = 0; % plots velocity magnitude versus distances
plotvelcmdv3 = 0; % plots velocity magnitude versus distances

plotvelcompv1 = 0; % plots commanded velocity vs velocity in NED components
plotvelcompv2 = 1;
plotvelcompv3 = 0;

% Save figures as PNG or not
saveplot = 0;

% Need to change the below to represent the test scenario
separation = [5 10 10 5 10 5 10 10 5 5 10 10 5 5 5 5 5 5 5 5];
idealvvlist = [9.81 17.41 17.41 9.81 17.41 10.61 18.26 17.41 10.61 10.61...
    18.26 18.26 10.61 10.61 10.61 10.61 10.61 10.61 10.61 10.61];
velocity = [2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3];

lognum = zz; % index of flight parameters
```

```
sepdist = separation(lognum); % value in meters
idealvvd = idealvvlist(lognum); % value in meters
vmax = velocity(lognum);
idealmedian = idealvvd/sqrt(3); % meters

%% Load vehicle files, not required to load all three at this time

mat1_path = fullfile(Veh1Files(lognum).folder,Veh1Files(lognum).name);
mat1_name = Veh1Files(lognum).name;
mat1 = load(mat1_path);
mat1_L = structfun(@numel,mat1);
fprintf('Vehicle 1 File %s\n\n',mat1_name);

% for j=1:length(Veh2Files)
%     if strcmp(Veh2Files(j).name,mat1_name)==1
%         mat2_path = fullfile(Veh2Files(j).folder,Veh2Files(j).name);
%         mat2_name = Veh2Files(j).name;
%     end
% end
% mat2 = load(mat2_path);
% mat2_L = structfun(@numel,mat2);
% fprintf('Vehicle 2 File %s\n\n',mat2_name);
%
% for k=1:length(Veh3Files)
%     if strcmp(Veh3Files(k).name,mat1_name)==1
%         mat3_path = fullfile(Veh3Files(k).folder,Veh3Files(k).name);
%         mat3_name = Veh3Files(k).name;
%     end
% end
%
% mat3 = load(mat3_path);
% mat3_L = structfun(@numel,mat3);
% fprintf('Vehicle 3 File %s\n\n',mat3_name);

%% Order data from matrices into one structure and report update frequency
% passes mat1 structure and returns multiple doubles for all vehicle data

[vehicle1,vehicle2,vehicle3,v1vel,v2vel,v3vel,matcomb,T,f1,f2,f3] = matcondense(mat1);
fprintf('V1vel frequency averaged %.1f Hz\n\n',f1);
fprintf('V2vel frequency averaged %.1f Hz\n\n',f2);
fprintf('V3vel frequency averaged %.1f Hz\n\n',f3);

%% Determine Log Timings

startv1 = min(vehicle1(:,8));
startv2 = min(vehicle2(:,8));
startv3 = min(vehicle3(:,8));
starttime = max([startv1 startv2 startv3]);
endv1 = max(vehicle1(:,8));
endv2 = max(vehicle2(:,8));
endv3 = max(vehicle3(:,8));
```

```
endtime = min([endv1 endv2 endv3]);
logtime = endtime-starttime;
fprintf('Total log time = %.3f seconds \n\n',logtime);

% Trims log start and end times to when all 3 vehicles are engaged
v1s = find(vehicle1(:,8) > starttime, 1 );
v2s = find(vehicle2(:,8) > starttime, 1 );
v3s = find(vehicle3(:,8) > starttime, 1 );

v1L = find(vehicle1(:,8) < endtime, 1, 'last');
v2L = find(vehicle2(:,8) < endtime, 1, 'last');
v3L = find(vehicle3(:,8) < endtime, 1, 'last');

%% Find outliers in Z direction for knowledge purposes and to correct plots
ii=1;
jj=1;
kk=1;

for i=v1s:v1L
    if vehicle1(i,4)>-200
        outliers.v1(i,:) = [vehicle1(i,1:7) i];
        vehicle1(i,4) = vehicle1(i,4)*-1;
        ii=ii+1;
    end
end
for j=v2s:v2L
    if vehicle2(j,4)>-200
        outliers.v2(jj,:) = [vehicle2(j,1:7) j];
        vehicle2(j,4) = vehicle2(j,4)*-1;
        jj=jj+1;
    end
end
for k=v3s:v3L
    if vehicle3(k,4)>-200
        outliers.v3(kk,:) = [vehicle3(k,1:7) k];
        vehicle3(k,4) = vehicle3(k,4)*-1;
        kk=kk+1;
    end
end

%% Calculate estimated centroid based on vehicle positions
% Calculates vehicle to vehicle position also
n=1;
centpt = [];
for i=v1s:v1L
    t1 = vehicle1(i,8);
    p1 = [vehicle1(i,3) vehicle1(i,2) -1*vehicle1(i,4)];

    for ii=v2s:v2L
        if vehicle2(ii,8) > t1 - 0.15 && vehicle2(ii,8) < t1 + 0.15
            t2 = vehicle2(ii,8);
```

```
            p2 = [vehicle2(ii,3) vehicle2(ii,2) -1*vehicle2(ii,4)];
            for iii=v3s:v3L
                if vehicle3(iii,8) > t1 - 0.15 && vehicle3(iii,8) < t1 + 0.15
                    t3 = vehicle3(iii,8);
                    p3 = [vehicle3(iii,2) vehicle3(iii,2) -1*vehicle3(iii,4)];
                    m = 3;
                    centpt = [(p1(1)+p2(1)+p3(1))/m (p1(2)+p2(2)+p3(2))/m (p1(3)...
                        +p2(3)+p3(3))/m (t1+t2+t3)/m];
                    centroid(n,:) = centpt;
                    v1centdist(n,:) = [p1-centpt(1:3) centpt(4)];
                    v2centdist(n,:) = [p2-centpt(1:3) centpt(4)];
                    v3centdist(n,:) = [p3-centpt(1:3) centpt(4)];
                    v1v2dist(n,:) = [p1-p2 centpt(4)];
                    v1v3dist(n,:) = [p1-p3 centpt(4)];
                    v2v3dist(n,:) = [p2-p3 centpt(4)];
                    n = n+1;
                    break
                end
            end
        end
    end
end

%% Determine vehicle distances from the centroid
% Distance in NED is already computed above

halfmedian.v1=(0);
halfmedian.v2=(0);
halfmedian.v3=(0);
for i=1:length(v1centdist)
    halfmedian(i).v1 = sqrt(v1centdist(i,1)^2 + v1centdist(i,2)^2 + v1centdist(i,3)^2);
    halfmedian(i).v2 = sqrt(v2centdist(i,1)^2 + v2centdist(i,2)^2 + v2centdist(i,3)^2);
    halfmedian(i).v3 = sqrt(v3centdist(i,1)^2 + v3centdist(i,2)^2 + v3centdist(i,3)^2);
    halfmedian(i).time = v1centdist(i,4);
end

v1halfmedmean = mean([halfmedian.v1]);
v2halfmedmean = mean([halfmedian.v2]);
v3halfmedmean = mean([halfmedian.v3]);

%% Determine how far over or under ideal halfmedian distance

medexceed.v1=(0);
medexceed.v2=(0);
medexceed.v3=(0);
for i=1:length(v1centdist)
    medexceed(i).v1 = halfmedian(i).v1 - idealmedian;
    medexceed(i).v2 = halfmedian(i).v2 - idealmedian;
    medexceed(i).v3 = halfmedian(i).v3 - idealmedian;
    medexceed(i).time = v1centdist(i,4);
end
```

```
v1exceed = mean([medexceed.v1]);
v2exceed = mean([medexceed.v2]);
v3exceed = mean([medexceed.v3]);

%% Determine vehicle to vehicle distances
% Distance in NED is already computed above

vvd.v1v2=(0);
vvd.v1v3=(0);
vvd.v2v3=(0);
violation = 0;
for i=1:length(v1v2dist)
    vvd(i).v1v2 = sqrt(v1v2dist(i,1)^2 + v1v2dist(i,2)^2 + v1v2dist(i,3)^2);
    vvd(i).v1v3 = sqrt(v1v3dist(i,1)^2 + v1v3dist(i,2)^2 + v1v3dist(i,3)^2);
    vvd(i).v2v3 = sqrt(v2v3dist(i,1)^2 + v2v3dist(i,2)^2 + v2v3dist(i,3)^2);
    vvd(i).time = v1v2dist(i,4);
    if vvd(i).v1v2 < sepdist
        violation = violation +1;
    end
    if vvd(i).v1v3 < sepdist
        violation = violation +1;
    end
    if vvd(i).v2v3 < sepdist
        violation = violation +1;
    end
end
direct_L = max(centroid(:,4));

v1v2mean = mean([vvd.v1v2]);
v1v3mean = mean([vvd.v1v3]);
v2v3mean = mean([vvd.v2v3]);

violationpercent = violation/(3*length(v1v2dist))*100;

%% Determine how far over or under ideal vehicle-vehicle distance

vvdexceed.v1v2=(0);
vvdexceed.v1v3=(0);
vvdexceed.v2v3=(0);
for i=1:length(v1centdist)
    vvdexceed(i).v1v2 = vvd(i).v1v2 - idealvvd;
    vvdexceed(i).v1v3 = vvd(i).v1v3 - idealvvd;
    vvdexceed(i).v2v3 = vvd(i).v2v3 - idealvvd;
    vvdexceed(i).time = v1centdist(i,4);
end

v1v2exceed = mean([vvdexceed.v1v2]);
v1v3exceed = mean([vvdexceed.v1v3]);
v2v3exceed = mean([vvdexceed.v2v3]);
```

```
%% Determine vehicle velocities

for i=v1s:v1L
    vehicle1(i,9) = sqrt(vehicle1(i,5)^2 + vehicle1(i,6)^2 + vehicle1(i,7)^2);
end
for i=v2s:v2L
    vehicle2(i,9) = sqrt(vehicle2(i,5)^2 + vehicle2(i,6)^2 + vehicle2(i,7)^2);
end
for i=v3s:v3L
    vehicle3(i,9) = sqrt(vehicle3(i,5)^2 + vehicle3(i,6)^2 + vehicle3(i,7)^2);
end

v1vm = mean(vehicle1(:,9));
v2vm = mean(vehicle2(:,9));
v3vm = mean(vehicle3(:,9));

v1max = max(vehicle1(:,9));
v2max = max(vehicle2(:,9));
v3max = max(vehicle3(:,9));

v1velmax = max(v1vel(:,18));
v2velmax = max(v2vel(:,18));
v3velmax = max(v3vel(:,18));

%% Print data to screen

fprintf('Mean Velocity (m/s) v1=%.3f, v2=%.3f, v3=%.3f \n\n',v1vm,v2vm,v3vm);
fprintf('Max Velocity (m/s) v1=%.3f, v2=%.3f, v3=%.3f \n\n',v1max,v2max,v3max);
fprintf('Max Commanded Velocity (m/s) v1=%.3f, v2=%.3f, v3=%.3f \n\n',...
    v1velmax,v2velmax,v3velmax);
fprintf('Mean half-median Distance(m) v1=%.3f, v2=%.3f, v3=%.3f\n\n',v1halfmedmean,...
    v2halfmedmean,v3halfmedmean);
fprintf('Mean Inter-vehicle Distance (m) v1v2=%.3f, v1v3=%.3f, v2v3=%.3f\n\n',...
    v1v2mean,v1v3mean,v2v3mean);
fprintf('Mean Inter-vehicle Distance Exceedance (m) v1v2=%.3f, v1v3=%.3f, v2v3=%.
3f\n\n',...
    v1v2exceed,v1v3exceed,v2v3exceed);
fprintf('Overall Minimum Separation Violation Percent= %.2f \n\n',violationpercent);

%% Plotting Data
minz = 220;

% starttime = 150;
% direct_L = 210;

if plotall == 1
    if plot3Dpath == 1
        % Plot positions of all 3 vehicles as lines in 3D
        figure
        plot3(vehicle1(v1s:v1L,3),vehicle1(v1s:v1L,2),-1*vehicle1(v1s:v1L,4),'m')
        hold on
```

```
        plot3(vehicle2(v2s:v2L,3),vehicle2(v2s:v2L,2),-1*vehicle2(v2s:v2L,4),'b-.')
        plot3(vehicle3(v3s:v3L,3),vehicle3(v3s:v3L,2),-1*vehicle3(v3s:v3L,4),'r--')
        legend('veh1','veh2','veh3','centroid')
        axis([-inf inf -inf inf minz inf])
        xlabel('East from Home (m)'); ylabel('North from Home (m)'); zlabel('Up');
        grid on

        % Plot centroid of swarm
        %plot3(centroid(:,1),centroid(:,2),centroid(:,3),'k')
        hold off
        if saveplot == 1
            filename = sprintf('%s-plot3D.png',mat1_name);
            saveas(gcf,filename)
        end
    end

    if plotvelvec == 1
        figure
        % Plot velocity vectors of all 3 vehicles on existing plot
        quiver3(vehicle1(10:10:end,3),vehicle1(10:10:end,2),-1*vehicle1(10:10:end,4),...
            vehicle1(10:10:end,6),vehicle1(10:10:end,5),vehicle1(10:10:end,7),.5,'m')
        hold on
        quiver3(vehicle2(10:10:end,3),vehicle2(10:10:end,2),-1*vehicle2(10:10:end,4),...
            vehicle2(10:10:end,6),vehicle2(10:10:end,5),vehicle2(10:10:end,7),.5,'b-.')
        quiver3(vehicle3(10:10:end,3),vehicle3(10:10:end,2),-1*vehicle3(10:10:end,4),...
            vehicle3(10:10:end,6),vehicle3(10:10:end,5),vehicle3(10:10:end,7),.5,'r--')
        hold off
        if saveplot == 1
            filename = sprintf('%s-3Dvec.png',mat1_name);
            saveas(gcf,filename)
        end
    end

%% Plot 2D positions of all vehicles in NE coords

    if plot2Dpath == 1
        figure
        set(gcf,'Renderer','painters','Position',[100 100 900 900]);
        set(gca,'LooseInset',get(gca,'TightInset'));    % this works better
        set(gca,'FontSize',12);
        plot(vehicle1(v1s:v1L,3),vehicle1(v1s:v1L,2),'m')
        hold on
        plot(vehicle2(v2s:v2L,3),vehicle2(v2s:v2L,2),'b-.')
        plot(vehicle3(v3s:v3L,3),vehicle3(v3s:v3L,2),'r--')

        % Plot details
        legend('veh1','veh2','veh3','centroid')
        xlabel('East from Home (m)'); ylabel('North from Home (m)');
        axis('square')
        grid on
        text(6,32,'1','FontSize',18); text(34,22,'2','FontSize',18)
```

```
        text(9,16,'3','FontSize',18); text(10,-42,'4','FontSize',18)
        text(13,12,'5','FontSize',18); text(37,12,'6','FontSize',18)
        text(37,-12,'7','FontSize',18); text(11,-11,'8','FontSize',18)
        text(7,20,'9','FontSize',18);
        hold off
        if saveplot == 1
            filename = sprintf('%s-plot2D.png',mat1_name);
            saveas(gcf,filename)
        end
    end



%% Plot velocity vectors in 2D NE coords

    if plotvelvec2D == 1
        figure
        set(gcf,'Renderer','painters','Position',[100 100 900 900]);
        set(gca,'LooseInset',get(gca,'TightInset'));    % this works better
        set(gca,'FontSize',12);
        hold on
        % Plot velocity vectors in 2D
        quiver(vehicle1(10:10:end,3),vehicle1(10:10:end,2),...
            vehicle1(10:10:end,6),vehicle1(10:10:end,5),.5,'m')
        quiver(vehicle2(10:10:end,3),vehicle2(10:10:end,2),...
            vehicle2(10:10:end,6),vehicle2(10:10:end,5),'b-.')
        quiver(vehicle3(10:10:end,3),vehicle3(10:10:end,2),...
            vehicle3(10:10:end,6),vehicle3(10:10:end,5),'r--')
        legend('veh1','veh2','veh3','centroid')
        xlabel('East from Home (m)'); ylabel('North from Home (m)');
        axis('square')
        grid on
        text(6,32,'1','FontSize',18); text(34,22,'2','FontSize',18)
        text(9,16,'3','FontSize',18); text(10,-42,'4','FontSize',18)
        text(13,12,'5','FontSize',18); text(37,12,'6','FontSize',18)
        text(37,-12,'7','FontSize',18); text(11,-11,'8','FontSize',18)
        text(7,20,'9','FontSize',18);
        hold off
        if saveplot == 1
            filename = sprintf('%s-2Dvec.png',mat1_name);
            saveas(gcf,filename)
        end
    end

%% Plot distance from centroid of all vehicles

    if plotcent == 1
        figure
        set(gcf,'Renderer','painters','Position',[100 100 900 550]);
        set(gca,'LooseInset',get(gca,'TightInset'));    % this works better
        set(gca,'FontSize',12);
```

```
        plot(v1centdist(:,4),[halfmedian.v1],'m')
        hold on
        plot(v1centdist(:,4),[halfmedian.v2],'b-.')
        plot(v1centdist(:,4),[halfmedian.v3],'r--')
        plot([starttime,direct_L],[idealmedian, idealmedian],'k:')
        legend('veh1','veh2','veh3','Ideal')
        xlabel('Time (s)'); ylabel('Distance (m)');
        xlim([starttime,direct_L]);
        grid on
        hold off
        if saveplot == 1
            filename = sprintf('%s-halfmed.png',mat1_name);
            saveas(gcf,filename)
        end
    end

%% Plot vehicle to vehicle distance

    if plotvvdist == 1
        figure
        set(gcf,'Renderer','painters','Position',[100 100 900 550]);
        set(gca,'LooseInset',get(gca,'TightInset'));    % this works better
        set(gca,'FontSize',12);
        plot(v1v2dist(:,4),[vvd.v1v2],'m')
        hold on
        plot(v1v2dist(:,4),[vvd.v1v3],'b-.')
        plot(v1v2dist(:,4),[vvd.v2v3],'r--')
        plot([starttime, direct_L],[sepdist, sepdist])
        plot([starttime, direct_L],[idealvvd, idealvvd],'k:')
        xlabel('Time (s)'); ylabel('Distance (m)');
        xlim([starttime,direct_L]);
        legend('v1v2','v1v3','v2v3','r_{min}','Ideal')
        grid on
        hold off
        if saveplot == 1
            filename = sprintf('%s-vvd.png',mat1_name);
            saveas(gcf,filename)
        end
    end

%% Plot vehicle-vehicle deviation from ideal

%     if plotvvexceed == 1
%         figure
%         set(gcf,'Renderer','painters','Position',[100 100 900 500]);
%         set(gca,'LooseInset',get(gca,'TightInset'));    % this works better
%         set(gca,'FontSize',12);
%         subplot(3,1,1)
%         hold on
%         plot(v1centdist(:,4),[vvdexceed.v1v2],'m')
%         title(sprintf('Deviation from Ideal Vehicle-Vehicle Distance %s',mat1_name))
```

```
%         legend('v1')
%         ylabel('Distance (m)');
%
%         subplot(3,1,2)
%         plot(v2centdist(:,4),[vvdexceed.v1v3],'b-.')
%         ylabel('Distance (m)');
%         legend('v2')
%
%         subplot(3,1,3)
%         plot(v3centdist(:,4),[vvdexceed.v2v3],'r--')
%         xlabel('Time (s)'); ylabel('Distance (m)');
%         legend('v3')
%         hold off
%     end

%% Animated plot of vehicle points

    if plottrace == 1
        figure
        set(gcf,'Renderer','painters','Position',[100 100 900 900]);
        set(gca,'LooseInset',get(gca,'TightInset'));    % this works better
        set(gca,'FontSize',12);
        xlabel('East'); ylabel('North');
        axis([-10 60 -30 60])
        grid on
        h=animatedline('MaximumNumPoints',12,'LineWidth',1,'Color','blue');
        for i=1:numel(matcomb)
            addpoints(h,matcomb(i).E,matcomb(i).N);
            drawnow
%             frame = getframe(gcf);
%             img = frame2im(frame);
%             [img,cmap] = rgb2ind(img,256);
%             if i == 1
%                 imwrite(img,cmap,'flight15.gif','gif','LoopCount',1,'DelayTime',.005);
%             else
%                 imwrite(img,cmap,'flight15.↙
gif','gif','WriteMode','append','DelayTime',.005);
%             end
            pause(0.005)
        end
    end

%% Plot vehicle velocity versus commanded velocity

    if plotv1vel == 1
        figure
        set(gcf,'Renderer','painters','Position',[100 100 900 500]);
        set(gca,'LooseInset',get(gca,'TightInset'));    % this works better
        set(gca,'FontSize',12);
        if v1vel(1,22)>0
            plot(v1vel(:,22),v1vel(:,18),'k')
```

```
        end
        hold on
        plot([starttime, direct_L],[vmax, vmax],'k-.')
        plot(vehicle1(v1s:v1L,8),vehicle1(v1s:v1L,9),'m--')
        xlabel('Time (s)'); ylabel('Velocity (m/s)');
        xlim([starttime,direct_L]); ylim([0 vmax*1.75])
        if v1vel(1,22)>0
            legend('commanded','v1 velocity','v_{max}')
        else
            legend('actual','v_{max}')
        end
        grid on
        hold off
        if saveplot == 1
            filename = sprintf('%s-velcmd1.png',mat1_name);
            saveas(gcf,filename)
        end
    end

    if plotv2vel == 1
        figure
        set(gcf,'Renderer','painters','Position',[100 100 900 500]);
        set(gca,'LooseInset',get(gca,'TightInset'));    % this works better
        set(gca,'FontSize',12);
        plot(v2vel(:,22),v2vel(:,18),'k')
        hold on
        plot(vehicle2(v2s:v2L,8),vehicle2(v2s:v2L,9),'b--')
        plot([starttime, direct_L],[vmax, vmax],'k-.')
        xlabel('Time (s)'); ylabel('Velocity (m/s)');
        xlim([starttime,direct_L]); ylim([0 vmax*1.75])
        legend('commanded','actual','v_{max}')
        grid on
        hold off
        if saveplot == 1
            filename = sprintf('%s-velcmd2.png',mat1_name);
            saveas(gcf,filename)
        end
    end

    if plotv3vel == 1
        figure
        set(gcf,'Renderer','painters','Position',[100 100 900 500]);
        set(gca,'LooseInset',get(gca,'TightInset'));    % this works better
        set(gca,'FontSize',12);
        plot(v3vel(:,22),v3vel(:,18),'k')
        hold on
        plot(vehicle3(v3s:v3L,8),vehicle3(v3s:v3L,9),'r--')
        plot([starttime, direct_L],[vmax, vmax],'k-.')
        xlabel('Time (s)'); ylabel('Velocity (m/s)');
        xlim([starttime,direct_L]); ylim([0 vmax*1.75])
        legend('commanded','actual','v_{max}')
```

```
        grid on
        hold off
        if saveplot == 1
            filename = sprintf('%s-velcmd3.png',mat1_name);
            saveas(gcf,filename)
        end
    end

    if plotvelsub == 1
        figure
        set(gcf,'Renderer','painters','Position',[100 100 900 550]);

        ha = tight_subplot(3,1,[.025 .01],[.1 .05],[.06 .04]);
        axes(ha(1))
        if v1vel(1,22)>0
            plot(v1vel(:,22),v1vel(:,18),'k')
        end
        hold on
        plot([starttime, direct_L],[vmax, vmax],'k-.')
        plot(vehicle1(v1s:v1L,8),vehicle1(v1s:v1L,9),'m--')
        ylabel('Velocity (m/s)'); ylim([0 vmax*1.75]); xlim([starttime,direct_L]);
        if v1vel(1,22)>0
            legend('commanded','v1 velocity','v_{max}')
        else
            legend('actual','v_{max}')
        end
        grid on
        hold off

        axes(ha(2))
        plot(v2vel(:,22),v2vel(:,18),'k')
        hold on
        plot(vehicle2(v2s:v2L,8),vehicle2(v2s:v2L,9),'b--')
        plot([starttime, direct_L],[vmax, vmax],'k-.')
        ylabel('Velocity (m/s)'); ylim([0 vmax*1.75]); xlim([starttime,direct_L]);
        legend('commanded','v2 velocity','v_{max}')
        grid on
        hold off

        axes(ha(3))
        plot(v3vel(:,22),v3vel(:,18),'k')
        hold on
        plot(vehicle3(v3s:v3L,8),vehicle3(v3s:v3L,9),'r--')
        plot([starttime, direct_L],[vmax, vmax],'k-.')
        xlabel('Time (s)'); ylabel('Velocity (m/s)');
        ylim([0 vmax*1.75]); xlim([starttime,direct_L]);
        legend('commanded','v3 velocity','v_{max}')
        set(ha(1:2),'XTickLabel','');
        grid on
        hold off
```

```
        if saveplot == 1
            filename = sprintf('%s-velcmd.png',mat1_name);
            saveas(gcf,filename)
        end
    end

%% Plot vehicle velocity versus distances to center and inter-vehicle dist

    if plotvelcmdv1 == 1
        figure
        set(gcf,'Renderer','painters','Position',[100 100 900 550]);

        ha = tight_subplot(3,1,[.025 .01],[.1 .05],[.06 .04]);
        axes(ha(1))
        plot(v1centdist(:,4),[vvd.v1v2],'m--');
        hold on
        plot(v1centdist(:,4),[vvd.v1v3],'b-.');
        plot([starttime, direct_L],[sepdist, sepdist]);
        plot([starttime, direct_L],[idealvvd, idealvvd],'k-.');
        ylabel('Distance (m)'); xlim([starttime,direct_L]); ylim([0 idealvvd*4]);
        legend('v1v2 distance','v1v3 distance','r_{min}','Ideal')
        grid on
        hold off

        axes(ha(2))
        plot(v1vel(:,22),v1vel(:,18),'k');
        ylabel('Velocity (m/s)');
        ylim([0 1.2*vmax]); xlim([starttime,direct_L]);
        legend('Commanded Velocity')
        grid on

        axes(ha(3))
        plot(v1centdist(:,4),[halfmedian.v1],'r--');
        hold on
        plot([starttime,direct_L],[idealmedian, idealmedian],'k-.')
        xlabel('Time (s)'); ylabel('Distance (m)');
        xlim([starttime,direct_L]); ylim([0 idealmedian*4]);
        legend('half-median distance','Ideal')
        set(ha(1:2),'XTickLabel','');
        grid on
        hold off

        if saveplot == 1
            filename = sprintf('%s-cmdvsd1.png',mat1_name);
            saveas(gcf,filename)
        end
    end

    if plotvelcmdv2 == 1
        figure
        set(gcf,'Renderer','painters','Position',[100 100 900 700]);
```

```
        ha = tight_subplot(4,1,[.025 .01],[.1 .05],[.06 .04]);
        axes(ha(1))
        plot(v2centdist(:,4),[vvd.v1v2],'m--');
        hold on
        plot(v2centdist(:,4),[vvd.v2v3],'b-.');
        plot([starttime, direct_L],[sepdist, sepdist]);
        plot([starttime, direct_L],[idealvvd, idealvvd],'k-.');
        ylabel('Distance (m)'); xlim([starttime,direct_L]); ylim([0 idealvvd*4]);
        legend('v2v1 distance','v2v3 distance','r_{min}','Ideal')
        grid on
        hold off

        axes(ha(2))
        plot(v2vel(:,22),v2vel(:,18),'k');
        ylabel('Velocity (m/s)');
        ylim([0 1.2*vmax]); xlim([starttime,direct_L]);
        legend('Commanded Velocity')
        grid on

        axes(ha(3))
        plot(v2centdist(:,4),[halfmedian.v2],'r--');
        hold on
        plot([starttime,direct_L],[idealmedian, idealmedian],'k-.')
        xlabel('Time (s)'); ylabel('Distance (m)');
        xlim([starttime,direct_L]); ylim([0 idealmedian*4]);
        legend('half-median distance','Ideal')
        set(ha(1:2),'XTickLabel','');
        grid on
        hold off

        axes(ha(4))
        plot(vehicle2(:,8),-1*vehicle2(:,4)-223,'b--');
        hold on
        plot([starttime,direct_L],[10, 10],'k-.')
        xlabel('Time (s)'); ylabel('Distance (m)');
        xlim([starttime,direct_L]); ylim([5 20]);
        legend('Altitude','r_{min}')
        set(ha(1:3),'XTickLabel','');
        grid on
        hold off

        if saveplot == 1
            filename = sprintf('%s-cmdvsd2.png',mat1_name);
            saveas(gcf,filename)
        end
    end

    if plotvelcmdv3 == 1
        figure
        set(gcf,'Renderer','painters','Position',[100 100 900 550]);
```

```
        ha = tight_subplot(3,1,[.025 .01],[.1 .05],[.06 .04]);
        axes(ha(1))
        plot(v3centdist(:,4),[vvd.v1v3],'m--');
        hold on
        plot(v3centdist(:,4),[vvd.v2v3],'b-.');
        plot([starttime, direct_L],[sepdist, sepdist]);
        plot([starttime, direct_L],[idealvvd, idealvvd],'k-.');
        ylabel('Distance (m)'); xlim([starttime,direct_L]); ylim([0 idealvvd*4]);
        legend('v3v1 distance','v3v2 distance','r_{min}','Ideal')
        grid on
        hold off

        axes(ha(2))
        plot(v3vel(:,22),v3vel(:,18),'k');
        ylabel('Velocity (m/s)');
        ylim([0 1.2*vmax]); xlim([starttime,direct_L]);
        legend('Commanded Velocity')
        grid on

        axes(ha(3))
        plot(v3centdist(:,4),[halfmedian.v3],'r--');
        hold on
        plot([starttime,direct_L],[idealmedian, idealmedian],'k-.')
        xlabel('Time (s)'); ylabel('Distance (m)');
        xlim([starttime,direct_L]); ylim([0 idealmedian*4]);
        legend('half-median distance','Ideal')
        set(ha(1:2),'XTickLabel','');
        grid on
        hold off


        if saveplot == 1
            filename = sprintf('%s-cmdvsd3.png',mat1_name);
            saveas(gcf,filename)
        end
    end


%% Plot velocity vs velcmd in NED components
    if plotvelcompv1 == 1
        figure
        set(gcf,'Renderer','painters','Position',[100 100 900 550]);

        ha = tight_subplot(3,1,[.025 .01],[.1 .05],[.06 .04]);
        axes(ha(1))
        plot(v1vel(:,22),v1vel(:,20),'k');
        hold on
        plot(vehicle1(v1s:v1L,8),vehicle1(v1s:v1L,6),'m-.')
        ylabel('Velocity (m)'); xlim([starttime,direct_L]); ylim([-vmax*1.75 vmax*1.75]);
        legend('commanded vel N','v1 vel N')
```

```
        grid on
        hold off

        axes(ha(2))
        plot(v1vel(:,22),v1vel(:,19),'k');
        hold on
        plot(vehicle1(v1s:v1L,8),vehicle1(v1s:v1L,5),'m-.')
        ylabel('Velocity (m)'); xlim([starttime,direct_L]); ylim([-vmax*1.75 vmax*1.75]);
        legend('commanded vel E','v1 vel E')
        grid on
        hold off

        axes(ha(3))
        plot(v1vel(:,22),v1vel(:,21),'k');
        hold on
        plot(vehicle1(v1s:v1L,8),vehicle1(v1s:v1L,7),'m-.')
        xlabel('Time (s)'); ylabel('Velocity (m)');
        xlim([starttime,direct_L]); ylim([-vmax*1.75 vmax*1.75]);
        legend('commanded vel D','v1 vel D')
        set(ha(1:2),'XTickLabel','');
        grid on
        hold off

        if saveplot == 1
            filename = sprintf('%s-cmdvsvel1.png',mat1_name);
            saveas(gcf,filename)
        end
    end

    if plotvelcompv2 == 1
        figure
        set(gcf,'Renderer','painters','Position',[100 100 900 550]);

        ha = tight_subplot(3,1,[.025 .01],[.1 .05],[.06 .04]);
        axes(ha(1))
        plot(v2vel(:,22),v2vel(:,20),'k');
        hold on
        plot(vehicle2(v2s:v2L,8),vehicle2(v2s:v2L,6),'b-.')
        ylabel('Velocity (m)'); xlim([starttime,direct_L]); ylim([-vmax*1.75 vmax*1.75]);
        legend('commanded vel N','v2 vel N')
        grid on
        hold off

        axes(ha(2))
        plot(v2vel(:,22),v2vel(:,19),'k');
        hold on
        plot(vehicle2(v2s:v2L,8),vehicle2(v2s:v2L,5),'b-.')
        ylabel('Velocity (m)'); xlim([starttime,direct_L]); ylim([-vmax*1.75 vmax*1.75]);
        legend('commanded vel E','v2 vel E')
        grid on
        hold off
```

```
        axes(ha(3))
        plot(v2vel(:,22),v2vel(:,21),'k');
        hold on
        plot(vehicle2(v2s:v2L,8),vehicle2(v2s:v2L,7),'b-.')
        xlabel('Time (s)'); ylabel('Velocity (m)');
        xlim([starttime,direct_L]); ylim([-vmax*1.75 vmax*1.75]);
        legend('commanded vel D','v2 vel D')
        set(ha(1:2),'XTickLabel','');
        grid on
        hold off

        if saveplot == 1
            filename = sprintf('%s-cmdvsvel2.png',mat1_name);
            saveas(gcf,filename)
        end
    end

    if plotvelcompv3 == 1
        figure
        set(gcf,'Renderer','painters','Position',[100 100 900 550]);

        ha = tight_subplot(3,1,[.025 .01],[.1 .05],[.06 .04]);
        axes(ha(1))
        plot(v3vel(:,22),v3vel(:,20),'k');
        hold on
        plot(vehicle3(v3s:v3L,8),vehicle3(v3s:v3L,6),'r-.')
        ylabel('Velocity (m)'); xlim([starttime,direct_L]); ylim([-vmax*1.75 vmax*1.75]);
        legend('commanded vel N','v3 vel N')
        grid on
        hold off

        axes(ha(2))
        plot(v3vel(:,22),v3vel(:,19),'k');
        hold on
        plot(vehicle3(v3s:v3L,8),vehicle3(v3s:v3L,5),'r-.')
        ylabel('Velocity (m)'); xlim([starttime,direct_L]); ylim([-vmax*1.75 vmax*1.75]);
        legend('commanded vel E','v3 vel E')
        grid on
        hold off

        axes(ha(3))
        plot(v3vel(:,22),v3vel(:,21),'k');
        hold on
        plot(vehicle3(v3s:v3L,8),vehicle3(v3s:v3L,7),'r-.')
        xlabel('Time (s)'); ylabel('Velocity (m)');
        xlim([starttime,direct_L]); ylim([-vmax*1.75 vmax*1.75]);
        legend('commanded vel D','v3 vel D')
        set(ha(1:2),'XTickLabel','');
        grid on
        hold off
```

```
        if saveplot == 1
            filename = sprintf('%s-cmdvsvel3.png',mat1_name);
            saveas(gcf,filename)
        end
    end
end
% end
%% Printing data to file

% fileID = fopen('F:Analysis Code/testdata.txt','a');
% fprintf(fileID,'%s %.3f %.3f %.3f %.3f %.3f %.3f %.3f %.3f %.3f %.3f %.3f %.3f %.↙
3f %.3f %.3f %.3f %.3f %.3f\r\n',...
%     mat1_name,logtime,v1vm,v2vm,v3vm,v1max,v2max,v3max,v1velmax,v2velmax,...
%     v3velmax,f1,f2,f3,v1halfmedmean,v2halfmedmean,v3halfmedmean,v1v2mean,...
%     v1v3mean,v2v3mean);

% fileID = fopen('F:Analysis Code/violationpercent.txt','a');
% fprintf(fileID,'%s %.3f\r\n',...
%     mat1_name,violationpercent);
```

```
function [v1,v2,v3,v1vel,v2vel,v3vel,matcomb,T,f1,f2,f3...
    ] = matcondense( m1 )
% matcondense is meant to condense 3 vehicle matrices to 1
    % It pulls each matrix loaded in the script and sequentially stacks
    % them on the end of each other based on their lengths.
    % It is then converted to a table, sorted by the LCM timestamp, and
    % passed back to the script as matcomb structure

if isfield(m1,'vehicle1')
    v1 = m1.vehicle1;
else
    v1 = 0;
end
if isfield(m1,'vehicle2')
    v2 = m1.vehicle2;
else
    v2 = 0;
end
if isfield(m1,'vehicle3')
    v3 = m1.vehicle3;
else
    v3 = 0;
end
if isfield(m1,'v1vel')
    v1vel = m1.v1vel;
else
    v1vel = zeros(1,22);
end
if isfield(m1,'v2vel')
    v2vel = m1.v2vel;
else
    v2vel = zeros(1,22);
end
if isfield(m1,'v3vel')
    v3vel = m1.v3vel;
else
    v3vel = zeros(1,22);
end

j = 1;

matcomb = struct('ID',{0},'N',{0},'E',{0},'D',{0},'Vn',{0},'Ve',{0},...
    'Vd',{0},'time',{0});

v1L = length(v1);
v2L = length(v2);
v3L = length(v3);

for i=1:v1L
    matcomb.ID(j,1) = m1.vehicle1(i,1);
    matcomb.N(j,1)  = m1.vehicle1(i,2);
```

```
    matcomb.E(j,1)  = m1.vehicle1(i,3);
    matcomb.D(j,1)  = -1*m1.vehicle1(i,4);
    matcomb.Vn(j,1) = m1.vehicle1(i,5);
    matcomb.Ve(j,1) = m1.vehicle1(i,6);
    matcomb.Vd(j,1) = m1.vehicle1(i,7);
    matcomb.time(j,1) = m1.vehicle1(i,8);
    j=j+1;
end

for i=1:v2L
    matcomb.ID(j,1) = m1.vehicle2(i,1);
    matcomb.N(j,1)  = m1.vehicle2(i,2);
    matcomb.E(j,1)  = m1.vehicle2(i,3);
    matcomb.D(j,1)  = -1*m1.vehicle2(i,4);
    matcomb.Vn(j,1) = m1.vehicle2(i,5);
    matcomb.Ve(j,1) = m1.vehicle2(i,6);
    matcomb.Vd(j,1) = m1.vehicle2(i,7);
    matcomb.time(j,1) = m1.vehicle2(i,8);
    j=j+1;
end

for i=1:v3L
    matcomb.ID(j,1) = m1.vehicle3(i,1);
    matcomb.N(j,1)  = m1.vehicle3(i,2);
    matcomb.E(j,1)  = m1.vehicle3(i,3);
    matcomb.D(j,1)  = -1*m1.vehicle3(i,4);
    matcomb.Vn(j,1) = m1.vehicle3(i,5);
    matcomb.Ve(j,1) = m1.vehicle3(i,6);
    matcomb.Vd(j,1) = m1.vehicle3(i,7);
    matcomb.time(j,1) = m1.vehicle3(i,8);
    j=j+1;
end

T = struct2table(matcomb);
T = sortrows(T,'time');
matcomb = table2struct(T);


%% frequency calculations for sent velocity

if isfield(m1,'v1vel')
    v1size = size(m1.v1vel);
    for i=1:v1size(1)
        sendvel1.time(i,1) = m1.v1vel(i,22);
    end
    F1 = struct2table(sendvel1);
    for i=1:v1size-1
        t1 = F1.time(i);
        t2 = F1.time(i+1);
        td = t2-t1;
        Td(i) = td;
```

```
    end
    tdavg = mean(Td);
    f1 = round(1/tdavg,3);
else
    f1= 0;
end

if isfield(m1,'v2vel')
    v2size = size(m1.v2vel);
    for i=1:v2size(1)
        sendvel2.time(i,1) = m1.v2vel(i,22);
    end
    F2 = struct2table(sendvel2);
    for i=1:v2size-1
        t1 = F2.time(i);
        t2 = F2.time(i+1);
        td = t2-t1;
        Td(i) = td;
    end
    tdavg = mean(Td);
    f2 = round(1/tdavg,3);
else
    f2 = 0;
end

if isfield(m1,'v3vel')
    v3size = size(m1.v3vel);
    for i=1:v3size(1)
        sendvel3.time(i,1) = m1.v3vel(i,22);
    end
    F3 = struct2table(sendvel3);
    for i=1:v3size-1
        t1 = F3.time(i);
        t2 = F3.time(i+1);
        td = t2-t1;
        Td(i) = td;
    end
    tdavg = mean(Td);
    f3 = round(1/tdavg,3);
else
    f3 = 0;
end
```

```
clearvars; clc; close all

% for z = 1:8
%     nums = [4 6 8 9 10 12 13 15];
% i = nums(z); % flight log index in Vehicle 1 folder, comment for for loop

i = 16;

% Create structures of file name to reference
Veh1Files = dir('F:Analysis Code/Vehicle1/Matrices/*.mat');
Veh2Files = dir('F:Analysis Code/Vehicle2/Matrices/*.mat');
Veh3Files = dir('F:Analysis Code/Vehicle3/Matrices/*.mat');

%% Load vehicle files, not required to load all three at this time

mat1_path = fullfile(Veh1Files(i).folder,Veh1Files(i).name);
mat1_name = Veh1Files(i).name;
mat1 = load(mat1_path);
mat1_L = structfun(@numel,mat1);
fprintf('Vehicle 1 File %s\n\n',mat1_name);

for j=1:length(Veh2Files)
    if strcmp(Veh2Files(j).name,mat1_name)==1
        mat2_path = fullfile(Veh2Files(j).folder,Veh2Files(j).name);
        mat2_name = Veh2Files(j).name;
    end
end
mat2 = load(mat2_path);
mat2_L = structfun(@numel,mat2);
fprintf('Vehicle 2 File %s\n\n',mat2_name);

for k=1:length(Veh3Files)
    if strcmp(Veh3Files(k).name,mat1_name)==1
        mat3_path = fullfile(Veh3Files(k).folder,Veh3Files(k).name);
        mat3_name = Veh3Files(k).name;
    end
end

mat3 = load(mat3_path);
mat3_L = structfun(@numel,mat3);
fprintf('Vehicle 3 File %s\n\n',mat3_name);

%% Order data from matrices into one structure and report update frequency
% passes mat1 structure and returns multiple doubles for all vehicle data

[m1v1,m1v2,m1v3,m1v1vel,m1v2vel,m1v3vel,m1matcomb,m1T,m1f1,m1f2,m1f3] = matcondense
(mat1);
[m2v1,m2v2,m2v3,m2v1vel,m2v2vel,m2v3vel,m2matcomb,m2T,m2f1,m2f2,m2f3] = matcondense
(mat2);
[m3v1,m3v2,m3v3,m3v1vel,m3v2vel,m3v3vel,m3matcomb,m3T,m3f1,m3f2,m3f3] = matcondense
(mat3);
```

```
%% Determine Log Timings

v1L = min([length(m1v1) length(m2v1) length(m3v1)]);
v2L = min([length(m1v2) length(m2v2) length(m3v2)]);
v3L = min([length(m1v3) length(m2v3) length(m3v3)]);

v1E = max([length(m1v1) length(m2v1) length(m3v1)]);
v2E = max([length(m1v2) length(m2v2) length(m3v2)]);
v3E = max([length(m1v3) length(m2v3) length(m3v3)]);

%% Compare v1 broadcast to v2&v3 receive

C1 = find(ismember(m2v1(:,1:7),m1v1(:,1:7),'rows'));
C2 = find(ismember(m3v1(:,1:7),m1v1(:,1:7),'rows'));
C3 = find(ismember(m1v2(:,1:7),m2v2(:,1:7),'rows'));
C4 = find(ismember(m3v2(:,1:7),m2v2(:,1:7),'rows'));
C5 = find(ismember(m1v3(:,1:7),m3v3(:,1:7),'rows'));
C6 = find(ismember(m2v3(:,1:7),m3v3(:,1:7),'rows'));

P1 = length(C1)/v1E*100;
P2 = length(C2)/v1E*100;
P3 = length(C3)/v2E*100;
P4 = length(C4)/v2E*100;
P5 = length(C5)/v3E*100;
P6 = length(C6)/v3E*100;
fprintf('%.1f %.1f \n%.1f %.1f \n%.1f %.1f \n\n',P1,P2,P3,P4,P5,P6);

v1TD1=0;
v1TD2=0;
try
for ii = 1:v1L
    comp1 = m1v1(ii,1:7);
    for iii = ii:ii+100
        A = intersect(comp1,m2v1(iii,1:7));
        diff1 = m2v1(iii,8) - m1v1(ii,8);
        if length(A)==7 && diff1 >= 0
            v1TD1(ii,1) = diff1;
            v1TD1(ii,2:4) = comp1(2:4);
            v1TD1(ii,5:7) = comp1(5:7);
            break
        end
    end
    for iii = ii:ii+100
        B = intersect(comp1,m3v1(iii,1:7));
        diff2 = m3v1(iii,8) - m1v1(ii,8);
        if length(B)==7 && diff2 >= 0
            v1TD2(ii,1) = diff2;
            v1TD2(ii,2:4) = comp1(2:4);
            v1TD2(ii,5:7) = comp1(5:7);
            break
```

```
        end
    end
end
end
v1TD1AVG = mean(v1TD1(:,1));
v1TD2AVG = mean(v1TD2(:,1));

fprintf('Avg Time Delay V1V2=%.4f V1V3=%.4f \n\n',v1TD1AVG,v1TD2AVG);

v2TD1=0;
v2TD2=0;
try
for ii = 1:v2L
    comp1 = m2v2(ii,1:7);
    for iii = ii:ii+100
        A = intersect(comp1,m1v2(iii,1:7));
        diff1 = m1v2(iii,8) - m2v2(ii,8);
        if length(A)==7 && diff1 >= 0
            v2TD1(ii,1) = diff1;
            v2TD1(ii,2:4) = comp1(2:4);
            v2TD1(ii,5:7) = comp1(5:7);
            break
        end
    end
    for iii = ii:ii+100
        B = intersect(comp1,m3v2(iii,1:7));
        diff2 = m3v2(iii,8) - m2v2(ii,8);
        if length(B)==7 && diff2 >= 0
            v2TD2(ii,1) = diff2;
            v2TD2(ii,2:4) = comp1(2:4);
            v2TD2(ii,5:7) = comp1(5:7);
            break
        end
    end
end
end
v2TD1AVG = mean(v2TD1(:,1));
v2TD2AVG = mean(v2TD2(:,1));

fprintf('Avg Time Delay V2V1=%.4f V2V3=%.4f \n\n',v2TD1AVG,v2TD2AVG);

v3TD1=0;
v3TD2=0;
try
for ii = 1:v3L
    comp1 = m3v3(ii,1:7);
    for iii = ii:ii+100
        A = intersect(comp1,m1v3(iii,1:7));
        diff1 = m1v3(iii,8) - m3v3(ii,8);
        if length(A)==7 && diff1 >= 0
            v3TD1(ii,1) = diff1;
```

```
            v3TD1(ii,2:4) = comp1(2:4);
            v3TD1(ii,5:7) = comp1(5:7);
            break
        end
    end
    for iii = ii:ii+100
        B = intersect(comp1,m2v3(iii,1:7));
        diff2 = m2v3(iii,8) - m3v3(ii,8);
        if length(B)==7 && diff2 >= 0
            v3TD2(ii,1) = diff2;
            v3TD2(ii,2:4) = comp1(2:4);
            v3TD2(ii,5:7) = comp1(5:7);
            break
        end
    end
end
end
v3TD1AVG = mean(v3TD1(:,1));
v3TD2AVG = mean(v3TD2(:,1));

fprintf('Avg Time Delay V3V1=%.4f V3V2=%.4f \n\n',v3TD1AVG,v3TD2AVG);

fileID = fopen('Delay.txt','a');
fprintf(fileID,'%s & %.2f & %.2f & %.2f & %.2f & %.2f & %.2f %.4f %.4f %.4f %.4f %.4f %.↙
4f\r\n',...
    mat1_name,P1,P2,P3,P4,P5,P6,v1TD1AVG,v1TD2AVG,v2TD1AVG,v2TD2AVG,v3TD1AVG,v3TD2AVG);

%end
% figure
% plot(m1v1(:,3),m1v1(:,2),'m')
% hold on
% plot(m2v1(:,3),m2v1(:,2),'b-.')
% plot(m3v1(:,3),m3v1(:,2),'r--')
% % Plot details
% axis('square')
% grid on
% hold off
%
% figure
% plot(m1v2(:,3),m1v2(:,2),'m')
% hold on
% plot(m2v2(:,3),m2v2(:,2),'b-.')
% plot(m3v2(:,3),m3v2(:,2),'r--')
% % Plot details
% axis('square')
% grid on
% hold off
%
% figure
% plot(m1v3(:,3),m1v3(:,2),'m')
% hold on
```

```
% plot(m2v3(:,3),m2v3(:,2),'b-.')
% plot(m3v3(:,3),m3v3(:,2),'r--')
% % Plot details
% axis('square')
% grid on
% hold off
```

Appendix C. Flight Testing 5028 for TRB/SRB

TEST PROJECT TECHNICAL AND SAFETY REVIEW

SECTION I PROJECT INFORMATION

Test Project Title	Overall Risk Level	Control #	Test Dept
Multiple Autonomous UAV Swarm – Mesh Ad-hoc Network	LOW		ENV

Typed Name and Grade	Signature	Email Address	Phn Number	Date
Principal Investigator				
Dr. John Colombi		john.colombi@afit.edu	X3347	
Alt Principal Investigator				
Dr. David Jacques		david.jacques@afit.edu	X3329	
Project Safety Lead/Test Team Member				
Mr. Jeremy Gray		Jeremy.gray@afit.edu	X4278	
Safety Pilot				
Mr. Rick Patton		rick.patton.ctr@afit.edu	X4911	
Safety Pilot 2				
Mr. Dave Thacker			X4911	

SECTION II TECHNICAL/SAFETY REVIEW

Typed Name and Rank	Position	Signature	Date	Coord Comments? Yes	No
AFIT Flight Test Safety Officer					
Lt Col Amy Cox	AFIT FTSO				
	AFIT/EN				
	AFIT/EN				

SECTION III COORDINATING COMMENTS

(Reviewer should initial next to any comments)

PROJECT DESCRIPTION

1. SUMMARY OF CHANGES
    a. This is a new set of test objectives to be flown on a proven vehicle. The configuration changes associated with the X-8 multi-rotor vehicles are associated with a Beagle Bone black microcomputer, Wi-Fi adapter, and auxiliary battery mounted on each of the X-8 vehicles. The vehicles will be flown in a pair, then as a trio governed by a swarming algorithm processed on board each vehicle.

2. BACKGROUND
    a. Description:
    These tests are to assess the functionality of an open source swarming architecture. The tests will utilize three 3D Robotics X-8 multi-rotor equipped with autopilots, GPS receivers, Wi-Fi adapters, and on-board microprocessors to perform autonomous swarm movement. The vehicles fall under the small Unmanned Aerial System (UAS) Group 1 category, and are covered under the AFLCMC Military Flight Release (MFR) R0013 dated 17 October 2016. These vehicles are monitored from a ground control station (GCS), which is composed of a laptop with an autopilot comm/telemetry modem. The ground station can control the vehicles but during in-flight testing, an on-board script will control vehicle movement based on the positions and velocities of the vehicle(s) around it.

    Test Objective 0 involves one X-8 multi-rotor equipped with a Beagle Bone black microcomputer, Wi-Fi adapter, and auxiliary battery. The vehicle will be controlled by the on-board processor and script and by ground radio. The flight will cover an area of approximately 150 meters x 150 meters over the runway and cross road from an altitude of 15-50 meters.

    Test Objective 1 involves two X-8 multi-rotors equipped with a Beagle Bone black microcomputer, Wi-Fi adapter, and auxiliary battery. One vehicle will be controlled by the on-board processor and script, while the other will be directly controlled by ground radio, although it will broadcast the same data as the script-controlled vehicle. The flight will cover an area of approximately 150 meters x 150 meters over the runway and cross road from an altitude of 15-50 meters.

    Test Objective 2 involves three X-8 multi-rotors equipped with a Beagle Bone black microcomputer, Wi-Fi adapter, and auxiliary battery. Two vehicles will be controlled by the on-board processor and script, while the third will be directly controlled by ground radio, although it will broadcast the same data as the script-controlled vehicles. As in Objective 1, flight will cover an area of approximately 150 meters x 150 meters over the runway and cross road from an altitude of 15-50 meters.

    Test Objective 3 involves three X-8 multi-rotors equipped with a Beagle Bone black microcomputer, Wi-Fi adapter, and auxiliary battery. All three vehicles will be controlled by the on-board processor and script, to observe the behavior when no input commands are given. Some individual waypoints will be given to each vehicle to observe how the rest of the swarm reacts. As in Objectives 1&2, flight will cover an area of approximately 150 meters x 150 meters over the runway and cross road from an altitude of 15-50 meters.

Test Objective 4 involves three X-8 multi-rotors equipped with a Beagle Bone black microcomputer, Wi-Fi adapter, and auxiliary battery. Two vehicles will be controlled by the on-board processor and script, while the third will be controlled by the autopilot, preprogrammed to follow waypoints and will broadcast the same data as the script-controlled vehicles. As in previous Objectives, flight will cover an area of approximately 150 meters x 150 meters over the runway and cross road from an altitude of 15-50 meters.

For all flight test objectives, the multi-rotor vehicles will fly as commanded by the on-board script in an area adjacent to, but not directly overflying the test team setup. The ground-controlled vehicle will be flown in such a way as to familiarize the pilot with the swarm response and capabilities. Network data will be collected during the flight, but post flight processing will be used to evaluate swarm performance. Test team positions will include, at a minimum, a safety pilot (external operator and remote pilot-in-charge) for each aircraft, ground station operator (internal operator), and observer(s).

b. Purpose:
These tests are designed to evaluate the performance of an open-source swarm architecture using an ad-hoc Wi-Fi network. Tests are designed to evaluate network performance, swarm capabilities, familiarize the ground pilot with swarm capabilities, and determine the architecture's feasibility for future algorithm development.

c. List of AFIT and non-AFIT assets at risk:
   i. Three (3) 3D Robotics X8 multi-rotor UAS with Pixhawk autopilots.
   ii. AFIT Personnel (a mix of military and civilian staff and students)
   iii. Personal vehicles and government (AFIT) owned flight test trailer

d. Location of test:
Camp Atterbury, Indiana or WPAFB Area B test range

e. Planned dates of the test:
22-25 October (29-31 October)

f. Number of projected flights during the test period:
Approximately 20-30 flights during the planned test period.

3. MISHAP RESPONSIBILITIES

a. Should an incident occur in which one of the UAVs is damaged or destroyed, the AFIT Flight Test Safety Officer (FTSO) will be notified via the After-Action Report (Section VII of this document).

b. <u>Camp Atterbury, IN or WPAFB</u>. If an incident occurs in which property or personnel on Camp Atterbury or WPAFB is damaged/injured, the installation emergency 911 service will be notified immediately. In addition, the AFIT Safety Office will be notified within 5 working days per AFIT's Mishap Notification Procedures. Serious mishaps must be reported immediately.

AFIT Safety Office (Tim Thomas, 255-3636 x3627, Cell Phone 937-479-9715)

c. If an injury or illness results in lost duty time or hospitalization, then the AFIT Safety Office will be notified immediately. The Principal Investigator will be responsible for

submitting any of the required mishap reports as defined in AFIT's Mishap Notification Procedures. For further information, refer to the Mishap Notification Procedures posted in the Safety folder under the 'Mishap Reporting' tab on the AFIT Intranet site.

4. TEST OBJECTIVES

Summarize the top-level objectives listed in the test plan:

a. Objective 0– Collection of swarm telemetry with one leader
b. Objective 1– Collection of swarm telemetry with one leader and one follower
c. Objective 2– Collection of swarm telemetry with one leader and two followers
d. Objective 3– Collection of swarm telemetry with homogeneous swarm
e. Objective 4– Collection of swarm telemetry with one leader and two followers via autopilot

5. TEST ITEM DESCRIPTION

a. Vehicle description
    i. X8 Multi-rotor (3 vehicles)
        1. Vendor:
            3D Robotics, Berkeley, CA
        2. Model:
            X8
        3. Characteristics:
            dimensions – 13.7" x 20.1" x 11.8"
            Propeller Size – 11x4.7 SF (4), 11x4.7 SFP (4)
            Weight – 10 lbs.
        4. Payload description:
            a. Vehicle 1: Beagle Bone black microcomputer, Alfa Wi-Fi adapter, aux. battery, serial connection between the Beagle Bone black and Pixhawk Telem2 port.
            b. Vehicle 2: Beagle Bone black microcomputer, Alfa Wi-Fi adapter, aux. battery, serial connection between the Beagle Bone black and Pixhawk Telem2 port.
            c. Vehicle 3: Beagle Bone black microcomputer, Alfa Wi-Fi adapter, aux. battery, serial connection between the Beagle Bone black and Pixhawk Telem2 port.
        5. Flight computer (autopilot):
            Pixhawk
        6. Power plant system:
            800KV brushless outrunner motor.
            3D Robotics 2A Speed Controller.
            3D Robotics Power Distribution Board
            4S 14.8V 10,000 mAh 10C LiPo battery.
        7. Datalink:
            a. 3dRobotics 915 MHz modems (autopilot comm/telemetry),
            b. FrSky Taranis (RC) transmitter with FrSky X8R receiver (2.4 GHz FHSS)
            c. Alfa AWUS036NHA Wi-Fi adapter (2.4 GHz)
        8. Range: < 1000 ft.
        9. Airframe Modifications
            a. Microprocessor and Wi-Fi adapter mount
            b. On-board swarm algorithm script

b. Flight Conditions

i. Launch & Recovery Method
Vehicles will be manually controlled by the safety pilot during launch. Multi-rotor aircraft will perform a vertical takeoff and landing (VTOL) in stabilize mode from a level stationary position away from personnel and equipment. The autonomous control script will be running on the on-board processor but will not be broadcasting any data or sending any control commands in the stabilize mode. Both the pilot and safety observer will maintain positive communication and ensure the aircraft is free from obstructions.

ii. Method of Piloting
Manual pilot control will be used for takeoff and landing, and full-time for the "lead" vehicle. The follower vehicles will be switched over to scripted control when all UAS achieve 15 meters AGL. Swarm commands are provided by the Beagle Bone black microcomputer when in Guided mode. The safety pilot can take manual control at any time during operations. If communications between GCS and UAS are lost, the aircraft will continue to be controlled by the swarm algorithm. If communications are lost for any given algorithm-controlled vehicle, that vehicle will hold its position.

iii. Flight Altitude(s)
The maximum allowable altitude is 120 m (~400 ft) AGL. The minimum operating altitude is set at 3 m (10 ft) AGL for multi-rotor aircraft. Altitudes for this test will range from 10 to 80 m.

iv. Flight Range
Visual LOS distances only (< 300 m for these test objectives). Maximum tested range of Radio Control (RC) radio link is one km. Maximum specified range of Autopilot Command link is two km. Maximum estimated range of the ad hoc Wi-Fi network is 300 m between nodes.

v. Flight Duration
Maximum duration of flight with charged batteries has been tested to 20 minutes.

vi. Flight Speed
The X-8's are capable of a cruise speed of 2 – 10 m/sec (~6-20 mph), but will be limited to 5 m/sec by parameter setting. The swarming algorithm limits commanded speeds to 5 m/sec or less.

vii. Weather Conditions
Operation is not allowed after sunset, under rain or when lightning warning is issued. Electronics are not shielded and water will result to short circuits in the system. Operation is limited to a maximum wind speed (including gusts) of 10 knots, with a cross wind of less than 10 knots.

c. Failure Modes

A complete list of failure modes is found in the AFIT Small Multi-rotor Air Worthiness Memo, October 2016.

d. Describe the test facilities to be used:

WPAFB Area B: The right to fly SUAS at this airfield is outlined in a FAA Certificate of Authorization (COA), owned and maintained by AFRL. To fly under this COA, AFIT must maintain all rules outlined in the COA and associated MOA. The airfield is maintained by AFRL and located adjacent to the AF museum. This

airfield is in good condition and is often used by AFRL and the RAMS model aircraft club. See *figure 1*.

<u>Camp Atterbury sUAS Range:</u> The right to fly SUAS at this airfield is outlined in a FAA Certificate of Authorization (COA), owned and maintained by AFRL. To fly under this COA, AFIT must maintain all rules outlined in the COA and associated MOA. The airstrip is maintained by Camp Atterbury is located in the eastern region of the Camp Atterbury range. The airstrip is in good condition, paved, and dedicated to sUAS use only. *See figure 2.*

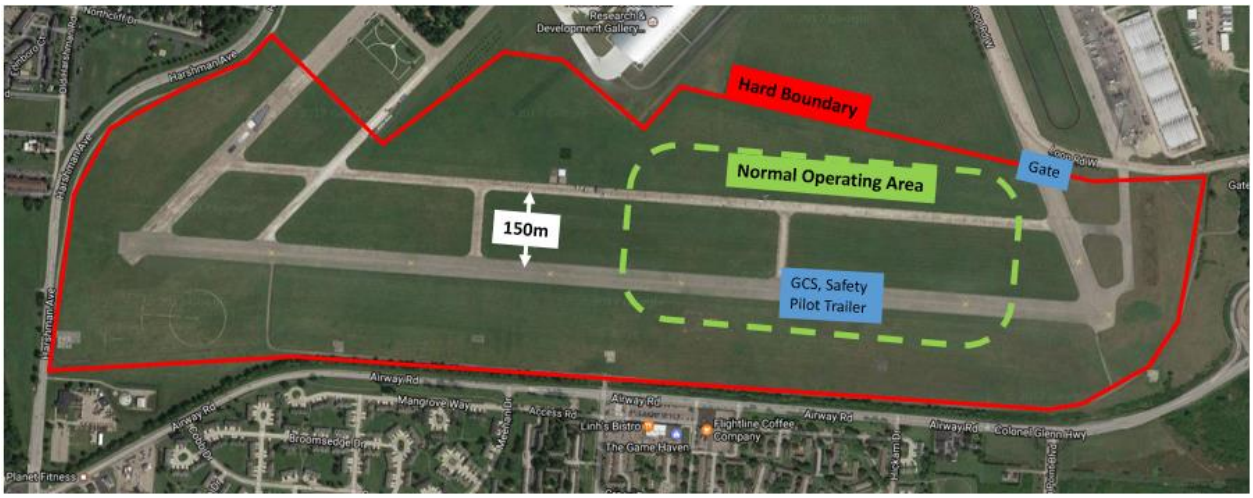


Figure 1: Flight area

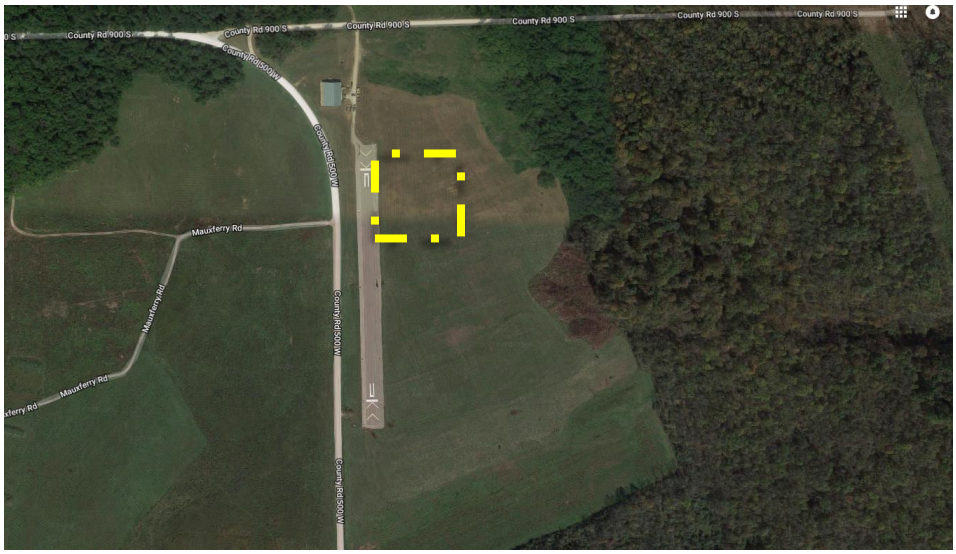


Figure 2. Camp Atterbury

6. SYSTEM MATURITY

   a. Describe testing that supports readiness:

Previous to this test, each airframe has been flight tested with similar size/weight payloads onboard to ensure airworthiness. Autopilot and communication systems utilized in this test have been used extensively on the X-8 multi-rotors over the past four years.

The serial cable providing velocity command inputs to the autopilot and supplying telemetry to broadcast via Wi-Fi has been previously tested using multiple ground tests in October 2017. No problems associated with this connection to the autopilot have been observed.

b. Previous lessons learned:

Ground personnel will avoid standing at launch origination point so as to avoid contact with the vehicle in the event of an unexpected return-to-launch (RTL) event.

Multi-rotors may experience rapid descent in the event of low battery voltage. This necessitates the following procedural behaviors:

- Battery voltage and current consumption must be closely monitored by the ground station operator. If battery voltage drops below 14 V, and/or current consumption reaches 70% of rated battery capacity, the vehicle must be quickly and safely landed.
- A multi-rotor vehicle should never directly overfly personnel. This greatly reduces the probability of impact with personnel in the event of power loss to the vehicle.

c. Authorized flight:
This flight is authorized by the AFIT MFR which has been approved by the Unmanned Aerial Systems Airworthiness office at AFLCMC.

ORM Checklist Form

Date: _____ Control #:_____

	GREEN	YELLOW	RED
Crew Rest	Good	Marginal	Poor
Crew/Personal Concerns	None	Minor	Major
Primary Crew Qualified	All Qualified	1 Unqualified	2+ Unqualified
7+ Days TDY/Leave	2nd duty day back or later	1st duty day back	
Perceived Scheduling Pressure	None	Some	Significant Pressure to Complete Mission
Duty Day	<8 hours	>8 hrs	>12 hours
Showtime	0600-1600	0300-0600/1600-2200	2200-0300
Planning Changes (Last 24 hrs)	Minimal/No impact	Minor	Major
Mission Complexity	Low/Normal	Demanding	Extremely Demanding
Test Mission/Safety Risk	Low	Medium	High
Cross Winds/Wind Speed	<10 kts	10-13 kts	13-15 kts
Time of Day	Day	Night	0200-0500 TO/Landing
Airframe Modification	Minor	Significant	Severe
Maturity-Hardware/Software	Nothing New	1st Flight of Hardware/Software Mod	1st Flight of NEW Hardware/Software
Additional Risk Not Addressed	Low	Medium	High

This checklist is to be briefed at the beginning of each test day.

Each green box is 0 points. Each yellow box is 1 point. Each red box is 2 points.
A score of 0-3: Attempt to mitigate any red boxes to reduce the risk. Test director's discretion to continue the mission.
A score of 3-5: If unable to lower the score to 0-3, it is the Principal Investigator's discretion to continue the mission.
A score of 6 or higher: If unable to lower the score, it is the AFIT FTSO's discretion as to whether or not to continue the mission.

IF YOU ARE NOT READY TO FLY… DON'T!

TEST DESCRIPTION

Objective 0 – Initial Swarming Algorithm Test – Leader

TEST SCENARIO 0	
Description	The objective of this test is to verify functionality of the modified Reynolds flocking algorithm implemented on X-8 multirotor aircraft, specifically the flight floor limits placed on the aircraft. Broadcast position/velocity data from each aircraft will be captured and stored by Wi-fi nodes. Velocity commands and resulting velocity will be recorded on-board the Pixhawk autopilots and Beagle Bone Black. During the test, the vehicle will cover an area approximately 50 m X 50 m over a portion of the runway and grass. Vehicle speeds will vary from 0-5 m/sec, and a software limit of 5 m/s will be placed on the autopilots.
Stakeholders	Capt. Corey Combs; Instructor: J. Colombi

Success Criteria	Successful if: Safety pilot comfortable with response to algorithm flight floor limits.
Evaluation Criteria	**<u>Satisfactory if:</u>** 1. Telemetry and command data are collected during desired intervals. 2. Safety pilot is comfortable with response and timing of vehicle.
Data Requirements	Required: 1. Telemetry from X-8 multi-rotor UAS, specifically velocity commands and achieved velocity, "bucket" breakout of velocity commands, and distances to other vehicles in the swarm. 2. Telemetry sent via Wi-Fi UDP multicast.
Algorithms	1. Mission Planner software for ground station 2. Lightweight Communications and Marshalling data log 3. On-board modified Reynolds flocking algorithm
Expected Results	1. Telemetry data collected
Assets	1. AFIT X8 multi-rotor 2. Beagle Bone black microcomputer 3. Alfa AWUS036NHA Wi-Fi adapter 4. Ground station computers

Test Methodology	**Test Procedures** 1. BEFORE TAKEOFF: a. Open airspace with range control. b. Check that weather is within limits and determine launch/recovery locations and headings. c. Setup GCS and operating area IAW AFIT Document 5028. d. Enable GCS Wi-Fi in ad-hoc mode. e. Complete all required preflight checklists for UAS. f. SSH connect from GCS to each UAS. g. Start autonomous algorithm script in background on each UAS. h. Start LCM listeners for each UAS on the GCS. i. Safety pilot changes mode to alt-hold, ensure GCS receives position/velocity data via LCM listeners. 2. LAUNCH: a. Ensure that all present personnel are aware of launch. b. Move multi-rotor to desired return to launch position c. Arm multi-rotor d. Safety pilot performs manual takeoff of leader multi-rotor in stabilize mode e. Safety pilot announces that aircraft is airborne. f. Climb to pre-briefed transition altitude. g. Transition to pre-briefed test-point operating area. 3. EXECUTE TEST POINTS: a. Switch vehicle to guided mode. b. Execute planned test points, monitor both vehicles for erroneous behavior, and monitor battery voltage and current consumed. c. If sufficient battery power remains, an additional test point may be conducted if it does not require any changes d. If remaining battery power is insufficient to accomplish additional test points, proceed to recovery 4. RECOVERY: a. Transition aircraft to safety pilot control. b. Ensure vehicles are in stabilize mode. c. Ensure landing area is clear of personnel and equipment. d. Navigate aircraft to pre-briefed recovery location. e. Begin descent and entry into landing pattern. f. Safety pilot announces landing to all present personnel. g. Execute recovery. 5. AFTER RECOVERY: a. Disarm multi-rotors b. Ensure telemetry on GCS is saved. c. Remain clear of propeller. d. Disconnect battery prior to moving aircraft by hand. e. Power off RC transmitter (as required). f. Close airspace with range control if done for the day.

Objective 1 – Initial Swarming Algorithm Test – Leader/Follower

TEST SCENARIO 1	
Description	The objective of this test is to verify functionality of the modified Reynolds flocking algorithm implemented on X-8 multirotor aircraft. Broadcast position/velocity data from each aircraft will be captured and stored by Wi-fi nodes. Velocity commands and resulting velocity will be recorded on-board the Pixhawk autopilots and Beagle Bone Black. During the test, the vehicles will cover an area approximately 150 m X 150 m over a portion of the runway and fly linear, circular, and box patterns as needed by the safety pilot for familiarization with follower behavior. Vehicle speeds will vary from 0-5 m/sec, and a software limit of 5 m/s will be placed on the autopilots.
Stakeholders	Capt. Corey Combs; Instructor: J. Colombi

Success Criteria	Successful if: Safety pilot comfortable with follower response to leader movement. Required data was collected for flight patterns.
Evaluation Criteria	**<u>Satisfactory if:</u>** 1. Telemetry and command data are collected during desired intervals. 2. Safety pilot is comfortable with response and timing of follower vehicle.
Data Requirements	Required: 1. Telemetry from X-8 multi-rotor UAS, specifically velocity commands and achieved velocity, "bucket" breakout of velocity commands, and distances to other vehicles in the swarm. 2. Telemetry sent via Wi-Fi UDP multicast.
Algorithms	1. Mission Planner software for ground station 2. Lightweight Communications and Marshalling data log 3. On-board modified Reynolds flocking algorithm
Expected Results	1. Telemetry data collected
Assets	1. AFIT X8 multi-rotor 2. Beagle Bone black microcomputer 3. Alfa AWUS036NHA Wi-Fi adapter 4. Ground station computers

Test Methodology	**Test Procedures** 1. BEFORE TAKEOFF: a. Open airspace with range control. b. Check that weather is within limits and determine launch/recovery locations and headings. c. Setup GCS and operating area IAW AFIT Document 5028. d. Enable GCS Wi-Fi in ad-hoc mode. e. Complete all required preflight checklists for UAS. f. SSH connect from GCS to each UAS. g. Start autonomous algorithm script in background on each UAS. h. Start LCM listeners for each UAS on the GCS. i. Safety pilot changes mode to alt-hold, ensure GCS receives position/velocity data via LCM listeners. 2. LAUNCH: a. Ensure that all present personnel are aware of launch. b. Move multi-rotors to desired return to launch position c. Arm multi-rotors d. Safety pilot performs manual takeoff of leader multi-rotor in stabilize mode e. Safety pilot announces that aircraft is airborne. f. Climb to pre-briefed transition altitude. g. Transition to pre-briefed test-point operating area. 3. EXECUTE TEST POINTS: a. Switch both vehicles to altitude hold mode. a. Switch follower vehicle to guided mode. b. Execute planned test points, monitor both vehicles for erroneous behavior, and monitor battery voltage and current consumed. c. If sufficient battery power remains, an additional test point may be conducted if it does not require any changes d. If remaining battery power is insufficient to accomplish additional test points, proceed to recovery 4. RECOVERY: a. Transition aircraft to safety pilot control. b. Ensure vehicles are in stabilize mode. c. Ensure landing area is clear of personnel and equipment. d. Navigate aircraft to pre-briefed recovery location. e. Begin descent and entry into landing pattern. f. Safety pilot announces landing to all present personnel. g. Execute recovery. 5. AFTER RECOVERY: a. Disarm multi-rotors b. Ensure telemetry on GCS is saved. c. Remain clear of propeller. d. Disconnect battery prior to moving aircraft by hand. e. Power off RC transmitter (as required). f. Close airspace with range control if done for the day.

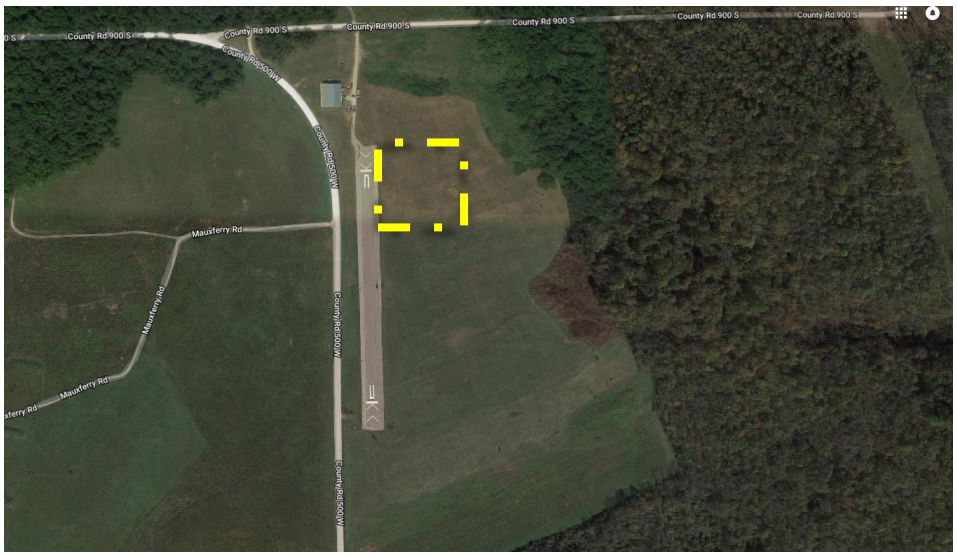
Objective 2 – Swarming Algorithm Test – Leader with Two Followers

TEST SCENARIO 2	
Description	The objective of this test is to verify functionality of the modified Reynolds flocking algorithm implemented on X-8 multirotor aircraft. Broadcast position/velocity data from each aircraft will be captured and stored by Wi-fi nodes. Velocity commands and resulting velocity will be recorded on-board the Pixhawk autopilots and Beagle Bone Black. During the test, the vehicles will cover an area approximately 150 m X 150 m over a portion of the runway and fly linear, circular, and box patterns as needed by the safety pilot for familiarization with follower behavior. Vehicle speeds will vary from 0-5 m/sec, and a software limit of 5 m/s will be placed on the autopilots.
Stakeholders	Capt. Corey Combs; Instructor: J. Colombi

Success Criteria	Successful if: Safety pilot comfortable with follower response to leader movement. Required data was collected for flight patterns.
Evaluation Criteria	**<u>Satisfactory if:</u>** 1. Telemetry and command data is collected during desired intervals. 2. Safety pilot is comfortable with response and timing of follower vehicles.
Data Requirements	Required: 1. Telemetry from X-8 multi-rotor UAS, specifically velocity commands and achieved velocity, "bucket" breakout of velocity commands, and distances to other vehicles in the swarm. 2. Telemetry sent via Wi-Fi UDP multicast.
Algorithms	1. Mission Planner software for ground station 2. Lightweight Communications and Marshalling data log 3. On-board modified Reynolds flocking algorithm
Expected Results	1. Telemetry data collected
Assets	1. AFIT X8 multi-rotor 2. Beagle Bone black microcomputer 3. Alfa AWUS036NHA Wi-Fi adapter 4. Ground station computers
Test Methodology	**Test Procedures** 1. BEFORE TAKEOFF: a. Open airspace with range control. b. Check that weather is within limits and determine launch/recovery locations and headings. c. Setup GCS and operating area IAW AFIT Document 5028. d. Enable GCS Wi-Fi in ad-hoc mode. e. Complete all required preflight checklists for UAS. f. SSH connect from GCS to each UAS. g. Start autonomous algorithm script in background on each UAS. h. Start LCM listeners for each UAS on the GCS. i. Safety pilot changes mode to alt-hold, ensure GCS receives position/velocity data via LCM listeners. 2. LAUNCH: a. Ensure that all present personnel are aware of launch. b. Move multi-rotors to desired return to launch position c. Arm multi-rotors d. Safety pilot performs manual takeoff of leader multi-rotor in stabilize mode e. Safety pilot announces that aircraft is airborne. f. Climb to pre-briefed transition altitude. g. Transition to pre-briefed test-point operating area. 3. EXECUTE TEST POINTS:

	a. Switch all vehicles to altitude hold mode. b. Switch follower vehicles to guided mode. c. Execute planned test points, monitor both vehicles for erroneous behavior, and monitor battery voltage and current consumed. d. If sufficient battery power remains, an additional test point may be conducted if it does not require changes e. If remaining battery power is insufficient to accomplish additional test points, proceed to recovery 4. RECOVERY: a. Transition aircraft to safety pilot control. b. Ensure landing area is clear of personnel and equipment. c. Navigate aircraft to pre-briefed recovery location. d. Begin descent and entry into landing pattern. e. Safety pilot announces landing to all present personnel. f. Execute recovery. 5. AFTER RECOVERY: a. Disarm multi-rotors b. Ensure telemetry on GCS is saved. c. Remain clear of propeller. d. Disconnect battery prior to moving aircraft by hand. e. Power off RC transmitter (as required). f. Close airspace with range control if done for the day.

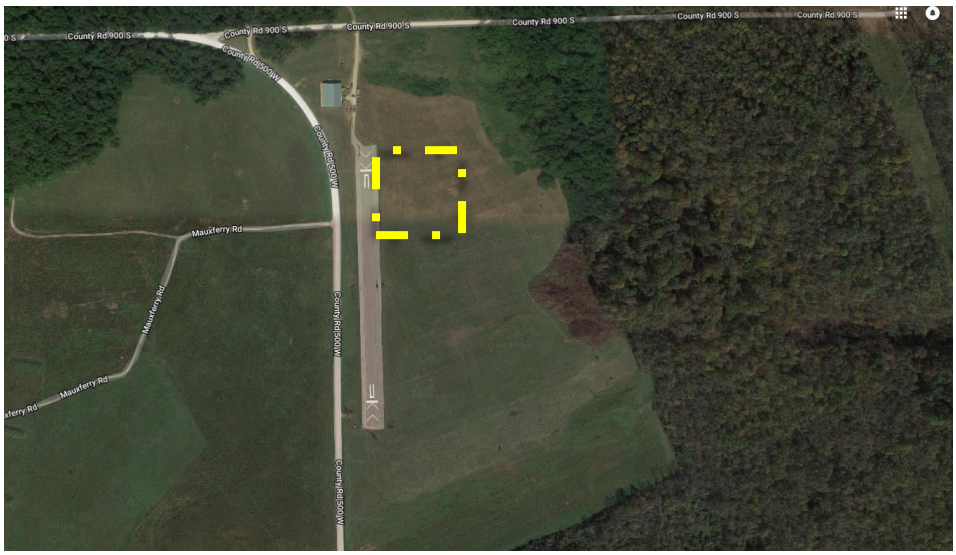
Objective 3 – Swarming Algorithm Test – Swarm Stability

TEST SCENARIO 3	
Description	The objective of this test is to verify functionality of the modified Reynolds flocking algorithm implemented on X-8 multirotor aircraft. Broadcast position/velocity data from each aircraft will be captured and stored by Wi-fi nodes. Velocity commands and resulting velocity will be recorded on-board the Pixhawk autopilots and Beagle Bone Black. During the test, the vehicles will cover an area approximately 150 m X 150 m over a portion of the runway and will be controlled via autopilot in a point and go manner. Vehicle speeds will vary from 0-5 m/sec, and a software limit of 5 m/s will be placed on the autopilots. 
Stakeholders	Capt. Corey Combs; Instructor: J. Colombi

Success Criteria	Successful if: Safety pilot comfortable with follower response to leader movement. Required data was collected for flight patterns.
Evaluation Criteria	**<u>Satisfactory if:</u>** 1. Telemetry and command data is collected during desired intervals. 2. Safety pilot is comfortable with response and timing of follower vehicles.
Data Requirements	Required: 1. Telemetry from X-8 multi-rotor UAS, specifically velocity commands and achieved velocity, "bucket" breakout of velocity commands, and distances to other vehicles in the swarm. 2. Telemetry sent via Wi-Fi UDP multicast.
Algorithms	1. Mission Planner software for ground station 2. Lightweight Communications and Marshalling data log 3. On-board modified Reynolds flocking algorithm
Expected Results	1. Telemetry data collected
Assets	1. AFIT X8 multi-rotor 2. Beagle Bone black microcomputer 3. Alfa AWUS036NHA Wi-Fi adapter 4. Ground station computers
Test Methodology	**Test Procedures** 1. BEFORE TAKEOFF: a. Open airspace with range control. b. Check that weather is within limits and determine launch/recovery locations and headings. c. Setup GCS and operating area IAW AFIT Document 5028. d. Enable GCS Wi-Fi in ad-hoc mode. e. Complete all required preflight checklists for UAS. f. SSH connect from GCS to each UAS. g. Start autonomous algorithm script in background on each UAS. h. Start LCM listeners for each UAS on the GCS. i. Safety pilot changes mode to alt-hold, ensure GCS receives position/velocity data via LCM listeners. 2. LAUNCH: a. Ensure that all present personnel are aware of launch. b. Move multi-rotors to desired return to launch position c. Arm multi-rotors d. Safety pilot performs manual takeoff of leader multi-rotor in stabilize mode e. Safety pilot announces that aircraft is airborne. f. Climb to pre-briefed transition altitude. g. Transition to pre-briefed test-point operating area.

	3. EXECUTE TEST POINTS: f. Switch all vehicles to altitude hold mode. a. Switch all vehicles to guided mode. b. Execute planned test points, monitor both vehicles for erroneous behavior, and monitor battery voltage and current consumed. c. If sufficient battery power remains, an additional test point may be conducted if it does not require changes d. If remaining battery power is insufficient to accomplish additional test points, proceed to recovery 4. RECOVERY: a. Transition aircraft to safety pilot control. b. Ensure landing area is clear of personnel and equipment. c. Navigate aircraft to pre-briefed recovery location. d. Begin descent and entry into landing pattern. e. Safety pilot announces landing to all present personnel. f. Execute recovery. 5. AFTER RECOVERY: a. Disarm multi-rotors b. Ensure telemetry on GCS is saved. c. Remain clear of propeller. d. Disconnect battery prior to moving aircraft by hand. e. Power off RC transmitter (as required). f. Close airspace with range control if done for the day.

Objective 4 – Swarming Algorithm Test – Leader with Two Followers

TEST SCENARIO 4	
Description	The objective of this test is to verify functionality of the modified Reynolds flocking algorithm implemented on X-8 multirotor aircraft. Broadcast position/velocity data from each aircraft will be captured and stored by Wi-fi nodes. Velocity commands and resulting velocity will be recorded on-board the Pixhawk autopilots and Beagle Bone Black. During the test, the vehicles will cover an area approximately 150 m X 150 m over a portion of the runway and fly linear, circular, and box patterns as needed by the safety pilot for familiarization with follower behavior. Vehicle speeds will vary from 0-5 m/sec, and a software limit of 5 m/s will be placed on the autopilots. 
Stakeholders	Capt. Corey Combs; Instructor: J. Colombi

Success Criteria	Successful if: Safety pilot comfortable with follower response to leader movement. Required data was collected for flight patterns.
Evaluation Criteria	**<u>Satisfactory if:</u>** 1. Telemetry and command data is collected during desired intervals. 2. Safety pilot is comfortable with response and timing of follower vehicles.
Data Requirements	Required: 1. Telemetry from X-8 multi-rotor UAS, specifically velocity commands and achieved velocity, "bucket" breakout of velocity commands, and distances to other vehicles in the swarm. 2. Telemetry sent via Wi-Fi UDP multicast.
Algorithms	1. Mission Planner software for ground station 2. Lightweight Communications and Marshalling data log 3. On-board modified Reynolds flocking algorithm
Expected Results	1. Telemetry data collected
Assets	1. AFIT X8 multi-rotor 2. Beagle Bone black microcomputer 3. Alfa AWUS036NHA Wi-Fi adapter 4. Ground station computers
Test Methodology	**Test Procedures** 1. BEFORE TAKEOFF: a. Open airspace with range control. b. Check that weather is within limits and determine launch/recovery locations and headings. c. Setup GCS and operating area IAW AFIT Document 5028. d. Enable GCS Wi-Fi in ad-hoc mode. e. Complete all required preflight checklists for UAS. f. SSH connect from GCS to each UAS. g. Start autonomous algorithm script in background on each UAS. h. Start LCM listeners for each UAS on the GCS. i. Safety pilot changes mode to alt-hold, ensure GCS receives position/velocity data via LCM listeners. 2. LAUNCH: a. Ensure that all present personnel are aware of launch. b. Move multi-rotors to desired return to launch position c. Arm multi-rotors d. Safety pilot performs manual takeoff of leader multi-rotor in stabilize mode e. Safety pilot announces that aircraft is airborne. f. Climb to pre-briefed transition altitude. g. Transition to pre-briefed test-point operating area.

	3. EXECUTE TEST POINTS: a. Switch all vehicles to altitude hold mode. b. Switch follower vehicles to guided mode. c. Execute planned test points, monitor both vehicles for erroneous behavior, and monitor battery voltage and current consumed. d. If sufficient battery power remains, an additional test point may be conducted if it does not require changes e. If remaining battery power is insufficient to accomplish additional test points, proceed to recovery 4. RECOVERY: a. Transition aircraft to safety pilot control. b. Ensure landing area is clear of personnel and equipment. c. Navigate aircraft to pre-briefed recovery location. d. Begin descent and entry into landing pattern. e. Safety pilot announces landing to all present personnel. f. Execute recovery. 5. AFTER RECOVERY: a. Disarm multi-rotors b. Ensure telemetry on GCS is saved. c. Remain clear of propeller. d. Disconnect battery prior to moving aircraft by hand. e. Power off RC transmitter (as required). f. Close airspace with range control if done for the day.

SAFETY PLAN

1. QUALIFICATION AND TRAINING
    a. Dr. John Colombi – SENG651 UAS Design and Test Course Instructor. Several years' experience with AFIT UAS simulations and testing. Successfully completed the Camp Atterbury Range Safety course.
    b. Dr. David Jacques – Lead faculty member of the AFIT UAS program. Experienced in UAS and real-world testing. Flies RC aircraft as a hobby.
    c. Mr. Rick Patton – CESI employee and safety pilot with many years of experience flying RC aircraft. Certified by the FAA under Part 107.
    d. Mr. Dave Thacker – MacB contractor and safety pilot with many years of experience flying RC aircraft
    e. Mr. Jeremy Gray - Graduate of AFIT SE program with SUAS specialization. Now acts as the principle research engineer for SUAS. Certified by the FAA under Part 107.

2. GENERAL MINIMIZING CONDITIONS

    The following general minimizing procedures and considerations will be followed for the duration of this flight test program:

    1. All test flights will be conducted in day Visual Meteorological Conditions (VMC) conditions.

2. Communications will be maintained between the ground operator, safety observers, safety pilot and test crews at all times.
3. Where necessary, the safety pilots will maintain positive radio communications with range control/tower at all times.
4. Flying over non-participating personnel and facilities will be avoided.
5. Hardware fail-safes will be utilized to minimize impact of lost communications between the aircraft and RC transmitter.
6. To minimize probability of in-flight low battery power, all flight durations will be timed and, cumulatively, remain below 20 minutes. Thereafter, it will be replaced by a fully charged battery before the test flight can be continued. All batteries will be charged prior to flight testing and marked charged.
7. Personnel without assigned roles for a given test will be observers of flight operations while outside the flight test trailer. Minimize all unnecessary conversations and distractions during flight.
8. A multi-purpose fire extinguisher is located in the rear of the trailer in the event of a fire.
9. Utilize “Knock-It-Off” and “Terminate” procedures in accordance with AFI 11-214 paragraph 3.4.
10. Minimum altitude for flight test is 30 m AGL unless a lower altitude is approved for test requirements.
11. To minimize exposure of UAS flying overhead, test crew will only enter the UAS landing strip to change the beacon setting upon instruction by project team lead.
12. Maintain visual contact at all times.
13. If propulsion battery is connected, the propeller will be considered powered at all times. As such, avoid the propeller area of the aircraft.
14. Flight patterns performed by multi-rotor aircraft will be performed in front of and never overhead of the test team.

3. TEST HAZARD ANALYSES (THA's)
   A. Battery Fire
   B. Collision with Object
   C. Collision with Personnel
   D. Loss of Communication between GS laptop and UAS
   E. Loss of Communication between Safety Pilot Controller and UAS
   F. Total loss of Flight Control (Failure of Pixhawk)
   G. Loss of GPS signal
   H. Collision with another UAS
   I. Algorithm Failure

		Mishap Severity Category			
		Catastrophic – I Death, System/Facility Loss, Severe Environmental Damage (e.g. Class A Mishap)	**Critical – II** Severe Injury, Occupational Illness, or Major System/Facility/ Environmental	**Marginal – III** Minor Injury, Occupational Illness, or Minor System/Facility/ Environmental	**Negligible – IV** Less than Minor Injury, Occupational Illness, or System/ Facility/ Environmental Damage
Probability of Mishap Occurring During the Test	**Very Likely (A)** Highly expected to occur – Many significant concerns even after mitigation applied.	**1**	**3**	**7**	**13**
	Likely (B) Expected to occur – Significant concerns remain after mitigation applied.	**2**	**5**	**9**	**16**
	Less Likely (C) Not expected but possible – Some concern exists even with mitigation applied.	**4**	**6**	**11** B	**18**
	Unlikely (D) Unexpected – Minor concerns after mitigation applied.	**8**	**10**	**14**	**19** D, H
	Very Unlikely (E) Highly unexpected – Little or no concern after mitigation applied.	**12**	**15**	A, C, F **17**	E, G **20**

HIGH
MEDIUM
LOW
NEGLIGIBLE

[Place the appropriate letter of each THA in the appropriate test hazard box.]

TEST HAZARD ANALYSIS (THA)	Page 1/8
TEST SERIES Multiple Autonomous UAV Swarm – Mesh Ad-hoc Network	MISHAP CAT/PROBABILITY III/Very Unlikely
PREPARED BY Capt. Corey Combs	SIGNATURE
AFIT FLIGHT TEST SAFETY OFFICER Lt Col Amy Cox	SIGNATURE

HAZARD: Battery Fire

CAUSE:
1. Uncontrolled discharge of power from the battery leading to overheating and fire (thermal runaway)
2. Overcharging of battery leading to thermal runaway due to charger malfunction or human input error
3. Battery circuitry or subsystem component failure or wiring malfunction
4. Battery puncture

EFFECT:

1. Loss of UAS
2. Injury to personnel
3. High temperature, toxic fire

MINIMIZING PROCEDURES:

1. (1,2,3) All batteries will be placed in fireproof metal containers to prevent damage during transportation.
2. (1,2,3) All batteries will be charged using authorized battery chargers and by personnel trained in the proper recharging techniques. Battery will be placed in fireproof pouch during charging.
3. (2) All batteries will be charged in approved locations while monitored and placed in fire proof pouches.
4. (1,2,3) Only the proper battery types for the specified aircraft will be used
5. (1,2,3) Only authorized Electronic Speed Controllers (ESCs) for the specified battery size will be utilized.
6. (1,2,3) Load balancer will be used when charging flight batteries.
7. (2) Downed aircraft will be approached with caution due to the increased possibility of a battery fire.
8. (1,2,3) Damaged batteries will be safely stored away from other flammable items and within a fireproof metal container.

CORRECTIVE ACTIONS:

If the battery begins to smoke while charging:
1. Move battery in fireproof pouch outdoors
2. Keep battery in fireproof pouch and place it on a hardened surface away from flammable objects.

If the battery catches fire during ground operations:

1. Announce battery fire and avoid toxic fumes created by the battery fire.
2. The person nearest to the fire extinguisher will use the fire extinguisher to put out the fire.
3. The person in communication with the field controller will notify the field controller of the emergency via the radio.

If the battery catches fire while in flight:

1. Announce battery fire.
2. The pilot in command will immediately land the aircraft (make attempt to land on hard surface).
3. All personal will remain away from the aircraft until the safety pilot deems it safe to approach and put out the fire with the fire extinguisher.
4. The person in communication with the field controller will notify the field controller of the emergency via the radio.

REMARKS: None

TEST HAZARD ANALYSIS (THA)	Page 2/8
TEST SERIES Multiple Autonomous UAV Swarm – Mesh Ad-hoc Network	MISHAP CAT/PROBABILITY III/ Less Unlikely
PREPARED BY Capt. Corey Combs	SIGNATURE
AFIT FLIGHT TEST SAFETY OFFICER Lt Col Amy Cox	SIGNATURE

HAZARD: Collision with Object

CAUSE:

1. Bird strike
2. Collision with other aircraft
3. Collision with ground based obstructions

EFFECT:

1. Loss of UAS
2. Property damage

MINIMIZING PROCEDURES:

1. (1,2,3) Safety observers will be used to augment safety pilot.
2. (2) Set algorithm to maintain a generous physical offset between vehicles.
3. (1,2,3) Give safety pilots full authority to react if they feel the vehicles are going to collide.
4. (2) Communicate with the tower before testing to verify clear airspace.
5. (3) Identification of ground based obstructions (hazard)in area of operation before testing.
6. (3) Flight path will be adjusted in order to avoid ground based obstructions.

CORRECTIVE ACTIONS:

1. Announce collision with object.
2. Discontinue testing and verify there are no injuries.
3. Assess extend of damage.
4. Notify tower if hit or near miss with non-AFIT aircraft occurs.
5. Document exact damage with photos/video.
1. Follow mishap reporting procedures per section 2 of Project Description (page 2).
6. Examine and, if possible, repair the UAS.
7. If operational, perform a trim flight to verify functionality.

REMARKS: Normally this risk is “Very Unlikely” but since autonomous algorithms are being tested with aircraft in close proximity, it has been elevated to “Less Likely,” until the system baseline and algorithms have been proven to avoid collisions.

TEST HAZARD ANALYSIS (THA)	Page 3/8
TEST SERIES Multiple Autonomous UAV Swarm – Mesh Ad-hoc Network	MISHAP CAT/PROBABILITY III/ Very Unlikely
PREPARED BY Capt. Corey Combs	SIGNATURE
AFIT FLIGHT TEST SAFETY OFFICER Lt Col Amy Cox	SIGNATURE

HAZARD: Collision with Personnel

CAUSE:

1. Personnel interference during takeoff/landing
2. Loss of control of vehicle

EFFECT:

1. Personnel injury
2. Loss of UAS

MINIMIZING PROCEDURES:

1. (1) Launch/landing area will be cleared of all nonessential personnel during these phases of flight and launch and recovery of the aircraft will be announced loudly to all personnel.
2. (1, 2) All personnel will maintain situational awareness of vehicle/flight status and personnel in and around the test area.
3. (2) Test crew will only enter the UAS landing strip to change the beacon setting upon instruction by project team lead.

CORRECTIVE ACTIONS:

1. Discontinue testing and assess extend of injuries.
2. Coordinated through range control (812-526-1351) if injury is severe and requires emergency services; perform any necessary first aid until help arrives.
2. Follow mishap reporting procedures per section 2 of Project Description (page 2).
3. Verify suitability of crew composition to carry on.
4. If suitable, examine and, if possible, repair the UAS.
5. If operational, perform a trim flight to ensure functionality.

REMARKS: None

TEST HAZARD ANALYSIS (THA)	Page 4/8
TEST SERIES Multiple Autonomous UAV Swarm – Mesh Ad-hoc Network	MISHAP CAT/PROBABILITY IV/ Unlikely
PREPARED BY Capt. Corey Combs	SIGNATURE
AFIT FLIGHT TEST SAFETY OFFICER Lt Col Amy Cox	SIGNATURE

HAZARD: Loss of Communication between GS laptop and UAS

CAUSE:

1. Signal interference
2. Onboard APM transmitter /receiver failure
3. GS transmitter /receiver failure
4. UAS flying out of range
5. Automatic waypoint algorithm overrides GS inputs

EFFECT:

1. Vehicle spirals to ground
2. Unplanned off-field landing
3. Loss of control of aircraft

MINIMIZING PROCEDURES:

1. (1,2,3,5) Verify serviceability of communication equipment prior to test. Verify integrity of the antennae. Verify communication equipment batteries are adequately charged.
2. (4) Limit operation within maximum range of previously conducted test with same autopilot configuration.
3. (1) Coordinate flight operations and frequencies with Atterbury authorities.
4. (1,2,3,4) Lost link fail-safes (return to launch and loiter) will be pre-programmed.
5. (1,2,3,4) Separate communication system for safety pilot.
6. (2,3) Pre-flight checklist will be conducted.

CORRECTIVE ACTIONS:

1. Ground operator will immediately announce lost communications so the test team can help visually track the UAS.
2. Attempt to re-establish communications with the UAS.
3. If link cannot be re-established, switch to manual mode.
4. Safety pilot will land the UAS manually.
5. Examine and repair communication system.
6. Disable all scripts running on Ground Station when manual mode is selected by safety pilot

REMARKS: None

TEST HAZARD ANALYSIS (THA)	Page 5/8
TEST SERIES Multiple Autonomous UAV Swarm – Mesh Ad-hoc Network	MISHAP CAT/PROBABILITY IV/Very Unlikely
PREPARED BY Capt. Corey Combs	SIGNATURE
AFIT FLIGHT TEST SAFETY OFFICER Lt Col Amy Cox	SIGNATURE

HAZARD: Loss of Communication between Safety Pilot Controller and UAS

1. Signal interference
2. Onboard RC transmitter /receiver failure
3. Safety Pilot controller transmitter /receiver failure
4. UAS flying out of range
5. Automatic waypoint algorithm overrides safety pilot inputs

EFFECT:

1. Vehicle spirals to ground
2. Unplanned off-field landing
3. Loss of control of aircraft

MINIMIZING PROCEDURES:

1. (1,2,3,5) Verify serviceability of communication equipment prior to test. Verify integrity of the antennae. Verify communication equipment batteries are adequately charged.
2. (4) Limit operation within maximum range of previously conducted test using same RC configuration.
3. (1) Coordinate flight operations and frequencies with Atterbury authorities.
4. (1,2,3,4) Lost link fail-safes (return to launch and loiter) will be pre-programmed.
5. (1,2,3) Separate communication system for auto pilot.
6. (2,3) Pre-flight checklist will be conducted.

CORRECTIVE ACTIONS:

1. Safety pilot will immediately announce lost communications so the test team can help visually track the UAS.
2. Attempt to re-establish communications with the UAS.
3. If link cannot be re-established, ground operator will set way points to return the UAS back to the launch area.
4. Ground operator will initiate automatic landing from the GS.
5. Examine and repair communication system.
6. Disable all scripts running on Ground Station when manual mode is selected by safety pilot

REMARKS: None

TEST HAZARD ANALYSIS (THA)	Page 6/8
TEST SERIES Multiple Autonomous UAV Swarm – Mesh Ad-hoc Network	MISHAP CAT/PROBABILITY IV/Very Unlikely
PREPARED BY Capt. Corey Combs	SIGNATURE
AFIT FLIGHT TEST SAFETY OFFICER Lt Col Amy Cox	SIGNATURE

HAZARD: Total loss of Flight Control (Failure of APM or motors)

CAUSE:

1. APM failure
2. Connector (power supply to APM) failure
3. Multiple motor/ESC failure

EFFECT:

1. Damage to UAS
2. Damage to property
3. Injury to personnel

MINIMIZING PROCEDURES:

1. (1,2,3) Visual inspection of the aircraft will be accomplished prior to flight.
2. (1,2,3) Perform preflight control check.

CORRECTIVE ACTIONS:

3. Announce loss of control.
4. Notify Himsel AAF UNICOM of UAS status.
5. Attempt immediate landing away from personnel.
6. Keep personnel away from landing path.
7. Verify there are no injuries.
8. Follow mishap reporting procedures per section 2 of Project Description (page 2).
9. Document exact damage with photos/video.
10. Examine and, if possible, repair the UAS.
8. If operational, perform a trim flight to verify functionality.

REMARKS: None

TEST HAZARD ANALYSIS (THA)	Page 7/8
TEST SERIES Multiple Autonomous UAV Swarm – Mesh Ad-hoc Network	MISHAP CAT/PROBABILITY IV/ Unlikely
PREPARED BY Capt. Corey Combs	SIGNATURE
AFIT FLIGHT TEST SAFETY OFFICER Lt Col Amy Cox	SIGNATURE

HAZARD: GPS Signal Loss

CAUSE:

1. Signal interference
2. GPS receiver failure
3. Poor receiver/satellite geometry
4. Connector failure

EFFECT:

1. Loss of navigation (autopilot will not fly waypoints)
2. Unplanned off-field landing

MINIMIZING PROCEDURES:

1. (1,2,3,4) Follow approved preflight procedures for ensuring GPS signal.

CORRECTIVE ACTIONS:

1. Announce GPS loss.
2. Switch to manual control
3. Safety pilot maintains controlled flight.
4. If GPS is not re-acquired as determined by test team, recover the UAV using manual mode.

REMARKS: None

TEST HAZARD ANALYSIS (THA)	Page 8/8
TEST SERIES Multiple Autonomous UAV Swarm – Mesh Ad-hoc Network	MISHAP CAT/PROBABILITY IV/ Unlikely
PREPARED BY Capt. Corey Combs	SIGNATURE
AFIT FLIGHT TEST SAFETY OFFICER Lt Col Amy Cox	SIGNATURE

HAZARD: Algorithm Failure

CAUSE:

1. Signal interference
2. GPS receiver failure
3. Poor receiver/satellite geometry
4. Connector failure
5. Companion Computer Failure
6. Autopilot Failure

EFFECT:

1. Loss of navigation (autopilot will not fly waypoints)
2. Unplanned off-field landing
3. Loss of swarming safety

MINIMIZING PROCEDURES:

1. (1,2,3,4) Follow approved preflight procedures for ensuring GPS signal.
2. (5,6) Revert to safety pilot manual flight mode

CORRECTIVE ACTIONS:

1. Announce algorithm loss.
2. Switch to manual control
3. Safety pilot maintains controlled flight away from other swarm members
4. Recover the UAV using manual mode.

REMARKS: None

AFTER ACTION REPORT

Per ENOI 91-6, the After Action Report should be submitted to the FTSO no later than 7 calendar days from the completion of the test.

1. Use this section to briefly describe how the test was carried out. Were there any unusual events?

2. What test execution/safety lessons were learned during the test event?

APPENDIX A: SUAS PREFLIGHT CHECKLIST

In - Lab Pre-Flight Checklist		
Vehicle Name/Number		
Checked By/Date		
C/W	**Item**	**Description**
	Configuration	Based on Flight Test Approval (i.e. 5028, MFR, COA). 1 full tail number. 2 large vehicle numbers
	Vehicle C.G.	Ensure CG within tolerances
	Communication Configuration	Check trans/Rec power and node ID (node ID = Tail Number) **Power:_____Node: ID_____**
	GCS Check	Mavlink packets recieved: **>90%,** No loss of communication, No hanging on connection (slow param loading)
	PID & Parameters File Name	Load Approved parameter file
	Auto Pilot Flight Settings	Verify Cruise Airspeed, Airspeed/Bank Limits, Min/Max throttle settings
	Failsafes	Check RC short and long failsafes are turned on and working (turn off transmitter while powered)
	Vehicle Power on GCS	Ensure autopilot configured to display mAh drawn and voltage on GCS (if correct power module used)
	RC Control	Check motor engagement for roll/pitch/yaw
	Compass Calibration	Perform compass calibration
	Calibrate Accelerometer	Perform three axis calibration
	GPS	Ensure GPS gets lock
	Propeller Direction	Check propeller(s) tight and in the correct position and orientation. Run motor to ensure direction of thrust
	Check Mode switch	Stabilize/Altitude Hold/Guided/Auto
	Check Ad Hoc LCM Comms	Ensure LCM messages are being sent and received and all vehicles are on the network
	UgCS Comm Check	Ensure vehicle shows up on UgCS window
	Check Payload	Ensure payload is secured and properly plugged in
	Check ALL Cables	Ensure all cables are tightly plugged into the correct location on the pixhawk and peripheral equipment (GPS, mag, AS).

On completion of this checklist, **place the checklist on/in the vehicle**.
IF anyone changes any aspect of a checked out aircraft, the **aircraft MUST be checked out again**.

This checklist must be completed before each flight.

Pre-Flight Checklist						
Vehicle Name/Number						
Date/Time						
Flight #					Item	Description
1	2	3	4	5		
					Range and Safety	Range is open OR NOTAM is up, ORM checklist, safety & test brief
					Battery Voltage	Manually Check Battery Voltage prior to install into vehicle
					Vehicle C.G	Ensure CG within tolerances
					Zero Gyros (on power)	Ensure vehicle is motionless on power & GCS doesn't indicate "Bad Gyro Health"
					GPS	Ensure GPS Lock
					Communication Check	Mavlink packets received: **>90%,** No loss of communication
					Vehicle Power on GCS	Ensure autopilot configured to display mAh drawn and voltage on GCS (if used)
					Load Flight Plan	Verify way point altitudes/radius, flight path, and Home and rally positions.
					Geo-Fence	Set geo-fence boundaries & switch (min altitude, max altitude, vertical bounds)
					AutoPilot Settings	Verify Cruise Airspeed, Airspeed/Bank Limits, Min/Max throttle settings
					Failsafes	Check RC failsafe is turned on and working (turn off transmitter while powered)
					RC Control	Check Surface deflection direction, rates, and trims
					Propeller Direction	Check propeller(s) tight and in the correct position and orientation.
					Check Mode switch	Stabilize/Position Hold/Altitude Hold/Guided/Auto
					Calibrate Accelerometer	Perform single axis calibration
					Calibrate Airspeed (Plane Only)	Zero; Check for airspeed response
					Stabilize mode (Plane Only)	Check surface response to vehicle Pitch/Roll/Yaw
					Check Video	Transmitter/receiver, video clear & no **degradation in comm (>90%)**

APPENDIX B: TEST POINTS AND FLIGHT PROFILES

Test Points

Test Points		Flight Profile			Flight No.
S/No	Details	Alt	Flight Time	Notes	
0	Test Point 0	10-25m			1
0.1	Takeoff X-8	0-10m			1
0.2	Hold at 10m, begin algorithm	10m			1
0.3	Switch to guided mode	10m		Flight floor set to 20m	1
0.4	Observe multirotor behavior	10-25m		Vehicle should rise beyond 20m flight floor	1
0.5	Switch to manual mode and recover X-8	0-25m			1
1	Test Point 1	10-50m			
1.1	Takeoff X-8s	15m		Flight floor will be 10m	
1.2	Switch lead to altitude hold	15m			
1.3	Switch follower to guided	15m			
1.4	Fly leader in straight line away from follower	15m			
1.5	Fly leader in straight line toward follower	15m			
1.6	Fly leader in a box or circle pattern	15m			
1.7	Switch both aircraft to manual and recover	0-15m			
2	Test Point 2	10-50m		Similar to Point 1, 3 vehicles	
3	Test Point 3	15m		No pilot inputs required, testing oscillation behavior of swarm for various parameter sets	
4	Test Point 4	10-50m		Similar to #2, except lead is following autopilot route in auto mode	

Appendix D. Flight Test After Action Report

Capt Corey Combs
5 Nov 18

AFTER ACTION REPORT

Per ENOI 91-6, the After Action Report should be submitted to the FTSO no later than 7 calendar days from the completion of the test.

1. Use this section to briefly describe how the test was carried out. Were there any unusual events?

The first day of flight testing was Monday, 29 Oct, 2018. A total of 5 flights of the swarm, encompassing 9 sorties, occurred in the afternoon in winds of 5-15 knots. All flights were executed without incident. The first flight was used to confirm the flight floor rule functioned as required with only one vehicle. A second flight was performed with one vehicle airborne and another on the ground being held by a member of the test team. It was during this flight an abnormal behavior resulted with the airborne vehicle progressing through the flight floor toward the vehicle being held. A recovery to a safe altitude was performed by the safety pilot and this behavior was never observed during any following flight. Deeper analysis is required into the flight logs to determine the source of this anomaly.

The third and fourth test flights involved 2 vehicles airborne running the control algorithm. The lead aircraft was controlled by the safety pilot and the follower was set to guided mode. A maximum velocity of 2 m/s, separation of 10m, flight floor of 10m, and broadcast rate of 5 Hz was used for the third flight. During this flight, the lead aircraft was able to move faster than 2 m/s since it was not under full control of the algorithm and it was noted that the follower's actions were sluggish. Typical response time for the follower was 5-7 seconds after the lead had moved positions. The aircraft were recovered and the broadcast rate was increased to 10 Hz. With this change, the follower aircraft was more responsive, in the region of 3-5 seconds after the lead had moved positions. Aside from the lag in the response time, the follower vehicle would move toward the lead vehicle's position and maintain separation. Ground monitoring of the LCM traffic showed both aircraft broadcasting properly, although further analysis is needed to determine the exact number of received and/or dropped messages.

The fifth and final test for the day involved all 3 vehicles airborne running the control algorithm. Similar behavior in the follower aircraft were observed with a broadcast rate of 10Hz, the response was 4-6 seconds. It appeared to the ground crew that the 2 followers were more responsive to one another than the lead aircraft under manual control. The aircraft did appear to swarm successfully when the followers caught up with the lead aircraft.

Day two of testing was 30 October, 2018 in which a minimum of 15 swarm flights occurred, generating more than 40 sorties. Winds were calm to start the day, with observed winds between 5-10 knots later in the day with some gusts to 15. The exact number of sorties for the day must be determined when the logs of the vehicles are processed in conjunction with the ground station logs. All flights were successful in providing data, with some anomalies and some successful swarming observed. Various tests were run with 2 and 3 vehicles airborne, with variations in the maximum velocity between 2 and 3 m/s, separation of 10 and 5 meters, a "b" value of 0.5, 0.25, & 0.1 and update frequency of 10 and 5 Hz. There were no collisions with vehicles under algorithm control, however there appeared to be some interactions within the separation bubbles.

The majority of tests were completed with one lead aircraft in "AUTO" flying the same waypoint pattern to keep consistent. Follower aircraft were not always positioned the same when the flights were started and some early flights led to communication drops or swarm behavior that was unexpected. The largest issue of the day was something also observed in prior simulation for which a root cause has not been identified with reasonable conclusion. In this instance, the algorithm will still broadcast the vehicle's position, however the velocity commands that should be driving the aircraft do not appear to be received by the autopilot. As a result, the vehicle will

remain stationary and not follow the other members. This was most common with the 2nd vehicle. In these cases, the 3rd vehicle still appeared to perform as expected, maintaining position relatively halfway between the leader and the stationary 3rd vehicle.

As the values were altered for the algorithm, it was observed by the ground team that a higher maximum velocity and lower "b" value provided the "smoothest" responses. Further analysis will be completed on the data to quantify this response. Other notable occurrences during flights included:

- Increasing amplitude of an oscillatory behavior in one of the follower vehicles
- Sluggish, delayed response of follower vehicles when all aircraft were limited to the same maximum velocity (3-5s)
- Double message rates recorded via LCM traffic, i.e. broadcast rate of 10 Hz produced 18 Hz traffic and dropped messages
- Several instances where vehicles appeared to be within the separation bubble, however this could have been due to 3D perception of the team on the ground

A final observation was the first field testing of a new ground control software, UgCS. The benefit of this software over Mission Planner is the ability to control and observe multiple vehicles from the same ground station and reduce manpower requirements for a large swarm test. Dr. Jacques was the ground station operator who had previously not used the software in an extensive manner. After a brief overview and spending 2 days using it, he felt proficient in navigating the interface and was comfortable the data it provided. There were some items during the testing that are important to note:

- Vehicle 2 on the first day had many drops with the ground station software. For day 2, the 915 MHz transceiver pair was swapped for a spare set and the issue was resolved.
- Vehicle 3 dropped from the ground station during 2 flights, but rejoined during one of them. The other flight it was not visible from the beginning of the flight.
- Powering the USB hub appeared to improve comms, but all 3 915 MHz transceivers to the ground station were going in one port on a laptop through a single extension cable. Perhaps a laptop with multiple USB ports would be better suited so each transceiver had a unique path to the computer.
- Configuration of the parameters in the autopilot is still be done with Mission Planner.
- A screen with 1920x1080 resolution is best for viewing the small text on the UgCS interface. Higher resolution results in difficult to read text.
- A config file should be used to tell the software which aircraft is on a specific COM port in Windows 10.
- Telemetry logs are stored and viewable within the software, which provides some immediate feedback and a log of information if the test team needs to review a flight.

2. What test execution/safety lessons were learned during the test event?

Several useful test execution and safety lessons were learned throughout the 2 days of testing. There were no major safety issues throughout the days of testing, but the lessons are useful to future testing.

- Ensure preflight checks are completed with a knowledgeable faculty and not solo by the student. One aircraft had a failsafe still engaged that is normally not used by AFIT which resulted in loss of an aircraft on a landing approach. (specifically RTL on low battery and proper low battery voltage failsafe)
- Practicing Crew Resource Management (CRM) was crucial with multiple vehicles airborne in close proximity to one another.

- Staggering takeoffs was beneficial to the safety pilots. The best sequence of starting a test flight was the lead aircraft to take off, then the followers take off. The lead would switch to AUTO, then in 2-3 seconds the followers would switch to GUIDED.
- Having an experienced crew with few distractions provided for a smooth day of testing in addition to having pre-determined data points to be collected. Flexibility and patience were crucial in a successful day.
- Troubleshooting comms for the Mesh Network needed to be better, as the second day there were issues with the LCM message decoding on the ground node. However, the Wi-Fi link appeared to be stable and remain connected throughout all test flights.
- After some flights the Beagle Bone was not restarted and this occasionally resulted in issues restarting the control algorithm for the next flight. However, it was not an issue on every flight, so additional troubleshooting of the comm network is required.
- Bring spare airframes in case of mechanical issues.
- Having color identified lights on each vehicle allowed the safety pilots an additional level of recognition of their vehicle.
- Clean radio traffic, with efficient communication of intent was practiced. Additionally, verbal communication was used when the safety pilots were in close proximity and fast intent was desired.
- Maintaining control of the testing environment as the Test Director through effective communication of intent and a plan to follow for the day allowed for the high number of sorties. Within the last few hours of the day, over 20 sorties were accomplished according to plan with minor anomalies with the issues addressed above.

Appendix E. Additional Figures

Bibliography

Timothy J Allen. Design and Test of a Uav Swarm Architecture over a Mesh Ad-Hoc Network. Master's thesis, Air Force Institute of Technology, 3 2018.

Janna Huuskonen and Timo Oksanen. Soil sampling with drones and augmented reality in precision agriculture. *Computers and Electronics in Agriculture*, 154: 25–35, 2 2018. ISSN 01681699. doi: 10.1016/j.compag.2018.08.039. URL `https://doi.org/10.1016/j.compag.2018.08.039`.

Dan Gettinger. Drones in the Defense Budget. Technical report, Bard College, 10 2017. URL `http://interactive.africandefence.net/dronesinthedrc/?mc_cid=5a72d197e1&mc_eid=b57909e368`.

Darryl K Ahner and Carl R Parson. Workshop Report: Test and Evaluation of Autonomous Systems. Technical report, STAT T&E Center of Excellence, 2016. URL `https://www.afit.edu/stat/doclib.cfm?dl=105`.

Matthew Clark, Jim Alley, Paul Deal, Jeffrey DePriest, Eric Hansen, Connie Heitmeyer, Richard Nameth, Marc Steinberg, Craig Turner, Stuart Young, Darryl Ahner, Kelly Alonzo, Barry Bodt, Pete Friesen, Jim Horris, Jonathan Hoffman, Kerianne Gross, Laura Humphrey, Marshal Childers, and Michael Corey. Autonomy Community of Interest (COI) Test and Evaluation, Verification and Validation (TEVV) Working Group: Technology Investment Strategy 2015-2018. Technical report, DoD, 5 2015. URL `http://www.dtic.mil/dtic/tr/fulltext/u2/1010194.pdf`.

Jeremy Gray. Design and Implementation of a Unified Command and Control Architecture for Multiple Cooperative Unmanned Vehicles Utilizing Commercial off the Shelf Components Thesis. Master's thesis, Air Force Institute of Technology, 3 2015.

Jacob L Lambach. Integrating UAS Flocking Operations with Formation Drag, Thesis. Master's thesis, Air Force Institue of Technology, 3 2014.

Jennifer N Kaiser. Effects of Dynamically Weighting Autonomous Rules in an Unmanned Aircraft System (UAS) Flocking Model. Master's thesis, Air Force Institute of Technology, 3 2014.

Luis A Cortes. Best practices for highly effective test design ; Part 1 Beginners guide to mapping the T & E strategy. Technical report, STAT T&E COE, 2014a.

Defense Science Board. The Role of Autonomy in DoD Systems. Technical report, DoD, 7 2012.

Ruth A David and Paul Nielsen. Defense Science Board Summer Study on Autonomy. Technical report, DSB, 6 2016.

Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics*, 21(4):25–34, 1987. ISSN 00978930. doi: 10.1145/37402.37406. URL http://portal.acm.org/citation.cfm?doid=37402.37406.

Tom Standage. Commercial drones are the fastest-growing part of the market. *The Economist*, pages 6–13, 2017. ISSN 00130613. URL https://www.economist.com/news/technology-quarterly/21723003-most-drones-today-are-either-cheap-toys-or-expensive-weapons-interesting.

T. Hovhannisyan, P. Efendyan, and M. Vardanyan. Creation of a digital model of fields with application of DJI phantom 3 drone and the opportunities of its utilization in agriculture. *Annals of Agrarian Science*, 16(2):177–180, 2018. ISSN 15121887. doi: 10.1016/j.aasci.2018.03.006. URL https://linkinghub.elsevier.com/retrieve/pii/S1512188718300848.

Peter Liu, Albert Y. Chen, Yin Nan Huang, Jen Yu Han, Jihn Sung Lai, Shih Chung Kang, Tzong Hann Wu, Ming Chang Wen, and Meng Han Tsai. A review of rotorcraft unmanned aerial vehicle (UAV) developments and applications in civil engineering. *Smart Structures and Systems*, 13(6):1065–1094, 2014. ISSN 17381991. doi: 10.12989/sss.2014.13.6.1065.

Bryan Hubbard, Heng Wang, Michael Leasure, Tim Ropp, Tamara Lofton, Sarah Hubbard, and West Lafayette. Feasibility Study of UAV use for RFID Material Tracking on Construction Sites. In *51st ASC Annual international conference proceedings*, pages 669–676, 2015.

Edgar P. Small. Opportunities for UAVs in construction planning, performing and contract close-out. In *Creative Construction Conference 2018 - Proceedings*, pages 595–601, 7 2018. ISBN 9786155270451. doi: 10.3311/CCC2018-079. URL https://repozitorium.omikk.bme.hu/handle/10890/5741.

Tarek Rakha and Alice Gorodetsky. Review of Unmanned Aerial System (UAS) applications in the built environment: Towards automated building inspection procedures using drones. *Automation in Construction*, 93:252–264, 3 2018. ISSN 09265805. doi: 10.1016/j.autcon.2018.05.002. URL https://doi.org/10.1016/j.autcon.2018.05.002.

John T Feddema, Rush D Robinett Iii, and Raymond H Byrne. Military Airborne and Maritime Application for Cooperative Behaviors. Technical report, Sandia National Laboratories, 9 2004.

David Smalley. News: Autonomous, swarming UAVs fly into the future - Office of Naval Research, 2015. URL https://www.onr.navy.mil/en/Media-Center/Press-Releases/2015/LOCUST-low-cost-UAV-swarm-ONR.

Naval Today. Raytheon gets $29m for work on US Navy LOCUST UAV prototype — Naval Today, 2018. URL https://navaltoday.com/2018/06/28/raytheon-wins-contract-for-locus-inp/.

UAS Task Force: Airspace Integration Integrated Product Team. Unmanned Aircraft System Airspace Integration Plan. *Department of Defense*, pages 1–25, 3 2011. doi: 1-7ABA52E.

S. Drake. Converting GPS Coordinates to Navigation Coordinates. *System*, page 24, 2002.

G. Vásárhelyi, Cs Virágh, G. Somorjai, N. Tarcai, T. Szörényi, T. Nepusz, and T. Vicsek. Outdoor flocking and formation flight with autonomous aerial robots. *IEEE International Conference on Intelligent Robots and Systems*, pages 3866–3873, 2014. ISSN 21530866. doi: 10.1109/IROS.2014.6943105.

Sabine Hauert, Severin Leven, Maja Varga, Fabio Ruini, Angelo Cangelosi, Jean Christophe Zufferey, and Dario Floreano. Reynolds flocking in reality with fixed-wing robots: Communication range vs. maximum turning rate. *IEEE International Conference on Intelligent Robots and Systems*, pages 5015–5020, 2011. ISSN 2153-0858. doi: 10.1109/IROS.2011.6048729.

Development Team PX4. Pixhawk 1 · PX4 User Guide, 2019. URL https://docs.px4.io/en/flight_controller/pixhawk.html.

Marcin Odelga, Paolo Stegagno, Heinrich H. Bülthoff, and Aamir Ahmad. A Setup for multi-UAV hardware-in-the-loop simulations. *2015 Workshop on Research, Education and Development of Unmanned Aerial Systems, RED-UAS 2015*, pages 204–210, 2016. doi: 10.1109/RED-UAS.2015.7441008.

Widyawardana Adiprawita, Adang Suwandi Ahmad, and Jaka Semibiring. Hardware In The Loop Simulator in UAV Rapid Development Life Cycle. *Icius*, pages 1–6, 2007. doi: arXiv:0804.3874v1. URL papers2://publication/uuid/1E3A6CEF-59D9-4C27-920D-3AE1716D8E9C.

Stephen Balakirsky, Chris Scrapper, and Elena Messina. Hierarchical control and performance evaluation of multi-vehicle autonomous systems. *Proc. SPIE*, 5804: 13–22, 5 2005. ISSN 0277786X. doi: 10.1117/12.603772. URL http://dx.doi.org/10.1117/12.603772.

I.I. Kaminer, O.A. Yakimenko, V.N. Dobrokhodov, M.I. Lizarraga, and A.M. Pascoal. Cooperative control of small UAVs for naval applications. *43rd IEEE Conference on Decision and Control (CDC)*, pages 626–631, 2004. ISSN 0191-2216. doi: 10.1109/CDC.2004.1428714. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1428714.

Nick Anderson, Bryan Hagenauer, Ronnie Erickson, and Subodh Bhandari. Flight-Testing of a UAV Aircraft for Autonomous Operation using Piccolo II Autopilot. In *AIAA Atmospheric Flight Mechanics Conference and Exhibit*, pages 1–15, 8 2008. ISBN 9781563479458. doi: doi:10.2514/6.2008-6568.

John M Colombi. 8 May 2017 MEMORANDUM FOR ENV EN IN TURN FROM : AFIT Department of Systems Engineering and Management SUBJECT : Response to Systems Engineering Review Panel Review 1 . Per EN OI 36-117 , the Systems Engineering (SE) program conducted an External Program, 2017.

Luis A Cortes. Best practices for highly effective test design ; Part 2 Beginners guide to design of experiments in T & E. Technical report, STAT T&E COE, 2014b.

Douglas C Montgomery. *Design and Analysis of Experiments*. Wiley, 9 edition, 2017. ISBN 978-1-119-11347-8.

Csaba Virágh, Gbor Vásárhelyi, Norbert Tarcai, Tams Szörényi, Gerg Somorjai, Tams Nepusz, and Tams Vicsek. Flocking algorithm for autonomous flying robots. *Bioinspiration and Biomimetics*, 9(2), 2014. ISSN 17483190. doi: 10.1088/1748-3182/9/2/025012.

Jorge Gonzalez. MFR AFIT-R0006 X-8 Multirotor.pdf, 2014.

Corey Combs. UgCS Operator Workload Study. Technical report, Air Force Institute of Technology, 2018.

Smart Projects. Ardupilot VSM User Guide, 2015.

Justin R. Rufa and Ella M. Atkins. Unmanned Aircraft System Navigation in the Urban Environment: A Systems Analysis. *Journal of Aerospace Information Systems*, 13(4):143–160, 2016. ISSN 2327-3097. doi: 10.2514/1.I010280. URL `http://arc.aiaa.org/doi/10.2514/1.I010280`.

Andrew M. Hyslop and J. Sean Humbert. Autonomous Navigation in Three-Dimensional Urban Environments Using Wide-Field Integration of Optic Flow. *Journal of Guidance, Control, and Dynamics*, 33(1):147–159, 2010. ISSN 0731-5090. doi: 10.2514/1.43778. URL `http://arc.aiaa.org/doi/10.2514/1.43778`.

Duane Q Nykamp. The exponential function - Math Insight, 2019. URL `https://mathinsight.org/exponential_function`.

Python Software Foundation. argparse Parser for command-line options, arguments and sub-commands Python 3.7.2 documentation, 2019. URL `https://docs.python.org/3/library/argparse.html`.

Adedeji Badiru. ENOI 31-102, UAS Test Safety Review Process.pdf, 2014.

Math Open Reference. Median of a triangle - math word definition - Math Open Reference, 2018. URL https://www.mathopenref.com/trianglemedians.html.

MIT. RobotLocomotion, 2019. URL https://github.com/libbot2/libbot2.

SPH Engineering. How to use utilities to convert TLM into CSV or KML, 2018.