PADUA RADIOLOGICAL SEARCH SYSTEM

A Dissertation Presented for the
Doctor of Philosophy
Degree
The University of Tennessee, Knoxville

Benjamin Lajos Magocs
December 2019

Copyright © 2019 by Benjamin Lajos Magocs.
All rights reserved.

ACKNOWLEDGEMENTS

There are numerous people to thank for assisting me in getting to the point at which I have arrived. First, I would like to thank the professors and teachers who gave me the skills and knowledge to succeed academically. From elementary school through graduate school, each instructor has taught me something that helped me get to where I am today. I would also like to thank my advisory committee specifically: Dr. Robert Bond, Dr. Matthew Cook, Dr. Zhili Zhang, Dr. John Schmisseur and my co-advisors Dr. John Auxier and Dr. Howard Hall. To my family and friends who have believed in me and helped me along the way I would like to thank all of you as well. There are a few people I would like to thank in particular though. First and foremost, I would like to thank my dad for helping me in everything I have done growing up both academically and in life. Secondly, I would like to thank Pastor Pat for always believing in me and holding me to a high standard. Last but not least, I would like to thank Amy Shuang Han who always believed in me and proved to be an excellent motivation to work and study hard to finish my degree as fast as humanly possible.

ABSTRACT

The primary intention of this project was to improve upon the current state-of-the-art radiological detection methods by integrating autonomous control methods, Bayesian statistics, Savitzky-Golay filtering, other search methods, and novel swarming techniques together to create a single push-button multi-agent radiation search algorithm (the PADUA algorithm). The primary agents the algorithm was designed for onboard usage with were multirotor-type unmanned aerial vehicles (UAV's) and, in certain portions of search, fixed wing UAV's. The development of PADUA for this project was performed entirely in simulation using the Python Dronekit SITL API. The final algorithm performed exceptionally well with the ability to narrow down the predicted location of a large radioactive source to only a few meters or less from a space that was 1 sq. km. in less than an hour.

EXECUTIVE SUMMARY

This document outlines the background, design, development, testing, and results of the Peripatetic Aerial Detection Updatable Algorithm (PADUA). While there are no search systems quite like PADUA, there are several previous research efforts that inspiration was drawn from in order to create the search system presented here. At its core, PADUA is a multi-tier, multimodal algorithm that is mostly platform-agnostic and somewhat detector-agnostic but designed primarily for unmanned aerial radiation detection. Furthermore, PADUA operates all major parts of the search system (control, communications, and the search methods) and allows for both single and multi-aircraft capabilities.

The primary goal of PADUA is to act as a tool to better locate radiological point sources and hot spots. The algorithm was designed with two main situations in mind; the first is the event of a stolen radiological source while the second is that of nuclear fallout from a power plant meltdown, nuclear weapon attack, or radiological dispersion device detonation. Most particularly, PADUA is designed with as few assumptions as possible as to the nature of the radiological target and the environment so that the algorithm is not bound in terms of its ability by its own limitations. In the event of a stolen radiological source, PADUA will attempt to locate the source as fast as possible using all three of its search tiers. In the event of nuclear fallout, all three tiers can still be used, but the first tier's methodology can be used to locate multiple hot spots in descending order of priority. Lastly, while the algorithm has a beginning and end in the testing instances shown later in this document, in practice the algorithm is designed only to have a finite Tier 1 and Tier 2 search with the possibility of a limitless Tier 3 search. PADUA was designed this way specifically so that if a stolen source is being targeted, the ability remains to try to locate the source if it is moving.

The primary data utilized for all the search systems in PADUA is radiation count rate. While the detector used for the development of PADUA was a plastic scintillator, any type of detector that converts its detected signal into a number of counts per unit time (i.e. count rate) can be used with PADUA. One assumption that is made in the algorithm's development though is that the primary data being detected is radiation, which tends to follow an inverse square response-per-distance change. If this is not necessarily the case (such as what might be found with chemical weapon signatures), then some modifications might want to be made to certain parts of the algorithm. This is especially true for the Bayesian statistics search portion in Tier 2 that utilizes a Normal Gaussian Distribution intended primarily for inverse square response changes.

In PAUDA, each of the tiers operates according to its own set of control guidelines and search methods. The framework of PADUA operates on a triple-tier search system where each progressive search tier is a smaller more refined region

centered on the previous tier's calculated hot spot. The first tier uses either a code-generated pattern flight path or a user-defined flight path that is generated prior to the algorithm's initialization that is based on waypoint-to-waypoint control. In the first tier, the search in performed in such a way that the ground station sends control commands to the primary aircraft and the primary aircraft sends count rates tagged with location data (i.e. latitude, longitude, and altitude) back to the ground station. In the event that a multi-aircraft/swarming search is being performed, the secondary aircraft(s) relay their respective location and count rate data back to the ground station. The search method used in Tier 1 is a Savitzky-Golay filtering where the primary hot spot is determined as the global maximum of the Savitzky-Golay values. As mentioned, capabilities are included to be able to locate multiple hot spots in Tier 1, as the situation requires. The second tier performs control in the same manner as the first tier but using only a geometric code-generated flight path waypoint sequence. The center of the Tier 2 start path is centered on the Tier 1-calculated hot spot. In the occurrence of Tier 2, the search method consists of a combination of both Savitzky-Golay and Bayesian statistical methods. Rather than using a global/local maximum method as in Tier 1 though, since it is assumed that there should be a source within the confines of the Tier 2 search region, a weight average mechanism is instituted. Thus, the generated Savitzky-Golay values are applied as weights to each of their corresponding detection locations. The Bayesian probabilities can be treated the same way or the global maximum can be used (there is little difference if the source is obvious though). Once the Savitzky-Golay and Bayesian hot spots are determined, their locations (latitude and longitude) are averaged together to generate the final hot spot location for Tier 2. It is important to note that if the Bayesian and Savitzky-Golay individual hot spots for a detector are not near each other, then this would be a likely indicator that either the source is small and difficult to find or there is no source present within the Tier 2 search region. With both Tiers 1 and 2, if a swarming approach is used, then the hot spots are calculated in the same manner for each individual aircraft's data. During PADUA testing, the hot spot locations for all of the aircraft are averaged together at the end of a tier to determine the final tier hot spot before the system progresses to the next tier. Lastly, though, Tier 3 operates in a significantly different manner than the first two. Instead of using ground control guidance and allowing for multi-aircraft searching, the third tier only is designed for single aircraft searching and enables fully autonomous capabilities for the aircraft in question. Using the detector (i.e. radiation detector) as its sensory equipment, the aircraft performs its flight by implementing the novel Right Angle Turn (RAT) Method without any preliminary flight path. Using a perimeter averaging method to locate the approximate centroid progressively, the third tier is utilized in order to both verify and pinpoint the location of the target radiation source.

The name of this algorithmic system describes the framework for how it operates. "Peripatetic" can be defined in adjective form as "travelling from place to place, in particular working or based in various places for relatively short periods[1]."

PADUA operates using a methodology based on the manipulation of small amount of information in using optimized methods according to the situation. In other words, as aircraft(s) obtain count rate data in the first tier, it can be assimilated to performing numerous small 'jobs' in a specific instance over numerous location points as fast as possible. Once the first tier is completed, then a new set of small 'jobs' are performed for the second tier in the same manner as the first but with a new search system. The third tier behaves slightly differently than the other two but still follows the same *peripatetic* approach. Additionally, the system is of course designed primarily for use with aircraft (although certain options are built-in to the algorithm to allow for moderate ease if the control methods need to be switched for usage with other types of vehicles). While PADUA is designed with the specific interest of radiation detection in mind, as discussed later in this document, there is the ability for the algorithm to be slightly modified in different instances for it to be implemented in conjunction with other types of 'non-directional' detectors. The last part of the PADUA name is perhaps the most important when applying the system to real world situations: "Updatable Algorithm." PADUA was developed as an object-oriented code in the open-source Python language giving it the flexibility to evolve and specialize to each situation it is applied to accordingly. With its object-oriented approach, PADUA is made with the options to both remove/change search, control, and communications methods and add more of them if needed. Furthermore, because high-level threading is relied upon to ensure asynchronous data control throughout the algorithm on both the aircraft(s) and ground station sides, not only do no data queues or 'wait' statements need to be inserted if search/control methods are updated in the future, but the flexibility of what types of communication hardware are used with PADUA is also intrinsic to the system. It is not a single search method that makes this algorithm important. All of these parts from the overall search/control system to the vehicles the algorithm is designed for and the precautions taken to ensure the final design is flexible and updatable go into the integral framework of PADUA.

Perhaps the hardest part of PADUA to grasp is the measurement of its ability. While there are numerous instances throughout this document describing the accuracy of both the individual tiers and the complete algorithm's multi-tier operation, there is very little in the way of being able to compare how accurate the system is with respect to other systems. While the most similar system is one developed by the University of Bristol in the United Kingdom, it still is not nearly as versatile as PADUA. The system developed at the University of Bristol tested a number of different search mechanisms (including versions of Bayesian) intended for mapping radiation in a rural environment after a power plant meltdown and does not emphasis as much on searching in a time-constrained manner (such as in the instance of a stolen source). Furthermore, although in a power plant meltdown setting, the University of Bristol's mechanism is likely to operate better than PADUA, it likely would not in an urban environment since mapping is not nearly as accurate when large objects and high variations in background measurements are

present. To map a radiation flux field in a rural environment, it is sufficient to use Bayesian statistics to predict the approximate flux between sampled locations, but in a dense urban environment, it is far better to give emphasis on the areas near high likelihood locations and nearly neglect the effects of low likelihood ones; this is why a weighted average method is often used in PADUA rather than mapping techniques (although because of the updatable characteristic built into PADUA, mapping could be added if desired). Lastly, while the system developed at the University of Bristol works excellently for the missions that it was designed for, it does not allow for nearly as autonomous of a system as PADUA. PADUA not only can be used as a push-button system, but for both Tier 3 and for all secondary/follower aircrafts in swarming configuration, their flights are fully autonomous. [2]

The last part of the development of PADUA consisted of the development of the swarming mechanisms. Two different swarming techniques were used: Pattern and Bound-Continuity swarming. The Pattern swarming algorithm is based solely on the original BOIDS swarming algorithm and simplifies the theory further by implementing waypoint-command control capabilities available in flight controller firmware today. Given an initial target bearing and distance from the lead aircraft and with respect to the lead aircraft's flight direction, the follower aircraft is able to continually update its internally generated flight commands based on the data it receives from the lead aircraft. For the Bound-Continuity swarming algorithm, a much more detailed algorithm is implemented that combines fixed distance bounds on the follower aircraft to limit its separation and cohesion aspects with respect to the lead aircraft and an alignment mechanism derived from the Navier-Stokes Steady-State Incompressible Continuity Equation. A few important discoveries were made with the implementation of these two swarming methods. First, especially for a slightly more complex swarming algorithm such as the Bound-Continuity method, it is imperative that the communications hardware has low-latency. Also though, it was found that it is not necessarily true that swarming aircraft systems will increase the speed of the search or that the search will be more accurate (these are both especially true in a more restricted urban environment where large separations within the swarm are not possible). Instead, what was found was that, using the PADUA triple-tier methodology, swarming will simply increase the chances that a source is found in the first place.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION AND GENERAL INFORMATION

General Background

"Be extremely subtle, even to the point of formlessness. Be extremely mysterious, even to the point of soundlessness. Thereby you can be the director of the opponent's fate"—Sun Tzu, Art of War.

While it has always been critical to a nation's wellbeing, surveillance is perhaps more critical in today's world than ever. As the technology and methods of the intelligence and military communities continue to advance, so does the technology, ability, and cunning of the enemy. With numerous radical groups scattered throughout the world[1], continual narcotics manufacturing and smuggling, and unstable governments, there is an ever-present need to detect critical threats arising from multiple sources with different intentions. Thus, multi-modal detection offers the opportunity to survey many of these threats from a single package.

Furthermore, many currently used systems have extravagant costs and operate in only very limited types of environments. For example, one of the most common tools being used currently for locating stolen or missing radioactive material is the radiation portal monitor (RPM)[2]. These are often placed at ports of entry and roadway checkpoints (such as at a national border) for detection of illicit smuggling through vehicular or pedestrian means. For cargo checks, the unit cost is just over $300K with over $10K in yearly upkeep costs. Newer versions of the detectors that use spectroscopic means for detection (Advanced Spectroscopic Portals or ASP's) will have an even higher unit cost of well over $800K with yearly costs possibly approaching $100K[3]. While RPM's are somewhat effective in their mission and at the least provide a measure of security, the expenditures are exceptionally high considering the detector is static and can only be used to monitor commercial traffic. With some knowledge of an area, bypassing these detectors would not be impossible at all for a dedicated thief or terrorist in many situations.

Overall though, the key point to remember is that terrorism as well as enemy and rogue states truly necessitate that even today, nuclear security is a requirement for any targeted nation that wishes to remain sovereign and stable. Mal-intentioned instances of smuggling regarding radiological materials are still relevant today. This is shown plainly in Figure A.1 where the data constructed by the IAEA Incident and Trafficking Database is displayed to represent the number of *known malicious* instances of stolen or lost nuclear material. All figures are displayed in the Appendix A: Figures. This does not include the numerous other occasions of

nuclear material being lost or stolen that were not malicious or the intentions were unknown.

However, while current defense and surveillance methods are functional, more efficient and effective systems can be developed by combining autonomous vehicle control with detection methods into a single algorithm-based package. This project seeks to prove that such a system is not only able to be developed but also that such a system can be designed in such a way that the resulting algorithm can be applied to low-cost open source electronics and aerial vehicles. This last part of the task means that wireless communication data transfers must be designed to minimize data size since latency of low-grade systems must be considered, and simplicity of working sub-algorithms must be prioritized so low-grade companion computers are able to run the required calculations.

Background of Bayesian Search Research at the Institute for Nuclear Security

Over the past several years, the Institute for Nuclear Security has made large strides in the research of utilizing Bayesian Statistics for rapid radiological material search. These efforts have progressed rapidly from mere theory in 2014[4] to the present research that is using and building on the theories already proven to create more efficient and powerful systems.

BASBP and Early Efforts

BASBP (or Broad-Area Search Bayesian Processor) was the initial effort by the Institute for Nuclear Security (INS) to improve radiation detection with the implementation of more sophisticated statistical methods than were used at the time. Although Bayesian statistics had been used in radiological material search instances at times, the primary method used for ground-based radiation searches at the time simply consisted of soldiers fanning out with radiation detectors and sweeping an area. This method contained multiple problems ranging from the excessive time required for the search to the fact that the method puts soldiers into potentially dire situations and excessively dangerous environments.

Originally, BASBP was written as a FORTRAN code and tested against Monte Carlo N-Particle (MCNP) models. Later, in the initial parts of the project, further proposals were made and testing was performed to implement the code onto an unmanned aircraft. Theoretical search simulations were performed with set sample locations in perfect grid patterns. While this served to prove the Bayesian-based search algorithm could locate a known source at an offset distance away from it, further research was required for code refinement and more realistic testing. [4]

In the second part of the BASBP project, two main research steps were taken with the original BASBP code. First, rather than leaving the code in the FORTRAN language, it was translated into Python 2.7. Since Python is a higher-level open source language, this served to enable easier manipulation and updating of the code. In addition, since Python is far more common than FORTRAN, the translation of the BASBP code's language also meant that more people would likely be able to work on the improvements to the algorithm in the future. The second aspect of the research project was to prove that the algorithm had functionality in the real world. This consisted of simplifying the code and flying a large unmanned aircraft to perform a simulated search. This was done by coupling a Raspberry Pi on the aircraft with the BASBP code and a GPS. Also on the Raspberry Pi was simulated flux data developed for the search region by MCNP. [5]

SWARM

After the BAPSB project was completed, the INS received a grant to proceed further with Bayesian search efforts by refining the search algorithm and implementing a search method with multiple aircraft. The SWARM (Semi-autonomous Wide Area Radiological Measurements) project consisted of improving and simplifying the Bayesian search method, but it also included far more complex coding needed for the integration of other search methods, multi-aircraft control, multi-detector data compiling and analysis, and simulating the algorithm on a computer and in the field. The end-goal of SWARM is to have a more versatile algorithm capable of searching larger areas than BAPSB. To incorporate all the different parts of the system into a single low-computation-cost platform that can easily be updated for future usage, Python is again being used but with a heavily object-oriented approach compared to the more scripted approach used is the past.

A Background History of Autonomous Flight

Early UAV's and Targeting Drones

While most people are familiar with modern usage of military UAV's, unmanned flight is not a modern concept by any means. The first designs and construction of UAV's began not in Middle Eastern wars or Vietnam but rather in WWI with the Kettering "Bug" designed and build by Orville Wright and Charles Kettering[6].

While perhaps not considered to be a true UAV in terms of how they are thought of today, the "Bug" was intended to be an autonomously delivered bomb. Figure A.2 shows the basic structure of the "Bug" as a light biplane-type aircraft with a small four-cylinder Ford engine and a 180 pound bomb[7]. Rather than just

a turnkey system where the aircraft's engine was turned on and then the vehicle was launched to fly randomly, the principle behind the weapon was instead to calculate how many engine revolutions would be required for the vehicle to reach its target upon knowing the distance to the target, wind speed, and wind direction. The engine was designed to shut off once the set number of revolutions was reached by dropping a small cam that would not only cause the engine to shut off but would also cause the wings to release and allow the weapon to free fall kinetically to its target[8].

While the "Bug" can loosely be considered to be the first true UAV effort, its principles and mission are most definitely forbearers of modern cruise missiles. The "Bug" never did see combat and research was halted due to lack of funding soon after the war, but the idea of a self-delivering aerial weapon as well as an aerial vehicle that did not have to fly itself would not die altogether[7]. In fact, only a few years later in WWII that both sides (Allies and Axis powers) began to utilize pilotless aerial vehicles—some with devastating consequences.

The "Bug" was successful insomuch that it showed that a pilotless aircraft was at least feasible (although perhaps only with a very limited mission), but in WWII, the very same principles were combined with more modern technology to create much more elaborate unmanned systems. Perhaps the most infamous of them all were the Vergeltungswaffen—also known as the V-weapons. Used by the Germans primarily against Great Britain during the War, the V-1 and V-2 were pilotless aircraft/cruise missiles that allowed for the bombing of London without the need to place both German pilots and aircraft that are more expensive in harm's way. With their usage leaving a trail of devastation with most of those killed by the V-1 and V-2 being civilians (over 15,000), the V-1 and V-2 refined a host of new technologies and combined them with advanced unmanned control systems to produce the forerunners of many of the systems used today[9].

Neither the V-1 nor the V-2 utilized the propeller propulsion common in the era's aircrafts; rather, the V-1 utilized a pulse jet[10] and the V-2 was rocket propelled[9]. Using more advanced autopilot mechanisms than the "Bug" and other earlier aircraft, the V-weapons were able to not only prove that unmanned systems were effective, but were able to do so with relatively simple low-cost platforms. At the height of the V-weapons' activity, there were over 100 launches per day[11].

Additionally, the use of the V-weapons might be considered one of the first swarm-type aerial attacks. Using similar principles as many modern swarming systems, the V-weapons utilized *quantity delivery.* In other words, rather than relying on a few very advanced weapons to complete their missions with high accuracy and precision, a number of adequate weapons were used to complete a common mission goal. When a swarm operates in this manner, there is often the understanding that some vehicles will likely fail in their missions, but the goal would

still be likely achieved and at a lower cost than using the alternative since some vehicles will still accomplish the task. These concepts will be discussed in more detail in further sections.

While much of the technology from the V-weapons, other weapons, and other aerial vehicles contributed to the advancement of rocketry and aircraft design in WWI and WWII, the relatively slow but continual and steady increase in unmanned vehicle technology continued to grow during this time. From the gyroscopic beginnings of control used in the Kettering “Bug” in WWI to the radio controlled methods used by vehicles in WWII such as the OQ-2 and OQ-3 of which around 15,000 were produced[12], drone control technology continued to advance. However, even by the end of WWII, there was still a severe gap between the theory and technology required for the development of large-scale effective Unmanned Aerial Systems (UAS).

During the Cold War, UAS continued to be further developed into more effective tools and weapons, but perhaps the numerous missile systems that were developed during this time were the most influential to unmanned system technology. Although there were many, a few of the most notable developments during this time that showed the clear progression of relevant unmanned vehicle technology were the TOW anti-tank missile, the Phoenix air-to-air missile, and the Tomahawk cruise missile. While none of the aforementioned weapons are what would be considered traditional UAV’s, they all are capable of flying and completing a mission without any pilot onboard and thus can be used as excellent examples of precursors to modern UAV technology.

Although similar systems were used in WWII (notably the German Fritz X and the Henschel 293 wire-guided bombs), the Tube-launched, Optically tracked, Wire-guided (TOW) missile utilizes what is arguably the first highly effective wire guided missile system. A ground- or tank-launched anti-tank missile, the TOW missile’s speed varies greatly depending the target distance, but older versions flew between 175 and 200 m/s at a range of ~3000m [13]. While not currently applicable to many other types of UAS types, the wire spool guidance method showed that even wired rather than wireless control methods are still useful in certain types of conditions. Wired control methods for UAV’s today primarily include tethered drones, which are usually of multirotor type.

The Phoenix AGM-54 air-to-air missile was one of the most important military developments in the United States for unmanned semi-swarming autonomous air-to-air missile technology. Carried by the F-14 Tomcat, the AGM-54 had a range of well over 100 miles—far outside the naked-eye view of the fighter pilot. This fire-and-forget missile showed that a high-speed (3000 mph+) aerial vehicle was capable of being guided effectively by something other than radio or wires. Using radar guidance as the primary control input, the AGM-54 opened the doors to more

advanced detector or sensor input in complex aerial vehicles.[14] While much of the Phoenix missile remains classified, it signifies an obvious leap in guidance technology and provides evidence of vast increases in computational power compared to only a few decades earlier when radar guidance would have been unobtainable due to computational speed and computer size limitations.

The BGM-109 Tomahawk missile is a subsonic sea-to-land cruise missile currently built by Raytheon[15]. The Tomahawk has a range of up to 1500 miles[16]. Most importantly though, this is one of the most effective long range pilotless aerial vehicles the United States has ever produced with the ability to attack specific targets with high precision and operate in large numbers of individual weapons with variable flight plans/instructions that are modifiable during flight. The key to the successfulness of the missile though is the multi-module guidance system; this consists of a combination of internal guidance (i.e. gyroscope and gyrocompass)[17], GPS[18], terrain following guidance (operating by Terrain Contour Matching equipment that utilized preloaded terrain maps)[18], digital scene preloaded mapping (operated by DSMAC)[18], and radar homing[17]. A system such as this where multiple guidance systems are utilized to more precisely and accurately guide an aerial system is critical to the success of today's unmanned systems. Especially where multiple swarming aerial systems are concerned, multiple types of guidance methods are necessities as short range mechanisms are needed to prevent collisions and long-range guidance systems are required to lead a UAV to its target accurately and efficiently. Redundant communications systems are also required so contact/control between the aircraft(s) and ground station is not lost due to specific types of signals being blocked or jammed. Merging data rapidly and efficiently from different types of devices though can be difficult.

MQ-1 Predator: The Turning Point

While the United States and other nations continued to improve unmanned aerial weapon and vehicle technology throughout the years during WWI, WWII, and through the Cold War, it can be strongly argued that there was one UAV that finally demonstrated that the theories and technology underlying effective UAS had caught up to each other. That aircraft is the MQ-1 Predator.

The Predator was the result of a rather complicated legacy starting with a Jewish Iraqi, Abraham Karem. Born in 1937 in Baghdad and growing up in Israel, Mr. Karem graduated from Technion in Haifa, Israel with a degree in Aeronautical Engineering. Upon graduation, Mr. Karem ended up working for Israel Aircraft Industries (IAI); however, he soon set out to start his own business ventures in a realm of aviation he believed was due for serious advancement—unmanned aerial vehicles. However, as he became increasingly frustrated with the Israeli

government bureaucracy and project management, he eventually realized that his best option would be to continue his work in the United States.[19]

So, Mr. Kedem and his wife immigrated to the United States and began working on DARPA contracts with a small firm in Las Angeles, but he soon ventured out again on his own to create the first truly successful fixed wing UAV and the precursor to the Predator. Working with two others, engineer Jack Hertenstein and pre-med student Jim Machin, the 56-hour endurance Albatross was created. This light unmanned aircraft (only 200 pounds) had a very rudimentary cylindrical form and rather unconventional wing and tail designs with a parasol wing design that maximizes wing area and an inverted v-tail along with a pusher propeller. This aircraft design can be seen in Figure A.3. After impressing DARPA with the Albatross, Kedem was awarded funding to build the next UAV in the series—Amber. With some small changes requested by DARPA, the Albatross was transformed into a similar looking vehicle named Amber. Equipped with a television camera, these two UAV's were the immediate precursors to the Predator. However, with the end of the Cold War in the early 1990's, funding was soon reduced for such projects, so Mr. Kedem sold his company (Leading Systems Incorporated) to General Atomics. However, General Atomics, with the technology from its own products and research combined with the technology developed by Leading Systems, was awarded a contract in 1994 by the United States government to produce a quiet surveillance drone. The Predator first flew in 1996, only a year and a half after the contract was awarded. This proved to be a turning point not only in unmanned flight, but also in aeronautics altogether. [19]

With no pilot or cockpit, large UAV's can be designed with an exceptional amount more freedom compared to conventional piloted aircraft. Without the need to carry a pilot, cockpit, or as much armor, smaller and/or faster vehicles can be produced due to the high reduction in weight revolving around the transporting of the human pilot. Additionally, with the lack of a human on board, a UAV also is able to be designed with lower safety factors in many areas than otherwise[20] and, if designed with proper structural strength, a UAV is also capable of higher G-loaded than the alternative since human pilots are limited physically even with compression suits to approximately 12 G's sustained[21]. With smaller aircraft possible with less weight, higher vehicle loading is also possible since the moments and inertia become lower if an aircraft is constructed smaller compared to a larger pilot-carrying aircraft (which requires a cockpit and due to higher weight needs larger wings and more powerful engines).

The RQ-1 Predator was the first UAV the United States military used that combined numerous modern technologies such as satellite communications, GPS guidance, composites, and multiple types of camera and video surveillance technologies[22]. Although designed from technology developed for a war with the Soviet Union that never materialized, the RQ-1 Predator led the way for those aircraft that came after

it such as the high endurance MQ-4 Global Hawk, the armed MQ-9 Reaper, and the hand-launched RQ-11 Raven among others.

Computational Advances

While the potential advantages for UAV's have always been evident, the key to all modern UAV's is the host of technological advances that have occurred over the past century. Today, these advances have reached a point where many of them can be combined and integrated into far more advanced and effective platforms than ever before. While there are numerous technologies that could be discussed in reference to unmanned systems, a few specific ones are essential to understand the operation of modern drones. Computational advances, improvements to guidance systems, and the onset of composite materials all have played large parts to the making of today's UAV's and are perhaps some of the most important sub-systems used in drone manufacturing and design today.

The integrated circuit can be stated without doubt to be the backbone of modern computers. The invention of the integrated circuit (IC) chip is generally credited to Jack Kilby and Robert Noyce in the late 1950's. Jack Kilby technically first invented the IC while working at Texas Instruments, and Noyce discovered the same invention only six months later[23] before going on to found Intel with Gordan Moore[24]. Gordan Moore is also famous for proposing what become known as Moore's Law, which theorized that every two years the computational power/component density on an IC chip would double. [23]

The IC consists of a microcircuit on a semiconductor. The microcircuit is based primarily on micro transistor technology. A transistor, in the simplest terms, is a modern replacement for a much larger and higher power vacuum tube; instead of using a vacuum to control an electrical potential difference, a transistor normally operates with three leads and a very simple but small circuit[23]. There are numerous types of transistors, but they primarily stem from two main types: bipolar and field effect.

The bipolar junction transistor (BJT) uses a triple lead format with an emitter, base, and collector lead where, if a small current is applied to the base lead, voltage is allowed to flow between the emitter and collector. This process is done in one of two configurations: n-p-n or p-n-p; these identifiers represent the material layout of the transistor. An n-type material is a negatively doped semiconductor while a p-type is a positively doped semiconductor. In either case, the base is the middle lead.

The field effect transistor (FET) operates with three leads as well, but they are source, drain, and gate respectively. A controlling voltage is applied to the gate lead for a FET rather than a current applied to a base lead as with a BJT. Although similar in construction, a FET sometimes will use an oxide substrate at the gate

(hence, a metal oxide semiconductor FET or MOSFET); otherwise, if the materials are entirely non-metal oxide, then the FET is referred to as a junction FET or JFET. Furthermore, because the actual layout of a transistor is basic, combining micro transistors into a single microcircuit can be done simply by a form of printing. This allows for complex circuitry to be made both small and in mass quantities. [25]

The invention of the IC and the fact that the computational power of IC chips did indeed continue to increase rapidly led to one of the keystone elements of modern UAV's. Since the IC is the basis for the computer processor, it is evident how widely this small invention has affected the world. As computational power density increased, required computations for advanced aircraft control, communication, and other onboard control became increasingly possible due to the underlying physical issue of weight. With more less space and weight needed for onboard computers, more computations and less space is required.

To better understand some of the reasons that computational advances have benefitted UAS technology so greatly, the design process of an aerial vehicle should first be discussed. The four main principles in aircraft design are the four opposing forces of weight, lift, drag, and thrust (see Figure A.4). During the design process of an aerial vehicle, solving the problem affecting one of these forces nearly always affect the others. This can be explained in a few simple examples.

For instance, if an aircraft design has a weight increase (for example if a pilot is added to the vehicle), then an increase in lift is required to compensate for the added weight. This can be accomplished by increasing the wing area/span. However, as the wing size increases, the parasite drag will also increase if the aircraft is moving in steady state. Additionally, the weight will increase from both the added wing area and the added structural support from the increased root moment; but ideally, it will increase at a lesser rate than the added lift. This means that an increase in thrust would be required to overcome the adverse contributions of the added drag; however, short of aerodynamic alterations and/or increases in propulsive efficiency, the obvious method of increasing thrust is by increasing engine size. If the size of the engine is increased though, then the weight will increase yet again and the cycle will continue until equilibrium is found between each of the four parts or it is concluded that the original weight-gain simply cannot be incorporated into the design. Alternatively, the cycle also can progress the opposite way. If weight is reduced (for instance, with a UAV, there does not need to be a pilot in the vehicle), then less lift is needed which means the wings can be shrunk thereby decreasing both lift and parasite drag. With smaller wings, reduced reinforcement can be applied to the wing structure since there is a lower moment about the root, which decreases weight yet again, and so on…

These same principles can be applied to UAV's though when it comes to computational hardware. In the 1950's computers were far too large to perform

calculations onboard an unmanned aircraft needed to operate the vehicle to the standards of a conventionally piloted aircraft. However, computer technology continued to become more computationally powerful and the hardware decreased in physical size. Eventually the technology reached a point that could operate an aircraft using computers rather than a pilot with similar or increased mission capabilities. With only sensors, communications hardware, flight computers, and other advanced electronics, high endurance reconnaissance to ground attack missions can now be performed without any pilot onboard an aircraft.

With this importance in weight reduction being understood, the value of smaller computers can be understood clearly for aircraft design, but why are computers so important for unmanned flight in the first place if simple UAS were able to be designed and produced during WWI? While early UAS might have operated well enough at times, realistically many preliminary UAV's were far less than adequate. For example, the Kettering Bug from WWI was designed to be aimed in the general direction of the target and hopefully it would end up hitting somewhere near the mark[26]. Even in the post-WWI eras though, UAV's still often lacked the necessary capabilities to be as effective as desired. For example, even though many of the surveillance drones used the United States during the Cold War and in Vietnam might have been adequate for simple surveillance, the technology of the time still imposed numerous limitations.

One of the most important limitations of early drones was the lack of high quality wireless video communications. Most of the reconnaissance aircraft (both manned and unmanned) throughout the Cold War era utilized film for high quality photographic surveillance[27], [28]. One of the aspects of the Predator UAV that has made it so valuable is its wirelessly transmitted video from a television camera via satellite link[29]. Again though, without the advances in digital circuitry, such capabilities would never have been possible. Digital cameras and high speed processing have allowed high quality video images to be obtained and relayed to a ground station where a remote pilot can fly the aircraft and/or survey the target. This allows for near-real time decision-making, surveillance, tracking, and even targeting with unmanned vehicles. Combining the video technology with satellite communications allows the pilot or observer to be stationed on the opposite side of the planet if so desired.

Digital cameras have significantly altered aerial surveillance and surveying. The first true digital camera was invented by Steven Sasson at Kodak in 1975. Although very rudimentary in design (the invention had a 100x100 pixel resolution and used tape for the recording device) this device was revolutionary. Still though, Kodak decided to continue with film technology rather than embrace digital photography; this was likely due to the vertically integrated business model that Kodak used where each part of the film-based photography process was built, owed, run, and therefore profitable by Kodak. However, while Kodak owned the patent for the

digital camera, it was not Kodak that led the digital revolution. Regardless though, digital photography did progress to not only be equal with, but also surpass film photography. [30]

A digital camera normally operates using a digital image sensor that is one of two types: charge-coupled device (CCD) or complementary metal-oxide-semiconductor (CMOS). Both of these operate using semiconductor-based pixel sensors that transmit the detected image data to either a digital memory device or an analog/digital converter on older digital cameras where tape was used as the storage medium[31]. The reasoning for the significance of the digital camera though comes on many fronts. Firstly, as far as aircraft are concerned, the lack of requiring film storage can save immense space and weight. Additionally, with digital sensors rather than analog ones, the photographs from digital cameras also allow for rapid manipulation and switching from one setting to another. This can be immensely valuable in varying light situations and even varying wavelength situations.

While photographic elements are important, navigational technologies have also been critical to the success of UAS. Arguably, two of the most important navigational aids developed during the rise of UAS have been the gyroscope and Global Positioning System (GPS). Although separated by many years, both are still used today in countless devices.

Using the angular momentum from a spinning wheel(s), a gyroscope allows a set direction to be known. When a wheel spins in a set position, by conservation of momentum, the wheel will resist a change in orientation. By combining three spinning wheels/gyros together, three dimensions of space can be controlled (positive and negative of x, y, and z directions).

GPS is considerably different from gyroscopic control as the control basis is external rather than internal. Instead of utilizing a compass, gyro, or other means, GPS uses an onboard receiver but all emissions and primary timekeeping are done on satellites. GPS utilizes triangulation from satellites to determine the precise location of the device holding the receiver. By identifying the signals from at least four satellites, a GPS receiver is able to use simple trigonometry to find is geographical coordinates with the first two satellites, its altitude with the third, and verify the time with the fourth. Accurate time recording is critical for GPS operation due to the method at which it determines the distance between satellites and the receiver; by calculating the distance that a radio wave can travel in the difference in time between emission and reception, a receiver is able to calculate the distance of the satellite. By using the method of trilateration, GPS emission signals can be used to refine the terrestrial receiver location. GPS trilateration operates on the principle of overlapping radial circumferences. This is shown in Figure A.5 where, if the distance from the receiver to one satellite is known, then there is a circle with

that radius around the receiver of where the satellite might be located. However, if the same thing is done with multiple satellites, then the location of the receiver can be pinpointed. Note that while the figure shows the method with circles, in reality the interchanges are spherical. [32]

The last technology that to be discussed is that of materials. When the first airplane was flown by the Wright brothers in 1903, the primary materials used for aircraft were wood and canvas. However, within only a few years, metal (primarily aluminum) began to be used for aircraft construction. In fact, the first all-metal aircraft was built only twelve years later in 1915 by the Germans as the Junkers J1[33]. While metal airframes were more expensive to build, the higher structural loads possible with aluminum allowed for much higher maneuverability and speed than the previous canvas coverings and wood spars. Furthermore, due to its higher strength, less support wires and beams are required for a metal aircraft to keep its form compared to a wooden one; reducing the number of structural pieces with external contact surface area leads directly to a reduction in the amount of drag. Today however, aircraft can be constructed with a host of different alloys and are not restricted only to aluminum but also use more exotic metals like magnesium and titanium. There is one material though that is arguably transforming aircraft structure and manufacturing today more than any other in history.

Composites are nothing new as far as a construction concept is concerned, but the composites used today are far different from those used in the recent past. Because a composite can be defined to be any material made with multiple sub-materials[34], it has been used for thousands of years. Even structures made out of concrete are considered composite-based since concrete is composed of multiple types of aggregate held together with a cement. However, the modern composites that have begun to change aviation as well as other industries are fiber-based. Carbon fiber in particular is a material that has numerous properties making it ideal for aircraft—both manned and unmanned.

A structure made with carbon fiber starts with single strands a few micrometers in diameter composed of carbon atoms. These strands, just as the fibers in most fabrics, are aligned together to make larger fibers prior to being woven into sheets of cloth. From this point in the process though, there are certain characteristics that set fiber-based composites apart from others. Firstly, with a woven fiber, by altering the layout of the weave the resulting sheet can be produced for an optimal structural loading application given the orientation of the carbon fiber threads[35]. For example, in Figure A.6, the cloth on the left is composed of a unidirectional weave. In other words, all the fibers are woven to be aligned in the same direction (they are usually cross-woven sparingly with thin threads of cotton or other material). In the example on the right, the cloth is woven with carbon fiber in two directions. Thus, for a usage on a part that is known to have load primarily only in once direction or will have more loading in one direction than another, then that

part can be made entirely or partially with unidirectional cloth with the fibers aligned with the direction of the highest loading. An application for unidirectional carbon fiber cloth on an aircraft would be around the root of the landing gear where it is known nearly all of the force ever applied to the location will be upwards from when the aircraft is touching the ground. On the other hand, for something like an underwing engine mount, bidirectional cloth is more likely to be used since there is not only stress opposite of the airflow direction (primarily from thrust), but especially if the engine is propeller powered, there is likely to be a large amount of torque as well applied to the engine mount and wing.

Once the carbon fibers have been woven into cloth, a mold must be constructed for the desired part on which or in which the fiber can be placed and spread with epoxy. Repeated layers of carbon fiber cloth and epoxy are then layered with as many sheets as are required to fabricate the part. For most pieces in aerospace and other applications, the part is placed in a vacuum to remove any air pockets and residual epoxy (removing excess epoxy removes unneeded weight). The part is then removed from the mold once the epoxy is dried. [35]

Outside of directional strength given by the weave, the other two main advantages to carbon fiber composites are the construction method and the strength-to-weight ratio. As discussed, carbon fiber-constructed materials use a woven carbon fabric combined with epoxy (although some types have strands of other materials woven in as well such as Kevlar). However, because the fibers are flexible until the epoxy is dried, very complex shapes can be produced compared to what is possible with metals. This allows much more aerodynamic parts can be made than previously possible. Furthermore, even if the part can be made as aerodynamically with other materials, carbon fiber allows for a large complex part (such as a fuselage) to be made as a single piece compared to using metals or other materials that might require multiple smaller parts to be made and then combined to make the single larger piece. By fabricating a large part of an aircraft as a single piece rather than multiple ones, there can be less seams and rivets thereby decreasing turbulence, drag, and weight.

All of these technologies among others have allowed UAV’s to become what they are today. By combining technological advances in computers, navigational aids, and materials, more efficient and effective aircraft have been able to be developed than ever before. Furthermore, with the necessity for rapid and efficient computations and electronic communications to function, unmanned aircraft have not only been improved, but they have become possible because of the advent of these technologies. However, with the further advancement of these technologies and the dawn of others such as artificial intelligence (AI) mechanisms, UAV’s still are being further improved compared to what exists today.

Modern Civilian and Military UAV Applications

UAV's have obvious uses today for both military and civilian missions. The military, as discussed, has been using drones for missions of various types for an exceptional amount of time. However, most notably, they have increasingly been used for not only surveillance, but also more complex missions including strike. Medium and high altitude surveillance is perhaps the simplest type of mission since the only task for the aircraft is to fly with a camera or other reconnaissance device (not to say the equipment is not difficult to make and design, but the control methods themselves are fundamental). However, as the mission of a UAV becomes more complex (such as ground-strike, low altitude surveillance and reconnaissance, and even air-to-air combat missions), future unmanned aircraft will require more sophisticated control code[36] and more advanced technologies than available today.

As civilian usage of UAV's is concerned, there are numerous capacities where drones are currently being utilized, but there are also a large number of areas that are yet to have the full potential of UAV's applied. While photography is by and large the most common usage for civilian/personal drones today, real estate and utility uses are also common as is insurance adjustment[37]. However, the versatility of UAV's is likely to continue to expand for civilian usage in the near future. From search and rescue to firefighting and construction, there are uses for drones that have not even been imagined yet.

Background of Radiological Search Systems and Bayesian Statistical Searching

Directional vs. Non-directional Detectors/Targets

Although a fundamental principle, a very clear distinction needs to be made between different types of sources and targets. For the purposes of this project, it can be assumed that the majority of detection can be split into two categories: directional and non-directional. Directional detectors must search in one direction to gain data. This includes most spectrometer-based devices, cameras, laser-based devices such as LIDAR, radar, and infrared-sensor devices among others. On the other hand, non-directional detectors do not require the target to be a specific direction from the detector or even in clear sight of the detector. While smaller and sometimes harder to define, this group of detectors can be considered to include scintillators, aerosol filters, and magnetometers among others.

Directional detectors have both advantages and disadvantages. As an example, if a camera is mounted on an aircraft and is being used to search for a target using AI or other methods, the detector (i.e. the camera) will not be able to detect the target at all if it is not in the view of the lens even if the target is right behind the

detector. However, if the target is in view of the camera, then the probability of the target being in view is ~100% (save for any inaccuracies in AI software or human error). Thus, a directional detection will often have nearly a Boolean-type probability output when searching for a target.

Non-directional detectors differ from directional detectors in the methodology used for target search, the accuracy of the resulting estimate, and the field-of-view that the detector is able to search for a target. An example of a non-directional detector is a plastic scintillator such as the one used for the underlying development of the PADUA algorithm. This type of detector is able to detect radiation from a source no matter which direction the source is from the detector. Thus, if the source is 1 meter in front of the detector, the average signal over time should be approximately the same as if the source was one meter behind or next to the detector (neglecting the effects of possible shielding). However, the other difference in the two types of detectors is the probability of the target's location. While a directional detector can determine with nearly 100% certainty the location of a target once the target is in view of the detector, a non-directional detector will yield a signal that can be processed to determine a probability of how far away the source resides. The non-directional detector itself generally has no way in which to predict the direction of the source without further data processing. While there are a few detectors that are exceptions to this statement (such as large neutrino detectors[38]), they are usually impractical or simply not applicable to the types of scenarios on which this project is focused.

While this division of detector types is not definite, it can be considered a general truth. It should be noted too that some types of directional detectors are able to act as both directional and quasi-non-directional detectors in some instances. An example of this is a rapidly rotating 360-degree field of view Light Detection and Ranging (LIDAR) detector. Although each instance of beam emission/detection is directional, each instance occurs very rapidly. By scanning an area with a laser, a 360-degree LIDAR is able to piece together numerous directional detections into a single image. In this way, it can be considered a quasi-non-directional detector. There are other instances of this, but in general, the two types of detectors still holds true in the vast majority of cases.

In this project, only non-directional detection will be utilized since the scintillator this project is based on is a non-directional detector. Specifically for PADUA, the detection needs centered on the obtaining of radiation count rates from the surrounding environment. Furthermore, because non-directional detection tends to utilize generalized methods and statistical analysis, the methodology and processes outlined in this project should be able to be applied not only to radiation detection but also to other non-directional detection systems. Some examples of other applications include real-time aerial particulate analysis for chemical weapon identification or other chemical compound search missions, real-time aerial

radioactive fallout particulate analysis, and other radiation and particulate detection methods as well as non-directional light detection.

Bayes

The approach behind the PADUA algorithm is split into two main parts: control methods and detection methods. While the swarming theories already described in the previous section touch on both sides of the project, the detection methods are primarily centered on specific types of search techniques. Later in the document, the exact theory and layout of the search techniques used in this project will be discussed, but first Bayesian search methodology must be described, as it is the origin of the final algorithm.

Bayesian statistics is a mathematical method for locating a target with its basis coming from the general Bayes Theorem. It should be noted that although much of the usage of Bayes Theorem used and discussed within this project consists of radiological material locating, this theory is applicable in many areas and not limited at all to merely radiological detection. For example, assessing lending risk has often utilized Bayes Theorem[39] as has the evaluation of vaccine efficiencies[40] among numerous other applications.

Thomas Bayes first proposed Bayes Theorem in his paper “An Essay towards solving a Problem in the Doctrine of Chances,” but it was not until after his death that the paper was published by Richard Price in the mid-1700’s. Laplace was later the first known scientist to have used Bayes Theory for any type of common practical application when he was looking for a method in which to estimate more accurately the locations of stars and planets observed by astronomers[41]. At the time, with the rudimentary instruments available multiple readings would not necessarily produce the same or correct answer. By implementing Bayes Theory though, Laplace believed it was possible to estimate rather accurately the locations of celestial objects. Furthermore, the same theory could be applied towards the analysis of any large dataset according to Laplace. As proof, he tested the theory upon gender birth rates throughout the world. Again, utilizing some basic approaches of Bayes Theorem, Laplace was able to determine the ratios of male and female humans born throughout the world. His estimate from the data he received showed that more males than females were being born (although only slightly) and sure enough, after his death, data continued to be gathered that showed his predictions were correct. However, it was not until later that Bayes Theorem became popular since mathematicians and scientists alike, even after Laplace, continued to grapple with the concept that probability-based theories were anything more than guesswork.[42]

Modern Uses of Bayes Theory

Perhaps the first notable modern usage of Bayes Theory was during the aftermath of the Palomares B-52 crash. During the Cold War in 1966, an American B-52 from Seymour Johnson Air Force Base was flying a defense mission over the Mediterranean armed with four MD-28 hydrogen bombs. However, during a routine refueling above Southern Spain, the bomber collided with the KC-135 tanker. All crewmembers on board the tanker and three airmen on board the B-52 were killed. Additionally, all four of the bombs carried on board were released during the collision around the town of Palomares. Of the four, one was found mostly intact in a riverbed while two of the weapons' conventional explosives detonated. This caused essentially a dirty bomb situation where radioactive material (especially Plutonium and Tritium, the latter of which bonds easily to water[43]) was scattered throughout the surrounding fields and farmland. The fourth bomb could not be found immediately though as it plunged into the Mediterranean Sea. [44], [45]

While there have been numerous explanations proposed for the urgency of finding the missing bomb, the most likely reason was fear that the Soviets would find it first. This being said, Dr. John Craven was designated by the United States Navy (USN) to be in charge of finding the missing weapon. With mounting pressure from the Johnson Administration to locate the bomb and the case beginning to appear as if the USN was looking for a needle in a haystack, Dr. Craven decided to take a more unconventional scientific approach. Implementing Bayes Theorem, he figured they might have a better chance of localizing the location of the weapon by updating a probability grid based upon the detected count rates found by numerous ships. Although President Johnson was not favorable towards the methodology Dr. Craven was utilizing, Bayesian statistics soon vindicated the scientist. By updating the probability field with both the data received off the detectors on board the ships and information gained from a fisherman who claimed to have seen the weapon fall into the water in a general region that coincided with the high probability location found with the Bayesian-updated field, the weapon was found directly under the calculated location.[46]

Not long after the Palomares incident, there was a second accident; this time, the subject of interest was the USS Scorpion (SSN-589) nuclear submarine. One of the main differences between the Palomares incident and the USS Scorpion was the initial search region size. While the Palomares search was large, it was still regulated to the coast of Spain. Conversely, the search for the USS Scorpion began with the knowledge that the submarine had not arrived in Norfolk after leaving Greece. Bayesian methods very similar to those used with the B-52 crash were utilized for the Scorpion's search, but a second method was used jointly with Bayesian in the Scorpion search.

Rather than relying solely upon Bayesian probability updating with radiation measurement devices, hydroacoustic detection was also utilized in order to narrow

the region where radiation detection needed to be performed. When the USS Scorpion sank, instrumentation in the Canary Islands was able to detect a signal believed today to be the sound of the submarine imploding. With this, a bounding box was able to limit the extent of the search. This vastly decreased the time to locate the submarine. [47]

Both of these instances showed that it is not only reasonable to utilize Bayesian statistics as a mathematical tool, but also that they are incredibly useful in real-world scenarios. Moreover, in both each of the cases, it was shown that either truth-values and/or bounding boxes are also very important for efficient Bayesian methods. Bayesian statistics will work without bounding boxes and initial estimates, but with initializations that are more accurate and smaller search regions, arriving at a solution should take far less time than the alternative. As such, there must be methods to not only search for an object using statistical methods but also to refine the region before such statistical methods are introduced to have a fully effective and efficient search system; this is the premise of PADUA.

The most important research effort with respect to the development of the PADUA algorithm is that performed by a research group at the University of Bristol. Published only in 2016, their work used a similar approach to that laid out by the PADUA search algorithm but with a more specific mission goal. Both the PADUA algorithm and that developed by the University of Bristol utilize a multi-tier method where a large area is narrowed down to a smaller one. The primary differences are that, while the system developed by the University of Bristol uses a faster and more spread out search for the first sequence followed by a tighter and lower flight path for the second sequence, the mission's goal is centered on radiation mapping for evaluating a nuclear plant disaster area rather than individual source locating. In addition to this, although multiple methods were tested in the research experiment and system development, there was no single tier swarming evaluated or method combination tested. Additionally, although Bayesian methods were tested (especially with regards to contour plotting) in the experiment, assumptions were made that the cleanup area is rural (such as around a power plant) and there is a large amount of scattered debris. PADUA seeks to assist with the same effort, but rather than focusing only on meltdown instances, PADUA focuses primarily on single-source searches where time is of the essence. Additionally, contour plotting could be added to PADUA later, but in a dense urban environment, this type of mapping analysis makes far too many assumptions to yield any high-confidence results. [48]

Impact of Project

As the final product of this project, an algorithm-based system was developed that enables autonomous systems to search for radiological material in varying region-

sizes and with varying levels of autonomy. While this is an important step in the advancement and integration of UAV technology and autonomous radiological material search, there are still numerous further advancements that can be made to the system and methodology in the future. As detectors themselves advance in technology, these should be incorporated into the detector package and implemented into the algorithm; as software and AI mechanisms are developed, they should also be incorporated into the package where applicable. While this project intends to develop a stand-alone package that can successfully show that the locating of radiological material can be performed more autonomously than with today's methods, there will still be continual improvements that can—and should—be made to the system.

Additionally, while this project is intended specifically towards military/security purposes, detector-agnostic techniques derived for this project can likely be altered for usage with industry and agriculture as well as other civilian uses. For example, rather than just using a predetermined flight path for crop dusting, a UAV might be flown completely autonomously using the information obtained from image and hyperspectral data to determine where to fly and where to spray. Industry might be able to improve assembly line robotics in a similar manner by combining more autonomous robotics with detectors rather than just pre-determined functions and paths. These are only two examples to what this project might lead, but it is definite that nuclear security will benefit directly from the development of this system.

Perhaps the most important benefit PADUA offers to national security is the life- and cost-savings. By allowing vehicles/robots to perform critical tasks that are reasonably simple but require a high degree of autonomy, a warfighter is not just aided—the warfighter can be supplanted. Using values found for 2004 and then adjusted for inflation to 2018 dollars, the approximate cost for an enlisted soldier that will serve for 20 service years is approximately $58,000 per year. For an officer over the same service time, the approximate adjusted cost comes out to $117,000 per year[49]. These costs include salaries, health care, retirement, personal equipment, etc… The boots-on-the-ground warfighter will probably never be entirely replaced, but by decreasing the number of soldiers via replacing tasks such as guarding, surveillance, tracking, and pinpoint strike missions, not only are human lives being taken out of harm's way, but the overall cost of the military should continually decrease as systems that are more autonomous are implemented. Even if the cost of a vehicle and control hardware seems high, a robot requires no salary, benefits, college assistance, or retirement. For this project, the test version of the full system costs approximately $25,000 per vehicle *including the detector*; even if a military-grade system costs multiple times more, it would take a fairly large increase to come close to assimilating the cost-per-year of such a robotic system to that of a soldier doing the same tasks.

Furthermore, there are indirect cost savings as well when it comes to using autonomous vehicles to complete military duties. For example, less armor is required for a robot to complete a mission since the cost of losing a device is simply the cost of the device—not a life. If a robotic agent must be transported in a ground vehicle (or is a ground vehicle), the weight and size should both be able to be significantly decreased compared to a similar manned mission requirements. While this might appear to be a tangent, logistics are extremely important in the military. By decreasing the size and weight of equipment required for a mission, more rapid and cheaper deployment is possible since more vehicles can be delivered. With smaller, lighter, and more efficient autonomous vehicles, the fuel savings alone should be significant since, in 2012, the United States Air Force alone spent nearly $9 billion on fuel[50].

Saving lives is the most important goal of this project, but at the time of writing, saving money in defense is important too. Personnel cost reductions, fuel savings, armor and vehicle safety reduction, and other financial decreases due to the advent of more intelligent and capable robotic systems promises to be a winning prospect on all fronts. Fewer soldiers will have to be placed in harm's way and at a lower cost than with which the U.S. military currently operates. Perhaps then, the greatest possibility of this system is the ability to save lives, while also saving money.

CHAPTER 2

LITERATURE REVIEW

This chapter is split into two main parts that correspond loosely to the two parts of the PADUA algorithm. The first section discusses modern UAV usage and swarming algorithms and theories while the second section describes Bayesian mathematics and the intricacies of the statistical calculations used in the PADUA system. While there are other parts of PADUA, this chapter presents an overview of some of the subsystem backgrounds used in this project.

A Background History of Swarming Theories and Applications

Very relevant to this project are the theories underlying swarming. While swarming has become an extremely extensive and complex subject, a brief history overview and a description of basic swarming theories will be presented in this section so that some of the methodology in PADUA can be more easily understood.

Note that the definition of a “swarm” used in this project is much broader than that which is commonly used. Rather than defining a swarm as simply a large group of individuals moving together as a group, the definition for a swarm in this project will be as follows: “any group of two or more individual agents moving with a common purpose using shared data.” This means that, for testing, a series of two agents can be used for proof-of-concept and while agents must move with respect to the locations of other agent(s), they need not necessarily move in a close-knit group in order to be considered a swarm.

BOIDS

The first complete swarming theory was outlined by Craig Reynold’s publication “Flocks, Herds, and Schools: A Distributed Behavioral Model” describing his “Boids” algorithm (‘Boid-oid’ or Bird-like object). This theory laid the foundation of all the swarming methods that came afterwards. Although there are new methods being used and developed today, many of the fundamental theories in Boids serve as the basis for the swarming algorithm developed for this project (PADUA) due to the wide range of area sizes, search patterns, number of vehicles, and variety of location types that will be encountered by the system. Boids stated that there are three major parts to any swarming algorithm: collision avoidance, velocity matching, and flock centering.[51]

The three pillars of Boids can be explained effectively in simple terms. Collision avoidance (i.e. separation) consists of ensuring that each of the agents remains at least a minimum distance from all other agents to prevent crashes and congestion. The velocity-matching rule states that each agent must attempt and match the

velocity and orientation (speed and heading) of nearby agents. This characteristic assists in making certain that agents actively attempt to move in a pattern without too much deviation in the group's average path. The last part of Boids, flock centering, states that each agent should attempt to travel towards the centralized location of nearby agents. It should be noted here that each agent (or as Craig Reynolds calls them, "boids") is very limited in its total scope/sight. This being said, each of the three pillars applies to an agent in relation *only* to nearby agents. Thus, rather than flock centering stating that an agent should go to the center of the entire flock of agents/boids, that specific agent should try and locate itself in the averaged central location of neighboring agents. [51]

Although not specific to swarming, a separate aspect that Reynolds discusses in "Flocks, Herds, and Schools…" is extremely relevant to this project: prioritization of individual agent actions. In artificial intelligence, when an individual unit or agent is 'thinking' and making decisions, certain of those decisions must take priority over others. Furthermore, while this is true for all artificial intelligence, this is even more important and unforgiving for real-world flying machines for two reasons. Firstly, if real machine rather than a simulated agent reads information that causes a collision or other fatal error, there is no way to fix the problem other than to send a new agent to the location. In a simulated environment, if there is a failure, there are a number of ways in which to artificially start the algorithm over to a previous point or to attempt parts of the simulation incrementally to find a successful method prior to running a full instance. Secondly, for machines to use artificial intelligence in a swarming manner adds enough difficulty to a system, but when a machine is flying, these difficulties are compounded exponentially. In such a case, often the machine cannot stay in one location to process external sensor information (unless the machine is capable of hovering), and algorithm efficiency is of the essence since flight time is usually limited due to onboard fuel/battery storage.

Reynolds attempted to solve similar difficulties in his swarm theorization when discussing acceleration requests. These requests can be attributed to the methods for prioritization used in this project for external collision avoidance and flight pattern control. Reynold states that while each agent in a Boids algorithm makes decisions on a very individual basis to attempt to go to a certain location, each agent cannot make a decision until the main algorithm allows. Thus, for an agent to change direction, speed, etc. an acceleration request must be issued and accepted. However, because each agent makes decisions selfishly, there must be a way to allow, not allow, or compromise with each agent's request. Reynolds states that averaging all the requests together and then allowing the compromise as the final decision is a fundamental answer to the prioritization problem, but its results were lacking during testing. Instead, a better method is *prioritized acceleration allocation*[51]. This method operates by setting priorities for each type of acceleration request and then summing the magnitude of the total request. As each small request for an agent is accepted, the acceleration magnitude is added

to the total sum until a threshold magnitude value is reached whereupon lesser priority requests are decreased in magnitude (compromised). This means that certain events could have very high priorities (such as real-world obstacle avoidance) that might even cause an agent to miss implementing part or all of one of the three pillars if necessary. While PADUA does not utilize this exact method, prioritization of environmental variables and incoming sensor data is an important aspect that carries over from Reynolds' theories.

PSO

A step beyond Boids theory is Particle Swarm Optimization (PSO). Built algorithmically and tested in 1995, PSO is a basic but very effective method of utilizing swarming particles/agents to achieve a common goal[52]. In the fundamental PSO method, a global minimum value in space is found based on a combination of the data an agent finds with the data the swarm finds. This allows each agent to have an additional property added to its movement set, so rather than only moving with the basic three principles from Boids, each agent can move according to knowledge from both group and local sources. Although a modest concept, this allowed swarming functions to become applicable to real-world problems in terms of using multiple agents to complete a task.

PSO was first outlined in effective detail by Kennedy and Eberhart in their paper "Particle Swarm Optimization." The original reason for PSO that they described was similar to that of Boids—to simulate a flock of birds or other swarming animals. The difference with PSO is that rather than only using three simple rules as with Boids, PSO utilizes some rather simple rules as well but utilizing structured mathematical equations rather than the looser pillars theorized by Reynolds. However, before beginning their discussion of the mechanics that make up PSO, Kennedy and Eberhart point out a very important aspect of all swarming theories. That is, while a swarming theory is often intended to be used primarily for a specific situation (such as flocking birds or schooling fish), there are often chances that the same or a slightly modified theory might be applied to a wide range of situations. The example that is alluded to in the paper though is that of an individual human's thoughts when organized following a version of swarming due to the need for 'non-collision.' While this is an extremely abstract theory for the implementations of swarming theories, it is still an important aspect that needs to be remembered that, during the discussion and study of all types of swarming methods, there are likely to be unintended uses for these theories that might not have previously been thought.

As far as the PSO methodology is concerned itself though, Kennedy and Eberhart outline a few procedures that can be used to create a simulation algorithm that corresponds better to the actual characteristics of flocking animals compared to the more aesthetically inclined Boids algorithm. The first method they attempted to

satisfy a true PSO algorithm they described as “velocity matching and craziness”[52]. With the velocity matching, each agent is intended to attempt to match its velocity and magnitude with that of its neighbors. However, a term called ‘craziness’ must be applied otherwise the agents all simply move in a pattern rather than a realistic-like movement that includes small aberrations from a perfect arrangement. Because of the need for a ‘craziness’ factor, they looked instead to utilize past research performed by Heppner on how birds locate a roost. From this, Kennedy and Eberhart developed a short formula (Eq. 2.1[52]) that shows the evaluation of a current position given that the location (100,100) had an evaluated value of zero (representing a roost or other target location)[52].

$$eval = \sqrt{(presentx - 100)^2} + \sqrt{(presenty - 100)^2} \qquad \textbf{Eq. 2.1[52]}$$

From this, they continued to develop the PSO algorithm into a function representation of a flock of birds locating a roost or food with a final equation representing the change in an individual agent’s velocity (Eq. 2.2). While there are many other steps they took to arrive at this point, the essence of the algorithm is that an agent must have at least a short term ‘memory’ by which the agent can store the best location value thus far. In Eq. 2.2, the calculation for the new velocity the agent should use is shown. The equation can be described as the new velocity is equal to the old velocity plus twice a randomly generated value multiplied by the best local position (i.e. the best position from the agent’s memory) minus the present local position (i.e. the agent’s present location). This is then added again to twice a randomly generated value and multiplied by the global best location (i.e. the group’s best recorded location and written as $pbestx[\,][gbest]$) minus the present local position. The value of 2 used to multiply the random value is simply from experimentation and is not necessarily optimized but is rather used so that each agent will slightly overshoot the target creating a more realistic visualization.

$$vx[\,][\,] = vx[\,][\,] + 2 * rand(\,) * (pbestx[\,][\,] - presentx[\,][\,]) + 2 * rand(\,) * (pbestx[\,][gbest] - presentx[\,][\,]) \qquad \textbf{Eq. 2.2[52]}$$

While the PSO algorithm laid out by Kennedy and Eberhart is a large step from Boids towards replicating flocks and schools, there is a larger idea that should be realized. PSO showed that not only could swarms or flocks be replicated with patterned movements, but also swarming algorithms can be used in conjunction with a target or goal. By using locally and globally stored values, treating a group of agents (whether virtual or real) as a swarm can allow a very difficult task to be accomplished faster and more efficiently than treating each of the agents as a sole individual trying to reach a goal. Rather than just having many agents move together then, by instituting PSO-type swarming methods, goals can be reached with little effort on the part of each of the agents; however, a large amount of work is actually being done by the combined swarming group.

Applied Insect Swarming Methods

Since the proposition of PSO algorithmic methods, there have been numerous other theories tested with the common intention being to increase a swarm's task completion efficiency; one of the most important sources of inspiration for swarm theories has been nature. Even from the beginning of swarm theory research, many of the ideas and algorithms revolve around natural phenomenon such as flocking birds and schooling fish and insect colonies[53]. In particular, optimization controls based on insect colonies tend to bring a much more complicated but useful measure to efficient multi-agent mission completion. The easiest way in which to explain the difference between many insect-based algorithm compared to others is that non-insect algorithms tend to focus on aligning and moving agents as a correlated group and/or to complete a single task as a group; insect-based algorithms use more complex data transfer and multiple individual tasks agent-specific abilities to complete an overall mission.

While the details will be discussed later, when describing the state-of-the-art methods used today, the two primary types of insect swarming that should be mentioned are those of ants and honeybees. These are two types of creatures use swarming in their everyday lives to accomplish extremely difficult tasks.

Instead of using a central location for all data transfer as honeybees do, ants use a pheromone trail that other ants can sense in order to relay data from one to another in real time and in any area[54]. A pheromone trail method can be assimilated to a real-world robotic system where agents send data out wirelessly either to each other or to a central computer (where control of other agents is then updated). However, as swarming techniques become more complex, so do difficulties in system optimization. For example, with an Ant Colony System (ACS), optimizing the time/range for each individual agent's tasking can be exceptionally difficult with excessively complex algorithms having been tested[55].

During a search using pheromone communication, ants utilize agents (i.e. ants) with different individually assigned tasks. First, a scout ant will go out from the nest and look for food. As the ant locates leaves or other types of food, it will release a trail of pheromones that other ants can follow backwards towards the food source[54]. Thus, the scout ant acts primarily as an individual agent searching for a local maximum while the other worker ants decide which pheromone trail to follow from the scout ant(s), which is equivalent to determining a global maximum based on given detection information. A diagram of this type of system can be seen in Figure A.7 below. In the figure, a basic system is shown where a scout ant moves around sporadically until it finds food. Once it does locate food, it heads back towards the nest while releasing a pheromone trail for the two worker ants to follow. It should be noted that, compared to that which has just been described, ant organizational structures are usually more variable and complicated in

actuality; however, the system described is still the base framework by which most ant colonies operate.

Algorithms based on honeybees are the second insect-based swarm design. Similar to an Ant Colony Optimization (ACO), a beehive operates with a swarming technique that allows individual bees to contribute towards the location/completion of both individual and group goals. Also similar to an ant system, bees communicate with each other to relay details about local maximums so that a global task can be continually updated. However, unlike ant colonies, honeybees act as substantially more complicated agents and use a different system of communication than ants. First, while honeybees acting as swarming agents do need to search for and locate local maximums, there is usually more than one task that a single bee is attempting to complete. For example, if a bee is flying through a field, it might be looking for pollen as well as watching for potential predators and locations for a new hive. Second, rather than relying on a pheromone trail, the bees communicate to one another via dancing. Thus, once a bee has acquired useful data, it will return to the hive and perform a communication dance in a centralized location where most of the other bees can learn what the dancing bee has found[56].

By performing data collection communication in these ways, there are both benefits and drawbacks. A pheromone trail allows for near real-time updates that other ants can follow. Unfortunately, using such a method restricts the type and complexity of information that can be transferred. However, while communicating as bees do by dancing in a central location allows very complicated information to be transferred from one agent to another, it is time-consuming and imposes the limitation that data cannot be relayed in real time. Advantageous parts for both of these systems are used in PADUA in the swarming sequence.

Modern UAV and Swarming Methodology and Technology

As the primary goal of PADUA is to form the underlying controlling algorithm for a system of multiple UAV's, modern systems and technologies surrounding swarming systems must be described. There have been numerous advances in swarming methodology in recent decades of which portions from many different ones are used in PADUA. For a subset of aviation and robotics that is rapidly becoming more frequently used, swarming is a relatively new field.

Modern UAV Swarming in Practice

As discussed in Chapter 1, UAV's have been continuously improved and used more often over the past several decades, but this project does not focus only on the technical issues and improvements of a UAV. Rather, this project focuses upon the combination of swarming technology and control coding combined with search

methodology to create a single-package unmanned aerial system algorithm with a specific mission purpose. Some of the individual parts of PADUA are expanded versions of systems found in other modern research; however, the combination of many of the subsystems and the implementation into a single algorithm coupled with other novel subsystems and a term that will be coined “mixed control intelligence” are new and unique in this project.

Swarming UAV’s are increasingly being used by both nations and organizations alike. First for surveillance, swarming UAV’s have been used in several notable instances for more effective and/or continuous reconnaissance. In 2014 the Russians utilized a loose sense of swarming UAV’s during their invasion of Ukraine when numerous daily UAV flights were observed[57]. Another instance of foreign swarming drone usage has been that by Syrian militants. In early 2018, the Russian Hmeimim airbase was attacked by ten rudimentary fixed wing single-prop UAV’s armed with small rockets, and three more attacked a Russian Naval location that was also nearby. Figure A.8 shows an image of one of the UAV’s allegedly to have been used to attack the Russian installation. While no damage to personnel or the bases were reported, this was still a first instance of swarming drone usage by non-state actors during war with nefarious intent[58]. While this case might have not yielded damage to the target, such devices can be continually tested and improved at fairly low costs to obtain drones that can in fact effectively swarm and carry out an attack mission by terrorist or militant groups with little funding.

Perhaps the most notable modern instances of American swarming military drones have been those with the intentions to either survey areas or attack an adversary with a mass quantity of aircraft. For both of these missions, unmanned systems have advantages that manned aircraft lack. Small UAV’s are cheap and light. They have a small radar cross-section, are often made of radar absorbent materials, and can be transported and launched from a wider range of location-types than larger manned vehicles. While multiple research agencies (mostly with the U.S. government) have produced swarming drone systems, production aircraft have also begun to be used with swarming mechanisms on both research and active surveillance levels; one of the most used UAV’s by the U.S. for these purposes is the Q-27 ScanEagle[59].

Application of Swarming methods to Autonomous Vehicles

While there are numerous types of swarming systems, methods, and algorithms, the application to a fully or semi-autonomous swarm of aerial vehicles can be exceptionally difficult. For a vehicle to be fully autonomous, it must operate completely on its own without any directions given by a ground station or human. Because of this, semi-autonomous systems are not only more often used, but are much more trustworthy systems in most cases; this is because with the ability to have some human/ground station input, the aircraft will inherently be more limited

in its operational choices than otherwise and a human will usually have the final decision-making ability.

The control code of a UAV can be split into two main parts: inner-loop and outer-loop coding. Inner-loop algorithm development typically consists of the intricacies associated with the actual interpretation of commands sent to the aircraft in order to translate the messages into hardware interface commands. For example, applying formulae to commands sent to an aircraft such as the examples given in Lee's and Shim's paper allows for the commands to be interpreted into signals that can be sent to the actuators, motors, and other control hardware on board the aircraft[60]. Outer-loop coding can be considered more of a 'big picture' algorithm when considering control codes though. For example, an outer-loop control code might send a message to an aircraft telling it to move to a specific waypoint whereas the inner-loop codes will take the message sent by the outer-loop codes to continue to send signals to the control hardware on the aircraft to move it towards the waypoint. In other words, the inner-loop code handles aircraft stabilization, real-time movement control, and interpretation of outer-loop commands; the outer-loop takes care of swarming commands, waypoint determination, and interpretation of aircraft-mounted detection equipment for translation into flight path commands. This split process is used clearly in the PADUA algorithm.

UAV inner-loop code research consists primarily of determining the most accurate, efficient, and smooth methods of controlling an aircraft. This often consists not only of creating a response function based on input commands, but also by determining a relaxation factor method to prevent a positive feedback loop from occurring and to avoid over-stimulation of the aircraft by the commands. Lee and Shim use a first order high-pass filter method in their paper for primary inner-loop flight control architecture as can be seen in their paper[60]. Similar work has been performed with fixed-wing UAV inner-loop codes by Mystkowski but using a sixth-order transfer function[61]. While there are countless other examples of inner-loop research and methodology progress that have been done, the primary goal is generally to interpret the commands given to the aircraft in the most accurate manner possible that are communicated to the inner-loop code by either an outer-loop control code or a human-determined input (such as from an RC controller).

Of more interest in this project is that of outer-loop coding, specifically as it relates to swarming aircraft control. Because outer-loop coding controls overall aircraft movement directions rather than the intricacies of single aircraft hardware control, this is the part of an algorithm system that is developed for swarm control. While slower computationally to send commands from a swarm algorithm through high-level codes to be interpreted by low-level ones, this method allows for simpler programming as well as far easier abilities for adding in updates and making changes to the code later.

There are two primary methods that an outer-loop control code can typically use to determine an aircraft's actions. The first is simply by sending the aircraft to a waypoint while the second is by adjusting the aircraft's direction and/or velocity components. Sending an aircraft a waypoint command is very simple in high-level coding and allows for the majority of the direct control operations to be performed by the inner-loop. This is very advantageous when only small bandwidth communications systems are available since very little information needs to be relayed to the aircraft generally. For example, rather than receiving an aircraft's position several times during the course of sending it from point A to point B and then transmitting new direction or velocity commands to the aircraft, by sending the aircraft a single waypoint, the algorithm on board the aircraft can do all the work in adjusting the aircraft's movements to move itself from point A to point B. This method does have some severe limitations though when discussing swarming capabilities. Assuming the aircraft has the computational and sensor capabilities on board, it can be assumed that the aircraft should be able to avoid obstacles and other aircraft it might approach meaning that a waypoint message would be more than enough information for the ground station to send, but this still has other problems. Firstly, having the ability for a ground station to act as a redundancy in collision avoidance is of course a tremendous advantage whether or not it is needed, but this can be difficult depending on the coding mechanisms used during mid-flight waypoint commands. Secondly, if the aircraft is small, it likely does not have all the sensor equipment needed for accurate active collision avoidance due to weight and/or battery limitations; in such a case, the ground station must be relied on—at least in some capacity—to prevent the aircraft from flying into other aircraft in the swarm as well as into other objects. Including the limitations of low-cost sensory equipment, it becomes obvious that swarming with small low-budget unmanned aircraft can become exponentially difficult with the number of vehicles in a swarm without ground station input. For example, a small unmanned aircraft, even those with collision avoidance sensors mounted, often have a very limited view of the area around itself. This being said, if the aircraft either encounters another aircraft in its swarm, multiple ones, or simply other obstacles, it can become very difficult for the aircraft to continue its journey from point A to point B while remaining in the swarming pattern. This can be assimilated to the difficulties of find one's own way out of a maze compared to having someone shouting directions that can see the entire maze and the obstacles in it from above; the latter example is an assimilation of a ground station acting as a redundant collision avoidance guide for an unmanned vehicle.

However, applying a set formula relating the desired location or velocity of an aircraft to the actual one can still limit the function of a swarm tremendously. Instead, it has been shown in numerous instances of research that applying some of the nature-based swarm theories to a robotic swarm can yield surprisingly effective results without sacrificing mission capabilities. Some of the methods being researched and utilized by the U.S. military are outlined very well by the

Harts where they discuss the usage of pheromone-based swarm systems for automatic weapon systems[62]. In their paper, it is mentioned that the application of pheromone-type swarming systems have been shown to have the advantage of being able to mesh multiple types of robotic systems fairly easily (such as different radar systems) since the signals they detect can easily be translated into a simple pheromone-like signal of where an enemy target might be located. But, while the Harts' paper presents a good overview of the advantages of a pheromone system, there are many intricacies that go into the actual methodology of such an algorithm. Combining a loose theory with exact calculations is perhaps the most difficult part of applying a swarming system to real-world situations.

A balance must be found in the theory being used and the formulae being applied to generate an effective swarming system. Some of the best-described examples of this can be found in the doctoral thesis of Airlie Chapman. One example of a swarming system Chapman describes is the application of a method based off advection. That is, if a swarming aircraft system is desired to act in such a way that it mimics a flow field, then a method coined "advection protocol" can be implemented. Chapman describes the method similar to what happens when an oil drop is placed in water. If the water is stationary, then the oil drop will distribute itself evenly on the surface; however, if the water then begins to move, then the oil will change its concentration based on location in the water and depending on the location in the oil field itself, the velocity components will differ as well. Furthermore, since advection is very similar to diffusion, the generic transport equation can be used (which is usually applied to diffusion). As the Chapman correctly points out, the transport equation (specifically advection) has been used to model a number of complex systems including the spread of disease, migration of animals, and financial market movements. The advection transport equation can be seen in Eq. 2.3. In the equation, the change in concentration-time differential is shown to be equal to the divergence dotted with the concentration field multiplied by the velocity field. More often, the transport equation is shown as the entire equation set equal to zero showing that the sum of the parts must always be equal to zero. [63]

$$\frac{\partial \psi}{\partial t} = -\nabla \cdot (\psi \vec{u}) \qquad \textbf{Eq. 2.3}$$

Chapman goes on to solve the PDE and develop an equation from the general transport equation to represent the position movement command based on advection, but this is only one of many methods that can be used for swarming outer-loop command production. Utilizing advection equations has some major advantages, but it also of course has some drawbacks. For a very large swarm (consisting of perhaps hundreds or thousands of small aircraft), using a system such as this to determine the next general location for each of the aircraft by the ground station is likely to be advantageous. This is because, with numerous

aircraft, they will be able to operate similar to a single fluid-like system with suspended particulates. However, if the system only has a few aircraft, then an algorithm based on diffusion will likely have a difficult time making the system appear fluid at all since the system is more similar to throwing a few logs into a river rather than flakes in a snow globe. In a snow globe that is shaken, there are numerous particulates that follow the flow of the fluid and show the flow fairly well (some particulates become stuck in small eddies while others move quickly in the central flow of the fluid). When logs are thrown in a river though, one or two that are in the center might move quickly with the main flow lines, but then several might also become clumped in a corner of a riverbank. This is acceptable if there are numerous objects, but with only a few, the system is sparse and giving an accurate representation of a whole fluid field is much more difficult and prone to error. More importantly for this project, utilizing only advection equations with a few aircraft (perhaps ten or less) in a large area will cause insufficient coverage of the area. The entire point of this project is to be able to search a large area as rapidly as possible and then be able to narrow down to a pinpointed location for a source. An advection method does not allow for that very easily. Chapman goes on in the document to develop full swarm modeling equations for both sparse and dense swarms attempting to account for wind gusts as well. It can be seen in his paper that combining wind gusts and a dense formation, the model becomes extremely complex very quickly[63]. Furthermore, while his model is developed exceptionally well, the final models focus on ensuring that the UAV system functions only as a swarm without a particular task in mind. When mission-centric swarming systems are developed, pure swarming models are not always the best choice for full functionality, as variable situations are often not accounted for in the development process of a generalized swarm model.

Another problem that has been mentioned in passing already is that of *emergence*. This is a term that can be summarized for swarming systems as an instance where parts of the swarm do not act as expected. This can be for a number of reasons including inability to deal with wind gusts or weather, incorrect communications being transmitted or received, and many other causes. In regards to research with large swarms, the most important type of emergence problem is due to unforeseen results from the complex interactions between the vehicles. Because with large groups of agents this simply turns into a statistical problem of trying to predict when emergence will occur, a function(s) to predict the likelihood of when emergence will occur can be applied to notify the system to take preemptive corrective action. One example of this of the application of the *f*-divergence formula. [64]

f-divergence (as seen in Eq. 2.4) explains the difference between two probability distributions. In this equation, the difference as a function of the two distributions can be set equal to the integral (integrated over the entire probability space) of the divergence function of the two probability distributions multiplied by the denominator's with respect to the denominator's distribution taken with respect to

a reference distribution (μ)[65], [66]. The divergence function can be a number of different types of functions, but for UAV swarms, a nonparametric multivariate kernel density estimation-type function can be used[64]. This means essentially that by implementing a PDF for a set of data, the comparison of the probability distribution can be compared between that set and another. The result of the comparison yields a probability distribution itself that can be used to find where the two sets of data might match versus where there might be a discrepancy.

$$D_f\big(P(x)\big|Q(x)\big) = \int f\left(\frac{p(x)}{q(x)}\right) q(x)\,d\mu(x) \qquad \textbf{Eq. 2.4}$$

Implementing a method such as this can be useful in attempting to ensure that faulty actions do not occur in a swarming system, but there are still numerous variables to deal with such as the choice of the divergence function. However, if utilizing a function-centered method for a macro-level swarm design, then this is an effective approach to minimize possible errors in the execution of the system. While *f*-divergence is not used in PADUA, it would be an important tool for further research for comparing and combining Bayesian results from multiple aircraft.

Problems other than mathematical inaccuracies and algorithm difficulties can also arise when applying a swarming system to a real-world environment. For example, a swarming system of UAV's used by the military might encounter enemy jamming equipment during an operation that impairs long distance communications. Another instance might be if GPS accuracy is decreased due to a high horizon limiting the number of satellites' signals capable of being read by an aircraft. In both of these cases, while any swarming system is going to have difficulties with these obstacles, designing a swarm with a single-function design and an entire outer-loop code being located at the center of system intelligence (usually a ground station) will likely increase the amount of complexities involved in such a situation. This is due to the lack of system knowledge aboard each of the aircraft. If primary communications are lost between the individual aircraft and the ground station *or* if the agents no longer can identify where they are located and thereby not be able to transmit accurate information back to the ground station, then there is nothing to prevent the entire swarm's organization from collapsing.

A few attempts have been made to counteract these complications though; one example is the implementation of an ad-hoc network. An ad-hoc or decentralized network differs from the typical centralized system in that the rather than the ground station communicating with each of the agents individually, the ground station communicates to a single aircraft that then relays the information outwards to the other aircraft that continue to relay the information further. Compared to this type of decentralized design, for a centralized network, the weight and electrical power required of the equipment needed for long distance communications can

rise quickly to levels too high for small and medium-sized drones to accommodate. Furthermore, for long distance communications, it is likely that alternative forms of wireless connectivity (such as cellular or satellite) must be used to relay signals from the ground station to the vehicles since obstacles such as hills, buildings, or the horizon will block line-of-sight communications. Decentralized communication systems do not have these problems since, as long as there are enough aircraft, the signals should be primarily short-range communications and all line-of-sight. However, for a decentralized network, many complexities can arise. First, in the methodology of how to distribute and read commands from the ground station, the instructions must be shared in such a way that they are both used correctly and not reused (this can often be solved with a time stamp, but this requires additional information to be transmitted). Secondly, in a real-world situation, the ad-hoc network must be able to operate in such a way that the primary link between the ground station and first relay aircraft is variable. In other words, if the aircraft that the ground station is communicating with is either lost or the link is disturbed, the ground station must have a way in which to communicate to a different aircraft in the hope that the ad-hoc system will relay the signal to the appropriate agent. [67]

Through the methods thus far in the paper, there have been certain problems that have continued to arise, and while the research efforts that have been discussed tend to be exceptionally effective in their scopes, the application of the theories adds additional layers of complexities. Additionally, although this project is focused primarily on radiation detection, the system developed here is required by the sponsor to operate with as few changes as possible to be detector agnostic. That is, while radiation detectors are being used for this project, the system needs to be able to be easily converted to work with any chemical, biological, and radiological material (CBRM) detection. In addition to CBRM detection though, the system is also being designed to operate with other detection and targeting capabilities such as facial recognition cameras, various types of AI image identification systems, and semi-autonomous targeting capabilities. Therefore, while modeling a theoretical swarm based purely on birds or fish can be performed decently with a single formula-based algorithm and other patch formulae that exist today, the swarming system of PADUA does not necessarily meet the same criteria. A swarm looking for radioactive sources in Manhattan does not need to operate only as a single flock, but instead it needs to operate as a group with a common task governed by the aspiration to locate the sources as quickly as possible. A swarm operating on a mission to observe a crowd of people to detect weapons using cameras equipped with AI algorithms for image analysis would likely operate by hovering or moving in such a way that the entire crowd is surveyed for the most time possible. This is all to say, although swarm theories are still being applied at certain stages of the mission, the swarm in the project is expected to operate on a higher level than simply group-formation. Assimilating to a swarm of bees or a flock of birds serves little purpose for the scope of this project.

Another problem with many of the theoretical methods devised for modeling swarm behavior is the standpoint from which the model is built. For example, a formula based on a simple fluid flow PDE or heat transfer PDE is often used for the starting point in developing a swarm algorithm (such as the advection method discussed earlier) and, when modeling a system in a purely computational environment this type of method might work acceptably to simulate a flock. However, in reality, a flock cannot ever operate this way. While a flock or school might mostly take the form following a formula, it will *never* fit it perfectly for one simple reason: the agents in a swarm of animals can think. This is where the very important line between outer-loop and inner-loop coding comes arises. Modeling a swarming system from a transport equation will be close, but the particulates in a fluid flow do not have the capability to think for themselves. And while all the models discussed so far make note of algorithm parts that must be performed in relation to the individual agents (such as velocity matching), the actual choices that the agents are able to make in a robotic system are still quite limited. In other words, most of the methods discussed and looked at up to this point can definitely be considered swarms, but to call them autonomous swarms might going be too far in many cases since directions still must be sent/computed at a central location or at least via a central/single algorithm.

Instead, for this project, a variable autonomy system was developed. At certain stages of a mission, there is no need for full autonomy while in other stages, full autonomy is a very helpful method to apply. During parts of the mission where swarming capabilities are used though, semi-autonomous and mixed-autonomy methods will be used. The smooth transition of autonomy levels throughout different stages of the system will hereby be referred to as the coined term *"mixed intelligence"* due to the fact that given a total control value of the swarm as a constant, at some points all the control (i.e. intelligence/thought) will occur at a central location while at other times it will be distributed throughout the swarm or even mixed between the agents of the swarm and the central location.

A good example of how "mixed intelligence" might be seen in the natural world is a situation in which you accept God making a school of fish. While God might give each of the fish an inherent ability and desire to act as a school following the general transport equation when a predator is around, they must still individually decide to do so. Some will wonder away at some points while some might not follow the school perfectly, but in general, they will follow the transport equation so long as they are in fact schooling. However, the same fish might change from a school formation to a shoal when there are no predators around. They are the same fish as in the school, but they have then adapted to their current situation/mission. This presents a much more difficult system to model than only a school, but for an effective system to be developed, agents operating as a swarm must *also* be able to operate outside of a swarm with seamless integration.

Some of the ways that distributed intelligence has been used in swarm research have had excellent results. One example was shown by Capt. Allen in his Master's thesis where a swarm of UAV's was operated with an ad-hoc network as discussed earlier[67]. This is a basic form of giving each aircraft its own portion of control. Another instance can be seen in the research involving the tracking of radio-tagged fish populations. In the study, a swarm of UAV's was used to radio-locate the oceanic targets. The swarming part of the research was focused on developing a better statistical method of navigating the aircraft(s) based on the obtained radio signatures to locate the fish as quickly as possible using one or more UAV's. This shows an example of variable swarm size. Furthermore though, the methodology used in this research experiment also utilized a system operating with intelligence and control shared between a ground station and the agents in varying capacities throughout different types of flights. The swarm used the information gathered on fish locations to decide which directions to move, and each aircraft was able to adjust its course based on its perceived movements and its location in relation to other aircraft as well as to the target. During some of the flights though, and a ground station was used to relay preset flight paths. This is very similar to the methodology that was developed and used in the final algorithm for PADUA since a statistical method is used by both systems to locate a target, and semi- or fully-autonomous flight is being used for aircraft control based on the detected information by both algorithms as well.[68]

UAV Swarming Research

Swarming technology has advanced considerably in the past several years, but there are still some very prominent gaps. Some of the work that has been done that is most relevant to this project has come out of the Air Force Institute of Technology (AFIT) located out of Wright-Patterson Air Force Base in Dayton, Ohio. Other notable instances of modern swarming advancements have come from the USN where swarms with numerous aircraft and boats have been not only simulated, but also demonstrated with actual vehicles.

In most of the research reviewed, a few common themes coincide with this effort. First, reduction of cost is a necessity for swarming systems. Swarming low-cost expendable robots allows systems to operate under the assumption that not all vehicles will reach/locate the desired target; but furthermore, if the vehicles being swarmed are not cheap, then the price will grow exponentially because, by definition of a swarming system, there are numerous devices needed. Additionally, as the price of weapons systems continues to rise for the US Department of Defense (DoD)[69], the need to expand weapons technology with reduction of cost in mind is another necessary requirement[70].

One of the common methods used in swarming research has been to try to use as much commercial off-the-shelf (COS) technology as possible. There are two

primary reasons behind this decision. The first is that, as already discussed, lowering the cost of modern weapons is imperative. COS equipment is often cheaper than military-grade counterparts. The second argument that can be made for trying to incorporate as much COS as possible is what might be said to loosely be a lack of accountability; that is, without military-developed equipment, there is likely to be far less (if any) classified material/equipment onboard such an aircraft[71]. This mean that, other than software or payloads that may or may not be included on the aircraft (depending on the control method being utilized), in the event of a crash, failure, or shoot-down, there may not be an immediate need to retrieve the aircraft. Retrieval can be exceedingly costly and time-consuming in addition to the fact that soldiers on the ground are usually needed for such a retrieval operation (and possibly in an unfriendly area)[72]–[74]. For this reason, using COS equipment as much as possible can return both a tremendously lower lifetime-cost and an increased ability to send a system into a location that normally would not be approachable since the loss of a low-cost unclassified UAV would likely be considered an acceptable risk. Furthermore, since UAV crashes are exceptionally more common than manned aircraft crashes[75], such savings are likely to be even higher than might be had with a similar situation involving manned flight.

One of the common COS systems used in multiple instances of research has been the combination of the Pixhawk flight controller, a companion microcomputer (often a Raspberry Pi), and software built using Python utilizing MAVProxy, MAVLink, and Dronekit[67], [76]. This is a system that not only allows for aircraft control and the testing of a wide range of control methods using unclassified equipment, but while the Pixhawk flight controller and Raspberry Pi are cheap, Python, MAVProxy, MAVLink, and Dronekit are all free and open-source. This allows for extensive testing to be performed without any fear of classified equipment loss (save for any classified or export-controlled software loaded on the microcomputer). Additionally, Dronekit includes a software-in-the-loop (SITL) that allows very accurate simulations to be run without the need for an actual aircraft. In turn, this means that methodology and codes can be tested and refined extensively before actually running the codes onboard the aircraft thereby minimizing the likelihood of crashes and also reducing testing time (since computer simulation is much quicker than charging batteries, driving to a test facility, setting an aircraft up, etc.).

Dronekit, MAVLink, and MAVProxy are all Python libraries that allow for unmanned system simulations to be produced on a laptop or desktop computer and for the systems developed to be easily transferrable to hardware systems. Dronekit is a Python library that communicates via MAVLink either to the Dronekit SITL or to the Pixhawk flight controller firmware to relay commands to the aircraft and receive flight data from the aircraft. CanberraUAV created MAVProxy with the express purpose of companion computing (i.e. utilizing a microcomputer to run additional codes for flight control aiding) and communicating with the autopilot system

ArduPilot[77]. MAVLink operates as a signal traffic director and communications protocol so that commands and information can be split and/or sent to UAV flight controllers, ground station mapping software (such as ArduPilot), simulation software (such as Dronekit SITL), and other UAV equipment using address-specific inputs[78]. In this project, other than for initial testing and setup, ArduPilot is not used since other user interface (UI) outputs are made and used instead. Furthermore, as the aircrafts in this project are meant to fly with mixed control intelligence, there is far less reason to have numerous aircraft outputs sent over the telemetry device(s) that would otherwise crowd the wireless communications system.

One of the primary problems with large groups of swarming aircraft that has been attempted to be solved through research has been that of mass communications. The main problem is that, by sending data from a ground station to each of the aircraft, too many data links must be established and used for a dependable system to operate[79]. Some of the attempts to overcome this problem have included timed pheromone release[80] and ad-hoc inter-aircraft communications networks[67].

Digital pheromone communications have shown much promise for usage in swarming systems, particularly those involving stationary or slow-moving perimeter systems[62]. Such a system might operate in a few different ways. One type of digital pheromone scheme developed under DARPA funding proposes a future method that places small ground systems throughout an area that communicate between themselves locally to broadcast a variable strength field that alters based on local targeting information. At the same time, UAV's swarming through the area would update the system based on their own detection data as well as being guided by the strength of the fields from the ground devices. Conversely, if the situation is such that ground devices could not be deployed, the project also stated that synthetic signals could be stored by a ground station and then updated and broadcast to each of the aircraft as appropriate[62]. However, it appears that reverting to a ground-based pheromone system would nullify most of the advantages of the digital pheromone system proposed in the first place (i.e. increase of range beyond that of ground station communications, a decreased size and computational power of the ground station, etc.).

Ad-hoc communications networks are important not only for UAV swarms but also for the overall progress of multi-UAV system capabilities. Swarms of UAV's using an ad-hoc communication network operate the subsystem by relaying data between each other in a robust manner. For example, if a UAV transmits information outwards, it will likely try to transmit two different types of information: the first being its local information and the second being global information received from other sources (such as other aircraft or ground units). This means that the organization of the outwards signal must be optimized to minimize the

amount of data (as there might be an exceptional amount), and the receivers must each be able to digest and interpret mixed incoming and possibly incomplete signals. However, this type of system allows for signal/vehicle loss and therefore allows an extremely robust swarm to operate. This can be especially helpful in disaster relief mitigation and cellular signal extension among other uses.[81], [82]

Ad-hoc systems have been designed and proposed by many research projects for more robust UAV swarming activities. One example of this is explained exceptionally well by Allen in his Master's Thesis (Design and test of a UAV Swarm Architecture over a Mesh Ad-Hoc Network) where he designs a real-world swarming matrix to use not only ground-to-air communications but also air-to-air communications for a true ad-hoc network. Thus, rather than only being able to control each aircraft with the direct control from a ground station, this allows the ground station to emit a single signal that tells all the aircraft how to alter their control in a single command[67]. However, as shown in Allen's work and others, ad-hoc networks pose especially difficult among large swarms of aircraft that other systems often do not encounter.

Perhaps the most prominent obstacle UAV systems encounter compared to other vehicular networks is the combination of quantities and variability. While vehicular ad-hoc networks are difficult to manage, when using ad-hoc communications in a UAV swarm, four different major limitations are generally encountered. First, as already discussed at length, the weight of communications systems and the power input must be limited and optimized for use onboard a UAV. Second, UAV's tend to have a wide range of operating speeds (from hovering to 100mph or faster) which can cause delays and difficulties in obtaining accurate real-time data from an aircraft unless the data rate is exceptionally quick. Third, UAV's—unlike most other ad-hoc network-operating vehicles on the ground—move in three dimensions; while this does not always change the air-to-air transmission capabilities, this can limit air-to-ground and ground-to-air sections of an ad-hoc network. Lastly and similarly to the issue of adding a third dimension into the network is the reduction of pre-path knowledge compared to ground-based networks. Unlike road-going vehicles or ground robots, UAV's move in extremely variable paths and group densities which can cause tremendous stress to ground-based parts of an integrated ad-hoc network. Furthermore, this can cause difficulties in air-to-air communications, as each agent must be optimized for both long-distance communications as well as large quantity rapid local communications. [83]

Recent research from Brazil has also attempted to solve some of the problems involved in making swarming methods practical and applicable to real-world UAV missions. As discussed in the research, most modern theoretical methods of swarming do not work well when actually applied to real-world swarming drone systems; this can be due to a number of reasons, but often simply because field actions require much more robust algorithm construction that can take varying

situations and work climates. However, according to this research, Moraes and Freitas state that pheromone communications are indeed part of a robust solution if implemented properly. In their research, they attempt to develop a communications system that not only is robust—as mentioned—but can work in a number of different types of missions. Because of this, they develop a system that works with points-of-interest (POI's). Thus, this would be applicable to many search algorithm systems, especially those that run some sort of field detection (such as radiation, electromagnetic, etc.) where POI's are often developed and narrowed down. However, part of the pheromone complex that Moraes and Freitas lay out notes that, rather than only utilizing a conventional pheromone communications system to share data along that paths of individual UAV's, POI's themselves should be assigned a separate value, which the authors label as R. If the R-value is positive, then the UAV's will be drawn to the location while if negative, they will be repelled. Furthermore, a positive R value will affect all aircraft while a negative one will only apply to local aircraft[84]. A system similar is likely to be extremely useful in negating known false positives found with Bayesian Statistical searching.

Using this type of attraction/repulsion system, the same research also states that integrating the R value into what essentially is a field affect equation and then summing the resulting magnitudes can be used for each aircraft to determine the direction in which it should move incorporating POI's, nearby objects, and other aircraft in the vicinity all into a single calculation. The equation proposed by the research can be seen as Eq. 2.5 where S_{marker} and K are constants based upon the marker/POI set field strength and the field decay rate respectively. As can be seen though, the R-value is represented as the exponential value that the distance between the UAV and the POI. Each total field value is then supposed to be added to a total magnitude sum to determine the entirety of the directional control input for the UAV in question.

$$F(t) = S_{marker}\left(X_{UAV}(t) - X_{marker}(t)\right)^{R} e^{K(t-t_0)} \qquad \textbf{Eq. 2.5[84]}$$

However, while the research from Moraes and Freitas is indeed a large step forward in adequate swarm control theory and methodology, there are a few shortcomings as far as the applications of the PADUA effort is concerned. While Moraes and Freitas argue in favor of a field-based separation method, such a system is far more computationally intensive than the swarm instances governed by PADUA are likely to require. In other words, while the field-based system operates excellently compared to a more rudimentary digitally-stepped method that might be based upon an aircraft's distance from a POI or other aircraft/object, as the number of vehicles increases, so does the computational time to determine each aircraft's next desired position movement command based off the algorithmically determined prediction. Specifically, since the method described by Moraes and Freitas means a field affect value (F(t)) must be found for each UAV

and POI relationship, the number of F(t) values will grow and change excessively during a routine Bayesian search pattern due to the number of variations in an early probability field. This is moreover because, for each dataset recorded by an onboard radiation detector during a Bayesian search sequence, there is expected to be a record of the count rate, calculated Bayesian likelihood of the radioactive source's presence and the location for each detection instance. Thus, rather than updating *n*-number of POI relationships, false positives also would have to be registered as negative-R POI's. For example, using only a very rudimentary test that develops basic lists of array data (count rate and location) for *n*-number of aircraft shows that the field affect variable F(t) must be calculated twice per *n*-index (representing latitude and longitude) to cover positive and negative *R*-values. The approximate per-iteration time required to calculate each *n*-instance was approximately $2x10^{-6}$ seconds. However, while this might not appear to be very high, this result was found using very rudimentary calculations with only the bare essential calculations included in the short test code. With additional equations added into the code to determine POI strength constants, R-values, and others not to mention the additional memory required to keep track of each of the POI variable sets, using such a system for all POI's is not very feasible for this project without additional data structure changes (especially during later iterations since the time required to append to a list in Python increases with the size of the list). However, this does not mean that the field-based system described cannot be used for certain POI's. It is in fact a very simple and effective method in its fundamental nature to assist with swarming agents. Thus, rather than applying the system to all POI instances in the project, it should be considered for application to POI's such as high probability locations, low probability locations, other aircraft in the vicinity (including those that are in the swarming pattern and any that might not), and high danger areas. This can narrow down the list of POI's significantly and still produce a similar result. In other words, while Moraes and Freitas do not specifically describe the process of determining what qualifies as a POI, a system must be used to state what type of point is and is not a POI; otherwise, certain types of missions might fall into a problem of over-calculation thereby using excessive time and memory.

Modern Swarming Algorithms and their Relation to PADUA

In addition to BOIDS and PSO framework theories discussed in Chapter 1: A Background History of Swarming Theories and Applications, there are other modern swarming theories from which the Bound-Continuity method used in PADUA originates. While they are primarily distant derivatives of BOIDS and PSO methods, they have intrinsic qualities that were used as fundamentals in the development of the Bound-Continuity method. These secondary methods include primarily Dynamic Group Optimization methods and overarching theories used in insect-based algorithms.

Dynamic Group Optimization (DGO) is a swarming method that attempts to solve the discrepancy between local and global optimization that occurs with many PSO algorithms. Because PSO algorithms use a combination of local and global optimums to determine a solution/pattern for a swarm problem, if a local optimum is greater than many surrounding local optimums, it will attract many agents which can then bleed through the system. If a global optimum is not found or appears to be a lesser optimum than the overweighed local one, then it is liable to be lost with a local optimum serving as a global one. While there are numerous mechanism that have been used to try and counteract this tendency in PSO algorithms, DGO takes a slightly altered approach to solve the problem. [85]

Rather than only using a single large swarm with individual local optimums, DGO algorithms utilize a centroid method among smaller sub-swarms. These smaller groups update their local centroid agent/data point with their combined intelligence, but the centroid can also be updated by other better performing sub-swarms. Thus, although each small group acts on its own to search a small region, each small group also updates the overall swarm knowledgebase.[85] This is extremely similar to the techniques used in the triple-tier layout of PADUA as well as the design of the swarm systems.

One of the methods that DGO utilizes to minimize the likelihood of local optimum false positive entrapment is the multiplication of random percentages throughout the position assignments of the individual agents. This means that, although an underlying control algorithm might send the agents to certain locations, the random values act as relaxation factors and purposely skew the end location assignments slightly. This increases the likelihood that other local optimums might be found at a negligible accuracy expense. [85]

The primary equation that DGO is designed to use for ensuring population/search diversity can be seen in Eq. 2.6[85]. In this equation, the variable x denotes the location or value (depending on the implementation of the DGO algorithm) of the local agent j of the i^{th} sub-swarm in the midst of the k^{th} update iteration. If the random value generated (signified by the designation *rand*) is less than the mutation probability (*Mr*) then the new position assignment must be modified appropriately. The updated position assignment is determined to be equal to the previous assignment's location added to the random-value-weighted difference of the local sub-swarm centroid and the previous individual location (i.e. a randomly determined location between the two locations). The fixed-weighted (w) difference of the global optimum (G_{best}) and the previous individual position is then added as well. However, if the random value generated is greater than the mutation probability, then only the difference of the local optimum and previous individual position coupled with the random value within the set step size (μ) is used. As can be seen, merely in the population diversity assurance part of the DGO algorithm,

random values and random percentages play a critical part in ensuring the algorithm does not funnel itself into a false positive. [85]

$$x_{i,j,k+1} = \begin{cases} x_{i,j,k} + rand\left(C_i - x_{i,j,k}\right) + w\left(G_{best} - x_{i,j,k}\right) & rand < Mr \\ \left(C_i - x_{i,j,k}\right) + \mu & else \end{cases} \quad \textbf{Eq. 2.6[85]}$$

While DGO does improve greatly on the problems discussed with PSO algorithm false positive issues, it still has had other improvements made to it by other researchers in the past few years. One such example of this is the Cross Entropy Method based Hybridization of Dynamic Group Optimization Algorithm. Rather than only using randomly weighted modifications of PSO-based algorithms, the implementation of cross entropy in the DGO algorithm to modify individual agent locations allows for a similarly random assignment to be made to determine each agent's next position target but using the combination of two direct probability distributions. Eq. 2.7 is the governing position calculation for each agent's next position assignment (x_i). The position is assigned as the addition of the individual's mean position (m_i) and the standard deviation (σ_i) multiplied by the Gaussian cross entropy random value ($N(m, \sigma)$).[86]

$$x_i = m_i + \sigma_i * randn() \quad \textbf{Eq. 2.7[86]}$$

While this is not a severely different approach compared to the original DGO algorithm, both of these present important structures that are used in the PADUA algorithm. First, the concept of using sub-swarms rather than simply one large swarm is relied on heavily for the efficient usage of PADUA. The final PADUA algorithm is designed for *n*-number of aircraft, but realistically, it is likely that it would operate far better if multiple small groups, each made of 2 to 4 agents, were utilized for the overall swarm application. In such an instance, PADUA would be used to operate each individual sub-swarm with an additional ground station compiling the information from the multiple instances of PADUA. This would allow greater maneuverability and higher speed of individual aircrafts in an urban environment (since there would be less aerial obstacles) and allow more specific targeted searches.

Additionally, the issue raised of false positives resulting from swarming algorithms (particularly PSO-based ones) is critical to the purpose of PADUA. PADUA was built with the understanding that Bayesian search methods often have the same intrinsic problems of false positives/optimums/maximums overpowering true ones due to high background, outliers, or too high of update rates. PADUA seeks a different way to overcome this issue though. Rather than trying to create a single algorithm that both explores and searches optimally, PADUA couples multiple search techniques with multiple exploration and swarm control techniques into a single more powerful algorithm. This approach minimizes calculation complexities

and optimizes the search technique based on the search area size. Control can then be altered based on the area size as well with swarm agent position assignments controlled by either weighted or unweighted targets depending also upon the size of the area being searched. This combination and sectioning into multiple tiers requires lengthier coding than BOIDS, PSO, or DGO techniques, but the result allows for a much more versatile algorithm with specific optimization methodologies per situation.

Broad Area Bayesian Search Methodology and Associated Statistics

Summary of Bayes Application to the Project

While Bayes theorem has already been stated in a historical context, modern calculation methodologies used will be discussed further in this section to present a precursor to Tier 2 search methodology. While the calculations themselves are not particularly challenging, the implementation of boundary conditions and the usage of acceptable probability density functions (PDF's) are import aspects to this project.

Bayes theorem can be stated in simple terms as being the prior probability of an event occurring given new information about the event at a different location divided by the likelihood of the new information being true. In mathematical terms, Eq. 2.8 can be read as the probability of *A* given *B* (the posterior) is equal to the probability of *B* given *A* (the updated probability) multiplied by the prior probability of *A* (the prior) and divided by the probability of the accuracy of *B*.

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)} \qquad \textbf{Eq. 2.8}$$

In this project, the posterior will be the likelihood of a target radioactive source/material being at a specific location. The new data available from one location to another will be the raw count rates detected. As *P(B)* is the likelihood of the new data (i.e. the detected count rates) being correct, it will be assumed to always be ~1 and therefore can be removed from the theorem. In future work, if multiple types of vehicles and/or detectors are used with the PADUA algorithm where different accuracy probabilities are known or can at least be estimated (such as from detector efficiency differences or detector-specific variations in efficiency), then this can easily be updated. For example, if it is known that a hypothetical plastic scintillator being used during a search is usually 90% accurate in detecting a gamma source flux within a range of 90 to 200 counts per minute (c/min) and from 0 to 90 c/min the accuracy is 50%, then values for *P(B)* can be applied to the

equation depending on the current count rate detected. Additionally, if a location such as Manhattan, New York is being searched, then lower values for *P(B)* can be given to certain areas that have known higher background rates (such as the area North of 14th street around 1st Ave. where there is likely higher background among the numerous hospitals than in a sparser area such as Central Park[87]).

Probability Density Functions and Bayesian Input Techniques

A probability density function (PDF) is a function to which a set of data can be assimilated, in simplest terms. As far as an example of using a PDF applicable to this project, if it is hypothetically considered that there is a missing radioactive source, then by recording count rates and their associated locations, a normal (i.e. Gaussian) PDF can be implemented to assimilate the rate values detected to the formula shown in Eq. 2.9. This is the generalized version of the normal PDF where σ is the standard deviation of the *x* array values (i.e. the distance from the reference) and μ is the shift of the PDF. Often, the shift used is the mean of the *x*-values; however, other shifts can also be used. For instance, in the example given (as well as the method used in PADUA), the maximum value within the *x* values will be used for the shift as it is desired to relate all other values as they relate to the maximum value found since the rates will be normalized into values of *x*. Using the maximum equates to assuming that the maximum location found thus far is the most likely location of the source.

$$y(x) = \left(\frac{1}{\sigma\sqrt{2\pi}}\right)\left(\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)\right) \qquad \textbf{Eq. 2.9}$$

Figure A.9 shows the comparison of a set of continually increasing x values applied to the general PDF equation using the mean and the maximum values of *x* as the shifts. A max-based shift allows for the highest value in a sequence to be assimilated to the maximum probability whereas a mean-based curve gives the greatest weight to the average of a sequence.

Once an array of rates has been recorded, the values can be normalized before inserting them into the PDF. This means that the detected count rate values are treated as *x*-values in the PDF. As can be seen in Figure A.9, if a single normalized rate value is higher, it will approach an *x*-value of 1, which has a corresponding *y*-value of 1. If the PDF is used a single time then, a probability distribution can be found for the set of detected values based on the maximum resulting in a probability of 1.

By shifting according to the maximum rather than the mean, a set of data such as that shown in the top part of Figure A.10 can be analyzed in a far more valuable manner as far as the scope of this project is concerned. The data shown in the top

half is similar to what might be seen when transporting a detector past a source but at a lower order. In the bottom half, it is evident that using a mean-based calculation, the resulting data is given a higher probability if a data point is near the average. For some applications this is useful (such as the application of a bell-curve for class grades), but since this project is searching for a global maximum, a max-based shift yields far more useful results (as seen with the blue curve in the bottom half of Figure A.10).

Applying PDF's to the raw count rate data before performing Bayesian analysis allows for a narrower range of values to be analyzed. Once PDF-generated amplitudes are calculated, the resulting values can be treated as estimated likelihoods and then used directly with Bayesian statistics to update a range of historical markers. Thus, rather than imputing a set of normalized values into the Bayesian calculations which might yield more false positives since there would be a higher degree of noise, inputting PDF values should output a filtered dataset that loses all but the highest probability locations. This can be seen in Figure A.11.

However, not all applications of Bayesian statistics follow the same methodology used in PADUA. In fact, the process of preparing the prior data for inputting into a Bayesian calculation often must be tailored for the specific application. Bayesian statistics are incredibly broad in scope and can be applied to a variety of situations so long as the necessary steps are taken to make the calculations yield useful and accurate outputs. One of the major problems with statistical methods is that there is often an ability for the equations to produce values that appear at a glance to be accurate, but if the input sequence/methodology is not appropriate, small errors will propagate through updates. This can return false positives and can also cancel true positives.

For example, in artificial intelligence and machine learning for example, other versions of Gaussian processing are sometimes used to prepare prior information for Bayesian input. One such technique is the Gaussian-Markov where the distribution-type used cannot necessarily be a single function-type if the data points are of unknown error correlations. Instead, a variable function can be developed using least squares from one point to the next to fit the best possible linear (or higher order) function to the data section. This allows very noisy data to be filtered for use with either Fourier Transforms or other/combination analysis methods. The autoregressive formulae with order *p* can be seen in Eq. 2.10. [88]

$$X_t = \sum_{k=1}^{p} a_k X_{t-k} + b_0 Z_t \qquad \textbf{Eq. 2.10[88]}$$

Other methods of Bayesian search/refinement analysis utilize repeated Bayesian updates using the same normal distributions as earlier discussed for the priors. Specifically in reference to a large undefined search space, repeated usage of

Bayesian calculations can refine a generalized estimate to a small region very quickly. While this is not entirely unlike the triple tier method used in PADUA, it can be dangerous to use multiple instances of Bayesian if there is a high chance of a false positive. However, because Bayes Theorem is succinct, assuming that the likelihood of false positive results is low, a repeated Bayes method used to narrow a search space is a computationally quick method requiring only minimal cell refinement. [89]

No matter what method is used for a Bayesian refinement process, the important part is that an accurate process is defined and used for the specific situation. PADUA is designed to be as versatile as possible and be able to be implemented with minimal algorithm changes if a different non-directional detector is used which is one of the main reasons that a max-shifted Gaussian PDF is used. In most instances, a maximum value (whether it be count rate, voltage reading, light intensity, etc.) is often correlated directly to the target's location. A Gaussian PDF is easily shift-able and has a very generic curve that happens to be well associated with the inverse square law, which governs ideal radiation flux with distance increases. The comparison of PDF-generated values based off the normalized theoretical count rate versus the theoretical count rate generated using only the inverse square given an initial value can be seen in Figure A.12. This proves that a Gaussian distribution acts as a filter for lower count rate values but still returns a small distribution range thereby narrowing the likelihood location dramatically. More detail is given about the specific use of PDF's in PADUA's Bayesian processes in Chapter 5: Tier 2 Experiments.

CHAPTER 3

MATERIALS, SETUP, AND EXPERIMENT DESIGN

While not delving into the details of the individual experiment operations and design, this chapter will present an overview of the different subsections that are integral parts of the testing and algorithm designs. Primarily, the different codes will be described in terms of their operability and how they fit into the overall algorithm itself. The hardware setups used will also be described that were used for hardware-in-the-loop functions.

Simulated Aircraft

The first part of the overall algorithm to be discussed is that of the simulation method used. Dronekit from 3DR[90] is the primary software platform being used for this task; in particular, Dronekit SITL (Software-in-the-Loop). Although it has some issues in certain types of control, Dronekit is a very valuable resource that has proven its worth through its usage in numerous other experiments as mentioned in Chapter 2. For the purposes of this project, simulation for multirotor-type aircraft will be the only type used although allowances for the implementation of fixed wing aircraft, rotorcraft, and rover vehicles are also available.

Dronekit itself is a control library built with Python intended to work with a Pixhawk flight controller. With Dronekit, full control of an aircraft using a Pixhawk controller is simplified compared to using only direct flight-controller commands. While the use of a higher-level library (such as Dronekit) does increase latency slightly, more time and effort can be dedicated towards the construction and refinement of primary algorithms and framework structures rather than on controller interface intricacies. This enables much more complicated higher-level algorithms to be developed in an initial research environment without worrying about lower-level firmware coding. Dronekit coupled with MAVLink is able to handle the communication and translation of instructions from the algorithm to the Pixhawk controller. For example, a simple instruction to send an aircraft to a waypoint location can be relayed to the Pixhawk controller via Dronekit with short “goto” command (structured as “goto(waypoint latitude, waypoint longitude, altitude)”) rather than sending lengthy instructions to the aircraft to control every actuator and motor to move the vehicle from point A to B. Furthermore, the high-level coding syntax tends to have far more readable commands compared to complex low-level code used for firmware.

The second aspect about using Dronekit that is advantageous to development and testing is the Dronekit SITL portion of the library. An SITL allows for algorithm testing to be performed without the hardware via replicating it in software. In general, three levels of testing can be considered to be useful in most systems

engineering: Software-in-the-Loop (SITL), Hardware-in-the-Loop (HITL), and Real World. For the purposes of this project, SITL and some HITL testing will be used. To explain in more detail, SITL is the lowest level testing for a software-hardware combination. This involves a pure simulation where the active algorithm(s) interacts with a software form of any hardware. HITL differs in that, while the active algorithm processes are the same, at least some of the hardware that it interacts with exists in stationary form in the testing environment. For this project, the radios were used in certain parts of testing as HITL (this is discussed further in Chapter 4). The aircraft was always used only in SITL form. Finally, for Real World testing, the active algorithm works in full operability with active hardware. For this project, this means that the algorithm would operate with fully functioning and flying aircraft. However, the purpose of this project was to create the PADUA algorithm and not necessarily to perfect a prototype.

Early Work and Initial Efforts

There were numerous tests performed utilizing different types of hardware and software prior to creating the final system(s) used for the experiments described in this document. There were a few requirements that had to be incorporated into the final system that were used as the groundwork for each stage of system development.

One of the most important parts of the entire simulation system is that of MAVProxy and MAVLink which allow instructions to be routed to different addresses from the active algorithm to the location of Dronekit-SITL and also from the SITL to the port location of Mission Planner (although Mission Planner is not used other than for initial testing). See Chapter 2 for the differences in MAVProxy and MAVLink, but the primary difference is MAVProxy serves as a routing tool for command and position data between an aircraft and a ground station while MAVLink serves as a communication protocol for aircraft flight controller firmware interface. Dronekit-SITL is located at the TCP port location address 127.0.0.1:5760, but the SITL can be contacted on any third variation of the port (i.e. ‘:5763’, ‘5766’, etc.). Only a single instance of a simulated aircraft can easily be initiated per computer however, using the MAVLink/Dronekit combination. MAVProxy also allows the output from the SITL to be interpreted by Mission Planner if desired. Mission Planner is a software package from ArduPilot that allows for easy basic autopilot functions and map output to be used in conjunction with a UAS. Figure A.13 shows an example of the main screen from Mission Planner where commonly used gauges are available for use on the ground station including the artificial horizon, vertical speed indicator, ground speed indicator, altitude indicator, and compass. Additionally, the battery information can be seen in the lower left corner of the artificial horizon, the mode (i.e. “Guided”) in the lower right, and the GPS fix below the mode. The gauges are highly customizable, but depending on the number of instruments and connections available on the aircraft, certain ones may or may not be able to be

used. For initial setup and familiarization with the system, Mission Planner was used in this project, but once the framework system was developed experiments were started, it was no longer needed as all required outputs were obtained as simpler outputs from Python with lengthy data logs needed for post-processing analysis.

The first system for framework design during early testing was primarily used to become familiar with the intricacies of Dronekit and MAVProxy. This system was developed using Hyper-V virtual computing tool. There were some compatibility issues that were found with running MAVProxy on a Windows 10 OS, but Mission Planner did not operate well on a Linux OS at the time. While there are numerous ways to overcome this, the easiest method found (and the method used at first) was simply to use a virtual machine on the Windows OS. It was decided to use Hyper-V due to its excellent operating capabilities with Windows and its ability to operate without cloud services. Many virtual machine software products considered used cloud services for the virtual computing rather than solely the laptop itself which posed a potential security issue due to the bounds of Export Control required for this project when storing and operating the active algorithm. By using an Ubuntu desktop OS for the Hyper-V virtual machine, MAVProxy was able to operate smoothly on the virtual machine along with the simulation algorithm and was still able to communicate through UDP ports to Mission Planner, which was on the Windows 10 OS. Figure A.14 shows how this setup operated. This shows the SITL running on the left side of the subsystem diagram in the Ubuntu OS on the Hyper-V virtual machine. On the right side of the figure, an early test of a main algorithm was run. MAVProxy (shown in the middle section of the Ubuntu subsystem in the diagram) connects the main test algorithm and the SITL with TCP port 127.0.0.1:5760 and connects the SITL to Mission Planner using UDP connections where the first part of the UDP address is the virtual computer IP address. All of these parts were hosted on a single laptop running Windows 10 Education (except the virtual machine, which was using a full Ubuntu OS). One important part of the setup to mention was that the firewalls protecting the laptop against unauthorized UDP connections had to be rewritten manually to turn the firewall off. While this is not the safest method, fortunately a better system setup was implemented shortly thereafter.

The next setup used to build the system framework was a Linux subsystem (using Ubuntu) directly running on the Windows 10 machine. Due to the fact that the Windows Store was not available on the laptop being used, the subsystem app could not be downloaded. The Windows Store is not easily downloaded, as it is a part of the current Windows OS itself rather than add-on content. As such, the actions of the Store simply had to be performed manually. This consisted of enabling the windows subsystem for Linux via the PowerShell, downloading the Ubuntu distribution package, installing it via the PowerShell by invoking a web request, and then setting up the subsystem as one would any new OS (setting the

username and password, etc.). Additionally, certain .dll files had to be rewritten to allow subsystem usage. While tedious, once operational, the subsystem operated better than any virtual machine for this project for two primary reasons. The first is that there is no extra memory that must be allocated for the hard drive of a virtual machine. The second (and more important part) is that, because the Linux subsystem operates as its name states—as a subsystem rather than a virtual machine—its location path is a subsidiary of the main hard drive rather than a separate partition. This means that, rather than having to deal with rules and firewalls governing the communication between two different machines, although still difficult compared to operating through a single OS system, sharing data and files is easier than using a virtual computer. This means that running the Linux subsystem is far more similar to running a program on a Windows OS computer rather than running two partitioned machines on the same computer. While the subsystem still does not allow for easy file transfer from one location to another within the main drive via the subsystem, the main OS can access files in the subsystem. Note also, with a full version of Windows 10 that does have the Windows Store, installation of the Linux subsystem is far easier.

As can be seen in Figure A.15 is the original desired final setup of the system placed next to the diagram of the designed original system. The piece of the diagrams in this figure that should be observed in particular is the interface between the Python codes and the aircraft package compared to the interface between the Python codes and the simulated aircraft package (which includes the simulated detector) and Mission Planner. Although this is only the initial design for the algorithm operation, it shows that great efforts were taken to keep the design of the algorithm's operation in simulation as similar to that of a real-world operation as possible so that later transition from simulation to a real-world environment would be near seamless at the end of the SWARM project.

Once the Dronekit and Dronekit-SITL libraries were familiarized during initial testing, as stated earlier, there was no reason to use Mission Planner anymore as it is not a particularly useful UI for this project and just introduces additional unneeded complexities. Referring back to the previous diagram (Figure A.15), it can be seen that without Mission Planner, the three main parts remaining are MAVLink/MAVProxy, the main algorithm, and the SITL simulated aircraft. Furthermore, the primary problem with MAVProxy on Windows was its communication with Mission Planner. The only reason that a subsystem/virtual machine had to be used at all was to run MAVProxy, but since MAVProxy is only needed for communicating with Mission Planner, the subsystem/virtual machine part of the system was able to be disposed of completely. When running the system with only the SITL and not Mission Planner, only MAVLink protocol must be used. This simplified the system framework tremendously at the expense of needing to develop a UI output in-house. Although the virtual machine and subsystem methods were not used in the final framework for PADUA, these early

methods allowed many of the small intricacies and operating limitations of Dronekit to be found. Furthermore, for other parts of the SWARM project, the subsystem method in particular was still used for diagnostics and testing purposes.

Overall Algorithm Layout

When running any of the versions of PADUA, there are primarily two separate parts of the algorithm that must be run separately. The first part is the ground station half of the algorithm while the second runs on the aircraft. In a real-world environment, the aircraft part of the algorithm would be run onboard the aircraft via a companion computer while in simulation it runs in a separate terminal from the ground station (or on a separate computer so long as radio HITL is used). In most cases, there are four major codes that are used as these two halves.

On the ground station part of the algorithm, the two codes used are 'Ground_Station_Main.py' and 'Ground_Station_Codes.py.' Ground Station Main is the front-end part of the ground station-oriented code while Ground Station Codes is the class library for Ground Station Main and used to perform the back-end computations for the ground station. Some examples of some of the codes on Ground Station Codes include the communication string concatenation and extraction processing, the computing of the artificial response from the simulated detector, the concatenation and sending of initialization commands, serial port communication, and others.

On the aircraft, the two primary parts of the sub-algorithm that are used in the equivalent manner as the corresponding ones on the ground station are 'aircraft_main.py' and 'Onboard_Aircraft_Codes.py' respectively. Note that 'aircraft_main.py' often includes a number after "main" to represent a modified form of the code during certain experiments (i.e. 'aircraft_main2.py'). These two halves of the aircraft portion of the algorithm run very similarly to those on the ground station where Aircraft Main operates as the front-end processing code and Onboard Aircraft Codes is the class library for Aircraft Main. Many of the classes in Onboard Aircraft Codes are the same or similar as those in Ground Station Codes. Two distinct libraries were made though since each half of the algorithm should be able to operate without the other one on the same computer. Although minimal, this method reduces the amount of memory used by each computer and, more importantly, simply allows for better organization.

One additional part of the Aircraft Main code that is not a part of the Ground Station Main is the Tier 3 sub-algorithm. While the ground station performs many of the control computations for the lead aircraft in Tiers 1 and 2, in the third tier, the aircraft flies primarily on its own using the detector as its main sensory apparatus. The processing for this is performed entirely in the Aircraft Main code.

Threading Library

One of the most important structures used in the algorithm is centered on the Python Threading library. This is a higher-level threading library though rather than a lower-level one. The explanation for the usage of high-level threading rather than multi-processing is this: because most of the threads used in the algorithm are either idle or not exceedingly computationally intensive, using multiple threads rather than multiple processes allows the algorithm to be deployed on CPU's with lesser numbers of cores while still allowing for asynchronous processing.

To understand this better, some terms should be defined. First, a CPU (Central Processing Unit) consists of a minimum of one core. Most personal computers today consist of a single CPU with multiple cores (both dual and quad core are exceptionally common). A core, for the purposes of this paper, is simply a processing unit within a CPU. Thus, a dual-core CPU is a single CPU with two cores within itself.

CPU's can be connected (for example, with the hardware used for this project, two Raspberry Pi's could be connected) to double the computational power. However, while the computational power might be doubled in such a situation, a program running on such a system would suffer an increase in power consumption as well as additional latency gained during the sharing of information between the two CPU's. To overcome this most CPU cores today use multithreading or hyper-threading capabilities to separate artificially individual cores into multiple virtual cores. For example, a dual core CPU consisting of a single CPU with two physical cores can have each core separated into two more artificial cores yielding the equivalent of a quad-core system. This does not increase the overall computational power of a computer, but it does allow more tasks to be run at the same time thereby drastically decreasing the time required to perform several simple tasks. These artificial cores are often built into CPU cores by the manufacturer such as Intel[91].

The Python threading library operates similarly but not exactly the same as the multithreading abilities built into the cores. Most important to understand for Python multithreading capabilities is the Python Global Interpreter Lock (GIL). The Python GIL is a controversial but intrinsic and useful aspect of the Python language. The Python GIL is can be summarized in the following description. Python programs can only run a single thread at a time (unless implementing a multiprocessing library). Since Python can only run a single thread, a multithreading library in terms of this project obviously cannot operate in a true multithreaded manner where individual threads would run concurrently and independently. Rather, when two threads are run using the Threading library, rather than running both simultaneously as conventional threads would, Python switches between the two threads rapidly (every 100 bytecodes) thus giving the illusion of and many of the capabilities of multithreading. [92]

Since the GIL prevents Python from running multiple threads and multiple concurrent processes, then the advantages of using the GIL must be explained in more depth to better understand the reasoning behind the GIL's implementation to PADUA. While basic functions can often be run faster in series than in multithreaded form, the advantage of sharing variables efficiently is lost in a series code. For example, the codes in Appendix C: Example Python Basic Iterative Multithreading Code and Appendix D: Example Python Basic Iterative Series Code have essentially the same simply calculations where *i* is iterated from 0 to 100 million twice, but the time to run the series-written code is just over 15 seconds while the threaded version takes 44 seconds—nearly three times longer. At its surface then, multithreading appears to be detrimental to a code. However, while a very simple task such as a single loop iterative process does not need to be multithreaded, when asynchronous signals are handled multithreading allows processes to continue in some threads while other threads wait for data input. This can be seen in Appendix E: Example Python Shared Variable Multithreaded Code and Appendix F: Example Python Shared Variable Equivalent Series Code where again, essentially the same calculations are performed, but because a variable is both updated *and* being used to update other variables, the multithreaded version runs significantly quicker. While the series version was able to finish in .071 seconds, the multithreaded version required only .029 seconds—less than half of the time required for the series version. From this, it can be seen that for certain aspects that a code like PADUA might need to handle (such as asynchronous or latent radio transmissions and detector data), multithreading using the Python GIL is a powerful and useful tool.

To explain a multithreaded system's operation in more depth, an example of a time delay is provided in Figure A.16 in which the data is shown from two threads run with a delay between start times. A one second delay is used before starting the process on the second thread. As can be seen, the total time of the threads is not representative of the total time elapsed since processes are run from both threads concurrently.

With the term "threading" now understood in the context of this document, the usage can be described more fully for PADUA. There are several threads used on both the aircraft side of the algorithm as well as the ground station. Certain threads run idly when not in use. For example, if Tier 1 is not being used, the Tier 1 control thread and the Tier 1 computation thread on the ground station will still run their usual endless while-loops, but the body of both threads will be passed over using a trigger. This prevents the algorithm from having to spin up threads multiple times which can be time-consuming. This is especially important for the aircraft-mounted code where it cannot be expected that the computer being used in the field will be significant in size or power. The individual threads will be discussed further in Chapter 3: Final Algorithm Thread Design.

Communications Systems (Control String)

The communications systems used in this project can be mostly split into two main parts: software and hardware. The hardware (radios) and associated handling code will be described mostly in Chapter 4. This section will describe the software aspect of the communication systems.

The software side of the communications primarily involves the command string concatenation method. In order to design the algorithm with the goal of eventually being able to operate with a large number of vehicles with a minimum amount of hardware, the code needs to operate fairly simply and, as far as the communications are concerned, transmissions need to be kept as short as possible.

To accomplish this, an encoding method was created that allows for a single string of numbers to be compiled by the controller that can be transmitted to the receiving body (i.e. the aircraft or other vehicle), translated back into usable commands, and then relayed in its decoded form to the vehicle controller via a wired connection. Because this communication method was created primarily with the understanding that PADUA will be expanded and updated in the future, there are certain values in the string that can be altered for controlling different types of vehicles (these include aircraft, ground vehicles, submersibles, and surface-water craft although others could be added). There are also indices that are empty with only zeros as placeholders reserved for future use. Throughout all the experiments, this encoding system has been proven to be extremely reliable and is successful in transmitting complex commands with a minimal data size.

The entire key/readme for the communications string can be seen in Appendix G: Complete Command Encoding Readme File while an overview will be described here. Note that the short command sequencing used for ping signals between the vehicle and the ground station as well as inter-vehicle ping signals is shown in the second half of Appendix G: Complete Command Encoding Readme File. Figure A.17 displays an example of how several command codes might be disassembled. The first line of the figure shows the overall nomenclature for the command string (this is explained in more detail in the key provided in the appendix). Each placeholder has a significance. A, E&F, O&P, and R&T indices are all exclusively for ensuring preserved string length as a method of transmission error detection. The vehicle number (B,D,Q,S) is used in the same manner, but it also tells the receiving vehicle whether or not to use the commands within the string depending on its vehicle number designation. In other words, the ground station sends out a command intended for a specific vehicle as a burst transmission on a common frequency, so when a vehicle receives and translates the command, it will only use it if the vehicle number in the command corresponds to its own vehicle number. The control method and mode change (C and G respectively) are always single value numbers, but H and I are variable depending

on their first number (the expansion method can be seen in the appendix and in the last part of Figure A.17). J,K,L,M, and N are all reserved spots for future use as it is expected more command types will be required such as for emergencies or search method changes. In the middle part of Figure A.17, an actual example is shown sending a routine empty command to Aircraft #1 with no changes to mode, commands, or altitude. The string length is thus at its minimum (20 digits). In the bottom part of Figure A.17, an example is shown that sends a command to Aircraft #1 changing the mode to guided, sending the aircraft to a waypoint, and changing the altitude all in a single command. It can be seen that placeholders H and I are expanded appropriately to accommodate the coordinates for the waypoint and the new altitude. Thus, the string length is increased to 43 digits. Using this type of nested method, the minimum string length possible for a specific command is always used with enough options to increase the string length as needed to send the required amount of information. During initial testing, the speed at which this method allowed communications to operate was faster than the time needed to turn on the transmission LED on the radio baseboard.

Simulated Detector

One of the early issues found with simulating the complete system developed for this project was how to simulate the radiation detector both accurately and efficiently. Past software used in other projects would prove to be rather difficult to use in terms of memory allocation and the amount of time required for running initial setups. The primary goal of the artificial detector developed from scratch for this project was to easily store simulation information such a way that the simulated detector could be used anywhere and anytime regardless of the region’s size with minimum computing capabilities.

In past experiments, the Monte Carlo N-Particle Transport Code (MCNP) from Los Alamos National Laboratory (LANL) was used to replicate source(s) in a region and the flux that should be observed at any location away from the source[93]. However, there were some issues with using this program for this particular project. While MCNP is an extremely accurate and versatile program, it does have two main problems. The first is the time it takes to run a simulation. While the newer versions of the program do run faster than the older ones[94], the time required to run a simulation with centimeter accuracy over a large volume can still require a considerable amount of time due to the need to calculate the estimated flux (and any other variables desired) at each cell in the simulated region. This leads into the second problem, which is that of storage. For an area the size of a football field, there might be thousands or tens of thousands of cells (depending on the refinement parameters) in an MCNP output. However, this project looks at search improvements not just for football field-sized regions but also for regions that can be the size of Manhattan or even a location as large as Eastern Ukraine. This is not a theoretical-only system; this project is intended to have a final product that is

near ready for real-world use. As such, running lengthy simulations on a separate program does pose an issue with experimental time. Furthermore, as stored lists of locations and flux values become large, the time required to lookup location data and interpolate the flux increases substantially as do the memory requirements.

To overcome these issues, rather than using MCNP and a lookup/interpolation system for detector simulation, a program was written based upon real-world laboratory calibration. The detector simulation code classes were written using Python 2.7 (which means the simulation can be performed in real-time in-line with the active algorithm) and operates off a combination of premade Lambda functions binned data. Furthermore, unlike MCNP, only the raw count rate is generated for a specific location from the source since this is the primary data type used for the PADUA algorithm search sequences.

The method developed and used for this project operates on the theory that, if the location of a simulated detector is known in relation to a simulated source, then a false signal should be able to be produced with two primary steps. The first step is to estimate the approximate location-specific average flux signal by using a previously developed best-fit multi-order logarithmic curve relating average count rate and distance from the source. The second step is to add a random noise signal based on the previously determined distribution at a known distance from the source from laboratory calibration results.

To write the simulated detector code, the entire process can be dismantled into five main phases. The first step consists of laboratory calibration. The scintillator detector package used for this project consisted of a single plastic scintillator, a photomultiplier tube (PMT), and a USB mount for the PMT coupled with a laptop (see the following “Hardware” section for a complete description). Multiple samples were taken at various distances from a radioactive source. The source used for the first tests in this project consisted of a single Cesium 137 10µCi button source. The data information about each of the sample locations is shown in Table B.1 for the Cesium source. All tables are shown in Appendix B: Tables. The minimum, maximum, and average count rate values are also displayed in Figure A.18 where the range of rates detected can be seen to be much noisier closer to the source. However, the average values correlate very closely to the expected inverse square die-off as should be expected. It should also be noted that while 180 test samples were used for most of the data sets, only 60 were used for the last two samples; because the majority of the response at these locations was background, the sample time grows exceptionally large with little change in the detected rates. Additionally, since the count rate decreases and the PMT base operates based on a fixed buffer size (340 counts), the length of time to obtain the same number of counts at a further distance from the source compared to close to the source increases substantially. Also, although 180 samples might appear to be too few, realistically, the number of samples to obtain the average count rate is 180 times

the number of buffer samples (i.e. there is a total real sample size of 61,200 for a sample size of 180 and 20,400 for a sample size of 60). *Note also, for the purposes of making string conversion during post-processing easier, the first and last values are excluded making the usable data set lengths 178 and 58 respectively.*

Once the sample data has been obtained, further manipulation is required to obtain the required values for the Lambda functions as the second phase of the process. The first set of values is the average count rate per sample location. Using only these values, an extremely accurate prediction curve can be produced to obtain the estimated source strength at any location including background radiation. However, if only this value is used, then an accurate representation of any predicted value cannot actually be obtained due to a lack of noise. Figure A.19 shows the estimation curve based upon only the average values. Additionally in the figure, the average background rate measured in a separate test without the source is also shown; the average rate curve can be seen to have an asymptote at the same value as the background rate.

With this data, the first Lambda function can be produced. A Lambda function is a function that can be stored inside a program and reused so long as the correct input in provided similar to a defined function but with shorter syntax. The input for a Lambda function must be the coefficients of the equation that it represents and any independent variables. The function type used for this project consists of a fourth order logarithmic function which is utilized as a best fit curve for the average source strength's change over distance. The reason for using a fourth order function is simply a matter of calculation time. It was found that the higher the order used, the higher the accuracy but the longer the computation time. A third order curve also yields an acceptable result and at a lesser computational cost, but a fifth order requires too much time to run. A fourth order curve then is implemented for this project although a third order curve could be used if desired. The lambda function used can be shown in Eq. 3.1 in code form and in Eq. 3.2 in numeric form.

$$\text{rate\_eq} = \text{lambda x: (W[0]*numpy.log(x)**4)+(W[1]*numpy.log(x)**3)+ (W[2]*numpy.log(x)**2)+(W[3]*numpy.log(x))+W[4]} \quad \textbf{Eq. 3.1}$$

$$y = (C_1 * \ln(x)^4) + (C_2 * \ln(x)^3) + (C_3 * \ln(x)^2) + (C_4 * \ln(x)) + C_5 \quad \textbf{Eq. 3.2}$$

$$\begin{aligned} where \quad C_1 &= 0.2589573951591668 \\ C_2 &= -13.301208919349744 \\ C_3 &= 202.67612225542177 \\ C_4 &= -1240.005893669782 \\ C_5 &= 2720.3784702269936 \end{aligned}$$

Natural log values were decided to be used due to the curve similarities with the Inverse Square Law. The Inverse Square Law for a uniform spherical distribution can be seen in Eq. 3.3, which can be used to calculate the theoretical radiation exposure amount from a source given the distance from the source. In the equation, the total power (i.e. flux) is represented as *P* and set as a function of the

initial ground-zero intensity of the source and the distance from the source *r*. The denominator is in the form of flux through a small area of the surface of a sphere or shell. However, to use the Inverse Square Law, the initial intensity must be solved for at a point using the known values for *P* and *r* at the same point. This means that even though the experimentation method is really looking at a point location rather than an area location, the constants from the surface area equation (*4 and* π) are absorbed into the intensity constant when calculating it anyways.

$$P = \frac{I}{4\pi r^2} \qquad \textbf{Eq. 3.3}$$

However, as can be seen in Figure A.20, while perfect for theoretical measurements, a calculation using only the Inverse Square Law is not accurate due to the lack of background radiation included in the calculation. The diagram shows three plotted lines. The blue curve represents the source response generated from the natural log-based lambda function while the green shows the generated curve using only the Inverse Square Law using the first point as the known reference point in terms of distance from source and detected count rate. The orange curve shows the known values from the calibration test performed in the laboratory. As can be seen, the results from the lambda function are extraordinarily well representative of the actual response.

Once the Lambda function is generated, the estimated perfect rate value should be able to be found at any given distance from the source, but further work must still be performed in order to incorporate accurate replicated noise into the artificially generated signal. A binning process was decided to be the best method for tackling this problem as the third stage of the overall process. First, each location's sample set was split into 20 evenly sized bins in the range between the maximum and the minimum values for the data set (i.e. 5% increments). An example at 200 mm is shown in Figure A.21. As can be seen from the test results, given a random set of values in the range of the sample set, the values with the highest probability of being chosen should be near the average of the range.

These bins can be seen as the percentage probability per bin more clearly in the accompanying Figure A.22. Here, the random probability of choosing a bin around 67 is far higher than choosing one on the lower or higher end of the spectrum such as 60 or 77. On the vertical axis, the values shown are 0 through 180 (i.e. the number of data samples taken at 200mm). On the horizontal axis, the count rates are shown for the corresponding bins.

Once the bin information is available, replicated noise can be developed using two random values as the fourth step in the process. The first value is determined as a random value between the minimum bin value (0) and the max value (178 for most of the instances discussed thus far). Once this value is found, the appropriate

bin can be determined. For example, from the previous figure (Figure A.22), the bin range is from 0 to 178. If the first random value generated is 140, then the bin that it falls into is the twelfth, which has a vertical axis range between 128 and 145. This is displayed is Figure A.23 with the purple line indicating the bin value of 140 coinciding with the use of the twelfth bin. Referring further back, Figure A.22 shows that there was approximately a 9.6% chance of generating (17 occurrences out of 178).

With the bin placement known, a second random value is then generated as an integer between 0 and 100. This gives the placement within the bin previously found. From the same example, assume that the second random value generated is 65. The twelfth bin has a count rate range of .996 (68.98-67.99). With this information, the final calculated count rate value is 68.63 (67.99+(65%*.996)). This can be seen in Figure A.24 where the small red star signifies the value determined through the noise-replication process.

As the last part of the noise generation part of the process, the mean of the count rates at the simulated detector location can be subtracted returning a final value that represents only the noise at the location. This is shown in the following two equations where Eq. 3.4 displays the process in code form using Python syntax and Eq. 3.5 shows the numeric form. The second step of the noise-generation process is shown first where the bin size (remembering that all the bins are equal size) is multiplied by the random percentage generated (i.e. the second random value found earlier). This is then added to the bin determined by the first randomly generated value. The mean of the overall data sample at the specific location is then subtracted from this result to give the overall noise-replicated value.

```
noise=(((bin_size*r2)+(bin_ref-bin_size)))-statistics.mean(d)
```

Eq. 3.4

$$\boldsymbol{noise} = \Big(\big((\boldsymbol{bin\ size}) * (\boldsymbol{random\ value})_{\boldsymbol{percentage}}\big) + \big((\boldsymbol{random\ value})_{\boldsymbol{instance}} - (\boldsymbol{bin\ size})\big)\Big) - (\boldsymbol{data\ sample})_{\boldsymbol{mean}}$$

Eq. 3.5

Here it should be stated that the way the background is incorporated into the generated signal is straightforward. The background calibration results are found in the same manner as the source calibration but without the Lambda function (since average value should be the same at any location). By preloading different types of background into the classed code (for example background calibration data taken above water, in a forest, etc…) different types of background can be incorporated into a simulation. Both the source-influenced value and the

background-only value are generated at each detection instance and the higher one is always used. This means that if a location is far enough away from the source that the value found from the source is negligible, then only background is used. Since the calibrated values from the source included background, this means that background values are never excluded.

For the fifth and final step in the process, the artificially generated noise value is then added to the theoretical perfect count rate value found by inserting the known distance *x* from the simulated source into the average response curve (i.e. the Lambda function). To determine the noise at a non-tested location, the nearest tested location can be used to determine the amount of noise. For a more accurate result, if the location being tested is between two calibration test locations, then the noise could be found at each of the test locations and the generated value can be found by interpolating between the two based on the distance between them. With either of these methods though, the final result allows an artificial but fairly accurate count rate to be generated at any location away from a simulated radioactive source. For this project, the nearest location is used for noise generation.

Note also that only the last parts of this process must be repeated each time a value needs to be generated. Since the lab data obtainment, Lambda curve generation, and bin development are all parts of the process that are performed one and then stored, all that must be done during normal operation of the simulated detector is the noise generation and average rate generation. By finding these two parts and adding them together, the source response is able to be found rapidly and accurately at any time.

Observing the results show that this method works exceptionally well with an average standard deviation difference between the actual sample data and the simulated sample data usually as ~5%; this is important so far as the noise generation is concerned since the standard deviation is a good measurement of the overall noise of the data. A sample is shown below in Figure A.25 for a sample run at 200mm from the source. The generated signal is the one that the algorithm develops while the sample signal is the one that is from the actual calibration test. It can be seen that the noise generated is very representative of the actual data samples with approximately the same percentage and value of outliers and a very close average.

Running further tests to observe the accuracy of the signal generate method, Table B.2 can be produced showing comparison values with the data from the generated signal being run with 10,000 iterations of noise development. Keeping in mind that there are 340 buffers per sample from the actual laboratory data and 178 or 58 sample sets were collected depending on the distance from the source, 10,000 is still substantially less data than what was collected as calibration data. However,

when tests were run at 1,000 iterations, the data results were very similar, so there is no need to increase the number of iterations. The ‘Theoretical Value Calculated’ is the value found using only the Lambda function. The ‘Generated Avg Value’ Column is a representation of the average final generated count rate value (that is, the theoretical generated value from the lambda function added to the noise generated).

It can be seen in the last column Table B.2 that the percentage difference/error tends to increase as the distance from the source increases. This can be explained easily though since the count rate decreases drastically as the distance from the source increases before finally only background is detected. Thus, any small perturbations in the resulting values will yield far higher differences in percentage form. This can be seen in Figure A.26.However, although there is a slight error in the data results (as can be seen in Figure A.26), it should be shown that, in reality, the generated data is still exceptionally close to the actual data. This can be seen in Figure A.27 where it is obvious that the method described in this section works more than well enough for the purposes of this project.

Using this methodology then, a rapid signal generator can be incorporated directly into the active algorithm thereby drastically reducing the amount of stored data required and increasing the area that is possible to be surveyed in simulation. Thus, although fairly simple, this is a very powerful tool to be able to use in the simulations. Variables such as detector efficiency and type can be simply ignored as they are already incorporated into the calibrated data directly. Furthermore, by shifting the lambda curve and noise data based on the inverse square law, the source can be scaled up very easily. For example, if the first value detected was at 7.5 meters instead of 75mm, then using the inverse square law, an estimation can be used to shift the rest of the distance values from calibration knowing the relative flux values. While not as accurate as using a true calibration test with a larger source, this allows a larger source to be simulated accurately without having to introduce a possibly dangerously large source.

Hardware

The descriptions of the individual experiments describe what hardware and software is used for each one respectively. The main pieces of equipment that are used throughout this project and will be described here include the scintillator system (i.e. scintillator, photomultiplier tube (PMT), and USB base), XBee radios, laptops, and the actual aircraft for which the system was designed.

The first part that will be discussed is the scintillator system. The one used for tests and calibration in this project is made up of three parts: a plastic scintillator, a PMT, and a USB base for translating the signals from the PMT. The scintillator is attached to the PMT with optical grease while the PMT plugs directly into the USB

base. The PMT and scintillator are also bound with Teflon and electrical tape. During much of the testing, the entire package was also rolled as a single package in cloth and then wrapped and epoxied into several layers of carbon fiber. The sequence can be seen in Figure A.28.

The origins for each of the primary parts of the scintillator package can be seen in the following table (Table B.3). The PMT is a linear focused type which has a fast response compared to a box grid version[95]. The USB base came with an API written in Python 2.7; there were several issues with compatibility and complexity of the API. Due to the complexity of the API and it being written in Python 2.7, one of the major limiting factors of this experiment was that the codes should be written also in Python 2.7. While this is not a major problem, it is something that must be remembered when observing any of the code parts written in this document.

The next part to discuss is that of the hardware used for wireless communications. Although the program resulting from this project is designed to operate with minimal changes required for compatibility with numerous different types of communications hardware (cellular, satellite link, radio, etc.), the only type of hardware that was actually used for the experiments in this project was radio. Specifically, 900MHz HP (high power) XBee radios were used. The details of how this hardware was integrated into the overall system is described in more detail in Chapter 4. XBee radios have proven themselves to be excellent communications hardware for the purposes of this project. Designed and built by Digi[96], the XBee radios were reliable and rugged, but they did have high latency issues.

Using 900MHz rather than 2.4GHz limits the byte rate slightly, but increases the range of the radios significantly. This can be shown with the equation for Free Space Path Loss (Eq. 3.6). Although not shown, the Free Space Path Loss (FSPL) equation is derived from the Friis transmission formula[97]. While Eq. 3.6 can be converted to produce a more useful decibel result, the loss can still be calculated as a dimensionless value with this form. The distance between the two antennas is represented by *d* while the wavelength (λ) can be converted to the speed of light divided by the wave frequency. Thus, comparing two different frequency radio systems the same distance apart will give an exponentially higher loss value for a higher frequency.

$$FSPL = \left(\frac{4\pi d}{\lambda}\right)^2 = \left(\frac{4\pi d}{\frac{c}{f}}\right)^2 \qquad \textbf{Eq. 3.6}$$

During the project, another important aspect of the XBee radios was that they are able to be connected to a computer via USB/microUSB cable. Furthermore, a software package from Digi called XCTU is easily downloadable and extractable on Windows and was used for setting up the radio firmware. Setup included testing different methods of sending and receiving messages via Python as well as setting

up channel pairs (since the channel can be set for any of the radios on XCTU). One last setting that was changed sometimes for testing was the repetition rate; that is, with XCTU, the radio can be set to send a signal a single time or a set multiple of times. The XBees have numerous other variables that can be changed, but for this project, most of these are determined through the Python serial commands.

Some of the most integral and important parts of this project were the laptops used. The primary laptop used was a Panasonic Toughbook with a dual core i7 processor and a solid-state hard drive. This is the computer that was used for the vast majority of the coding, simulation and testing, and even for some of the real world testing for the SWARM project acting as the ground station for one of the aircrafts. Secondary computers used included two small Lenovo N22 laptops with solid-state drives running Windows 10 and a Lenovo ThinkPad running Ubuntu. The N22's were used for simulating additional aircraft during swarming tests while the ThinkPad was used for laboratory calibrations with the detector mounted to it.

Lastly, there were two aircraft used for various parts of testing in this project. Although there were no real world tests as a deliverable for this project, testing was performed with the detector and laptop onboard the large aircraft with communications to the round station via the XBee radios. This was done to ensure that the designed system was feasible and reliable for later full conversion. While the large aircraft was intended for the final demonstration of the SWARM project, the smaller one was used as a test platform for Python controllability of a drone using the Pixhawk flight controller. Both of the aircraft are multirotor-type vehicles, but the smaller is a quadcopter while the two larger ones are hexcopters.

The small quadcopter is a generic Pixhawk-controlled UAS purchased from a third party; the reason for using this aircraft is to have the option to test flight control codes on a cheaper more expendable airframe rather than using the larger aircraft that are significantly more expensive. While the real world flight control testing was not a part of this project, the PADUA algorithm was designed to operate with the Pixhawk flight controller, thus this small airframe will be used for future work by the Institute for Nuclear Security. Furthermore, due to the large size of the other aircraft, if control was to be lost during a test flight then serious damage could result not only to the aircraft itself but also to any structures and/or people in the vicinity. Perhaps even more importantly, many of the flight tests were performed only a few miles from Oak Ridge National Laboratory (ORNL) and Y12 National Security Complex. Since the endurance of the large aircraft can approach an hour, if control was lost and the UAS continued to fly, it could pose a serious security issue if it was to fly towards either of the complexes. It should be noted that because these two campuses house facilities of high national security importance such as the Spallation Neutron Source, the Summit supercomputer, the High Flux Isotope Reactor (HFIR), and the U.S. nuclear stockpile of Highly Enriched Uranium

(HEU)[98]–[100], the issue of security and flight safety had to be of utmost importance. In addition to the price and small size of the aircraft, it was also used instead of alternatives since—although small—it is still capable of lifting a light payload for 5-10 minutes. This means it should be able to accommodate a companion computer (i.e. Raspberry Pi) which is needed for command translation and autonomous flight capabilities, and it should also be able to lift the associated battery and radios required for a simulated system.

The large UAS was purchased from Homeland Surveillance and Electronics (HSE) and is a type HL6 hexcopter. It has an empty ~1hr endurance. Although it has a large payload capacity, it is limited by the FAA UAS requirement that, flying under Part 107, the UAS cannot exceed a total weight of 55 pounds. The primary material composition is carbon fiber on the aircraft, and the designed purpose of the HL6 is for crop spraying (as the AG6 with 'AG' standing for 'agriculture' rather than 'HL' which stands for 'heavy lift'). Figure A.29 below the aircraft. Note that a second identical aircraft was also bought for the SWARM project, but it was not used in the development of PADUA. While there were no experiments run onboard the large aircraft, brief testing was used with the large aircraft to ensure the ability of the radio system and certain small parts of the search algorithm were operable.

It should be noted that, although hardware is described in this section, the essence of this project was the development of the PADUA algorithm and NOT the real world testing of it. The theoretical and pseudo-real-world simulation testing were all completed during this project, but real-world control and telemetry refinement were not performed. The hardware discussed here is being used on the same SWARM project that PADUA is funded under, and much of this hardware is used for simulation testing and development (for example, the detector package was used for the simulated detector development and the radios are integrated into PADUA as wireless hardware-in-the-loop for multi-aircraft simulations). However, some of the hardware (such as the UAS) will utilize all or part of PADUA for the final SWARM demonstrations during 2020. PADUA was built to work with the flight controllers and hardware onboard these UAS, but they were not used for the development of PADUA as discussed in this document. This enables PADUA to not only allow for operations with the hardware available for the SWARM project but also with other types of aerial platforms, radiation detectors, and communications equipment.

Final Algorithm Thread Design

In this section, each of primary parts (the lead aircraft and the ground station) of the *final* algorithm are discussed in broad terms for design and process clarification. Before the main two parts are discussed, two integral parts of the codes are explained (the radio threads and the lock sequence). After this, the ground station threads are discussed followed by the lead aircraft threads. For

more information about the radio methodology, see Chapter 4, and for more information about the final algorithm construction, see Chapter 5 and Chapter 6. It is helpful to have read these chapters before this section.

The first and most important stage of any UAV algorithm development (including those used for testing and research) is organization. While the organization of smaller code segments is not always complicated, there are several aspects of UAV simulation that make the coding sequence exceptionally more difficult. These obstacles include asynchronous data acquisition, requirements for real-time analysis, multi-aircraft controls, multiple radio connections, and loss of data integrity. When organizing a UAV code, it is helpful to sketch out first which code sequences are likely to be repeated throughout the algorithm and define the boundaries of those pieces.

The first thread to be described is the radio thread. This thread is the primary handler of asynchronous data transfer since it allows the radio transfers to be performed entirely separate from the main algorithm (whether Ground Station Main or Aircraft Main as discussed earlier in Chapter 3: Overall Algorithm Layout). While built into the algorithm itself, radio transmissions and receipts are performed continuously and overwrite separate saved files that can be read by any part of the rest of the algorithm at any time. This means that the algorithm itself is able to decide if the data has held integrity and if the data is new/updated. The following diagram in Figure A.30 shows the overall process. In Figure A.30, the diagram shown displays the sequence for a receiver radio; the process for a sending radio is simply reversed. The text files that are saved/read by the radio threads are where the further threads in the algorithm obtain transmitted information from and where they write information to depending on the sub-process. While a simple thread, the radio thread-types allow for drastically easier operation of the algorithm as it adds a buffer layer between the radio hardware/transmissions and the body of the algorithm. This means no waits/delays are required as in the case that artificial delays are put in place to create synchronized radio transmissions. Again, for more information about the radio system design, see Chapter 4.

The second intrinsic part of the algorithm that is not a main thread is the method used for the lock (i.e. the Python GIL). While there are other options to use the lock in synchronized systems, this was developed as a robust method of sharing global variables. In Figure A.31, the lock system that was described in code earlier is shown in diagram form. This is not a thread at all, but is rather a short code sequence used throughout both of the main algorithm parts. For more information about the lock and GIL, see Chapter 3: Threading Library.

From this point, the main algorithm parts will be described starting with the ground station. It is obvious from Figure A.32 that there are many manual inputs required for the initial operation of the algorithm, but oftentimes they do not need to be

changed from one instance to the next. In other words, if a simulation is run once, it is unlikely many of the inputs need to be altered for a second simulation. The bold objects in the diagram show the primary system process for the ground station main code. Note again that the radios are kept as separate from the main algorithm as possible to preserve the algorithm's sequence without hindering the asynchronous layout of the overall system.

From the summary, it can be seen that once the preparations and initializations are performed for the system, there are numerous threads used for the primary algorithm operations. These are described in the following figures and explanations. Threads "1_#" perform the tasks of sending waypoint information to the lead aircraft and Threads "6_#" perform analysis of the incoming data from the lead aircraft (and the following aircraft(s) In the case of Thread 6_3f). The value after the underscore usually tells the user for which tier the thread is intended. For example, Thread 1_1 is used for sending control commands to the lead aircraft during Tier 1. The exception for this is any follower aircraft threads. Thread 2 obtains the ping signal given off the lead aircraft and concatenates the information into global variables while Thread 4 does the same for the following aircraft(s).

The first thread from the ground station to be discussed is Thread 1_1 which sends waypoint commands to the lead aircraft during Tier 1 operations. The diagram in Figure A.33 shows the process for the thread. First, the waypoints are generated or imported prior to the idle loop being started. While not completely necessary depending on design, using an idle loop allows each thread to begin at the same time and avoid startup time later in the process of the algorithm. The thread then concatenates the waypoint command into minimal-length transmission form and sends the command to the text file. Until the aircraft has arrived at the waypoint (within 1 meter), the ground station continues to send the waypoint command to ensure that the aircraft is able to continue towards the waypoint no matter if there is temporary data integrity loss. Once the aircraft has arrived at the waypoint, the next waypoint is used for the new command sequence.

The next thread is Thread 1_2 which sends the waypoint for Tier 2 to the lead aircraft. As seen in Figure A.34, the majority of the process is the same as in Thread 1_1, but rather than simply generating waypoints based upon the largest search area, the hot spot (i.e. Tier 1 refined target location) is obtained from global variable from found in other threads and used for the start Tier 2 waypoint generation location. The Tier 2 waypoints are generated with the hot spot in the middle of the rectangular sequence.

The last command send thread from the ground station is in Thread 1_3. While this thread is far shorter than the other two command send threads, it is still needed for the overall action of the algorithm. Using the same process as in Thread 1_2, the hot spots are obtained from the various aircraft search refinement sub-algorithms and then averaged. Note that a weighted average could easily be

incorporated if the detector efficiencies were known or other characteristics were desired to be incorporated, but a simple average is used in PADUA instead since none of those assumptions can be made. In Figure A.35, it can be seen that only a single waypoint is sent to the lead aircraft; this waypoint is the hot spot developed from Tier 2.

After the command threads (Threads 1_#), the next thread receives the ping information from the lead aircraft. This can be seen in Figure A.36 where Thread 2 is shown. The data received from the lead aircraft's ping signal should yield a string that is a comma-separated list of six or eight values. Thus, if the ping signal is not six or eight values, the data integrity has been lost. If the signal is length six, then the tier is either Tier 1 or Tier 2. If the signal has an additional two values, then the tier in use is Tier 3 since the target latitude and longitude developed onboard the aircraft must be included.

While Thread 4 (shown in Figure A.37) is very similar to Thread 2, the primary difference is that there is no need for receiving a ping signal with a target estimate update since Tier 3 search is only performed by the lead aircraft. This makes the thread a bit simpler, but the follower ping signals received must be appended to the appropriate aircraft location/response history. While this process could be performed in Thread 2, splitting this process into a separate thread allows for the option to either have single or multi-aircraft algorithm usage as well as allowing for more rapid radio reads since the ping signals from the lead aircraft are transmitted over a different frequency than those from the follower aircraft(s).

The next threads are those that apply the appropriate search algorithms to the incoming data; Thread 6_1 shown in Figure A.38 is the thread that handles Tier 1 search. While the simplest of the three Tiers, Thread 6_1 still involves optimized real-time analysis methods. Note that the Savitzky-Golay filtering method is performed through class codes (along with many other portions of the algorithm) as a much more complex process than what is shown in the figure. In addition to the search sequence itself, in Tier 1, the method used determines the hot spot via the maximum value found from the Savitzky-Golay filtering. This differs from the weighted average method used in Tier 2 where each sample location is weighted by using each respective Savitzky-Golay-generated value as its weighting factor. The hot spot is continually updated and stored as a global variable so that it can be accessed by command threads via the GIL.

The next thread (Thread 6_2 shown in Figure A.39) becomes far more complicated than Thread 6_1. This thread is used for analyzing the lead aircraft's data for Tier 2; as such, it performs the same calculations as Thread 6_1 but also with Bayesian statistical implementation. This causes two separate waypoints to be generated that can later be combined in the command threads (one from the Savitzky-Golay method and one from the Bayesian process). This diagram shows the overall process of the thread, but there is a large amount of code that is not shown.

The last and most complex thread to be discussed from the ground station is the search thread for the follower aircraft(s) (Figure A.40). This thread collects the information from the follower aircraft ping receipt thread (Thread 4) and performs the appropriate analysis for either Tier 1 or 2 as stated by the global variable. In addition to the type of processing performed in Threads 6_1 and 6_2, this thread must also initialize multiple instances of analysis classes for *n*-number of follower aircraft and place the results into the correct logs. This requires a number of techniques that are not seen in other parts of the algorithm such as “exec” commands and dictionaries for class name storage.

The next set of flow diagrams are those that describe the operating process of the lead aircraft; for the lead aircraft, there are three primary threads with the first being the most complex (Thread 1 as seen in Figure A.41). Thread 1 performs all main control aspects for the aircraft for all three tiers of search (although some of the control responsibilities in Tier 3 are shared with Thread 3). Tiers 1 and 2 require nearly identical processes in Thread 1 where the commands from the ground station are simply translated into commands readable for the flight controller. The only main difference is that the speed and altitude used for Tier 2 commands are different from those in Tier 1. For Tier 3 however, the command generation process is entirely internal which allows for fully autonomous flight once the Tier 2-generated hot spot waypoint is reached. In Thread 1 for a Tier 3 search instance, the direction choice decision is obtained via global variable from Thread 3. This variable has four different form options: ‘( )’, ‘-1’, ‘0’, or ‘1’. If the variable has a value of ‘( )’, then Thread 1 will reassign the value to a pass value (‘-1’). If the value is determined to be a pass value (‘-1’) or a continue value (‘0’) then the command that the aircraft sends to itself is repeated and unchanged from previous (a 10 meter directional movement in either north, south, east, or west directions). If the variable has a change command though (‘1’), the aircraft will change the command to move the direction of flight 90 degrees clockwise. In Thread 1, the latitudes and longitudes are logged internally when the groundspeed is above 1m/s. By averaging these values and then calculating the moving average of those values, an estimated target location can be determined.

Thread 2 (Figure A.42) performs the ping send sequence. For Tiers 1 and 2, the ping signal is sent with the location and detector response for the aircraft while for Tier 3 the target location is also sent. This is critical to the operation of the entire system since it provides the only link between the aircraft and the ground station.

The last thread in the aircraft sequence is Thread 3 (Figure A.43). This thread serves two main functions. The first is to act as an artificial detector response generator and the second is to perform analysis for Tier 3 turn determination. The artificial response must be running all the time during simulation and uses the lambda function method to generate the source response knowing the source location (inputted into the generator at the beginning of the simulation) and the aircraft current location (stored and updated via global variables internally). In

addition to the source generator, if the current search tier is Tier 3, then the thread follows a set of requirements to determine if the current response can be appended to the search array. If it can, then once reaching a certain number of values, the thread determines if there is a rise or fall in count rates per the RAT method. The change choice determined is then relayed to Thread 1 via global variable form as earlier mentioned.

While the follower aircraft operates slightly differently than the lead aircraft, the only major difference is that there is no Tier 3 search and the incoming radio signal is the ping signal from the lead aircraft rather than commands from the ground station. All commands are internal for the follower aircraft and the determined swarming method is integrated into the command generation based on the input of the lead aircraft's last relayed position.

CHAPTER 4

RADIO SYSTEM DEVELOPMENT AND TESTING

This section goes into deeper descriptions of the reasoning behind and the operation of the radio systems used in PADUA. Although the methodology used is not entirely conventional is not necessarily the fastest method, it is very robust and dependable. Furthermore, it is able to handle numerous different agents for swarming capabilities.

Summary of Wireless Communications Systems

Two key aspects laid the groundwork for the successful development of PADUA. The first, as discussed earlier, was that of mixed intelligence. By giving the aircraft(s) a limited and variable amount of autonomy, the second aspect was able to be implemented too—communication data minimization.

While the primary wireless communications hardware used for the experiments in this project were Xbee radios, the algorithm is written in such a way that with the creation of an additional class, many different types of communication hardware should be able to be easily implemented including cellular, satellite, and other RF communication devices. Even Bluetooth was tested with PADUA.

Because the algorithm is designed to operate with numerous vehicles in the future, the radio transmissions are designed to use a burst method rather than a continuous link. In other words, instead of maintaining a continuous transmission between the ground station and every vehicle its controlling, the ground station sends out a radio burst with the commands for a specific aircraft before moving on to the next aircraft (if the situation calls for the ground station to control multiple aircraft). This allows the ground station to iterate through numerous vehicles without having to maintain links with all of them. Instead, by having all the systems on the same frequency and channel, all vehicles will receive the command given to any system, but only the system with the identifying vehicle number embedded in the command string will use the command itself. The transmissions from the aircraft(s) operate using the same methodology. Thus, any transmission will be received by all systems on the same frequency and channel limited only the distance from the transmitter. This can be seen in Figure A.44 where an example of signal range and reception is displayed. In the figure, the ground station transmits a signal that both aircraft receive since they are within range while each of the aircraft also transmits a signal that is captured by the opposite aircraft as well as the ground station (assuming they are on the same frequency and channel). Additionally, by using this type of burst signal method, a separate ad-hoc system could be designed to relay radio signals further to increase the range of a system.

Simulated Radio System

The first type of radio system that was used for this project used an extremely simple simulated radio system that was later modified slightly to allow the integration of actual radios with minimal changes to the overall system methodology. In order to run a realistic simulation on a single computer without any radios, the ground station and aircraft sides of the algorithm needed to be able to operate completely independently and communicate with each other through an intermediary.

To replicate the radios, a set of two text files were created that represent the two radios that would normally be used with the system. The first text file was named 'radio_1_replicator.txt' while the second was named 'radio_2_replicator.txt.' These two files can be read from and written to by either of the parts of the algorithm so long as they both run from the same folder that also contains the radio replicator files. This allowed early systems to be run with the ground station and a single aircraft to be run independently on a single computer while still maintaining the semblance of radio communications via the text files. For the purposes replicating the lag in real-world radio connections, the Python codes included sleep functions with the replicator instances. These were left in when the actual radios were introduced to the system though since it was found that the serial commands operated better with some lag between read/write functions and continuing with the code.

This type of process was used for later iterations of the code (including the final product) but with radios incorporated as HITL. This meant that the text files had to be duplicated on each platform and rewritten via the radios. This process will be described in the following subsections.

Single Radio Two-Way Half-Duplex System and Handling Methodology

For a system with few aircraft, a single radio method can be used as can be seen in Figure A.45. This means that the ground station and aircraft each only have one radio attached that handle both transmitted and received signals. This is set up to operate as a two-way radio system in a forced half-duplex communications method; a half-duplex method means the transmissions can only be performed in one direction at a time. While this is slower than a full-duplex method (i.e. a system where the radios send and receive signals simultaneously), using this type of method is more robust than a full-duplex method and allows for many more types of communications hardware to be implemented with minimal changes to the code. In such an operation, the ground station radio would send a signal *to* the aircraft and the wait for a signal *from* the aircraft afterwards before sending another signal. This is a robust but cumbersome technique. A modified version is instead implemented for PADUA's operations.

If the method shown in Figure A.45 is broken into two separate parts and expanded, essentially a two-step process is formed. Figure A.46 shows the two steps of a two-way half-duplex system. The top half of the figure shows the transmission of a signal from the ground station to the aircraft while the bottom half of the figure displays the second step where the transmission is from the aircraft to the ground station using the same set of radios. It was attempted to apply the single radio system to the simulations in this project, but due to the low bit rate characteristic of the XBee radios, a double radio one-way semi-full-duplex system had to be used instead.

Figure A.47 shows the sequence diagram for how the single-radio handling operated in line with an algorithm. First, the ground station side of the algorithm relayed all control, command, and position data between the threads running in the algorithm and the ‘radio_#_replicator.txt’ files. The file ‘radio_1_replicator.txt’ was used for sending commands from the ground station to vehicles while the ‘radio_2_replicator.txt’ was primarily for receiving information and data from the vehicles. Using the files in this manner allowed the radio to read/write data from/to the txt files with minimal interruption to the progression of the algorithm itself. (As mentioned, this replicator text file format is still used in the final versions of PADUA.) Furthermore, by allowing this information to be stored in the form of text files rather than variables, no locks are required and, if desired, other types of programs can easily read/write from the radio replicator files as well.

Most importantly though, using the text files allows for seamless change from fully simulated experiments (instances where the aircraft exists only as an SITL) to fully realistic experiments, tests, or usage. To understand this better, some examples are given below. Note that there are always copies of each radio replicator txt files in whichever main folder(s) is used. If there are not any existing in the folder to begin with then the code will create them as needed. With the text files only (i.e. no wireless communications hardware), the algorithm can be run in its entirety in simulation mode as long as both the ground station and aircraft codes are all run from a single main folder. Conversely, if two main folders are used either on one computer or two, then the text files are still used in the same manner, but all updates will occur via the wireless hardware between the two sets of radio text files.

Example 1: The algorithm is being run on a single computer. The files 'aircraft_main.py', 'Onboard_Aircraft_Codes.py', 'Ground_Station_Main.py', and 'Ground_Station_Codes.py' are all in a single file along with the associated text files including 'sim_speed.txt' and the radio files 'radio_1_replicator.txt' and 'radio_2_replicator.txt'. There is no wireless hardware used in this example. The two halves of the algorithm (the ground station and the aircraft) will read and write from/to the radio files. In this way, the radio text files serve as radio replicators.

Example 2: The algorithm is being run on a single computer. The files 'aircraft_main.py' and 'Onboard_Aircraft_Codes.py' are in a folder labeled 'Main_Folder_A' while the files 'Ground_Station_Main.py' and 'Ground_Station_Codes.py' are in a folder labeled 'Main_Folder_B'. Both main folders have copies of all the text files within them such that 'Main_Folder_A' contains 'sim_speed.txt' and the radio files 'radio_1_replicator.txt' and 'radio_2_replicator.txt,' and 'Main_Folder_B' *also* contains 'sim_speed.txt', and the radio files 'radio_1_replicator.txt' and 'radio_2_replicator.txt'. Additionally, there are two radios connected to the computer via USB cables. The aircraft side of the algorithm is run from its respective main folder while the ground station side is run from its main folder. The full algorithm (both the aircraft and ground station) will proceed in command generation, interpreting, reading, etc... by reading/writing from/to the radio text files located within their respective main folders. Additionally, the radio text files are updated between the two folders via the radios (since updating a text file in one folder will not update it in another). This allows changes made in one radio text file to be communicated to the other side of the algorithm.

Example 3: The algorithm is being run on two computers. The files 'aircraft_main.py' and 'Onboard_Aircraft_Codes.py' are in a folder labeled 'Main_Folder_A' on Computer A while the files 'Ground_Station_Main.py' and 'Ground_Sttion_Codes.py' are in a folder labeled 'Main_Folder_B' on Computer B. Both main folders have copies of all the text files within them such that 'Main_Folder_A' contains 'sim_speed.txt' and the radio files 'radio_1_replicator.txt' and 'radio_2_replicator.txt,' and 'Main_Folder_B' *also* contains 'sim_speed.txt', and the radio files 'radio_1_replicator.txt' and 'radio_2_replicator.txt.' Additionally, there are two radios connected to the computer via USB cables. The communication between the two sides of the algorithm proceeds exactly the same as in Example 2 but with a wireless radio connected to each computer rather than two radios connected to a single computer.

Double Radio One-Way Semi-Full-Duplex System

The algorithm has been written to work with either a single radio two-way half-duplex system (as just shown in the previous subsections) or as a double radio one-way full-duplex system. Referring to the latter version as a “full-duplex” system is a loose definition but is accurate in reference to the overall operation since signals can be sent in both directions simultaneously; however, instead of a conventional twin radio full-duplex system, the method used in PADUA utilizes half-duplex hardware operating in a full-duplex manner. Furthermore, because of the latency issues already discussed involving the XBee radios, the double radio system described here is the one that was utilized for all later experiments and for the final PADUA algorithm structure.

The double radio one-way semi-full-duplex system operates according to the following explanation. As an example, if there is a single aircraft, the aircraft will have two radios attached while the ground station also has two radios. On each platform, one of the radios is devoted to sending information while the second receives only. Additionally, by assigning a specific preamble signifier to each pair of receiving/transmitting radios, no interference should occur between the two pairs. By running each of the radios from an individual thread, what amounts to a full-duplex system can be created at the expense of a slightly higher load on the computer operating the algorithm and two additional radios. Figure A.48 shows the setup and operation of the double radio two-way semi-full-duplex system described. Because each radio is devoted to either sending or receiving signals and each radio set (Radio Set 1 and Radio Set 2) communicates with a specific assigned preamble value, there can be overlap between the sending and receiving of signals without data interference occurring. Such a system as the one shown in the figure would have four radios in all for a single aircraft system where the ground station has radios 1 and 2 and the aircraft also has a pair of radios 1 and 2. Using two threads as shown in Chapter 3: Final Algorithm Thread Design, asynchronous data can be communicated using this method.

CHAPTER 5

SINGLE AIRCRAFT EXPERIMENTS

Summary of Experimental Setup

Most of the experiments in this chapter use very similar setups with the exception of the transition experiments that introduce radio hardware rather than only using replicated radios. The primary hardware used for the experiments was the Panasonic Toughbook and, in later instances, the XBee radios connected via USB/microUSB and communicated with via serial protocol. The primary language used for all the codes in this project was, as mentioned, Python 2.7.

The experiments are ordered by search area size. The first experiment is a test of Tier 1 followed by Tiers 2 and 3 before finally combining all of them in a separate experiment. This allowed for the simplest format of code development since the final algorithm needs to have larger size

Tier 1 Experiments

Summary and Setup

The primary goal of the first tier experiments was to set up the framework of the overall system for large area operations. This would consist of developing both the ground station and aircraft sides of the full system algorithm as well as a method for single aircraft communication. The system needed to be able to automatically generate a waypoint-to-waypoint path for the aircraft of any rectangular size given an input. The only detection that needed to be performed for this tier consisted of a generalized search process in an unbounded region. One of the important goals in this experiment was to determine if there was an accurate numerical method of locating multiple radiation/source hot spots.

The code was run using Python 2.7 and replicated radios as described in Chapter 4 without hardware.

Methodology

The Tier 1 experiment was where the final framework design of the overall system was developed; therefore, much of the discussion in this section will be describing the design of not only the Tier 1 experiment but also of the entire final system. Although described in Chapter 3: Overall Algorithm Layout section, the basic design of the system will be reiterated here.

The algorithm was developed as a class-structured code with a class library for each half of the algorithm. This allowed the library to be continuously updated without necessitating changes in the primary main codes of the algorithm throughout the experimental sequence other than their own development. For example, the class library for the ground station side of the algorithm in the tier 1 experiment has the classes for communications, encoding transmittable communications, running radios, Savitzky-Golay- based search, and many others. When the Tier 2 experiments are performed, the same library is used, but further additions are made such as classes for the governing processing of Bayesian statistics. This allows the code itself to be extremely easy to update or change later which—as mentioned previously—is critical in the design of the final algorithm since it needs to be as detector agnostic as possible. Figure A.49 shows the principle architecture of the full algorithm. Note that the main codes communicate with each other while the class libraries only communicate with their respective main codes.

In this experiment, there were a few key design points of importance. These included the aircraft startup sequence, the predetermined path generation method, the source hot spot determination method, and intricacies of the algorithm.

In order to create an algorithm that can be easily applicable to the real world, the first set of commands to an aircraft was a ‘handshake’ system where the ground station would proceed by establishing a communications link with the aircraft in question before arming and finally sending takeoff instructions. To do this, a sequence was developed that ensures a minimal chance of miscommunication or lost signals between the aircraft and the ground station. The sequence can be seen in Figure A.50. *Note that although this ‘handshake’ process is used for all single aircraft systems and could be implemented if desired for multi-aircraft systems, it was removed from the final algorithm for multi-aircraft experiments in favor of faster testing.*

By implementing this type of reciprocal process, it was ensured that there were no problems with the radio communications on either the ground station or the aircraft prior to the aircraft actually lifting off the ground. This assists with early-flight crash-prevention. Ensuring a solid radio connection early in a flight will be critical for real world testing, as small UAS tend to be most unstable during low-level flight as it is due to ground effect aerodynamics.

The second important feature in the first experiment is the methodology the code uses to develop a predetermined path. Utilizing the class structure discussed already, seemingly endless path designs can be hardcoded into the class library later to allow for different search methods that either follow a simple pattern (such as the one used in this experiment) or could possibly follow a very one that might follow the streets in a city. As for this experiment, a simple rectangular-based back-

and-forth sweeping pattern is used with five inputs: length, width, spacing, number of waypoints per leg, and the lower left corner start location (in latitude and longitude). Figure A.51 shows an example of the type of path that is produced for Tier 1 experiments with the descriptions of the different input variables. The length, width, and spacing are all defined in units of meters.

Another aspect that was important in both Tier 1 and Tier 2 experiments was the method developed for analyzing raw count data. To perform large-scale analysis on raw count rate data, a few different problems needed to be overcome. The first was the issue of possible outliers. If there are one or two detection samples that yield extremely high-count rate values due to either detector error or abnormal temporary background rates, then the analysis system needed to be able to dissect the correct values from the total data. Secondly, the analysis system needed to be able to operate very quickly; although simulations are being used for these experiments and the SWARM project are using multirotor aircraft, in the real world, the PADUA and SWARM algorithms (specifically the Tier 1 portion) need to be able to operate with a slow fixed wing aircraft or a helicopter in addition to multirotor ability. This means that higher speeds could likely be achieved than what multirotor surveillance aircraft might be able to achieve, and thus the computational speed needs to be exceptionally rapid as well. Lastly, the base capability of detecting multiple sources was desired for this project as well. This means that the analysis method needs to be able to not only determine hot spot locations, but it also needs to be able to determine the locations of secondary, tertiary, etc. hot spots.

The main analysis method used for this project when encountering raw count rate data is the application of a Savitzky-Golay filter in Tier 1 and partially in Tier 2. The main concept in deciding on using this type of filtering technique revolved around two issues. First, the best way to analyze the data appeared to be by using a curve-fitting technique to cancel out some of the noise. Secondly though, the curve fitting needed to be able to be performed in near-real time and could not rely on a specific curve shape (i.e. natural log, quadratic, etc…) since there should be no significant frequency expected in the resulting data.

Savitzky-Golay (S-G) filters are fundamental smoothing methods resulting in a low pass filter that is variable and dependent upon the order and sample length used. S-G filtering starts with a least-squares approximation. The polynomial in a least-squares approximation can be described by Eq. 5.1 where the polynomial is found of order *k* meaning that for order 0, the polynomial is equal to the coefficient *a*[101].

$$p(n) = \sum_{k=0}^{N} a_k n^k \qquad \textbf{Eq. 5.1}$$

The second part of the basis for Savitzky-Golay filtering is convolution. Convolution is described best when considering a moving average. In a moving average, a set

length of data is considered whereby the average is determined to generate a representative point. To generate the next point, the same process is performed but with the addition of the previous data point in the sequence and the subtraction of the earliest point in the previous sequence to the array being averaged. This allows the most recent updates to be considered for each new data point while removing earlier data influences that do not apply to later data. Applying convolution, the Savitzky-Golay method can begin with Eq. 5.2—a generalized formula for determining single smoothed values Y_j^* where essentially a portion of a moving average is described. The original data values are designated as Y_{j+i}, and the summation is taken from the midpoint of the set array length between $-m$ and m. For each new value in the Y array, m will equal $m_{i-1}+1$[102].

$$Y_j^* = \frac{\sum_{i=-m}^{i=m} C_i Y_{j+i}}{N} \qquad \textbf{Eq. 5.2}$$

Applying the S-G method to convolution, the coefficients C_i must be determined in a slightly altered version of the convolution equation (Eq. 5.3). To calculate the coefficients, first a least squares method (Eq. 5.2) is applied to the points p_j from $-m$ to $m+1$ in the original dataset, and the value of the polynomial $p(n)$ is set as g_j at location j. From here, the coefficients can be found. This means that a polynomial that is similar or lower than the value g_j will be given a higher weight whereas a polynomial value at location i with a higher value than g_j will still be included in the final formula but with a fractional weighting [103].

$$g_j = \sum_{i=-m}^{i=m} C_i p_{j+i} \qquad \textbf{Eq. 5.3}$$

The most difficult part of the process in S-G filtering is choosing appropriate input values. While the S-G filter is used in numerous applications, applying the filter to the raw count data in this project is somewhat different from most other instances of use. This is for several reasons. First, it should be remembered that the primary reasoning for using the filtering technique for the raw count data here is that the original signal is exceptionally noisy and might have outliers within itself. Secondly, to locate the physical location of a signal hot spot, a simple maximum cannot be utilized because—as just stated—there can be an exceptional amount of noise in the raw data. Instead, it is better to observe the integral of the count rate curve and find the area that does not necessarily contain the global max but instead has the largest concentration of local/secondary max values (although this location may also contain the global max). If a refined signal is generated that is smooth and contains dampened high-value outliers rather than trying to just determine which values are outliers and which are not, the process to determine the hot spots becomes far easier.

Take Figure A.52 where the raw count rate data from a search can be seen as a red line and a blue line is used to represent the refined S-G filtered curve. The horizontal axis has no physical meaning other than it is a reference for the signal value at a specific point; that is, the iteration of detection progresses from left to right. The vertical axis is the raw count rate detected. The raw data has a global maximum at about the 580^{th} data point. However, because the area is less under the raw data curve than the point at ~480, the S-G filter places a global max at ~480 instead. Note that the flight path is a back and forth pattern (as seen in Figure A.51) with a source near the middle of the pattern, which is why there appears to be stable frequency in the signal. If the pattern was changed or there were multiple sources in the search area though, there would not necessarily be any useful frequency.

Once the S-G values are developed, the analysis code in the Tier 1 single aircraft experiment can locate the global maximum and index its iterative location. Using the index, the corresponding raw count value can be looked up whereby the geographical location can be logged as the location of the first hot spot. This result is intended to yield only the first hot spot with a generalized location of the source. Tier 2 is intended to refine the location further. The hot spot location found in Tier 1 simply needs to be a good estimation of where to progress to the next tier of search.

Another advantage of implementing the S-G filter is that of multi-source search capabilities. Although only a single source is used in this project, for future use the ability to locate multiple sources will an extremely important factor that was designed into the algorithm. To perform multi-source searches, once the first hot spot is located, the values of the S-G filtered data on either side of the hot spot index location are changed in the algorithm to values of zero until the differential changes sign. The new data is represented by the green curve in Figure A.53 after the first hot spot is located. From here, the process can be repeated to locate and log the geographical locations of as many hot spots as desired in decreasing likelihood.

An additional intricacy of the code that was added in for simulation purposes in the Tier 1 experiment includes the ability to speed up the simulation artificially. This is done by dividing any sleep statement time values (“time.sleep(#)” in Python syntax) that are inserted for latency replication with radios and the detector by a time speeding factor. For most simulations that use the time division factor, a value of three is used for single aircraft experiments. The simulation speed can be altered by changing a text file in the main folder that the algorithm reads at initiation of the code. While this factor does not change the overall result at all (other than the timestamp data), it does allow for more data to be obtained quicker than would be otherwise. For final experiments, no simulation speed increase is used so as to completely replicate a real-world situation as closely as possible.

Results and Discussion

The first result that became apparent in Tier 1 testing was the difference in the actual flight path and the planned flight path. As mentioned already, the code generates a back-and-forth flight pattern for the aircraft to fly with a set number of waypoints in the y-direction (i.e. the latitude direction). Figure A.54 shows an example of part of a flight path (black) overlaid onto the planned flight path (red). The red points along the path are the generated waypoints while the red line represents the path that the aircraft should take (starting from the bottom left corner). This is an excerpt from a 675m x 900m flight path. However, using waypoint-to-waypoint command methods, it was discovered that drift sometimes occurs when the aircraft approaches a waypoint and begins to slow down (this is an automatic response from the Dronekit library/API). Furthermore, while the drift is not significant, the change in speed is an important aspect of this experiment since Tier 1 is supposed to be the largest area search and thus needs to be flown at higher speeds than the other tiers; if the aircraft slows when approaching each waypoint, then the average speed will be far lower than desired. Thus, the least number of waypoints possible are used in the final algorithm for Tier 1.

Usage of fewer waypoints can be seen in Figure A.55 where a far smoother flight path results when only the minimum number of waypoints is utilized. Figure A.55 is an excerpt from a 900m x 500m area coverage flight path.

The next part of the experiment is the refinement of the S-G filter search method. To develop a curve used for estimating the optimum window size for searching, a script was written that uses the same path generation method as discussed above. A “window size” is defined for this project as the number of data points used for each stage of the S-G filtering. Several different paths were generated with different longitudinal spacing distances. For each of the tests, rates were generated using the artificial detector method discussed in Chapter 3 with a source located in a randomly generated location within the bounds of 20%-80% of the search region bounds as shown in Figure A.56.

To develop an adequate set of data, numerous tests were performed with each using the following process. First, a range of window sizes from 5 through 201 were used for the initial test procedure in odd increments. The path used for a particular test was the same each time, but with a different randomly generated location for the source. Each path used an area 500m x 500m (North-South x East-West) with 51 points in the North-South legs and a set separation distance for the E-W directions. Figure A.57 shows three of the many types of tests used. By using differently spaced flight paths, more locations for the generated source can be tested and therefore more detection distances between the flight path and the source can be tested. Figure A.58 shows an example of the source test

locations for a 250-meter separation path. In the test code, rather than using a real-time simulation which would require an immense amount of time, rates were generated at each of the generated path waypoints (thus the explanation for why the North-South separation distance is only 5m between waypoints).

Once count rates are generated for each of the detection points for a single path and single source location (i.e. a single test), an S-G filter is applied to the raw data using two primary variables. First, the order used for the filter was third degree with the reasoning being that a radiation flux over distance curve with background radiation takes on a form similar to a symmetric Boltzmann or Sigmoid curve which is approximately a third degree polynomial as well for one side of the curve (Figure A.59). The second variable for the S-G filter input is the window size. For a single test, window sizes of 0 through 201 were used with the hot spot location being considered the maximum value from the S-G filtered results. The distance from the hot spot location to the source location was the calculated and stored as well. Once all the window sizes were tested, the hot spot location with the minimum distance to the source was considered the best result, so the respective distance from the source and window size were stored.

Once all the tests were run for all the different paths utilized, a large quantity of data was available where boundary conditions could be implemented. Four boundary conditions were used. The first was that, if the distance from the hot spot to the source location was 0, then the data point was removed. The second was that, if the percentage difference of the distance from the hot spot to the source location was more than 10% different than the minimum distance from the detection locations to the source, then the data point was also removed since the hot spot was considered to yield a false positive. The third condition was that if the distance was more than 110 meters, the data point was removed since the response of the source is approximately equal to background at 100 meters (using 110 meters gives a 10% buffer zone). Lastly, because the real detector uses a buffer rather than just time to produce a count rate result, a limit of 140 was placed on the window size since it is unlikely that a significantly larger sample set would be obtained using the experiment detector with PADUA. This is an arbitrary number that could be altered based on the needs of a mission including the characteristics of the detector, the speed of the aircraft, and the size of the search region.

With these conditions implemented, the remaining sample set contained 9586 data points. The last part of the process was to average the distances from the source at each window size (since there were multiple data points for each window size). This then gave a single data point for each window size tested. As might be expected, a negative Sigmoid-like curve resulted whereby it was found that the best curve fit technique was a third order Gaussian. This data is all displayed in Figure A.60. In the figure, it can be seen that a third order polynomial is also a

usable approximation, and with more data a third order polynomial best-fit curve might be able to be used in the future. However, while the third order polynomial best fit curve is simple and makes sense for the data, given the boundaries imposed onto the data as previously discussed, it was found that the third order Gaussian best fit curve was a better match for the data. The third order polynomial best-fit curve shown in the figure can be seen in Eq. 5.4. Note that the third order approximation was made with the window size on the *y*-axis and the distance on the *x*-axis as it follows the shape of a third order polynomial far better. This is the equivalent of using *x=f(y)* rather than *y=f(x)*. As can be seen in the figure though, while the Gaussian approximation is decent, as the window size grows, there is a wider and wider spread of the distance results. Thus, while Gaussian approximation (Eq. 5.5) can be used as a good estimation for what size window to use if the maximum possible distance from the source is known, for this project, there is no reason to ever use a window size less than ~70 for maximum distances less than 80 meters. To explain these results more, when the distance is far from the source, there is not only a smaller response from the detector, but it also occurs for a shorter period of time since the aircraft is flying through a spherical range of source influence. If the aircraft is closer to the source, it will have to go through a larger area that is influenced by the source whereas if the aircraft is further from the source, more of the detected data will simply be predominated by background radiation.

$$y = a * x^3 + b * x^2 + c * x + d \qquad \textbf{Eq. 5.4}$$

$$where\ a = -.001392, b = .2992, c = -22.28, d = 655.2$$
$$for\ 0 < x < 110\ and\ 5 \leq y < 140$$
$$where\ y == window\ size$$
$$x == maximum\ path\ distance\ from\ source$$

$$y = a1 * exp\left(-\left(\frac{x - b1}{c1}\right)^2\right) + a2 * exp\left(-\left(\frac{x - b2}{c2}\right)^2\right) + a3 * exp\left(-\left(\frac{x - b3}{c3}\right)^2\right) \qquad \textbf{Eq. 5.5}$$

$$where\ a1 = 15.5, b1 = 64.77, c1 = 12.39, a2 = 47.2, b2 = 31.19,$$
$$c2 = 41.27, a3 = 3.32e12, b3 = -1.33e5, c3 = 2.67e4$$
$$for\ 0 < y < 110\ and\ 5 \leq x < 140$$
$$where\ y == maximum\ path\ distance\ from\ source$$
$$x == window\ size$$

Although this is a novel method for determining the window size for S-G filtering (because the usage of S-G filtering in this project is itself novel), for any distance less than 80 meters, the window size does not matter as much so long as it is over the previously stated 70. However, it should be noted too that if the window size

becomes too large for this scenario, the results become far noisier and therefore less trustworthy as shown by one test's standard deviation curve in Figure A.61. With all this information known, it can also be noted that the width of a Tier 2 search area should be the same approximately as the maximum leg separation distance. For example, for a Tier 1 search maximum separation of 100m, the Tier 2 search area should be at least 100m x 100m.

Having discussed the methodology behind the primary parts of the Tier 1 search system, a few different factors can be seen to be very important. First, the search pattern is extremely important since it determines error sizes, window sizes, and further tier search area sizes. This is an important aspect to keep in mind if more complicated search patterns are utilized since the maximum separation should be known to obtain the best results. Again though, if shielding is introduced, then the error methods shown here will not necessarily be accurate. Rather, the methods shown here operate under the stance of best-scenario since the worst scenario is simply that the source is completely shielded in which case there is no way to locate the source without neutrino identification which is not practical today. However, the results displayed here also show that by applying some basic filtering methods, very noisy count rate signals can be refined into usable data with ease. Furthermore, because basic methods are used that take into account only a simple count rate, this methodology should be able to be applied to any non-directional source with minimal or no changes.

Tier 2 Experiments

Summary and Setup

The Tier 2 single aircraft experiments had the main aspiration of determining an efficient method of utilizing Bayesian statistics for hot spot verification and resolution. Combination methods for determining the source's location using both Bayesian statistics and raw count data filtered via Savitzky-Golay were utilized in the end result. Although multi-aircraft operations will later be discussed including the methods used for mixed intelligence, these experiments use only a single aircraft controlled by the ground station.

The code was run using Python 2.7 and replicated radios as described in Chapter 4 with no radio HITL.

Methodology

The primary goal of Tier 2 single aircraft experimentation was to implement Bayesian statistical searching into the algorithm. This was performed by converting a Bayesian code developed in part by Josh Gurka[104] for the SWARM project from Python 3 to Python 2.7. The resulting code that manages the Bayesian search processes was simplified utilizing basic array formatting rather than Pandas data

frames as in Gurka's code due to problems with implementing Numpy natural log commands with the Pandas-stored data in Python 2.7 format. The overall process is the same though and runs the exact same functions in the same order as the test version written in Python 3. This was done so that, when the algorithm developed in the project is later updated to Python 3 (since Python 2.7 which is rapidly becoming archaic), the Bayesian search class will be able to use either an array format or a data frame format as desired with no change to the code other than which class is used. To clarify, the rewritten Bayesian code class is named "Bayesian2" and all the functions within the class are identically named as those in Gurka's version of the code. Since his code is stored in the class library as "Bayesian", the only change that has to be made to the code later (other than Python version format changes) is removing the "2" behind the library name when it is imported.

Because the S-G filtering system operated so well in Tier 1 searching, there was no reason not to attempt to apply the system to Tier 2 search analysis as well. To do this, a system had to be developed to determine where the next source hot spot would be located based on a combination of the Bayesian and S-G filtered raw count rate data. A number of tests were run in sequence in a small area with a source placed in random locations, and a weighted average method was then used to combine the results by observing the differences in the hot spot location determined from each method on its own.

For a single aircraft, the flight control works much the same as it did for Tier 1 operations. The ground station develops a flight path and relays it one waypoint at a time to the aircraft. The aircraft interprets the commands, flies the path, and relays the detector data back to the ground station in real time using the same ping method as in Tier 1 search. This will change slightly when applying the system to multiple aircraft since mixed intelligence will be utilizes as will pattern swarming, but for this experiment, the primary operational change is simply in the search methodology.

Results and Discussion

The first question that had to be answered in the Tier 2 experiments was how to produce the flight path. It was decided that the flight path itself would be a fractal version of the Tier 1 flight path. By scaling the flight path down to have the span in width and height of the north-south leg separation distance in Tier 1 (i.e. the *x*-separation), the Tier 2 flight path is able to cover the entire error possible so long as the source is strong enough in the first place to be detected within half the leg separation span in Tier 1. In other words, there are two possible scenarios. First, if the source type/strength is known, then the Tier 1 search should be able to be designed to have a maximum leg separation of no more than twice the range of detection. This means that for Tier 2, the same flight pattern used in Tier 1 can be

used in Tier 2 but scaled down so that the width and height are each the half span of Tier 1 leg separation. Thus, if Tier 1 had a leg separation of 100 meters, Tier 2 should have a *minimum* corresponding side length of 50 meters. However, the second scenario is an instance where the realized strength of the source is not known. In such a case, the sizing of Tier 2 is arbitrary and should simply be scaled down by a set amount as determined by the user. In such a case, the minimum side length for Tier 2 should still be half of the maximum path separation in Tier 1 for the most accurate results. An example of a Tier 2 flight path can be seen in Figure A.62 in relation to the Tier 1 path that it is based off.

Once the flight control methodology for the single aircraft Tier 2 experimentation was completed, the next step was to refine the search methodology. Because it was used for Tier 1 successfully, S-G filtering was again utilized for Tier 2 in addition to the Bayesian prediction method. However, the intricacies of how the Bayesian model operates first had to be tuned before the combination method could be implemented.

The Bayesian model has already been discussed broadly in previous sections, but the Bayesian methodology used here does have specifics that are important to the implementation of the code for this project. Set up in class form, the Bayesian code used in PADUA runs using a list system rather than a data frame organization method as discussed earlier in this section. The main organization of the Bayesian class can be split into two main parts.

The two parts of the code consist of appending the data into the correct locations followed by running the Bayesian statistical functions over the data at set increments. Figure A.63 illustrates the framework order in which the code is run. While there other variable manipulations and storage instances in the actual algorithm, this figure strips away everything except for the bare Bayesian operations. The Bayesian class has already been initiated as “bys” prior to what is shown in the example. The first half of the code simply appends an array’s values into the proper storage locations within the Bayesian class itself. Note that before the array values are appended to their respective target variable lists, the array is checked for proper format and length; the array is a set of position and rate information from an aircraft and must be in the format “response”, “latitude”, “longitude”, and “altitude” when input into the Bayesian sequence. If any test fails during reading the array, it is disregarded by the code and the iteration value is not progressed. If the array values are appended successfully, then the iteration is increased by one. The iteration/count is used for two parts of the second half of the Bayesian algorithm. The first is that, for ease of coding, no updates are ever possible unless the count is above a set value (in this case, the value ‘2’ is the lower limit as seen in the if-statement on line 3 of Figure A.63). Secondly, if the count is evenly divisible by a value (in the example, the value is 10 in the figure), then the update functions can be run, otherwise the code will simply continue to

add values in the first step. This is important for two reasons. First, if the update is run too often, then the Bayesian distribution will not be a distribution so much as a vertical line at the highest count rate value, which can lead rapidly to false positives. Furthermore, the update functions run far slower than the first half of the Bayesian section; therefore, when a multitude of aircraft are present in a PADUA swarming system, the code is likely to slow down too much for the system to be usable. If the window is too long though between Bayesian updates, then there is little reason to implement Bayesian statistics in the first place. Once the conditions are met, the second half of the Bayesian code is run whereby the values are placed into natural log form, Bayes theory is applied to the data, prior and posterior values are updated from the results in the 'LogLike' function, and then finally, the values are all taken out of logarithmic form.

Natural log form is not necessary for PADUA's Bayesian operations but assists for future updates with large numbers of aircraft due to a lower computation time than without it. It was found through some of the work Mr. Gurka performed that, by placing the normalized count rate values into natural log form, the formula could be run using only simple arithmetic rather than multiplication and division due to the properties of logs[104]. While this is not very important for the simulations in this project, in the future updates for SWARM with *n*-number of aircraft (as per the sponsor's requirements), computation speed will be a necessity.

The first step in the Bayesian update process is to normalize the count rate as seen in (Eq. 5.6) is to normalize each count rate value (y_i) from the first count rate value to the latest value found at whatever point in the search the update is being performed.

$$\{w\}|_{i=0}^{n} = \frac{y_i}{y_{max}} \qquad \textbf{Eq. 5.6}$$

Once the raw count rates have all been normalized by the maximum value in the logged data, they can be assimilated to a logarithmic Gaussian probability density function (PDF) as the values of *x*. To do this, Eq. 5.7 shows the form of the code input used while Eq. 5.8 shows the generalized PDF and Eq. 5.9 shows the logarithmic version. The library used for the logarithmic pdf fitting is "scipy"; to access the PDF tools, the nesting used is the class "norm" within the "scipy.stats" library. The inputs in Eq. 5.7 are the set of normalized log rate values, the maximum normalized log rate value, and the standard deviation of the normalized log rate values. The PDF reasoning and methodology was explained in Chapter 2.

$$\{w_{pdf}\} = norm.logpdf(\{w\}, w_{max}, w_{std}) \qquad \textbf{Eq. 5.7}$$

$$f(x) = \frac{\exp\left(-\frac{x^2}{2}\right)}{\sqrt{2\pi}}$$ **Eq. 5.8**

$$f(x) = \ln\left(\frac{\exp\left(-\frac{x^2}{2}\right)}{\sqrt{2\pi}}\right)$$ **Eq. 5.9**

Before moving further, a very important aspect of the Bayesian methodology needs to be discussed at length: the PDF used for the probability update function. Using the natural log form of a normal PDF should yield the same results, and to prove that this is the case, Figure A.64 was generated using count rates generated at increasing distances from a source. As can be seen, while the PDF methods do not follow the normalized response curve exactly, they act in a very important manner when applying the update method to raw data in the field. That is, when raw data with significant noise is detected, only the higher portions of the data should be marked with higher probability since any lower spikes are difficult to determine whether or not the detected values are due to background or whether the data originates from the source. Since this effectively implements a high pass filter to the probability data, this should decrease the number of false positives. Additionally, because Bayesian statistics update the probability of each point continually based on new information, small spikes of background data can easily corrupt the overall result. Thus, any method of reducing noise and false positives without sacrificing accuracy is an important characteristic.

While other PDF options are available (such as some of the ones seen in Figure A.65), because the shielding, strength, and shape of the source must be considered to be unknown for this project, there is little reason to use anything more sophisticated than a normal Gaussian PDF. As can be seen too in Figure A.65, finding a PDF that assimilates well to a known complex set of data can be difficult. For example, an exponential PDF can only be used to locate a hot spot if the detector is directly above the source. However, in a situation where the source is open and uniform, a normal PDF or perhaps a logistic PDF would be acceptable since it should be assumed that the combination of background and source-originated radiation will yield a curve that can be fit with a polynomial in logarithmic form. Again though, because the situation cannot be known or assumed for this project, a normal distribution will be applied while utilizing the logarithmic version in order to decrease computational time for the algorithm.

Having applied the PDF to generate probabilities for the count rate data at each detected location, the next step is to update the Bayesian statistics themselves. Applying Bayes Theorem (restated in Eq. 5.10), the results from the PDF can be implemented as the update values for *P(B|A)*. Since for this project, *P(B)* can be

considered to be $\cong 1$, Bayes can be simplified to simply show that the new posterior probability value *P(A|B)* for each detection location is equal to the updated probability *P(B|A)* multiplied by the prior probability value previously found *P(A)*.

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)} \qquad \textbf{Eq. 5.10}$$

It is here that the advantage occurs for using logarithms with regards to computational time since simple arithmetic can be used. Rather than using multiplication to update the posterior value, the prior is added to the update probability as seen in Eq. 5.11 in text form and Eq. 5.12 in mathematical form. Alternatively, if the posterior calculation in question is the first of the search sequence, then the update probability found with the PDF can be used as the posterior itself; this will serve as an initial guess for the next update.

$$\begin{gathered} Posterior = Prior + Update \\ or \\ Posterior = Update \end{gathered} \qquad \textbf{Eq. 5.11}$$

$$\begin{cases} i > 1 & p_{1_i} = p_{1_{i-1}} + w_{pdf} \\ i = 1 & p_{1_i} = w_{pdf} \end{cases} \qquad \textbf{Eq. 5.12}$$

Once the posterior has been updated, the last step of an update sequence is to pull the values back out of logarithmic form. This consists of subtracting the maximum posterior from each posterior and then taking the exponential of each value. Because the posterior probability values are still in log form when the maximum is subtracted, this has the same effect as normalizing each of the values by the maximum posterior meaning that the maximum posterior non-log output will always have a value of 1 making it easier to analyze and display the probability data in a meaningful manner.

$$\{p_{final}\}|_{i=0}^{n} = e^{p_{1_i} - p_{1_{max}}} \qquad \textbf{Eq. 5.13}$$

Using this process then, Bayesian probabilities are able to be used to more accurately and—in some cases—more quickly locate a likely source location than simply using raw count data. However, it is important to remember that Bayesian statistics run on an iterative update pattern which means that if there is bad data input into the statistical model (such as higher-than-normal background rates), then a false positive can arise and grow in likelihood according to the method. This is the primary reason that Bayesian search is only used on a smaller search area compared to Tier 1 searches.

There were still many other aspects of the search methodology that needed to be refined once the Bayesian code was written. First, because Bayesian runs on an update method, the update window size needed to be optimized. Secondly, the Bayesian hot spot determination method was able to be improved to significantly reduce the estimation error. Lastly, while Bayesian statistics are not applicable for larger search spaces, that does not mean that S-G filtering of raw data cannot be applicable to smaller search spaces; therefore, a method also had to be determined on how to incorporate the different methods together.

To determine the optimum window size, several tests needed to be run to more easily analyze the change in accuracy depending on the window size. Tests were first run using a nested loop code where window sizes ranged from 2 to 250 with a complete search test run 100 times for each window size; these can respectively be referred to as the window tests and the individual tests. Each combination of a complete run of window tests and individual tests will hereby be referred to as a full test. For each individual test, a source was randomly generated within the bounds of 20%-80% of the search space (in the same manner as shown earlier for Tier 1 in Figure A.66). Multiple full tests were run using different sized complete flight paths in order to alter both the search space size and the number of detection points. Lastly, in order to determine the location of a hot spot from the Bayesian algorithm, once the maximum Bayesian probability was found, the corresponding latitude and longitude with the same index as the probability value was used.

Several results were found, but the first one was an unexpected source of error. To display the data, the points from all the individual tests within a window test were averaged to give a single accuracy point for each window test. This means that there are 100 search tests that go into each average point. During the first test using the intended search space size for tier 2 (100m x 100m with a delta x of 5m and a delta y of 10m), data resulted showing a gradual increase in accuracy followed by an unexpected jump in error from window sizes of ~75 to 120. This can be seen in Figure A.66.

To check the results from the data in Figure A.66, another chart was produced showing the difference between the distance from the hot spot to the source and the minimum distance from the set of detection points to the source. Again though, as seen in Figure A.67, there is still an obvious error around a window size of 120.

To verify if the error only occurs at the specific search area size used, a larger region was tested as well. As can be seen in the raw and average hot spot accuracy data in Figure A.68, there is again error, but this time at a different window size and to a much higher degree. The search region used had dimensions of 500m x 500m with a delta x of 10m and a delta y of 100m. To confirm these results, Figure A.69 shows the differences between the distances from the hot spot

locations to the source location and the minimum possible distance from the path to the source location.

The source of this problem was found to be a combination of the size and repetition of the window size. Figure A.70 shows the 500m x 500m search path with a waypoint x-separation of 10 meters and a waypoint *y*-separation of 100 meters.

However, this means that, with the total waypoints numbering 306 (51 waypoints per leg), if the window size is up to half of the number of waypoints, then the update for the Bayesian process occurs only once and only with the waypoints that occur first. In other words, if there are 306 sample points and the window update size is 160, only one update will occur and only on the first 160 values. Thus, if the source occurs in a location past the 160th sample location, then the hot spot accuracy is extremely limited yielding a high degree of error. Such a path and situation is shown in Figure A.71.

An example of a high error instance is shown in Figure A.72 where the update widow size only allowed for a single update and the source was not located within the searched area. The minimum distance from the path to the source (shown with a red line) would likely be near the hot spot location if there were two updates performed during the entire flight path even if the second window had to be smaller than the first one. As the average source location should be near the middle of the region (since it is randomly generated within a 20%-80% search region box), it will not always be there thus yielding the spike in data. Lastly, it should be pointed out that, because the count rate data being used is realistic with proper noise included in both the source radiation and the background radiation responses, locating the source of error with single tests can be difficult since the hot spot location will usually be near but not necessarily exactly where the minimum distance from the updated path to the source is located. Therefore, finding a frequency in the error changes with the number of waypoints can be made a little more difficult.

There are two simple ways to fix this problem. The first is the put into effect that at the end of the search, whether or not the final window limit was reached, the Bayesian code should run an update. The second is simply to not use a window size that is more than half the length of the number of waypoints used (as seen in Eq. 5.14.

$$window_{max} \leq \frac{len([waypoints])}{2} \qquad \textbf{Eq. 5.14}$$

Forcing the Bayesian code to update during the final iteration of a test, it can be seen in Figure A.73 that the problem is indeed solvable by implementing this simple modification into the code set. However, two important aspects of this situation should be mentioned. First, while this is an important error to find in the

Bayesian system being used for window size determination, it is an error that is unlikely to ever occur in real-world testing (either in simulation or not) due to the fact that the number of detection points is likely going to be larger and the search path will never have uniformly distributed detection points. Secondly, it should be remembered that for an accurate Bayesian-based search to be performed, the window size should be minimized to give the most updates without yielding false positives, the velocity should be lowered compared to Tier 1 flights to avoid having too few detection points, and a Bayesian result should be checked with a Tier 3 search if possible.

With the error source resolved, further refinement of the Bayesian code was able to be performed. Returning to a search space of 100m x 100m (with a *y*-separation of 5m and an *x*-separation of 10m this time), a very specific pattern can be seen given the flight path geometry in the count rate results. While the following resolution is somewhat dependent on the path geometry, it is a geometry that is extremely applicable to urban environments where streets and avenues could be used as path legs for larger search spaces if necessary.

Because the path moves back and forth through a rectangular region, the detection rate often has continually increasing peaks as the aircraft moves back and forth across the region and continues to approach the source followed by the opposite effect as it flies away from the source still in a back and forth pattern. Because of this, on occasion, two peaks of high strength appear in both the raw count data as well as the Bayesian probability data when the source is between two legs. Because of this, the hot spot should be able to be determined by placing it between the two peaks. However, in the instance of a set of data with only one major peak, this would result in a high degree of error. Instead, it was found that a simple weighted average of each location with respect to its corresponding Bayesian probability yielded a far better approximation of the source location than simply using the maximum probability. Eq. 5.15 shows the process used to determine the location of the hot spot.

$$loc_{hot\ spot} = \frac{\sum_{i=0}^{n}(p_{final_i} * loc_i)}{\sum_{i=0}^{n} p_{final_i}} \qquad \textbf{Eq. 5.15}$$

Furthermore, by only considering locations where the probability falls above the mean of the probabilities yields slightly better results than an all-inclusive weighted averaging method. However, because only considering locations with probabilities above the average will yield a similar result to the complete weighted average as the number of detection points increases, in favor of higher computational speed, only the all-inclusive weighted average is used in the final PADUA algorithm. The comparisons of the average distances from each method-determined hot spot to the source with respect to the window size can be seen in Figure A.74. In the figure, the “Single Maximum Method” signifies the original hot spot determination

method results that used the single maximum Bayesian probability value. The "Weighted Average Method" shows the results using the weighted average method consisting of all the values. The "Limited Weighted Average Method" uses the weighted average method that only considers values above the average probability to determine the hot spot location.

The following chart (Figure A.75) shows the average increase in accuracy using a full weighted averaging method for the hot spot determination. In the figure, it can be seen that there does not appear to be a proportional change in the accuracy from window size to the next except that as the window size increases past ~75, the incremental accuracy drops slowly. Rather, where the method has the highest increase in accuracy is approximately between the window sizes of 40 and 70, which corresponds to a window size approximately ¼ of the number of waypoints. More important is the fact that, because the window region that the increase in accuracy is most prevalent in also is where the accuracy of the window approaches an asymptote, once the window size is determined, the modified Bayesian hot spot locator method should be able to be applied to the code without discrimination.

To determine the optimum window size, referring back to Figure A.74, it can be seen that the data can be aligned to a third order negative logarithmic function fairly well (as seen in Figure A.76). Using the minimum of the curve in Figure A.76, the window size can be approximated to be 61 which coincides with the estimations made earlier (i.e. the highest accuracy is between window sizes of 40 and 70). Thus for a 100m x 100m search space (and any similar), it should be accepted that an update window size of 61 can be the most usable window length for the Tier 2 search tier. However, a window size as low as 20 should yield similar results since the difference between the hot spot accuracy from a window size of 60 to 20 is only about 7 meters (5.9m vs. 13.2m respectively).

Once the window size was determined, the last part of the Tier 2 experiments was to determine the optimum method to combine the S-G filtering and the Bayesian inferences (if any). In Tier 1, it was determined that ~136 was the optimum window size for the S-G filtering method. Because the Tier 2 search space is assumed to be significantly smaller than that of Tier 1 (1/10 in the testing, to be exact), the optimum window size for S-G filtering in Tier 2 can be assumed to be $1/10^{th}$ the size which means the window size should be 15 (since it has to be odd, instead of 14). Using this window size and running 1000 tests on the 100m x 100m region with flight paths using an *x*-separation of 10m and a range of *y*-separation values from 1m to 10m, it was found that there was no clear relationship as to when the S-G method yielded more accurate results compared to Bayesian statistics. If the source was detected early in the flight path, then usually Bayesian methods were more accurate than S-G filtering, but if fewer updates were able to be run (i.e. if the source was near the end of the flight path), then often S-G filtering yielded a

better result than Bayesian. Thus, there is a solid premise for combining both methods for Tier 2 searches.

Because no clear relationship was able to be found though, numerous different analysis methods were tested on tens of thousands of data points to try and discover if there was a way to combine methods to yield a reliably more accurate estimation of the source location. The methods tested can be seen in Table B.4.

Running numerous tests with each of these methods yielded the conclusion that S-G filtering, weighted Bayesian, and a combination average of the two methods should be analyzed further. To analyze each of these on a search grid similar to that which would be likely seen in a real situation, a 100m x 100m search space was used with a *y*-separation spacing of 10m and a varying *x*-separation spacing between 1m and 10m. As found earlier, an optimum Bayesian update window size of 61 is used. The tests returned a varying result based on how close the flight legs were together. In Figure A.77, it can be seen that, on average, S-G filtering will yield a result not better than 16m from the source. The weighted Bayesian method tends to produce more accurate results while the average of the two methods of course is slightly worse than the Bayesian method in most cases. However, the points shown on the figure each represent 1000 tests in which the most accurate method varied significantly from one test to the other.

For each of the *x*-separation sets (*dx*), the instances of each method being more accurate than the other was logged, summed, and plotted to produce Figure A.78. In it, there is no clear trend, but—not including the first point, a decent average can be gathered from each to determine the final weights for each method. To do so, weights were only taken from *dx* sizes 3 and higher since it is unlikely that a Tier 2 environment will ever have legs less than 3 meters from each other.

Using this process, it was found that the weights for the weighted Bayesian method and the S-G filtering should be 653 and 347 respectively. By plotting the results however (Figure A.79), it could be seen that the accuracy is very similar to the pure Bayesian weighted method.

Furthermore, by observing the changes in standard deviation for each complete set of data (Figure A.80), the Bayesian results also yield higher accuracy overall than does the weighted average of the two primary methods. This of course is a result that should be expected though since the weighted average of the two methods will have Bayesian parts within it as well as S-G filtered parts. Since the source location cannot actually be known in the real world though, little other methodology can be utilized to try to increase the efficiency of the hot spot determination.

Thus, for this part of the project, only the weighted Bayesian method will be utilized for Tier 2 hot spot determination, but in later portions of the project (i.e. multi-aircraft methods), a simple average of the weighted average of the Bayesian locations and the weighted average of the S-G filtered locations will be used when the Tier 2 search space is small enough to do so. Using weighted S-G values works well to help mitigate error developed by Bayesian statistics if the source is near the end of the path, but such a method can only be used in smaller search spaces since the S-G values fluctuate far more than do Bayesian probability results.

It is important to remember here that the figures just shown in the refinement process for Tier 2 search methodology may appear basic, but the amount of data that is condensed down into the figures is immense. As mentioned, in the section discussing the determination of the hot spot method, each point was a result from 1000 individual tests. Preceding that, the window size had to be determined which consisted of 100 individual tests per window; many tests were run with window sizes ranging 300 steps. Considering that the response generator itself was made from thousands of data points, the resulting condensed methodology is determined from millions of data points measuring various relationships, distances, accuracies, and characteristics. Because of this, averages are often used to display the data and determine what step should be taken next; however, it is important to note that when averages are taken, if the data is extremely noisy, then the average may be worthless. This is why the standard deviation is often discussed in this project. The standard deviation (Eq. 5.16) can be considered a measure of the spread of a data set and thus can be used to ensure that a variable being averaged is reliable.

$$\sigma = \sqrt{\left(\frac{1}{N-1}\right)\sum_{i=1}^{N}(x_i - \bar{x})^2} \qquad \textbf{Eq. 5.16}$$

With the methodology completed for single aircraft Tier 2 experiments, real-world simulations were able to be run in order to compare the results between the theoretical search space tests that were used to develop the methods for the full algorithm and the practicality of how they are applied. The following figures show four different tests. Figure A.81 had a source near the end of the search space and close to the path. Figure A.82 and Figure A.83 used the same source location while Figure A.84 had the source located just outside the search region. In each of the figures, the location with the maximum counts along the path is signified by a red marker while the location with the maximum Bayesian probability has a blue marker. Each of the four tests shown have the same location for the maximum count rate and the maximum Bayesian probability, so only the blue marker can be seen in each figure. The location found using the Bayesian probabilities as the

weights for each of the locations tested is shown with a purple marker, and the source is placed for each test at the location shown with the yellow star. The beginning of each flight path is located in the lower left corner of each figure shown with a small black cross. Each planned path has the same dimensions (100m x 100m with an *x* spacing of 10m and a *y* spacing of 50m between waypoints).

Using the data shown in the four figures above, the accuracy for each test can be shown as the distance from each max or weighted location to the source. As can be seen in Table B.5, the accuracy for such a large search area is exceptionally high. While the first test has the highest accuracy (as it should since the source is closest to the path in the test), even when the source is outside the search region and approximately half way between longitudinal legs of the flight path as in the fourth test, the weighted Bayesian method was still able to predict the location of the source to less than 6m from a 10,000m^2 initial Tier 2 search region.

From the single aircraft Tier 2 experiments, it is obvious that the application of Bayesian statistics in the correct context has very high advantage over analysis simply using a single non-manipulated set of raw count data. While there are numerous other changes that could be applied to the method if, for example, energy spectrums and neutron/gamma discrimination were introduced, such detector data analysis techniques are beyond the scope of this project. The incorporation of swarming methods will be discussed later in Chapter 6 where the source-locating methods refined in this section will be applied to multiple aircraft and detectors, but refining the analysis methods much further would likely result in an algorithm that is very specific to radiation detection. Additionally, while the results found in this section were exceptional, it must be remembered that Bayesian statistics have a high chance of error in certain situations; this is the primary reason for introducing Tier 3 search methods. In Tier 3, the exact location of the source will be both refined further and verified.

Tier 3 Experiments

Summary and Setup

This section describes the final tier experiments for single aircraft testing. The first goal was to refine and test the right-angle-turn (RAT) detection method while the second was to develop the code in such a way that the aircraft is able to fly completely autonomously. This is the most important tier test of the three since nearly all of the methods tested here are unique.

The code was run using Python 2.7 and replicated radios as described in Chapter 4.

Methodology

The basis behind the third tier of searching, as discussed, is to refine the location of the source to a more exact location if possible and verify that the hot spot location determined in tier 2 is accurate. To do this, a combination of raw counts and Bayesian statistics were used in testing to take the RAT method and refine it from theory into a workable application for real-world detection.

The RAT method was developed by the author as a possible way to verify Bayesian results since there have been numerous difficulties in both real-world application of Bayesian statistics as well as in research with false positives occurring. The theory behind the RAT method is simple, but works best with a near-uniform source flux emission pattern.

The RAT method operates under the presumption that, if a source emits radiation uniformly or near-uniformly and a detector is moved through the source's sphere of influence, the count rate should generally rise as the detector approaches the source and then begin to fall again once the maximum count rate along the path is crossed. If the source emission is spherically uniform, the point of maximum average count rates detected will always occur at a point along a chord directly perpendicular to a spherical radius of the emission. This can be seen two-dimensionally in Figure A.85 where a black circle indicates the sphere of influence around a source and the blue line shows an example of a detector's path through the area. No matter where the path lands inside the circular/spherical area, there will always be a point midway that is closest to the source and also is perpendicular to a line of radius.

With this geometry understood, it should then be inferred that if a detector moves forwards as long as the detected rates continue to increase and then makes a 90-degree clockwise turn when the rates start to decrease, then eventually the detector should locate the source. The main limitations to this would be the noisiness of the response and the accuracy of the detector. To demonstrate this, a series of snapshots from a theoretical code written in Matlab are provided in Figure A.86 below where a detector's path progresses in the figures from top-to-bottom and left-to-right. In the figure, the black line represents a part of the detector/flight path where there is no detected rate above background. Once the path is indicated in blue, then the rates detected are higher than background and the RAT method is initiated. Although this is only a very theoretical example to prove the validity of the RAT method, it shows that, at least in theory, the premise is correct and the hypothesis stands that using a series of right angle turns based upon the changes in count rates, a source should be able to be located given a uniform distribution of flux.

While the perfect theoretical situation is not applicable to the real-world (since variables such as response noise and flight dynamics were not taken into account),

a basic algorithm was able to be developed that could be built upon and modified as needed to make the method operate correctly in the real-world. The main search algorithm diagram then can be seen in Figure A.87. If the count rate detected at an instance is greater than a threshold count rate, then the RAT system can be activated. Additionally, if the inputted target count rate is exceeded, then the algorithm can state that the target has been located. Otherwise, so long as the count rate exceeds the threshold, then it must be checked if the count rate is greater than or less than the previous sample. If the count rate is increasing, then the flight should continue forwards while if it is decreasing, the flight should make a 90 degree turn. It was found in testing that a method had to be developed and implemented to determine when to make a turn due to the noise in the data; this will be discussed later.

Applying this theory to the full algorithm that includes flight control was an exceptionally difficult task for such a simple process. However, because using the RAT method means that the flight path is not predetermined and the search area should be small, the flight can be performed entirely autonomously without any ground control needed for control or search algorithms. The only purpose the ground station serves in Tier 3 other than sending the aircraft to an initial waypoint for starting the tier search is to receive the position, count rate, and target estimate information from the aircraft. Also, because the third tier is only designed for use with a single aircraft, there is no need to include swarming in the third tier algorithm.

Initial results using the RAT method were promising, but there were several variables within the method that had to be optimized. Primarily, changing the altitude, the ground speed, and the length of the response array were the first variables tested. The response array refers to the length of the recorded array of count rates between detection points; this array is used to determine if the trend of detection values is increasing, decreasing, or constant during forward aircraft movement. Numerous combinations of averaging methods, comparisons of changes with standard deviation, and differential changes across the array were all attempted. But at the end of the testing, it was found that simply looking at the first and last values of the array produce the best results as they usually represent near the maximum and minimum values of the array since they are recorded at the furthest distance apart.

To explain the response array further, consider an aircraft moving towards a source during detection in Tier 3. In such an instance, so long as the aircraft is within the sphere of influence of the source, the count rate should continue to increase. However, because there is noise in the detected count rate, the response array must be long enough that the majority of noise is dismissed but short enough that the travel distance is not excessive which would require too much search time.

To display the refinement process of the response array, consider a hypothetical instance as illustrated in Figure A.88. The possible array length from a single value to ten values is represented on the *x*-axis. The left-most side of the figure signifies the beginning of the flight path while the right side represents the end of the flight path for the illustration. The array would append *y*-values during the course of the flight for the length determined by the *x*-axis. The blue region shows the possible hypothetical count rates at each position. Instead of having a single line of average values, a region must be shown since noise must be accounted for in the detected signal. For example, for an array of length 1 (x=1), the hypothetical count rates (for this example only) range from 10c/min to 13c/min. Similarly at x=10, the range is between 19c/min and 22c/min.

Considering this hypothetical example and data, if the array length to determine whether or not to make the 90 degree turn in the RAT algorithm (i.e. the turn choice) is taken to have a length of 3 as shown in Figure A.89, then the range of possible detection rates for the first value range from 10 to 13 while the range of rates for the last/third value are 12 to 15. This means that, although there is a chance that the array will signify an increase in count rates (for instance, if the first value of the array is 10 and the third value is 14), there is also a chance that the array will show a false positive such that the count rate decreases from the first to last array value (such as having a first value of 13 and a last value of 12).

Alternatively, if the array has a length of 10 in this example, the length used would be far too long compared to what is necessary, thus causing the aircraft to fly too far which wastes time and decreases the accuracy of the source location estimation. In the example used for Figure A.90, the range of the first possible value is again between 10 and 13, and the range of possible values for the last value in the array is between 19 and 22. This means that, even if the maximum possible value at the first location for the array is yielded (13c/min) and the minimum possible is yielded for the last index/location in the array (19c/min), there is still a gap of 6c/min in this example. Whether or not there is a difference of 1c/min or 6c/min, the algorithm will still be able to identify that the count rate is increasing; therefore, there is no reason to use such a large array in this situation.

Rather, the optimum array length for this example can be seen in Figure A.91 where there is, as just stated, a 1c/min spacing between the maximum possible value for the first index in the array and the minimum possible value in the last index. In the example, this would be an array length of 5.

Results and Discussion

The first step in refining the actual Tier 3 search algorithm was to compare the results of the basic algorithm using the main three variables (detection array length, ground speed, and altitude). With these variables, it can be inferred that

changing the array length and altitude should be sufficient (since changing the ground speed effectively changes the distance between the first and last values of the array length. Each test was performed with the primary analysis result being the average location of the path (the average of all the latitudes and longitudes) being considered the initial source location estimate. The average of all the locations was used since it was found that due to the speed, detection time, and detection array length, if the source can be located, the path quickly becomes a square around the source. Since the average of a rectangular shape's perimeter values should always be the middle of the shape, averaging the flight path positions as the aircraft moves around the source should refine the source's location to be the middle of the perimeter path. An example of the estimation progression using this theory is displayed in Figure A.92 where the start estimation is signified by an orange marker, the actual source location by a green marker, and the refinement of the source estimation by the blue line starting at the orange marker. As can be seen, because the aircraft was flying in a right turn configuration, the refinement ends up as a spiral shape moving towards the source (since a moving average is used using the entire path length as the length increases).

The first set of tests were used to determine the optimum altitude. While the best altitude for detection would of course depend upon the source strength and emission geometry, two main bounds should be put in place during any Tier 3 search. The first is that the altitude flown by the Tier 3 vehicle should be lower than that of the Tier 1 and Tier 2 search patterns so that, in a real world environment, if other tier searches are being performed by other aircrafts at the same time, collisions can be avoided. The second bound is that the altitude flown should be higher than the majority of obstacles such as telephone wires, streetlights, and—if the location has them—houses and other low-lying structures. The method of determining the test length was when the number of responses recorded reached an index of 250; the responses were only recorded when the aircraft was moving at a groundspeed greater than 1m/s since the aircraft slows when approaching a waypoint/target destination.

Figure A.93 through Figure A.97 show the results of the tests with varying altitudes. The black line represents the flight path in each figure while the blue line represents the refinement of the source location estimate (just as in Figure A.92). For testing, the aircraft is flown to the initial waypoint (indicated by the teal marker) and then northwards 10 meters to the start location (indicated by the red marker). The source location is indicated by the blue marker. While many conclusions can be drawn from this sequence of tests, by just observing the path patterns, it can be seen that generally as the altitude increases, the path becomes increasingly square given a set detection array length. This is likely due to the response from the source changing faster over the same length of distance perpendicular to the source emission rays. For instance, if an aircraft is flown for 10 meters in a straight line, as the aircraft's altitude increases, the sphere of influence seen by the aircraft

from the source's emissions decreases in size thus effectively shrinking the size of the altitude-plane's 2D circle of influence. In other words, if a source is located on the ground, as the altitude increases away from the source, the cross section of the sphere of influence around the source at the same altitude becomes smaller. Therefore, if the array length is short enough and the altitude is low enough that the start and end responses contain count rates influenced by the source, a higher altitude should yield a more exact result. Alternatively though, although the flight path tends to increase in consistency as the altitude increases, the accuracy of the source location estimation decreases as small alterations in the path make far larger difference in the average of the source location which can occur far easier when the aircraft is already further from the source (compared to when it is at a lower altitude).

From the tests shown in the figures above, an optimum altitude of approximately 15m was found. Again, this is a general value that can be used for this specific source, but changing the source type, strength, and shielding will all change the optimum altitude. Additionally, while the location estimates are overall better than the estimates found in Tier 2 searching, the purpose of Tier 3 is not merely to attempt to refine the source location further, but rather, it is to verify the hot spot location determined by Tier 2 is accurate. The initial waypoint location shown in all the above figures is from a Tier 2 simulation test. The results from the altitude testing can be shown below in Table B.6.

The next part of the Tier 3 initial testing that had to be performed was that of the array length determination. All the altitudes resulted in a fairly accurate location estimate for the source, but as the array length changes, the estimate may or may not be as accurate depending on the amount of noise in the signals detected. In Figure A.98 through Figure A.102, it can be seen that as the array length decreases, smaller perturbations are detected which means a more accurate location should be able to be found with a smaller array size in theory. However, the flight patterns that use a smaller array size are far noisier themselves meaning that a longer flight path is needed for an accurate estimation to be developed. Note that in the longer flight paths, the geometry of the path is squarer since the changes calculated are definite changes while those paths that show a geometry with corner sub squares form due to noise detection (as described in the previous part of this section). This means that, while the shorter array lengths do have the potential to yield a more accurate result, they also have a higher chance of yielding a false positive or false negative.

From the tests shown above, Table B.7 was produced, which shows that the general trend of accuracy is a split function. That is, from an array length of 5 through 15, the estimation continually becomes more accurate; after a length of 15, the estimation becomes less accurate. This is also illustrated in Figure A.103. It is likely that with a response array length less than 5, the estimation becomes

exponentially less accurate, but from the results displayed in the figures, it could be seen that with an array length of 5, there was already a considerable chance to develop error within the testing. For the purposes of this experiment, it can be considered that an array length of 15 produces the optimum results and can be used for the duration of the testing.

With the approximate optimal values determined for the altitude and response array length for Tier 3 testing (15m and 15 values respectively), further attempts at refinement were then attempted. Numerous different methods were endeavored including varying the ground speed throughout the search sequence, moving the aircraft to a refined hot spot location after each circuit was completed, and various basic statistical methods. However, the combination of the noise incurred in the detected signal and the limits of precision of the flight control system did not allow any further complexities to be compounded into the sequence. There were only two successful alterations to the process that were discovered.

The first of these alterations to the original process consists of using a weighting method for the location estimation. Rather than simply averaging all the recorded latitude and longitude locations, a weighted averaging method was utilized using the normalized raw count rates as the weights for their respective latitudes and longitudes. This process is able to be performed because, while the average of all the locations recorded as the aircraft moves in a square or square-spiral-like pattern, the count rates will fluctuate based on both the actual distance from the source and the noise encountered from the source. By weighting each location with its respective normalized detected response rate, the estimated location should be able to be shifted in a direction that is consistently closer to the source compared to using an unweighted method. Furthermore, by using a weighted method with the RAT method, not only is the source location able to be verified and refined, but if the source being targeted yields a non-uniform flux distribution geometry, this method should be able to still locate the source fairly accurately since the method relies only upon the instantaneous raw count rates rather than any type of probability predictions/PDF’s or curve fitting. In Figure A.104, an example of the third tier search path can be seen where the aircraft flies towards the first waypoint (i.e. the hot spot generated from Tier 2 search) and then implement the RAT method. In the top part of the figure, a 3D flight path is shown where the *z*-axis represents the count rate. The same data is displayed in the bottom half of the figure where brighter markers signify higher count rates and darker/bluer ones show lower count rates.

Using the same limitations and bounds as those developed in the earlier sections, tests were run up to 400 recorded response rates (i.e. any responses detected while the aircraft was flying at a velocity over 1m/s). Implementing this refinement methodology, ten tests were run with the source and initial waypoint in the same respective locations for each test. The differences in the averaged location method

and the weighted averaged location method can be seen in Table B.8. From this data, it can be seen that the increase in accuracy is able to be seen in every instance except one. In fact, there is a 38.94% average decrease in error from using the normal average method to the weighted average method with an average distance error for the new estimated source location being 3.2m from the source compared to the previous method's average of 5.3m.

However, because this method does not currently have a way in which to handle outliers in a generic manner, it is not applied to the final version of PADUA including the multi-aircraft tests in the following chapter. This is a method that will require future work to refine further for real world usage. The data shown here is specific to simulation, but could be used tentatively in a real world environment.

With this information, the single aircraft design of each of the three tiers was found to not only be functional but also to be accurate and efficient. To have a tiered method that is able to narrow down the location of a radiological source from more than a square kilometer area to between 1 and 4 meters can be considered successful. The next steps of the project were to combine the tiers before moving on to swarming systems and multi-aircraft testing.

Combined Single Aircraft Triple Tier Algorithm Test

Summary and Setup

This experiment had the only goals of refining the system to operate fully with tier transitions automatically and determining the true computational requirements for the system developed. To do so, the experiment was run for a single test. The source was located far from the Tier 1 path to ensure the reliability of the search methods. This is not so much of an experiment as it is a description of the refining of the complete algorithm for a single aircraft system to see how it operates.

A second test is shown in the end of this section to display what might be expected if the Tier 3 search does not find a source.

The code was run using Python 2.7 and utilized replicated radios as described in Chapter 4.

Methodology

The variables for the main test are shown in Table B.9. All variables within the table are major characteristics of the code that must be input each time a simulation instance is run. Since this experiment is simply for the purpose of ensuring that each tier's code works well when combined into a full algorithm, there is little to

add to the methodology that has not already been discussed. The only main difference in the algorithm for this experiment was the method of determining tier change. Since each of the other sections thus far described single tier operation, a method had to be developed for changing tiers automatically. To do this, the ground station performed analysis just as described in the previous sections for each tier before updating a global variable in the code that specifies the determined hot spot location. Then, the ground station relayed both the new hot spot location and the new tier number to the aircraft.

Results and Discussion

Test 1

The primary two results for this experiment are the overall flight path and the accuracy of the final estimation. Both results were excellent and proved that the earlier efforts describes for refining PADUA for single aircraft use were valuable.

In the second test, the same type of inputs as the first test were used except that the source was made to be located between two legs of the Tier 1 search flight path rather than nearer to the path as in the first test. This should be exceptionally more difficult for the algorithm to locate the source since elevated count rate values will be smaller than if the source were located closer to the path. This tests the robustness of the Savitzky-Golay filtering method part of the algorithm.

In Figure A.105, each of the three tier paths are shown along with the location of the source which is in the middle of the Tier 3 RAT method search area. The refinement process worked just as well as in the first test even though the situation was more difficult for the algorithm to locate the source than in the first test.

In Figure A.106, it is evident that there are indeed few elevated count rates. However, due to the S-G filtering using a third order polynomial for the curve fitting in least squares, the search method was able to recognize that the elevated values were not merely outliers in this case. This enabled the Tier 1 search process to estimate a start location for Tier 2 search.

The results of the refinement process using hot spots can be seen in Table B.10. Here, the difference between the accuracies of each tier are evident. Using this type of refinement process allows the algorithm to search a large area as fast as possible using the optimum methods for the region size and flight speed as proven in these tests. Note that three different search method results are shown for the source location estimate from Tier 3. As stated previously, the normalized method operates the best out of the three methods, but it must be ensured that outliers are not possible when using such a method. This can be achieved by placing upper limits on the count rates recorded, but to ensure the estimate is accurate, the two

other primary methods' results are also shown. As can be seen from the table, the results are exceptional with the final source estimate using the weighted method being only .14 meters away from the actual source location and the normal average yielding an estimate that is .59 meters away. Using the weighted method based on the normalized raw count rates, the estimate is less than a foot from the source.

The errors of the three methods from the Tier 3 search analysis can be seen in Figure A.107. While the weighted method follows the average method closely but with a slightly higher accuracy, the moving average method curve is far smoother. This is the primary reason that both the average and the moving average are used. While the average (weighted or normal) will always yield a harmonic result when locating a source, the result can still appear noisy. When the moving average is used, if the result does not asymptote, then it can be stated assuredly that no source has been found.

In Figure A.108, the progression of the three methods' refinement locations are displayed. Again, while the weighted method yields a slightly better result than the normal average method, the moving average is used primarily to ensure the continual refinement shape of the process is spiraling inwards according to the RAT search theory.

The final part of the analysis to be discussed for the single aircraft demonstration is the time requirement for each part of the search compared to the whole. As can be seen in Table B.11, Tier 2 requires far less time than Tier 1; however, Tier 2 is also the only highly constrained tier. Tier 1 is able to be any shape and any size as needed by the mission requirements. Tier 3 should continue to have increased accuracy the longer that the flight endures since the estimated source location continues to be refined. However, Tier 2 must operate within a medium to small symmetrical pattern for the best results. This is primarily due to the weighted location method based off the Bayesian probabilities that is used to determine the hot spot. Note that the entire refinement process only required 39 minutes from start to finish. The time could be made shorter depending on the requirements of the accuracy needed from Tier 3.

Test 2

In Figure A.109, a test was run were the second tier purposely yielded a bad hot spot for the Tier 3 waypoint. This was done via code modification and would not happen during normal usage of PADUA in the same situation. However, to display what would happen if the Tier 3 search was performed in a location where there is no significant source emission, Figure A.109 is provided. This type of process is explained further is in the following section.

Single Aircraft False Positive Test

This section outlines the results found from a false positive test with a single aircraft full search. That is, the source location is not placed in the search region such that Tiers 1 and 2 are intended to yield poor results for the hot spot. This in turn should allow the differences to be seen in the individual tier refinement data. In particular, the Tier 3 refinement process is shown since part of the intensions for the Tier 3 search is to not only refine the location estimate of the source but also to determine whether or not there is a source present.

In Figure A.110, the flight path of the aircraft is shown with each tier's portion of the path color-coded. While Tiers 1 and 2 appear as they would normally since they are determined as waypoint-based paths, the path for Tier 3 obviously is random. In Tier 3, it is obvious that there is not a decent source signature found since the RAT method continues to tell the aircraft to move without any single extractable rectangular region.

More importantly than the path though is the refinement process of the location estimate itself. A graph showing the refinement of the latitude and longitude from a different test is shown in Figure A.111. This is a figure from Chapter 6 derived from a successful Tier 3 search. In the example figure, three main aspects are important. First, there is a clear separation between the moving average and the weighted average. While this is not always an absolute sign that there is a source located, it is a good signature to look for since the weighted average should continue to yield a different result than the normal average if there is a significant source influence in the region. Secondly, since this example shows a Tier 3 instance flown to near-full refinement, the moving average of latitude and longitude refinements nears a 0-degree slope. Since the source has been located, eventually there should be nearly no change in the moving average of the location estimate. Lastly, once a source has been located, there should appear to be a periodic element form in the location estimate results. This is clear in the example figure for both the latitude and longitude.

Conversely, Figure A.112 shows the data from a false positive Tier 3 search refinement process. While the moving average somewhat levels off, it does not appear to reach a significant asymptote. More obvious are the method separation and periodic indicators. There is virtually no difference in the results using the weighted or normal average methods, which should indicate that there is not a significant change in the detected count rates. Even more obvious though is the lack of periodic motion in the source estimate refinement location. Especially for the latitude, there is no periodic trend at all that can be seen in the estimate over the course of the process.

Lastly, the combined results shown in Figure A.112 are shown in Figure A.113 where it can be seen that the combination lacks the characteristic swirl as seen

with true positive results (reference Figure A.92 through Figure A.102 and Figure A.108). So long as Tier 3 is performed autonomously, this is perhaps the clearest indicator that there is a false positive being chased. The only exception should be if the source is moving in which case there should still be a separation between the weighted and normal average methods.

Walking Tests

To test the versatility of the search methods developed in the single aircraft sections, a test was performed with the same basic calculations with the simulated detector on the Toughbook. The Toughbook also had a GPS module connected to it from which the position was obtained. By walking around outside with the system, the framework version of the PADUA search methods could be tested in unfavorable situations. Specifically, the paths used for the search were pseudo-random with no symmetry. The latitude, longitude, and count rates were all logged during walking and were then analyzed in post processing using the methods previously described. The walking tests *only* tested Tiers 1 and 2.

For the first walking test, a source was placed artificially at the University of Tennessee. Because the test was performed by walking, the movement pattern was extremely limited. As mentioned, this was done purposely so that the underlying methods in the PADUA algorithm could be stress tested. Thus, these tests are not representative of exactly how an aerial search would be performed.

Figure A.114 shows the representation of the Tier 1 search performed during the test. The lighter colored markers indicate the locations with higher-value count rates detected while the darker ones represent those with lower count rate values. The red star indicates the location of the simulated source.

The Tier 1 path can be seen overlaid onto a satellite image in Figure A.115. This was created using Keyhole Markup Language (KML) in the desktop version of Google Earth. The image is situated with north being on the top edge of the figure.

The count rates differences can be seen more clearly in Figure A.116 and Figure A.117 where the height of the path indicates the value of the count rate detected. These figures show that the maximum count rate occurs on the Eastern side of the Tier 1 path.

Using the same Savitzky-Golay techniques that are the basis for the PADUA Tier 1 search, Figure A.118 was created showing that there was a slight error in the hot spot location. However, the x-axis in Figure A.118 only shows the total path distance and not the location of the detection points. Referring to Figure A.119, it can be seen that the hot spot location determined using the Tier 1 results was still very close to the source. The red dot at the end of the path indicates the Tier 1 hot

spot while the red star indicates the location of the source. The black line is the walking track from the test.

For Tier 2 testing, a second walking test was performed in the same manner that the first test for Tier 1 was performed but using the Tier 1 hot spot as the general search area. Again, in the same manner as the underlying calculations used for the Tier 2 analysis in PADUA, the data was processed with weighted Bayesian statistics and the hot spot was determined. In the top half of Figure A.120, the data points are again signified as having a higher count rate if the color is lighter compared to those with a lower count rate that are shaded darker. The source location is shown with a red star in the top half of the figure. In the bottom half, the track is shown in black while the Tier 2 generated hot spot is shown in cyan. The source is shown with the red marker.

The analysis results from the Tier 2 post-processing can be seen in Figure A.121 where it is obvious that the Tier 1 hot spot was very close to source since the count rates detected during tier 2 search had such a high discrepancy between the maximum and minimum.

These results show that the underlying principles used for the development of PADUA do indeed operate as they should; however, they also show that the path pattern and sample spacing do matter. Especially for a short Tier 1 path such as the one shown here, the S-G filtering has difficulty in determining the maximum integral location. However, so long as there are enough sample points, the methods used in PADUA should operate as designed.

Lastly, although not important for the simulation project or the development of the PADUA algorithm, one other aspect was noted during the walking tests that should be mentioned. The accuracy of an estimate is only as good as the GPS systems being used on the detection platform. A small amount of testing was performed with two different antennae used on the GPS unit in two different scenarios. The first scenario was a test of how accurate the GPS is inside a brick building and the second was a test in unobstructed space outside.

As would be expected, in the obstructed situation, the GPS performed very poorly with an error of as much as 37 meters. The latitude and longitude variations shown in Figure A.122 are a series of 100 position samples in the same place (i.e. the GPS was not moving). The same is shown in Figure A.123 for the unobstructed environment test. In the unobstructed test, the error ranged from 4 to 8 meters. However, when the PADUA algorithm operates, if the GPS used with it only has 6 meters of accuracy hypothetically, then an estimate can never exceed this accuracy. The position ranges can be seen in Table B.12.

CHAPTER 6

MULTI-AIRCRAFT (SWARMING) EXPERIMENTS

Summary of Multi-Aircraft Experiment Setup

The primary goal of this chapter to describe the swarming algorithm developed specifically for this project, the initial testing of the algorithm in theoretical form, and the implementation and testing of the swarming algorithm(s) into the full system algorithm in simulation. Two types of swarming methods are used: pattern- and algorithm-swarming. While pattern-swarming is quite basic albeit effective when implemented properly, the Bound-Continuity swarming algorithm used in this project was designed specifically for the type of search required by this system. The pattern-swarming method is used simply as a base with which to compare the results of the Bound-Continuity algorithm. Based off Navier-Stokes Continuity Equation, the swarming algorithm used for the PADUA System is designed not so much with natural pattern assimilation or replication in mind but rather optimizable moving detection area. While testing was successful with the Bound-Continuity algorithm, faster radios would enable better results.

Pattern-Swarming Algorithm

Pattern swarming is by far the easiest type of swarming model to use today for any type of experiment, application, or simulation. Not far removed from the original concepts of swarming, for PADUA, the pattern-swarming methodology is exceedingly simple and was originally implemented primarily as a comparison tool to use against the Bounded-Continuity swarming algorithm. However, outside of comparison testing, it was found that pattern-swarming holds major advantages for certain parts of the search sequence.

The pattern swarming method used in PADUA accepts two inputs and utilizes parts of the aircraft control API from Dronekit to reduce the calculation and coding complexity. The two inputs required for the swarm pattern for each aircraft are the target bearing and the target distance. Rather than using velocity commands such as those proposed by Craig Reynolds, instead the waypoint commands are utilized from Dronekit to direct the follower aircraft to iteratively attempt to go towards a target point that is the specified target bearing and distance away from the lead aircraft. It should be noted that the bearing is in relation to north at the following aircraft's start position.

Using a "goto" method (i.e. send to waypoint) is not identical to fundamental swarm theory since the three rules are not implemented explicitly (see Chapter 1), but it is similar in many respects. Using a waypoint method instead of a target location with continual velocity changes can work for the PADUA system because the

compactness of the swarm in the system is quite loose and expected to be composed of relatively few aircraft. In a high number of aircraft are required, then multiple swarm systems can be implemented in different regions and a separate ground control unit can filter the combined incoming data. Additionally, the goal of the swarm is generally not going to be to replicate a tight-knit natural pattern but rather will be to cover the maximum amount of area possible in order to hasten the detection of the unknown source location. Thus, while velocities could be continually changed to try and keep the distance margin between the lead aircraft and a following aircraft more constant, in reality it simply does not matter as much for this type of system compared to tight-pattern swarm systems. Furthermore, because it is unlikely that more than three or four aircraft would be close together in a PADUA swarm, additional sensors for collision avoidance combined with staggered altitudes should be adequate to provide the majority of separation required for swarming. Lastly, each following aircraft broadcasts its location in a burst manner over a similar frequency; this means that a very short code can easily be written to identify if there is another aircraft nearby based on comparison of received locations and the local location.

Bounded Continuity Swarming Algorithm

The swarming algorithm developed for this project was built from scratch to attempt to alleviate some of the difficulties involved in real-world low-cost detector-based searches incurred by other swarming methods. While the Bound-Continuity method draws inspiration from other swarming models that are based off natural phenomenon, this model is intended not to replicate natural swarms at all. Rather, it is designed to use equations from fluids to attempt to achieve a shaped result that allows the swarming system to operate similarly to a variable form fluid control volume. The primary intention of this algorithm is to allow near-constant detection time-per-area across a small group of swarming agents.

The main attraction of utilizing swarming agents for radiation detection (as well as many other types of non-directional detection) are the increase in both the detected area-per-time (since the agents should be able to cover more lateral area for a line of linear movement) and the accuracy of detections due to an increase in sample quantity-per-unit-area. However, while using a simple pattern swarming system indeed helps the overall robustness of the system, there is still an issue with optimized detection abilities so far as uniformity is concerned.

To explain this issue, consider a small plastic scintillator package that is able to obtain an accurate count rate in a background environment every ~3 seconds. When the aircraft approaches/departs a waypoint, its velocity is lower than cruise (since it slows down on approach and has to then speed up again when departing). Thus, if its cruise speed is hypothetically 10m/s in a search space and its average velocity is 3m/s within 10m of a waypoint, far more points detected occur around

the waypoint than during the main legs of the search. At cruise in such a situation, there is a detection point every 30m whereas around the waypoint there is a point every 9m. With multiple aircraft, the effective distance between detection points can be significantly reduced even if only flown in pattern form (assuming that the detectors on the two aircraft are not perfectly synchronized). Also, if the detector approaches a source, the amount of time needed to fill a buffer used to find the count rate will drastically decrease (again, following the inverse square law as the detector moves towards/away from a source). But assuming that the speed used during main cruise sections of the legs is low enough to obtain an accurate detection reading with two aircraft, then obtaining more data points closer together near waypoints is unnecessary. In other words, when the forward velocity of the swarming agents speeds up, it would make sense for them to move closer to the lead aircraft's flight path, and they should spread out when the lead aircraft slows down.

Perhaps the best way to model this type of action is with the first equation of Navier-Stokes. The continuity equation can be seen in differential form as Eq. 6.1. In the equation ρ represents the density of a fluid volume, t represents times, and u represents the velocity of the fluid volume. In the process discussed here, the density can be said to be the number of agents per rectangular area made between them. That is, if there are two agents, the latitude and longitude between them in a rectangular manner is considered to be the density (i.e. 2 agents/A). However, it will be seen that this is negligible since it is desired to keep the density constant with the area.

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho u) = 0 \qquad \textbf{Eq. 6.1}$$

Expanding the velocity component of Eq. 6.1in three dimensions, Eq. 6.2 can be formed.

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u)}{\partial x} + \frac{\partial (\rho v)}{\partial y} + \frac{\partial (\rho w)}{\partial z} = 0 \qquad \textbf{Eq. 6.2}$$

Separating the velocity variables onto on side of the equation and moving the density components to the other with the assumption that density will remain constant, Eq. 6.3 can be produced. Furthermore, it is assumed for the purposes of this experiment that z-direction variables can be ignored, as all operations will take place only with latitude (y) and longitude (x) alterations.

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = -\frac{\partial \rho}{\partial t}\frac{1}{\rho} \qquad \textbf{Eq. 6.3}$$

Since it is assumed that the density of the system will remain as close to constant as possible throughout the entire search time, the time-dependent density can also

be assumed to have a value of 0 thus cancelling the entire right side of Eq. 6.3. From this, the final simplified equation used for the governing of the swarm system can be seen in Eq. 6.4. This can be read as the change of velocity from point A to point B of the system in the linear direction should be met with an equal and opposite change in velocity in the lateral direction from point A to point B.

$$\frac{\partial u}{\partial x} = -\frac{\partial v}{\partial y} \qquad \textbf{Eq. 6.4}$$

While this is only the base equation of the swarming algorithm developed for this project, it allows for the swarming system of two agents to be described in simplified and familiar terms with a likeness of an incompressible fluid. What should happen in theory then with a double agent system can be seen below in Figure A.124. In the example, the upper left-hand corner of each instance is the location of the lead agent while the lower left is the location of the following agent. It is assumed that the lead aircraft *is not* governed by the swarming algorithm in any way but rather is following the guidance of either a predetermined flight path, a pilot, or automated commands from a ground control station. There are three instances shown in the figure. In each, it is assumed that the lead and following agents are both moving in the positive y-direction. The system shown furthest left is the initial state of the system, throughout all instances of the flight, the two aircraft should attempt to move to keep an equivalent area. The middle instance in the figure represents an instance where the lead agent has sped up from its initial state. In such a situation, the following agent can be seen to move inwards while attempting to match the forward velocity of the lead agent. In the furthest right instance, the system can be seen to have spread out in the *x*-direction since the lead aircraft has slowed down. Ideally, this is how the swarming algorithm should operate. This means that the appropriate time is given to each area as far as detection is concerned so long as the initial state covers adequate area and appropriate outer bounds are placed on the system to regulate how far the following agent is allowed to depart from the lead aircraft.

This brings up the second part of the swarming algorithm developed which is that of the bounding boxes used. There are four main bounds placed on the following agent: inner, outer, forward, and rearward. Depicted in Figure A.125, the lead agent is shown in the upper left-hand side moving in the positive *y*-direction again with a line of flight displayed with a rearwards dotted line and the perpendicular shown with a rightwards dotted line. The inner, outer, forward, and rearward bounds are each shown respectively with A, B, C, and D dimensions. This bound array sets limits on how far behind the lead agent the following agent is allowed to operate, how close to the line of flight, how far in front of the rear aircraft, and how far away from the line of flight.

The discussed swarming algorithm was tested in two stages before being fully implemented. The first was a proof-of-concept while the second used previously acquired simulation data.

The first test modeled the swarming algorithm using movement only in the positive *y*-direction of a lead agent and a following agent(s). The lead aircraft's forward velocity was ramped up in the form of a cosine to a constant velocity and then reduced back to naught with a cosine change. This can be seen in Figure A.126. All values in the first test are purely hypothetical but work well enough for proof-of-concept.

In addition to the forward velocity (y-velocity) being developed and a side-to-side (*x*-velocity) of naught for the lead agent, a constant time step of 1 second was also used for the first test. The initial state of the two agents was (2,5) for the lead agent and (4,0) for the following agent. In addition, a bound matrix of [1,10,5,20] was used following the same bound matrix format as before ([inner, outer, frontwards, rearwards]). An additional coefficient was added into the swarm function as well; this was the forward velocity buffer for the following agent. In other words, if a buffer coefficient of .8 was used and the lead agent's velocity increases 1m/s from one point to the next, the following agent's will only increase .8m/s once receiving that data. This process causes a separation distance to occur between the lead agent and the following agent with the rearward bound still ensuring that the following agent never falls too far behind. Similarly, when the lead agent slows down, the following agent will slow down at 80% of the lead agent's rate of velocity change until hitting the front bound. This allows the swarming algorithm to operate within strict limits. With a larger swarm system, differing velocity buffers and bound distances can be implemented to stagger the various vehicles.

The first part of the swarm algorithm to be tested was that of the flight path direction bounds and velocities. Since the swarming part of the algorithm only controls the perpendicular velocities of the following agent, the forward path direction needed to first be tested. To perform the tests, Figure A.127 shows the results of the *y*-direction separation between the lead and following agents at different velocity buffer coefficients. The negative sign used on the *y*-axis represents that the following aircraft is behind the lead aircraft. As can be seen, the higher the buffer value, the faster the separation increases to the rear bound. Once reaching the rear bound, the velocity is slightly increased pat that of the lead aircraft to return the following agent to be within the bounds.

Once the forward velocities were determined, the swarming algorithm itself could be implemented as well. With the entire system running properly, a theoretical test was performed with five agents (one being the lead aircraft). This can be seen in Figure A.128 where all agents start at the bottom of the figure (i.e. the lowest *y*-location) and progress according to the previous forward velocity descriptions

upwards in the figure. As the figure shows, the algorithm operates excellently with the aircrafts all coming in line when the lead forward velocity increases and then spreading apart when the velocity decreases.

The second test was to implement the theory into a code that uses simulation-generated data for the replication of the lead aircraft. A Tier 1 test was run as described in Chapter 5. The lead aircraft's flight path can be seen in Figure A.129 where the thin black line represents the overall flight path, the blue marks along the flight path indicate locations where the lead aircraft's velocity has increased, and the red marks indicate locations of decreasing velocity by the lead aircraft. Because the data used was from a true simulation, accurate time stamps could also be tagged to each data point.

With this information, a single following aircraft was initialized at a location 10m and 130 degrees from the start location of the lead aircraft. It was found that using two bound matrices allows for a smoother operation in practice where the first set of bounds is a set of close bounds and the second set is the true bound matrix. This means that minor adjustments can be made to the following aircraft if it falls outside the first bound matrix and major ones can be made if it falls outside the second. The first bound matrix used in the second test using the same format as before was [40,5,1,40] while the second was [50,1,5,50] (all values are again in meters). The adjustments determined for the following aircraft if it fell outside the bounds were as follows. First, if the following aircraft falls outside a first bound, then if the velocity adjustment determined by the swarming algorithm or buffer is going to move the following aircraft in the correct location anyways, then no changes are made to the velocities due to the bounds. If the following aircraft falls outside a first bound and the respective velocity determined by the swarming algorithm framework continues to send it the wrong direction, then a velocity change in the correct direction of 1m/s was added to the calculated velocity change. Lastly, if the following aircraft falls outside a second bound, the velocity in the correcting direction is set a 2m/s regardless of what the swarming algorithm calculations determine. While these methods could easily be improved to fit a specific situation more accurately or be changed to operate simply as a decayed change when approaching a bound, this methodology was used for the second test to determine if the theory was actually applicable at all to a real-world environment.

Calculating what the velocity changes should be from the data in each time step for the following aircraft and then what the resulting velocity and location should be for the following aircraft by the next time step iteratively, Figure A.130 was able to be produced. In the figure, the cyan markers indicate locations where the forward velocity of the following aircraft was increasing while the maroon ones indicate locations where the forward velocity was decreasing. The path of the following aircraft is that which the markers lie on while the other is the lead

aircraft's. A few observations can be made in Figure A.130. First, the locations where the following aircraft spreads out from the lead aircraft's flight path correspond well to the locations where the lead aircraft slows down. However, as the flight progresses, the locations where the following aircraft's path spreads out are slightly behind the location where the lead aircraft slows down since the following aircraft is rearwards of the lead aircraft after a little while. Also, while some locations do show the following aircraft crossing the path of the lead aircraft, this is not an issue during actual flight since the following aircraft is flown at a different altitude. Additional hardware and sensors can also be used for collision avoidance in a real-world situation, and it is unlikely that the following aircraft actually crosses the path at the same location where the lead aircraft is located. Most importantly though, the method used here allows rounded corners to be made by the following aircraft which covers a very efficient segment of area around corner waypoints.

Overall, the Bound-Continuity swarming algorithm is a very flexible model that can be used for swarming with the particular intention of increasing detection area in an accurate manner. The bound matrices can be updated based on the source being searched and the sensitivity of the detector mounted on the aircraft as well as the size of the search area and the cruise velocity of the lead aircraft.

Implementation of Swarming Algorithms (Bounded Continuity and Pattern)

Actually implementing the swarming algorithm into the real-time system was exceptionally more difficult than anticipated. Due to two main factors—radio latency and acceleration time required for the aircraft—the swarming algorithm produced results that were not quite as accurate as those found in the theoretical experiments (even those using time-accurate logged real-world data due to the acceleration time required for the following aircraft). However, this is the primary reason why a simple algorithm was purpose designed; that is: many modern complex swarming techniques are exceedingly difficult to accommodate into a real-world swarming environment with a mission-specific goal when using low-cost commercial off-the-shelf (COS) hardware.

In order to test the Bounded-Continuity swarming algorithm, it was integrated into the Tier 1 and 2 parts of the aircraft code and placed on a separate computer system. For initial testing, two one-way double radios were used in conjunction with internal radio communications (Figure A.131). This allowed for the easiest operation during testing for a double aircraft system. The Ground Station and Lead Aircraft were hosted on the Panasonic Toughbook while the Following Aircraft was hosted on a small Lenovo N22 Laptop. The radios used were XBee 900HP radio modules which were connected via raw serial protocol. In order to restrict the possibility of mixed signal transmission/receiving, the preamble value was set for

each pair of radios (the first set was set to a value of 2 while the second to a value of 3). This has the same effect essentially of setting each pair of radios to their own channel.

In addition to the hardware and software setup, the most time-consuming part of the sub-algorithm's integration with the full system algorithm was the repeated testing and adjustment of signal repetition parameters. In order to send an *x*, *y*, or *z* velocity command to an aircraft using Dronekit, MAVLink protocol is used inside a sub-function within the aircraft control code. The velocity commands require the *x*, *y*, and *z* velocity inputs for the aircraft as well as the duration that the command should be executed. Additionally, the command is recommended to be sent multiple times to the aircraft to ensure the integrity of the command communication. However, this means if the command is received multiple times in a loop, then the realized duration of the command is the input duration times the number of times received by the aircraft. Combining this with the need to compensate for the acceleration times of the lead and follower aircrafts, even with ideal communications between the algorithm and the flight controller, the execution of the commands was exceedingly difficult.

The most difficult part of transferring the Bound-Continuity algorithm from theory to real world implementation was that of the radio lag. The XBee radios are—as mentioned—very reliable and rugged; however, their operability is slightly hampered by the speed of the onboard firmware. While the data rate is adequate (up to ~200kbps), the XBee radios used required an exceptional amount of time to initialize and turn off each time a signal was sent/received. Since the PADUA algorithm must work with a burst method where radios do not necessarily only talk to one another but must also be able to communicate to other vehicles, a radio connection cannot just remain on the entire time. Thus, while the data transfer itself was very quick, the delay caused by the radios could sometimes be upwards of half a second to a full second. Thus, while acceptable for testing as part of this project, the XBee radio communications should be replaced in later implementations of PADUA with other faster forms of hardware communications. Lastly, although the results of the Bound-Continuity algorithm do not appear to have worked as well as the theoretical results, it should be remembered that the final result from the theoretical experimentation was based off of real-world logged data. Therefore, although there were some issues in the implementation of the algorithm in the simulations, with faster hardware the theoretical results using logged data prove that the Bound-Continuity algorithm is viable.

Full Algorithm Multi-Aircraft Experiment (Full System Test)

Single Tier Testing

For the final tests, swarming flights were performed with single tiers (Tier 1 and Tier 2) before being tested with multi-tier flights. This was done to ensure the system still operated properly after conversion from single to multi-aircraft. Figure A.132 shows the flight path of the lead aircraft and a single follower aircraft for a hypothetical Tier 1 application using the Bounded Continuity Swarming Algorithm. By using this swarming algorithm, the follower aircraft is able to cover more area in regions where the lead aircraft has slowed and thus covering more area while maintaining a swarming system. While the results are not quite as attractive as those found with the theoretical testing, the basis of the Bound-Continuity algorithm can still be seen with the following aircraft spreading out around corners (since this is where the lead aircraft slows down). Note that there are no mid-leg waypoints in this example. The spread of the system (i.e. how far the follower aircraft moves away from the lead aircraft flight path during deceleration areas) can be regulated or modified significantly by altering the bound values.

When used with Tier 1, the swarming algorithm can allow wider spacing of the lead aircraft's flight path. Note that in Figure A.132, the *x*-spacing for the lead aircraft was 250m instead of the usual 100m or 150m used. This wider pattern is still effective since the following aircraft should be able to make up the difference for area coverage. This is demonstrated by the count rates obtained by each aircraft from Tier 1 flights as seen in Figure A.133 where the following aircraft's maximum count rate can be clearly seen to spike and locate a hot spot even though the lead aircraft never was able to locate a significantly raised count rate.

Applying both the Bounded Continuity algorithm (Figure A.134) and the pattern-swarming method to the system (Figure A.135) on a Tier 1 search space more similar to previous tests, a few different characteristics that were very different from the previous single aircraft tests became evident. First, although the swarming algorithm can save time due to its inherent nature of covering more area, with larger bounds (as might be desired for a Tier 1 flight), the flight becomes almost overly-erratic. It can be expected that with decreased freedom (i.e. a smaller bound region) such as in the earlier example, the flight path would be much less erratic, but this would then decrease the ability of the system to operate in large urban areas without far more collision avoidance mechanisms. However, the pattern-swarming system operates exceptionally well on a large region although the middle regions are not covered quite as thoroughly as with the small-bound algorithm-based method.

In addition to the flight path itself, another important aspect that was found was that of the detection ability of a multi-aircraft system. Remembering that there are

certain principles this experiment must follow, accuracy does not necessarily increase with a multi-aircraft system. Rather, due to the system needing to be able to operate with multiple types of detectors (for example on one aircraft a linear focused PMT might be utilized with the detector compared to a box-grid PMT on the other), it cannot be assumed that count rates detected on one platform will be comparable to another. If they are comparable, then more options unfold such as being able to use only the hot spot calculated the corresponds to the highest count rate; however, without this assumption, the best way in which to determine the correct hot spot location given multiple aircraft is to merely average the hot spot locations. If one detector is more trustworthy or sensitive than the other, weights can then be easily incorporated into the hot spot determination using this methodology. However, this presents a rabbit hole where nearly infinite possibilities of calculation of hot spots can exist and thus should be an entirely different project. Instead, although a multi-aircraft system used with PADUA cannot be expected to yield more accurate results compared to a single aircraft system, the search time can be significantly decreased since a wider leg separation distance can be used than with a single aircraft system.

This being said, Tier 2 tests showed similar results as Tier 1. The pattern swarming results (Figure A.136) showed that for a cluttered search environment, the following aircraft remains close to the lead aircraft's flight path. However, since the area is assumed to be significantly smaller in a Tier 2 search, the slightly erratic appearance of the following aircraft's flight path under the influence of the Bound-Continuity algorithm (Figure A.137) is far less of an issue and still accomplishes the same goal of optimizing the searched area given time. It can be generally concluded then that, although pattern-swarming appears well for both Tier 1 and Tier 2, with a highly urban environment or a complex search area, pattern swarming is better suited. However, if the search area is more open (i.e. lacks many obstacles such as large buildings, telephone wires, etc., or it is a smaller region, then the Bound-Continuity swarming algorithm is better suited since a seemingly erratic flight path is acceptable in such a situation. Continuing with these conclusions, it can further be stated that, in many generalized cases, it is likely that pattern swarming is better to be used for Tier 1 while the Bound-Continuity method is better suited for Tier 2 unless the Tier 1 area is very open. Although Tier 2 is designed to be a smaller region than Tier 1, this does not guarantee that there will be fewer obstacles in general. A cautious assumption can be made that the area in Tier 2 will usually be empty enough to be able to implement the Bound-Continuity algorithm and in the real world, there should be enough collision avoidance sensory equipment onboard the aircraft for Tier 2 that even a few obstacles in the area should be able to be handled by the aircraft.

The statement that the Bounded-Continuity method is more suited for Tier 2 searches whereas the Pattern swarming method is better for Tier 1 searches (assuming the area is not open enough to use a larger leg separation than usual)

is reiterated by the hot spot results from the previous single tier tests. These are outlined in Table B.13 where it can be seen that the accuracies of the determined hot spots are very similar for both the Bound-Continuity and pattern swarming methods in Tier 1. However, for Tier 2, the hot spot error is measurably lower for the Bound-Continuity search method compared to that of the pattern swarming search.

The hot spot found for Tier 2 in each instance would then be used for a Tier 3 search later, but since the Tier 3 search is only designed to be single aircraft-capable, there is no need to show further testing results here. Multiple full search examples are shown in later sections that include Tier 3 results.

There are numerous variables that can be altered within this system purposely since it is designed to be as near detector-agnostic as possible for both radiation search and—if desired—for other types of search or targeting. With the multi-aircraft system, new variables are introduced that can be changed depending on how many aircraft are in the system, the speed of the aircraft(s), the altitude of the aircraft(s), and the size of the source and responsiveness of the detectors. Some of these variables include the inner and outer bound arrays used for the Bounded Continuity method, the altitude separation between the aircrafts, the target distance and bearing used for pattern methods, the velocity damper used for the Bounded Continuity method, and even the groundspeed of the lead aircraft itself can all make major differences on how a system operates. While this project sets up the overall system with initial parameters and options, for a specific mission the algorithm would be optimized if each of these variables was considered against all the others. To approximate these optimizations via simulation, it is likely that implementation of artificial intelligence (AI) would be the best option.

Triple-Tier Testing with Pattern Swarming

This section describes the full PADUA system test involving multi-aircraft pattern-based swarming search. Table B.14 displays the input variables for the simulation system. In Table B.14, the first column signifies the tier to which the variable of the same row corresponds. The value and units of the variable are shown in the fight-most column. In the simulation, two aircraft are used to test the algorithm (i.e. one lead and one following aircraft).

The results from this test are shown in Figure A.138 where the lead aircraft's flight path is shown in blue, the follower's in red, and the source location signified with a yellow star. As can be seen in the figure, the pattern search is exceptionally hectic for the Tier 2 search while following primarily the lead aircraft's flight path for the Tier 1 search. In Figure A.139, the paths are separated by color to represent each tier's path. The solid lines show the path of the lead aircraft while the dotted lines

show the path of the follower aircraft. Note that there is only a Tier 3 search pattern for the lead aircraft as it is a single aircraft search tier.

In Figure A.140, the locations of the hot spots for the simulated system are displayed. For the follower aircraft, the locations are signified with a round marker while for the lead aircraft they are shown with a triangular one. The combined hot spot for the tier (which is used for the reference start of the next tier) is shown with a star. Tier 1 locations are shown in blue while Tier 2 Locations are shown in red. The source location is shown with a yellow star. Figure A.141 shows the zoomed in version of the diagram with only the Tier 1 locations, and Figure A.142 shows the same for the Tier 2 locations.

While the Tier 1 and Tier 2 searches have definite beginnings and endings, the Tier 3 search is designed to be capable of continuous search and refinement. As mentioned before, this assists with decreasing the error of a stationary source's location, ensuring that the source is in the vicinity, and increase the ability to track a slow-moving source. There are three methods used for the Tier 3 search (see Chapter 5: Tier 3 Experiments for more information). While the moving average is primarily used for indicating the likelihood of a source having been located, the normal and weighted average methods can be used to indicate the exact source location estimated by the RAT method in Tier 3. Note again that while the weighted method can be used in simulation, it is recommended that the normal method be relied on more heavily for real-world usage unless limits are placed on the detector output to avoid outlier affects that are liable to skew the source location estimate. The progression of each of the three method results in terms of their error can be seen in Figure A.143. As would be expected, the weighted method yields the best result.

Figure A.144 shows the progression of the latitude and longitude refinement from Tier 3. During real-world application, the slope history of the moving average can be used to determine when the search is complete. Depending on the mission needs, the continuously updated latitude and longitude can be sent to additional ground/air agents to interdict the source.

The information displayed in the previous figures can be seen in more detail in Table B.15 where the latitude and longitude are shown for each hot spot in addition to the distance that the point of interest (POI) is from the source. As can be seen in the table, the further along in the tier sequence, the closer the estimate is to the source's actual location. This means that the PADUA system operated exactly as it was designed; since a very large area search was refined with optimized search method increments rather than with a single method, PADUA is able to locate a source much faster than other systems and with less manpower.

Note that, because the Tier 2 hot spot was exceptionally far from the source (14 meters), the Tier 3 search was still being refined when the data was cut off as seen

earlier in Figure A.143. This was done to allow for direct comparisons between the two types of swarming methods. If the refinement process was allowed to continue, then the end refinement result would be very similar to that which is normally found. This can be seen in Figure A.145 where the Tier 3 search was run for an additional 12 minutes (28 minutes for Tier 3 rather than 16 minutes). The result yielded a normal average result with an error of 4.106 meters from the source and a weighted average with an error 0.7711 meters from the source.

The last important part of the simulation results to show is the time required for each tier and the combined full system search. Table B.16 shows that the time for Tier 1 is indeed the longest (as it should be) with the entire system requiring less than an hour to complete the search of a ~1km^2 region. Note that the length of time for the total system to complete is longer than for the single aircraft instances for two reasons. First, in order to allow the follower aircraft to fly faster than the lead aircraft when necessary (i.e. when the follower aircraft falls behind the rear bound with respect to the lead aircraft), the lead aircraft's cruise speed for Tier 1 had to be dropped from 15m/s to 10m/s since Pixhawk has a speed limiter placed in its firmware of 15m/s. Secondly, the path has approximately the same dimensions as that of the earlier single aircraft tests. A flight sequence with multi-aircraft control with larger leg-separation would drop the time required.

Triple-Tier Testing with Bound-Continuity Algorithm Swarming

This section shows the same information as the previous one but for the full test that utilized the Bound-Continuity swarming algorithm for the multi-aircraft control rather than the pattern-based swarming method. The dimensions and variable inputs are shown in Table B.17.

Once again, the flight paths are shown in for each of the aircrafts in Figure A.146 and are shown in Figure A.147 with the tier breakdown. Note that although the swarming mechanism allows the follower aircraft to cover a great deal more area than the same system with only the lead aircraft would, it also finishes its path in approximately the same location at the same time as the lead aircraft. This is the key to allowing a fully operable swarming algorithm method to influence the detection efficiency of such a mission.

In Figure A.148, the hot spot locations for each of the tiers are shown with Figure A.149 displaying the zoomed in version of the Tier 1 results and Figure A.150 showing the same for Tier 2. In these, it can again be seen that averaging the hot spot locations for each of the tiers to develop the overall tier's hot spot does not promise a more accurate estimate, but it does ensure a reduction of false positives and noise. With more aircraft, the final estimate should be more accurate than with only two aircraft. Additionally, with numerous aircraft, it would also be possible to remove outliers with respect to the calculated overall hot spot location based on either a radial threshold or a radial max-reduction.

Although the moving average has not yet leveled out in Figure A.151 at the end of the shown data, it is continually increasing in accuracy which means that a source estimate is refined. In order to keep the number of data points equal with the previous test though, the *x*-axis is cut off in Figure A.151. Note that the accuracy for the weighted method is less than 5 meters from the source's actual location.

The refinement progression of the latitude and longitude of the source location estimate are displayed in Figure A.152 where again it is shown that the refinement is yielding a true positive for the source location since the moving average is nearing the normal and weighted averages for both the latitude and longitude values.

Table B.18 shows the refinement process just as was shown for the pattern-based swarming simulation. Note difference in source locations estimates between the two aircraft in Tier 1 and Tier 2 is mitigated by the effect of the averaging method applied to the two sets of data. In Tier 1, the lead aircraft's estimate was further from the source than the follower's aircraft while in Tier 2 the situation was flipped. If the location of the source is not known though (as in the real world), then there is not way to know for sure which aircraft's estimates are more accurate. Instead, it is better to accept a generalized estimate refinement process without worrying which agent is yielding a better result and instead simply examine the results of the whole.

Lastly, the time required for the system's running can be seen in Table B.19. The total time required is very similar to that of the pattern-based search method with only a few minutes different from the total. However, the slope of refinement progression from Tier 1 to Tier 2 is much steeper for the Bound-Continuity algorithm results since the follower aircraft is more effective in its coverage than in the pattern-based swarming.

Triple-Tier Testing with Combination Swarming Algorithms

The last multi-aircraft rural/generic test consisted of a combination swarm algorithm. Due to the need for the PADUA system to operate well in urban environments, upon observing the results from the Bound-Continuity multi-aircraft test it was found that the Tier 1 flight path spread between the lead and follower aircrafts was too large to be effective in a dense urban environment with a large number of obstacles. While the bounds could be constrained further to restrict the Bound-Continuity algorithm further, due to the limitations imposed by the low-rate communications, there would be little difference between the Bound-Continuity algorithm and the simpler pattern-based swarming method that only uses the target heading and distance from the lead aircraft to determine the next location target of the follower aircraft. While the advent of other communications methods might allow for increased operability of the Bound-Continuity algorithm in future

use, this test was intended to give an alternative system with optimized swarming methodology for Tier 1 and 2 rather than using the same method for both. Thus, for Tier 1, pattern swarming is utilized, and for Tier 2, Bound-Continuity swarming is implemented. The basic inputs for the system can be seen in Table B.20.

Figure A.153 and Figure A.154 each show the flight path portions for the test. As can be seen, the follower aircraft follows the lead aircraft much more closely for the majority of Tier 1 compared to the Bound-Continuity test. Similarly, the flight track for Tier 2 is spread out according the governing equations for the Bound-Continuity algorithm.

The error for Tier 3 can be seen in Figure A.155, and the refinement process for the location parameters for Tier 3 is displayed in Figure A.156. It is evident from the figures that the accuracy is not quite as high for the combination method as the single swarm methods since the Bound-Continuity algorithm's characteristic of covering more area would of course be more helpful for Tier 1 than Tier 2 (although it also assists the search process in Tier 2). However, in real world applications, sometimes accuracy must be sacrificed for functionality. Thus, this test shows what the most likely real-world rural/generic result should be expected.

The final results for the combined swarming test can be seen in Table B.21.While the distance from the hot spots to the source still decrease as they should from one tier to the next, the final distances are noticably further from the source than the single swarm algorithm tests. As mentioned, this is due to the optimum swarm methods not being used for their appropriate tiers in favor of applcability. Thus, if this type of combined method was instituted in a real world enviornment, then the length of time required for Tier 3 to operate should be higher than for single swarm method instances. Even so, in less than an hour, the algorithm was still able to locate the source to within just over 3 meters. Table B.22 shows that the time required for each tier and the entire system to run was very similar to the other multi-aircraft swarm tests.

Urban Tests of the Complete PADUA System

This section covers the results of more realistic simulations compared to those used in the primary development phase of the project to test the robustness of the PADUA algorithm. There are three tests discussed. All of them are centered on Manhattan-based search situations (the parts of the borough used for this testing is shown using Google Earth in Figure A.157). The first test consists of a Midtown Manhattan single aircraft search, the second test consists of a Lower Manhattan single aircraft search, and the last consists of a Lower Manhattan multi-aircraft search. The PADUA search operates in each of these searches with a predetermined Tier 1 lead aircraft search while the other flight paths simply follow the controls created by the PADUA algorithm. Obstacle avoidance is not included

as part of the PADUA algorithm, and it can be seen in some of the examples that an aircraft's track goes through an object. It is understood that obstacle avoidance equipment and software would be included on the vehicles in a real-world application.

East Village Search Test

This test consisted of a full PADUA test using a single aircraft with a takeoff location in Williamsburg, Brooklyn and then flown through Midtown and parts of East Village/Stuyvesant Town. The source was located in Union Square. This area is of particular interest for nuclear security due to the high number of medical facilities in the area (which tends to mean that a high number of radioisotopes are being used).

The Tier 1 path used consisted of a list of waypoints generated manually using ArduPilot Mission Planner. The Tier 2 and 3 portions are then performed using the same methodology as shown in previous generic examples. Note that in the real world it would be expected that obstacle avoidance systems would be in place onboard the aircraft. This would cause the flight paths for Tiers 2 and 3 to differ from those shown in the example results (Figure A.158, Figure A.159, and Figure A.160). As can be seen in these figures though, the results are exceptional and prove that the PADUA system is applicable to a real world situation. Table B.23 shows important results from the example simulation. Although the Tier 1 and 2 hot spot locations were not exactly close to the source, the Tier 3 method was still able to locate the source to within nearly 1 meter using the normal average RAT method and nearly a quarter of a meter with the weighted average RAT method. This is the equivalent of just over 10 inches. Furthermore, the entire search required less than an hour to complete with a total distance traveled of 17.5 km.

Lower Manhattan Search Test

The Lower Manhattan search is perhaps the most important experimental test presented in this project since it is the most likely applicable scenario in which PADUA would be used. There are numerous difficulties when flying small aircraft in heavily urban environments. From radio transmission losses to the issue of a large number of building obstacles that must be avoided, there are several aspects of real-world flight that would need to be further refined and tested prior to final application for a fully autonomous system.

Some parts of the flight track can be seen in Figure A.161 and Figure A.162 where the Freedom Tower and the memorial for the World Trade Center attacks can be seen. While security is still heavy in Lower Manhattan, the location remains a top target among enemies of the U.S. due primarily to the vast wealth that moves through the financial centers there. This includes primarily the New York Stock

Exchange (with a market cap of nearly $23 trillion as of 2019[105]), although the law, trading, and investment firms and banks comprise a significant amount of wealth and capitalist symbolism as well.

One example of the highly increased difficulty in urban compared to a generic or rural search can be seen in Figure A.163 where the track of the aircraft can be seen flying northwards on Broadway through the 'concrete canyons' of Lower Manhattan. Careful testing and redundancy must be included in a final system to allow such a situation to function fully autonomously.

The full flight path for the single aircraft Lower Manhattan test can be seen in Figure A.164. The flight started at the police station at the Manhattan-side of the Brooklyn Bridge and ended near the Federal Reserve. The path and source location can be seen with a blank background in Figure A.165. Again, it can be seen plainly that the algorithm shifted search tiers successfully through Tiers 1, 2, and 3.

The results from the test can be seen in Table B.24 where again the efficiency of the PADUA system can be seen clearly. With the hot spot refinement moving from over 100 meters to less than 2 meters in under an hour, the PADUA algorithm was successful in locating the source in simulation with just under 8.5km flown.

Multi-Aircraft Lower Manhattan Search

The final test for the Manhattan-based search tests with the PADUA algorithm uses the same situation as the single aircraft Lower Manhattan search test but with two aircraft instead of one. This test serves to show the capabilities of the multi-aircraft portion of PADUA in a near-real-world simulation. Additionally, although this test did show that multi-aircraft capabilities function as designed in PADUA, the primary advantages come from the ability to spread out the flight paths and cover an increased amount of search area per unit time. If the environment is heavily urban such as the test shown here in Manhattan, then it is likely that multiple small sub-swarms would be more effective than a single large swarm since there is the bounds for the follower aircraft(s) are so restricted due to the number of building obstacles in place. The one exception would be if the target source was small or well-shielded in which case a number of aircraft in close proximity should be able to locate the source more easily.

Figure A.166 shows the constraints that buildings place on an active swarm in an urban environment as well as the staggering of heights used in the multi-aircraft tests. In the figure, the Freedom Tower is shown with the aircraft tracks moving southwards on West St. towards the tip of Manhattan. The lead aircraft's path is shown in blue and the follower's in red. Additionally, it can be noted that the lead aircraft's cruise altitude is above the follower's in order to operate each aircraft on its own plane; this is done to reduce the chance of vehicle-to-vehicle collisions.

The top-down view of the flight paths can be seen in Figure A.167 where the lead aircraft is again shown in blue and the follower in red. The same flight path sequence is displayed in Figure A.168 without the satellite imagery. It can be seen especially in Figure A.168 that the Tier 3 portion of the system centers rapidly around the source's location.

The primary results from the test can be seen in Table B.25 where the total distance traveled is shown to be over 10.5 km and the total time to be just under an hour. Although the initial hot spot location was not as accurate the single aircraft one, the Tier 2 hot spot could be seen to be very low in relation to the first due to the multi-aircraft bound-Continuity swarming methodology. From this, it can be seen that, if the environment is too urban for Bound-Continuity swarm algorithm usage, then the follower aircraft's data is liable to be more of a hindrance than an advantage and could possibly be ignored if so desired. However, the advantage that the follower aircraft with the Bound-Continuity swarming algorithm yields in Tier 2 is significant. The only advantage of having additional aircraft in the Tier 1 search in this type of environment is to increase the chances of the source being located if it is a difficult-to-locate target.

Conclusions

While there is little different between the New York tests and the rural/generic tests used for development in earlier chapters, the New York tests still showed several important aspects of the PADUA algorithm. Designed with an adjustable Tier 1 path that does not need to be symmetrical, PADUA is able to operate well in a realistic environment with a preplanned waypoint-determined Tier 1 path. Furthermore, although the distance between the nearest sample location and the source location matters more than any other aspect for the success of the algorithm, even with a minimum distance of over 100 meters from the nearest Tier 1 location the algorithm was still able to locate the source to with a few meters or less using the combined tier search methods.

While full autonomy in a dense urban environment would require additional obstacle avoidance hardware and software, these tools already exist and should be able to be integrated into the system easily due to the object-oriented approach taken during the development of PADUA. The search and primary control parts of the system though (which are the parts that PADUA operates) were proven to work extremely efficiently with minimal errors during the simulated missions shown in the examples.

CHAPTER 7

FINAL CONCLUSIONS AND RECOMMENDATIONS

The experiments and their results shown in this document show that PADUA is an applicable and viable non-directional detector search solution for real world applications. Although there are a few issues that would need to be refined in prototype development, PADUA as a basis algorithm performs even better than was expected. Through single tier and multi-tier experiments, the developmental process for the algorithm was performed incrementally so that the exceptionally complex PADUA system could be built ruggedly and remain as versatile as possible. There were multiple instances where it was shown that this was in fact the case in the final algorithm. Additionally, some very important aspects of swarming implementation were found during multi-aircraft tests.

First, although it can be confidently stated that the PADUA algorithm was exceedingly accurate and successful in its operation with respect to the scope of its design, it becomes much more difficult to conclude exactly how accurate it is compared to other modern radiation detection systems. As mentioned, this difficulty arises from the fact that PADUA is simply not based on a single other search system or research effort. Although the mechanism devised and tested by the University of Bristol uses a relatively similar structure, the overall algorithm design, search methods and their sequence, control methods and their sequence, communications architecture, and primary intended mission are all still different. The primary similarity between the system developed at the University of Bristol and PADUA is the multi-tier approach used to refine hot spots consecutively. In addition, both systems use Bayesian statistics at least for part of the search mechanism.

The multi-tier approach was used in PADUA to increase the search speed such that only the locations of highest probability are searched finely rather than the entire initial search area. While all of the search times and accuracies are extremely situationally dependent, the advantages of using multiple search tiers is obvious when considering the test data. Take for example Table B.22 where the Tier 1 search consisted of an area coverage of approximately 1 km^2 and Tier 2 of 400 m^2. Including the difference of flight leg separation, in the tests shown the Tier 1 path covered a linear distance approximately 6.5 times longer than the Tier 2 path. If the Tier 1 search was flown in the same manner as Tier 2 such that the flight legs had the same separation as Tier 2, the total distance covered would be approximately 18.2 times longer. Furthermore, if both the Tier 2 path design and average flight speed were applied to a Tier 1 search over a total area of approximately a square kilometer, then the total time to perform merely the Tier 1 search would be nearly 11 times longer. In other words, instead of Tier 1 and Tier 2 combined requiring 30 minutes as they did in the test, using only Tier 2 methodology for Tier 1 would require 206 minutes or 3.4 hours. Thus, using the

hot spot refinement method is indeed appropriate and useful if time is a strong consideration.

So far as the third tier's efficiency and accuracy was concerned, since this Tier was progressive, the best method of determining the accuracy should also be a progressive one. Thus, because the error refinement has a periodic and exponential die-off (Figure A.107), two best-fit equations were produced to represent the general trends of the range of error. In other words, given the iteration of the Tier 3 refinement process and using the same situation variables as in the testing, the error should be within the confines of Eq. 7.1 and Eq. 7.2 where y_{min} represents the lower error estimate and y_{max} represents the high error estimate. In Figure A.169, these two curves are represented as the "Min Exponential" and "Max Exponential" curves.

$$y_{min} = 0.5178571 + (34.53051 * e^{-0.04163289*x}) \quad \textbf{Eq. 7.1}$$

$$y_{max} = 1.937254 + (31.69765 * e^{-0.02155485*x}) \quad \textbf{Eq. 7.2}$$

Both from the raw Tier 3 error data and the two equations just described, it is obvious that the situation that the PADUA system is applied to will govern the length of time required for the third tier's analysis to run. For example. In the event of a large stolen source that is able to be located using the compete PADUA system, it is likely that as soon as Tier 3 begins, additional ground forces will be sent to the location that Tier 3 is operating whereby the estimated location will be continually relayed. Because the error refinement slows down at an exponential rate, after only one or two perimeter laps, the Tier 3 refinement process can most likely be determined to be accurate enough to send additional ground assistance to the target location. Note that the normal progressive perimeter averaging method was used for the development of Eq. 7.1 and Eq. 7.2.

Thus, it can be concluded that the search methodologies used in PADUA overall were exceedingly accurate and successful so far as the scope and intentions of the algorithm was concerned. While again, the errors, accuracies, and search times are extremely dependent upon the mission's exact situation, the test situations all showed a large improvement over any similar systems currently available.

The second half of the development of PADUA consisted of implementing multi-aircraft capabilities from which multiple conclusions were drawn. First, it could be seen from comparing the Lower Manhattan search data as seen in Table B.24 and Table B.25 that the use of multiple aircrafts does not necessarily speed up a

search. This is primarily due to the fact that flight controllers (especially low-cost ones) often have speed limitations placed on them. Thus, to allow follower aircraft(s) to be able to increase their speeds greater than that of the lead aircraft if required to move to within the distance bounds, the cruise speed of the lead aircraft must be lowered when implementing a swarming mechanism to a speed lower than the maximum. Thus, the swarm's overall average speed will be lowered as well. Another aspect that could be seen in the Bound-Continuity tests was that the latency of the communications hardware must be exceptionally low if using a swarming system that relies on an algorithm any more complex that a fundamental BOIDS system. Even with an algorithm simplified as much as the Bound-Continuity method, wireless hardware latency proved to be a severe issue. Furthermore, without being able to make any assumptions as to the accuracies of each aircraft's onboard detector(s) that the count rates are gathered from, only a simple average was able to be used to develop a tier's hot spot from the individual aircrafts' hot spots. This means that there is no increase in accuracy ensured by a multi-aircraft approach in a small swarm. However, it could be concluded that the chances of finding a source are considerably improved with a multi-aircraft system. Attempting to place an exact probability on the increase in locating the source is pointless though since again, the case is extremely situationally dependent. For a rural environment, large spread out pattern swarming system has obvious advantages since smaller leg separations can effectively be performed in the same amount of time as the original single aircraft search type. However, with an urban environment, spread is severely limited by the objects in the environment. Thus, although the chances of locating a source are indeed higher with a multi-aircraft system, it is next to impossible to measure exactly how much higher due to environmental, source, and shielding specifics.

For future work, implementing PADUA onto an aircraft(s) for real world testing would require the integration of sensory equipment and obstacle avoidance methods. While many of these are already available, part of the integration process will involve prioritizing collision avoidance over PADUA algorithm control commands within the code itself. These subsystems would likely include—but not be limited to—LIDAR, ultrasonic detectors, high-precision GPS units, and IR detection. However, due to the object-oriented construction of the PADUA algorithm, joining collision/object avoidance mechanisms and the search/control algorithm that PADUA provides should be a fairly simple manner so far as the alterations and updates to the control code are concerned. Testing the modified system should be performed with low-cost aircraft first in stages of increasing complexity just as the development of PADUA was done in simulation form. Any type of testing involving aircraft always leaves little leeway for mistakes on the control code side since bad inputs into the flight dynamics of an aircraft will often lead to severe damage or total loss of the airframe. While mistakes and difficulties are expected in research, mitigating the cost should also be a constant consideration.

Additionally, the individual tier methods can also be developed further in the future. For example, Tier 1 further refinement might include urban path optimization and studies in the advantages/disadvantages if implementing an increasingly autonomous control method within the code. Future work for Tier 2 should include testing of variable path types, and Tier 3 work should incorporate other geometries other than rectangular for the RAT method. While there are of course numerous other details that can be explored further in depth when considering the individual tier control processes, these are a few of the most important. As far as the search side is concerned on Tiers 1 and 2, one additional code object that should be written and tested is a hot spot mapping option. While radiation mapping has been performed on numerous occasions at this point, it is still an important alternative to the methods developed specifically for PADUA that should be made available to the human operator. Again, while PADUA is designed to be extremely versatile and updatable, it is designed primarily with urban/constrained environments in mind. While the same system can be used in a rural environment, in the case of fallout analysis, radiation mapping is still likely to be needed. PADUA will be able to likely locate the most severe hot spots quicker than a mapping system, but if a mapping system can be run simultaneously with the PADUA algorithm, then additional flights would not be needed in areas already searched.

An additional point of further research that should be explored is that of environmental alterations involving the source flux and detector specifics. In this project, each tier was refined to an optimized variation using the uniformly shielded simulated source; however, the comparison of these results to other types of sources and non-uniformly shielded sources is an important future research topic. It would also be helpful to develop and catalog truth-value representations for different types of detectors; this would assist in increasing the abilities of multi aircraft Bayesian-based searches and combined tier result weights. For example, if it is known that two of the same detectors are being used for a double-aircraft system, then the option should be available in the code to input this information at the beginning thus allowing the aircrafts' hot spots to be combined with a weighted average of their locations based on corresponding weights rather than simple averages (at least for Tier 1 where simple maximums are used with corresponding indexed locations and count rates). Applying this same information to Bayesian statistics, a fractional weight could be applied to the resulting statistics for each aircraft if there was—hypothetically—a more efficient detector on one aircraft than on anther. However, as mentioned already, PADUA currently does not make any assumptions on detector types, efficiencies, or accuracies.

Overall, though, PADUA met and exceeded expectations. The ability to pinpoint a source autonomously in less than an hour from a square kilometer area is a new ability for nuclear security and cleanup missions. Furthermore, PADUA was written in such a way that the resulting algorithm requires minimal computing ability and

can thus be run on a generic laptop computer. As the enemies of the United States and democracy become wiser, we must stay ahead of them; PADUA accomplishes this and sets a new standard for radiation detection systems.

BIBLIOGRAPHY

[1] “Information on More than 180,000 Terrorist Attacks,” *County-Level Correlates of Terrorist Attacks in the United States*. [Online]. Available: www.start.umd.edu/gtd/.
[2] “Budget Overview: Congressional Justification,” United States, Department of Homeland Security, Domestic Nuclear Detection Office, 2018.
[3] D. A. Shea, J. D. Moteff, and D. Morgan, “The Advanced Spectroscopic Portal Program: Background and Issues for Congress.” United States, Congress, Congressional Research Service, 2018.
[4] S. Wilmon, “A Bayesian Approach to Broad-Area Nuclear and Radiological Search Operations,” University of Tennessee, 2014.
[5] R. B. Wilkerson, “A Bayesian Approach to Aerial Localization of Radioactive Sources,” University of Tennessee, 2016.
[6] J. Rogers, “The Origins of Drone Warfare,” no. April, 2018.
[7] “Kettering Aerial Torpedo ‘Bug,’” *National Museum of the US Air Force*, 2015. [Online]. Available: https://www.nationalmuseum.af.mil/Visit/Museum-Exhibits/Fact-Sheets/Display/Article/198095/kettering-aerial-torpedo-bug/. [Accessed: 12-Sep-2019].
[8] A. Finn and S. Scheding, “Chapter 2: Background,” in *Developments and Challenges for Autonomous Unmanned Vehicles: a Compendium*, Springer, 2010, p. 8.
[9] “German V-Weapons: Desperate Measures,” *National Museum of the US Air Force*, 2015. [Online]. Available: https://www.nationalmuseum.af.mil/Visit/Museum-Exhibits/Fact-Sheets/Display/Article/196145/german-v-weapons-desperate-measures/. [Accessed: 12-Sep-2019].
[10] “Republic/Ford JB-2 Loon (V-1 Buzz Bomb),” *National Museum of the US Air Force*, 2015. [Online]. Available: https://www.nationalmuseum.af.mil/Visit/Museum-Exhibits/Fact-Sheets/Display/Article/196227/republicford-jb-2-loon-v-1-buzz-bomb/. [Accessed: 12-Sep-2019].
[11] A. Calder, *The People’s War: Britain 1939-1945*. Pimlinco, 1969.
[12] R. Guttman, “Drones: The Hollywood Connection,” *Aviation History*, pp. 48–53, 2016.
[13] J. R. Angolia, “TOW Engagement in the Active Defense -- 3000 Meters or Less,” 1978.
[14] “The US Navy -- Fact File: AIM-54 Phoenix Missile,” 2017. [Online]. Available: https://www.navy.mil/navydata/fact_display.asp?cid=2200&tid=700&ct=2. [Accessed: 12-Sep-2019].
[15] “Tomahawk cruise missile,” *Raytheon*. [Online]. Available: https://www.raytheon.com/capabilities/products/tomahawk. [Accessed: 12-Sep-2019].
[16] “Tomahawk cruise missile,” *Encyclopaedia Britannica*. Encyclopaedia

Britannica, Inc., 2016.
[17] “The US Navy -- Fact File: Tomahawk Cruise Missile,” 2018. [Online]. Available: https://www.navy.mil/navydata/fact_display.asp?cid=2200&tid=1300&ct=2. [Accessed: 12-Sep-2019].
[18] J. P. Irani, Geoffrey B., Christ, “Image Processing for Tomakawk Scene Matching,” *Johns Hopkins APL Tech. Dig.*, vol. 15, 1994.
[19] R. Whittle, “The Man Who Invented the Predator,” *Air & Space Magazine*, Apr-2013.
[20] K. Neubauer, Martin, Günther, Georg, Füllhas, “Structural Design Aspects and Criteria for Military UAV,” 2007.
[21] W. E. Leary, “High-Tech Suits Help Pilots Avoid Gravity’s Perils,” *The New York Times*, 2000.
[22] “General Atomics Aeronautical Systems RQ-1 Predator,” *National Museum of the US Air Force*, 2015. [Online]. Available: https://www.nationalmuseum.af.mil/Visit/Museum-Exhibits/Fact-Sheets/Display/Article/196333/general-atomics-aeronautical-systems-rq-1-predator/. [Accessed: 12-Sep-2019].
[23] G. O’Regan, *Introduction to the History of Computing*. Springer International Publishing, 2016.
[24] “Intel Timeline: A History of Innovation,” *Intel*. [Online]. Available: https://www.intel.com/content/www/us/en/history/historic-timeline.html. [Accessed: 12-Sep-2019].
[25] “Field-effect Transistor,” *The Columbia Encyclopedia*. Columbia University Press.
[26] J. Stamp, “Unmanned Drones Have Been Around Since World War I,” *Smithsonian*, 2013. [Online]. Available: https://www.smithsonianmag.com/arts-culture/unmanned-drones-have-been-around-since-world-war-i-16055939/. [Accessed: 12-Sep-2019].
[27] “HEXAGON KH-9 Reconnaissance Satellite,” *National Museum of the US Air Force*, 2016. [Online]. Available: https://www.nationalmuseum.af.mil/Visit/Museum-Exhibits/Fact-Sheets/Display/Article/195921/hexagon-kh-9-reconnaissance-satellite/. [Accessed: 12-Sep-2019].
[28] “A Futile Fight for Survival,” *Central Intelligence Agency*, 2008. [Online]. Available: https://www.cia.gov/library/center-for-the-study-of-intelligence/csi-publications/books-and-monographs/a-12/a-futile-fight-for-survival.html. [Accessed: 12-Sep-2019].
[29] P. Szoldra, “Drone Spying Capabilities Are About To Take Another Huge Leap,” *Business Insider*, 2013. [Online]. Available: https://www.businessinsider.in/Drone-Spying-Capabilities-Are-About-To-Take-Another-Huge-Leap/articleshow/21411579.cms. [Accessed: 12-Sep-2019].
[30] “History of the Digital Camera and Digital Imaging,” *Digitalkamera*

Museum. [Online]. Available: https://www.digitalkameramuseum.de/en/history. [Accessed: 12-Sep-2019].
[31] H. Maître, *From Photon to Pixel: The Digital Camera Handbook*, 2nd ed. ISTE, 2017.
[32] E. D. Kaplan and C. J. Hegarty, *Understanding GPS/GNSS: Principles and Applications*, 3rd ed. Artech House, 2017.
[33] "The World's First All Metal Aircraft – The Junkers J1," *Junkers.de*, 2018. [Online]. Available: https://www.junkers.de/the-worlds-first-all-metal-aircraft-the-junkers-j1/. [Accessed: 12-Sep-2019].
[34] "Definition of Composite," *Merriam-Webster Dictionary*. 2019.
[35] M. Endo, "Chapter 20: Carbon Fiber," in *High-Perfromance and Specialty Fibers Concepts, Technology and Modern Applications of Man-Made Fibers for the Future*, Tokyo: Springer, 2018.
[36] B. T. Clough, "Unmanned Aerial Vehicles: Autonomous Control Challenges, A Researcher's Perspective," *J. Aerosp. Comput. Information, Commun.*, vol. 2, no. 8, pp. 327–347, 2005.
[37] D. Joshi, "Exploring the Latest Drone Technology for Commercial, Industrial and Military Drone Uses," *Business Insider*, 2017. [Online]. Available: https://www.businessinsider.com/drone-technology-uses-2017-7. [Accessed: 12-Sep-2019].
[38] L. K. Hilton, M. L. Morris, and R. O. Stenerson, "The University of Utah neutrino detector: II. Large area directional Čerenkov detectors," *Nucl. Instruments Methods*, vol. 51, no. 1, pp. 43–46, 1967.
[39] G. A. Karolyi, "A Bayesian Approach to Modeling Stock Return Volatility for Option Valuation," *J. Financ. Quant. Anal.*, vol. 28, no. 4, pp. 579–594, 1993.
[40] H. Chu and M. E. Halloran, "Bayesian Estimation of Vaccine Efficacy," *Clin. Trials J. Soc. Clin. Trials*, vol. 1, no. 3, pp. 306–314, 2004.
[41] S. B. McGrayne, "The Theory That Would Not Die: How Bayes' Rule Cracked the Enigma Code, Hunted Down Russian Submarines, and Emerged Triumphant from Two Centuries of Controversy," in *The Man Who Did Everything*, Yale University Press, 2012.
[42] S. B. McGrayne, *The Theory that would not Die : How Bayes' Rule Cracked the Enigma Code, Hunted down Russian Submarines, & Emerged Triumphant from Two Centuries of Controversy*. Yale University Press, 2011.
[43] "Tritium Atom - an overview | ScienceDirect Topics." [Online]. Available: https://www.sciencedirect.com/topics/chemistry/tritium-atom. [Accessed: 12-Sep-2019].
[44] O. C. Lind, B. Salbu, K. Janssens, K. Proost, M. García-León, and R. García-Tenorio, "Characterization of U/Pu Particles Originating from the Nuclear Weapon Accidents at Palomares, Spain," *Sci. Total Environ.*, vol. 376, no. 1–3, pp. 294–305, 2007.
[45] D. Stiles, "A Fusion Bomb over Andalucía: U.S. Information Policy and the

1966 Palomares Incident," *J. Cold War Stud.*, vol. 8, no. 1, pp. 46–67, 2006.
[46] D. Pierson, "Lost in the Sky, Found in the Sea," *Nav. Hist.*, vol. 23, no. 3, pp. 50–54, 2009.
[47] K. Caudle, "Searching Algorithm Using Bayesian Updates," *J. Comput. Math. Sci. Teach.*, vol. 29, no. 1, pp. 19–29, 2010.
[48] D. Connor, P. G. Martin, and T. B. Scott, "Airborne radiation mapping: overview and application of current and future aerial systems," *International Journal of Remote Sensing*, vol. 37, no. 24. Taylor and Francis Ltd., pp. 5953–5987, 16-Dec-2016.
[49] C. J. Dahlman, "The Cost of a Military Person-Year: A Method for Computing Savings from Force Reductions," Santa Monica, CA, 2007.
[50] T. Hubert, C. Guo, C. Mouton, and J. Powers, "Tankering Fuel on U.S. Air Force Transport Aircraft: An Assessment of Cost Savings," Santa Monica, CA, 2015.
[51] C. W. Reynolds, "Flocks, Herds, and Schools: A Distributed Behavioral Model," *Proc. 14th Annu. Conf. Comput. Graph. Interact. Tech. - SIGGRAPH '87*, 1987.
[52] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," *Proc. ICNN'95 - Int. Conf. Neural Networks*, 1995.
[53] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*, 1st ed. New York: Oxford University Press, 1999.
[54] T. Stützle and H. H. Hoos, "Max-Min Ant System," *Futur. Gener. Comput. Syst.*, vol. 16, no. 8, pp. 889–914, 2000.
[55] L. M. Gambardella, É. Taillard, and G. Agazzi, "MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows," in *New Ideas in Optimization*, London, UK: McGraw-Hill, 1999, pp. 63–76.
[56] K. Dervis, "An Idea Based on Honey Bee Swarm for Numerical Optimization," 2005.
[57] P. Karber and J. Thibeault, "Russia's New Generation Warfare," *Army Magazine*, 2016.
[58] D. Reid, "Swarm of armed DIY drones attacks Russian military base in Syria," *CNBC.com*, 2018. [Online]. Available: https://www.cnbc.com/2018/01/11/swarm-of-armed-diy-drones-attacks-russian-military-base-in-syria.html. [Accessed: 12-Sep-2019].
[59] "Boeing: Autonomous Systems - ScanEagle." [Online]. Available: https://www.boeing.com/defense/autonomous-systems/scaneagle/index.page. [Accessed: 12-Sep-2019].
[60] D. Lee and D. H. Shim, "Development of Mini-Drones and Feedback Linearization Based Velocity Control for Outdoor Autonomous Swarming Flights," *IFAC-PapersOnLine*, vol. 51, no. 22, pp. 178–183, Jan. 2018.
[61] A. Mystkowski, "Implementation and Investigation of a Robust Control Algorithm for an Unmanned Micro-aerial Vehicle," *Rob. Auton. Syst.*, vol.

62, no. 8, pp. 1187–1196, 2014.
[62] D. M. Hart and P. A. Craig-Hart, “Reducing Swarming Theory to Practice for UAV Control,” in *IEEE Aerospace Conference Proceedings*, 2004, vol. 5, pp. 3050–3063.
[63] A. Chapman, “Semi-Autonomous Networks : Effective Control of Networked Systems through,” University of Washington, 2013.
[64] Q. Liu, M. He, D. Xu, N. Ding, and Y. Wang, “A Mechanism for Recognizing and Suppressing the Emergent Behavior of UAV Swarm,” *Math. Probl. Eng.*, pp. 1–14, 2018.
[65] I. Csiszar, “Information Type Measures of Differences of Probability Distribution and Indirect Observations,” *Stud. Sci. Math. Hungarica*, vol. 2, pp. 299–318, 1967.
[66] S. M. Ali and S. D. Silvey, “A General Class of Coefficients of Divergence of One Distribution from Another,” *J. R. Stat. Soc. Ser. B*, vol. 28, no. 1, pp. 131–142, 1966.
[67] T. J. Allen, “Design and Test of a UAV Swarm Architecture over a Mesh Ad-Hoc Network,” Air Force Institute of Technology, 2015.
[68] A. M. Jensen, D. K. Geller, and Y. Chen, “Monte Carlo Simulation Analysis of Tagged Fish Radio Tracking Performance by Swarming Unmanned Aerial Vehicles in Fractional Order Potential Fields,” *J. Intell. Robot. Syst. Theory Appl.*, vol. 74, no. 1–2, pp. 287–307, Apr. 2014.
[69] Congressional Budget Office, “Long-Term Implications of the 2020 Future years Defense Program,” 2019.
[70] F. P. Filbert, “Joint Integration Testing of Swarming Unmanned Aerial Vehicles,” *Defense Systems Information Analysis Center*, 2015. [Online]. Available: https://www.dsiac.org/resources/journals/dsiac/winter-2016-volume-3-number-1/joint-integration-testing-swarming-unmanned. [Accessed: 12-Sep-2019].
[71] C. Whitlock, “How Crashing Drones are Exposing Secrets about U.S. War Operations - The Washington Post,” *The Washington Post*, 2015. [Online]. Available: https://www.washingtonpost.com/world/national-security/how-crashing-drones-are-exposing-secrets-about-us-war-operations/2015/03/24/e89ed940-d197-11e4-8fce-3941fc548f1c_story.html?noredirect=on. [Accessed: 12-Sep-2019].
[72] H. H. Seck, “US Marines Recover Crashed Predator Drone During Deployment to Iraq,” *Military.com*, 2019. [Online]. Available: https://www.military.com/daily-news/2015/12/02/us-marines-recover-crashed-predator-drone-deployment-iraq.html. [Accessed: 12-Sep-2019].
[73] D. Axe, “U.S. ‘Mulled Retrieving Drone,’” *The Diplomat*, 2011. [Online]. Available: https://thediplomat.com/2011/12/u-s-mulled-retrieving-drone/. [Accessed: 12-Sep-2019].
[74] C. Whitlock, “When Drones fall from the Sky,” *The Washington Post*, 2014. [Online]. Available: https://www.washingtonpost.com/sf/investigative/2014/06/20/when-drones-

fall-from-the-sky/?utm_term=.53824740655b. [Accessed: 12-Sep-2019].
[75] K. W. Williams, “A Summary of Unmanned Aircraft Accident / Incident Data: Human Factors Implications,” 2004.
[76] R. L. Mcclanahan, “Improving Unmanned Aerial Vehicle Formation Flight and Swarm Cohesion by using Off the Shelf Sonar Sensors,” Air Force Institute of Technology, 2017.
[77] “MAVProxy — MAVProxy 1.6.4 documentation.” [Online]. Available: http://ardupilot.github.io/MAVProxy/html/index.html. [Accessed: 12-Sep-2019].
[78] L. Meier, “Introduction - MAVLink Developer Guide,” *https://mavlink.io/en/*, 2018. [Online]. Available: https://mavlink.io/en/. [Accessed: 12-Sep-2019].
[79] D. T. Davis, T. H. Chung, M. R. Clement, and M. A. Day, “Consensus-Based Data Sharing for Large-Scale Aerial Swarm Coordination in Lossy Communications Environments,” in *IEEE International Conference on Intelligent Robots and Systems*, 2016, pp. 3801–3808.
[80] H. Van Dyke Parunak, S. A. Brueckner, and J. Sauter, “Digital Pheromones for Coordination of Unmanned Vehicles,” in *Lecture Notes in Computer Science*, 2005, vol. 3374, pp. 246–263.
[81] M. Sbeiti, N. Goddemeier, D. Behnke, and C. Wietfeld, “PASER: Secure and Efficient Routing Approach for Airborne Mesh Networks,” *IEEE Trans. Wirel. Commun.*, vol. 15, no. 3, pp. 1950–1964, Mar. 2016.
[82] S. Rohde, M. Putzke, and C. Wietfeld, “Ad hoc self-healing of OFDMA networks using UAV-based relays,” *Ad Hoc Networks*, vol. 11, no. 7, pp. 1893–1906, Sep. 2013.
[83] L. Gupta, R. Jain, and G. Vaszkun, “Survey of Important Issues in UAV Communication Networks,” *IEEE Commun. Surv. Tutorials*, vol. 18, no. 2, pp. 1123–1152, Apr. 2016.
[84] R. S. de Moraes and E. P. de Freitas, “Distributed Control for Groups of Unmanned Aerial Vehicles Performing Surveillance Missions and Providing Relay Communication Network Services,” *J. Intell. Robot. Syst. Theory Appl.*, vol. 92, no. 3–4, pp. 645–656, Dec. 2018.
[85] R. Tang, S. Fong, S. Deb, and R. Wong, “Dynamic Group Search Algorithm for Solving an Engineering Problem,” *Oper. Res.*, vol. 18, no. 3, pp. 781–799, Oct. 2018.
[86] R. Tang, S. Fong, N. Dey, R. K. Wong, and S. Mohammed, “Cross Entropy Method Based Hybridization of Dynamic Group Optimization Algorithm,” *Entropy*, vol. 19, no. 10, Oct. 2017.
[87] D. Shahbazi-Gahrouei, S. Setayandeh, and M. Gholami, “A Review on Natural Background Radiation,” *Adv. Biomed. Res.*, vol. 2, no. 1, p. 65, 2013.
[88] C. E. Rasmussen, *Gaussian Processes for Machine Learning.*, vol. 14, no. 2. MIT Press, 2006.
[89] V. Nguyen, S. Gupta, S. Rana, C. Li, and S. Venkatesh, “Filtering Bayesian Optimization Approach in Weakly Specified Search Space,” *Knowl. Inf.*

Syst., pp. 1–29, Jul. 2018.
[90] “Dronekit.” [Online]. Available: https://dronekit.netlify.com/. [Accessed: 13-Sep-2019].
[91] “Intel Hyper-Threading Technology.” [Online]. Available: https://www.intel.com/content/www/us/en/architecture-and-technology/hyper-threading/hyper-threading-technology.html. [Accessed: 13-Sep-2019].
[92] “Parallelising Python with Threading and Multiprocessing,” *QuarkGluon Ltd.* [Online]. Available: https://www.quantstart.com/articles/Parallelising-Python-with-Threading-and-Multiprocessing. [Accessed: 13-Sep-2019].
[93] “Los Alamos National Laboratory: MCNP Home Page,” *Los Alamos National Laboratory*. [Online]. Available: https://mcnp.lanl.gov/. [Accessed: 13-Sep-2019].
[94] G. W. Mckinney, F. B. Brown, H. G. Hughes, M. R. James, and R. L. Martz, “MCNP 6.1.1-New Features Demonstrated,” in *IEEE Nuclear Science Symposium and Medical Imaging Conference*, 2014.
[95] J. Dakin and R. G. W. Brown, “Spectral Response Characteristcs of Photodetectors,” in *Handbook of optoelectronics. Volume 1, Concepts, devices, and techniques*, CRC Press, 2017, p. 407.
[96] “About Digi | Digi International.” [Online]. Available: https://www.digi.com/about-digi. [Accessed: 13-Sep-2019].
[97] H. T. Friis, “A Note on a Simple Transmission Formula,” *Proc. IRE*, vol. 34, no. 5, pp. 254–256, 1946.
[98] “User Facilities | ORNL.” [Online]. Available: https://www.ornl.gov/content/user-facilities. [Accessed: 13-Sep-2019].
[99] “Y-12 National Security Complex.” [Online]. Available: https://www.y12.doe.gov/. [Accessed: 13-Sep-2019].
[100] Department of Energy, “Highly Enriched Uranium: Striking a Balance - Section 3: US HEU Inventory,” 2001.
[101] R. W. Schafer, “What is a Savitzky-Golay Filter?,” *IEEE Signal Process. Mag.*, vol. 28, no. 4, pp. 111–117, 2011.
[102] A. Savitzky and M. J. E. Golay, “Smoothing and Differentiation of Data by Simplified Least Squares Procedures,” *Anal. Chem.*, vol. 36, no. 8, pp. 1627–1639, Jul. 1964.
[103] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling, “Savitzky-Golay Smoothing Filters,” in *Numercal Recipes in Fortran 77: The Art of Scientific Computing*, 1986.
[104] J. Gurka, “A Bayesian Approach to Multi-Drone Source Localization Methods,” University of Tennessee, 2019.
[105] “The New York Stock Exchange | NYSE.” [Online]. Available: https://www.nyse.com/index. [Accessed: 13-Sep-2019].
[106] “IAEA Incident and Trafficking Database (ITDB),” 2019.
[107] “The Albatross That Turned Into A Predator – UAS VISION.” [Online]. Available: https://www.uasvision.com/2011/12/30/the-albatross-that-turned-

into-a-predator/. [Accessed: 13-Sep-2019].
[108] “Beginner’s Series: Four Forces of Flight.” [Online]. Available: http://smartflighttraining.com/beginners-series-four-forces-flight. [Accessed: 13-Sep-2019].
[109] D. Iturralde, C. Azurdia-Meza, N. Krommenacker, I. Soto, Z. Ghassemlooy, and N. Becerra, “A New Location System for an Underground Mining Environment using Visible Light Communications,” in *2014 9th International Symposium on Communication Systems, Networks & Digital Sign (CSNDSP)*, 2014, pp. 1165–1169.
[110] “Carbon Fiber Unidirectional Cloth.” [Online]. Available: https://www.nj-mkt.com/news/932.html. [Accessed: 13-Sep-2019].
[111] “mission_planner_flight_data.jpg (1000×608).” [Online]. Available: http://ardupilot.org/planner/_images/mission_planner_flight_data.jpg. [Accessed: 13-Sep-2019].
[112] “Eljen Technology - Plastic Scintillators.” [Online]. Available: https://eljentechnology.com/products/plastic-scintillators. [Accessed: 13-Sep-2019].
[113] “ET-Enterprises Ltd.” [Online]. Available: http://et-enterprises.com/?gclid=EAIaIQobChMI8uqZgb_O5AIVQeDICh2y7gn0EAAYASAAEgLO3_D_BwE. [Accessed: 13-Sep-2019].
[114] “Bridgeport Instruments MCA USB.” [Online]. Available: http://www.bridgeportinstruments.com/products/oem_base/oem_base.html. [Accessed: 13-Sep-2019].

APPENDICES

Appendix A: Figures

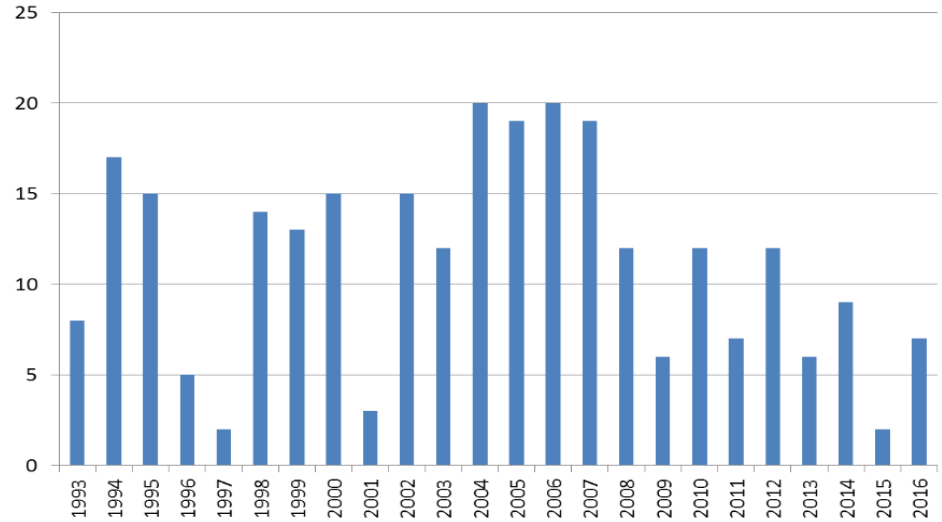

Figure A.1. "Confirmed Incidents Related to Trafficking or Malicious use of Lost or Stolen Nuclear Material[106]"

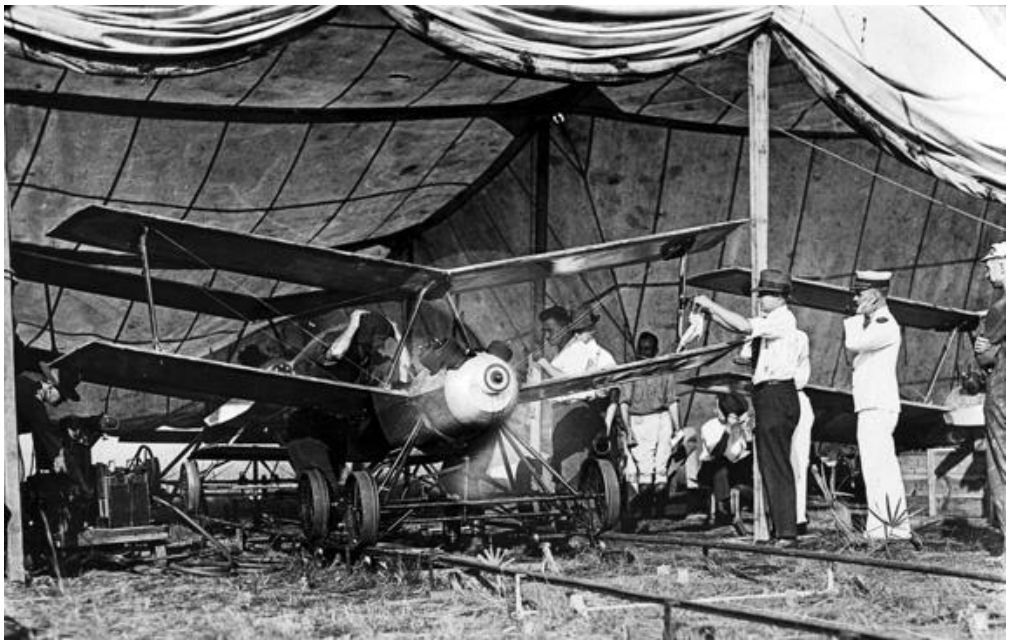

Figure A.2. The Kettering "Bug" [26]

Figure A.3. Mr. Kedem and the Albatross Aircraft[107]

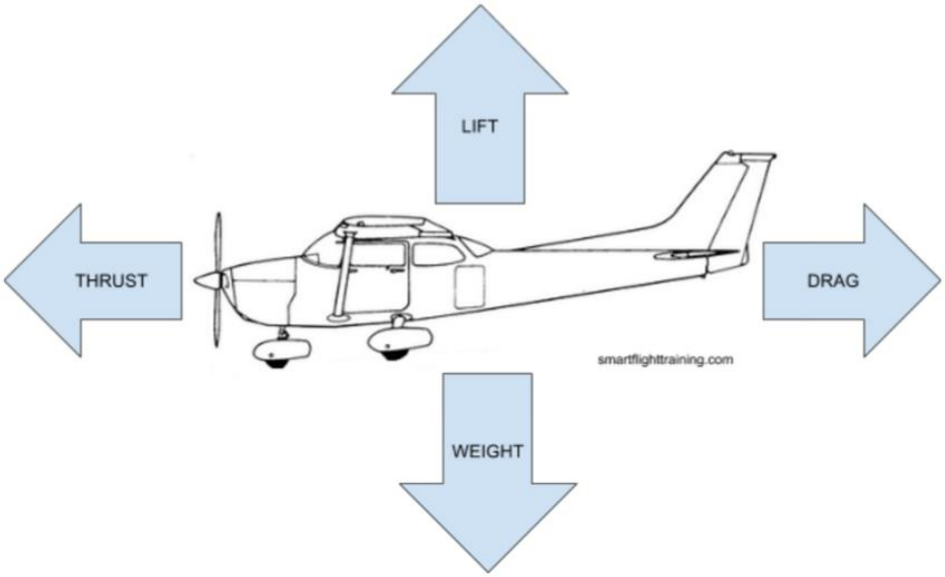

Figure A.4. Four Principles of Aircraft Design[108]

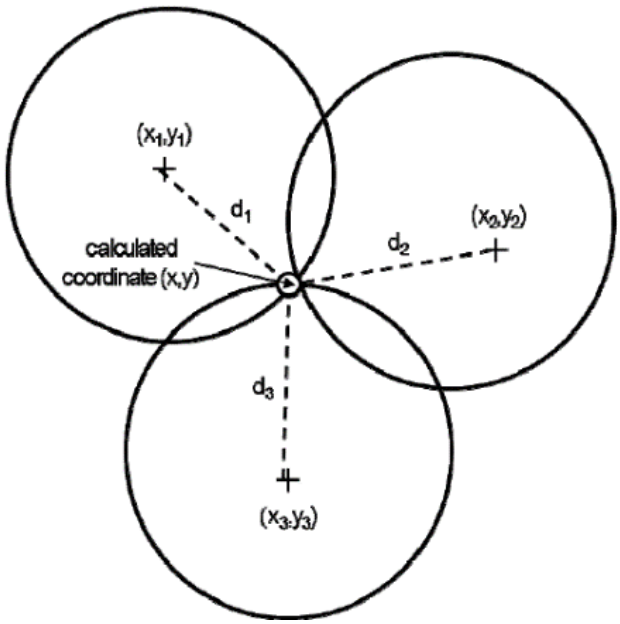

Figure A.5. Method of Trilateration[109]

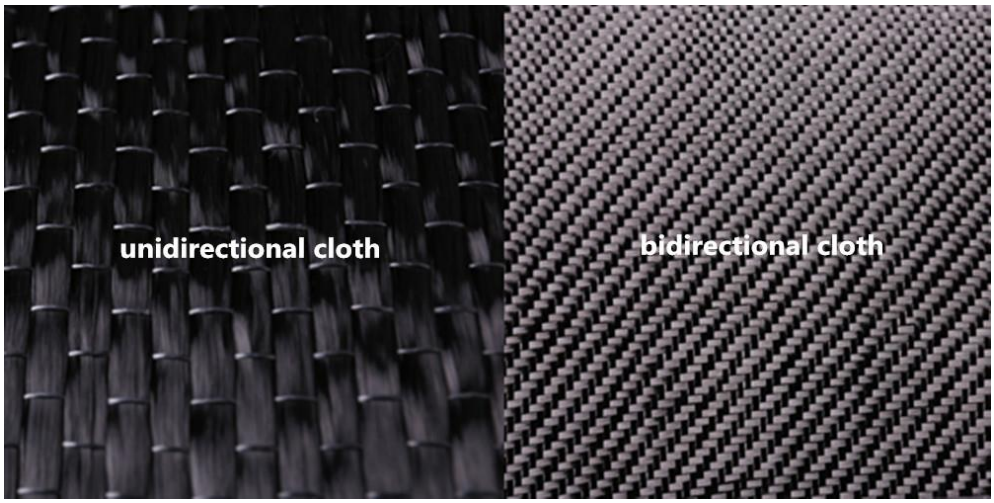

Figure A.6. Carbon Fiber Weave Comparison[110]

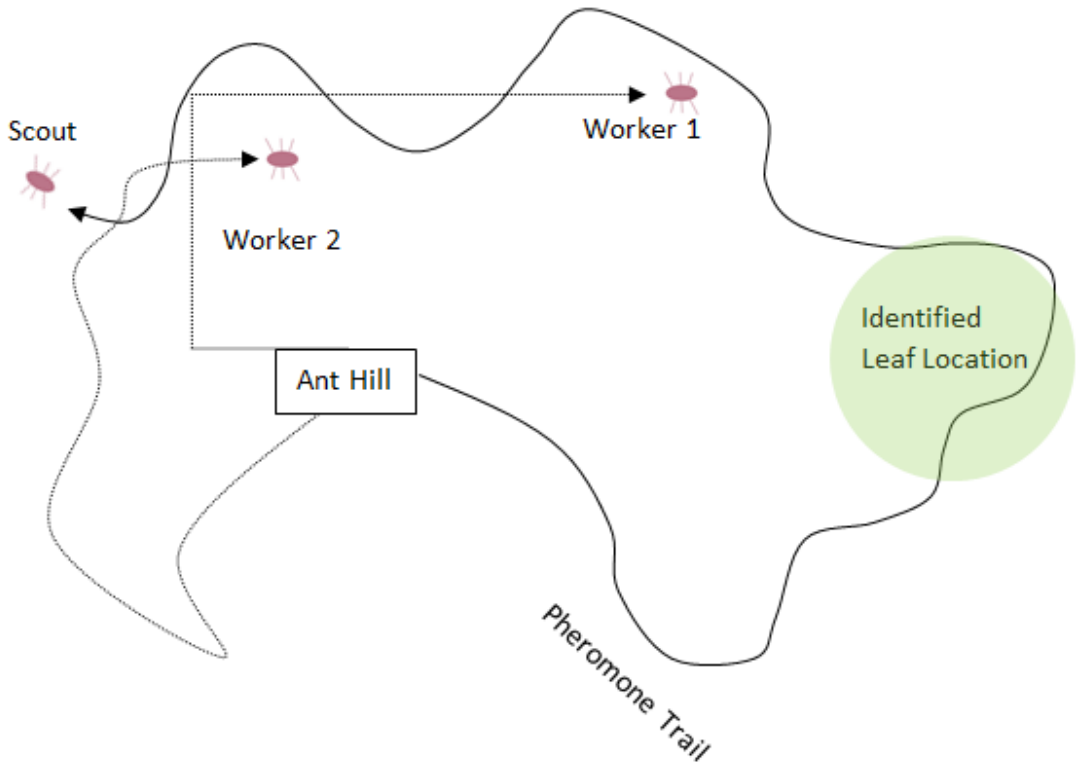

Figure A.7. Ant-based Swarming Methodology Using a Scout Ant and Two Worker Ants

Figure A.8. Armed UAV Used by ISIL[58]

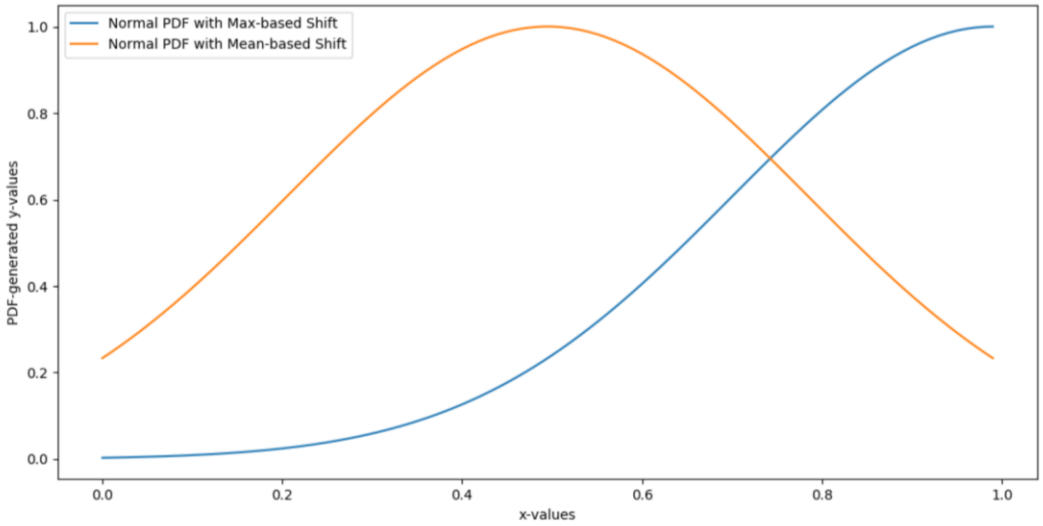

Figure A.9. PDF Shift Comparison

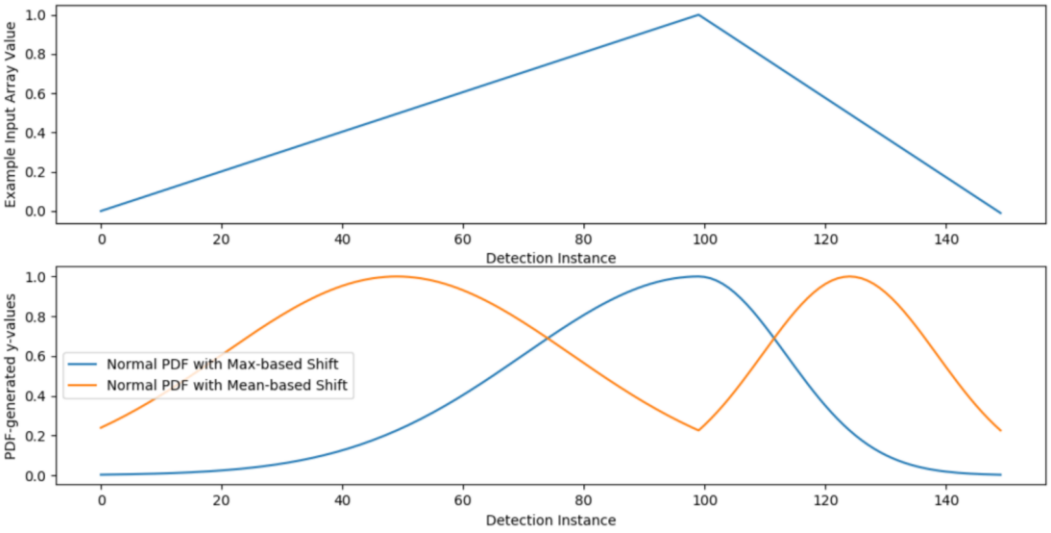

Figure A.10. Maximum Value-based Shift Reasoning

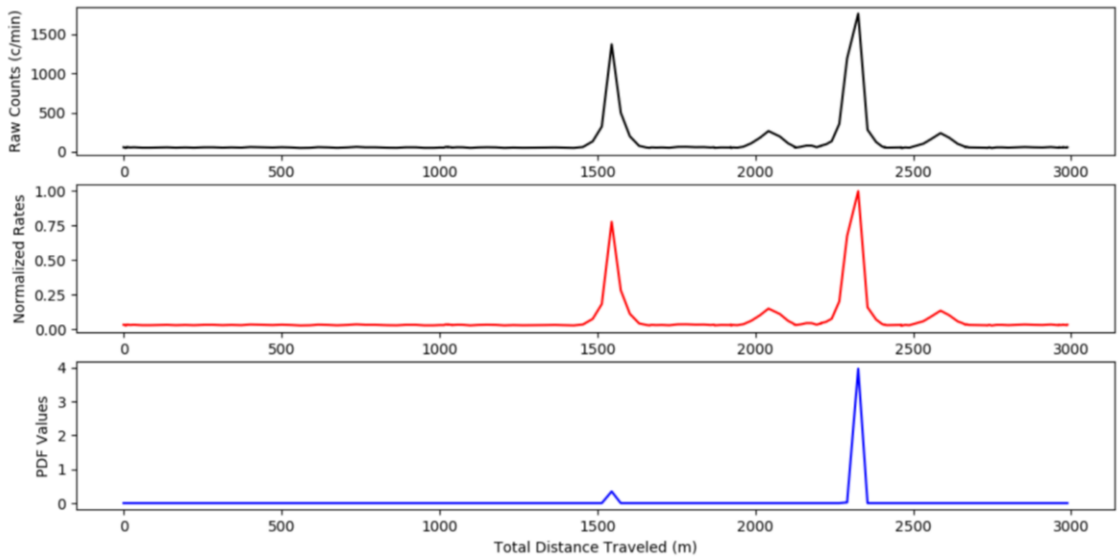

Figure A.11. Comparison between Raw Count Rates, Normalized Values, and PDF-Generated Values for Bayesian Calculation Input

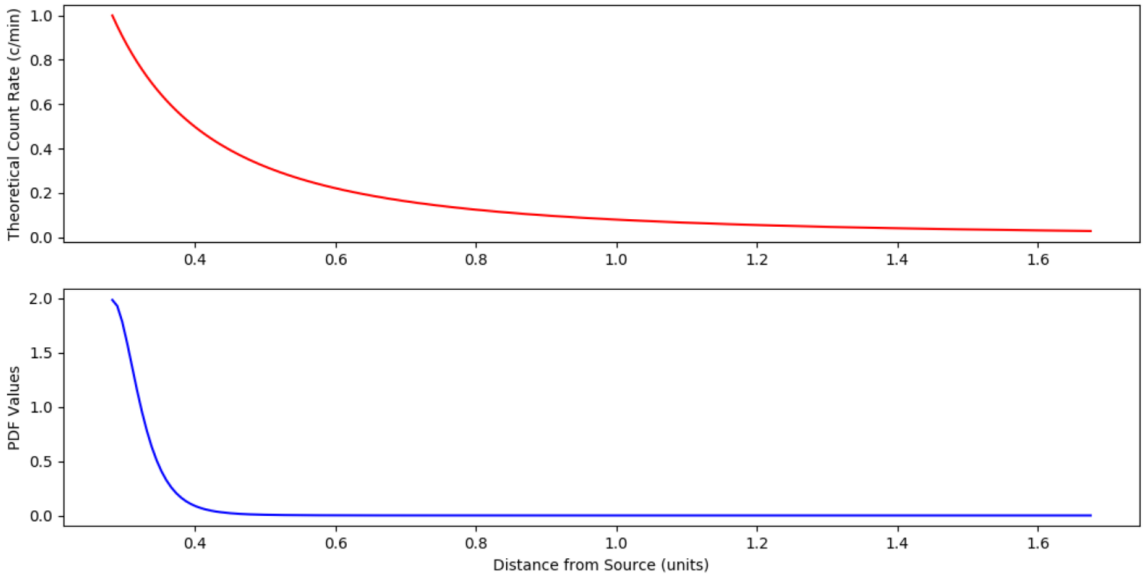

Figure A.12. Comparison of Theoretical Count Rate Curve and PDF-Generated Values Found Using the Inverse Square Law

Figure A.13. Example of Main Screen from Mission Planner with Basic Aircraft Gauges and Map Output[111]

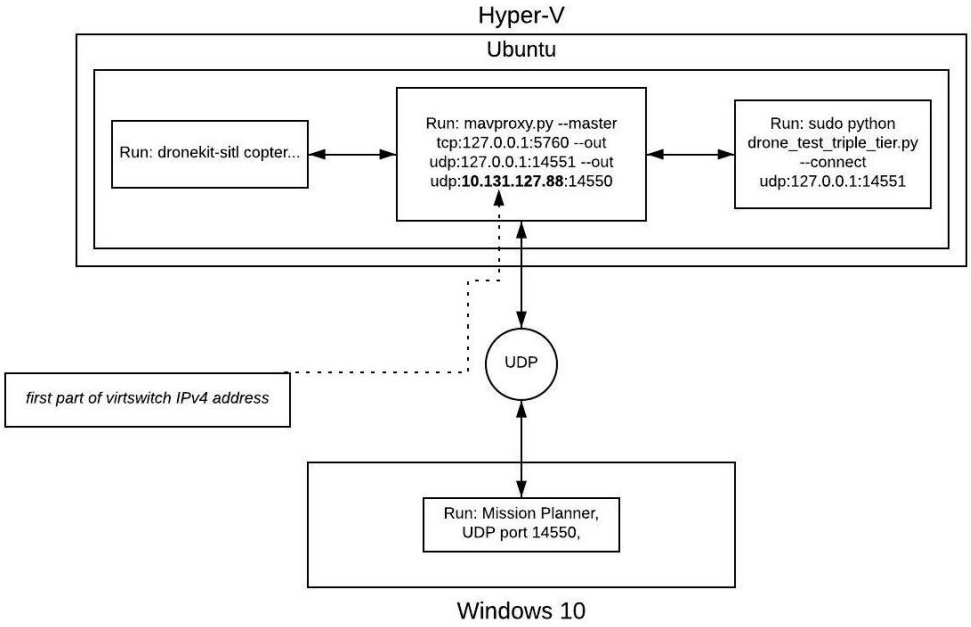

Figure A.14. First Successful Early Testing System Architecture Setup using a Real and a Virtual Machine

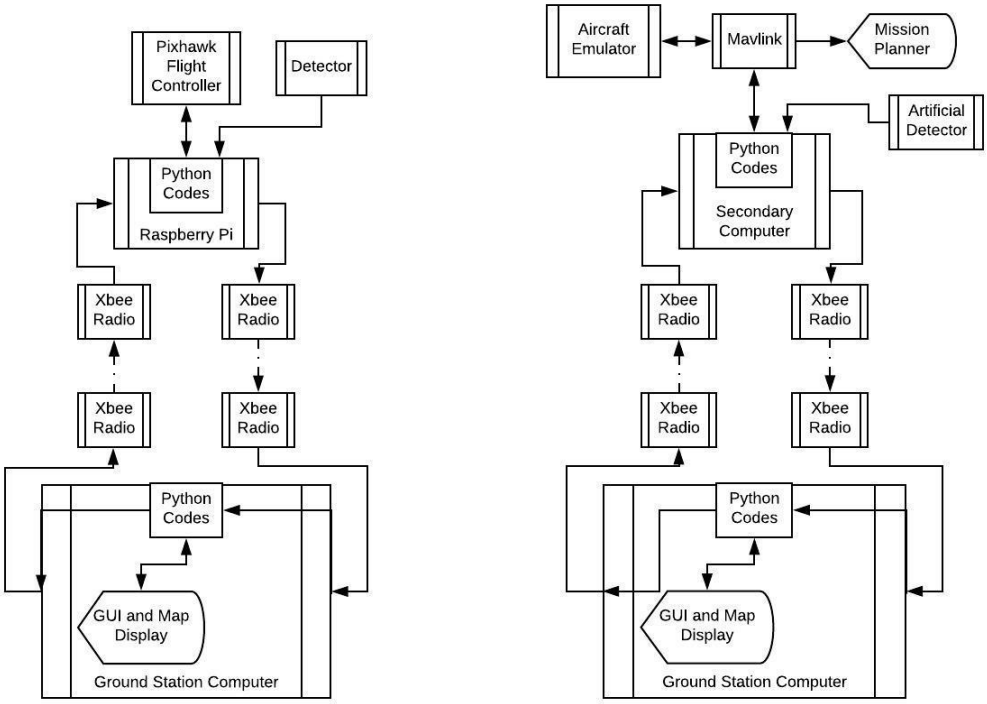

Figure A.15. Initial Diagrams of Algorithm Operation Sequence with Initial Desired Final Real-World Design (left) and Initial Simulated Algorithm Design (right).

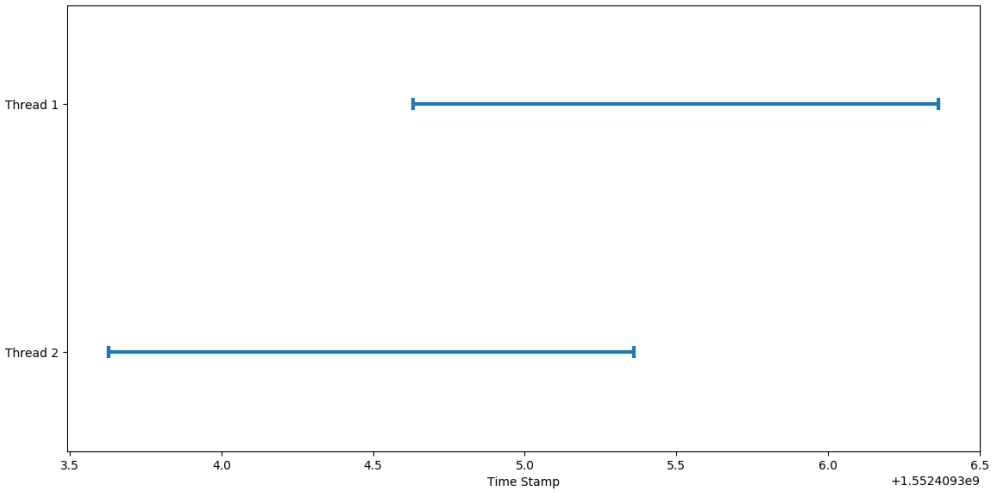

Figure A.16. Python Multithreaded Timed Example Code with Two Threads Showing Concurrency for Timing Comparison

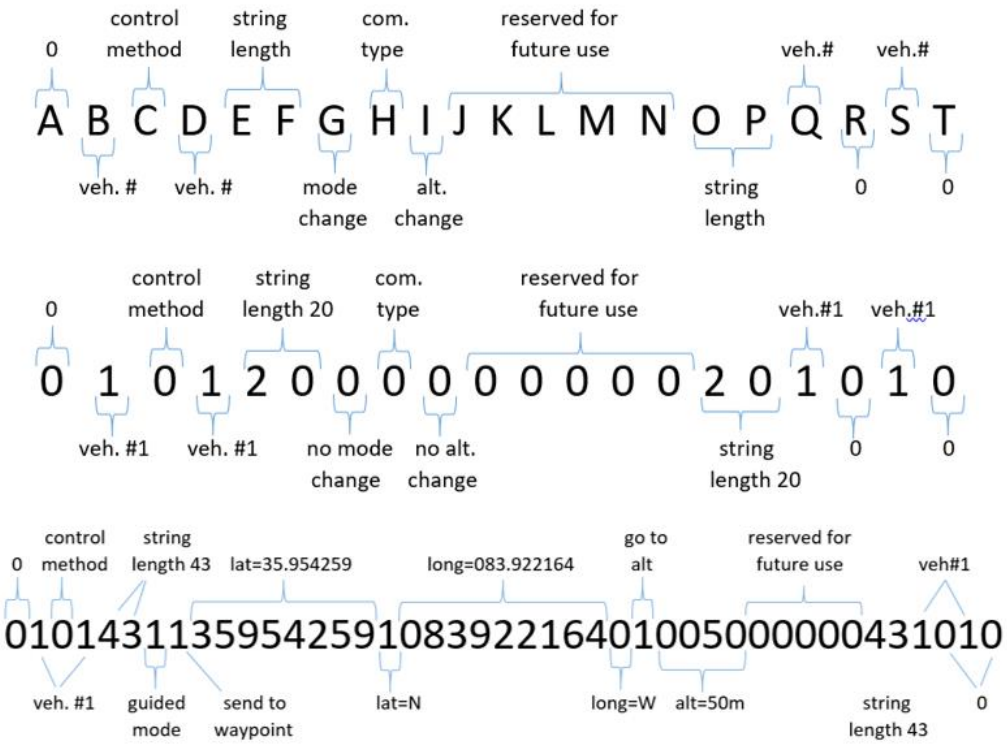

Figure A.17. Command Encoding Method Examples used in the Algorithm for Vehicle Control Commands

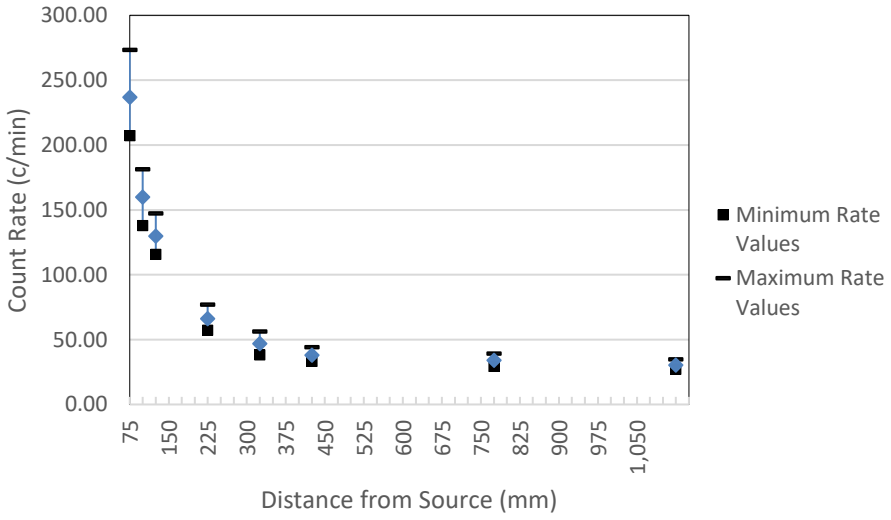

Figure A.18. Display of Cesium Calibration Test Results

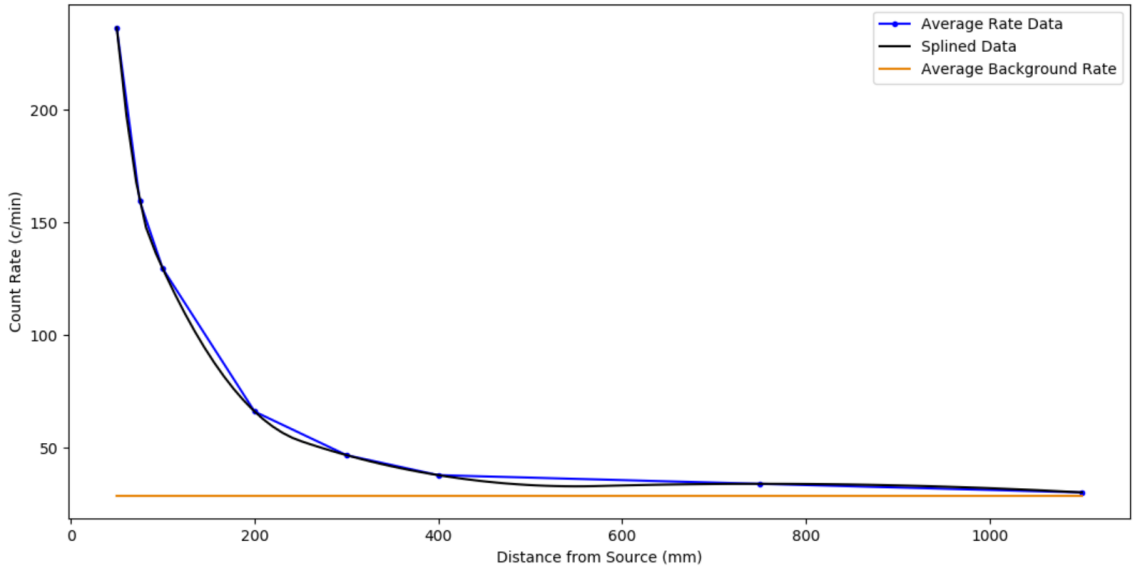

Figure A.19. Source Strength Curve with Mean Background Rate shown as Asymptote

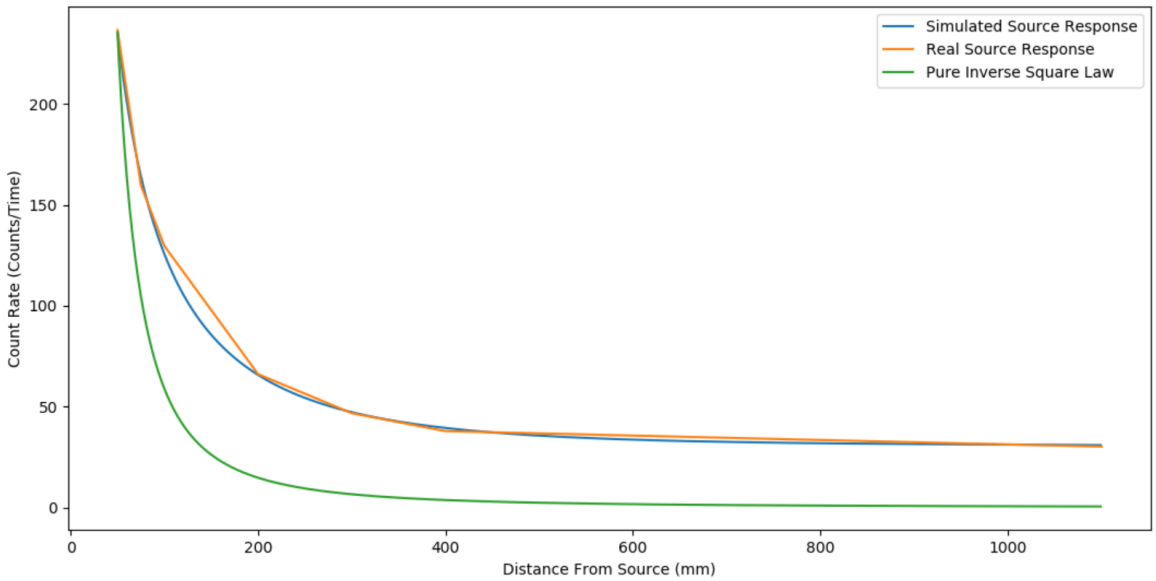

Figure A.20. Source Strength Replicator Curve Comparison for Lambda Function, Experimental Results, and Theoretical Inverse Square Law

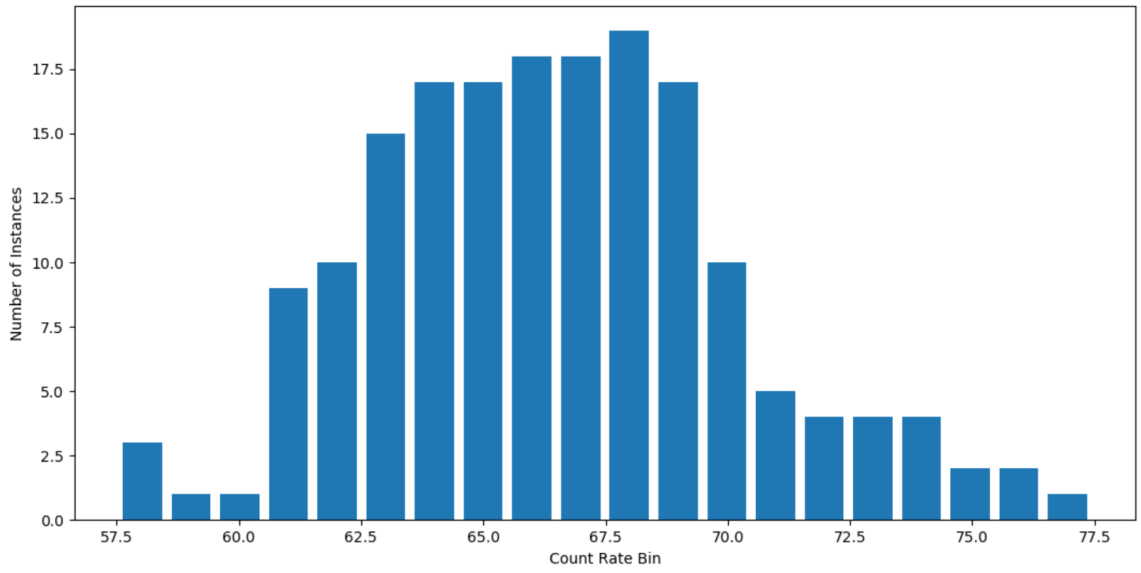

Figure A.21. Count Rate Data Distribution at 200mm from Source

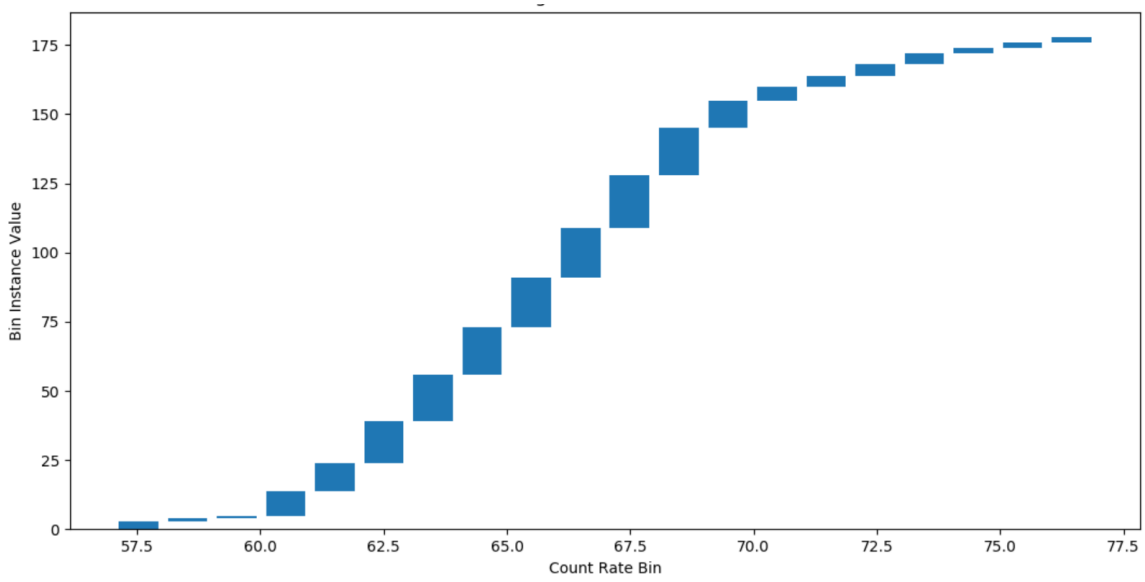

Figure A.22. Bins shown to Scale as Probability Instances at 200mm from the Source

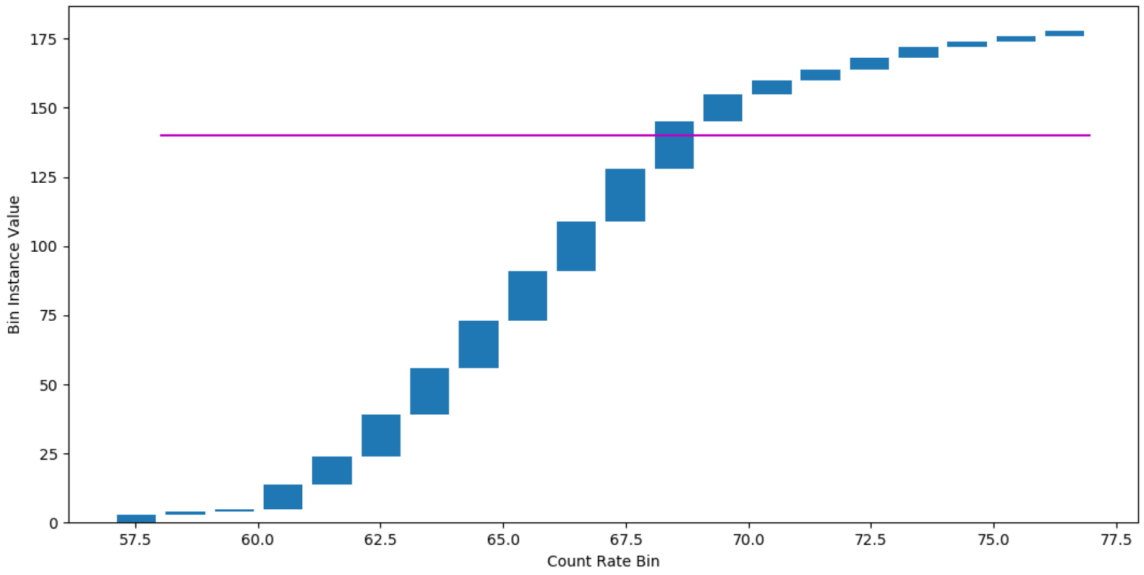

Figure A.23. Display of First Random Value Generated on Bin Distribution

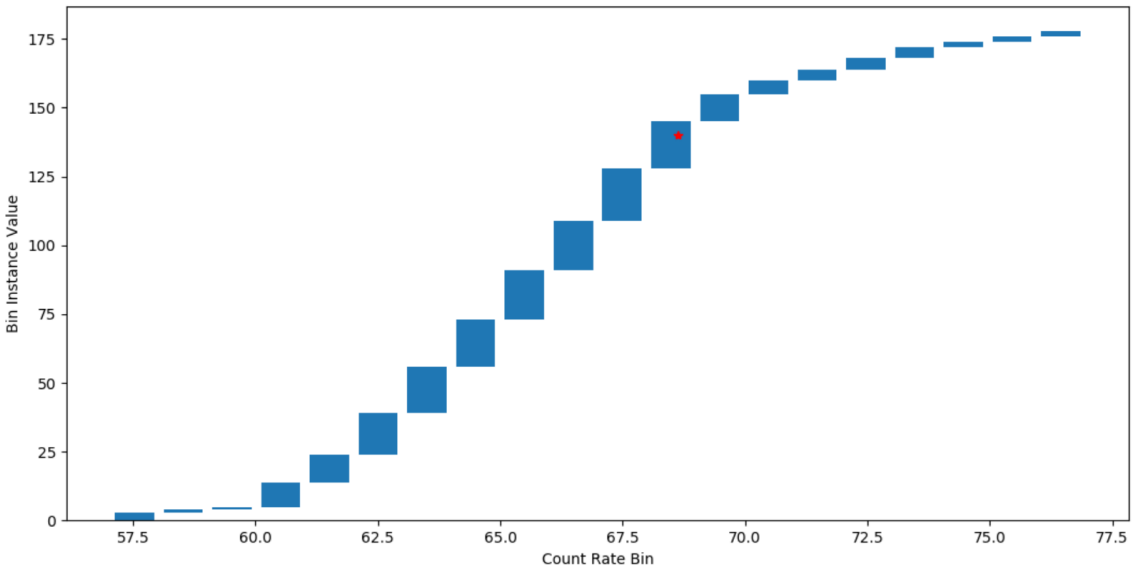

Figure A.24. Final Calculated Count Rate Result

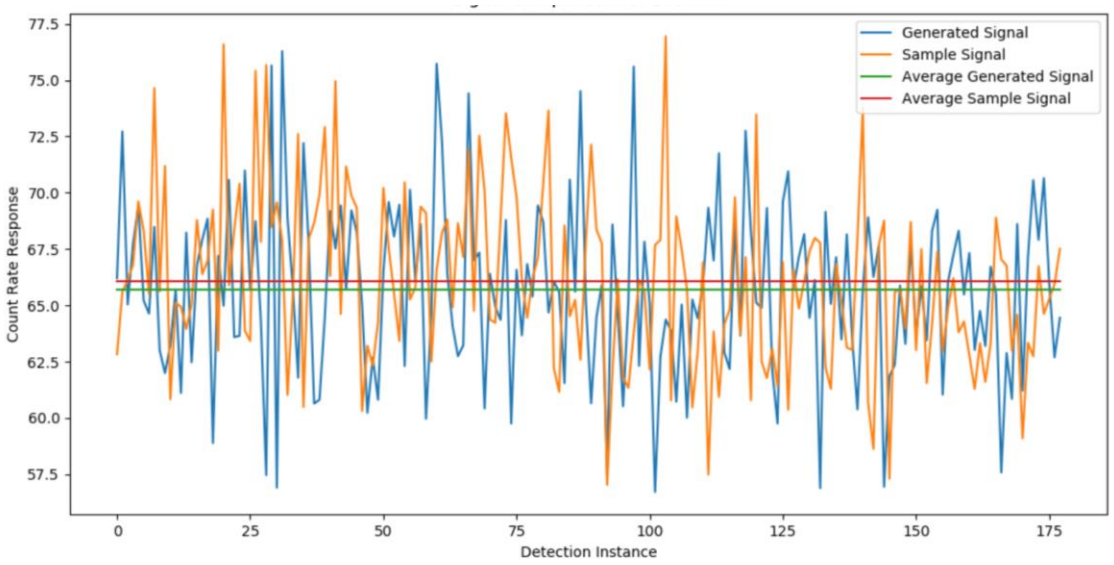

Figure A.25. Comparison at 200mm from Source between the Actual and Generated Signals

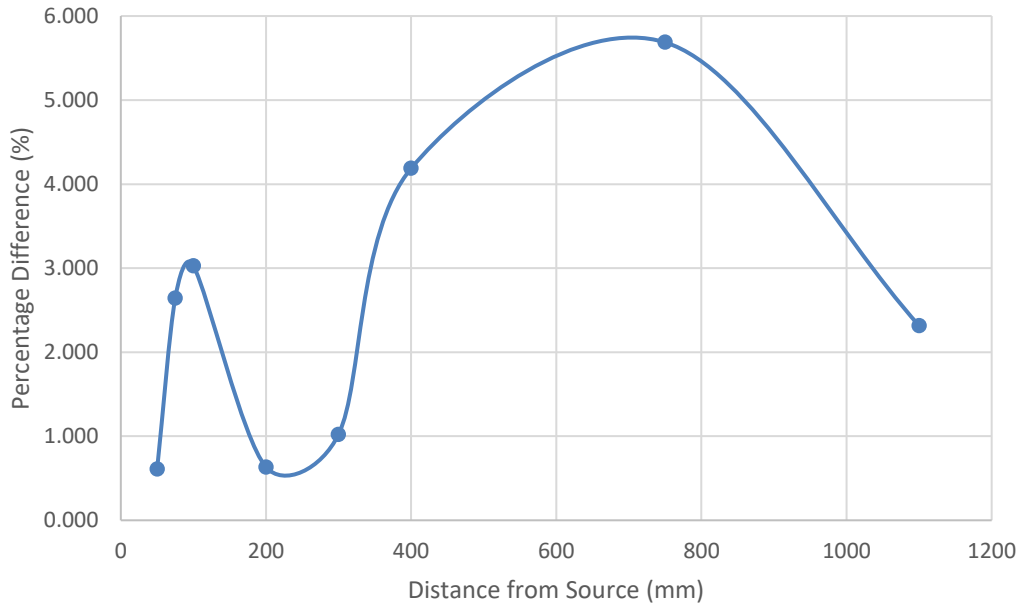

Figure A.26. Absolute Value of Percentage Difference of Average Generated Signal Value and the Actual Detected Calibration Value

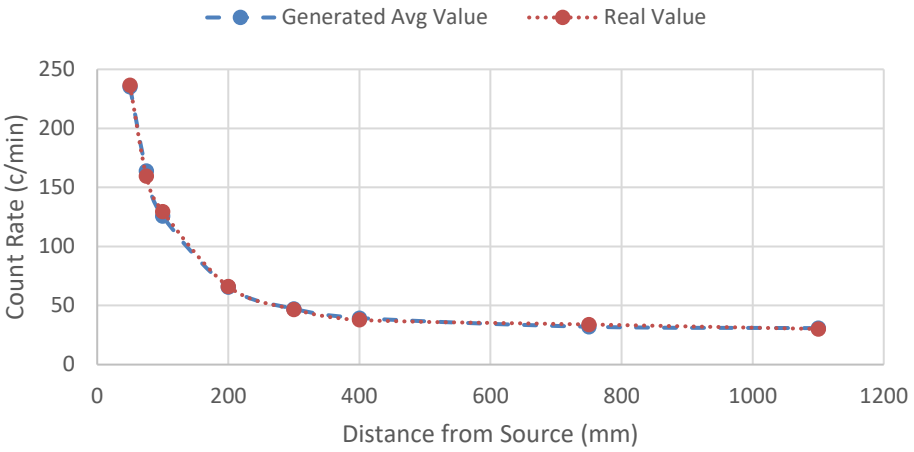

Figure A.27. Comparison between Real Calibrated Average Values and Lambda-Bin Generated Signal Values

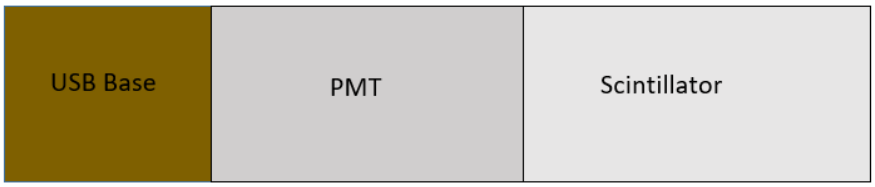

Figure A.28. Attachment Sequence for Scintillator Package

Figure A.29. HL-6 Aircraft with Flexible Carbon Fiber Platform Constructed between Leg Spreaders

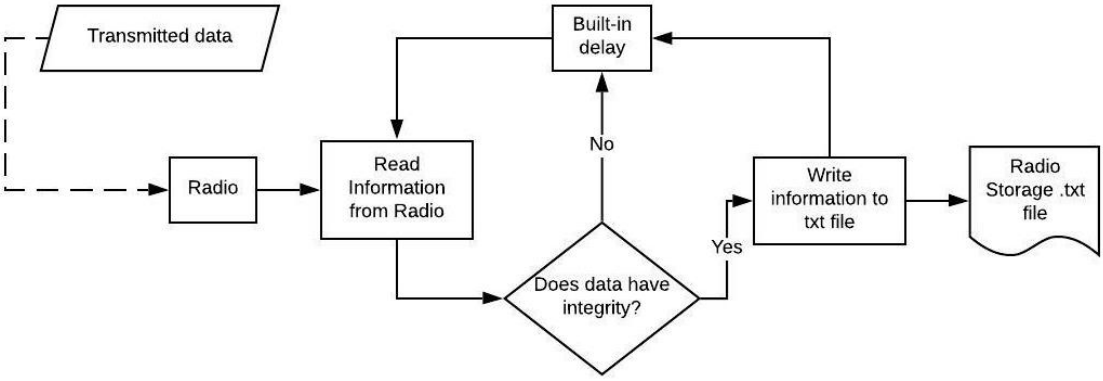

Figure A.30. Radio Thread (Reader/Receiver)

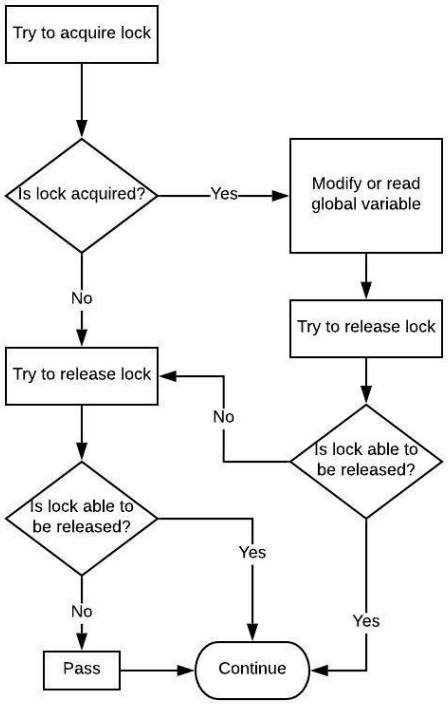

Figure A.31. Lock System for Global Variables Using Nested Try-Except

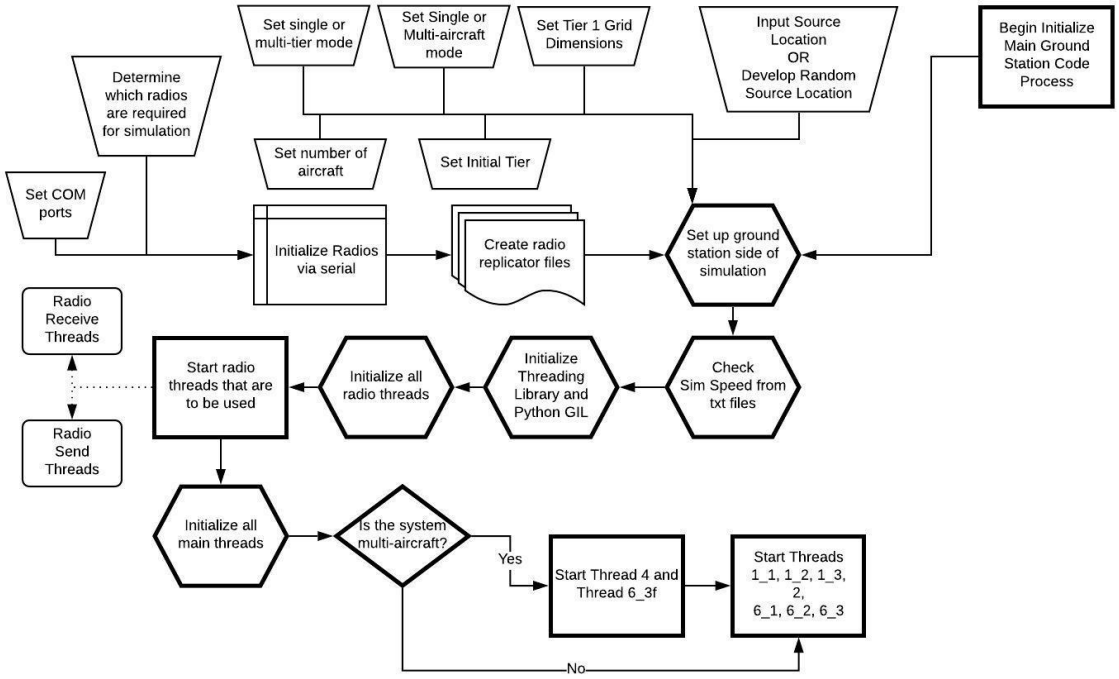

Figure A.32. Ground Station Summary

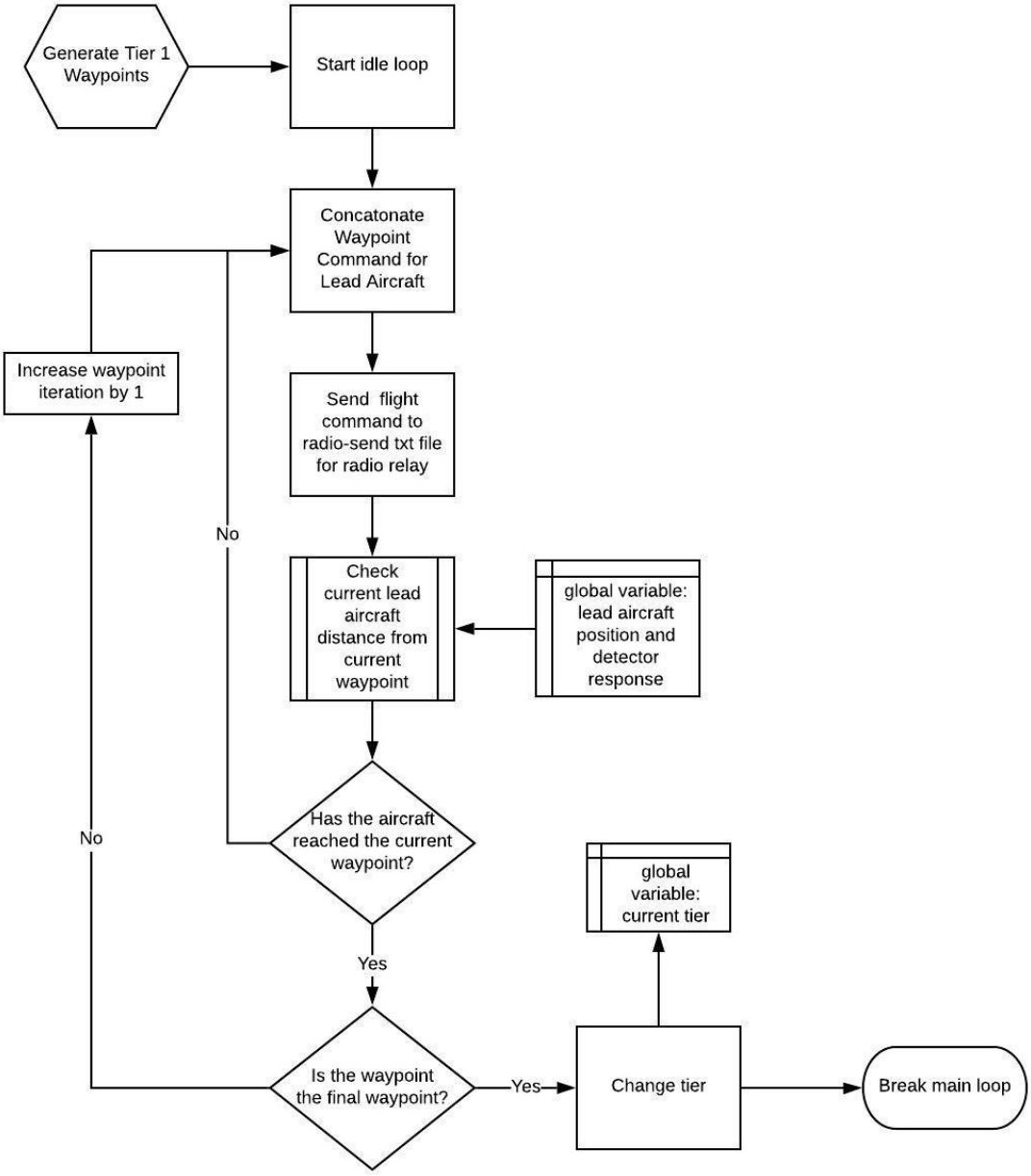

Figure A.33. Thread 1_1 (Ground Station Tier 1 Command Thread)

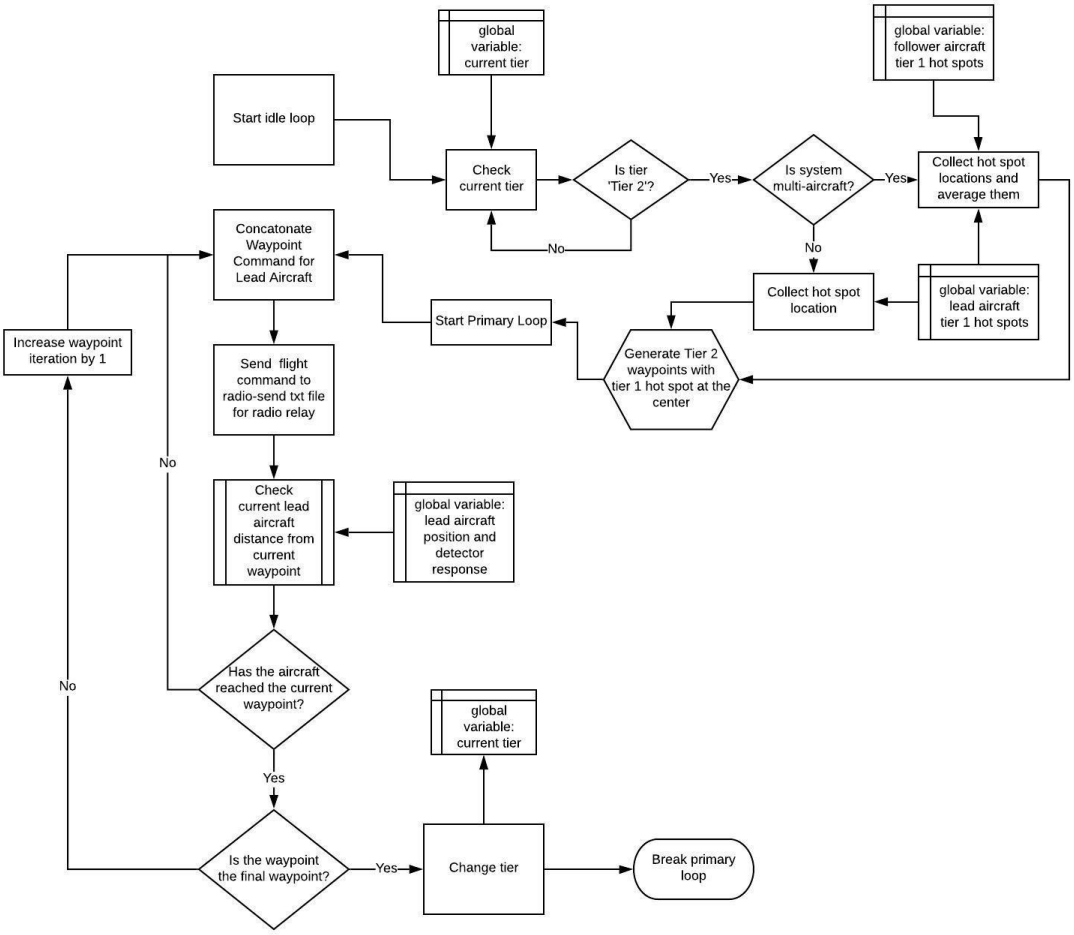

Figure A.34. Thread 1_2 (Ground Station Tier 2 Command Thread)

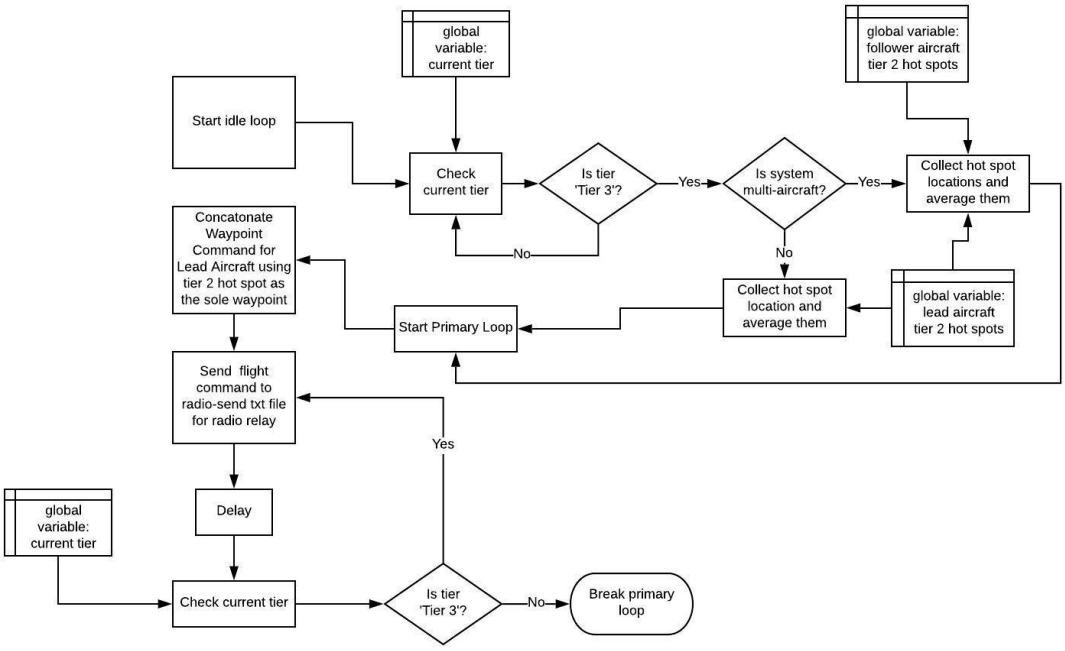

Figure A.35. Thread 1_3 (Ground Station Tier 3 Command Thread)

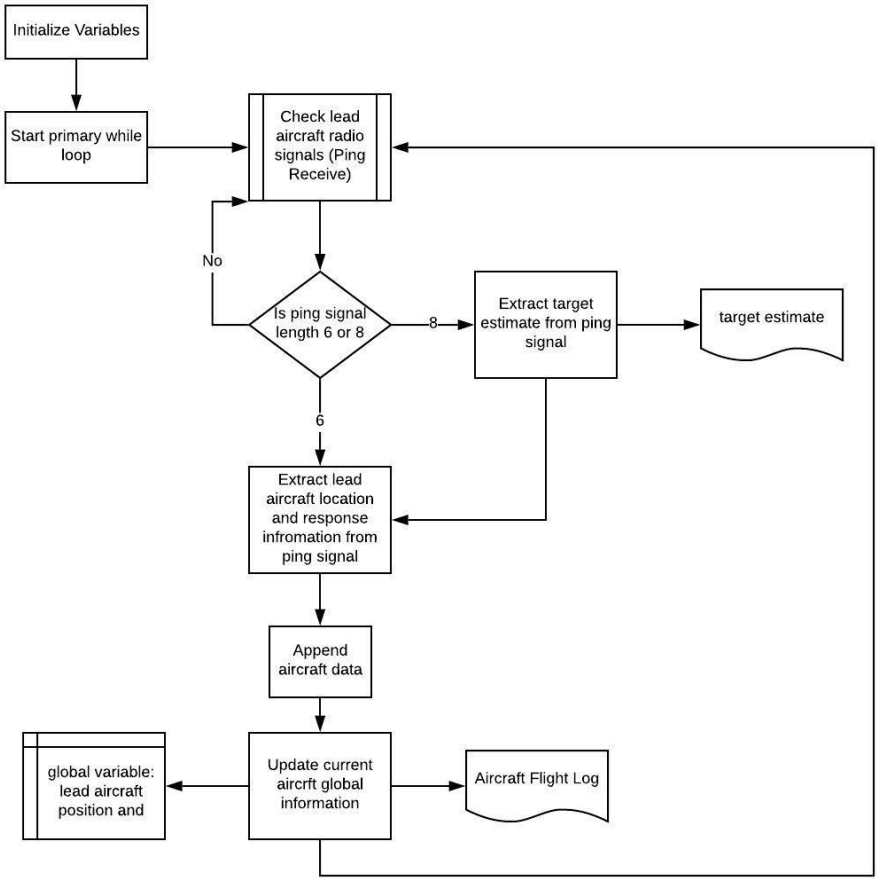

Figure A.36. Thread 2 (Ground Station Ping Receive Thread for Lead Aircraft)

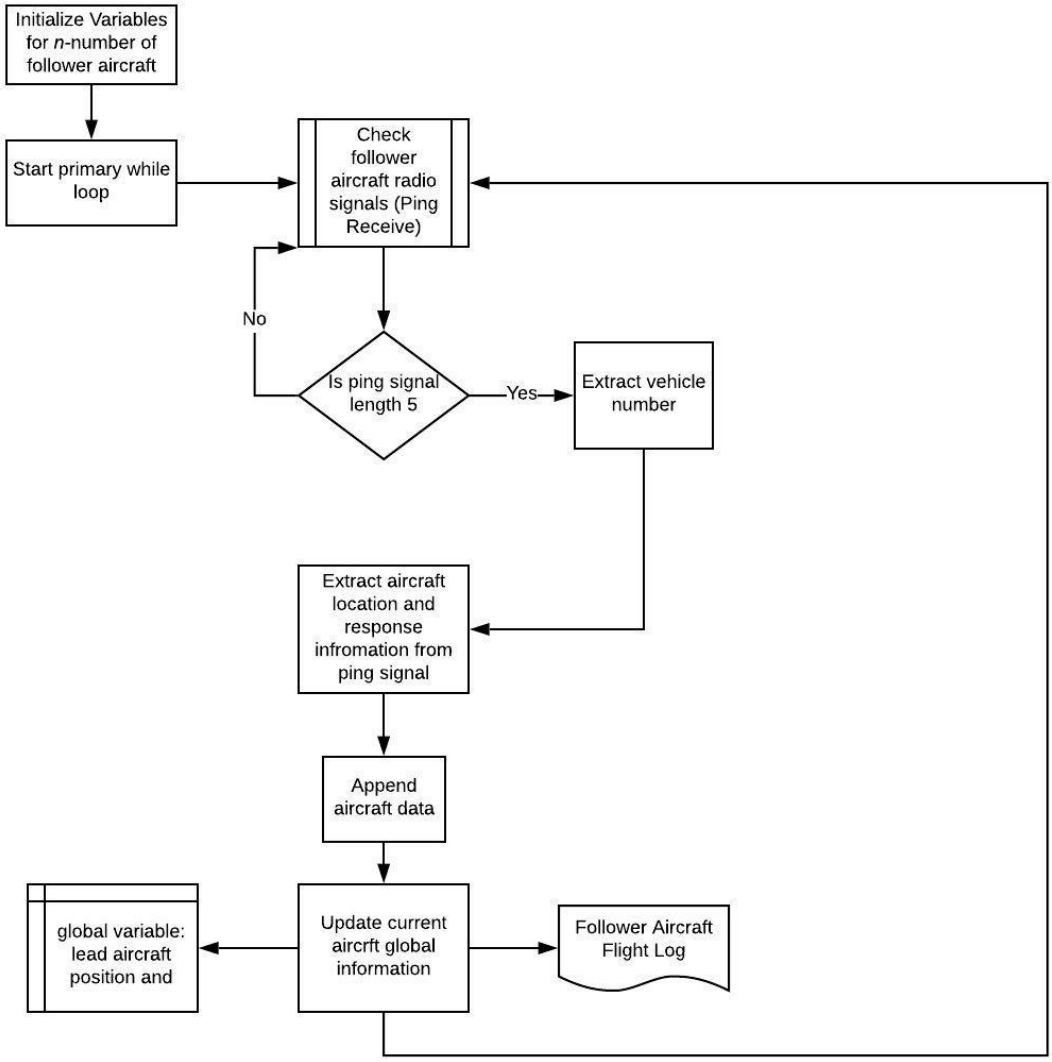

Figure A.37. Thread 4 (Ground Station Ping Receive Thread for Follower Aircraft(s))

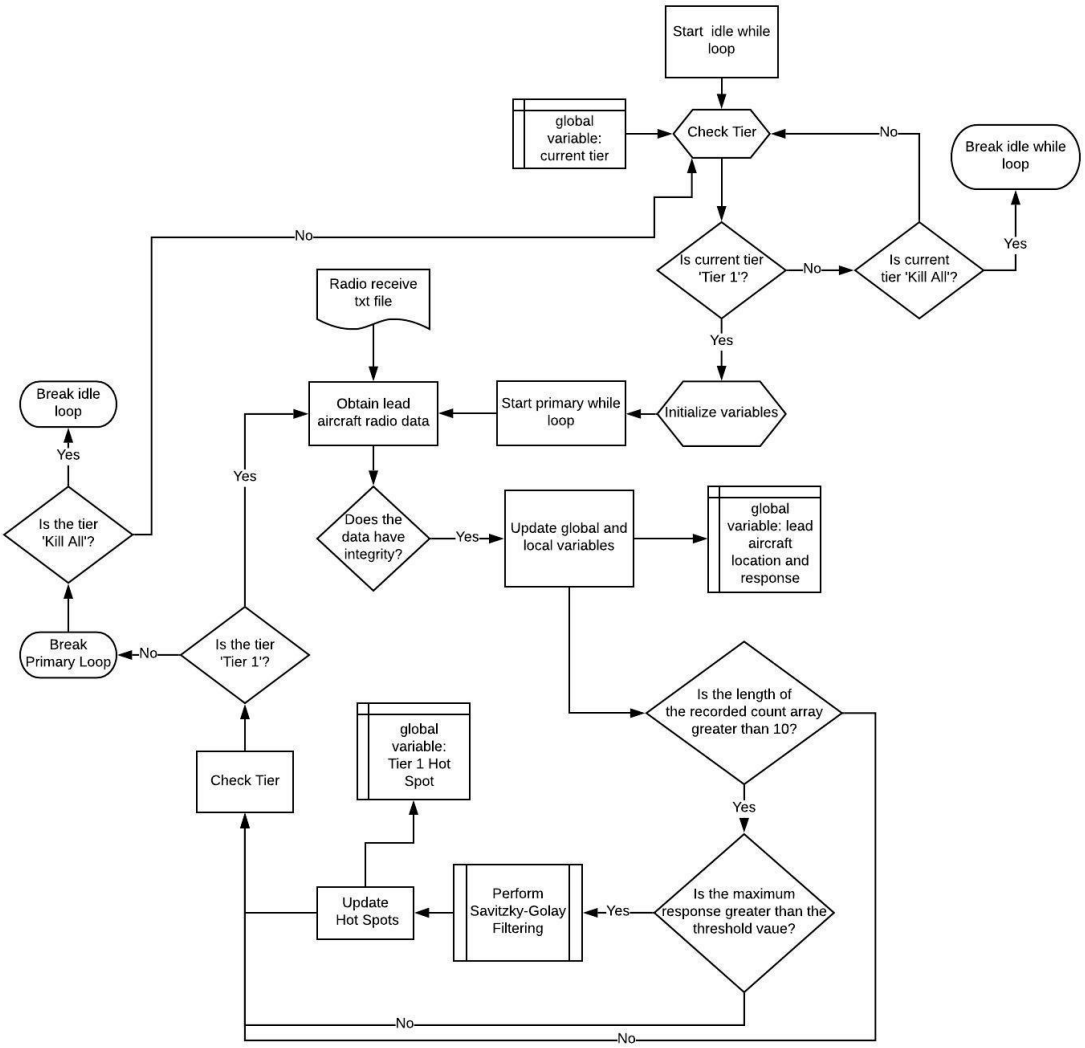

Figure A.38. Thread 6_1 (Ground Station Analysis Thread for Tier 1 lead Aircraft Data)

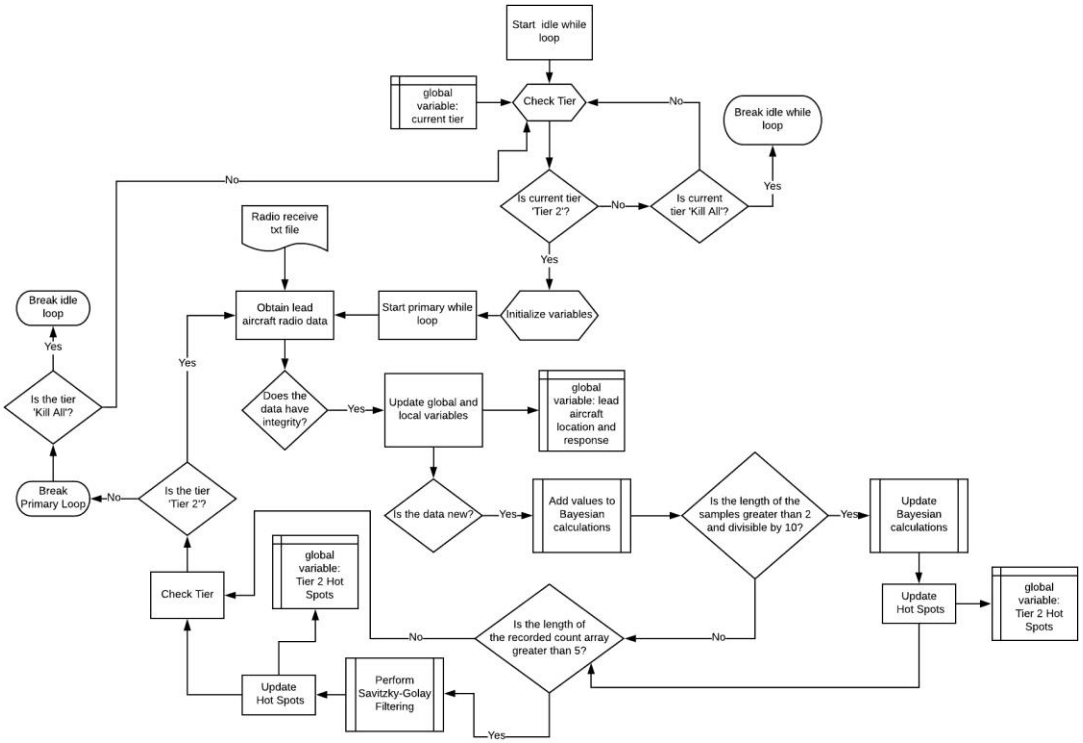

Figure A.39. Thread 6_2 (Ground Station Analysis Thread for Tier 2 Lead Aircraft Data)

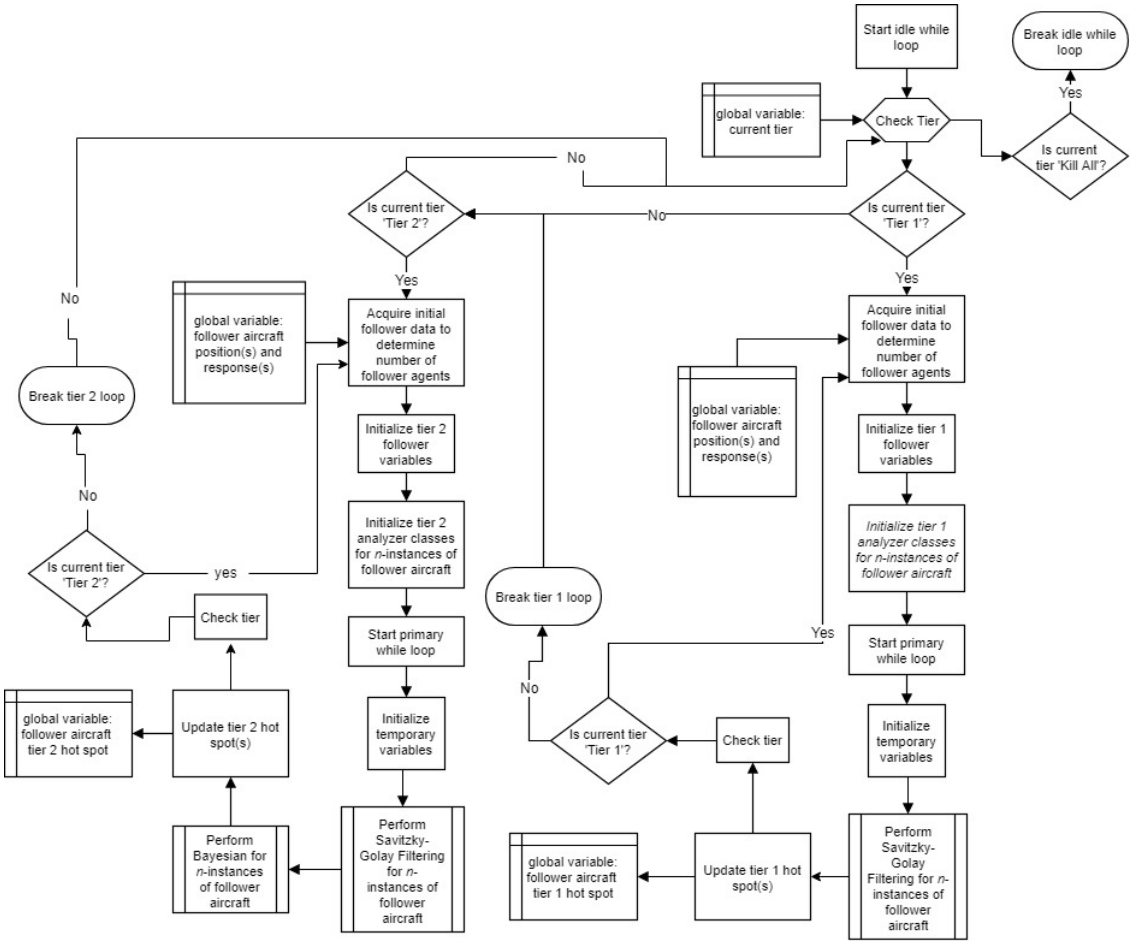

Figure A.40. Thread 6_3f (Ground Station Analysis Thread for Tiers 1 and 2 Follower Aircraft(s) Data)

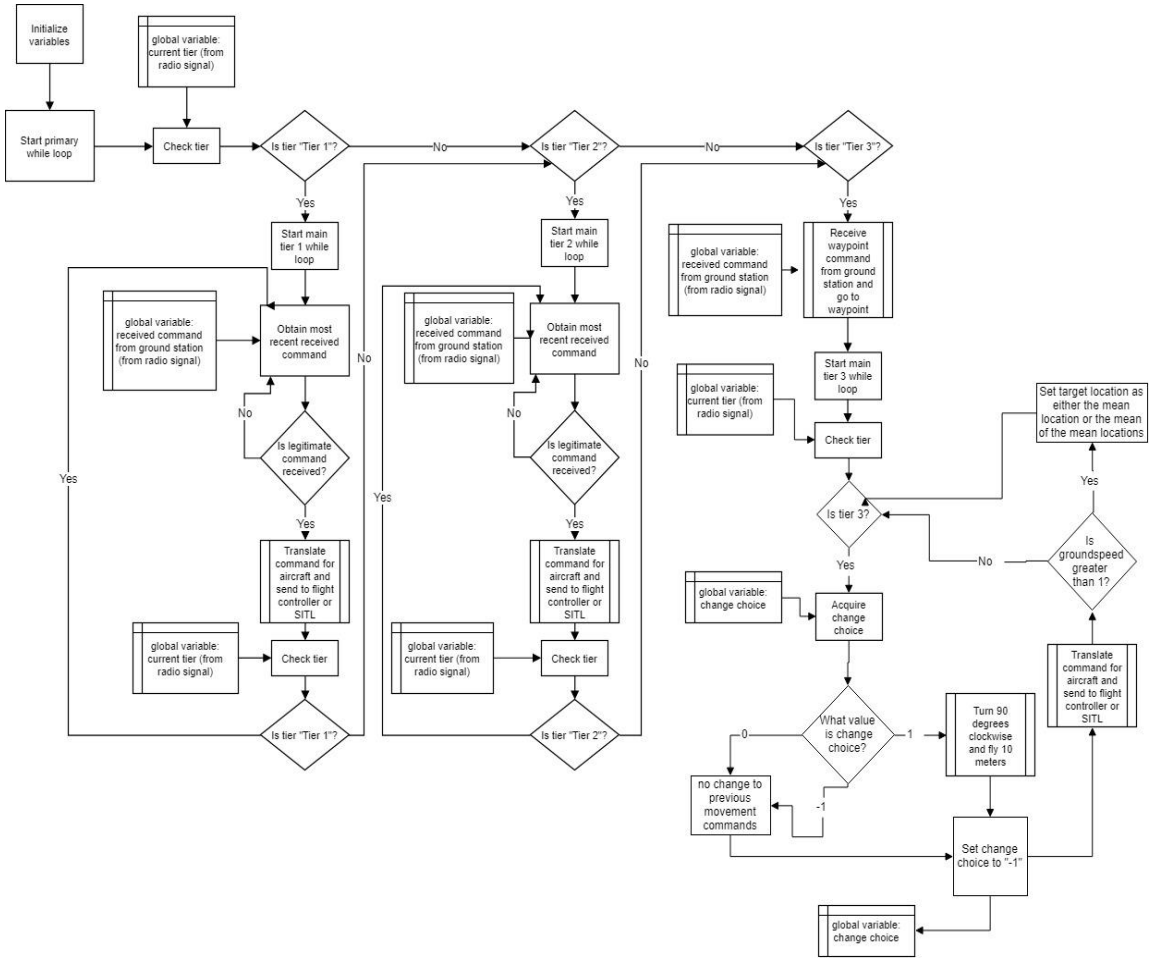

Figure A.41. Thread 1 (Lead Aircraft Command Translator Thread for All Tiers)

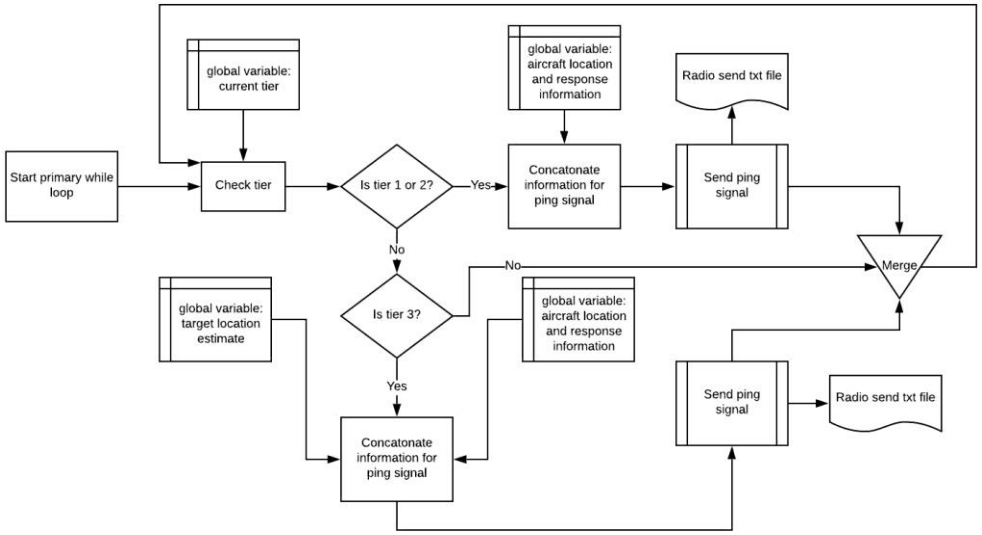

Figure A.42. Thread 2 (Lead Aircraft Ping Send Thread)

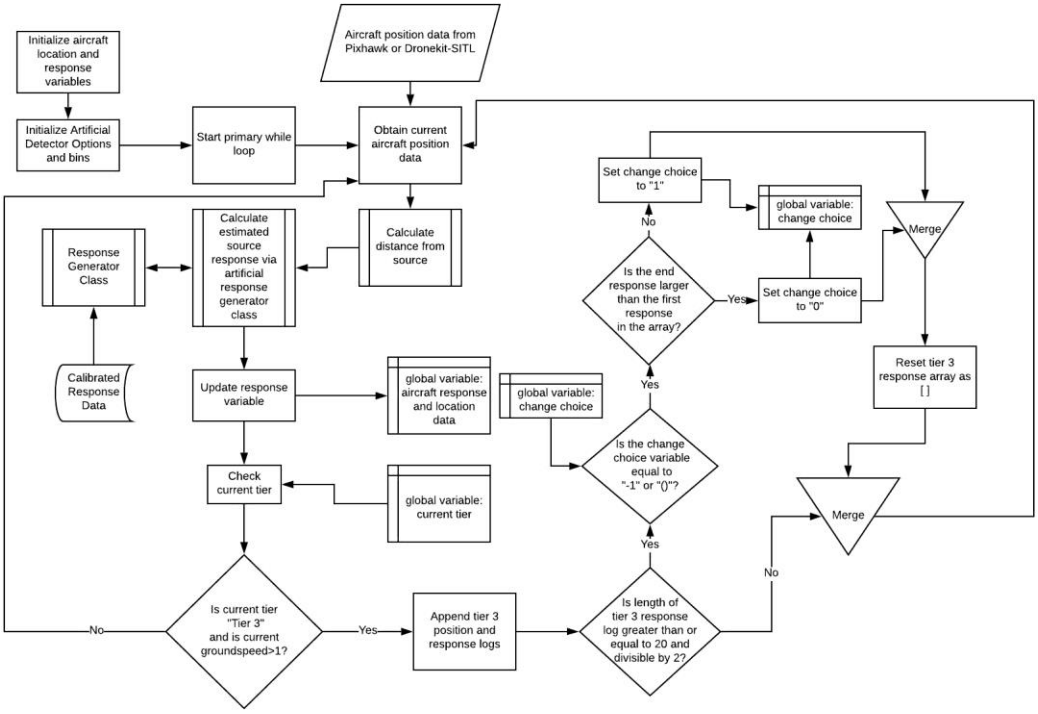

Figure A.43. Thread 3 (Lead Aircraft Artificial Detector Response Generator and Tier 3 Calculation Thread)

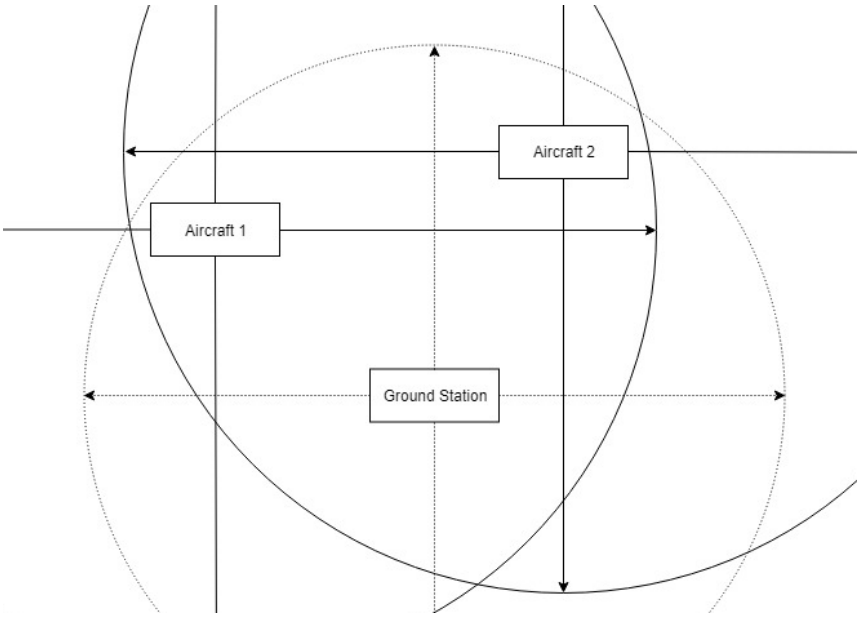

Figure A.44. Burst Transmission Method between Ground Station and Two Aerial Vehicles

Figure A.45. Diagram of Single Radio Communications Setup

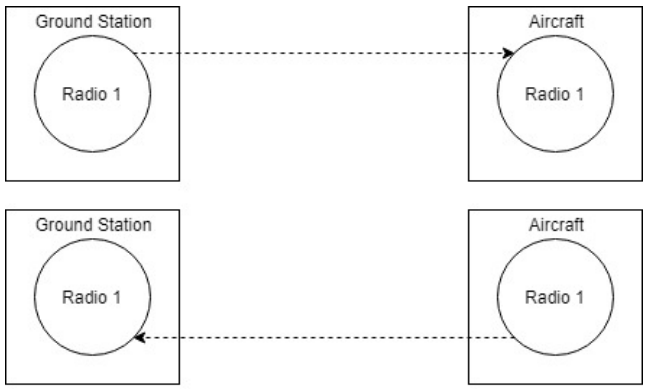

Figure A.46. Expanded Diagram of Transmission/Receipt Steps for Two-Way Half-Duplex System

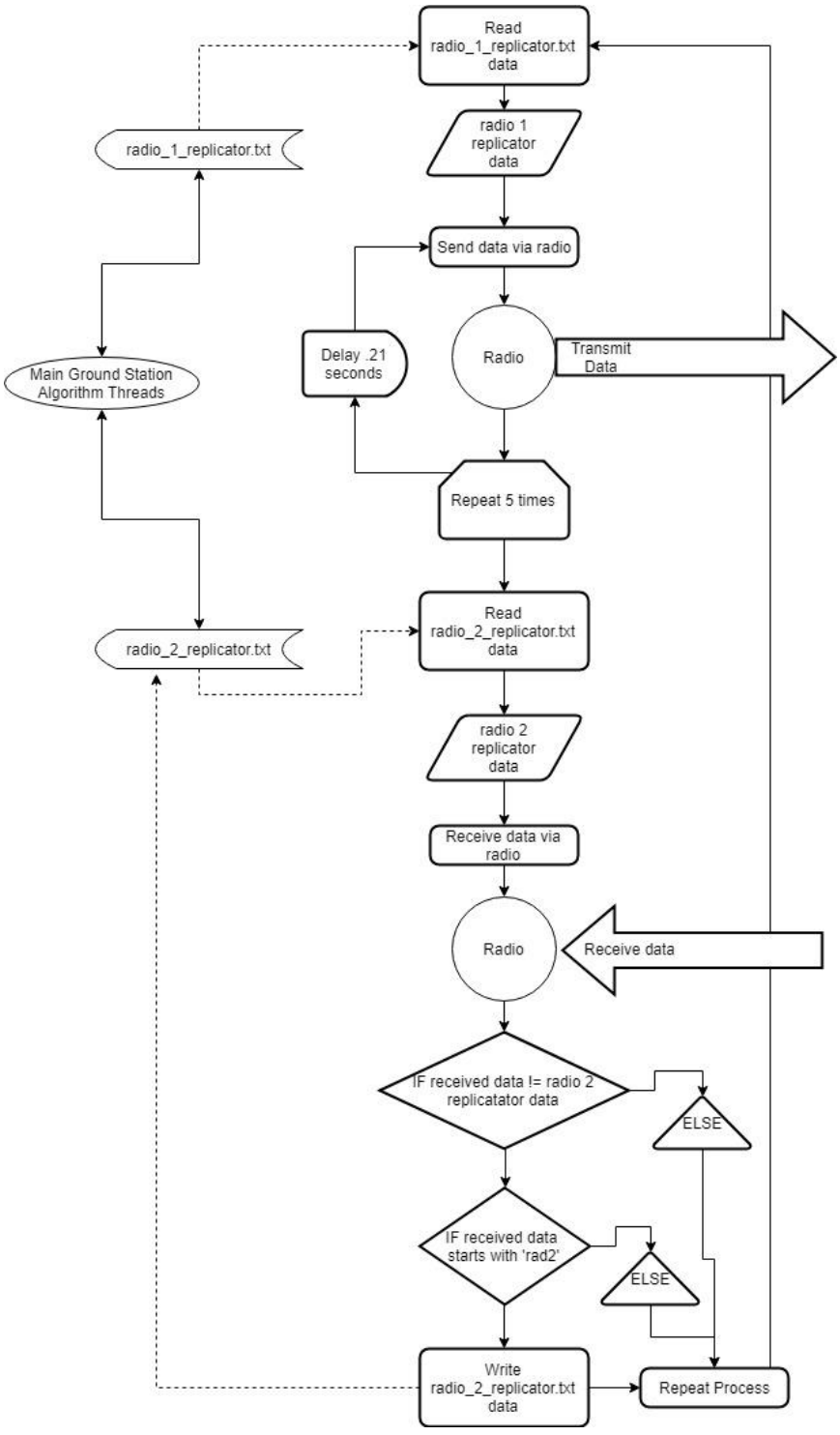

Figure A.47. Diagram of Single Radio Process Sequence

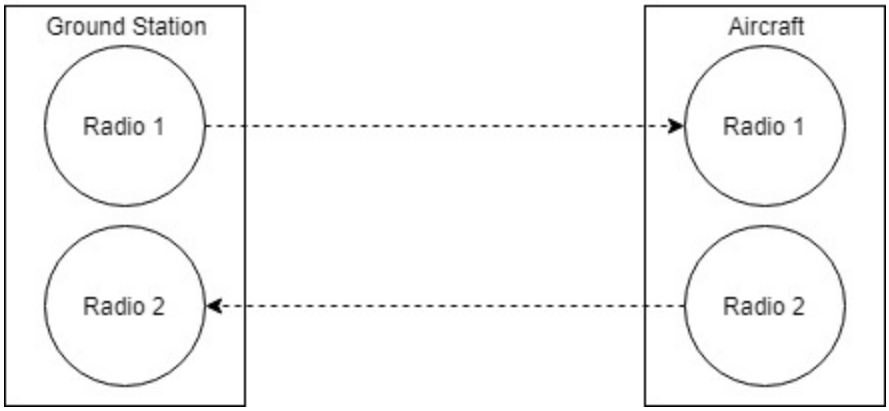

Figure A.48. Diagram of Double Radio One-Way Semi-Full-Duplex Communication Overview

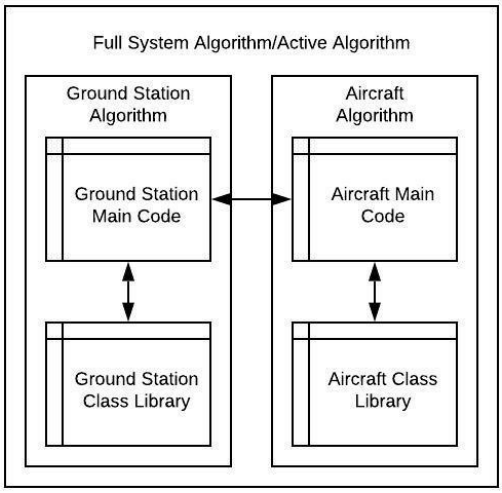

Figure A.49. Principle Full Algorithm Design

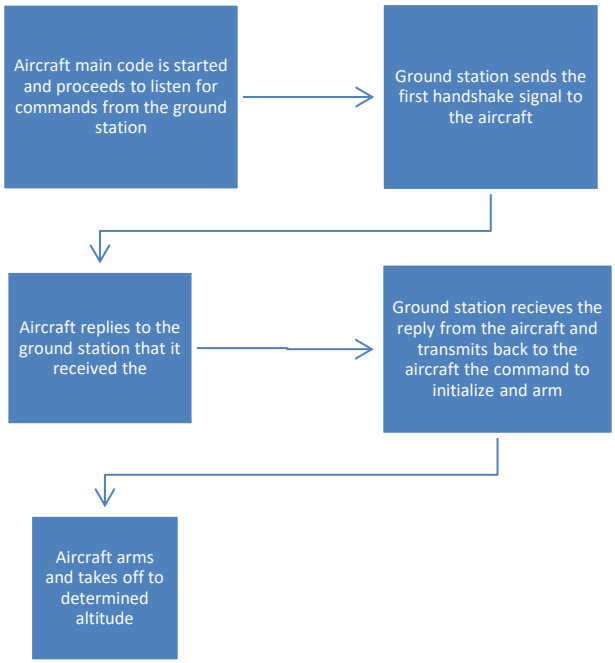

Figure A.50. Diagram of Startup Handshake Communications

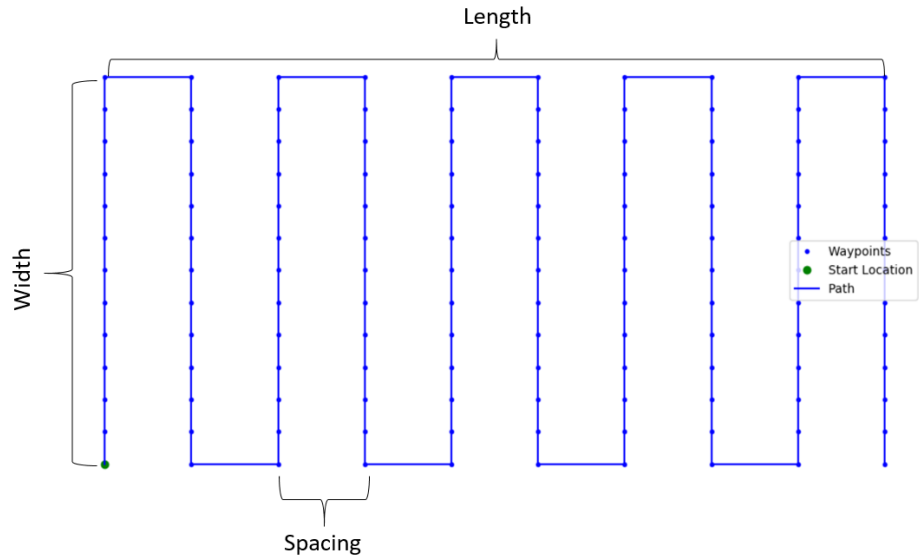

Figure A.51. Example of Predetermined Path for Tier 1 with Descriptions of Parts

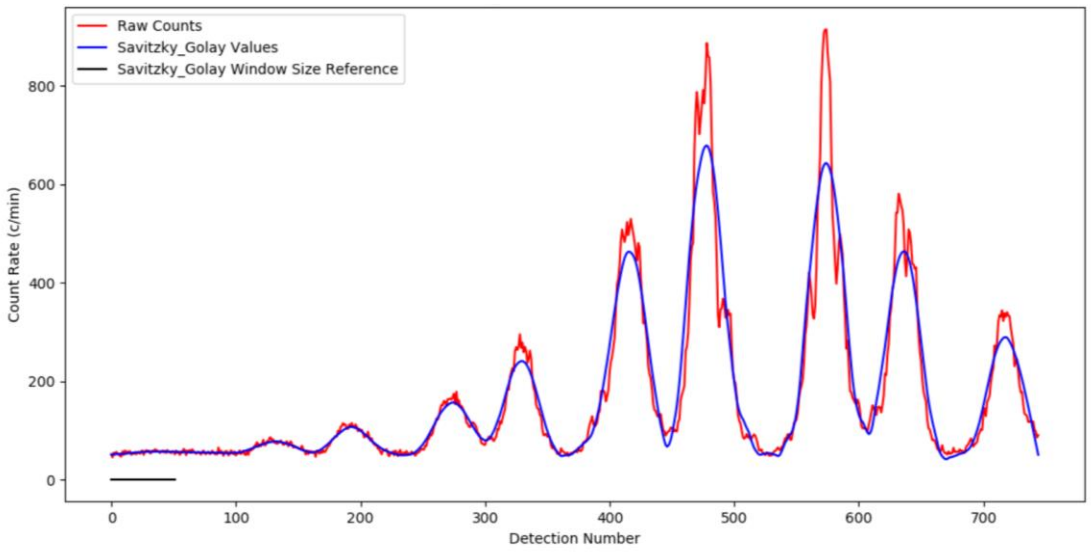

Figure A.52. Demonstration of S-G Filter on Raw Count Rate Data

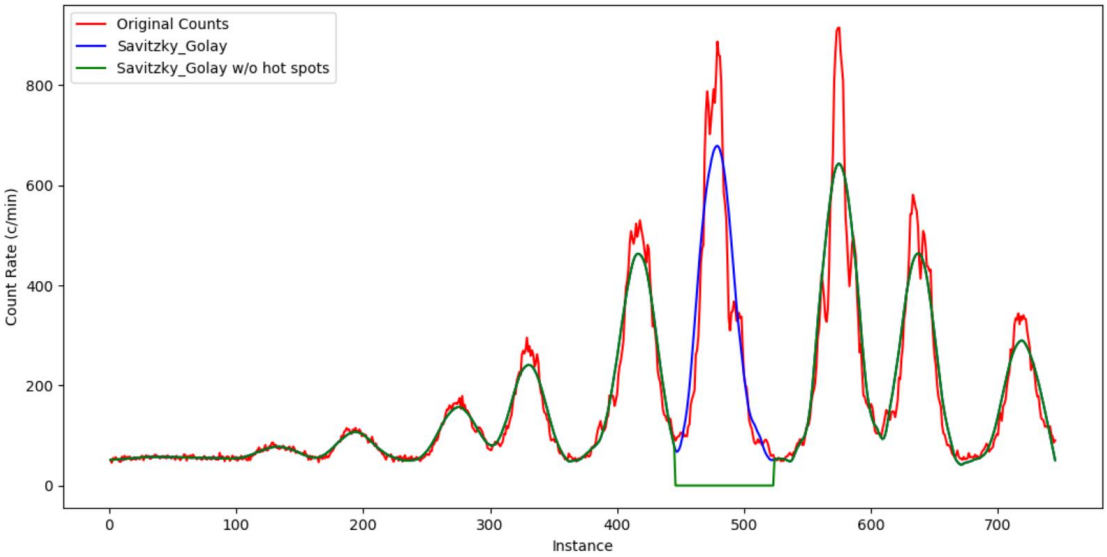

Figure A.53. Removal of Data around First Hot Spot

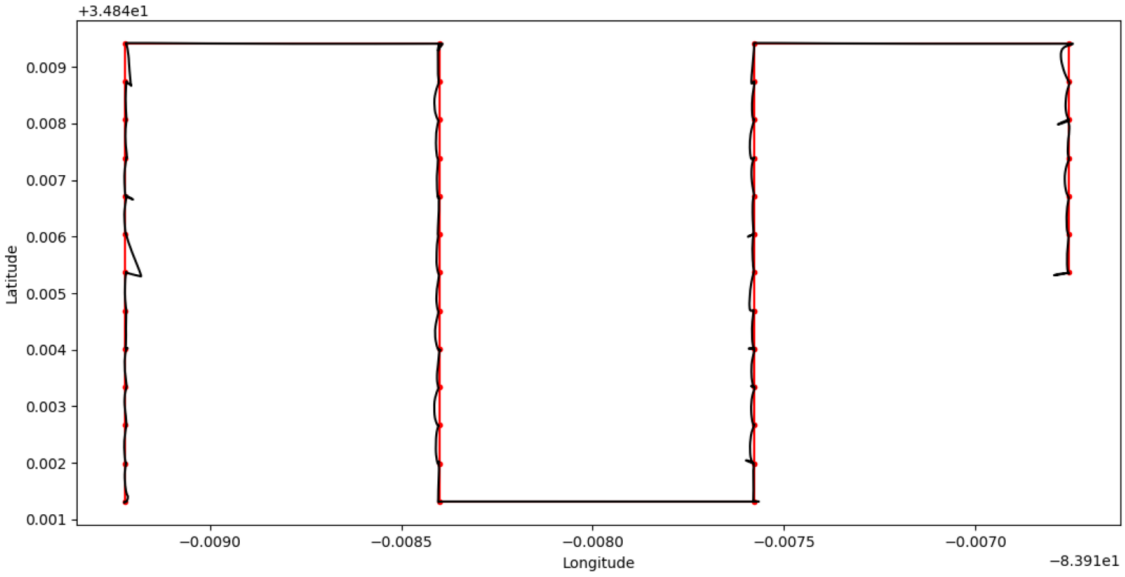

Figure A.54. Comparison of Flight Path Design (red) and Actual Flight Path Flown (black) for Numerous Waypoints

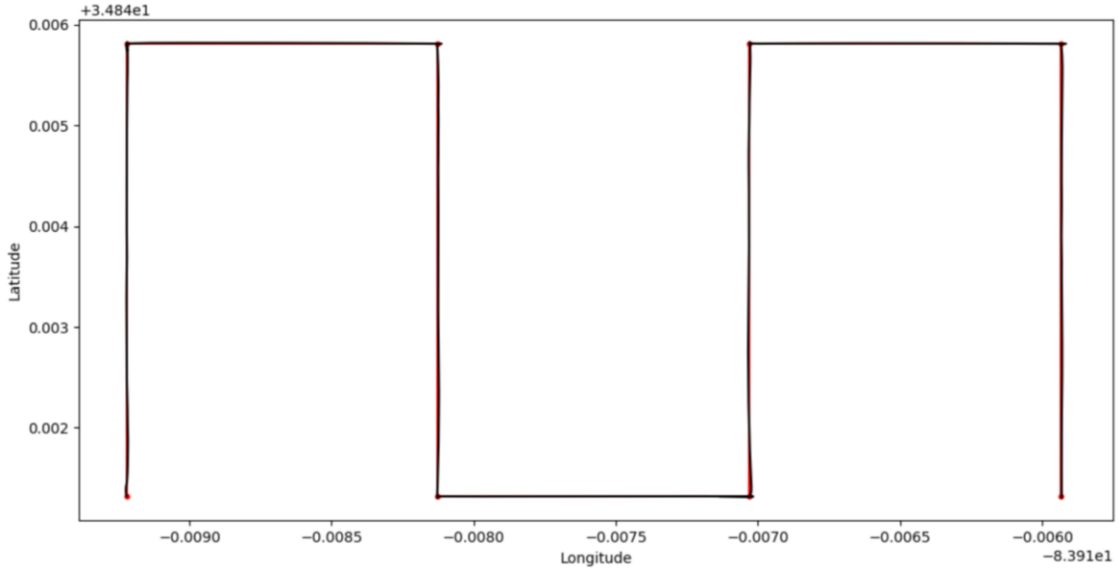

Figure A.55. Actual Flight Path Using Minimum Waypoints

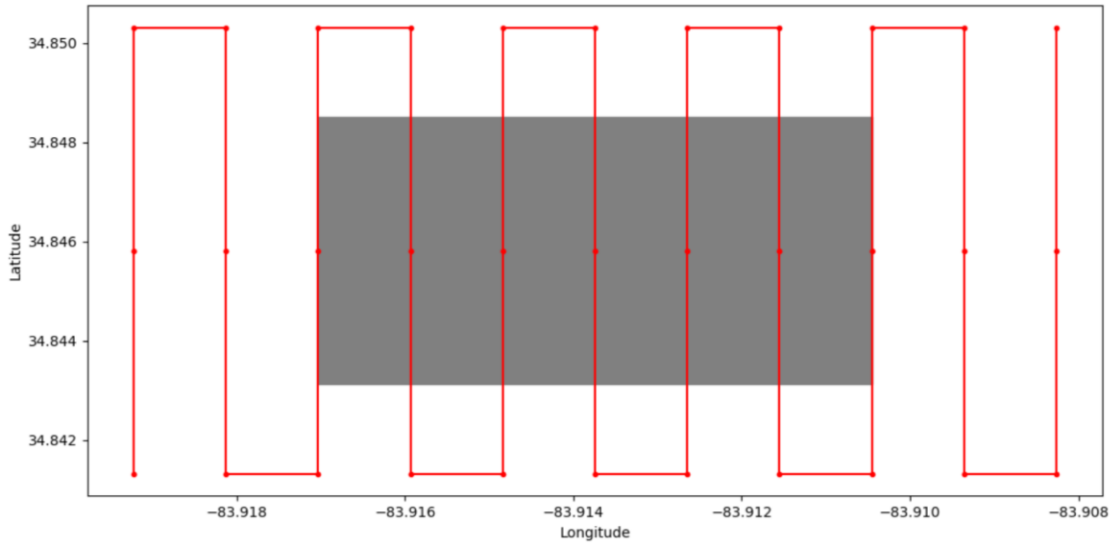

Figure A.56. Flight Path Design for Tier 1 Experiments and Possible Source Generation Region

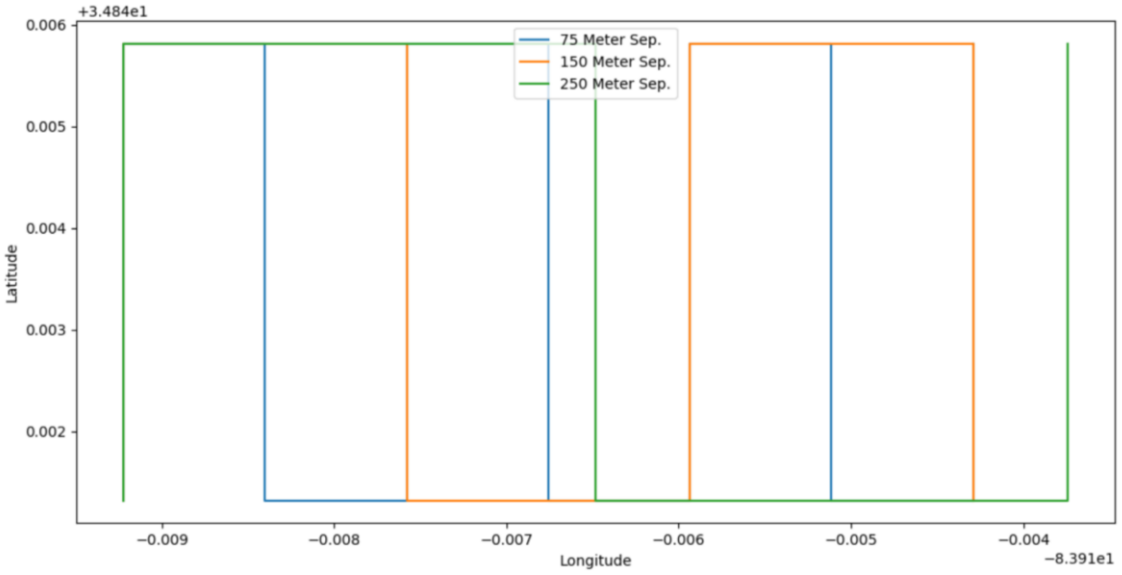

Figure A.57. Three of the Test Paths with 75, 150, and 250 Meter X-Separation Distances Used in S-G Filter Window Size Determinations

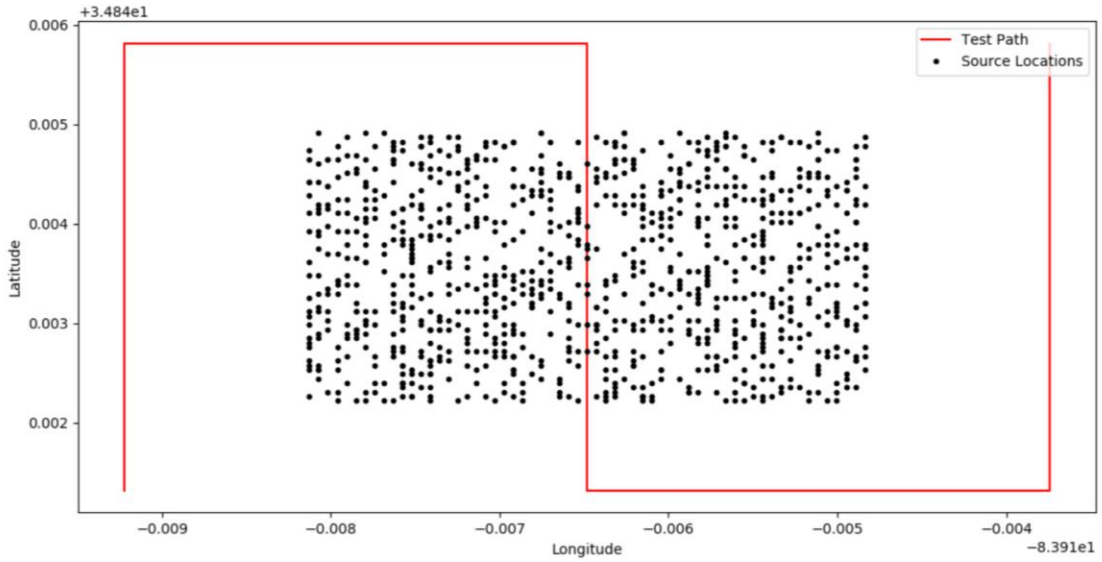

Figure A.58. Example of a 250 Meter X-Spaced Path's Source Locations Tested

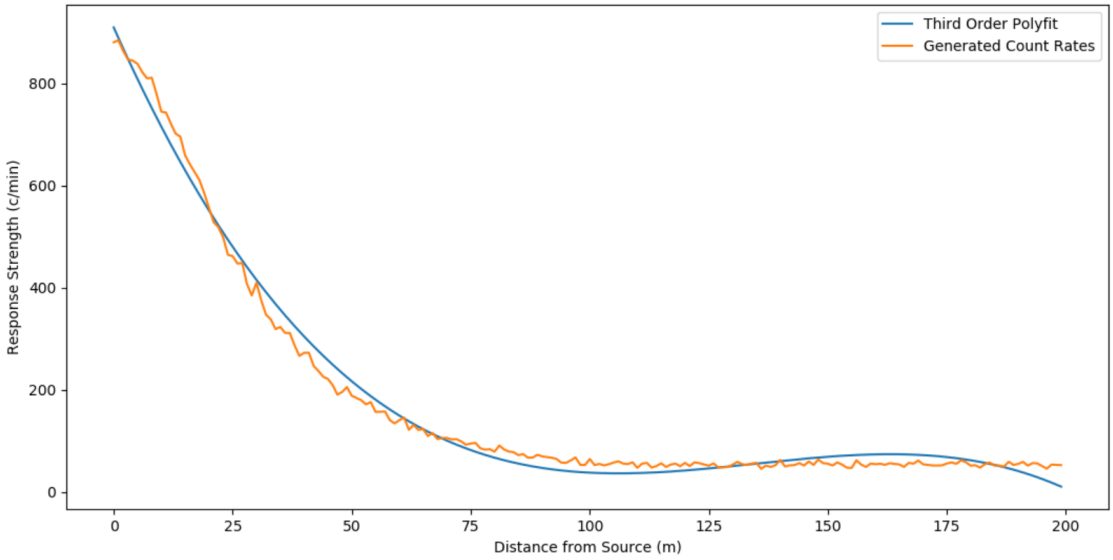

Figure A.59. Example of Code Generated Count Rates with Increasing Distance from the Source Compared to a Third Order Polynomial Curve Fit

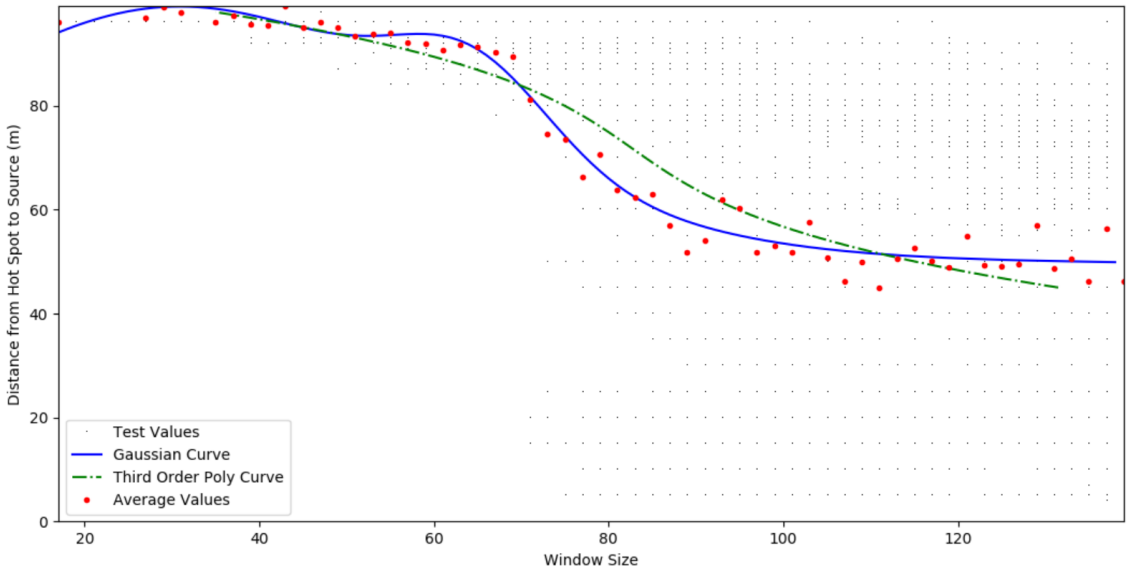

Figure A.60. S-G Filter Test Data and Results

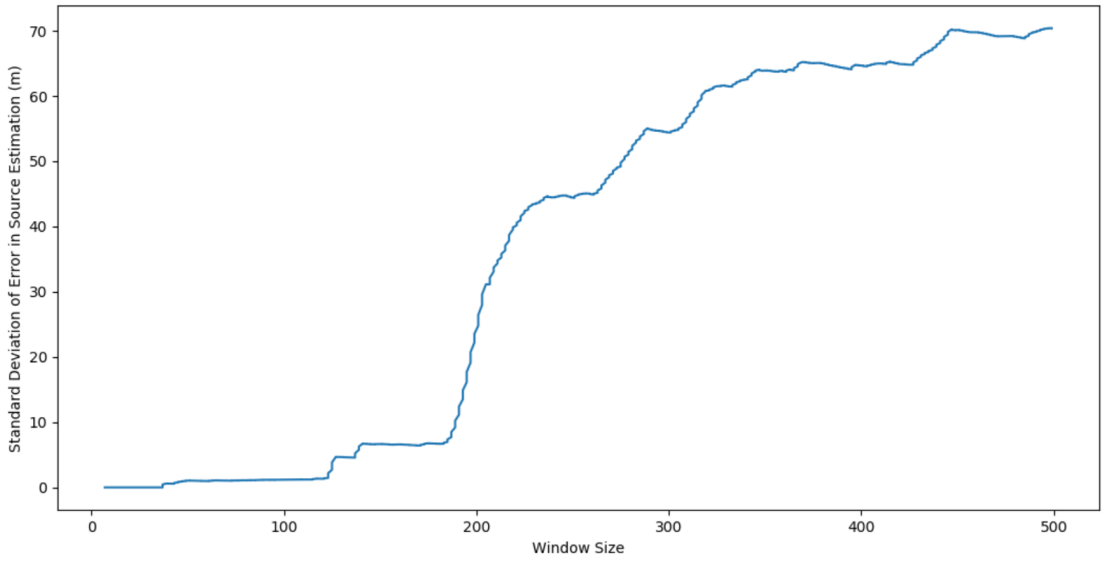

Figure A.61. Comparison of Progressive Standard Deviation of Hot Spot Distance Curve and Window Size for S-G Filter Technique

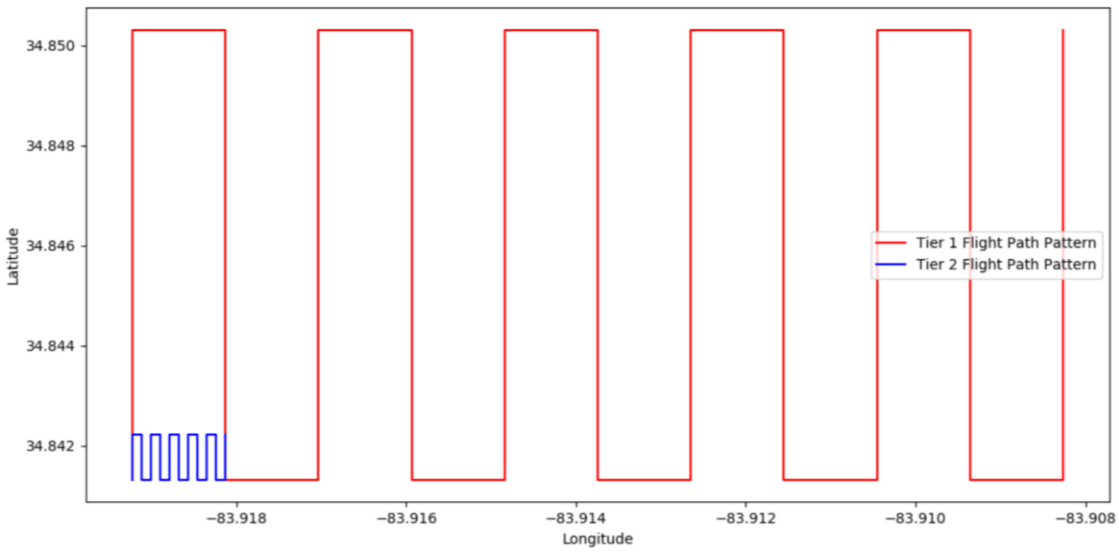

Figure A.62. Tier 1 and Tier 2 Flight Path Comparison

```
1.  bys.Add_values(array)#send the array to be appropriately appended in the
    Bayesian class
2.  count=count+1 #increase the iteration by 1
3.  if count>2 and (count%10)==0:
4.      bys.LogLike()#place values into log and perform bayesian calculations
5.      bys.Update()#update prior and posterior values
6.      ary_vals=bys.Exp() #take values (especially probability values) out of log
```

Figure A.63. Framework of Bayesian Code

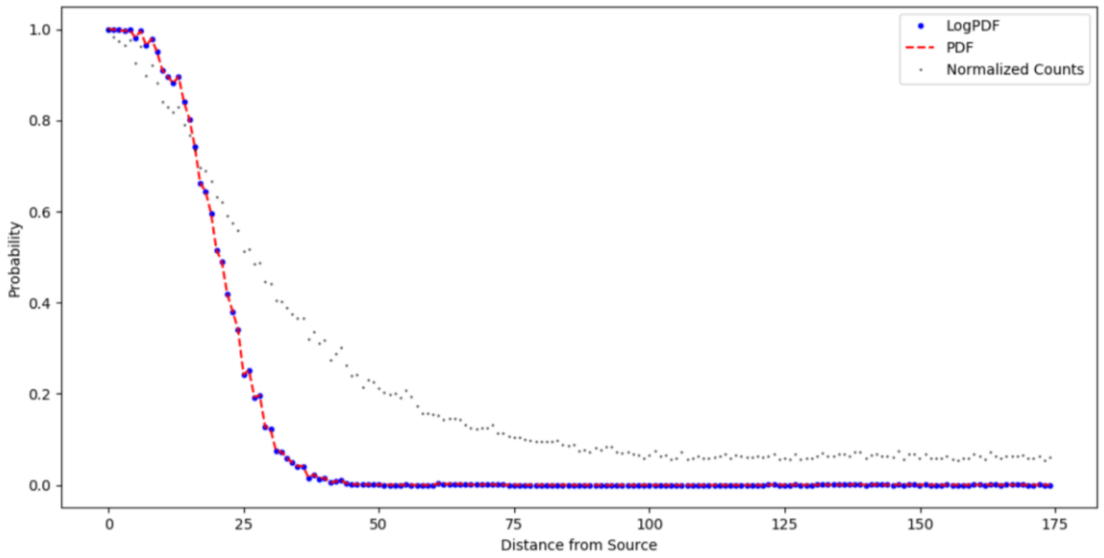

Figure A.64. Comparison of Log Normal PDF and Normal PDF Result Moving Away from a Source

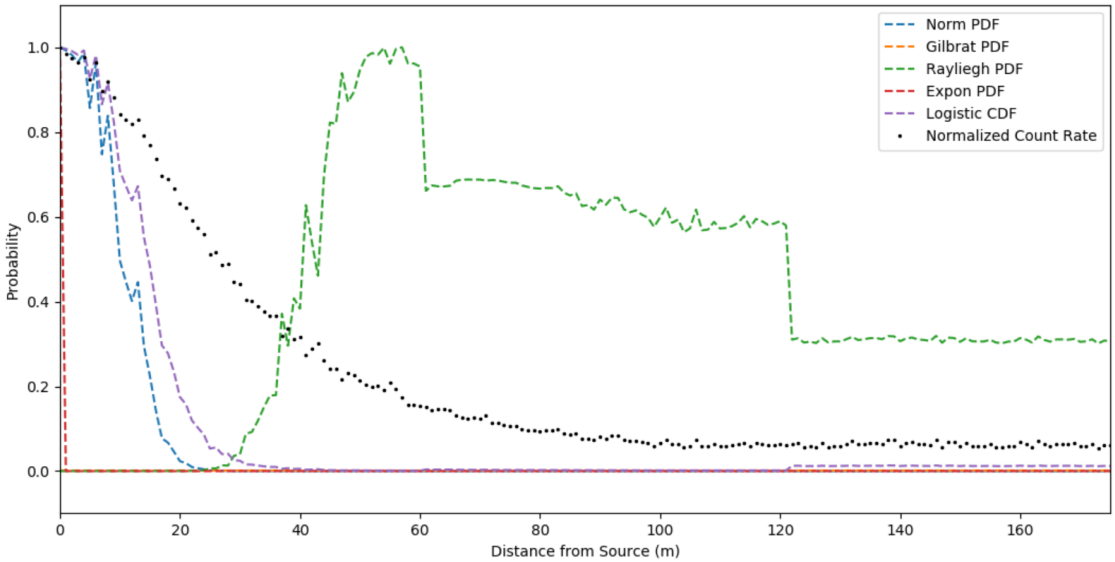

Figure A.65. Comparison of Different Types of PDF Results Moving Away from a Source

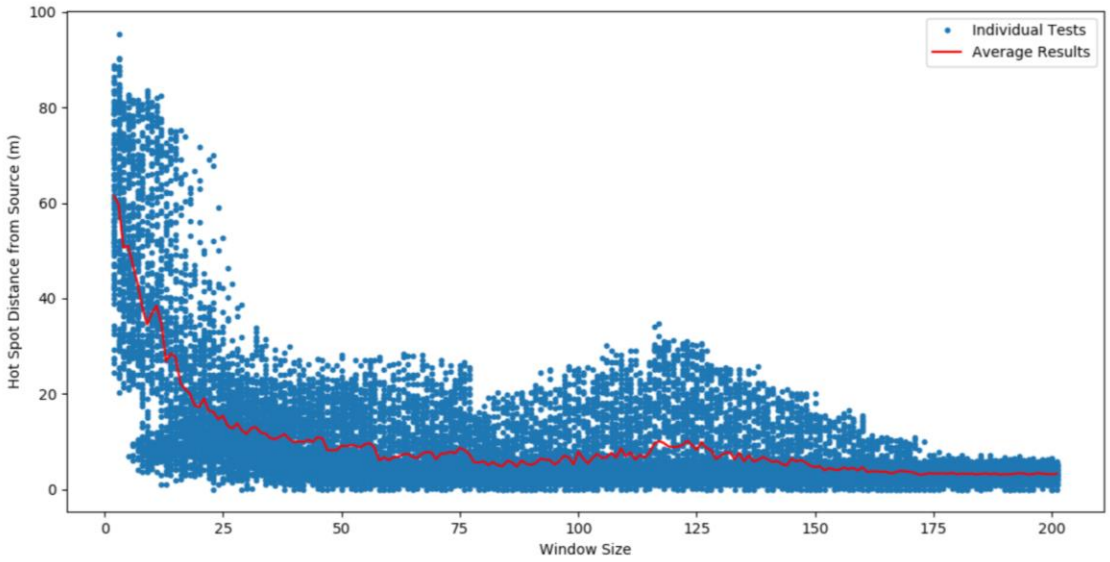

Figure A.66. Individual Test Hot Spot Accuracy and Average Window Test Accuracy with Respect to Distance from Source for 100m x 100m Search Space

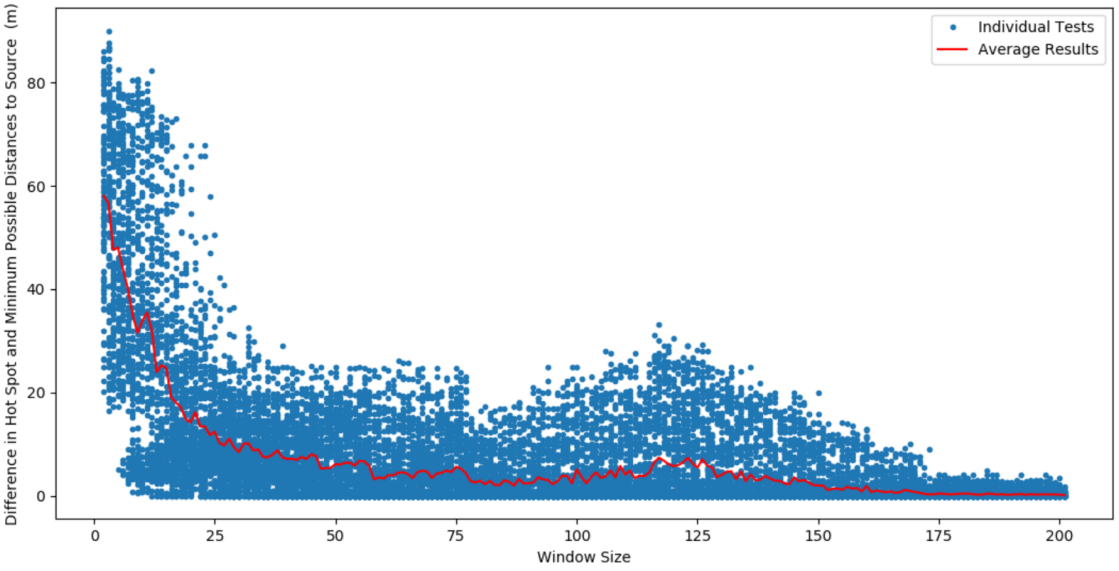

Figure A.67. Difference Between Hot Spot Distance from Source and Minimum Path Distance from Source for each Individual Test and each Window Test Average for 100m x 100m Search Space

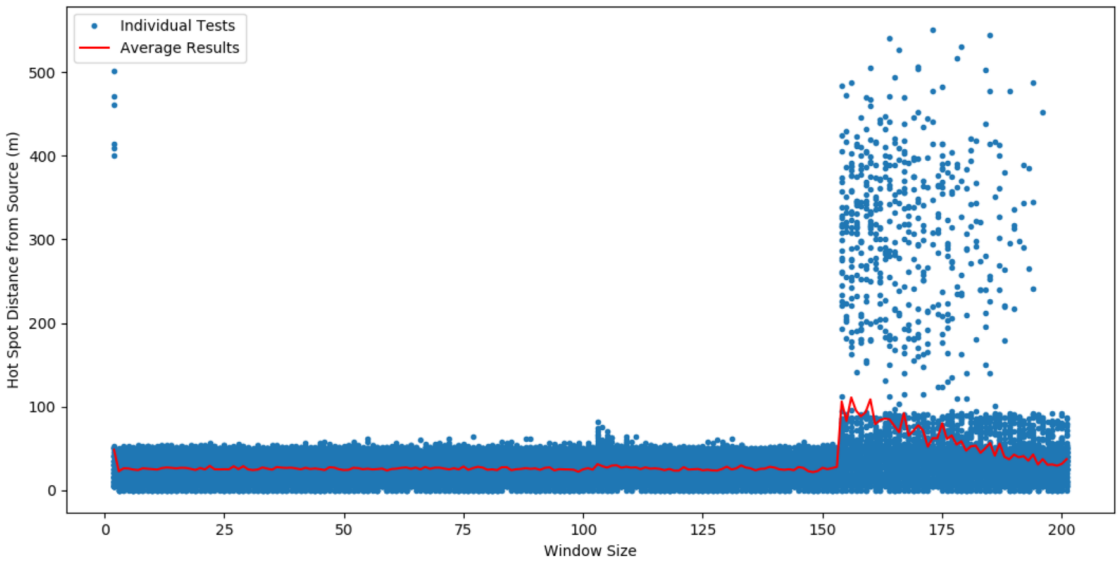

Figure A.68. Individual Test Hot Spot Accuracy and Average Window Test Accuracy with Respect to Distance from Source for 500m x 500m Search Space

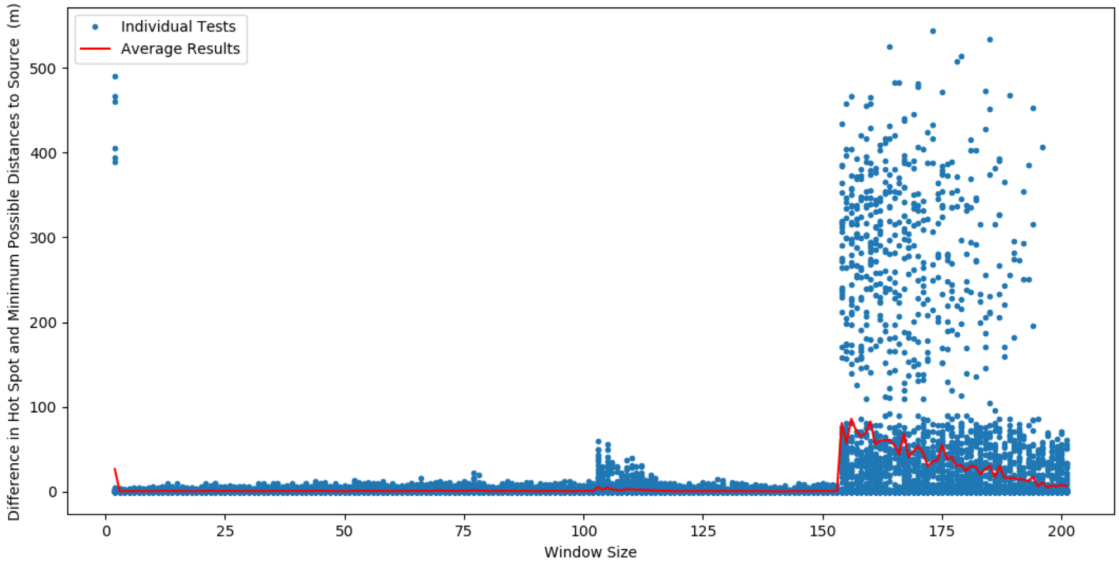

Figure A.69. Difference Between Hot Spot Distance from Source and Minimum Path Distance from Source for each Individual Test and each Window Test Average for 500m x 500m Search Space

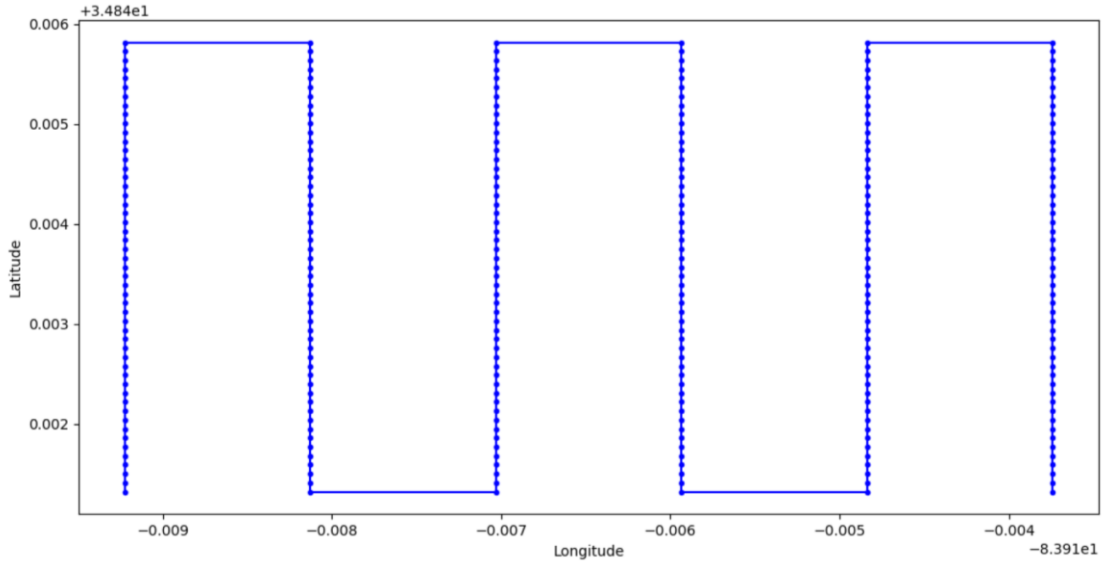

Figure A.70. Generated Flight Path of Dimensions 500m x 500m and Waypoints with Delta x of 10m and Delta y of 100m

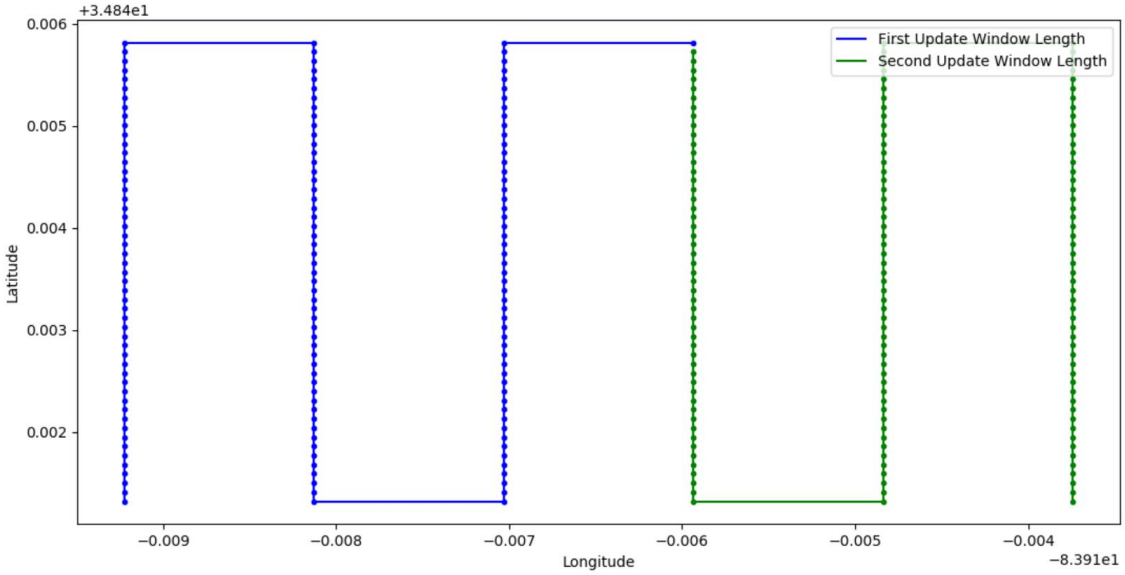

Figure A.71. Generated Flight Path of Dimensions 500m x 500m and Waypoints with Delta x of 10m and Delta y of 100m with Two Bayesian Updates Performed

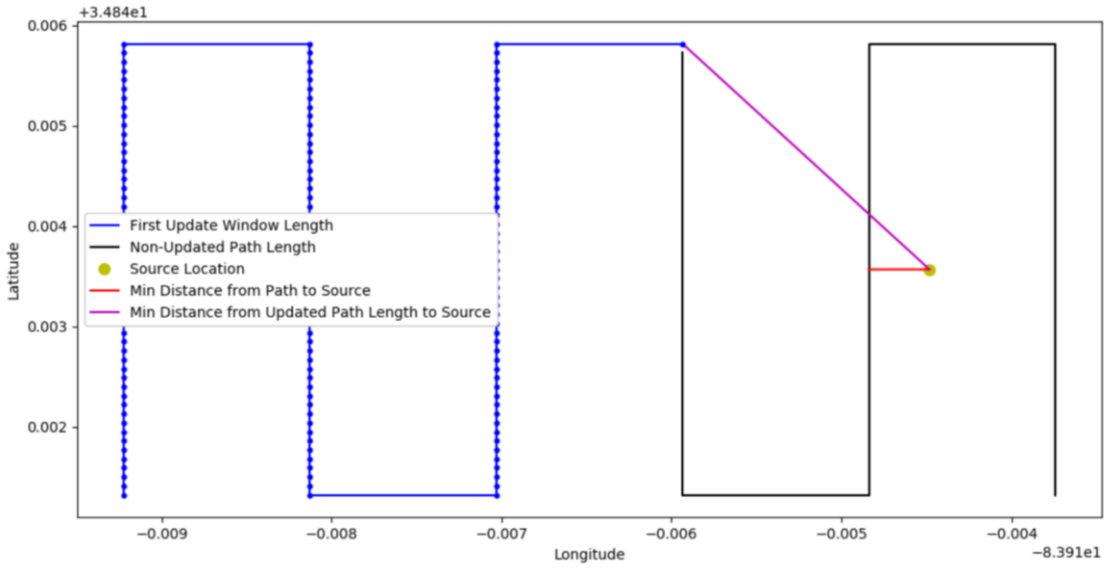

Figure A.72. Generated Flight Path of Dimensions 500m x 500m and Waypoints with Delta x of 10m and Delta y of 100m with One Update Performed and the Minimum Distances from the Hot Spot to the Source and the Path to the Source Shown.

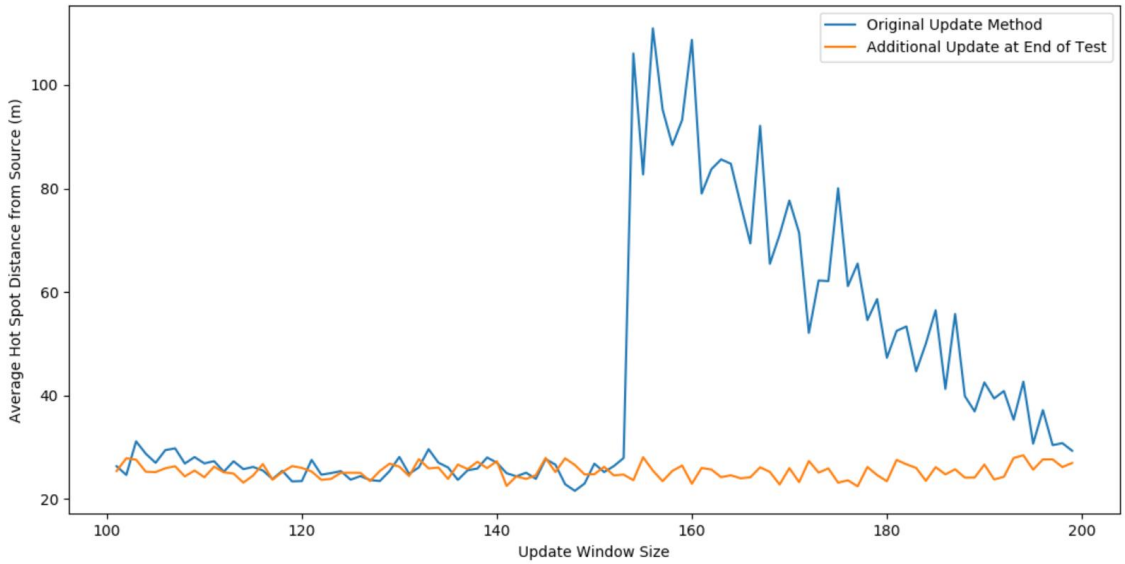

Figure A.73. Display of Hot Spot Error Resolution by using an Additional Update at the End of each Test

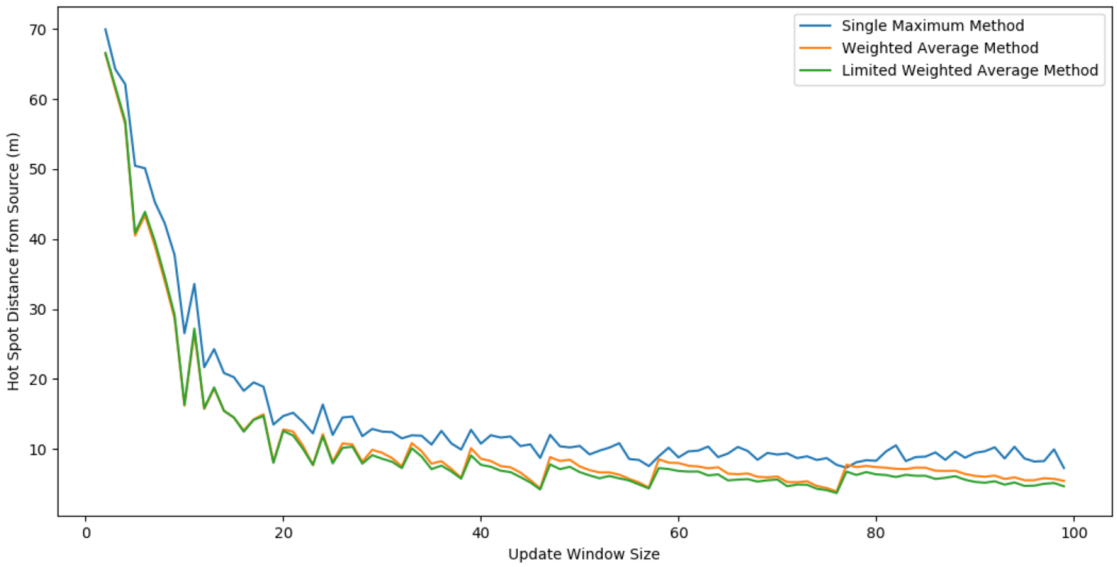

Figure A.74. Comparison of Different Hot Spot Location Estimation Methods

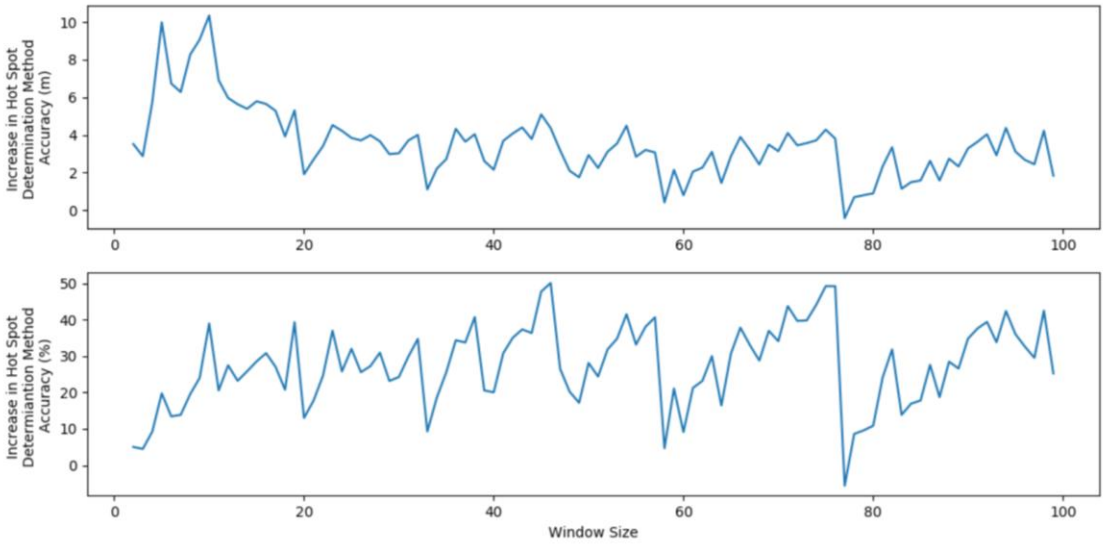

Figure A.75. Increase in Accuracy of Full Weighted Averaging Hot Spot Determination Method

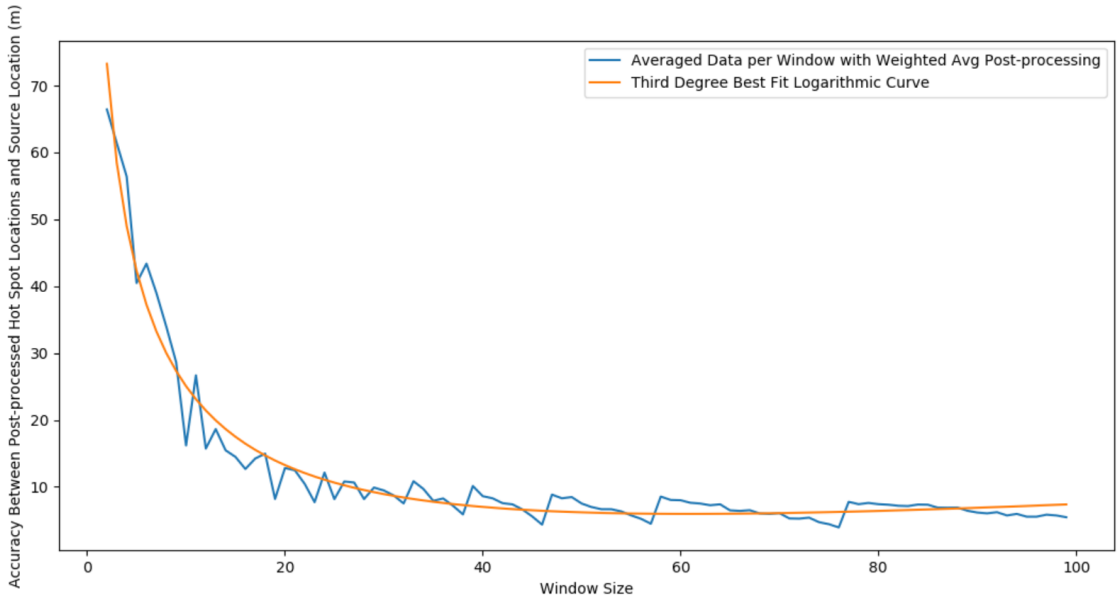

Figure A.76. Post-processed Hot Spot Accuracy Curve Compared to Third Order Logarithmic Best-Fit Curve

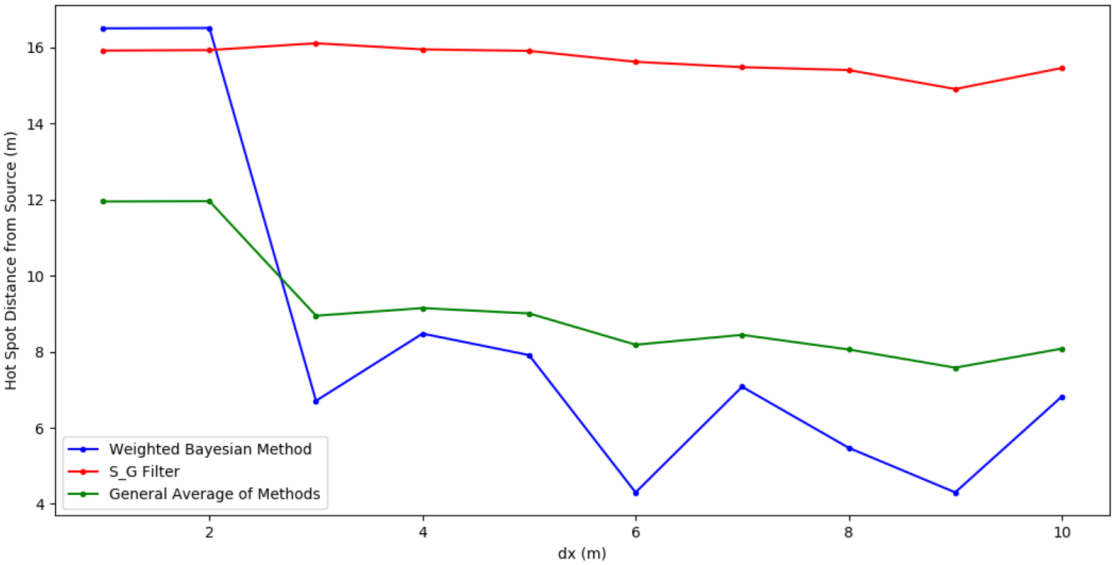

Figure A.77. Hot Spot Determination Method Comparison for Tier 2 Search

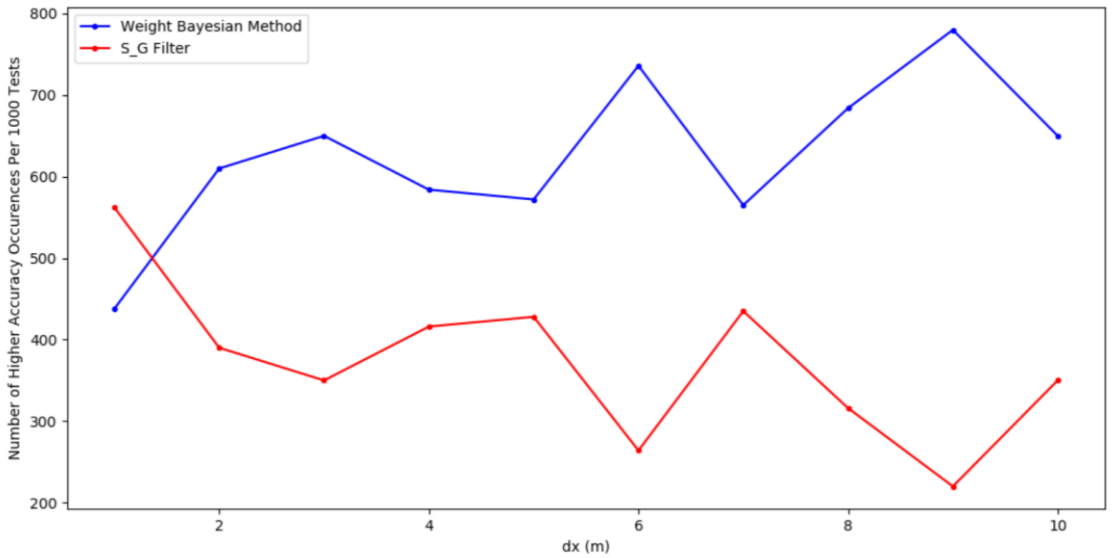

Figure A.78. Number of Times per Thousand Tests that each Hot Spot Determination Method is the Most Accurate

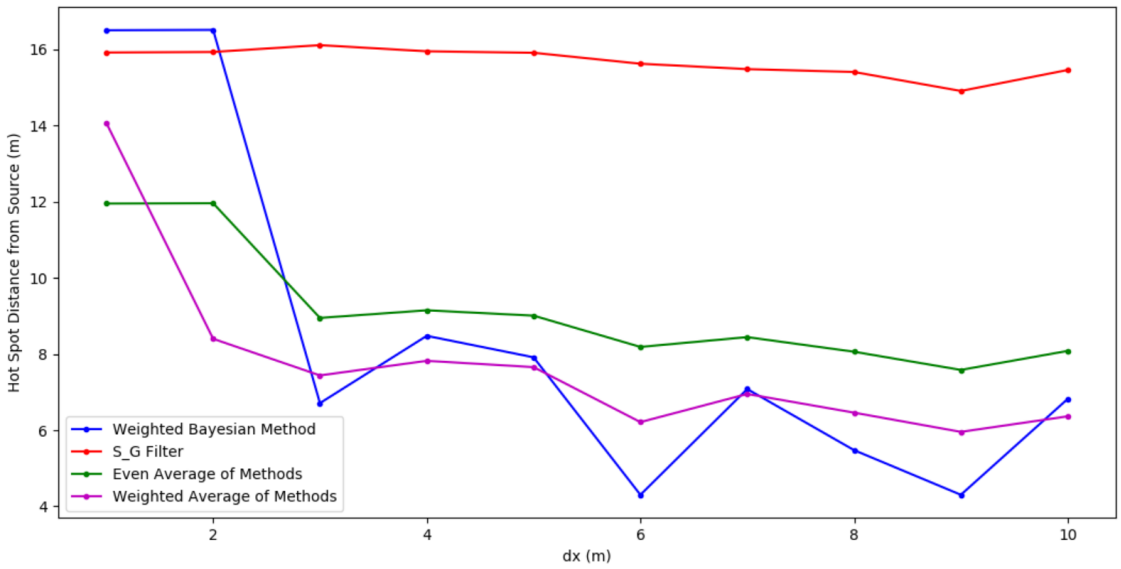

Figure A.79. Hot Spot Determination Method Comparison for Tier 2 with Weighted Average of Methods Data

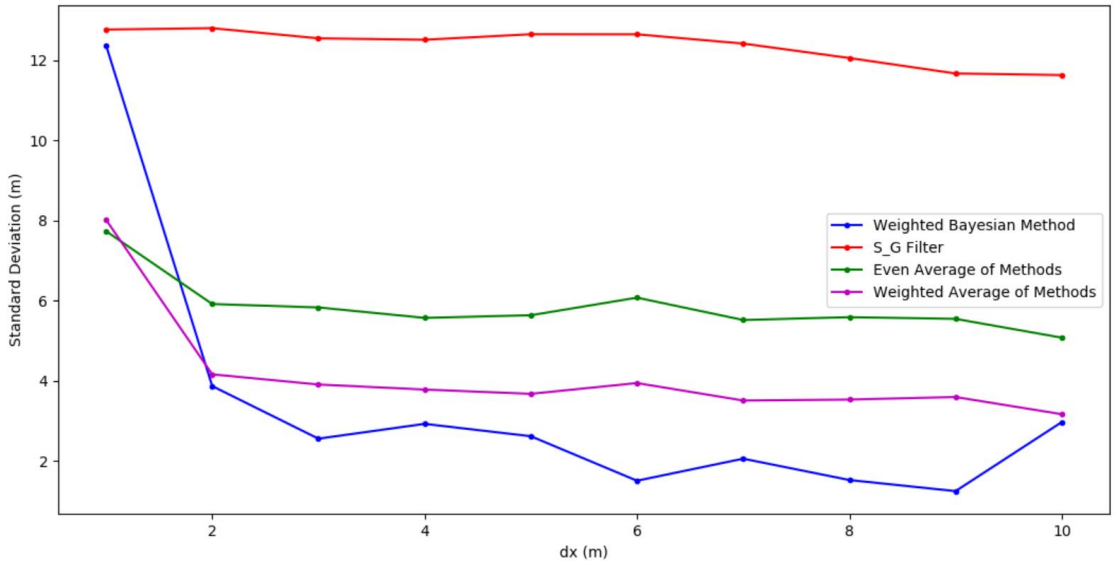

Figure A.80. Standard Deviation Trends for each Considered Hot Spot Locator Method

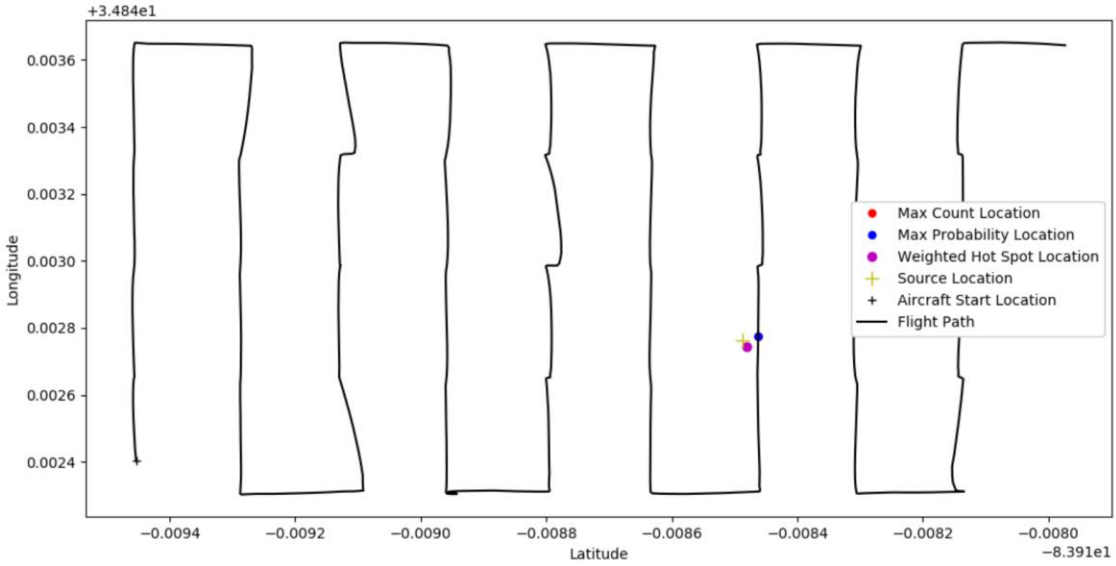

Figure A.81. Standard Deviation Trends for each Considered Hot Spot Locator Method

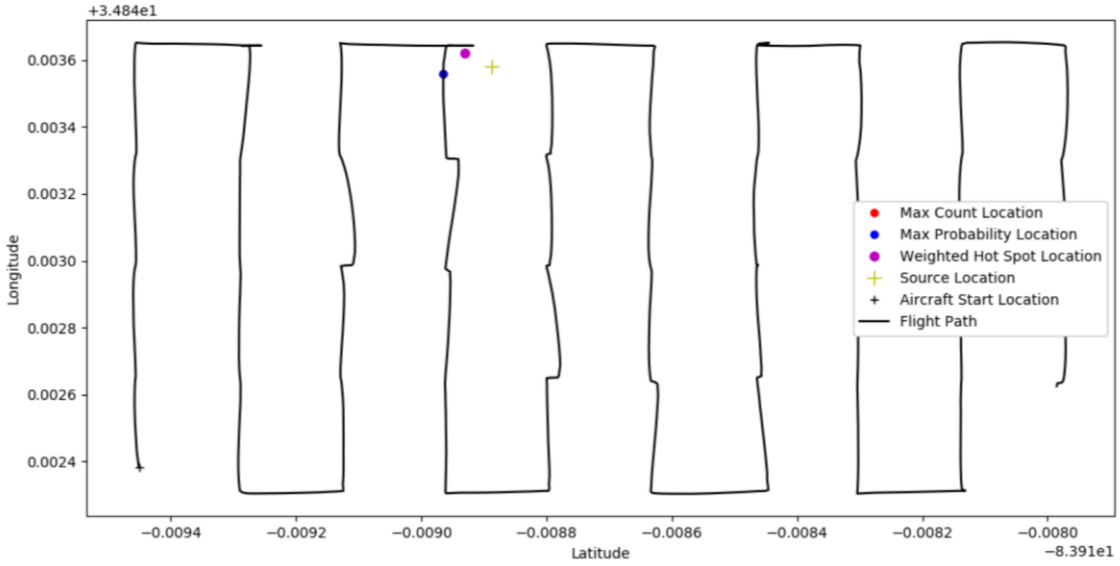

Figure A.82. Standard Deviation Trends for each Considered Hot Spot Locator Method

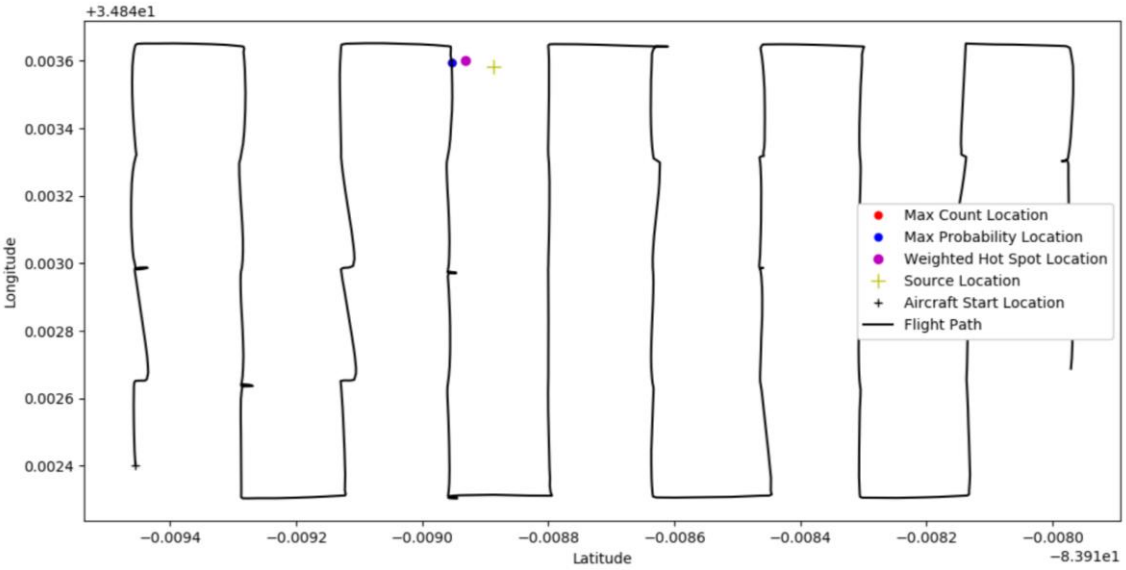

Figure A.83. Standard Deviation Trends for each Considered Hot Spot Locator Method

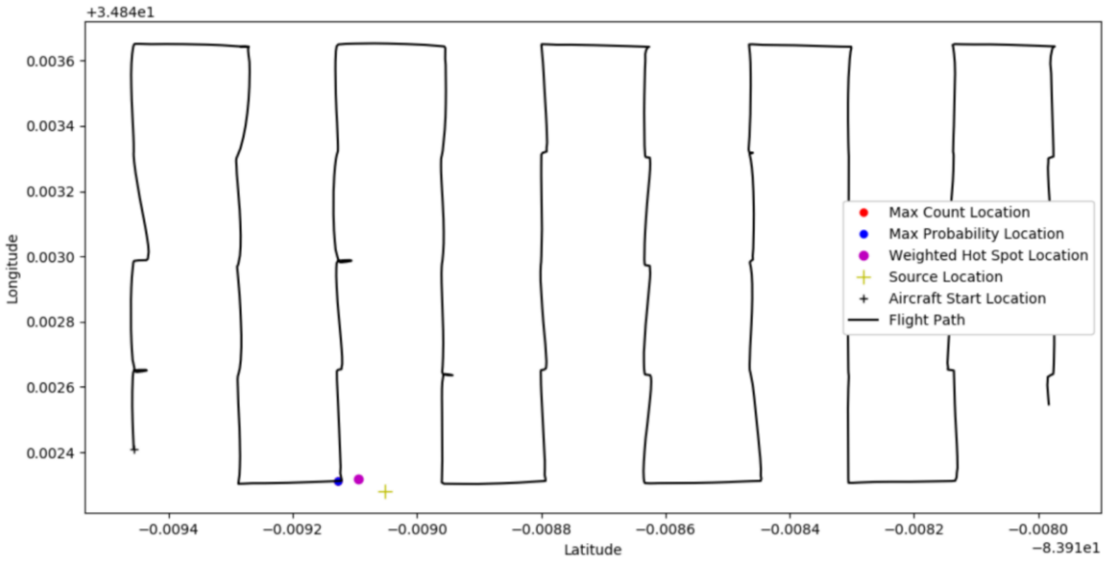

Figure A.84. Standard Deviation Trends for each Considered Hot Spot Locator Method

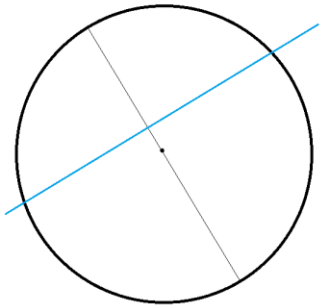

Figure A.85. Geometric Description of RAT Method Premise

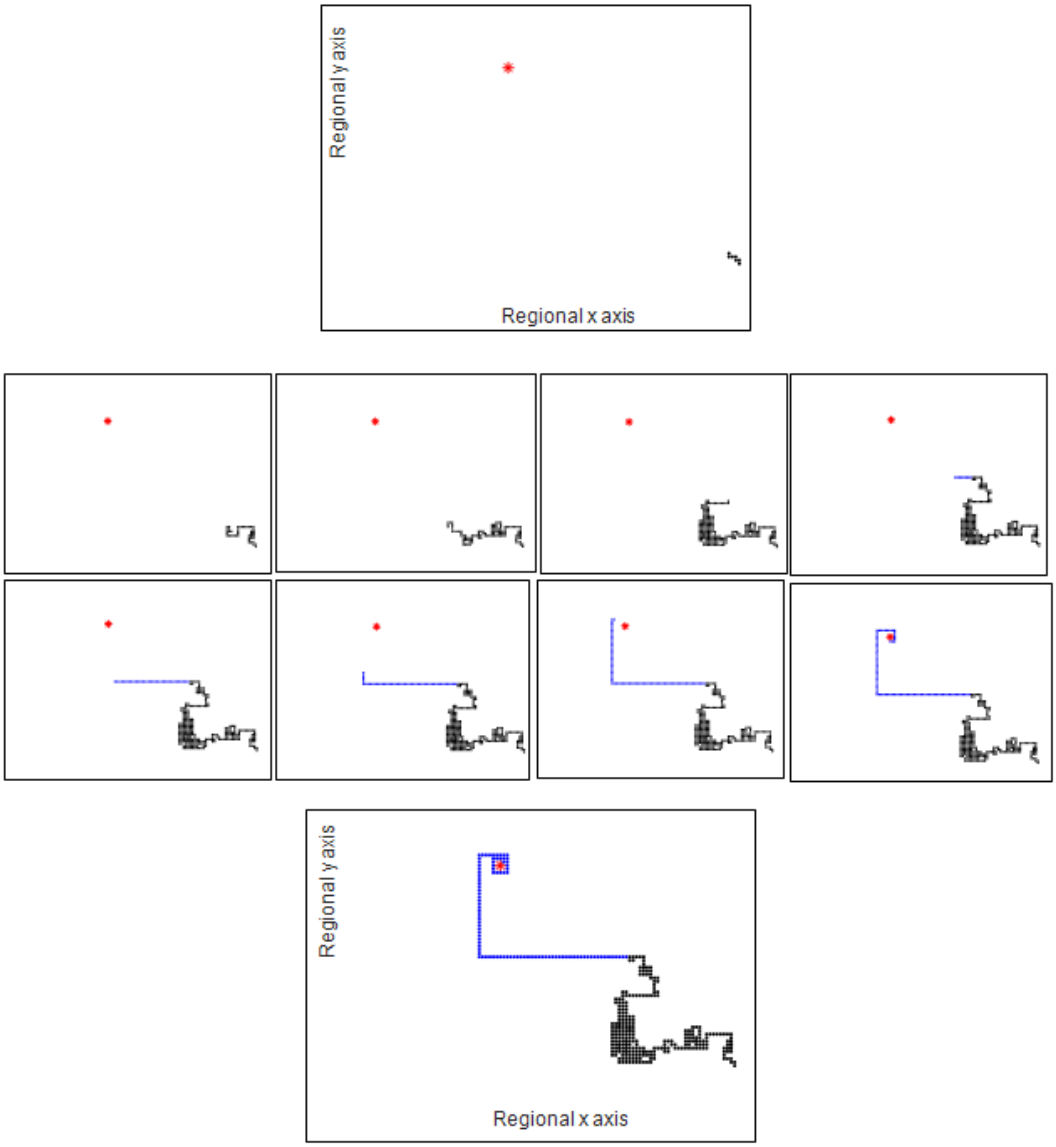

Figure A.86. Example of RAT Method in Action using Perfect Theoretical Data

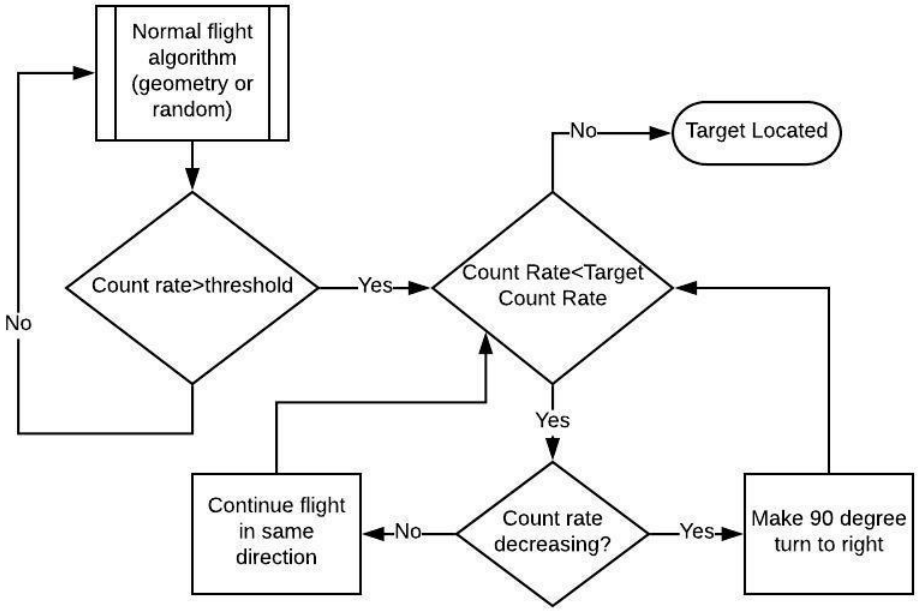

Figure A.87. Example of RAT Method in Fundamental Algorithm Processes

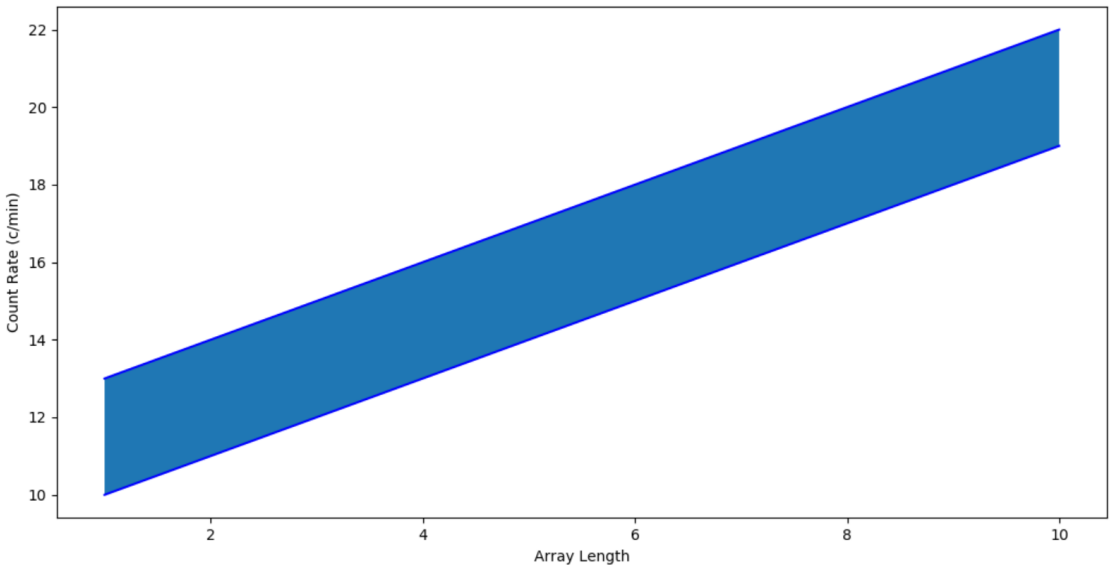

Figure A.88. Hypothetical Count Rate Increase Example over a Maximum Recorded Array Length of 10

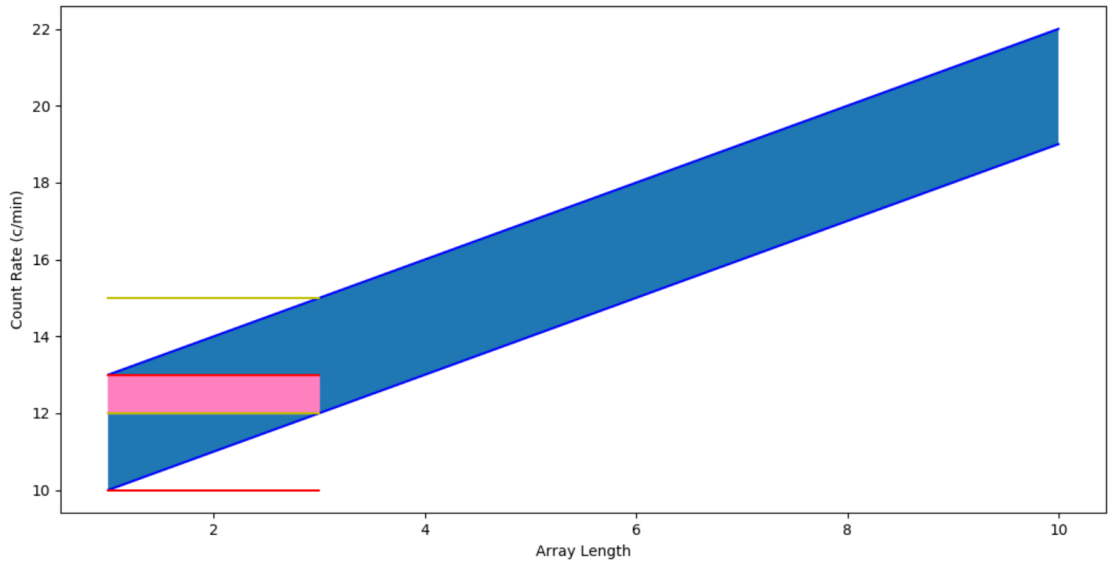

Figure A.89. Overlap of Potential Count Rates for a Hypothetical Example with a Recorded Array Length of 3

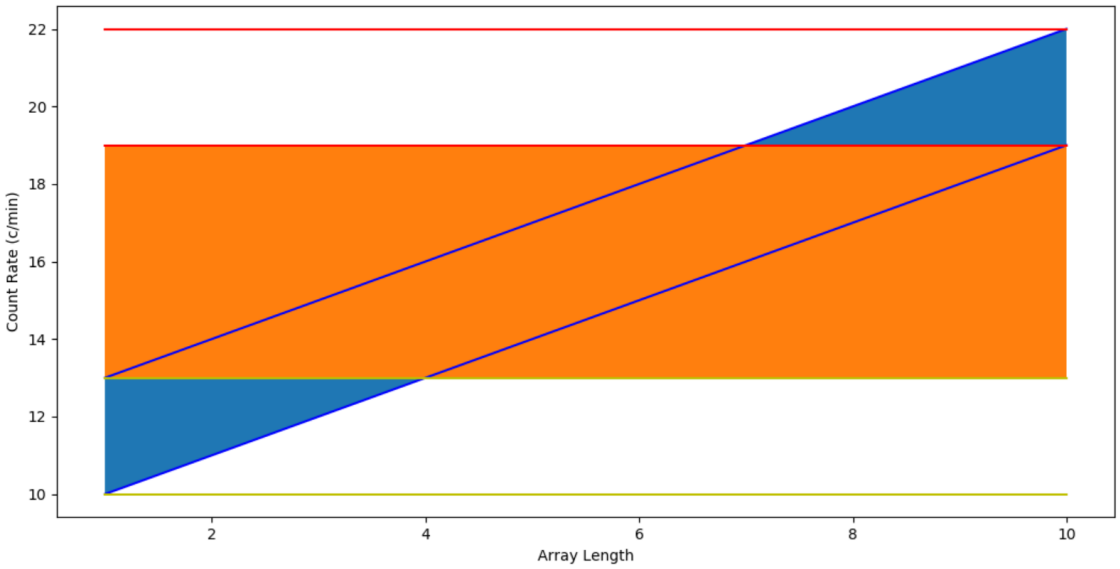

Figure A.90. Gap between Potential Count Rates for a Hypothetical Example with a Recorded Array Length of 10

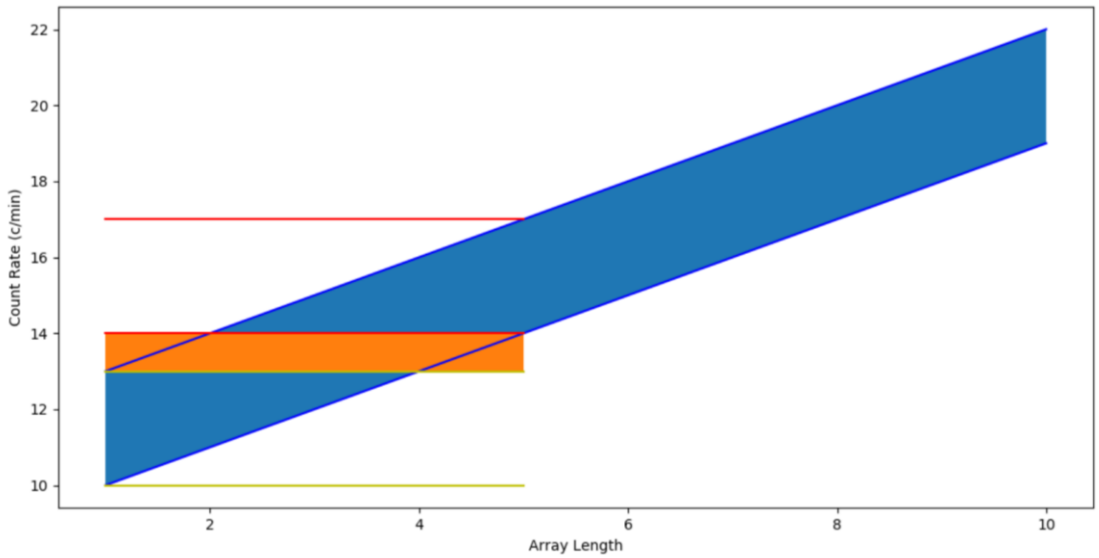

Figure A.91. Optimum Gap Size of Potential Count Rates for a Hypothetical Example with a Recorded Array Length of 5

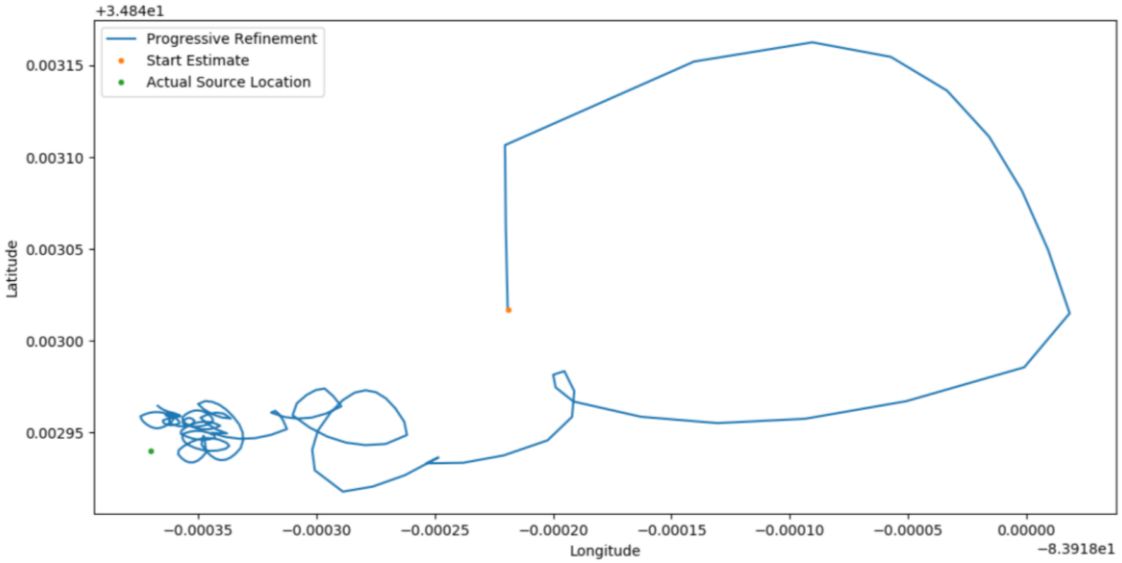

Figure A.92. Example from Real-World Simulation of Source Location Estimate Refinement

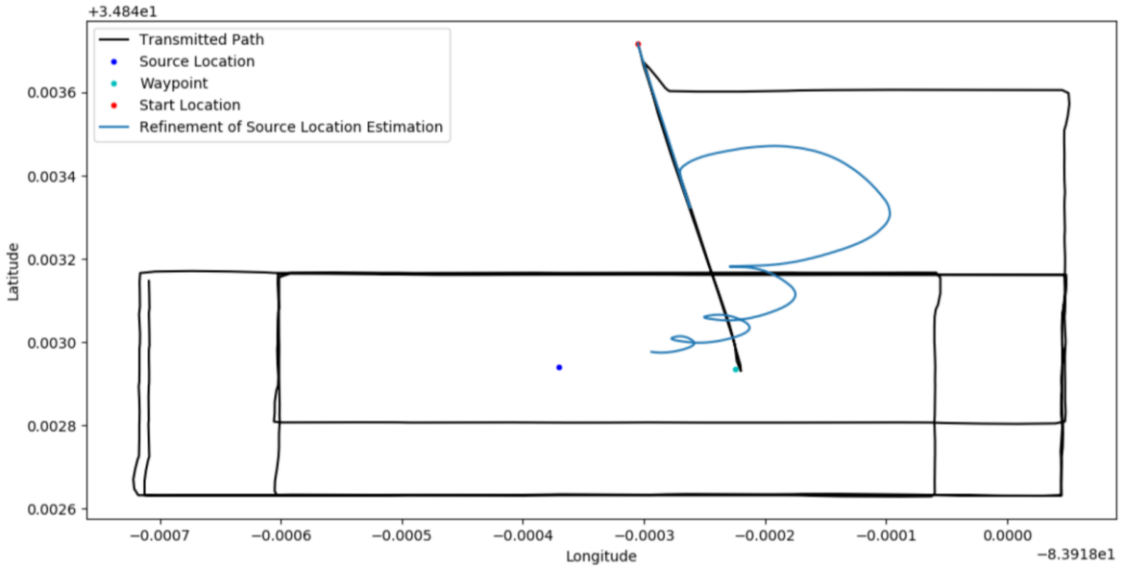

Figure A.93. Results from Simulation using Detection Array Length of 20 at an Altitude of 10m

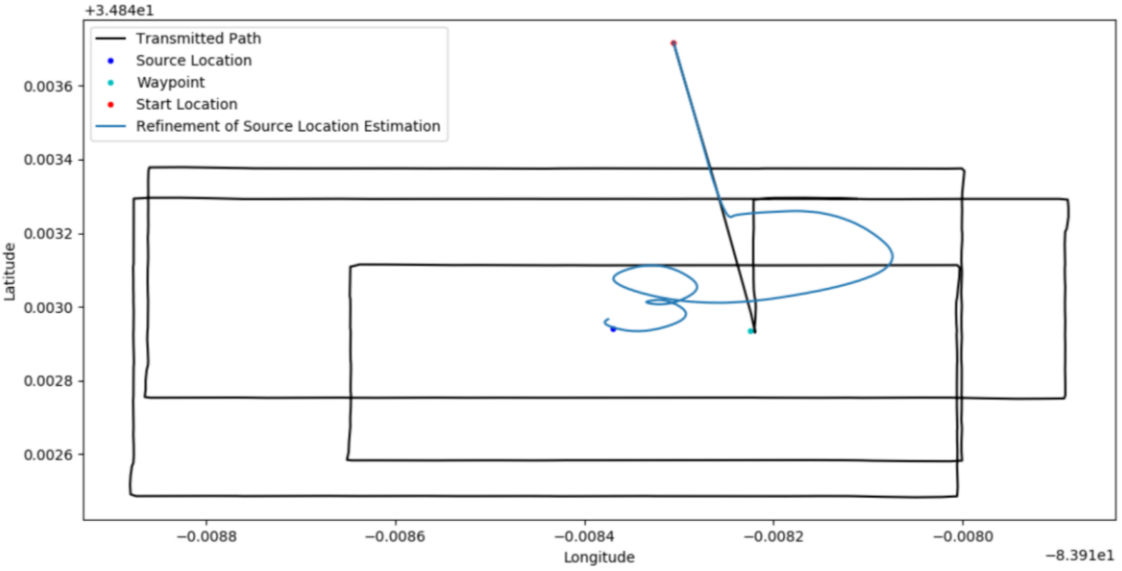

Figure A.94. Results from Simulation using Detection Array Length of 20 at an Altitude of 15m

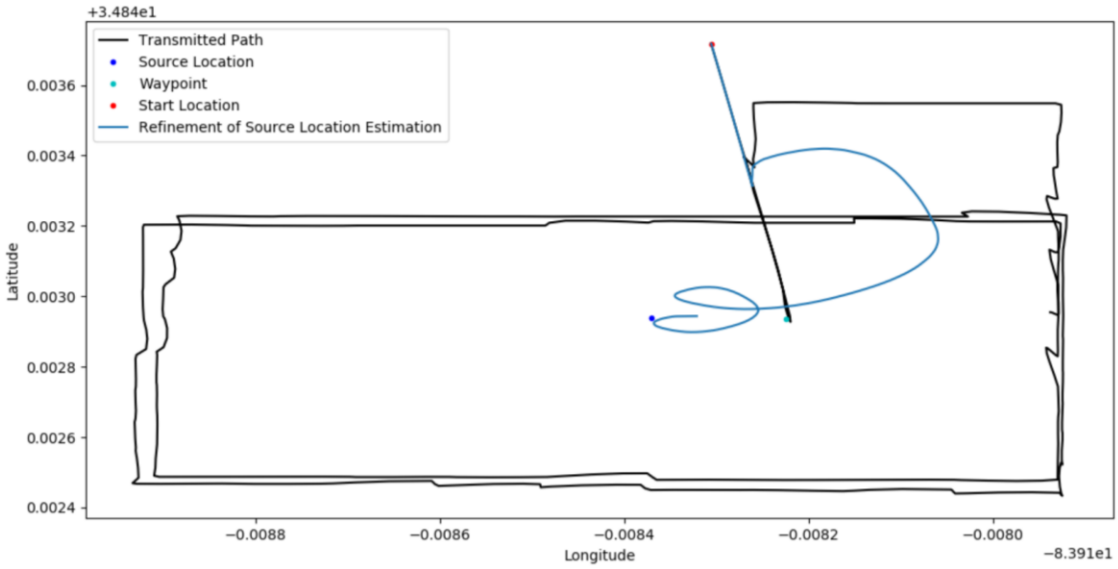

Figure A.95. Results from Simulation using Detection Array Length of 20 at an Altitude of 20m

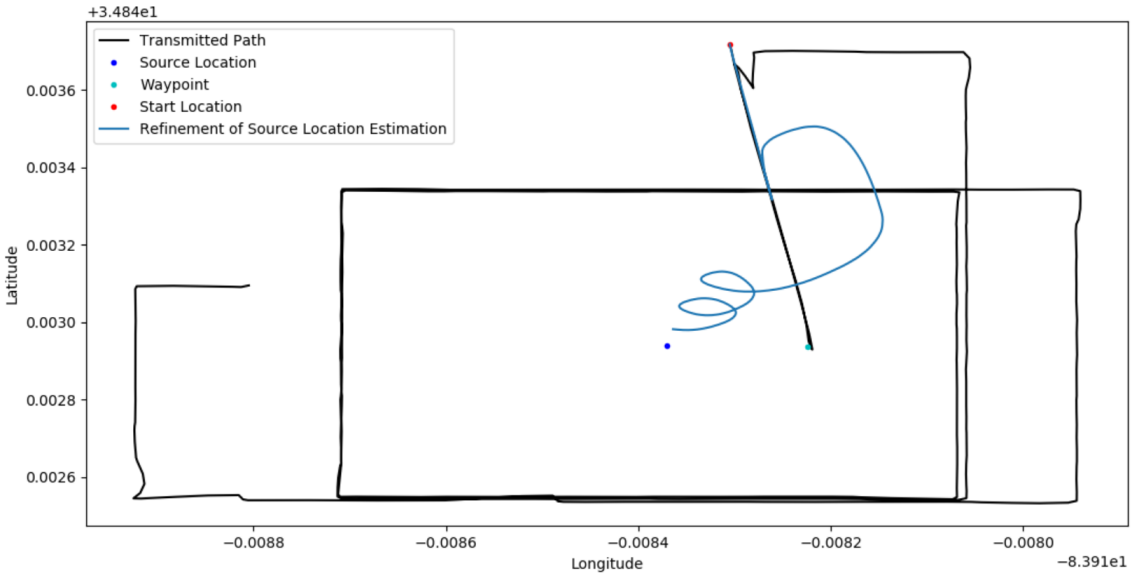

Figure A.96. Results from Simulation using Detection Array Length of 20 at an Altitude of 25m

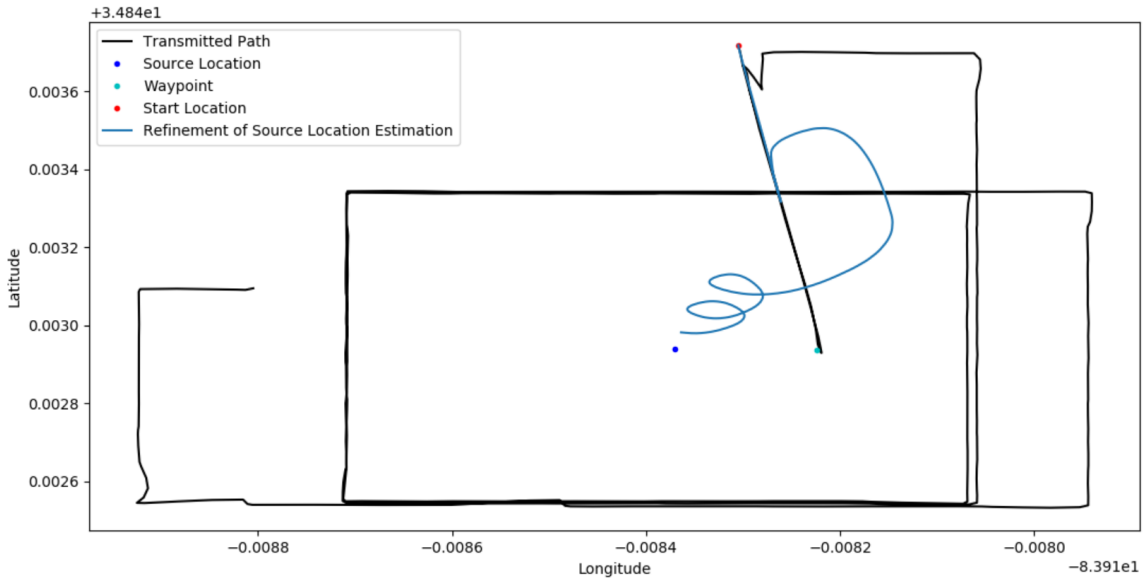

Figure A.97. Results from Simulation using Detection Array Length of 20 at an Altitude of 30m

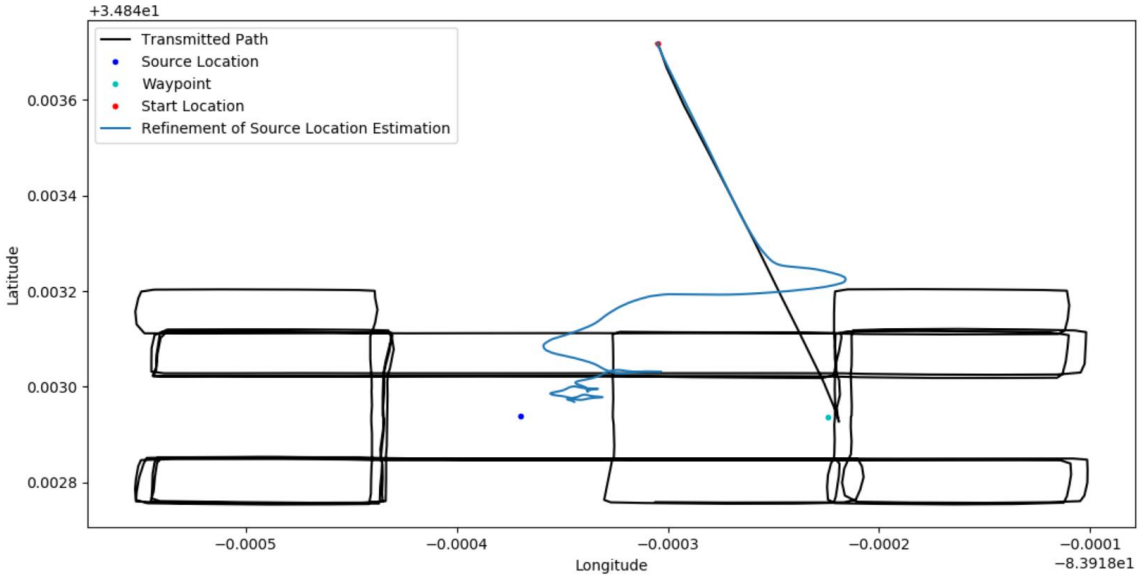

Figure A.98. Results from Simulation using Detection Array Length of 5 at an Altitude of 15m

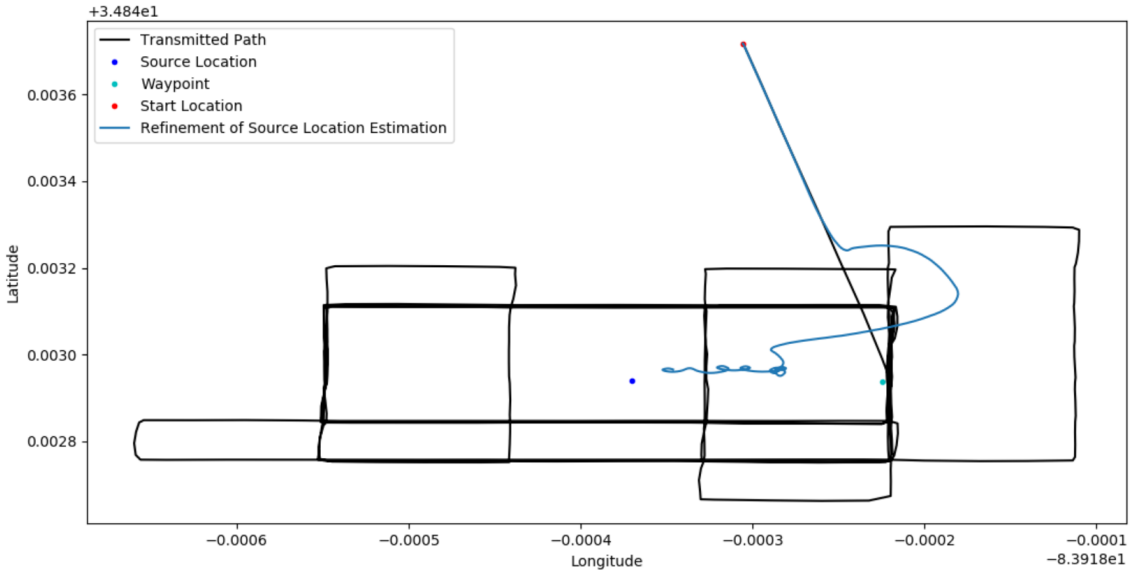

Figure A.99. Results from Simulation using Detection Array Length of 10 at an Altitude of 15m

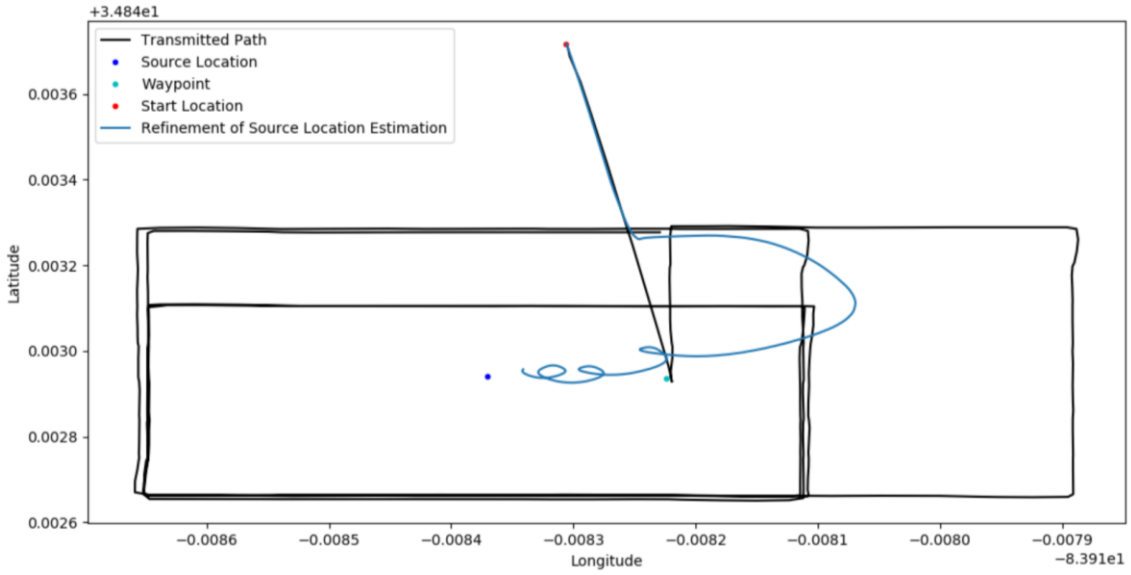

Figure A.100. Results from Simulation using Detection Array Length of 15 at an Altitude of 15m

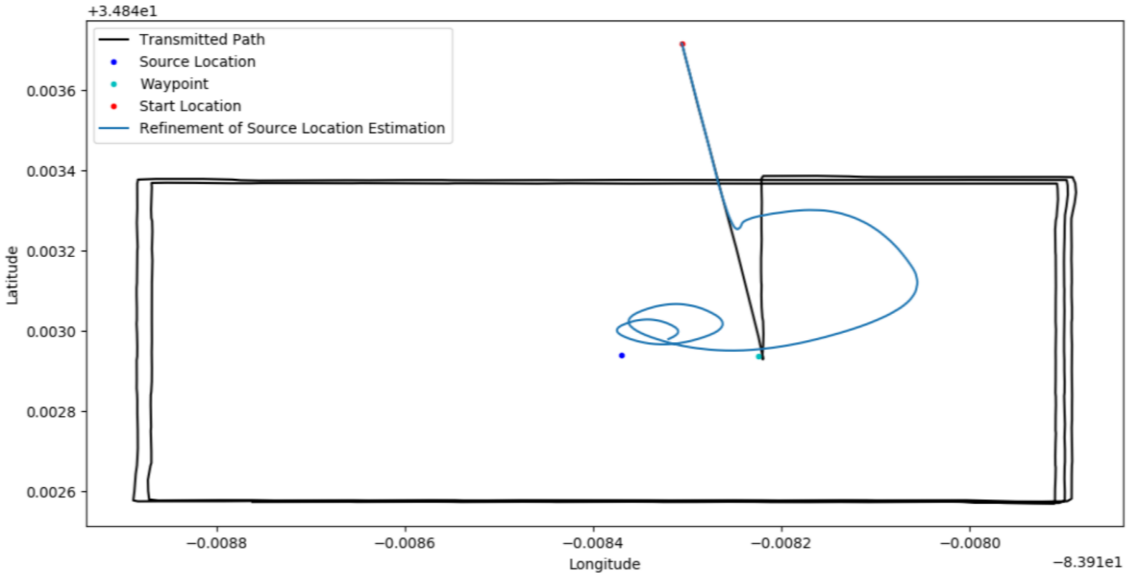

Figure A.101. Results from Simulation using Detection Array Length of 20 at an Altitude of 15m

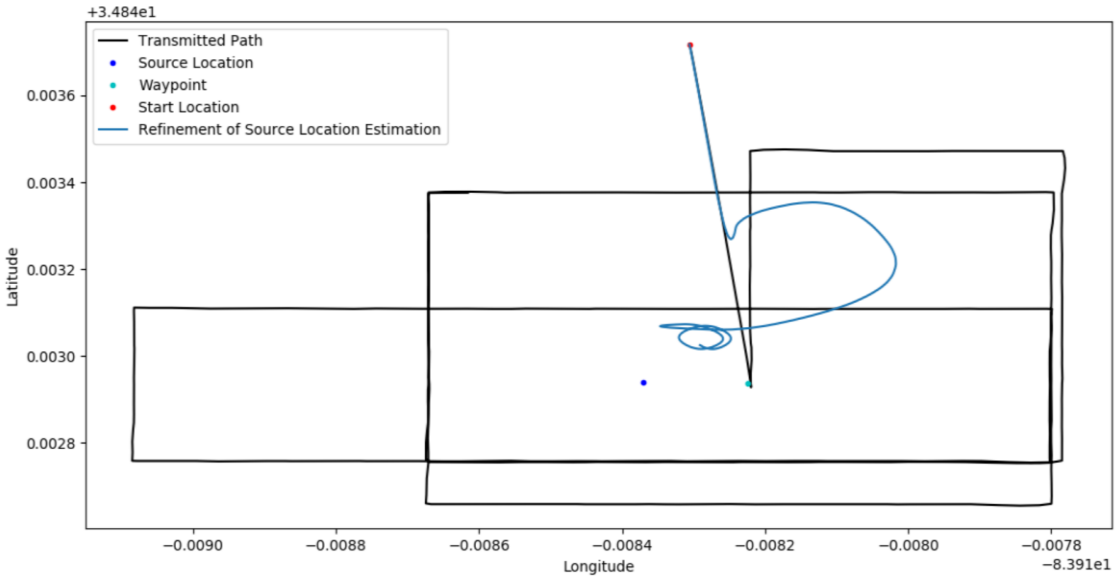

Figure A.102. Results from Simulation using Detection Array Length of 25 at an Altitude of 15m

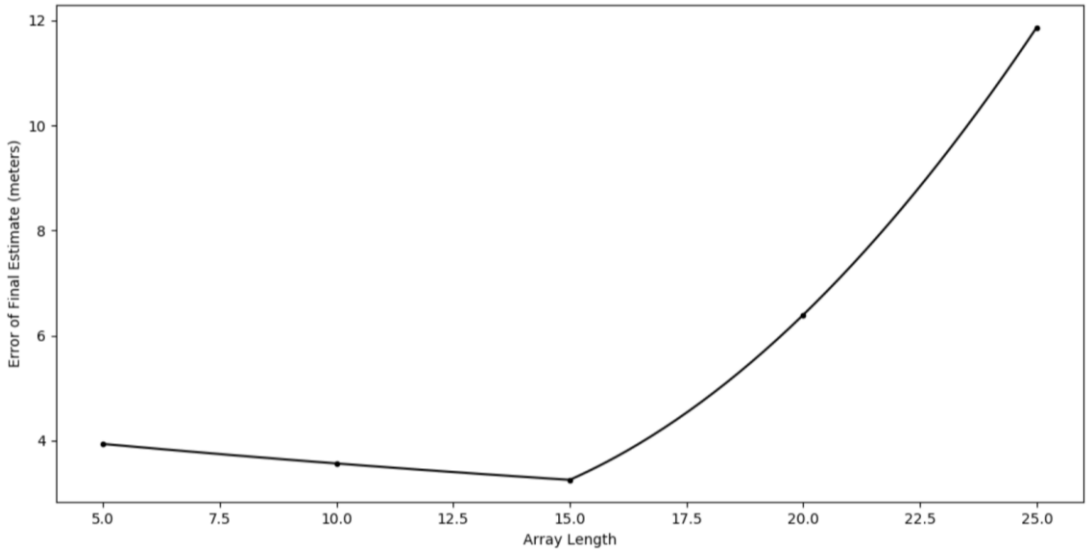

Figure A.103. Difference in Final Tier 3 Estimated Source Location and Actual Source Location for Array Length Tests

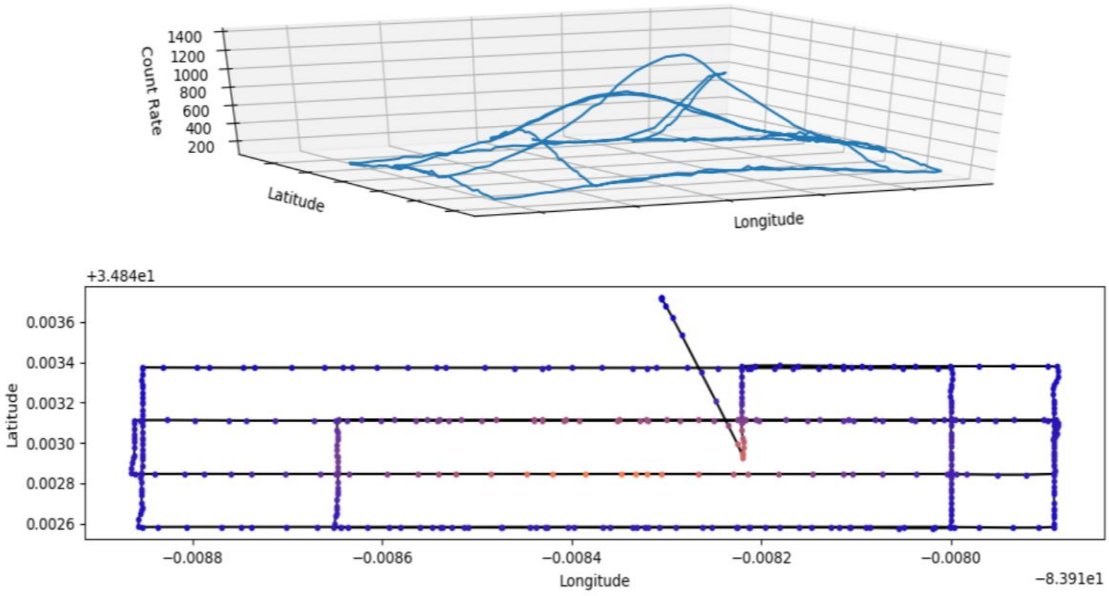

Figure A.104. Count Rate Change Displays for a Tier 3 Flight Path

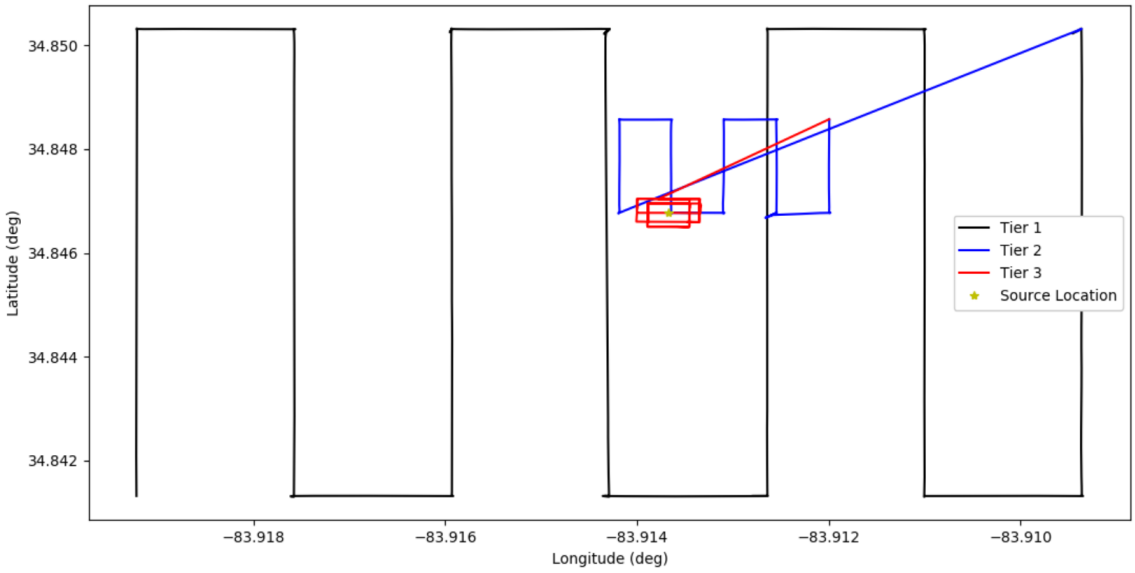

Figure A.105. Single Aircraft Triple-Tier Search with Source Located Far From Tier 1 Flight Path

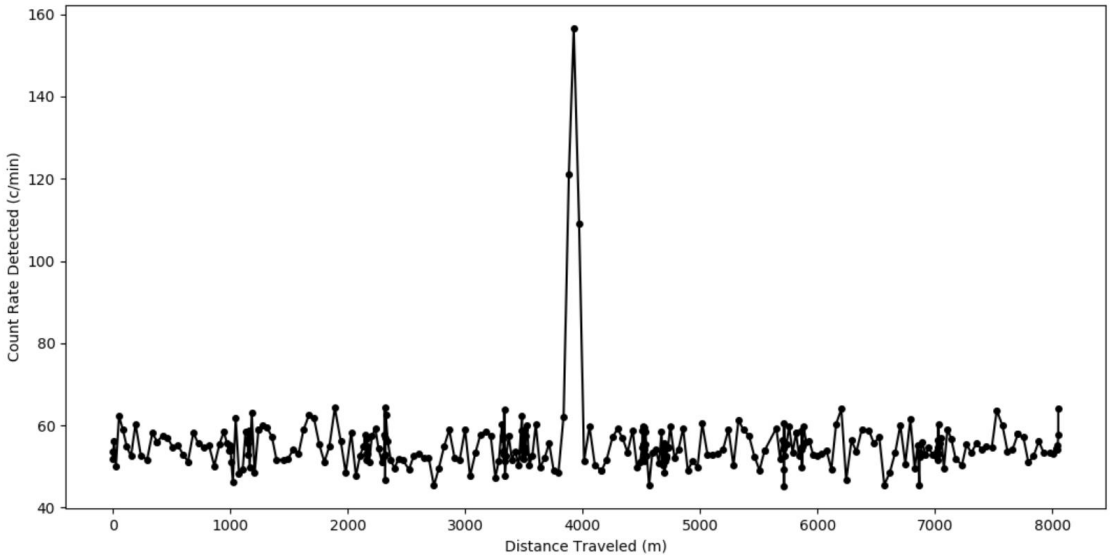

Figure A.106. Raw Count Rates Recorded During Tier 1 Single Aircraft Triple-Tier Simulation

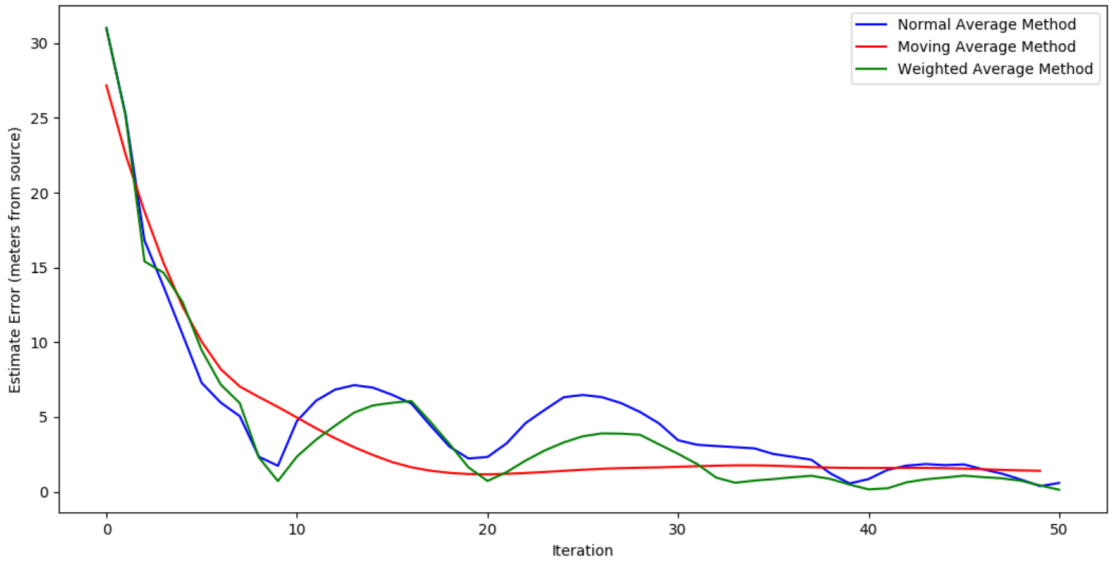

Figure A.107. Tier 3 Estimate Error Progression for the Normal Average, Moving Average, and Weighted Average Methods According to the RAT Search Method System

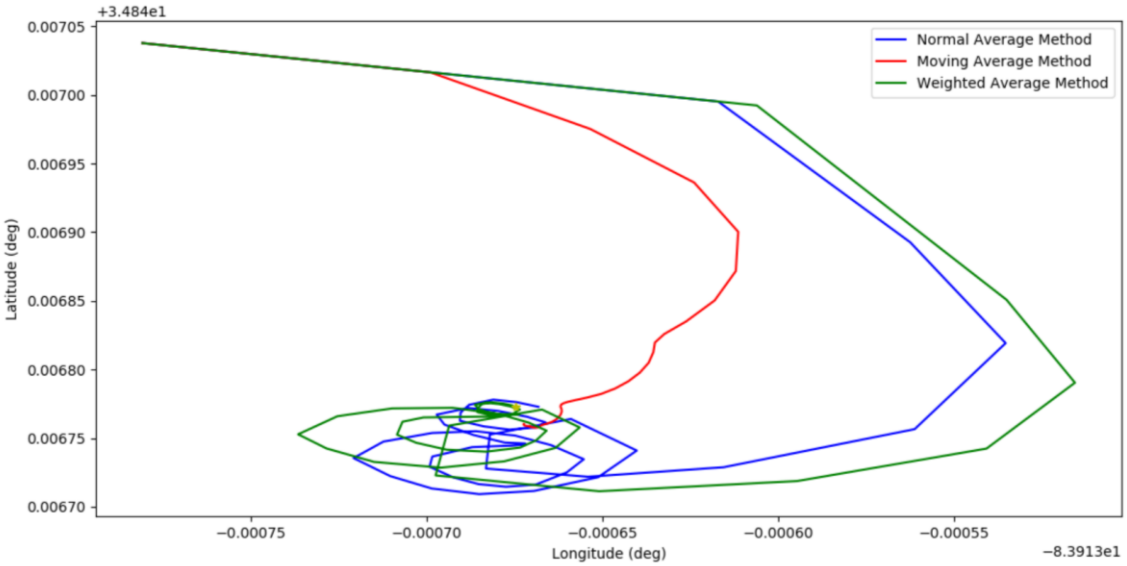

Figure A.108. Refinement Location Progression of Tier 3 Search for the Normal Average, Moving Average, and Weighted Average Methods as Subsets of the RAT Search Method

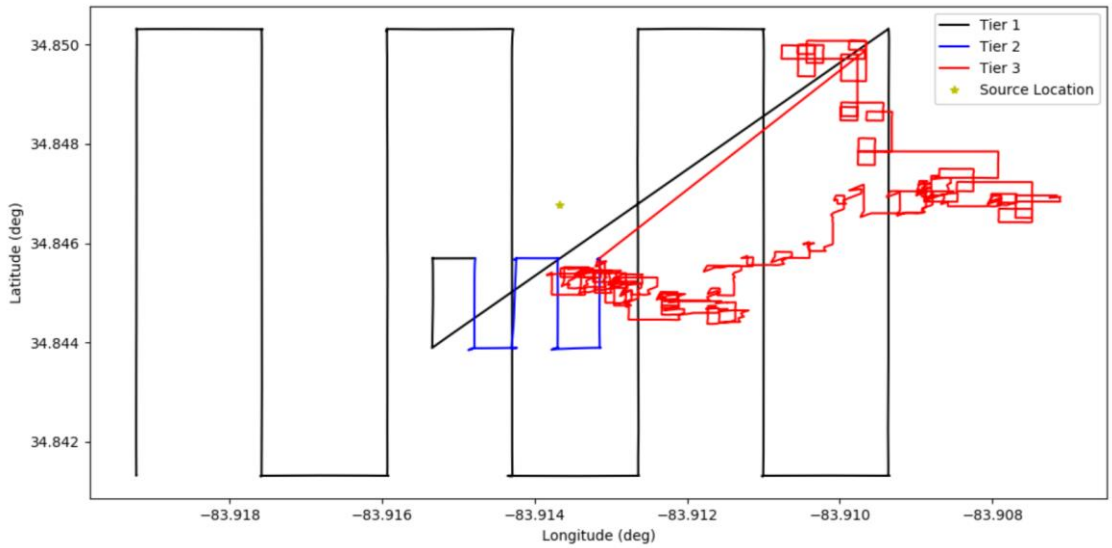

Figure A.109. Single Aircraft Search Failure Example

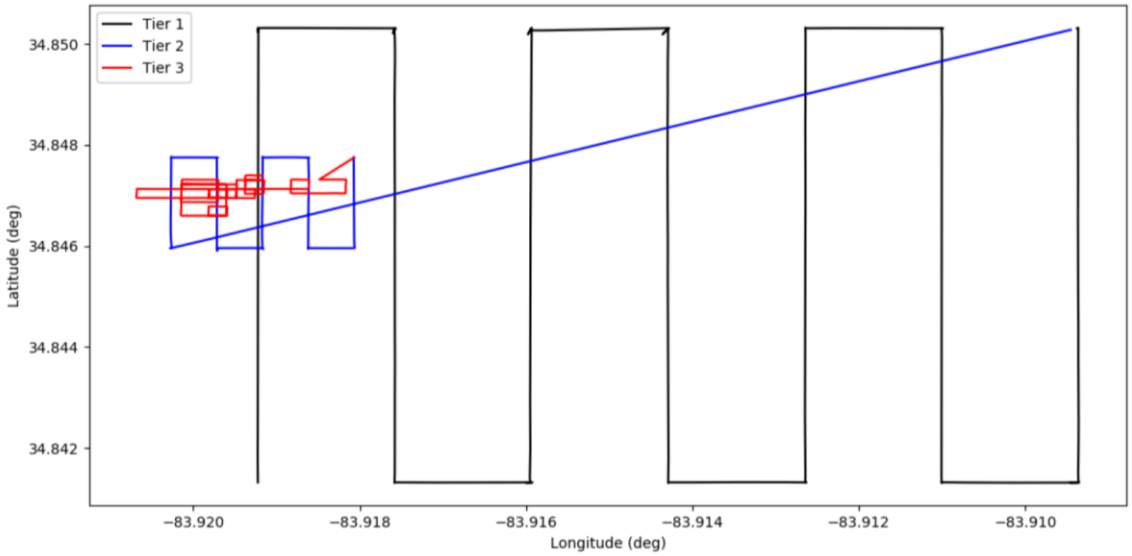

Figure A.110. Single Aircraft False Positive Triple-Tier PADUA Test

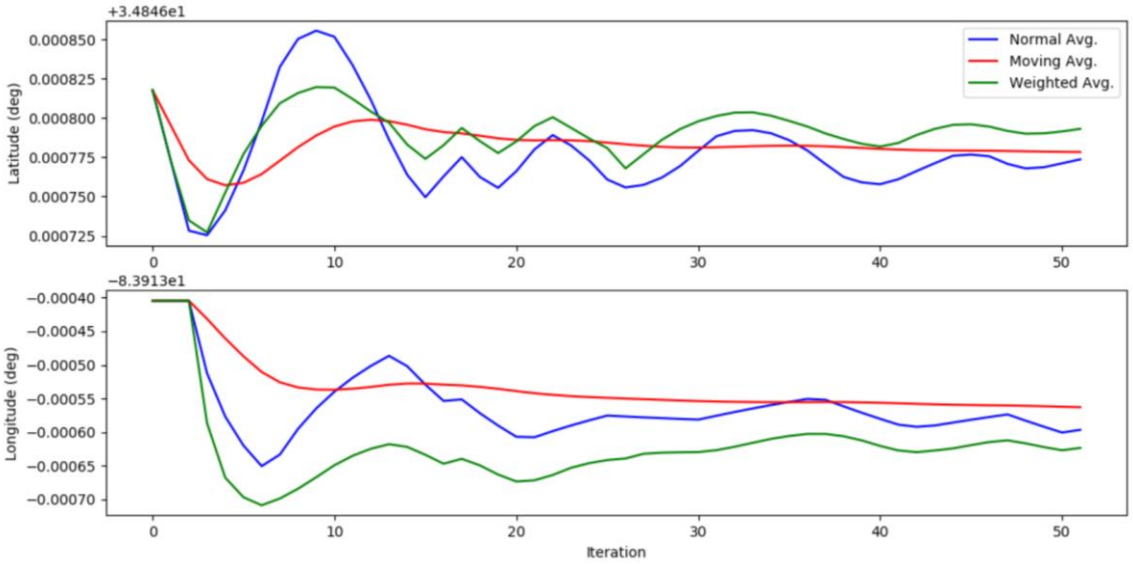

Figure A.111. Successful Tier 3 Latitude and Longitude Refinement Example

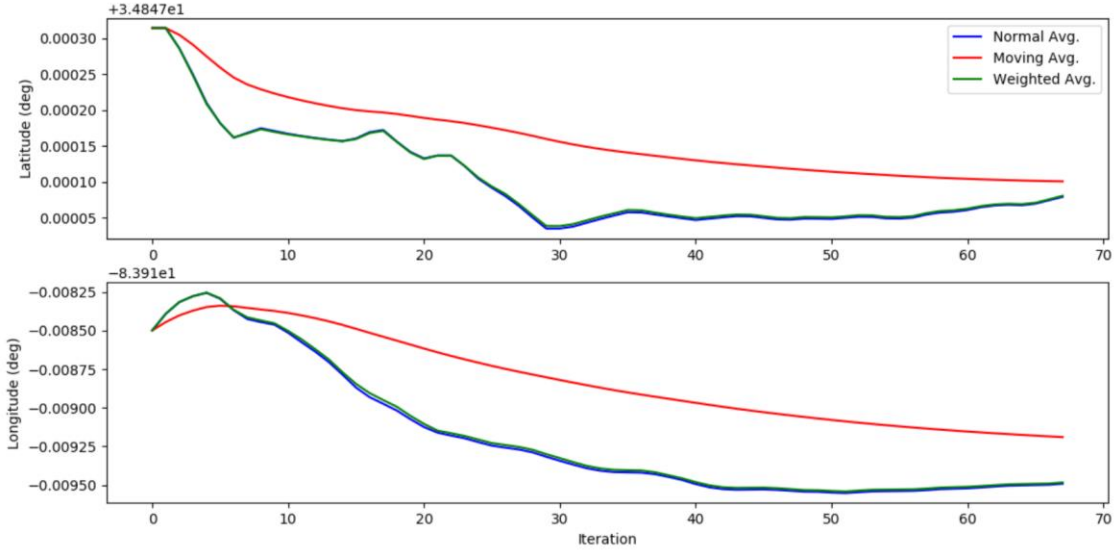

Figure A.112. Latitude and Longitude of Source Location Estimate for Tier 3 of a False Positive Search

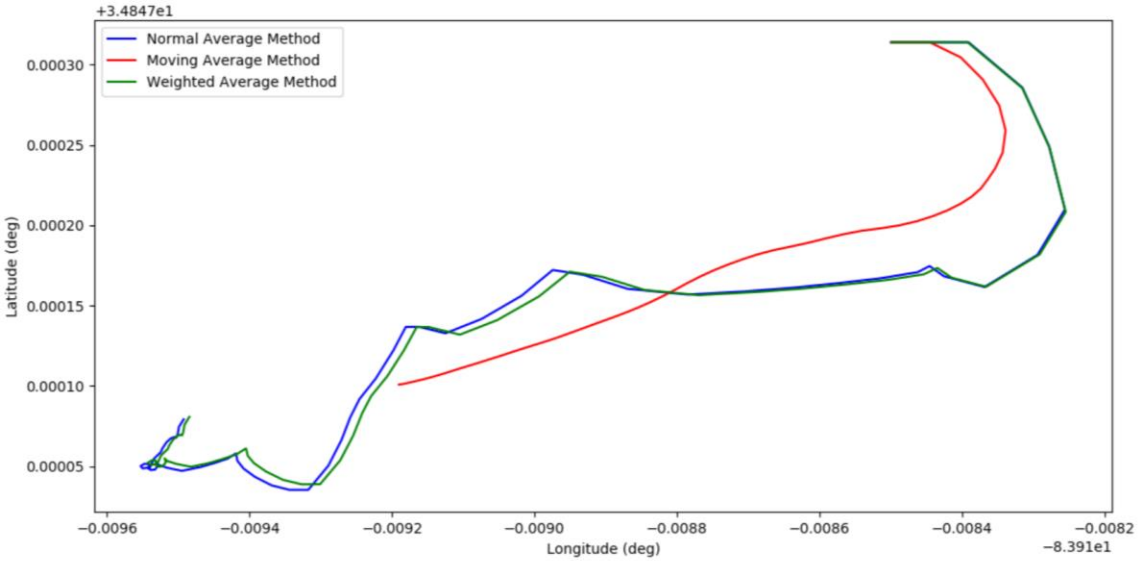

Figure A.113. False Positive Source Refinement Location

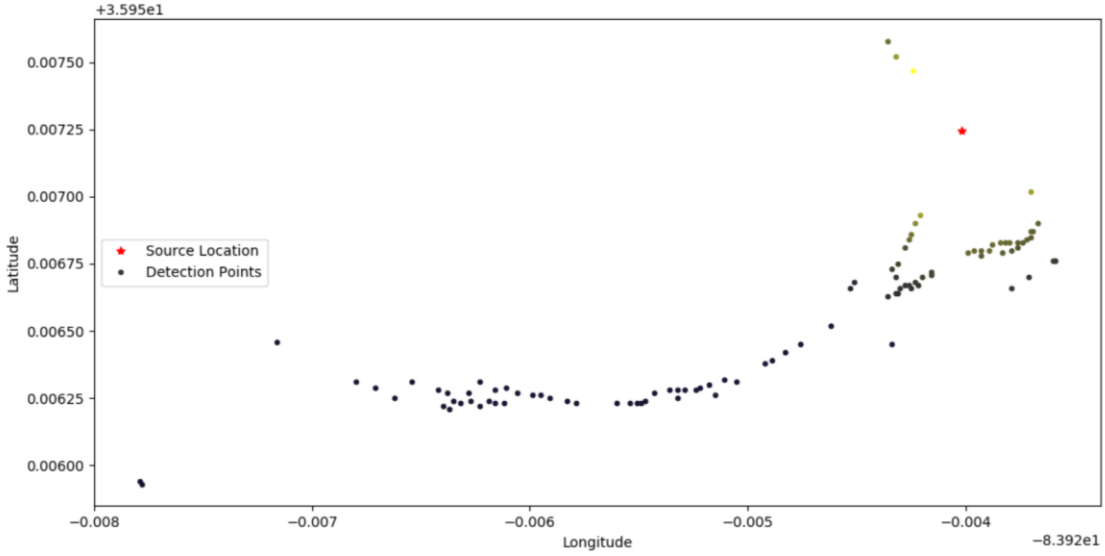

Figure A.114. Tier 1 Walking University of Tennessee Raw count Test Results

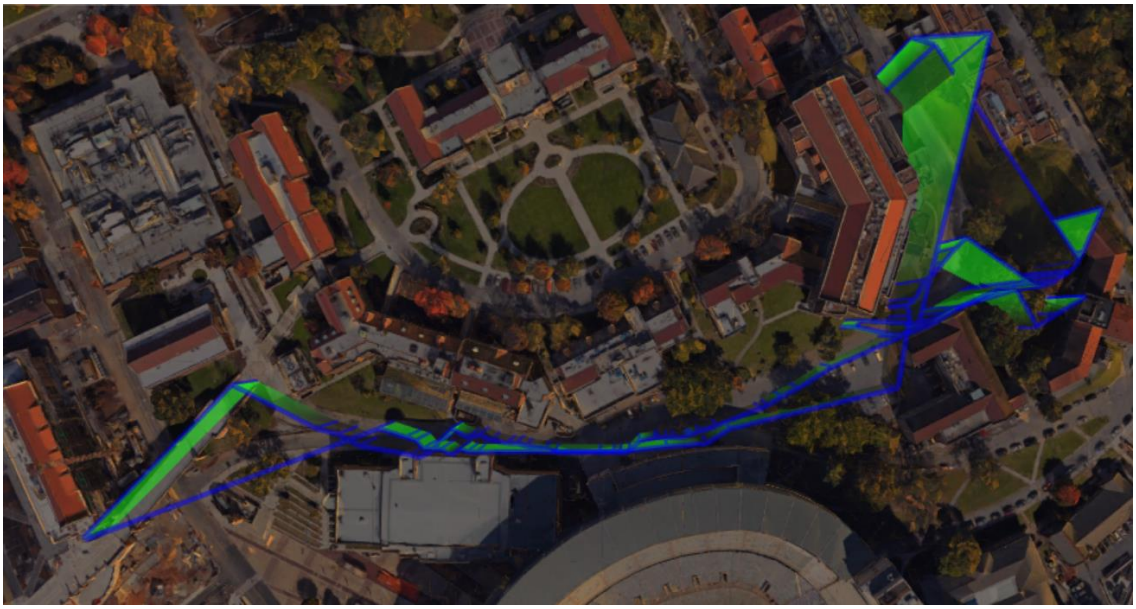

Figure A.115. Tier 1 University of Tennessee Walking Test Mapped Results (Overhead)

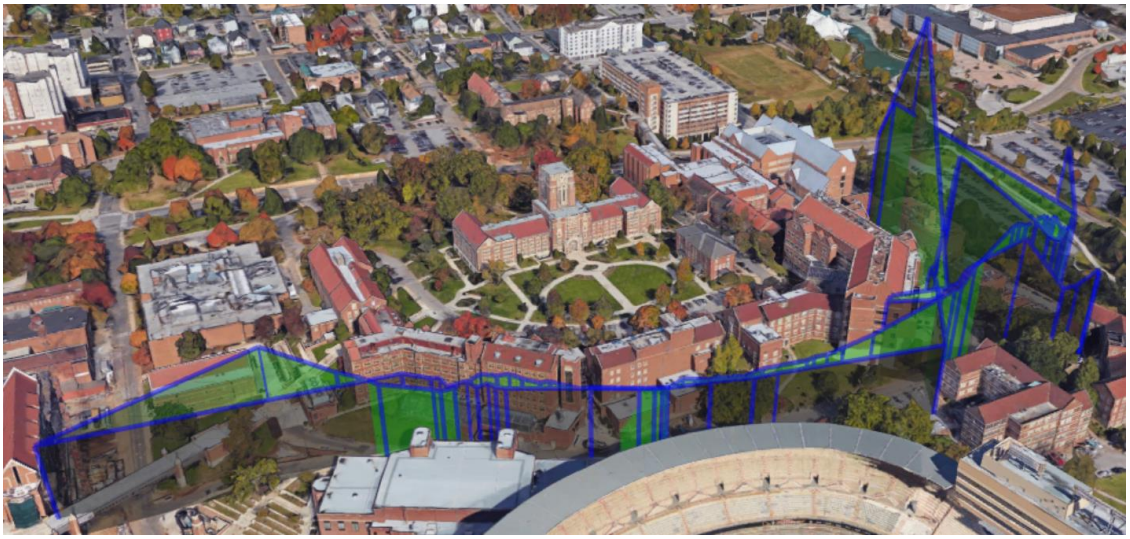

Figure A.116. Tier 1 University of Tennessee Walking Test Mapped Results (Side View 1)

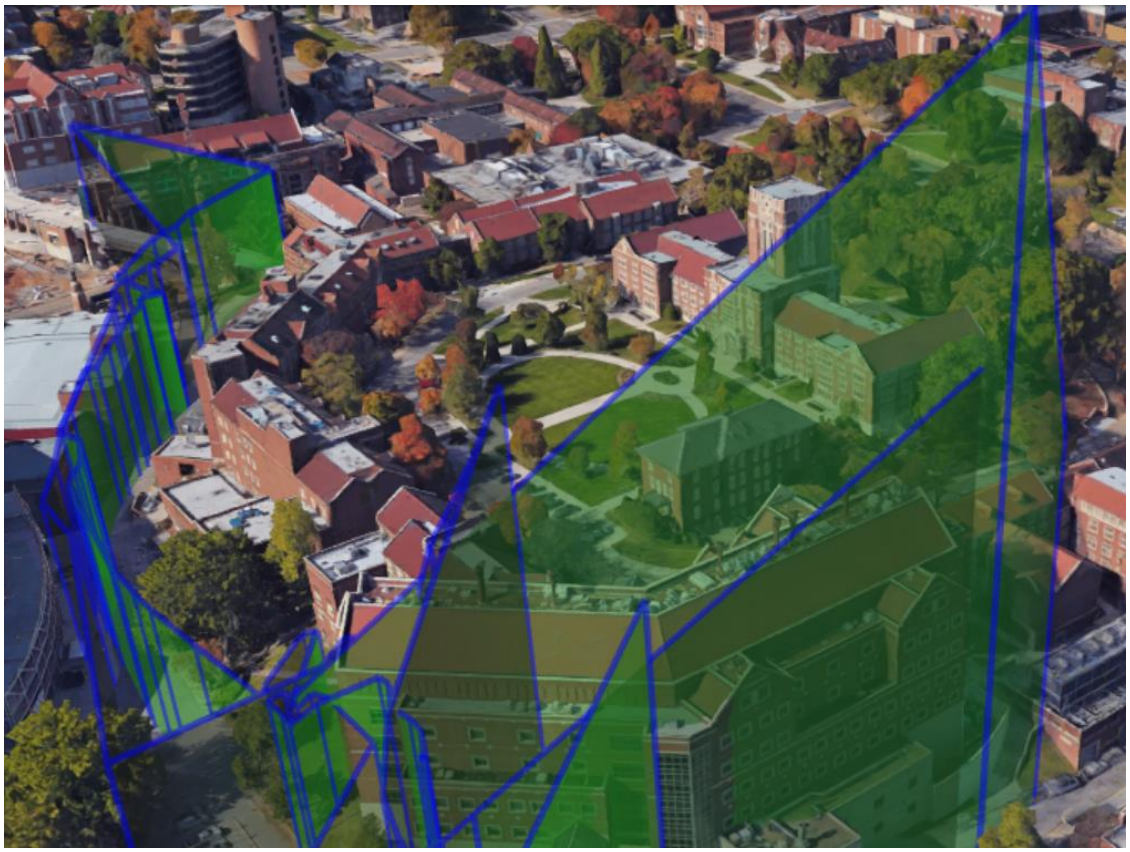

Figure A.117. Tier 1 University of Tennessee Walking Test Mapped Results (Side View 2)

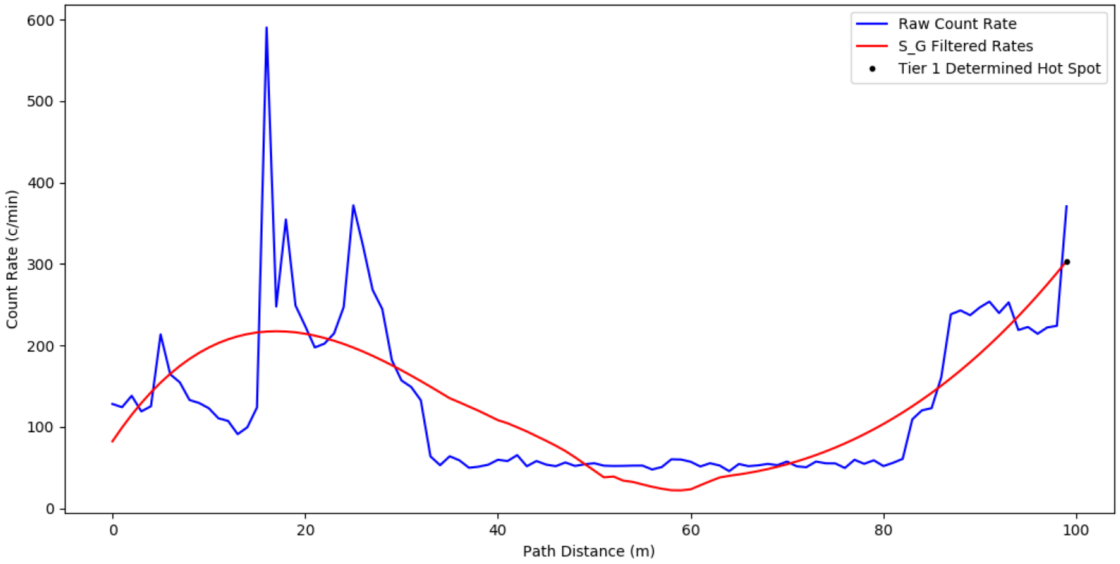

Figure A.118. S_G Filtered Hot Spot Generation for Walking Test Tier 1 Results

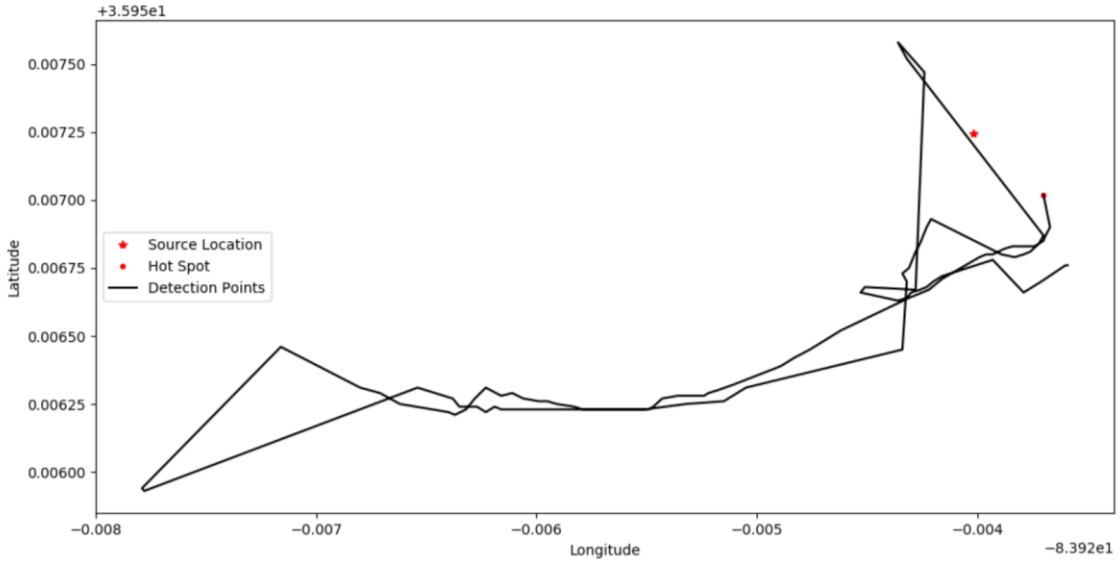

Figure A.119. University of Tennessee Walking Test Tier 1 Hot Spot Location

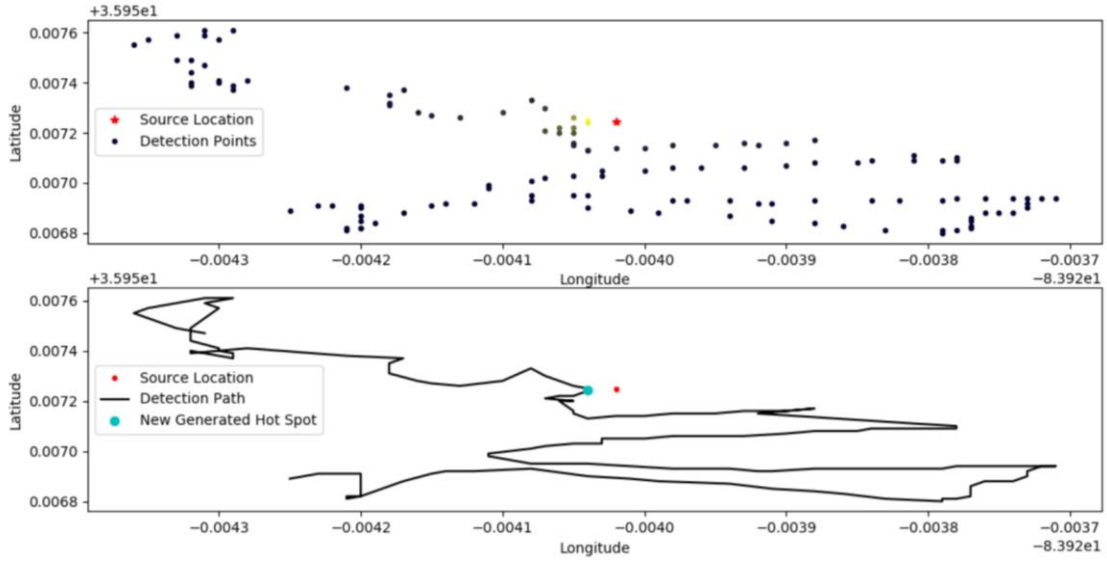

Figure A.120. University of Tennessee Walking Test Tier 2 Hot Spot Location

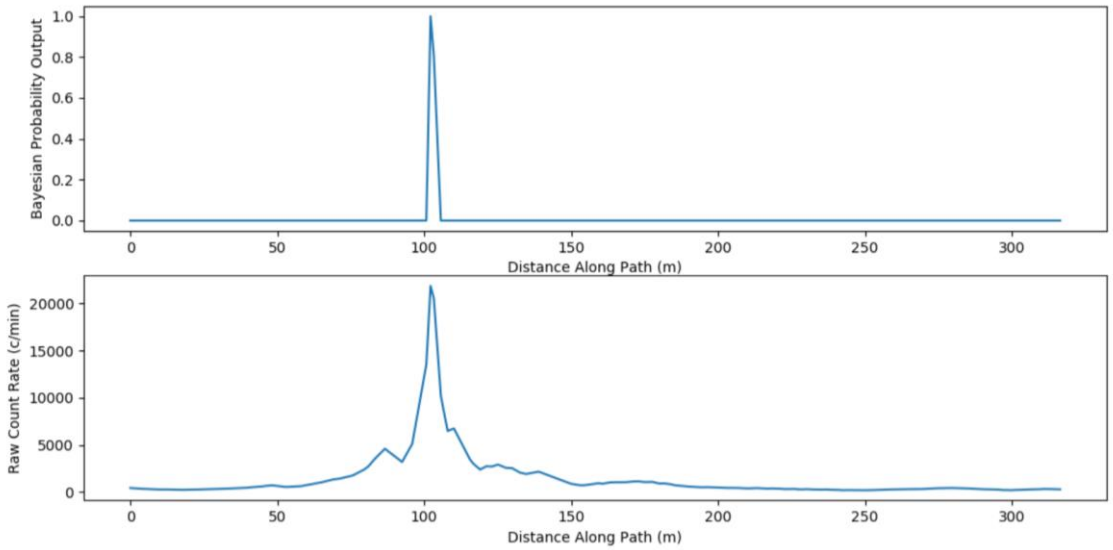

Figure A.121. University of Tennessee Walking Test Tier 2 Bayesian Refinement Results vs. Raw Count Rate Results

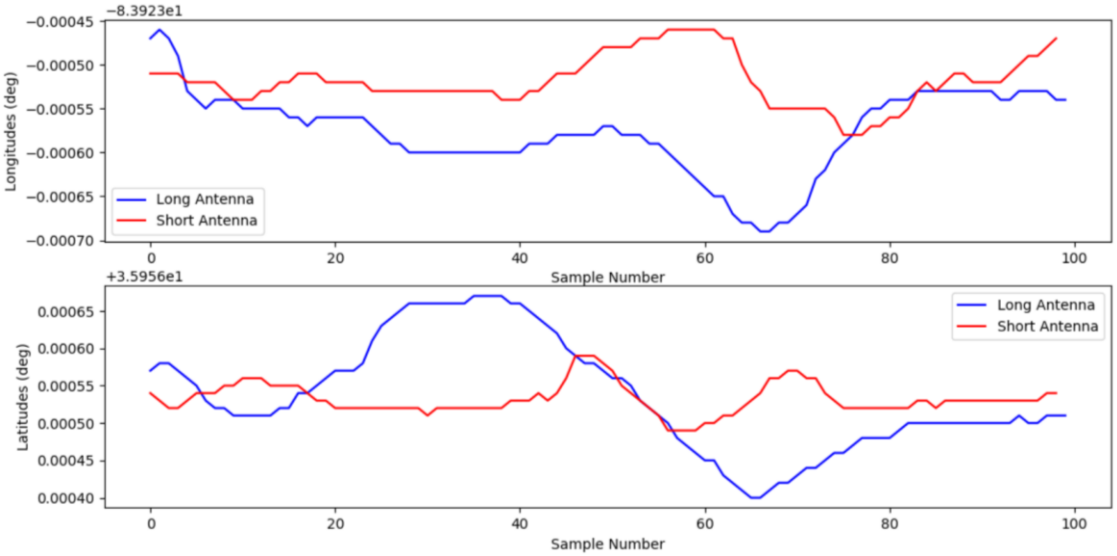

Figure A.122. Inside (Obstructed) Area Antenna Calibration Test Results

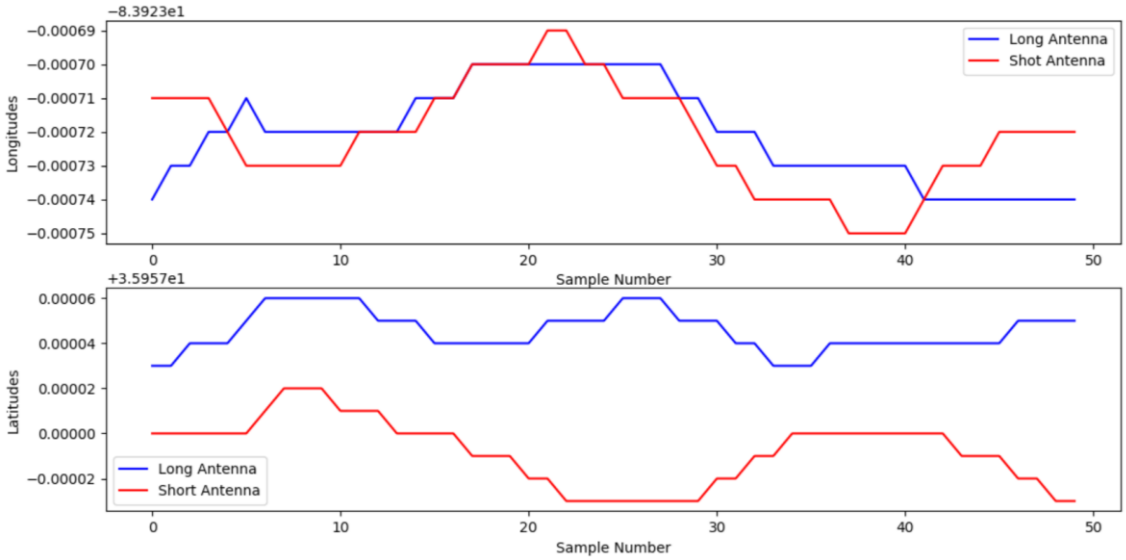

Figure A.123. Open Area Antenna Calibration Test Results

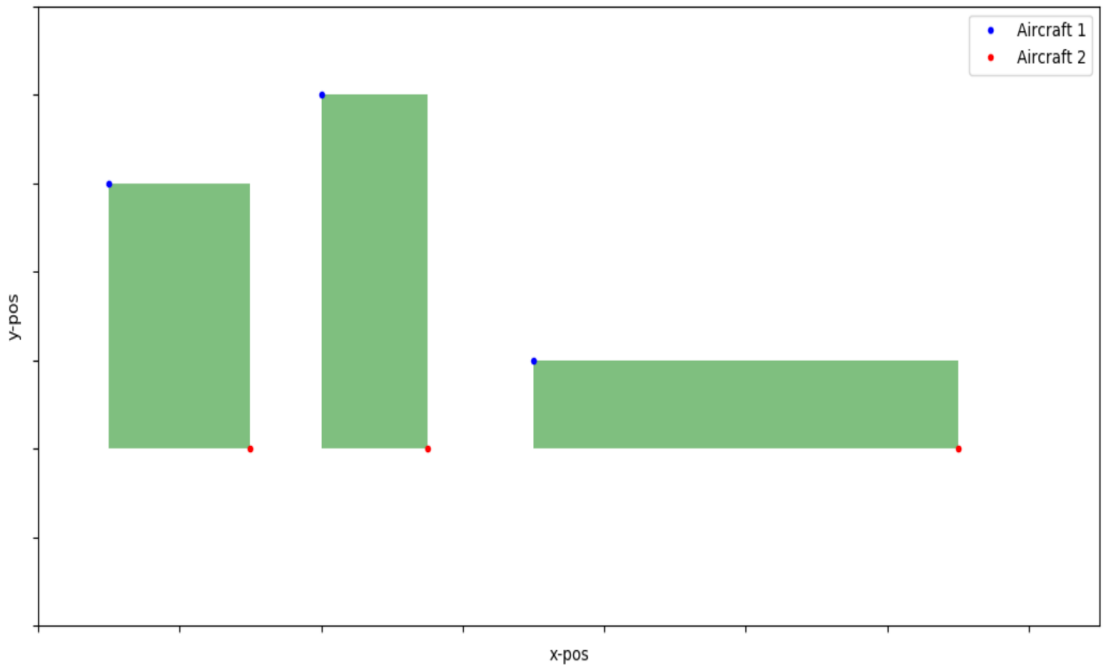

Figure A.124. Example of Constant Area Based on Two-Agent Directional (y-axis) Separation

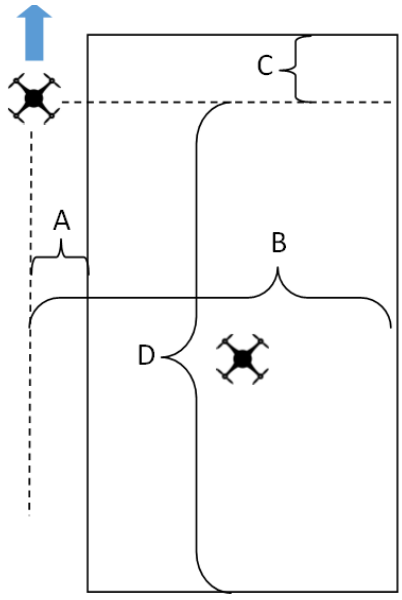

Figure A.125. Diagram of Bound References

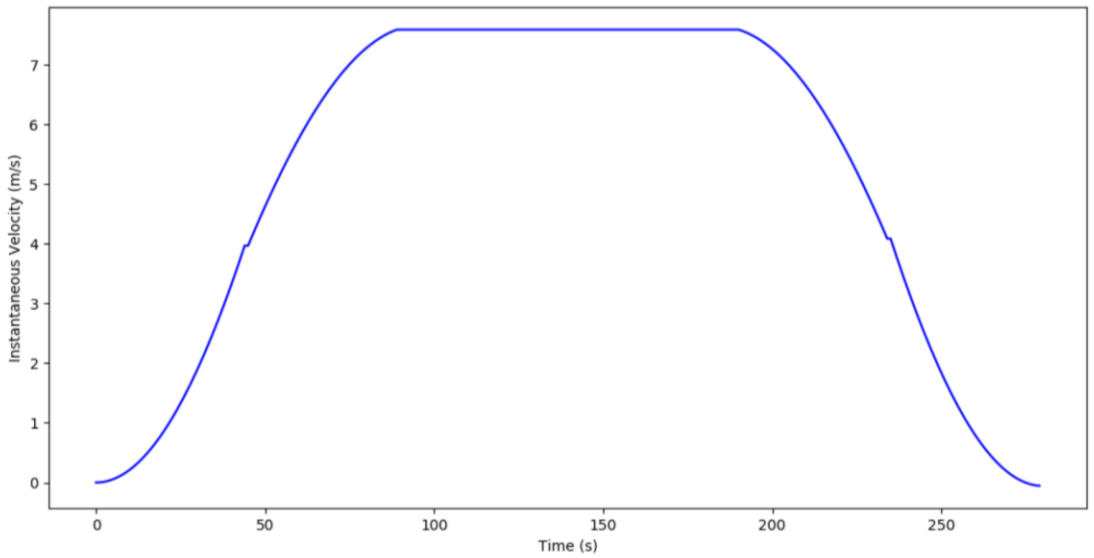

Figure A.126. Hypothetical One-Dimensional Lead Aircraft Velocity-Time Curve Used for Theoretical Verification of Bound Continuity Swarming Theory

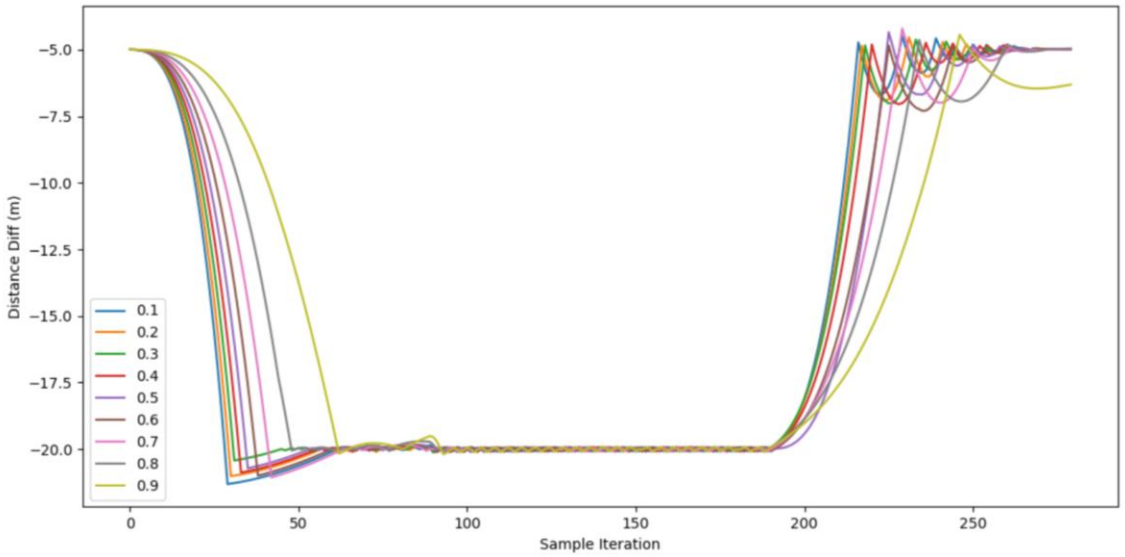

Figure A.127. Forward Velocity Difference Between Lead and Follower Aircrafts Compared Between Different Velocity Lag Rate Multipliers

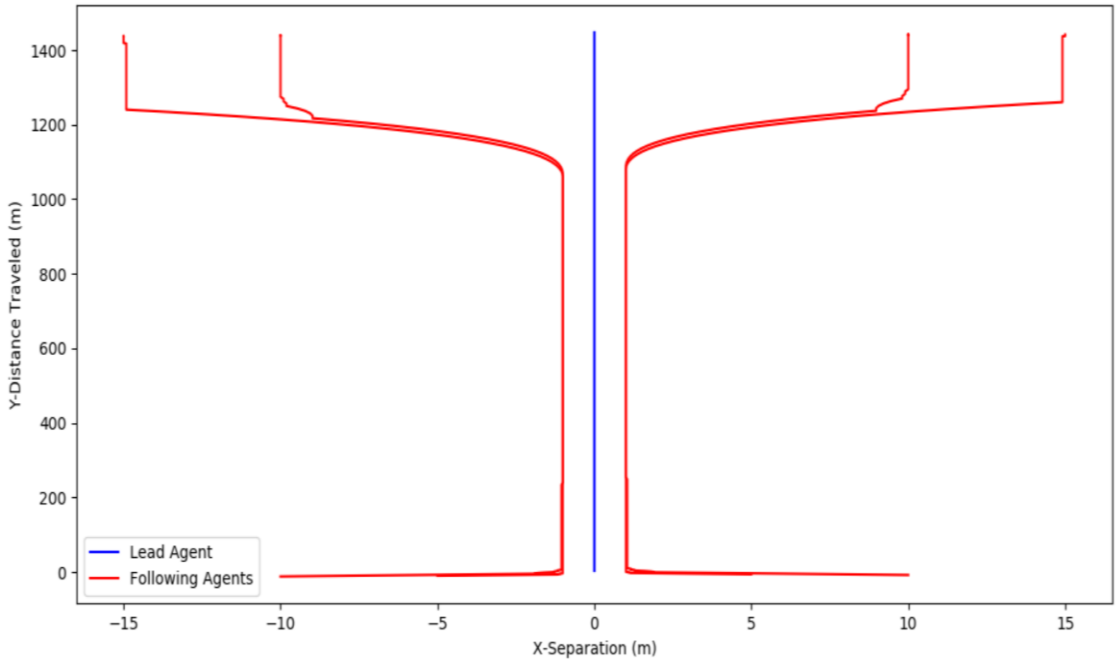

Figure A.128. Single-Direction Theoretical Results for Swarm of Four Following Aircraft and One Lead Aircraft Utilizing Bounded Continuity Algorithm

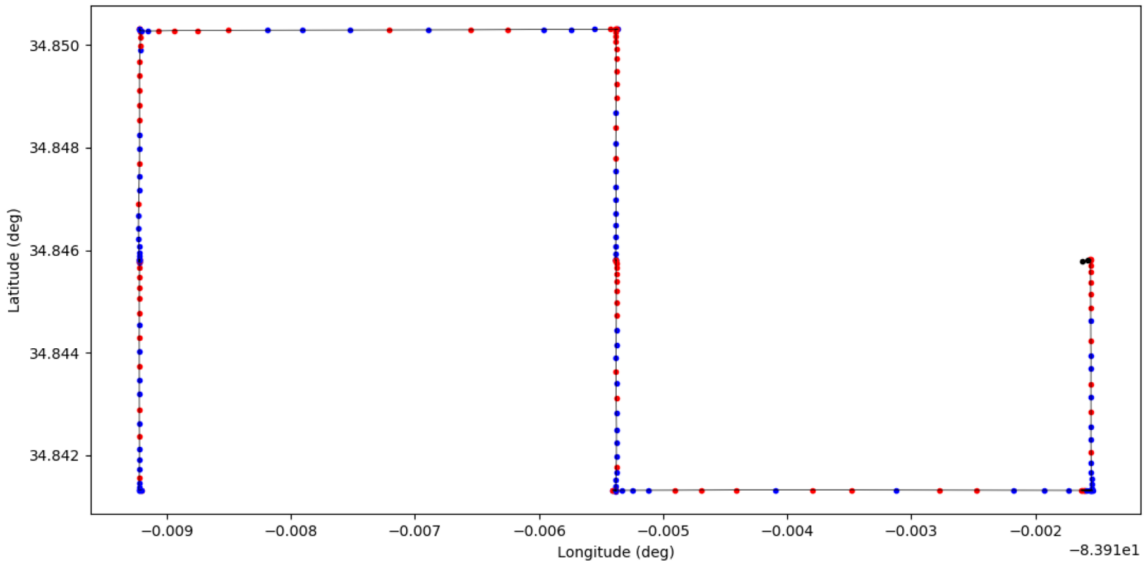

Figure A.129. Lead Aircraft Path with Velocity Decrease Points Shown in Red and Increase Points Shown in Blue

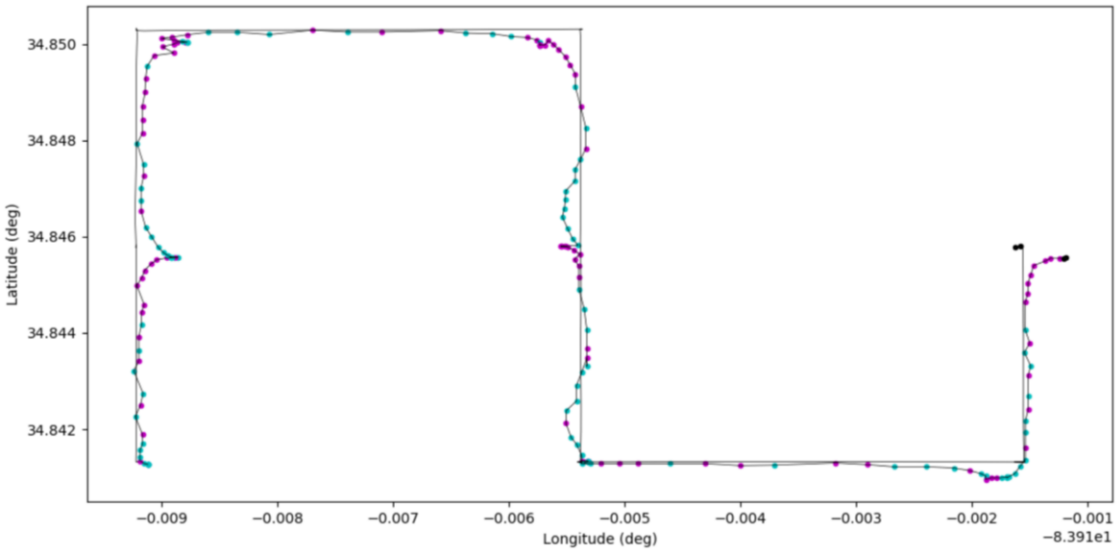

Figure A.130. Follower Aircraft Theoretical Path Generated from Realistic Lead Aircraft Data with Forward Velocity Increase Points Shown in Cyan and Decrease Points Shown in Magenta

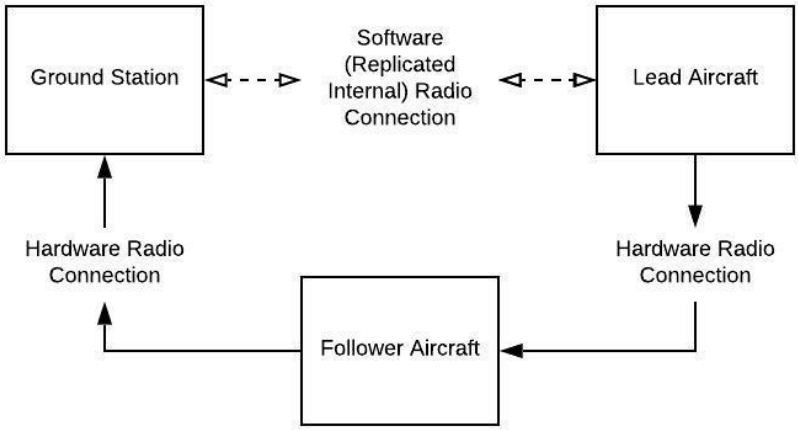

Figure A.131. Double Aircraft Simulated System Setup Framework

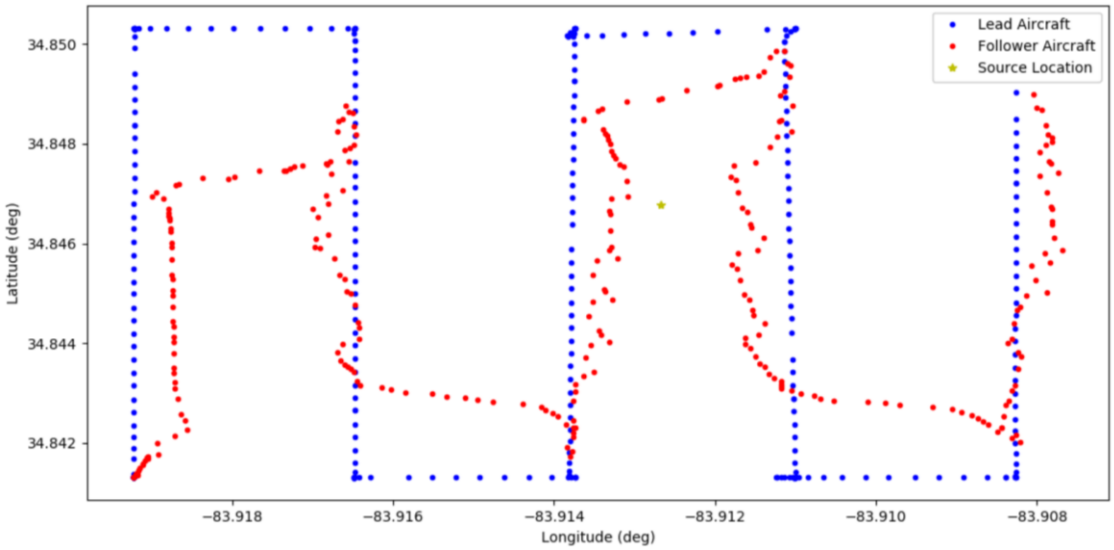

Figure A.132. 1000m x 1000m Space with x-spacing of 250m

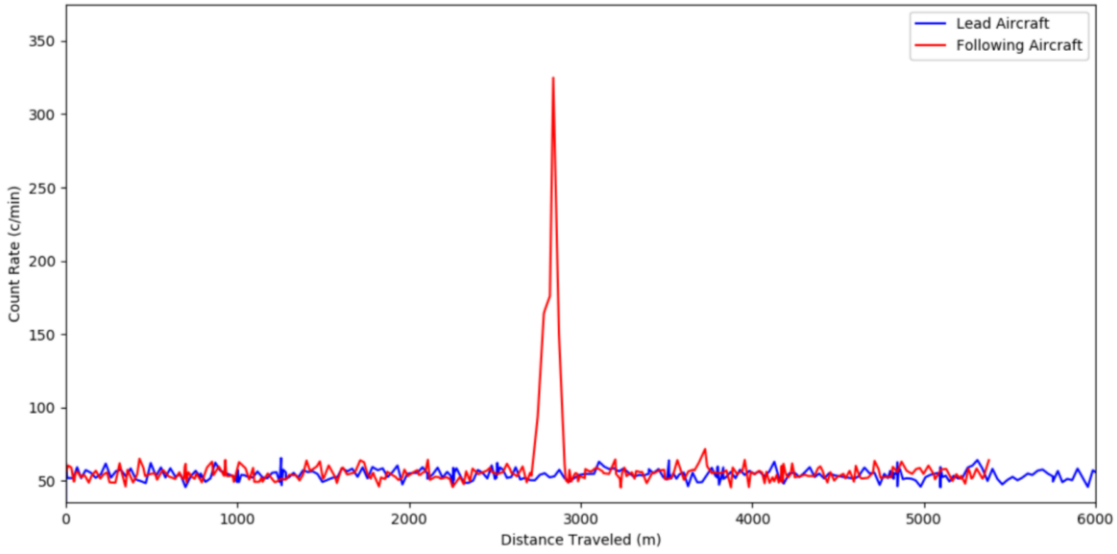

Figure A.133. Count Rates from Double Aircraft System

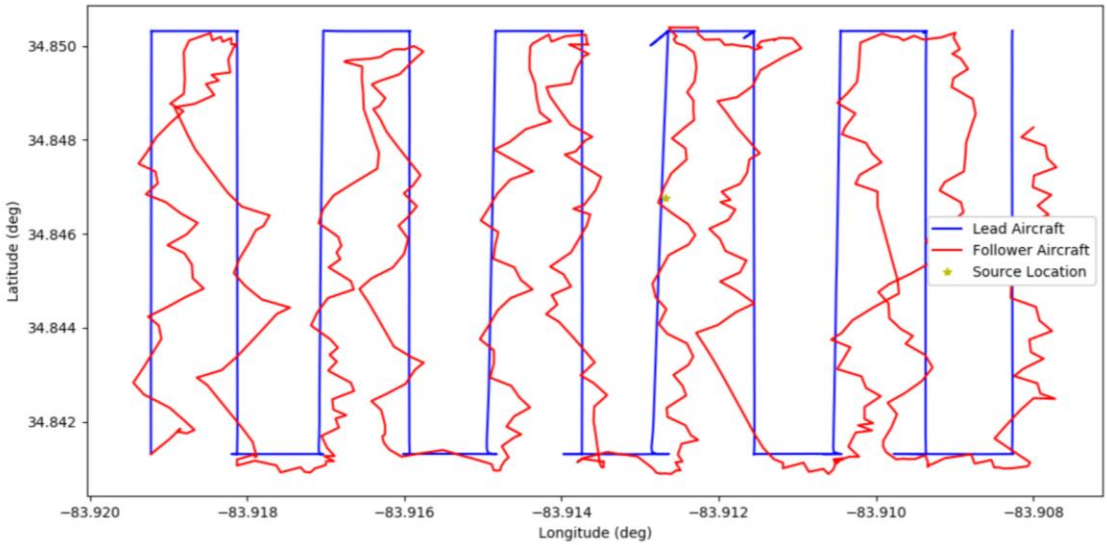

Figure A.134. Tier 1 Double Aircraft Bound-Continuity Swarming Paths

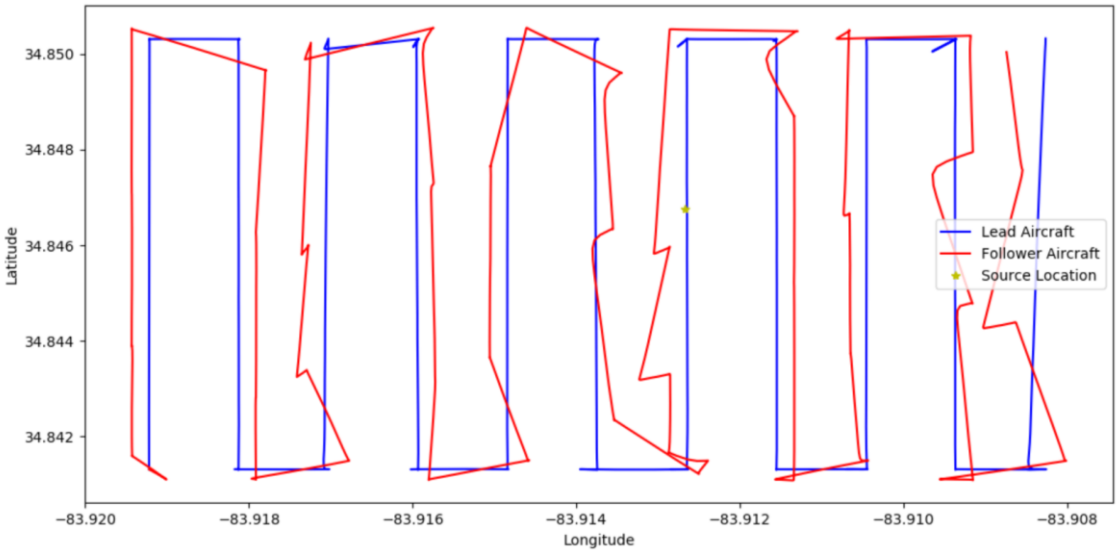

Figure A.135. Tier 1 Double Aircraft Pattern (By Target Waypoint) Swarming Paths

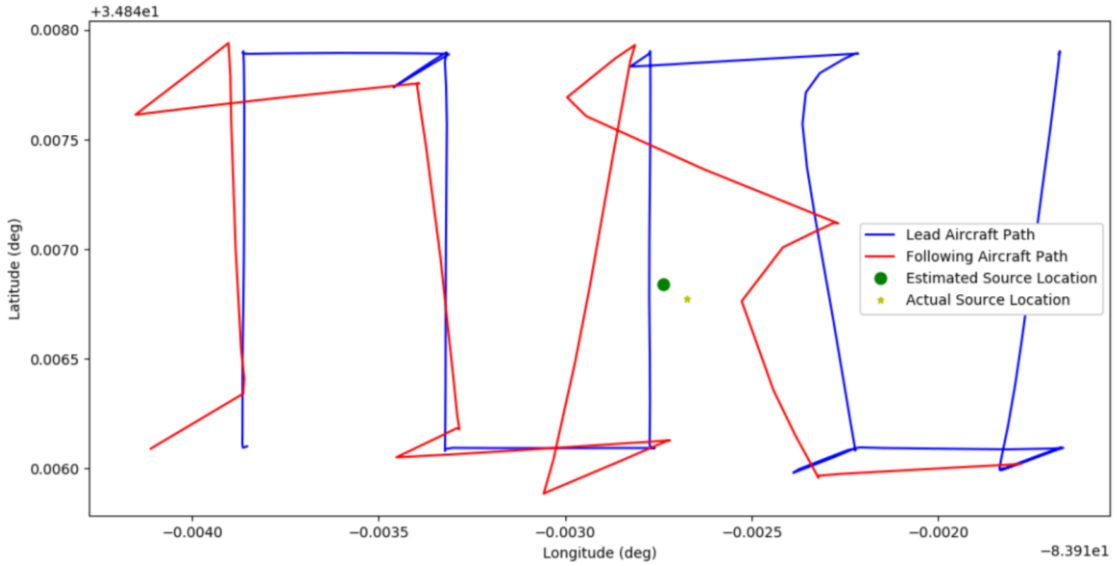

Figure A.136. Tier 2 Double Aircraft Pattern (By Target Waypoint) Swarming Paths

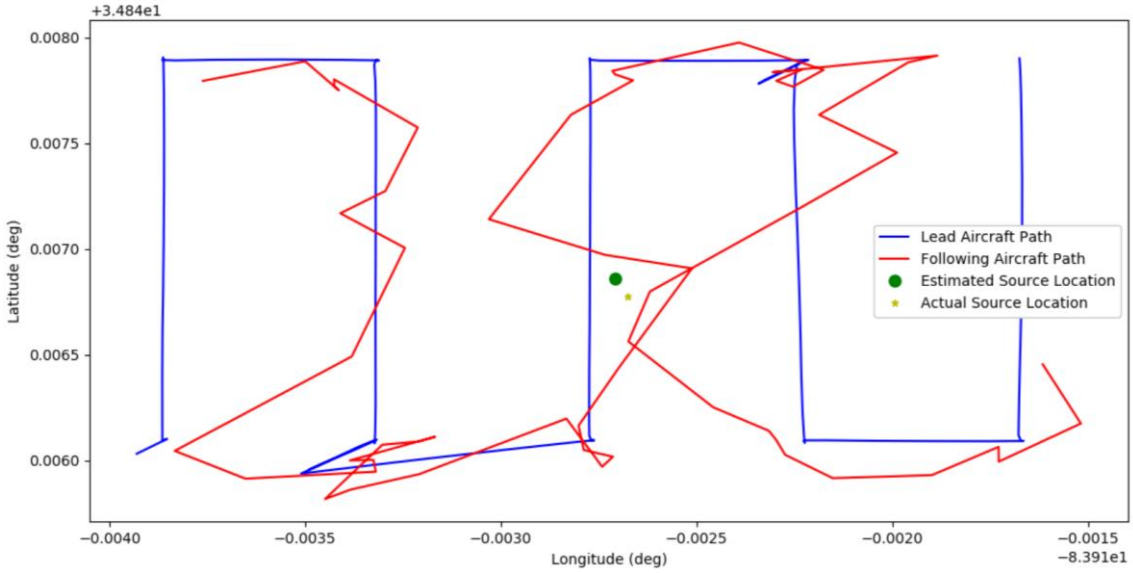

Figure A.137. Tier 2 Double Aircraft Bound-Continuity Swarming Paths

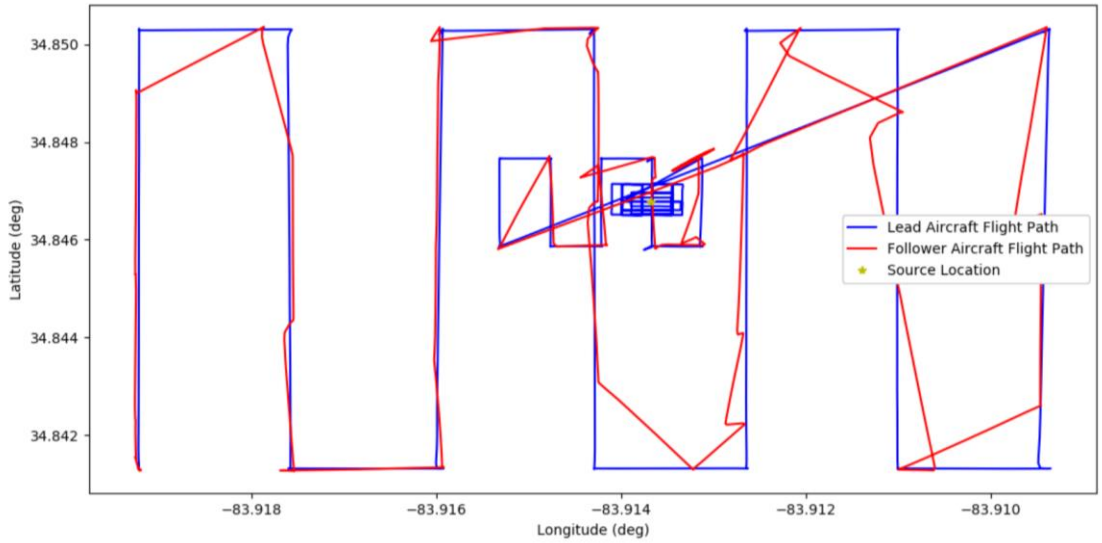

Figure A.138. Flight Paths of the Fully Simulated Triple-Tier Multi-Aircraft Pattern-Based Swarming System Using PADUA

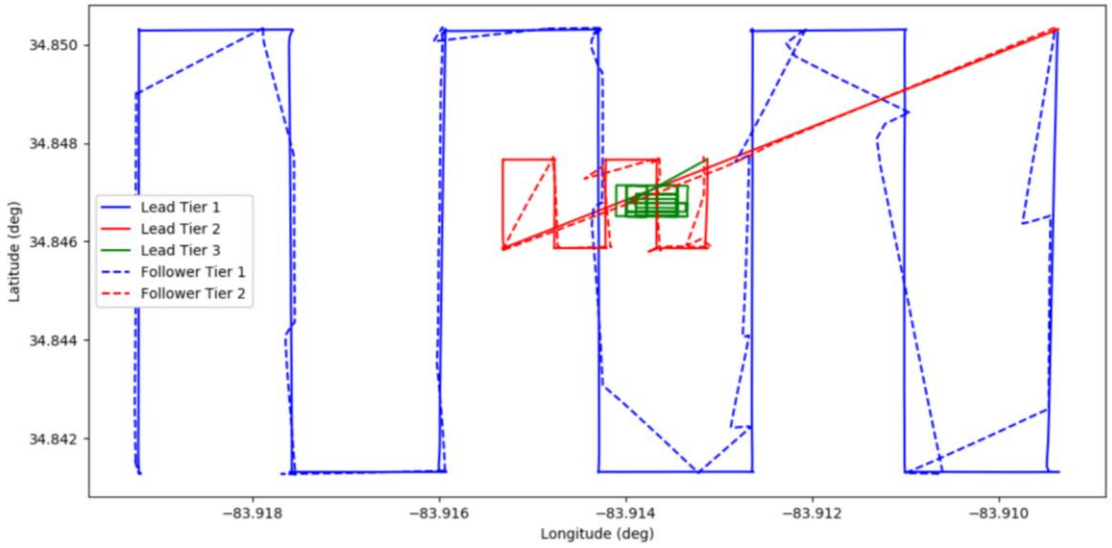

Figure A.139. Individual Tiers for Each Aircraft in Pattern-Based Multi-Aircraft Swarming Full PADUA Search

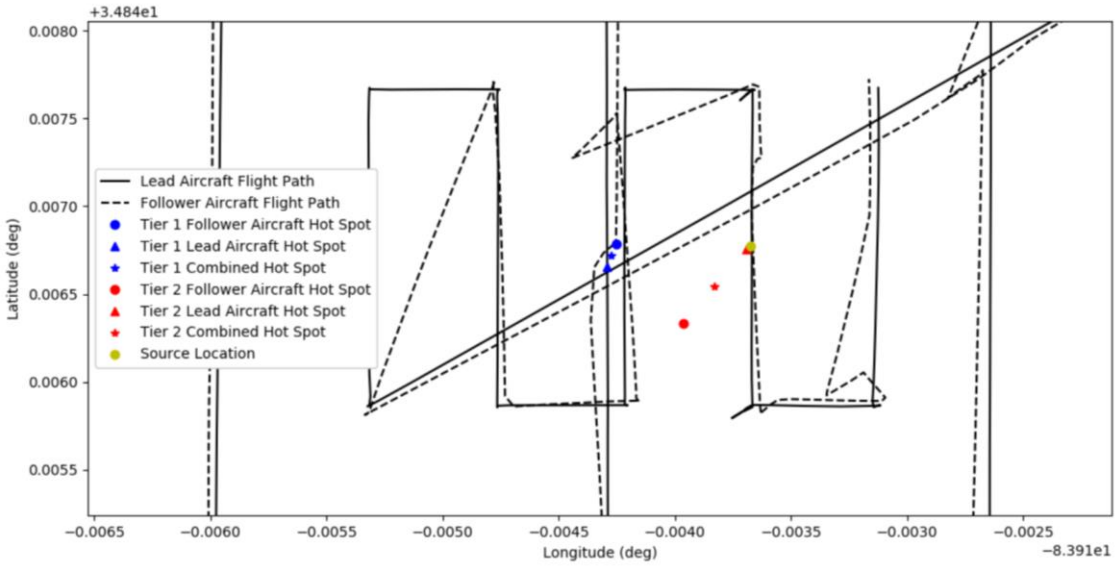

Figure A.140. Hot Spots and Source Locations for Pattern-Based Full Triple-Tier Simulation Search

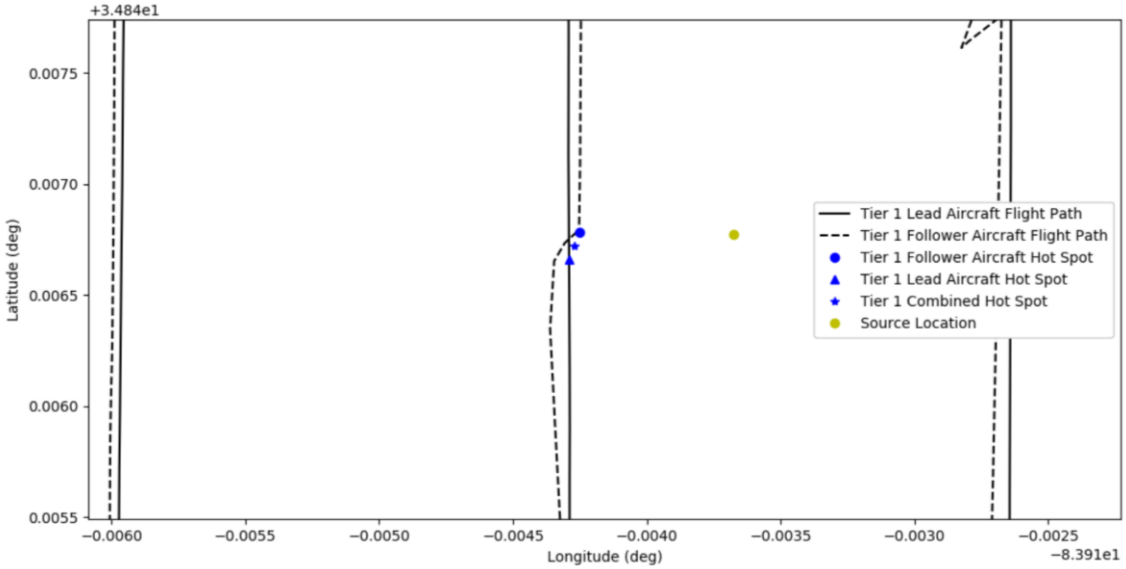

Figure A.141. Tier 1 Hot Spots and Source Location for Pattern-Based Full Triple-Tier Search

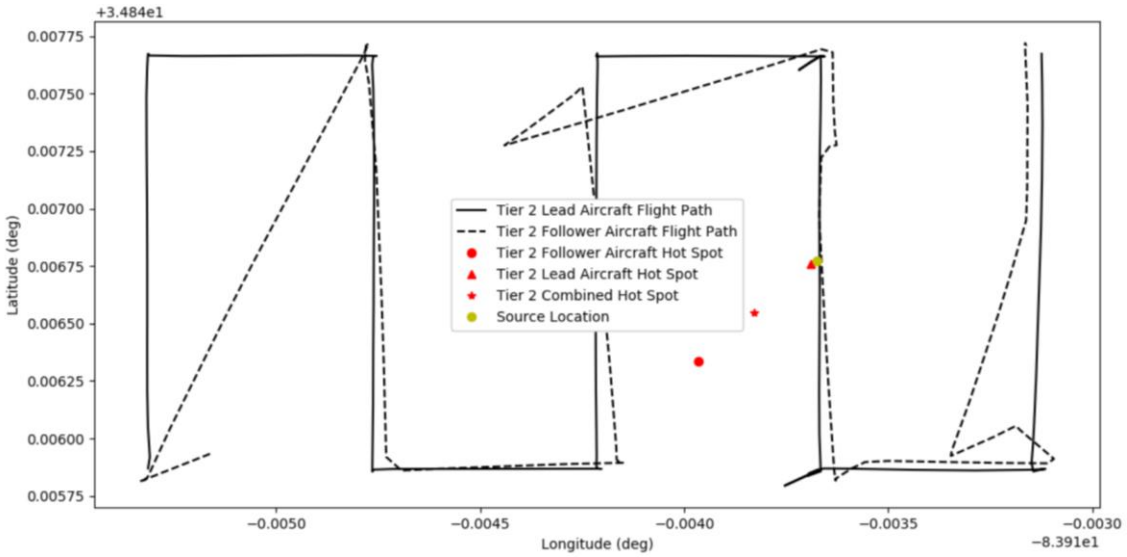

Figure A.142. Tier 2 Hot Spots and Source Locations for Pattern-Based Full Triple-Tier Simulation

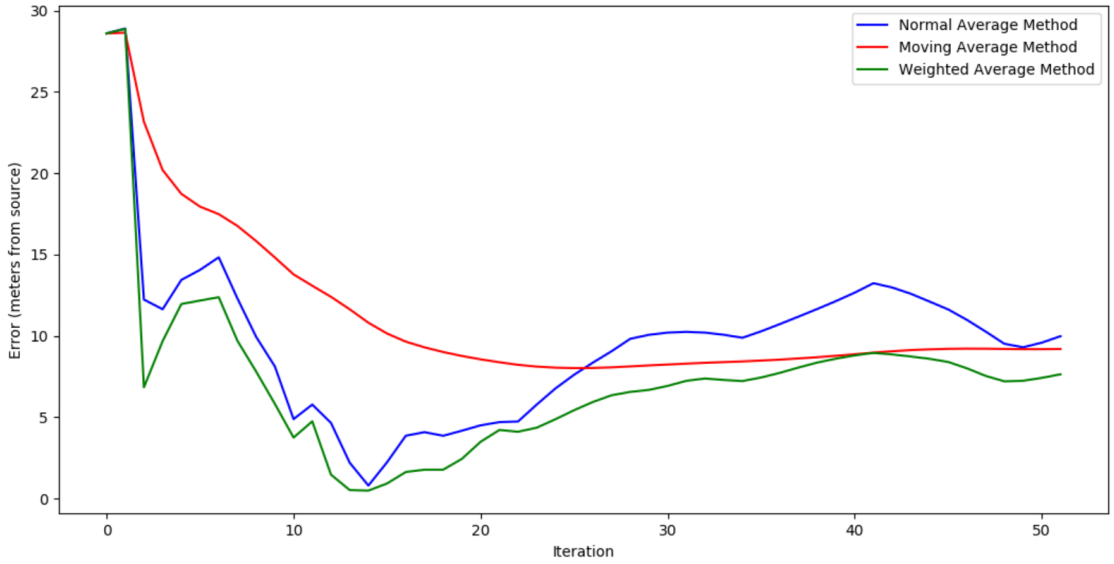

Figure A.143. Error of Tier 3 Search Methods from Pattern-Based Swarming Full Triple-Tier Simulation

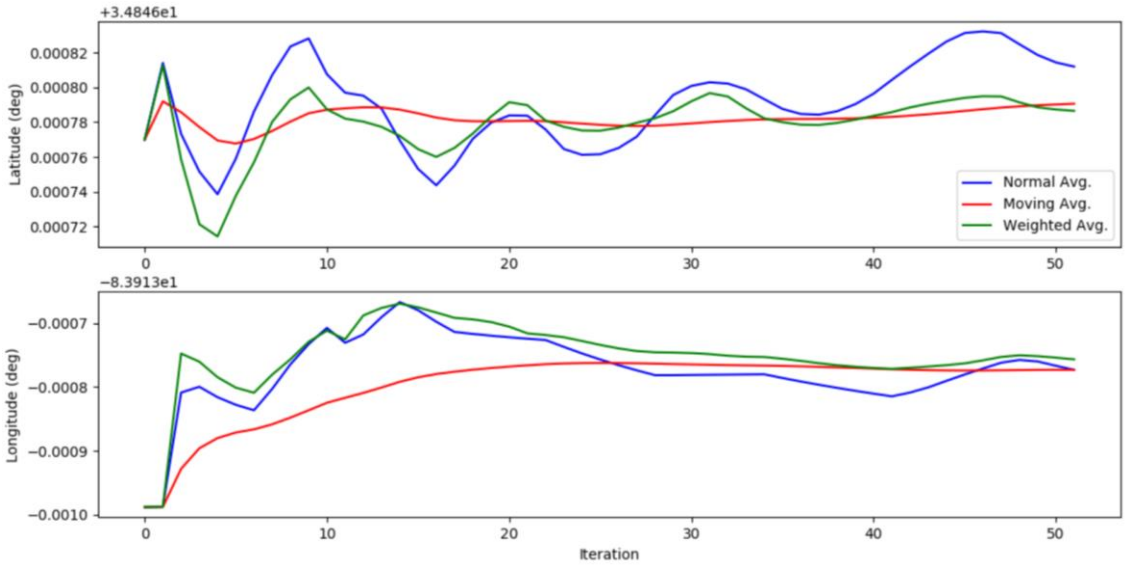

Figure A.144. Progression of Source Latitude and Longitude Estimates from Tier 3 Search Methods from the Pattern-Based Full Triple-Tier Simulation

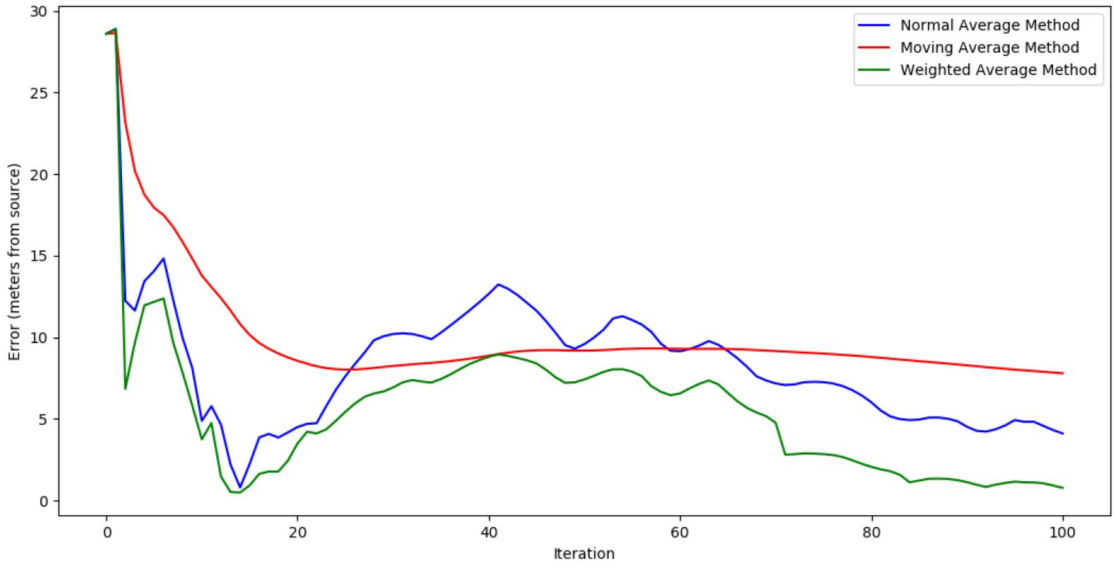

Figure A.145. Tier 3 Extended Refinement Time Search Results for Pattern-Based Swarming System

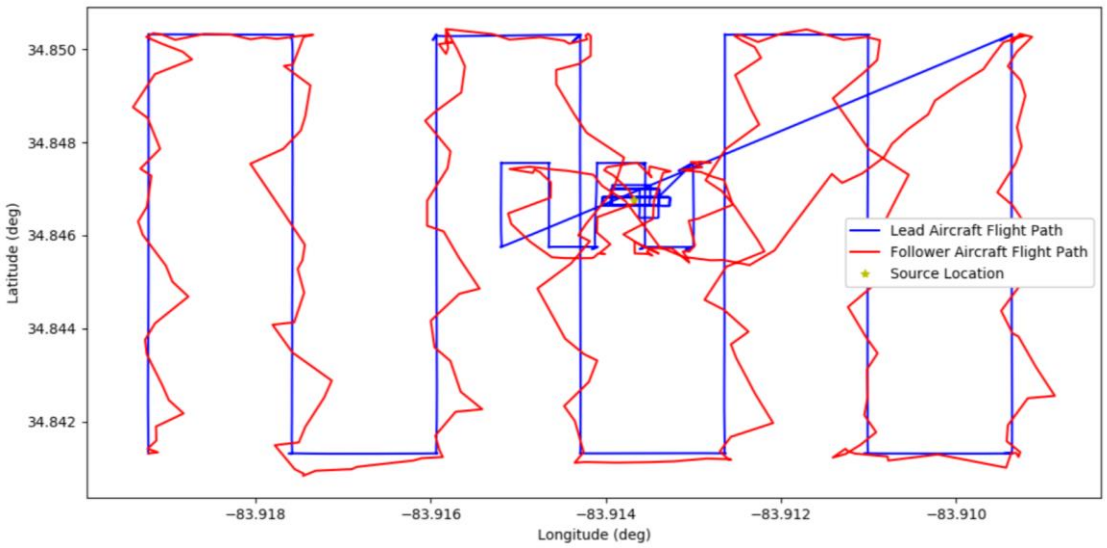

Figure A.146. Flight Paths of the Fully Simulated Triple-Tier Multi-Aircraft Bound-Continuity Swarming System Using PADUA

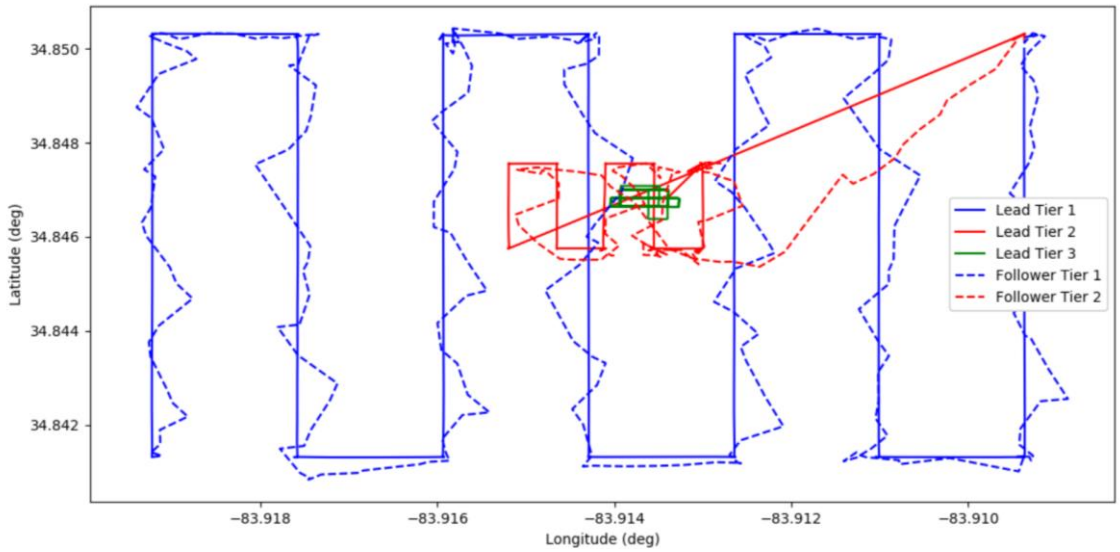

Figure A.147. Individual Tiers for Each Aircraft in Bound-Continuity Multi-Aircraft Swarming Full PADUA Search

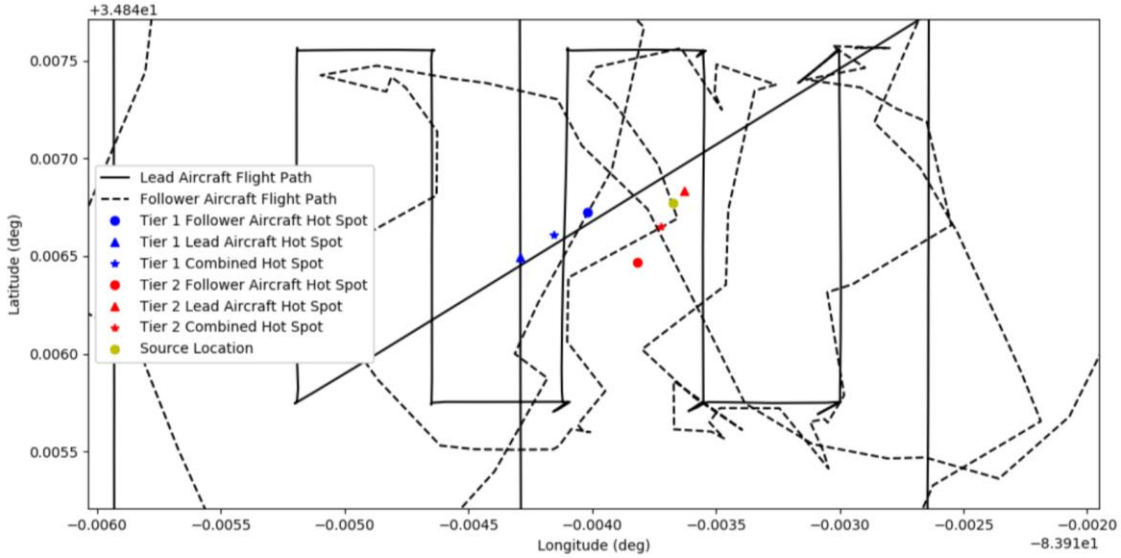

Figure A.148. Hot Spots and Source Locations for Bound-Continuity Full Triple-Tier Simulation Search

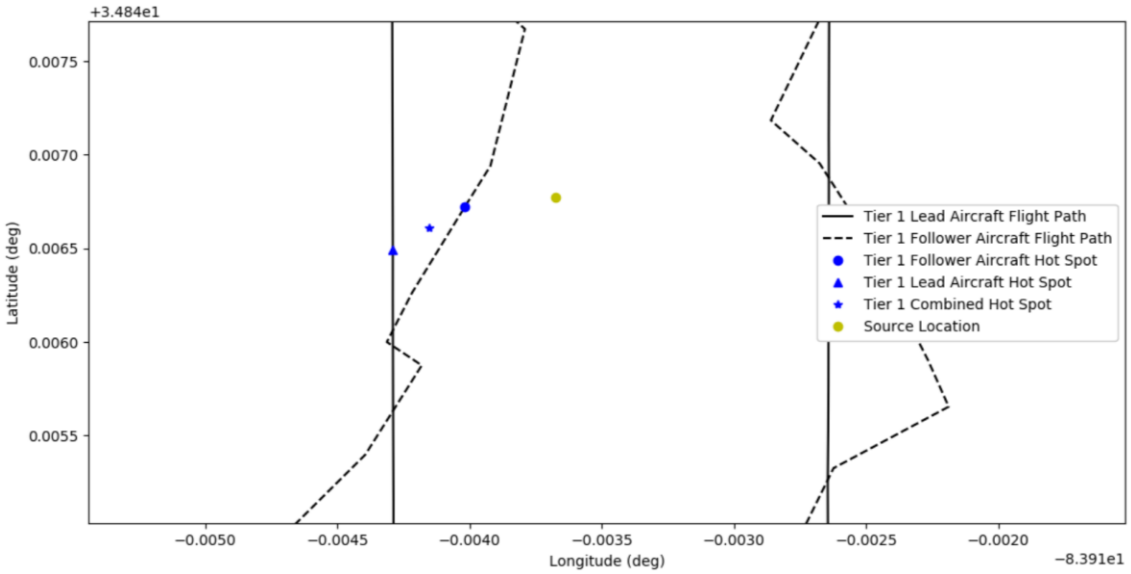

Figure A.149. Tier 1 Hot Spots and Source Locations for Bound-Continuity Full Triple-Tier Simulation

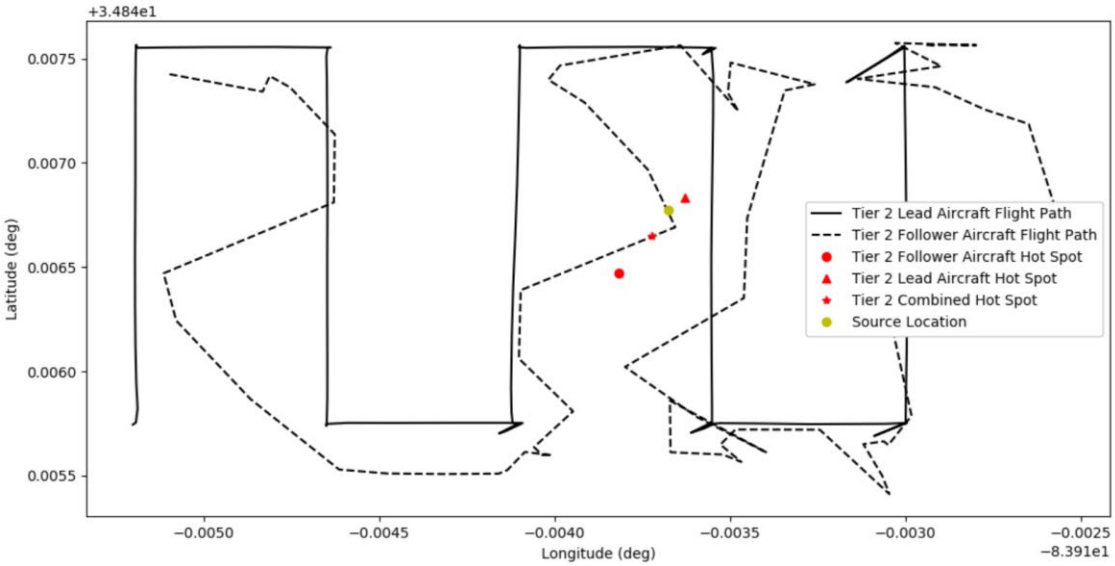

Figure A.150. Tier 2 Hot Spots and Source Locations for Bound-Continuity Full Triple-Tier Simulation

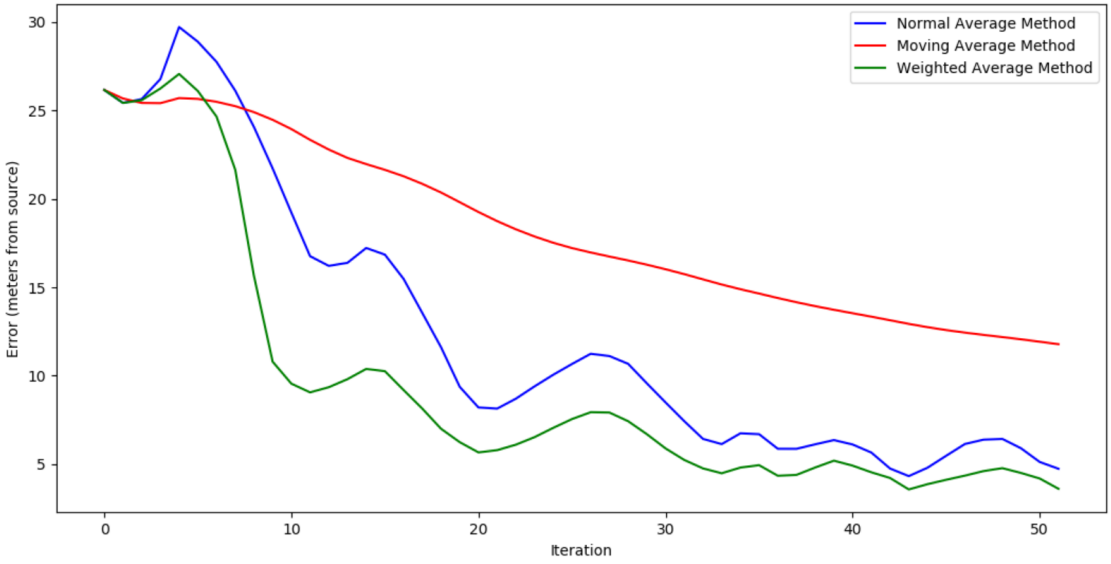

Figure A.151. Error of Tier 3 Search Methods from Bound-Continuity Swarming Full Triple-Tier Simulation

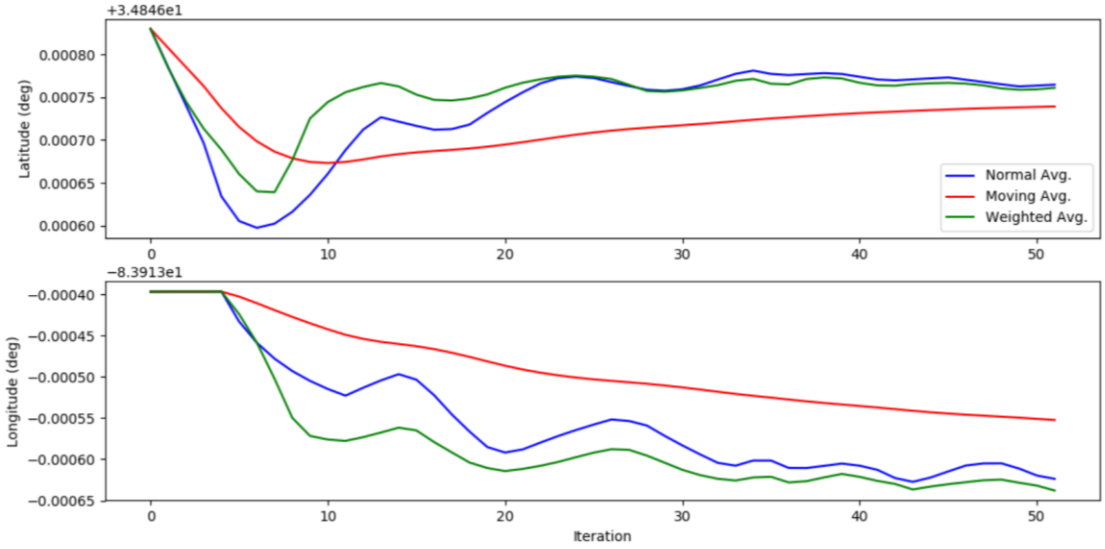

Figure A.152. Progression of Source Latitude and Longitude Estimates from Tier 3 Search Methods from the Bound-Continuity Full Triple-Tier Simulation

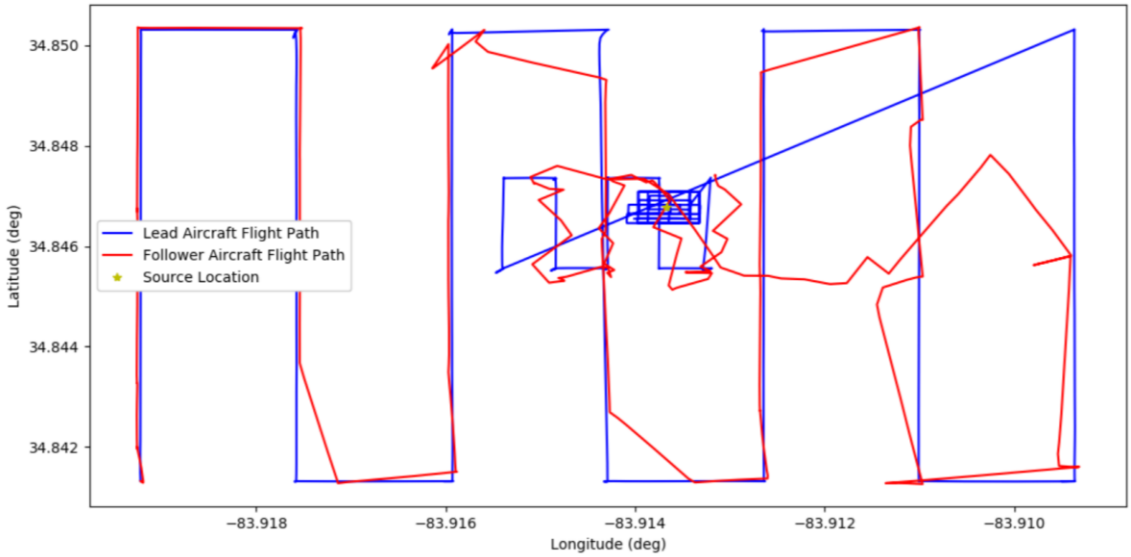

Figure A.153. Flight Paths of the Fully Simulated Triple-Tier Multi-Aircraft Combination Swarming System Using PADUA

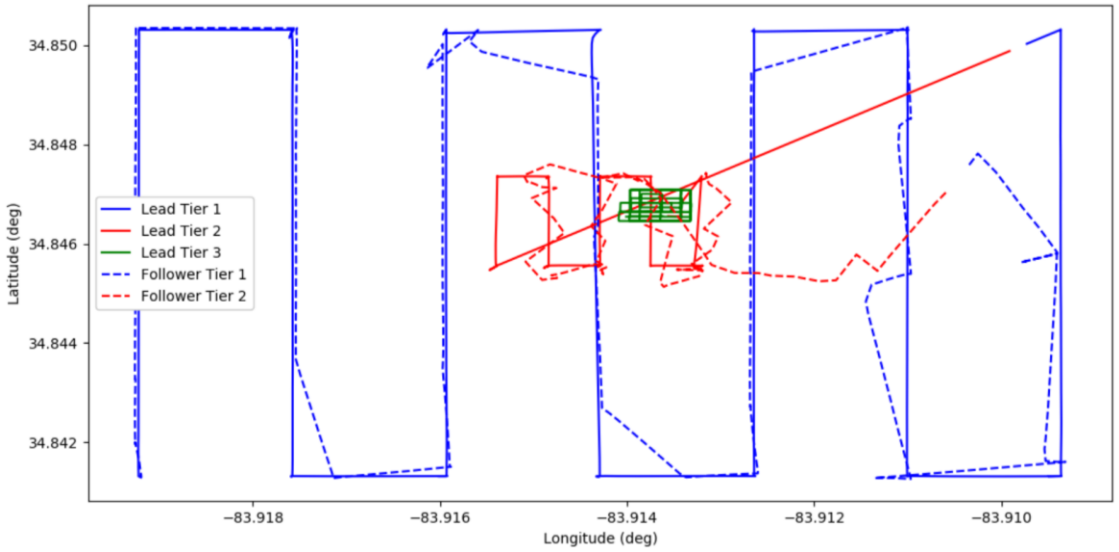

Figure A.154. Individual Tiers for Each Aircraft in Combination Multi-Aircraft Swarming Full PADUA Search

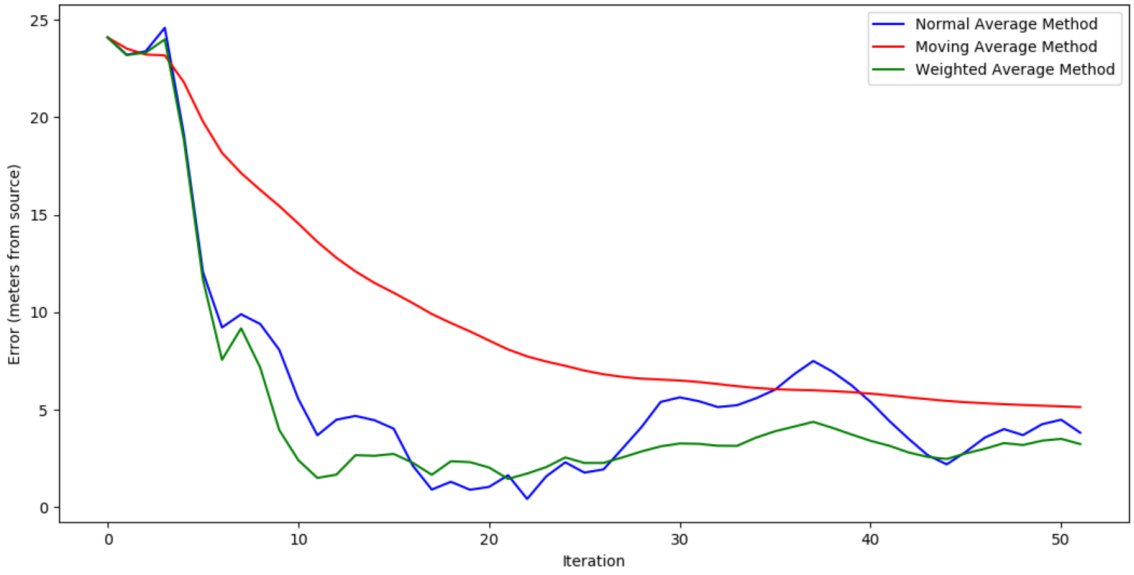

Figure A.155. Error of Tier 3 Search Methods from Combination Swarming Full Triple-Tier Simulation

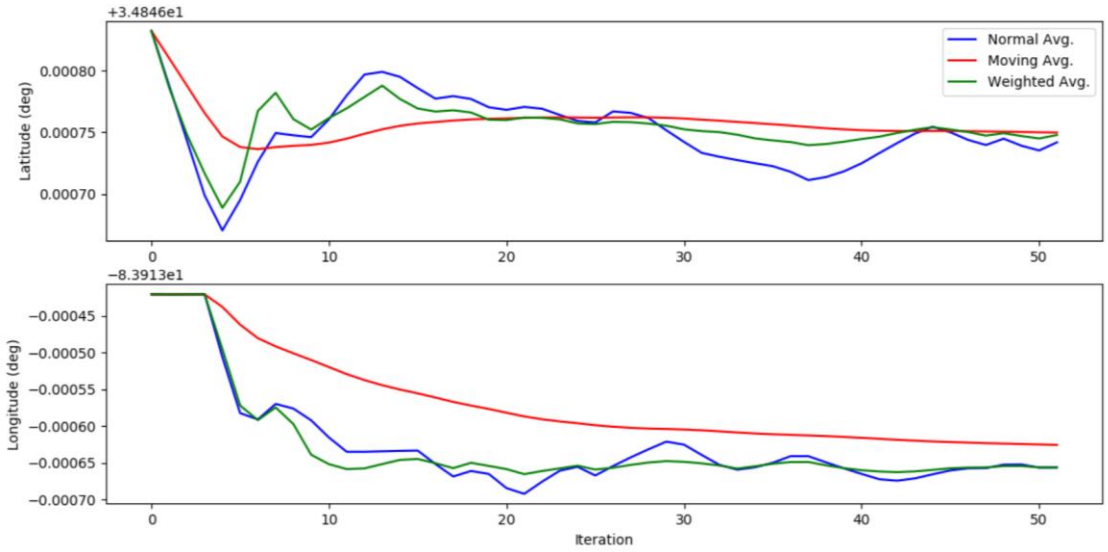

Figure A.156. Progression of Source Latitude and Longitude Estimates from Tier 3 Search Methods from the Combination Full Triple-Tier Simulation

Figure A.157. Google Earth Satellite Photograph of the Parts of Manhattan Used in PADUA Testing

Figure A.158. Flight Path of East Village Single Aircraft Test

Figure A.159. Zoomed Version of East Village Single Aircraft Test

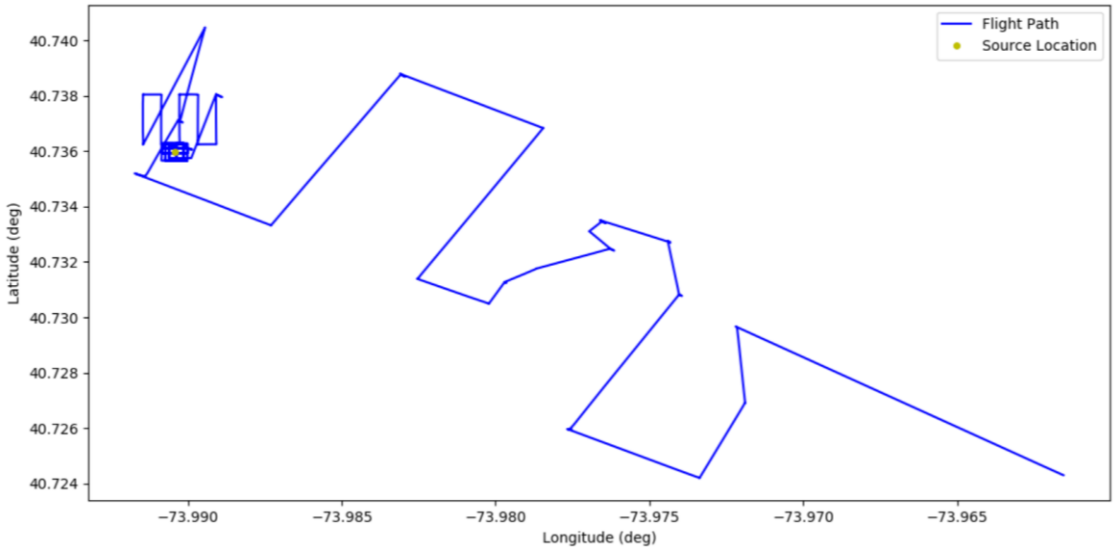

Figure A.160. Flight Path of East Village Test with No Background Underlay

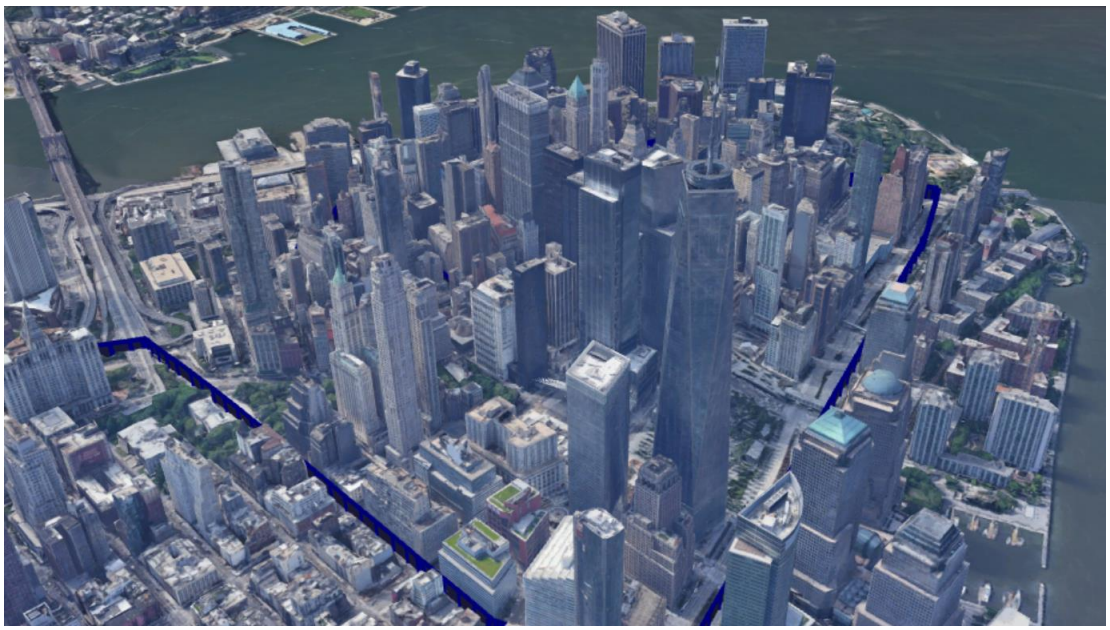

Figure A.161. Flight Path with View Looking South around Freedom Tower and One World Trade Center in Lower Manhattan

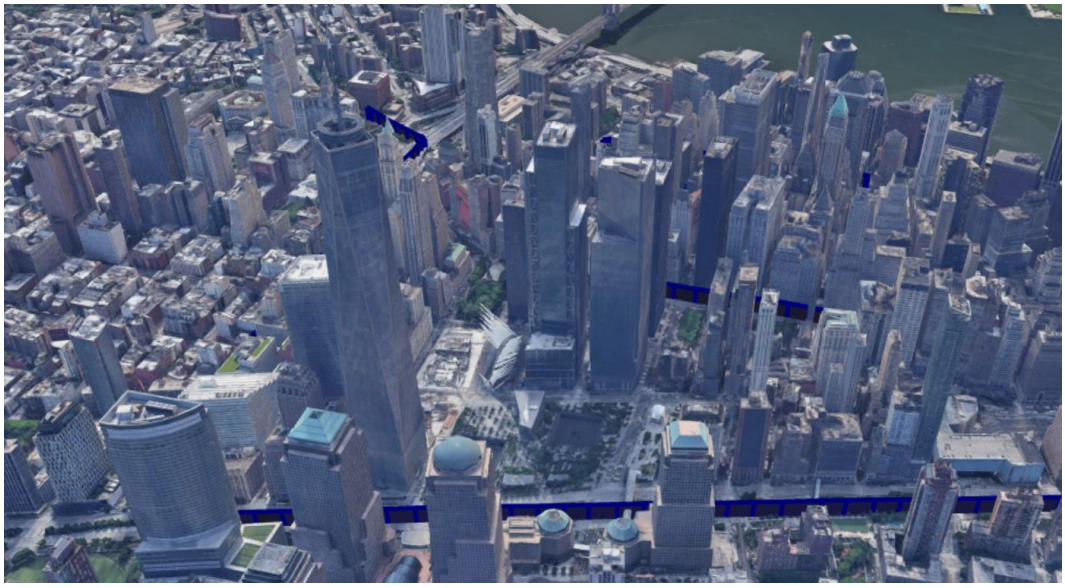

Figure A.162. Flight Path with View Looking East around Freedom Tower and One World Trade Center in Lower Manhattan with World Trade Center Memorial in View

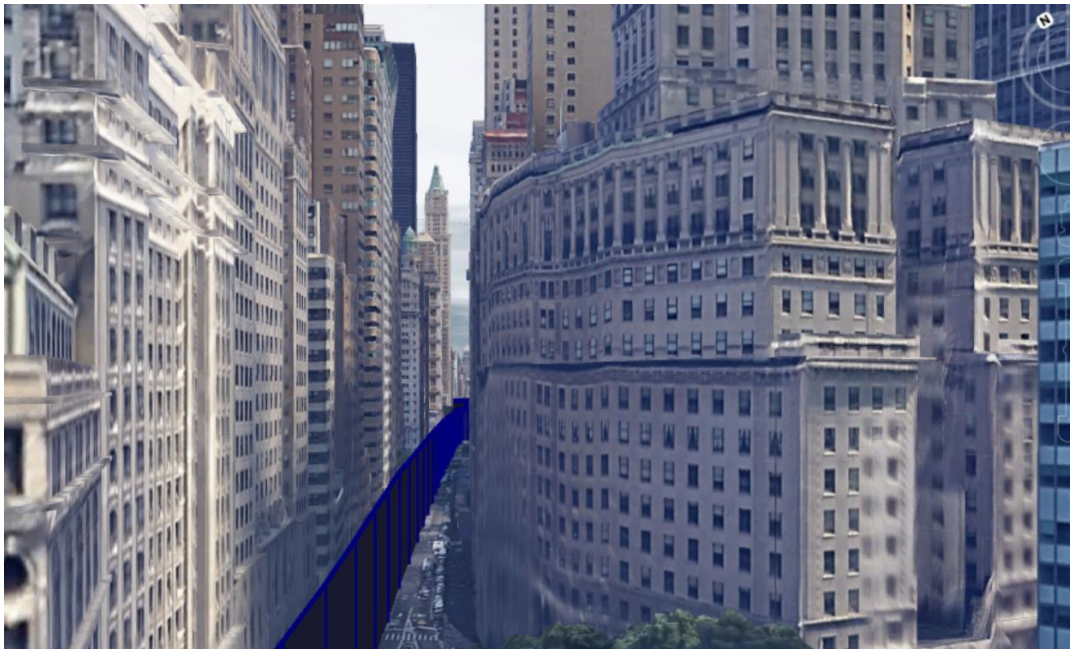

Figure A.163. Portion of Flight Path in Lower Manhattan with Flight Track through a 'Concrete Canyon'

Figure A.164. Downward Facing View of Flight Path with Google Earth Satellite Imagery Background

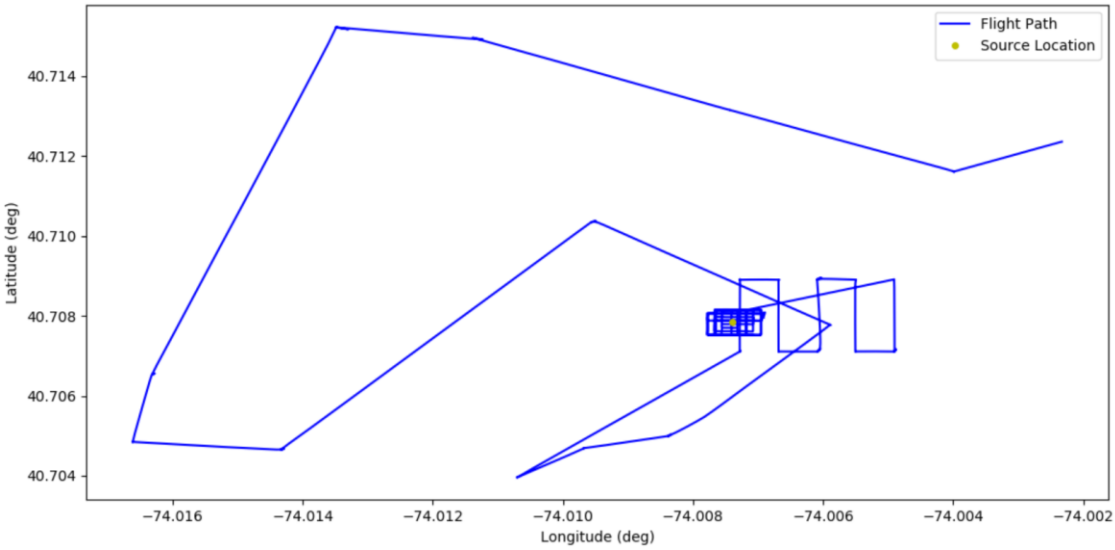

Figure A.165. Flight Path in Lower Manhattan with Blank Background

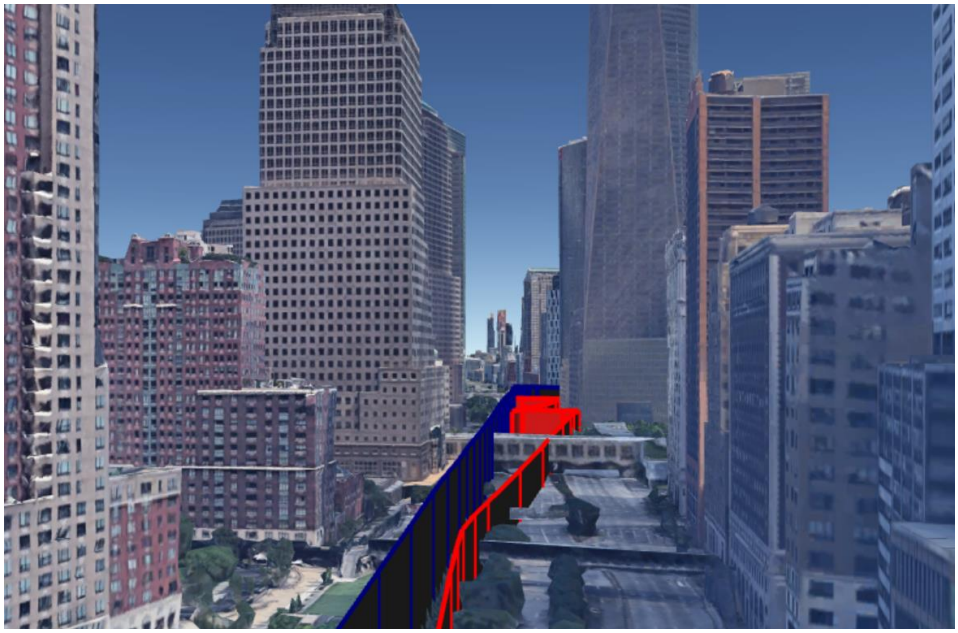

Figure A.166. Combination Swarming Algorithm PADUA Instance in Lower Manhattan at the Base of the Freedom Tower

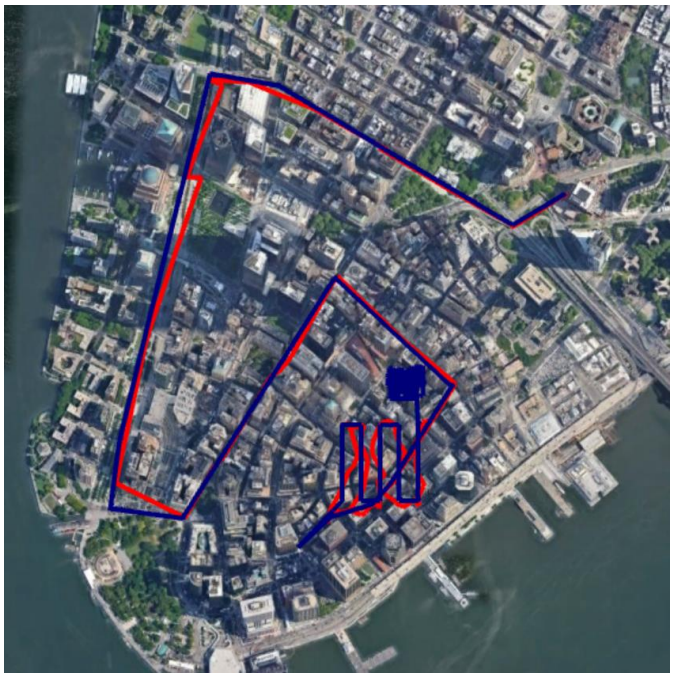

Figure A.167. Combined Algorithm Lower Manhattan Test Flight Paths (blue: lead aircraft; red: follower aircraft) with Satellite Background

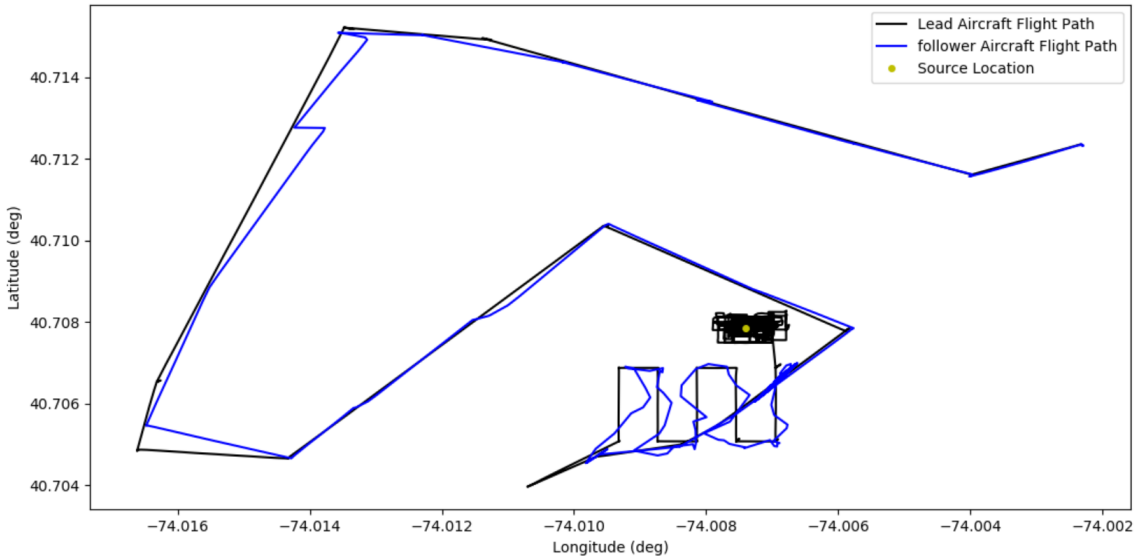

Figure A.168. Combined Algorithm Lower Manhattan Test Flight Paths with Blank Background

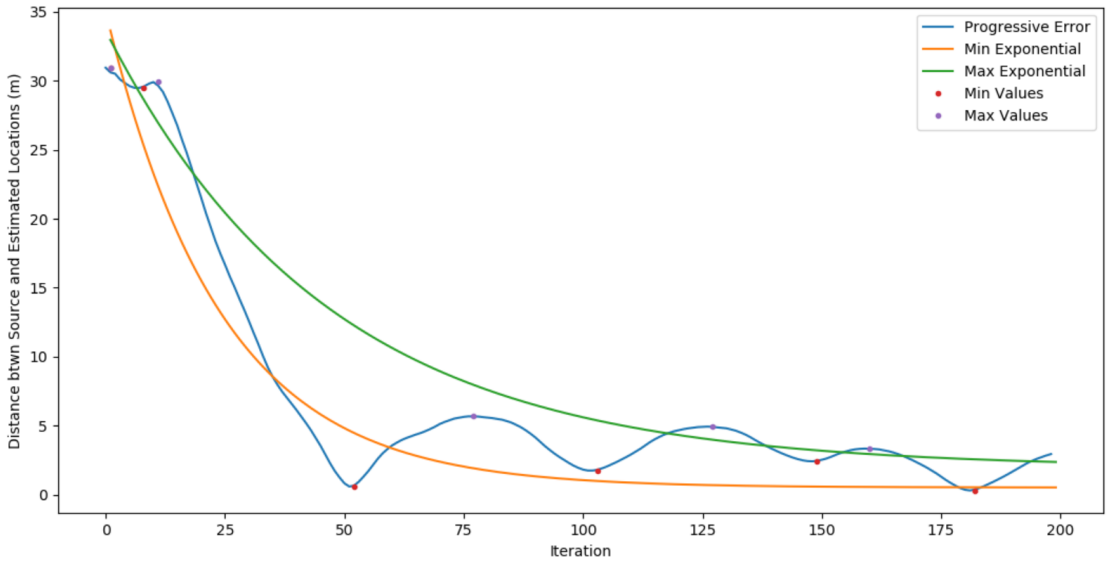

Figure A.169. Tier 3 Normal Average Best-Fit Progressive Error Prediction Curves

Appendix B: Tables

Table B.1. Basic Information about Initial Cesium Laboratory Calibration Testing

Distance from Source (mm)	Count Rate Samples Taken	Minimum Rate (c/min)	Maximum Rate (c/min)	Average Rate (c/min)	Standard Deviation (c/min)
50	180	207.20	273.23	236.64	11.41
75	180	137.76	181.30	159.65	8.53
100	180	115.62	147.19	129.61	6.52
200	180	57.03	76.95	66.08	3.84
300	180	38.25	56.10	46.72	2.98
400	180	33.17	44.17	37.83	2.06
750	60	29.25	39.09	33.98	2.18
1100	60	27.12	34.68	30.17	1.69

Table B.2. Comparison of Lambda Equation, Generated (Lambda Result Plus Noise), and Real Count Rate Values

Distance from Source (mm)	Theoretical Value Calculated (c/min)	Generated Avg. Value (c/min)	Real Value (c/min)	True Difference btw. Theoretical and Generated Values (c/min)	True Difference btw. Theoretical and Real Values (c/min)	True Difference btw. Generated and Real Values (c/min)	Percentage Difference Between Generated and Real Value (%)
50	235.503	235.200	236.639	0.303	1.136	1.439	0.608
75	164.180	163.871	159.650	0.309	4.530	4.221	2.644
100	125.625	125.688	129.613	0.063	3.988	3.925	3.029
200	65.703	65.658	66.077	0.046	0.373	0.419	0.634
300	47.228	47.203	46.725	0.025	0.503	0.478	1.022
400	39.409	39.413	37.829	0.004	1.580	1.584	4.187
750	32.133	32.049	33.982	0.083	1.849	1.933	5.687
1100	30.901	30.867	30.169	0.034	0.733	0.699	2.316

Table B.3. Scintillator System Part Descriptions

Part	Maker and Type
Plastic Scintillator	Eljen[112]
PMT	ADIT[113]
USB Base	Bridgeport Instruments Plug-on MCA Base[114]

Table B.4. Hot Spot Determination Methods Considered

Underlying Method	Determination Method	Method Notes
Bayesian	Simple Max	--
Bayesian	Weighted Average	Bayesian probability outputs used as weights
Bayesian	High Pass Weighted Average	Bayesian probability outputs used as weights if over average probability
S-G Filter	Simple Max	--
S-G Filter/ Bayesian	Weighted Average	S-G Filter applied to Bayesian probabilities to determine weights of locations
S-G Filter/ Bayesian	Average of S-G simple max and Bayesian weighted average determined locations	--

Table B.5. Accuracy Data from Real-World Simulation Testing for Tier 2 Search

Tier 2 Test Number	Distance from Max Count Rate Location to Source (m)	Distance from Max Bayesian Location to Source (m)	Distance from Max Weighted Bayesian Location to Source (m)
1	2.598	2.598	2.024
2	7.588	7.588	5.881
3	6.298	6.298	4.663
4	7.724	7.724	5.626

Table B.6. Results from Altitude Variation Tier 3 Initial Testing

Response Array Length	Altitude (m)	Distance from Estimation Location to Source (m)
20	10	3.013
20	15	2.969
20	20	4.487
20	25	4.691
20	30	16.35

Table B.7. Results from Response Array Length Variation Tier 3 Initial Testing

Response Array Length	Altitude (m)	Distance from Estimation Location to Source (m)
5	15	3.936
10	15	3.564
15	15	3.252
20	15	6.395
25	15	11.87

Table B.8. Tests of Non-Weighted vs. Weighted Averaging of Locations to Develop Source Location Estimate

Test Number	Average Method Estimated Source Location Error (m)	Weighted Average Method Estimated Source Location Error (m)	Percent Increase of Accuracy using Weighted Method (%)
1	3.252	1.387	57.36
2	6.648	4.591	30.94
3	6.725	2.075	69.14
4	5.512	4.339	21.27
5	6.082	3.142	48.34
6	6.673	4.684	29.81
7	6.190	3.430	44.59
8	3.599	1.466	59.28
9	3.583	4.626	-29.11
10	4.365	1.844	57.76

Table B.9: Single Aircraft Triple-Tier Test Inputs

Tier	Variable	Value
1	Start Latitude	34.841318 deg
1	Start Longitude	-83.919222 deg
1	Flight Path *x*-length	900m
1	Flight Path *x*-separation	150m
1	Flight Path *y*-length	1000m
1	Flight Path *y*-separation	1000m
1	Cruise Altitude	25m
1	Cruise Groundspeed	15m/s
2	Flight Path *x*-length	200m
2	Flight Path *x*-separation	50m
2	Flight Path *y*-length	200m
2	Flight Path *y*-separation	200m
2	Cruise Altitude	20m
2	Cruise Groundspeed	10m
3	Cruise Altitude	15m
3	Cruise Groundspeed	3m
3	Number of Recorded Count Rates	60
-	Source Latitude	34.84677282
-	Source Longitude	-83.91367476

Table B.10: Refinement of Source Location for Single Aircraft Triple-Tier Full PADUA Algorithm Search

Tier	Variable	Location ([lat,lon]) (deg)	Distance from Source (m)
-	Source Location	[34.8467728, -83.9136748]	0
1	Calculated Hot Spot	[34.8476248, -83.9131473]	106.3
2	Calculated Hot Spot	[34.8473179, -83.9135646]	61.43
3	Final Source Location Estimate Using Location Average	[34.8467728, -83.9136683]	0.5934
3	Final Source Location Estimate Using Moving Location Average	[34.8467603, -83.9136724]	1.404
3	Final Source Location Estimate Using Weighted Average	[34.8467738, -83.9136757]	0.1386

Table B.11. Time Required for Tier Instances for Single Aircraft Full Triple-Tier Test

Tier	Time Elapsed (min)
1	15.60
2	7.808
3	15.45
All	38.85

Table B.12: Antenna Comparison

Test Description	Standard Latitude Deviation (deg)	Standard Longitude Deviation (deg)	Deviation Range (m)
Short Antenna Inside (obstructed)	2.06E-05	2.89E-05	15.5
Long Antenna Inside (obstructed)	7.51E-05	4.77E-05	36.5
Short Antenna Outside (open)	1.47E-05	1.58E-05	7.8
Long Antenna Outside (open)	9.07E-06	1.43E-05	4.9

Table B.13: Algorithm Type Hot Spot Error Comparison For Tiers 1 and 2

Tier	Pattern Swarming Hot Spot Distance from Source (m)	Bound-Continuity Swarming Hot Spot Distance from Source (m)
1	23.88	23.70
2	10.72	7.442

Table B.14: Pattern-Based Swarming PADUA Search for Multi-Agent System

Tier	Variable	Value
1	Start Latitude	34.841318 deg
1	Start Longitude	-83.919222 deg
1	Flight Path *x*-length	900m
1	Flight Path *x*-separation	150m
1	Flight Path *y*-length	1000m
1	Flight Path *y*-separation	1000m
1	Cruise Altitude	25m
1	Cruise Groundspeed	10m/s
2	Flight Path *x*-length	200m
2	Flight Path *x*-separation	50m
2	Flight Path *y*-length	200m
2	Flight Path *y*-separation	200m
2	Cruise Altitude	20m
2	Cruise Groundspeed	10m
3	Cruise Altitude	15m
3	Cruise Groundspeed	3m
3	Number of Recorded Count Rates	62
-	Source Latitude	34.84677282
-	Source Longitude	-83.91367476

Table B.15: Hot Spot Location and Error Information for Pattern-Based Swarming Multi-Aircraft Full Triple-Tier PADUA Simulation

Tier	Aircraft	Variable	Location ([lat,lon]) (deg)	Distance from Source (m)
-	-	Source Location	[34.8467728, -83.9136748]	0
1	1	Calculated Hot Spot	[34.8466584, -83.9142905]	57.61
1	2	Calculated Hot Spot	[34.8467832, -83.9142531]	52.79
1	Combined	Calculated Hot Spot	[34.8467208, -83.9142718]	54.79
2	1	Calculated Hot Spot	[34.8467585, -83.9136905]	2.137
2	2	Calculated Hot Spot	[34.8463342, -83.9139650]	55.50
2	Combined	Calculated Hot Spot	[34.8465464, -83.9138277]	28.79
3	1	Final Source Location Estimate Using Location Average	[34.8468119, -83.9137732]	9.977
3	1	Final Source Location Estimate Using Moving Location Average	[34.8467906, -83.9137732]	9.194
3	1	Final Source Location Estimate Using Weighted Average	[34.8467865, -83.9137567]	7.635

Table B.16: Time Required for Each Tier of the Full System Pattern-Based Swarming PADUA Search Algorithm Simulation

Tier	Time Elapsed (min)
1	19.79
2	10.67
3	15.70
All	46.16

Table B.17: Bound-Continuity Swarming PADUA Search for Multi-Agent System

Tier	Variable	Value
1	Start Latitude	34.841318 deg
1	Start Longitude	-83.919222 deg
1	Flight Path *x*-length	900m
1	Flight Path *x*-separation	150m
1	Flight Path *y*-length	1000m
1	Flight Path *y*-separation	1000m
1	Cruise Altitude	25m
1	Cruise Groundspeed	10m/s
2	Flight Path *x*-length	200m
2	Flight Path *x*-separation	50m
2	Flight Path *y*-length	200m
2	Flight Path *y*-separation	200m
2	Cruise Altitude	20m
2	Cruise Groundspeed	10m
3	Cruise Altitude	15m
3	Cruise Groundspeed	3m
3	Number of Recorded Count Rates	62
-	Source Latitude	34.84677282
-	Source Longitude	-83.91367476

Table B.18: Hot Spot Location and Error Information for Bound-Continuity Swarming Multi-Aircraft Full Triple-Tier PADUA Simulation

Tier	Aircraft	Variable	Location ([lat,lon]) (deg)	Distance from Source (m)
-	-	Source Location	[34.8467728, -83.9136748]	0
1	1	Calculated Hot Spot	[34.8464932, -83.9142899]	64.17
1	2	Calculated Hot Spot	[34.8467219, -83.9140175]	31.79
1	Combined	Calculated Hot Spot	[34.8466076, -83.9141537]	47.41
2	1	Calculated Hot Spot	[34.8468329, -83.9136280]	7.927
2	2	Calculated Hot Spot	[34.8464705, -83.9138168]	36.03
2	Combined	Calculated Hot Spot	[34.8466517, -83.9137224]	14.15
3	1	Final Source Location Estimate Using Location Average	[34.8467646, -83.9136237]	4.747
3	1	Final Source Location Estimate Using Moving Location Average	[34.8467391, -83.9135523]	11.79
3	1	Final Source Location Estimate Using Weighted Average	[34.8467609, -83.9136379]	3.615

Table B.19: Time Required for Each Tier of the Full System Bound-Continuity Swarming PADUA Search Algorithm Simulation

Tier	Time Elapsed (min)
1	18.88
2	11.26
3	11.77
All	41.91

Table B.20: Combination Swarming PADUA Search for Multi-Agent System

Tier	Variable	Value
1	Start Latitude	34.841318 deg
1	Start Longitude	-83.919222 deg
1	Flight Path *x*-length	900m
1	Flight Path *x*-separation	150m
1	Flight Path *y*-length	1000m
1	Flight Path *y*-separation	1000m
1	Cruise Altitude	25m
1	Cruise Groundspeed	10m/s
2	Flight Path *x*-length	200m
2	Flight Path *x*-separation	50m
2	Flight Path *y*-length	200m
2	Flight Path *y*-separation	200m
2	Cruise Altitude	20m
2	Cruise Groundspeed	10m
3	Cruise Altitude	15m
3	Cruise Groundspeed	3m
3	Number of Recorded Count Rates	62
-	Source Latitude	34.84677282
-	Source Longitude	-83.91367476

Table B.21: Hot Spot Location and Error Information for Combination Swarming Multi-Aircraft Full Triple-Tier PADUA Simulation

Tier	Aircraft	Variable	Location ([lat,lon]) (deg)	Distance from Source (m)
-	-	Source Location	[34.8467728, -83.9136748]	0
1	1	Calculated Hot Spot	[34.8466775, -83.9143463]	62.19
1	2	Calculated Hot Spot	[34.8461452, -83.9143524]	93.24
1	Combined	Calculated Hot Spot	[34.8464114, -83.9143494]	73.52
2	1	Calculated Hot Spot	[34.8467135, -83.9138325]	15.83
2	2	Calculated Hot Spot	[34.8463589, -83.9138551]	48.88
2	Combined	Calculated Hot Spot	[34.8465362, -83.9138438]	30.50
3	1	Final Source Location Estimate Using Location Average	[34.8467418, -83.9136566]	3.821
3	1	Final Source Location Estimate Using Moving Location Average	[34.8467499, -83.9136259]	5.135
3	1	Final Source Location Estimate Using Weighted Average	[34.8467478, -83.9136566]	3.240

Table B.22: Time Required for Each Tier of the Full System Combination Swarming PADUA Search Algorithm Simulation

Tier	Time Elapsed (min)
1	18.87
2	11.25
3	19.03
All	49.15

Table B.23: Results of Interest from East Village Single Aircraft Search

Variable	Value
Total Distance Traveled	17560m
Total Time Required for Search	59.02min
Average Groundspeed	4.960m/s
Tier 1 Hot Spot Distance from Source	126.3m
Tier 2 Hot Spot Distance from Source	94.94m
Tier 3 Final Normal Average Method Distance from Source	1.176m
Tier 3 Final Weighted Average Method Distance from Source	0.2557m

Table B.24: Results from Lower Manhattan Single Aircraft PADUA Search

Variable	Value
Total Distance Traveled	8465m
Total Time Required for Search	49.29min
Average Groundspeed	2.390m/s
Tier 1 Hot Spot Distance from Source	105.8m
Tier 2 Hot Spot Distance from Source	28.12m
Tier 3 Final Normal Average Method Distance from Source	3.863m
Tier 3 Final Weighted Average Method Distance from Source	1.704m

Table B.25: Results from Lower Manhattan Multi-Aircraft PADUA Search

Variable	Value
Lead Aircraft Total Distance Traveled	10659m
Total Time Required for Search	58.98min
Average Groundspeed	3.010m/s
Tier 1 Hot Spot Distance from Source	223.8m
Tier 2 Hot Spot Distance from Source	11.10m
Tier 3 Final Normal Average Method Distance from Source	4.587m
Tier 3 Final Weighted Average Method Distance from Source	2.885m

Appendix C: Example Python Basic Iterative Multithreading Code

```
1.  from threading import Thread, Lock
2.  import threading
3.  # Rename Python 2's raw_input to input
4.  try:
5.      input = raw_input
6.  except NameError:
7.      pass
8.
9.  lock = Lock()
10.
11. def Thread_1():
12.     import time
13.     global ttt
14.     a1=float(time.time())
15.     i=0
16.     while i<100000000:
17.         i=i+1
18.         #print(i)
19.     b1=float(time.time())
20.     ttt=(b1-a1)
21.     print('\n                          time1=%s'%(b1-a1))
22.
23. def Thread_2():
24.     import time
25.     global ttt
26.     a1=float(time.time())
27.     i=0
28.     while i<100000000:
29.         i=i+1
30.         #print(i)
31.     b1=float(time.time())
32.     print('\n                          time2=%s'%(b1-a1))
33.     time.sleep(1)
34.     print('total time = %s'%(((b1-a1)+ttt)))
35.
36.
37.
38. threadOne = threading.Thread(target = Thread_1, args=()) #command send thread
39. threadTwo = threading.Thread(target = Thread_2, args=())
40.
41. threadOne.start()
42. threadTwo.start()
```

Appendix D: Example Python Basic Iterative Series Code

```
import time
a1=float(time.time())
i=0
while i<100000000:
    i=i+1
    #print(i)
i=0
while i<100000000:
    i=i+1
    #print(i)

b1=float(time.time())
print('\n                        time3=%s'%(b1-a1))
```

Appendix E: Example Python Shared Variable Multithreaded Code

```
import time
import math
from threading import Thread, Lock
import threading
# Rename Python 2's raw_input to input
try:
    input = raw_input
except NameError:
    pass

lock = Lock()

def Thread_1():
    global gi
    a1=float(time.time())
    ait=[]
    while True:
        try:
            lock.acquire()
            var=gi
            lock.release()
        except:
            try:
                lock.release()
            except:
                pass
            var=-1
        if var%10==0:
            print(var)
        if var>=999:
            break

        if var!=-1:
            for ii in range(100):
                ait.append(str('%s'%(var+ii)))
            var2=var**2
            var3=var**2**2
            var4=float(var2+var3)
            var5=var4+int(time.time())+int(math.pi)
            var6=var2**math.cos(.3)
            var7=var6**.2
            if int(var7)<int(var4):
                var8=math.sin(.1)+math.cos(.2)
            else:
                var8=math.sin(.1)-math.cos(.2)



    b1=float(time.time())
    print('\n                          time1=%s'%(b1-a1))

def Thread_2():
    global gi
    a1=float(time.time())
```

```
    i=0
    while i<1000:
        try:
            lock.acquire()
            gi=i
            lock.release()
        except:
            try:
                lock.release()
            except:
                pass
        i=i+1
    c1=float(time.time())
    print('\n                          time2=%s'%(c1-a1))

threadOne = threading.Thread(target = Thread_1, args=()) #command send thread

threadTwo = threading.Thread(target = Thread_2, args=())
threadOne.start()
threadTwo.start()
```

Appendix F: Example Python Shared Variable Equivalent Series Code

```
import time
import math
a1=time.time()
i=0
ait=[]
while i<1000:
    si=i
    i=i+1
    if si%10==0:
        print(si)
    if si>=999:
        break
    if si!=-1:
        for ii in range(100):
            ait.append(str('%s'%(si+ii)))
        var2=si**2
        var3=si**2**2
        var4=float(var2+var3)
        var5=var4+int(time.time())+int(math.pi)
        var6=var2**math.cos(.3)
        var7=var6**.2
        if int(var7)<int(var4):
            var8=math.sin(.1)+math.cos(.2)
        else:
            var8=math.sin(.1)-math.cos(.2)


b1=time.time()
print('                                                    total series time = %s'%
(b1-a1))
```

Appendix G: Complete Command Encoding Readme File

```
Instructions:
       To use this documentation, you must understand the method of the
communication arrays.
       For example, if a received string is written as ABC, each of the places
(A, B, and C) might have a length of 1 or a longer length such as a gps
waypoint. If a set of data is 8 digits represented by A, A will actually be
Aabcdefgh where A is a data signifier anda-h are the numerals themselves. But
using this method, if A is not used, it can be a single 0.
FORMAT FOR COMMUNICATION STRING
Key:
C=#: Primary signifier
     IF C=#  {#}: If the primary signifier meets a certain condition {number
of digits for this selection; for this case, Aab would have {3}}
       a=#: numeral
       b=#: numeral


Keys:
       RECEIVING AIRCRAFT: [ABCDEFGHIJKLMNOPQRST]
A= 0   {1}
B= vehicle number   {1}
C= 0: control method      {1}
       C=0: standard aircraft control
       C=1: standard ground vehicle control
       C=2: standard surface water vehicle control
       C=3: standard submersible control
D= vehicle number   {1}

E= string length numeral 1{1}
F= string length numeral 2{1}

G= mode change      {1}
       G=0: continue/no change
       G=1: GUIDED
       G=2: ALT_HOLD
       G=3: LOITER
       G=4: AUTO
       G=5: RTL
H= what kind of command is being sent
     IF H=0: continue/no change/no new commands      {1}
     IF H=1: simple goto (goto gps waypoint)  {20}
       a= waypoint latitude numeral 1
       b= waypoint latitude numeral 2
       c= waypoint latitude numeral 3
       d= waypoint latitude numeral 4
       e= waypoint latitude numeral 5
       f= waypoint latitude numeral 6
       g= waypoint latitude numeral 7
       h= waypoint latitude numeral 8
       i= latitude hemisphere (1=north, 0=south)
       j= waypoint longitude numeral 1
```

```
       k= waypoint longitude numeral 2
       l= waypoint longitude numeral 3
       m= waypoint longitude numeral 4
       n= waypoint longitude numeral 5
       o= waypoint longitude numeral 6
       p= waypoint longitude numeral 7
       q= waypoint longitude numeral 8
       r= waypoint longitude numeral 9
       s= longitude hemisphere (1=east, 0=west)
     IF H=2: fly by direction    {7}
       a= fly by direction north/south direction (1=north, 0=south)
       b= fly by direction north/south meters numeral 1
       c= fly by direction north/south meters numeral 2
       d= fly by direction east/west direction (1=east, 0=west)
       e= fly by direction east/west meters numeral 1
       f= fly by direction east/west meters numeral 2
     IF H=3: fly by velocity in a direction for given time
       a= fly by velocity north/south direction (1=north, 0=south)
       b= fly by velocity north/south m per s numeral 1
       c= fly by velocity north/south m per s numeral 2
       d= fly by velocity east/west direction (1=east, 0=west)
       e= fly by velocity east/west m per s numeral 1
       f= fly by velocity east/west m per s numeral 2
     IF H=4: hover {1}

I= altitude change?
     IF I=0: continue/no change/no new commands      {1}
     IF I=1: go to altitude in meters {5}
       a= altitude in meters numeral 1
       b= altitude in meters numeral 2
       c= altitude in meters numeral 3
       d= altitude in meters numeral 4
     IF I=2: move a number of meters up or down {6}
       a= change direction (1=up, 0=down)
       b= change in meters numeral 1
       c= change in meters numeral 2
       d= change in meters numeral 3
       e= change in meters numeral 4

J=0 reserved for future commands {1}
K=0 reserved for future commands {1}
L=0 reserved for future commands {1}
M=0 reserved for future commands {1}
N=0 reserved for future commands {1}
O= string length numeral 1{1}
P= string length numeral 2{1}

Q= vehicle number   {1}
R= 0   {1}
S= vehicle number   {1}
T=0    {1}
```

```
FORMAT FOR SHORT COMMAND SEQUENCE

ABCDEFGHI
000#(##)#000
Key:
A=0    {1}
B=0    {1}
C=0    {1}
D=contained information from vehicle    {1}
   D=0 No information available
   D=1 Initial communication contact successful
   D=2 Initialization of vehicle successful
E=extra vehicle information      {1}
   if on first contact, vehicle type:
       E=0: standard aircraft control (i.e. 'aircraft')
       E=1: standard ground vehicle control (i.e. 'rover')
       E=2: standard surface water vehicle control (i.e. 'boat')
       E=3: standard submersible control (i.e. 'sub')
   else:
        E=0
F=vehicle number    {2}
G=contained information from ground station   {1}
   G=0 Waiting for initial communication contact response
   G=1 Initialize vehicle and takeoff (if applicable)
   G=2 Begin Ping
H=0
I=0
J=0
```

VITA

Benjamin Lajos Magocs was born to Stephen and Nancy Magocs in Knoxville Tennessee. He attended First Lutheran School from preschool through middle school and then graduated from Berean High School. Lajos also earned the rank of Eagle Scout in 2011. He then studied Aerospace Engineering and Engineering Entrepreneurship at the University of Tennessee for his Bachelor's degree which he earned in May 2016. He continued working with the Institute for Nuclear Security upon accepted a graduate research position in June 2016 for his Master's in Aerospace Engineering which he earned in May 2017. He again accepted a graduate research position from the INS in June 2017.