A TRIDENT SCHOLAR PROJECT REPORT

NO. 469

Persistent Target Detection and Tracking By An Autonomous Swarm

by

Midshipman 1/C John J. Gainer Jr., USN

UNITED STATES NAVAL ACADEMY
ANNAPOLIS, MARYLAND

USNA-1531-2

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
05-21-18		

4. TITLE AND SUBTITLE
Persistent Target Detection and Tracking By An Autonomous Swarm

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Gainer, John J.

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

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

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Naval Academy
Annapolis, MD 21402

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
Trident Scholar Report no. 469 (2018)

12. DISTRIBUTION / AVAILABILITY STATEMENT

This document has been approved for public release; its distribution is UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This paper presents an autonomous multivehicle control algorithm capable of persistently searching and tracking targets in a defined search area subject to operational endurance constraints of individual agents. A small-scale system serves as proof of concept for larger systems that are employed in operational environments. The underlying goal is to design a modular control architecture that can be modified to any type of autonomous vehicle, search area, or target. In practical application, a target can be anything from heat signatures to radioactive material; therefore, this project will simulate a generic emitter-detector pair as a placeholder relationship for real world applications. The control strategy accounts for the appearance, motion, and disappearance of multiple targets in the search space constituting the utility of creating a team of multiple search agents. When agent battery level drops below a predetermined threshold, the agent returns to a base station to recharge and be relaunched into the mission. Remaining agents must account for this loss and gain of other team members as they exit the search environment. The contributions of this work are 1) the design of search trajectories for autonomous vehicles with limited endurance, 2) incorporation of return-to-base and recharge time requirements, and 3) coordination of multiple vehicles by developing a decision-making model to and assign agents to operational modes. Each of these components enable persistent multivehicle operations. Simulation results are intended for implementation on a system of quadrotors complemented by a system capable of autonomously recharging vehicles to sustain a multivehicle team beyond the mission life of a single vehicle.

15. SUBJECT TERMS
Persistent, Swarm, Detection and Tracking, Autonomous, Multivehicle Coordination

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**		44	**19b. TELEPHONE NUMBER** *(include area code)*

U.S.N.A. --- Trident Scholar project report; no. 469 (2018)

PERSISTENT TARGET DETECTION AND TRACKING BY AN AUTONOMOUS SWARM

by

Midshipman 1/C John J. Gainer Jr.
United States Naval Academy
Annapolis, Maryland

_____
(signature)

Certification of Advisers Approval

Assistant Professor Levi D. DeVries
Weapons and Systems Engineering Department

_____
(signature)

Assistant Professor Michael D. M. Kutzer
Weapons and Systems Engineering Department

_____
(signature)

_____
(date)

Professor Bradley E. Bishop
Weapons and Systems Engineering Department

_____
(signature)

_____
(date)

Acceptance for the Trident Scholar Committee

Professor Maria J. Schroeder
Associate Director of Midshipman Research

_____
(signature)

_____
(date)

USNA-1531-2

Final Trident Report

Persistent Target Detection and Tracking by an Autonomous Swarm

May 7, 2018

Abstract

This paper presents an autonomous multivehicle control algorithm capable of persistently searching and tracking targets in a defined search area subject to operational endurance constraints of individual agents. A small-scale system serves as proof of concept for larger systems that are employed in operational environments. The underlying goal is to design a modular control architecture that can be modified to any type of autonomous vehicle, search area, or target. In practical application, a target can be anything from heat signatures to radioactive material; therefore, this project will simulate a generic emitter-detector pair as a placeholder relationship for real world applications. The control strategy accounts for the appearance, motion, and disappearance of multiple targets in the search space constituting the utility of creating a team of multiple search agents. When agent battery level drops below a predetermined threshold, the agent returns to a base station to recharge and be relaunched into the mission. Remaining agents must account for this loss and gain of other team members as they exit the search environment.

The contributions of this work are 1) the design of search trajectories for autonomous vehicles with limited endurance, 2) incorporation of return-to-base and recharge time requirements, and 3) coordination of multiple vehicles by developing a decision-making model to and assign agents to operational modes. Each of these components enable persistent multivehicle operations. Simulation results are intended for implementation on a system of quadrotors complemented by a system capable of autonomously recharging vehicles to sustain a multivehicle team beyond the mission life of a single vehicle.

Keywords: Persistent, Swarm, Detection and Tracking, Autonomous, Multivehicle Coordination

Acknowledgments

Special thanks to my co-advisers to Professor DeVries and Professor Kutzer for whose guidance and mentoring made this project possible. Thank you to the Office of Naval Research, the Harry E. Ward Fund, and the Class of 1979 for their continued financial support of the Trident Scholar Program. Finally, thank you to the Weapons and Systems Engineering Department for their support of this project and the Trident Scholar Committee for the opportunity to participate in such a prestigious program.

Contents

1 Introduction

One advantage robots possess over humans is their ability to work indefinitely without distraction, provided they have the necessary battery life, on tasks that are often described as dull, dirty, and dangerous. Due to this distinct advantage, research on autonomous vehicles and their applications has grown significantly over the past decade. Consequently, Unmanned Aerial Vehicles (UAVs) now have the capability to accomplish a wide range of critical tasks in both civil and military applications. Work published in [1–5], among many others, demonstrate the collective knowledge on UAV technologies that continues to amass rapidly since the turn of the century.

An emerging application of UAV technology is a team, or swarm, of autonomous vehicles cooperatively completing an assigned task for an extended period of time. In addition to research on UAV technology, several researchers [6–8] have extensively developed multitarget search and track theory and techniques with a proposed application of maritime search and rescue; search and track consists of detecting, localizing, and following targets in a defined search area. This paper builds upon the extensive foundation of research on UAV technology and target detection and tracking theory by accounting for limited endurance of micro UAVs which will enable persistent target detection and tracking by a scalable multivehicle swarm.

Additionally, the United States military shows strong interest in the development of UAV technology. The US military employs a diverse collection of unmanned air systems ranging from small scale (less than 5lbs), such as the RQ-11B Raven Surveillance drone (AeroVironment Inc, Monrovia, CA) [9] shown in Figure 1, to large scale (>2000lbs), in the case of the RQ-4 Global Hawk (Northrop Grumman Corporation, Falls Church, VA) [10] shown in Figure 2. These assets are typically deployed as a single entity rather than a coordinated group and are used for Intelligence, Surveillance, Reconnaissance (ISR) missions, such as search and rescue.

Figure 1: *RQ-11B Raven [9]*

Figure 2: *RQ-4 Global Hawk [10]*

While these systems are highly effective they eventually need to land to refuel; a group of small UAVs, or agents, may offer a cost effective solution to persistent ISR missions. The Department of Defense (DoD) has recently demonstrated a high level of interest in small UAV systems in a publicized demonstration of 103 small UAVs, developed at MIT's Lincoln Lab, from an F/A-18 Hornet in flight [11], pictured during deployment from a small white under wing pod in Figure 3. The demonstration included UAV launch and waypoint navigation, illustrating the need for robust control algorithms to direct tasking of multi-vehicle systems.

Figure 3: *Recent DoD swarm demonstration [11] where small UAVs are launched from the small white pod attached under wing on a F/A-18 Hornet, these UAVs move so fast and are so small they are difficult to capture in an image.*

However, small UAVs often have a short mission life due to small on-board electrical power reserves, thus diminishing their utility in accomplishing a sustained task. Unless affordable alternate forms of power are found, the quickest solution to increasing the endurance of autonomous agents is an integrated robotic systems approach to mechanically sustain multiple autonomous agents simultaneously working together. In short, creating a swarm of agents that can persistently complete an assigned task by returning

to base to recharge when necessary. Agents that are recharged can be relaunched to rejoin the swarm and contribute to accomplishing the mission. The swarm concept increases mission life, data acquisition capabilities (video, acoustics, spectral, etc.), and area of operation. Work published in [1, 3, 12] provide proof of concept on the benefits of swarms. Developing a method to sustain a team of small UAVs may increase the utility of UAV technology on persistent ISR missions. This paper provides a solution for persistent swarm operations.

The problem statement is as follows: derive an autonomous multivehicle control algorithm coordinating vehicle motion to persistent search for and track targets in a defined search area subject to operational endurance constraints of individual agents. The underlying goal simplifies to developing a modular control architecture that can be modified to any type of autonomous vehicle, search area, or emitter-detector pair. In this case, a small scale experimental system can serve as proof of concept for a variety larger systems that can be employed in operational environments. In practical application, a target can be anything from heat signatures to radioactive material; therefore, this project will simulate a generic emitter-detector (searchee, searcher) pair as a placeholder relationship for real world missions.

The remainder of this report discusses the development of a control structure of coordinated multivehicle operations. Starting at the high level control strategy and working through the design of search paths, trajectory tracking, operational mode assignment, and future work.

2 Algorithm Development

The central focus of this work is to achieve a reliable algorithm to direct multiple vehicles on a coordinated task for an extended period of time, i.e. longer than a single mission life (battery charge, fuel tank, etc.). Theoretically, the algorithm is independent of the hardware system intended for implementation, and algorithm development will focus on creating modular functions that can be swapped out for different functions. For example, if a user wished to simulate a different search agent than the one in this work, they could input another kinematic model to support the simulation and run their own tests. The concepts presented in this work are general by intention to allow for application to other systems. In order to integrate multiple autonomous agents into a single task there are several steps working in a continuous loop, including: decide on task assignment, assign trajectory based on task assignment, collect data, assess agent health (battery status), analyze collected data, repeat. In our case, a simple search and track problem, our unique contribution is the decision on task assignment which takes into account the health status of the agent and the overall work completed by the group in accomplishing a reliable search and track. Therefore, the main focus of work up to this point has been focused setting up the infrastructure to rapidly iterate our decision making process.

2.1 Objectives

The objectives of the algorithm are to (1) simulate multiple vehicles, (2) account for limited flight time of the vehicles, (3) monitor a region for the appearance targets, and (4) detect and track targets as they move through the monitored region.The algorithm must account for the appearance, motion, and disappearance of multiple targets in the search space constituting the utility of creating a team of multiple search agents. Each of these objectives must be modular in structure to allow for modification. Next, tracking can most aptly be described as hovering over a target once found and continuing to maintain that position even as the target moves over time. When the battery level of a vehicle drops below the predetermined threshold, the agent will return to charge. Remaining agents must account for the loss and gain of agents a they exit and re-enter the search environment. Each of these objectives support the creation of a persistent multivehicle swarm executing the search and track mission.

2.2 Assumptions

This work makes a number of assumptions in deriving, simulating, and implementing a coordinated multivehicle control strategy. However, these general assumptions do preclude the solutions presented in this work from being applied to other problems. First, the search area maintains a defined boundary; however, the search environment is uncertain in nature due to the random appearance, motion, and disappearance of targets over the course of the experiment. Second, future experimental validation will leverage the use of an existing hardware testbed, resulting in simulations grounded in realistic system parameters, properties, and dynamics. The quadrotor planned for use in experimental validation is the CrazyFlie 2.0 made by Bitcraze, shown in Figure 4. The CrazyFlie is a relatively inexpensive development platform ideal for implementation in these experiments.

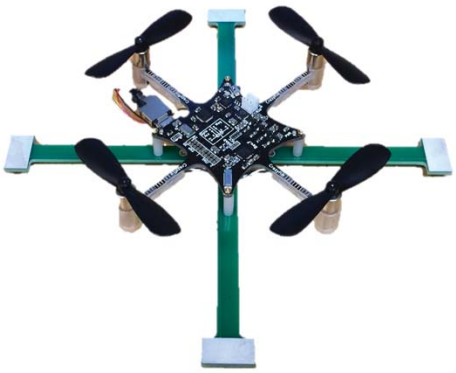

Figure 4: *A visual of the CrazyFlie 2.0 which informs simulation models [13].*

Third, the test bed, developed in [14–16], constrains the development of a multivehicle system to a centralized form of autonomy. Work done in [16] presents an autonomous recharging station that utilizes

a robotic arm housed on a home base station to retrieve and launch aerial vehicles. For the purposes of this work the home base will be a stationary and known location for the UAVs to launch from and return to for recharge. In this centralized autonomy construct computations are carried out at central base station in a motion capture lab due to the limited computational power of individual agents. In a motion capture lab we can track the position and orientation of every agent relative to a global frame and calculate relative positions to avoid collisions; however, the algorithm is designed to be independent of the autonomous agent employed.

In practice, if a "smarter" autonomous agent capable of decentralized multivehicle coordination is available for use the control restructured to push decision making computations onto the individual agents. These three factors aid in rapid system development as the researcher is able to collect position and orientation data on all components of the system to estimate the other states, conduct collision avoidance, and decrease debugging time.

2.3 Agent Dynamic Model

An important aspect of this work is that, while generalized, as many parameters as possible are grounded in hardware. The quadrotor discussed in Section 2.2 is an driving factor of algorithm development, and it's dynamic model is further discussed in this section.

Quadrotor motion is defined by the equations of motion presented in Equations (1) - (6), which are cited from [1,2,17]. The equations of motion define the X, Y, Z, Roll ϕ, pitch θ, and yaw ψ accelerations in the body frame for the assumed rigid body. Figure 5, as presented in [17], depicts the quadrotor rigid body and the associated translational and rotational body frames. Note that x_B, y_B, z_B define the inertial coordinate frame for the body and x_W, y_W, z_W define the world coordinate frame. According to these translational coordinate frame definitions the rotational components: roll, ϕ, occurs about the x_B axis; pitch, θ, occurs about the y_B axis; and yaw, ψ, occurs about the z_B axis.

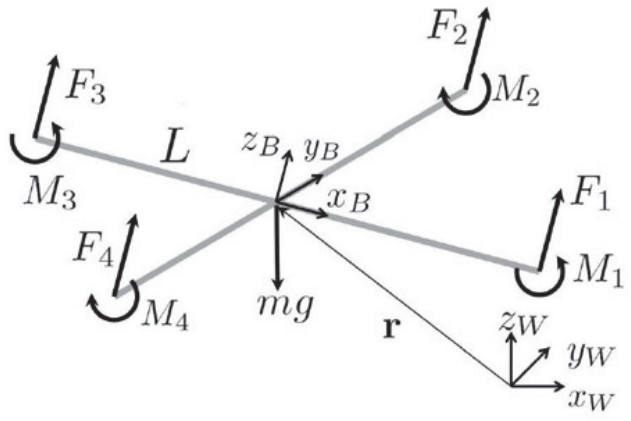

Figure 5: *A visualization of the quadrotor rigid body frames as presented in [17].*

Additionally, $F1$ through $F4$ and $M1$ through $M4$ are the individual forces and moments produced by each of the motors. The equation structure presented in Equations (1) - (6) support the definitions from Figure 5 where X is forward, $Y$is lateral, and Z is vertical in the body frame, and the rotations are defined as ϕ about the x-axis, θ about the y-axis, and ψ about the z-axis. Further, the following variables are defined as: I_{xx}, I_{yy}, I_{zz} are the vehicle's moment of inertia; m is the vehicle's mass; g is gravitational acceleration; U_1, U_2, U_3, U_4 are the input commands resulting from the trajectory controller calculation.

$$\ddot{x} = -\frac{U_1}{m}\sin\theta \quad \text{(X Acceleration)} \tag{1}$$

$$\ddot{y} = \frac{U_1}{m}\sin\phi\cos\theta \quad \text{(Y Acceleration)} \tag{2}$$

$$\ddot{z} = \frac{U_1}{m}(\cos\phi\cos\theta) - g \quad \text{(Z Acceleration)} \tag{3}$$

$$\ddot{\phi} = \frac{1}{I_{xx}}U_2 \quad \text{(Roll Acceleration)} \tag{4}$$

$$\ddot{\theta} = \frac{1}{I_{yy}}U_3 \quad \text{(Pitch Acceleration)} \tag{5}$$

$$\ddot{\psi} = \frac{1}{I_{zz}}U_4 \quad \text{(Yaw Acceleration)} \tag{6}$$

These equations of motion integrate with controllers structure from works such as [17] or [18] to relate global states from the desired trajectory to input commands in the inertial body frame that will direct the vehicle on the planned path. The basic structure is to calculate the error from the desired path and return control inputs to achieve that desired path. The trajectory controller is later discussed in Section 3.5.

2.4 Control Strategy

The control strategy for extended multi-agent coordination is a multi-modal control algorithm that runs a high level decision making loop to assign agents to different operational modes to accomplish the task. Figure 6 offers a visual representation of the control algorithm modes and their corresponding triggering events. Mode 1: Return encompasses the agent returning to the base station to recharge, which is triggered by the agent's battery level dropping below the designated battery threshold.

Mode 2: Recharge covers the recharging period where the agent is offline and no navigational computation is needed; the mode is triggered by a docking event in a recharge station following a retrieve delay. Mode 2: Recharge ends when the vehicles are fully charged again and are marked available for task assignment by the swarm, where they are relaunched into the swarm following a launch delay. The launch and retrieve delays are motivated by the autonomous recharging system presented in [16], which was discussed in Section 2.2.

Modes 3 and 4 are inextricably linked as the search and track task. Mode 3: Search constitutes the search trajectory assignment of the agent, which is the default mode when the battery level is above the designated battery threshold. Finally, Mode 4: Track comprises the target tracking portion of the mission that is triggered by a detection event. While the search mode starts as the default operating mode, tracking takes priority in the higher level decision making algorithm. As the goal of the task is to find and track targets. The algorithm also accounts for relief events where agents in tracking mode that must return to the base station for charging are relieved by an agent with more battery life to continue tracking the target.

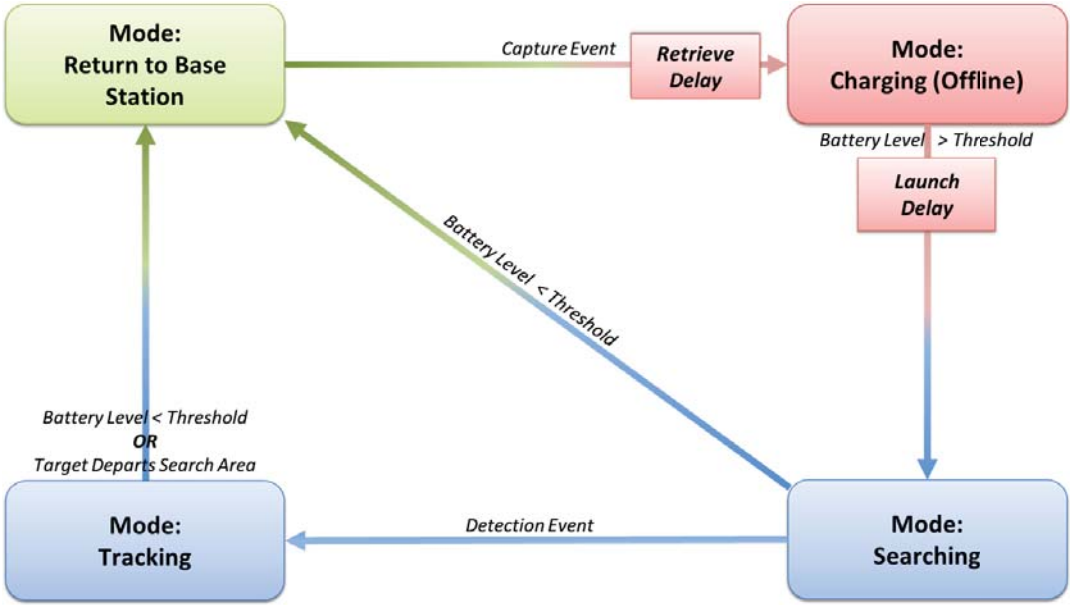

Figure 6: *A visual representation of the control algorithm modes and their corresponding triggering events; labeled arrows represent transitions between operational modes and the associated triggering events.*

2.5 Target Detection

Target detection is accomplished using the methods detailed in [19]. This section provides an overview sensor measurement procedures to declare detection events. In this work targets enter the field of view sensor measurements spike above noise thresholds, a detection event occurs, and the agent is switched into tracking mode. The agent follows the target until it runs out of battery and needs to recharge. The generic sensor cones can be seen in Figure 14.

This work utilizes a generic conical sensor field to maintain the generality of the simulation, as a conical sensor projection can aptly apply to realistic devices such as cameras, thermometers, infrared sensors, or radiation detectors. The conical sensor field of view (FOV) is projected onto the ground plane of the search space. The sensor foot print is projected onto the ground plane as a circle if the quadrotor is level or an ellipse if the quadrotor pitches or rolls. The projection step ensures the sensor model creates a realistic field of view for any agent orientation. For the purpose of this work, we assume that sensors are operating optimally and do not collect faulty data, false positives, because the agents search at a constant velocity discussed in Section 3.

Mode 3 and 4 will leverage likelihood ratio tracking techniques for the detection and tracking components of the control algorithm. Work published in [20], [8], and [19] detail the use of likelihood ratio tracking (LRT) as an extension of recursive Bayesian estimation. A LRT is an excellent method to "estimate the position of possibly multiple targets" [19]. This detection technique associates sensor measurements with a probability of target presence, and creates a ratio of likelihood of target presence to likelihood of target absence. This technique is cited in conjunction from [19–21].

The estimation proceeds mathematically as follows in several steps: predict the likelihood ratio of target presence given previous measurements, update likelihood of target presence with the “new” measurement, calculate the “new” likelihood ratio of target presence, repeat. "A likelihood ratio tracker replaces the measurement likelihood with the measurement likelihood ratio, i.e. the ratio of two likelihood functions [19]" Let $r_t = (x_t, y_t) \in R^2$ represent the position of the target and z_k represent the sensor measurement from a search agent at time t. After initializing the set, the prediction step begins with the likelihood function of target presence given the previous measurement:

$$\Pr(r_t \mid z_{t-1}) = \int_{R^2} \Pr(r_t \mid r_{t-1}) \Pr(r_{t-1} \mid z_{t-1}) dr_{t-1} \tag{7}$$

Where $\Pr(r_t \mid z_{t-1})$ in Equation (7) is the conditional probability of the target location r_t given the previous sensor measurement one time step back at $t-1$. Upon receiving a new measurement the likelihood of target presence is updated according to the product the measurement likelihood $\Pr(z_t|r_t)$ and the previous prediction given by Equation (8):

$$\Pr(r_t \mid z_t) = \frac{\Pr(z_t \mid r_t) \Pr(r_t \mid z_{t-1})}{\Pr(z_t \mid z_{t-1})} \tag{8}$$

In the update step, the denominator is the integral of the numerator given by the integral

$$\Pr(z_t \mid z_{t-1}) = \int_{R^2} \Pr(z_t \mid r_t) \Pr(r_t \mid z_{t-1}) dr_t$$

Integration of the product of measurement given target presence and target presence given the previous measurement accounts for the previously calculated likelihood forming a recursive predictor of target presence likelihood. Finally, the likelihood ratio is calculated to make a determination on a detection event. The numerator of the the likelihood ratio represents the conditional probability of the measurement given that the target is present (r_k^+), whereas the denominator represents the conditional probability of the measurement given that the target is not present (r_k^-). The likelihood ratio is given by the equation" [19]:

$$L = \frac{\Pr(z_k \mid r_k^+)}{\Pr(z_k \mid r_k^-)}$$

Subsequently, detection events are defined as instances where the likelihood ratio rises above a designated threshold. Thresholds are identified through sensor modeling and assigning a cost to the accuracy of the detector.

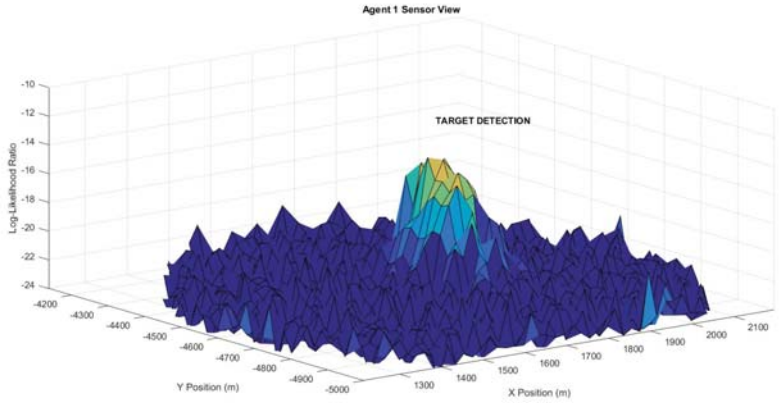

Figure 7: *A visualization of a detection event, where the log likelihood ratio has spiked well above the sensor noise.*

2.5.1 Target Motion Model

The intention of the target motion model is to create targets that randomly walk about the search space. The model includes constraints to given a the targets a realistic motion pattern, as they are constrained to reasonable heading shifts and speed changes; therefore they can not instantaneously reverse their speed or heading. At the start of an experiment the targets are randomly placed throughout the search boundary; at user defined time intervals through the experiment the targets randomly reset their relative heading and speed on a predefined interval to move about the space. Figure 8 shows the two dimensional target motion

model where S_T is the target speed and ϕ_T is the target heading.

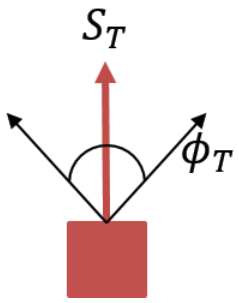

Figure 8: *A visualization of the target model and assigned states, speed S_T and heading ϕ_T.*

Equations (9) through (11) describe the target motion and predefined reset intervals for speed and heading, where $\dot{\vec{x_T}}$ is the target vector and $\vec{v_T}$ is the temporary randomized vector with components $v_{T,xy}$,

$$\dot{\vec{x_T}} = \vec{v_T}, \tag{9}$$

$$v_{T,x} = S_T cos(\phi_T), \tag{10}$$

$$v_{T,y} = S_T sin(\phi_T). \tag{11}$$

In this work, speed changes are defined as $S_T = Random \subset [0.1, 1.2](m/s)$ and heading shifts are defined as $\phi_T = Random \subset [\frac{-\pi}{4}, \frac{\pi}{4}](rad)$. Functionally, the target speed remains between 0.1 and $1.2m/s$ and the target heading can not change greater than $\frac{\pi}{4}rad$ from the current heading. These constraints give the targets a realistic walking trajectory through the space. Similar to other components of this work, the target model can be modified to conform to additional experimental requirements.

3 Search Trajectory Development

Accomplishing the search task first required the development desired trajectory path for the search agents to follow. For the purposes of the described problem we implemented a lawn mower search pattern which directs the search agents to sweep back and forth through the entire search space. While a large body of research exists on optimal search trajectories for multiple agents using many different mathematical strategies from dynamic space reconfiguration to probabilistic path determination, such as in [7, 8, 22] the focus of this work is persistent search. Therefore, any search algorithm can be used without loss of generality. We chose a simpler lawn mower path for ease of implementation; however, the modular concept of this approach allows for new functions to be added. Additionally, this approach can be implemented on any size or shape search space because the trajectory is created from user define waypoints. In this case, the space is defined as a 10m x 10m rectangle. The desired trajectory path defines

a position, velocity, acceleration, and yaw angle in the global reference frame for each agent in every time step over the entire time vector. The state vector for the desired trajectory, r_T, is a 3x1 vector containing $X, Y,$ and Z, in the form $r'_T = [X, Y, Z]$, and desired yaw, ψ_T, is a scalar value. The development of the search trajectory accounts for realistic limitations of the dynamics of our search agents by assigning velocities, accelerations, and yaw rates that create a smooth trajectory.

Trajectory generation is divided into the following steps: (1) define waypoints in the search space (*s* space), (2) calculate piecewise polynomials that continuously connect the waypoints in *s* space, (3) derive the expression to evaluate *s* over time *t*, and (4) numerically evaluate the trajectory and assemble the desired state vectors. These steps effectively create a set of piecewise polynomials that define the three dimensional position over time, $X(s(t))$, $Y(s(t))$, and $Z(s(t)) = h^*$ where h^* is the desired constant altitude. These sets of polynomials can then be differentiated to yield desired velocity and acceleration. Finally, each of the components assemble into the desired trajectory path defined by the three-dimensional position r_T, velocity $\dot{r_T}$, acceleration $\ddot{r_T}$, and scalar yaw angle ψ_T

3.1 Define Waypoints

User specified waypoints outline the desired path for the lawn mower trajectory. The waypoints designate an alternating series of straight aways and turn arounds. These waypoints are offset from each other by half of the sensor field of view radius resulting in a path that does not overlap sensor coverage, assuming the sensor is mounted in the center of the vehicle. Similarly, the search space is offset by the simulated sensor radius to prevent search outside the search space for efficiency. Additionally, the turn around radius is equal to the simulated sensor radius.

Waypoints consist of an $[X, Y, Z]$ position and are in numbered order in *s* space. The *s* space domain sequentially numbers the points in order in which the agent should reach them. Such that $X(s)$ where $s = 1$ is the first X waypoint in the desired path. Figure 9 visualizes the search space with these points and offsets. The green line defines the straight away boundaries, and the dashed red line shows the edge of the turn around. Lastly, the numbers near each way point defines the waypoint in *s* space.

Once these desired points are defined the next step is to calculate the piecewise polynomials that continuously connect them.

3.2 Calculate Piecewise Polynomials

Piecewise polynomials that connect the waypoints in *s* space are an alternating series of straight aways and turn arounds, which form the lawn mower patter shown later in Figure 10. Each straight away is simply a straight line that travels between two waypoints. Whereas, the turn around is a continuously differentiable path that travels through three waypoints. Therefore, the turn around equations are constrained by six conditions, the (1) start, (2) middle, and (3) end points of the turn, in addition to (4) incoming,

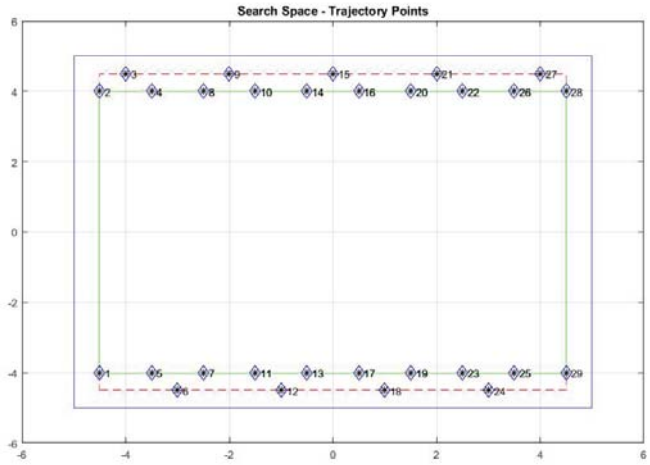

Figure 9: *A visualization of an example search space with sensor offsets and numbered waypoints.*

(5) tangential, and (6) outgoing velocity vectors. These three points and three first derivatives define a continuously differentiable trajectory.

The 6th order polynomials for $X(s)$ and $Y(s)$ that create the position component of the turn around trajectory in the *s* domain are Equations (12) ,(13), and (14).

$$X(s) = c_{x,1}s^5 + c_{x,2}s^4 + c_{x,3}s^3 + c_{x,4}s^2 + c_{x,5}s + c_{x,6} \tag{12}$$

$$Y(s) = c_{y,1}s^5 + c_{y,2}s^4 + c_{y,3}s^3 + c_{y,4}s^2 + c_{y,5}s + c_{y,6} \tag{13}$$

$$Z(s) = h^* \tag{14}$$

Where $X(s)$ and $Y(s)$ are calculated using the method described in [23], and $c_{xy,1}$ through $c_{xy,6}$ are numerical constants that precipitate from this calculation and dependent on the waypoints used to define the current piece of the pattern. Note that $Z(s)$ is set to a constant height (h^*), as the trajectory controller accomplishes the launch phase, further discussed in Section 3.5. Setting the desired height allows for altitude collision avoidance by placing agents in different modes at different heights to minimize the chances of collision.

3.3 Coordinate Temporal Execution of Polynomial Trajectory

The next step is to derive *s* in terms of *t*, time, to create $s(t)$ that will be used to evolve the desired trajectory path over time rather than numbered waypoint. In order to calculate this relationship the first step is to numerically calculate the Euclidean norm T_s composed of the desired velocities at the current *s* value. The Euclidean norm is defined as: $T_s = \sqrt{X'(s)^2 + Y'(s)^2}$. Next, this value for T_s relates to the first derivative in *s* space, $\dot{s}(t)$, via an inversely proportional relationship with the constant desired velocity

v^*. This relationship is defined as: $\dot{s}(t) = \dfrac{v^*}{T_s}$. This method allows for the user to define a constant velocity because the desired velocities are parametrized over arc length. Finally, $\dot{s}(t)$ is numerically integrated to find $s(t)$ or numerically differentiated to find $\ddot{s}(t)$. Evolution of the search trajectory over time allows for a the search agents to travel at a constant velocity, which enables the assumption that the sensor is collecting reliable data as it covers the space. With the equations to define the three dimensional trajectory in *s* space and the numerical approximation for $s(t), \dot{s}(t)$, and $\ddot{s}(t)$ the next step is numerical evaluation to assemble the desired state vectors for position, velocity, acceleration, and the scalar yaw value.

3.4 Trajectory Implementation

Finally, the trajectory is implemented through the numerical evaluation of the piecewise polynomials that map out the continuous lawn mower pattern yield the three element vectors for position r_T, velocity $\dot{r_T}$, acceleration $\ddot{r_T}$, and yaw ψ_T. Where $s(t)$ is the numerical approximation of *s* in terms of *t* that can be used to evaluate $X(s(t))$ and $Y(s(t))$; the final step prior to assembling the entire trajectory is employ the chain rule to define the velocity and acceleration state vectors. Equations (15) - (18) define the desired trajectory vectors in terms of $s(t)$

$$r_T = \begin{bmatrix} X(s(t)) \\ Y(s(t)) \\ h^* \end{bmatrix} \tag{15}$$

$$\dot{r_T} = \begin{bmatrix} X'(s(t))\dot{s}(t) \\ Y'(s(t))\dot{s}(t) \\ 0 \end{bmatrix} \tag{16}$$

$$\ddot{r_T} = \begin{bmatrix} X''(s(t))\dot{s}(t)^2 + X'(s(t))\ddot{s}(t) \\ Y''(s(t))\dot{s}(t)^2 + Y'(s(t))\ddot{s}(t) \\ 0 \end{bmatrix} \tag{17}$$

$$\psi_T = tan^{-1}(\frac{\dot{r_{TY}}}{\dot{r_{TX}}}) \tag{18}$$

Figure 10 shows an example of a lawn mower pattern in a rectangular search space using Equations (15) - (18). These equations can be adapted the create a lawn mower search patter in a search space of any size or shape, which further increases the utility of a lawn mower search pattern.

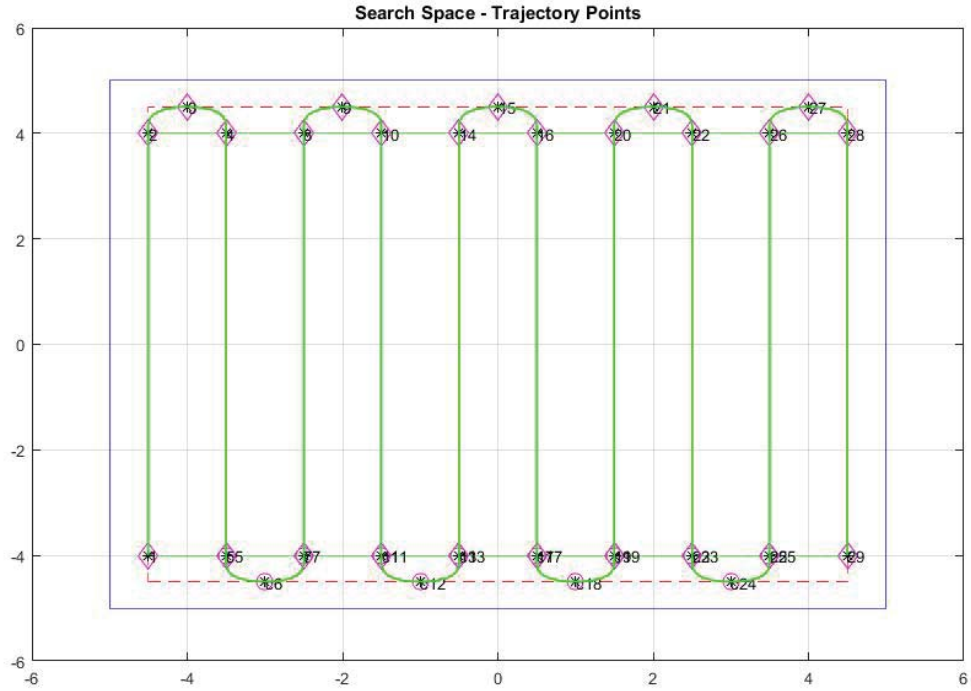

Figure 10: *A visualization of the piecewise polynomials that connect waypoints to define the lawn mower trajectory.*

3.5 Trajectory Tracking

Trajectory tracking is a well studied topic highly dependent on the dynamic properties hardware system employed. An important point of focus in our work was to create a well documented and modular control algorithm structure to allow another user to implement their own hardware system dynamics to utilize this control method, essentially a "plug and play" concept. Section 3.1 established the creation of the desired trajectories. This section will discuss the design and implementation of a controller such that the vehicle achieves the desired trajectory.

The quadrotor trajectory controller presented in [17] provides a reliable method for driving quadrotors on desired paths. The controller allows for assignment of desired position, velocity, acceleration, and yaw angle which are translated into control inputs for the quadrotor equations of motion. Control inputs into the equation of motion effect the quadrotor states; recall from section 2.3 that the quadrotor states include translational and rotational elements in the inertial frame. Equation (19) gives the control proportional derivative (PD) control law implemented,

$$\ddot{r}^{des} = k_p e_p + k_d e_v + \ddot{r_T}. \tag{19}$$

In Equation (19), the k terms, k_p and k_d, are control gains; the e terms, e_p and e_v, are state errors from the desired state; and $\ddot{rT}$ is the feed forward acceleration. The is the position state error, e_p, and velocity state error, e_v, are given by the following equations:

$$e_{position} = r_{x,y,z} - r_{T_{x,y,z}}, \tag{20}$$

$$e_{velocity} = \dot{r}_{x,y,z} - \dot{r_{T_{x,y,z}}}. \tag{21}$$

As noted in Subsection 3.1, our lawn mower path clearly defines position, velocity, acceleration, and yaw for each agent over time. The controller reliably tracks the desired search trajectory to include centrally located "base station" launch and return paths. In addition, the controller is robust enough to track trajectories not defined in all of these states. In fact, we rely on this robust control for the launch and return phases of the trajectory, where only position is defined for the agent. The controller takes over and brings the agent to the desired position using an acceptable response in acceleration and velocity. Close inspection of the launch and return phases of Figure 15 shows the the agent tracks the desired trajectory.

Figure 15 shows 4 search agents that have completed their full search trajectory and returned to the starting point of their search. The dashed green line is the desired path and the colored line is the path taken by each agent. By inspection, each agent reliably follows its desired path through the whole space.

4 Extending Swarm Operations

In order to extend the operations of the multivehicle team beyond the battery life of a single vehicle the control strategy must incorporate return to base and recharging requirements. Recall from Section 2.2, that the hardware developed in [14–16], supports autonomous recharging from a central base station. This allows agents to return to a known location when their battery level runs low to be recharged and eventually relaunched into the swarm. The following Sections 4.1 and 4.2 discuss the execution of both Mode 1: Return to Base and Mode 2: Recharge.

4.1 Return to Base

Despite the simple point to point trajectory, the return mode is vital to sustaining the swarm. As the shortest path between two points is a straight line, the return mode calculates and assigns a straight line trajectory from the starting location to the known home base location. Once set to this mode, after the agent's battery drops below the return threshold, the algorithm takes the current X, Y, Z location of the agent and the known home base point and calculates a straight line path between the two. Calculating the return path after being assigned to the return mode allows for the agent to be set to return mode at any point in the search boundary at any time. However, an immediate shift from a smooth search trajectory

or slow target tracking can result in an agent to become unstable and "crash"; which results in the need to filter these new control inputs to the trajectory controller to avoid causing the quadrotor to become unstable. The filter prevents any quadrotor state, position, velocity, acceleration, and yaw angle, greater than or less than 20% of the current state to be sent to the trajectory controller as the desired input; this filter has proved to be effective in shifting to the return mode.

Equations (22), (23), (24), and (25) describe generation of a return path similar to the method use in creating search trajectories from Section 3.2. Equation (22) defines vector, $m_{x,y}$, of the return path in X and Y where $H_{x,y}$ is the known home base location and $b_{x,y}$ is the current location of the agent,

$$m_{x,y} = H_{x,y} - b_{x,y}, \tag{22}$$

$$X(s) = m_x s + b_x, \tag{23}$$

$$Y(s) = m_y s + b_y, \tag{24}$$

$$Z(s) = h^*. \tag{25}$$

Consider $b_{x,y}$ as the start point and $H_{x,y}$ as the end point of the return path. Equations (23) and (24) break out the X and Y components of the trajectory for evaluation using the same method described in Sections 3.3,3.4, and 3.5. Altitude, Z, remains constant as shown in Equation (25). Figure 11 depicts an agent operating in return mode from a simulation.

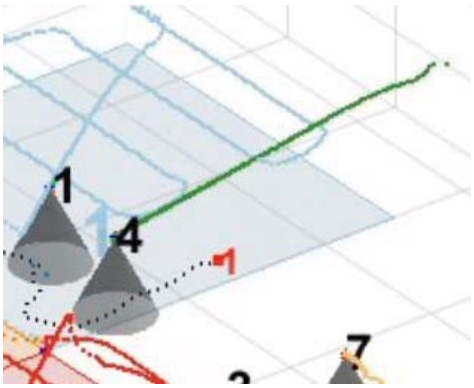

Figure 11: *Agent 4 executing the return mode in simulation.*

4.2 Recharging

Recharging is a relatively simple operational mode as the agent is marked offline and charged until it has a full battery. However, battery models are nonlinear and unique to the individual battery. Factors such as age of the battery, voltage draw by the agent, number of discharges, number of recharges, and temperature can have a significant effect on how the battery discharges. Therefore, we implemented a timed battery model based on the nominal flight time of the CrazyFlie2.0 provided by [13], which

is specified at 7 minutes by the manufacturer. A timer battery model is linear and much simpler to implement in simulation. Further, the modular structure of the control algorithm allows for modification of the battery model as desired.

4.2.1 Battery Model

For the purposes of this work, we use a simple battery model based on time; the battery level is represented as a percentage that decreased linearly over time. Since most batteries can not operate near 0%, the battery model is designed to reach 25% batter capacity at the 7 minute mark; this 25% threshold is a conservative estimate for the operable battery capacity of the CrazyFlie 2.0, mentioned in Section 2.2, that informs this simulation. However, the recharge is designed to charge the battery from 0% to 100% in 15 minutes, approximately double the discharge time. A full agent cycle is defined as seven minutes of flight time followed by fifteen minutes of recharge time, an approximately 1:2 flight to recharge ratio. While a 1:2 flight to recharge ratio is slightly unrealistic for a small UAV platform, this algorithm simulates general coordinated multivehicle operations and the battery model can be modified. We can manipulate the discharge and recharge time by modifying κ and β, which are the respective rates of change for discharge and recharge. Given in the Equation (26),

$$B(t) = \begin{cases} B_0 - \kappa t & B \not< 0\% \quad \text{Discharge Mode} \\ B_0 - \beta t & B \not> 100\% \quad \text{Recharge Mode,} \end{cases} \tag{26}$$

where B_0 is the battery state at the time of transition; $\kappa > \beta$ such that the battery is discharged faster than it is recharged. Again, the manufacturer specifications inform the battery model, but a more robust model can be implemented in the future. Figure 12 shows the rates of discharge and recharge.

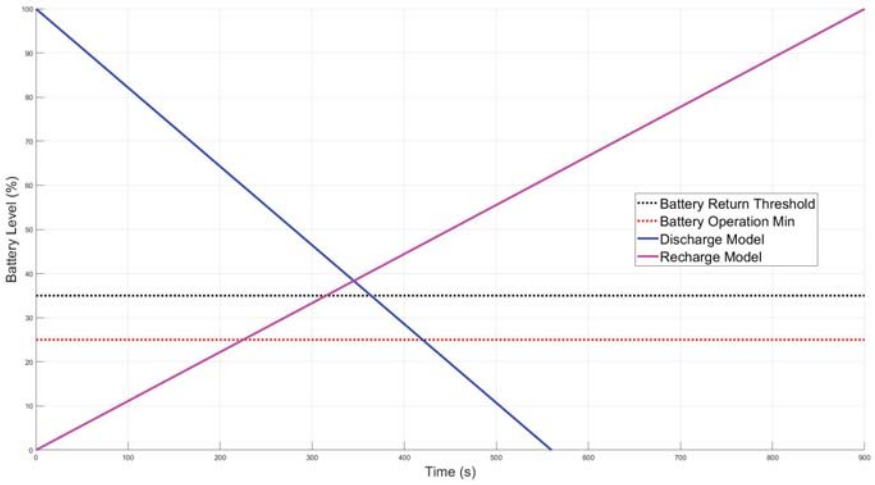

Figure 12: *Visualization of the linear battery model associating a battery level percentage with time according to user specified rates.*

5 Multivehicle Coordination

The main goal of this project is coordination of multiple vehicles to complete a search and track task. We attack this complex problem by creating states for the agents that describe their role in the swarm to include: a label number, assigned search cell, a mode number, and a mode operation time. Each of these states inform the algorithm for the efficient use of processing power and mode shifting.

Figure 13 highlights the Status Board that shows the user the mode, mode description, mode start time, mode running time, and battery level for every agent available to the swarm at every instance in time. This user readable output is extremely helpful for development and debugging. Each of these data pieces are used in the larger simulation to coordinate the actions of all agents. The simulation must track the operational mode of every agent for all time in order to properly assign and execute modes of operation.

	Mode	Description	Launch Time	Mode Time (s)	Batt Level (%)	isFly
1	1	Up, Returning	340.5000	11.9000	37.0614	☑
2	2	Down, Recharging	264	88.4000	65.3667	☐
3	2	Down, Recharging	191.8000	160.6000	88.9502	☐
4	7	In Launch Queue	344.9000	7.5000	100	☐
5	2	Down, Recharging	337.7000	14.7000	52.0961	☐
6	6	In Retrieve Queue	329.1000	23.3000	55.7608	☐
7	3	Up, Searching	137	215.4000	61.5296	☑
8	4	Up, Tracking	341.1000	11.3000	64.5300	☑
9	4	Up, Tracking	296.7000	55.7000	79.4431	☑
10	3	Up, Searching	311.9000	40.5000	92.7667	☑
11	3	Up, Searching	344.8000	7.6000	98.6426	☑
12	5	Down, Available to Launch	0	352.4000	100	☐

	Flying	Searching	Tracking	Returning	StandingBy	Charging	Retrieve Q.	Launch Q.
1	6	3	2	1	1	3	1	1

Figure 13: *The status board provides a readable summary of the swarm at any moment in time.*

5.1 Decision Making

An obstacle to creating a persistent multivehicle swarm is prioritizing and deciding on the task assignment for each agent. The general priority of assignment is to ensure the agent has enough battery to continue, track detected targets, and assign all search cells to agents. This objective is achieved using heuristic decision making model executed by coding logic.

By tracking many states of each search agent, such as its operating mode and battery level, a series of logic statements allow each agent to jump to calculations relevant to its own mode. For example, any agent in Mode 4: Track will not be set to any other mode unless a) the target leaves the search boundary or b) the agent runs out of battery; this maximizes the time tracking targets. Next, search cells are tracked and assigned to a single search agent, since they are created for the intention of a single agent to cover the entire area effectively. These rules result in a flow of agents from mode to mode and allow the entire swarm to accomplish the search and track mission well beyond the battery life of a single agent.

5.1.1 Simulation Architecture

The pseudo code shown in Algorithm 1 outlines the high level infrastructure in place to achieve a multivehicle simulation. This project has yielded an extensive code base that can be shared with other researchers for continuing development. The code architecture supports the priorities of the decision making model discussed in Section 5.1.

Algorithm 1: Swarm Simulation Time Loop

Input : Length of experiment, Number of search agents, Number of targets
Output: Graphics rendering of swarm executing persistent search and track mission

forall *Time* ($\Delta t = 0.1s$) **do**
  foreach *Agent* **do**
    Check battery levels
    if *Return Threshold* $>$ *Battery* $>$ *Minimum Threshold* **then**
      Set agent to Mode 1: Return
    else if *Battery* $<$ *Minimum Threshold* **then**
      Agent dead $\rightarrow$ marked offline
    end
  end
  foreach *Target* **do**
    Update target states
  end
  if *Agents Employed* < *Number of Agents* **then**
    if *Search cells unassigned* **then**
      Replan to cover all cells
    end
  end
  foreach *Agent* **do**
    if *Agent Flying* **then**
      Check Mode $\rightarrow$ Trajectory tracking calculations to support mode
      Calculate sensor readings
      if *Sensor Measurement* > *Detection Threshold* **then**
        Mark Detection
        Set target tracked
        Set agent to Mode 4: Tracking
      end
    else if *Agent Grounded* **then**
      Charge battery **if** *Battery* > *Charge Threshold* **then**
        Mark available for task assignment
      end
    end
  end
end

Within the simulation exist several deconfliction strategies: No agent tracking a target will be set to another mode other than return and sensor readings incorporate target tracking deconfliction to prevent double detections. As the goal of the simulation is to detect and track targets, all modes support the accomplishment of this task. However, the limiting factor is always battery life, which requires efficient task assignment and coordination of all vehicles.

5.2 Spatiotemporal Coverage

Tracking the swarm's sensor coverage of the search area in by both space and time allow for in depth analysis of the performance of the swarm. Ideally, the swarm would be able to cover all locations in the search area for all time, or 100% coverage, to maximize the detection of targets. We decided to simulate a generic sensor with a conical field of view to allow for project development to proceed independent of a sensor selection and to motivate the use of this control strategy for larger scale operations. A user could implement a wide range of sensors from cameras, IR sensors, temperature sensors, etc.; therefore, this generic simulation supports future scalability.

Spatiotemporal search is important to a persistent search and track task because it informs the search agents where they have been and how long it has been since they last visited a specific area. In this case, we assign a maximum coverage value to a place that has just been covered by the sensor field of view. This coverage value currently decays linearly as long as the sensor does not cover it again. Thereby simplifying the search task as we must maximize the coverage value across the entire search area to maximize the likelihood of detecting targets. According to Equation 27, the coverage metric is set high when a section of the search space is covered by a sensor field of view (FOV) and decays linearly according to γ when not covered by a sensor FOV. The coverage value decays linearly over time given by the equation,

$$C(t) = \begin{cases} C_{max} - \gamma t & C \not< 0 \quad \text{Out of FOV} \\ C_{max} & \text{In FOV,} \end{cases} \tag{27}$$

where γ is a user defined constant rate that speeds or slows the decay accordingly.

Figure 14 depicts the simulation environment with 4 search agents in uniformly partitioned search area completing the search task. The color map indicates the coverage value, with red high and blue low. Each agent follows a desired trajectory, dashed green line; drops its own trajectory, colored line; and has a simulated sensor field of view, translucent gray cone. Lastly, note that each agent is at a different point in the search pattern, they have been timed to launch at a user defined delay. This capability models the realistic time delay in launching multiple aircraft, but can also aid in maximizing search area coverage.

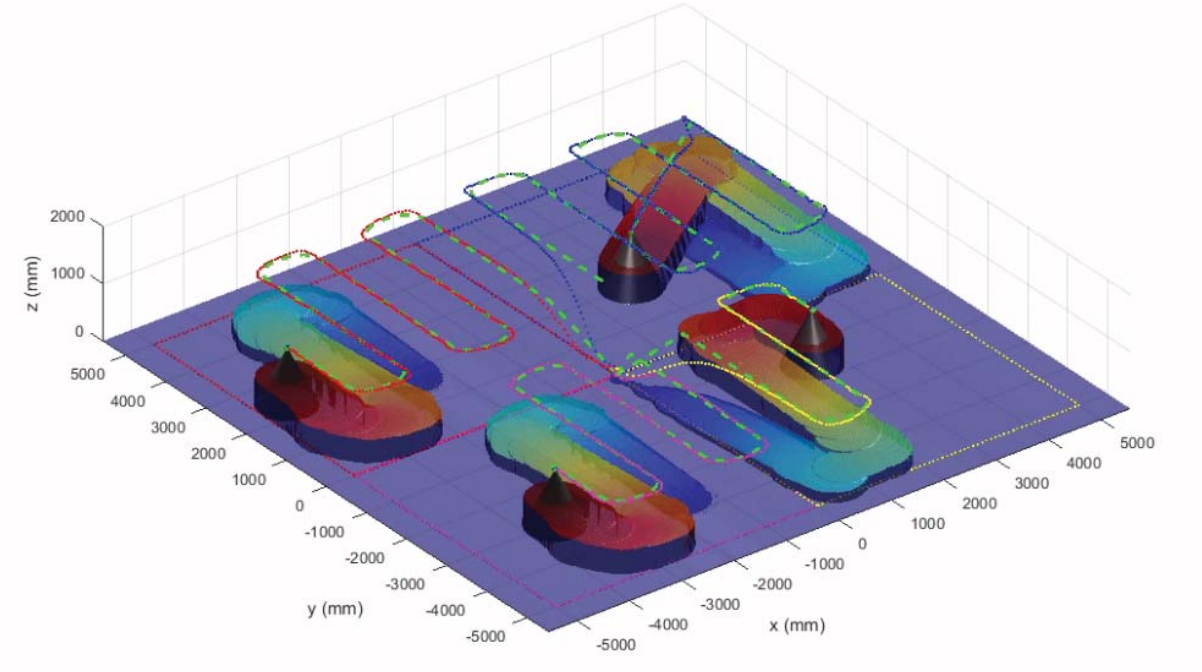

Figure 14: *A visualization of 4 simulated agents completing the search task. Red = Maximum Coverage Value, Blue = Minimum Coverage Value*

5.3 Multivehicle Planning

Each of the previous sections discussed the creation of a trajectory for a single agent. In order to run a coordinated multivehicle simulation we wrap planned search paths, target tracking, and battery information into a reliable control algorithm that directs our multivehicle team on an extended search and track mission. First, we create a uniform partition of the search space, in this case four equal quadrants. Next, each quadrant gets its own set of piecewise polynomials that define the trajectory through the space. However, each of these quadrants has the same waypoint numbering scheme as the original fist quadrant, meaning they all start in different locations in reference to the entire search space. In order to simulate a shared base station we must rotate and transform each set of equations to begin at the origin of the entire search space. The step of creating and transforming the trajectory equations occurs prior to the time iterated simulation loop.

The transformation is multiple steps to translate the start point to the origin, maintain the proper scaling, and restore any offsets; essentially mirroring the initial trajectory into each each search area. The transformation, $H(\vec{q})$ takes the form: $H(\vec{q}) = T_z(q_3)T_y(q_2)T_x(q_1)S_z(q_3)S_y(q_2)S_x(q_1)T_z(q_3)T_y(q_2)T_x(q_1)$, where $T_{x,y,z} \in \mathrm{SE}(3)$ are transformations in the specified directions given by

$$T_x(a) = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (28) \qquad T_y(b) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (29) \qquad T_z(c) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (30)$$

where a, b, and c are the magnitude of translation along the specified axis; and $S_{x,y,z} \in \mathrm{Aff}(3)$ are scales about the specified axis given by

$$S_x(d) = \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (31) \qquad S_y(e) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & e & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (32) \qquad S_z(f) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & f & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (33)$$

where d, e, and f are the magnitude of scaling in the specified axis. Figure 15 visualizes the result of transforming a pattern in the first quadrant to quadrants two, three, and four. Note each quadrotor launches and returns to a common base station, but searches a specified quadrant. In Figure 15 the dotted green line is the desired trajectory and the colored lines are the trajectories traveled by each agent. By inspection, combination of well defined paths a robust controller allow for the agents to reliably follow their defined path.

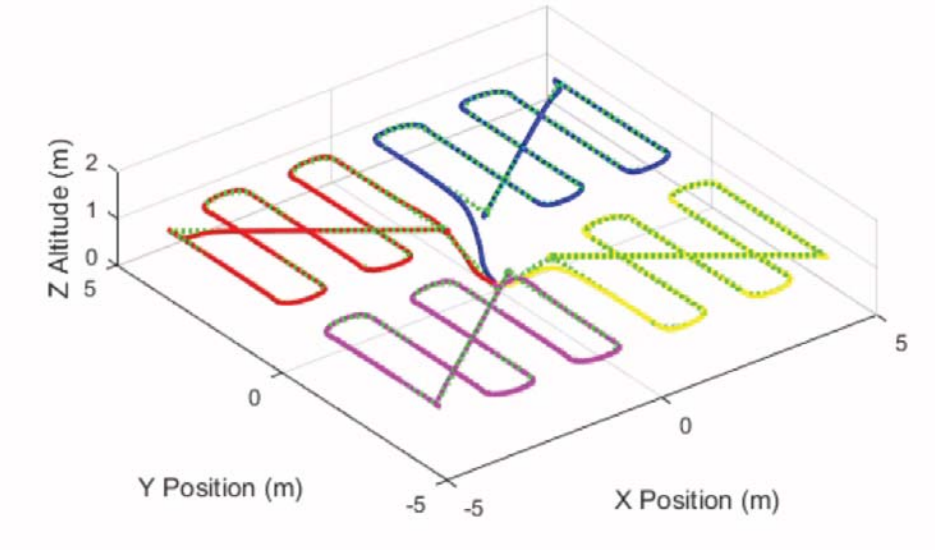

Figure 15: *A visualization of 4 search paths translated and scaled to cover the entire search space.*

5.4 Swarm Replanning

When agents change modes, drop in, or drop out of the swarm the other agents must be able to account for this loss or gain. This is accomplished via a replanning step. In our algorithm, any time an agent vacates a search cell replan is required; the driving force behind the replan is the assurance that all search cells are assigned to a search agent to maximize the area coverage by the swarm. For example, if an agent in Mode 3: Search detects a target and switches to Mode 4: Tracking a replan occurs to assign another available agent to the cell it just vacated.

The frequency of replans can be a limiting factor, if the swarm is attempting to replan multiple times a second then no agent is able to accomplish a task. For this reason, careful assignment of the replanning rules is required to ensure the swarm functions correctly. Consider an instance where all agents in the swarm are assigned to a mode but a replan is required, the algorithm must be able to reconcile this replan with the continuing execution of the current tasks. Further manipulation of the rules that govern replanning will yield more effective swarm activity to allow for more robust assignment. Assessment of battery levels and coverage values at the time of replanning can allow for the swarm to maintain tracking of detected targets and even coverage of the entire search space.

A corollary to mode replanning is search space reconfiguration. Rather than assigning agents to search under covered cells, which may still have inefficiencies if the cells are large, the search area can be repartitioned into new search cells by analyzing the coverage value of the entire search space at each replanning iteration. Section 5.4.1 discusses work done to prepare this dynamic spatial repartitioning function.

5.4.1 Dynamic Spatial Reconfiguration

As mentioned in Section 5.4, analyzing the coverage value of the entire search area at replanning iterations maximizes coverage for the whole search area. Places with low coverage values are top priority for the next search agent to be launched, where areas that were recently covered are low priority. Not only does this maximize coverage, but also maximizes the likelihood of detecting targets. Additionally, the spatiotemporal coverage measurements discussed in Section 5.2 are critical to informing these types of replanning decisions. This dynamic spatial repartitioning is accomplished using a method called Vertical Cell Decomposition (VCD).

VCD is typically used for path planning and obstacle avoidance, which are discussed in [24]. The basic steps for VCD are to draw line from every vertex of the polygon obstacles in the space to the outer boundary, use these divisions to divide the area into new cells, and calculate a desired path through the cells. In this case, we use VCD to partition our search space into polygons that become the new search cells, rather than use the cells to plan a path. The spatial repartitioning starts by excluding the areas above a coverage value threshold, or areas that have been recently covered, these ’holes’ in the search space

demand that new cells be created around them. In our algorithm VCD is simplified to four sided rectangles for predictability. Excluded regions are simplified to rectangles, the search area is a rectangle, meaning the remaining portions can be divided up into smaller rectangles. This dynamic behavior is difficult to implement due to the unpredictability of the number, size, and location of these smaller cells. This level of uncertainty requires additional conditions for proper implementation, such as area verification, if cells are too small they will not be effectively searched. Incorporating the dynamic spatial reconfiguration into the full simulation is the focus of current research.

Figures 16 and 17 depicts the creation of new search cells based on the coverage value of the search space at the time of the replan. The main steps are to exclude areas above the coverage threshold, execute the vertical cell decomposition, create new polygons, or cells, from adjacent vertices.

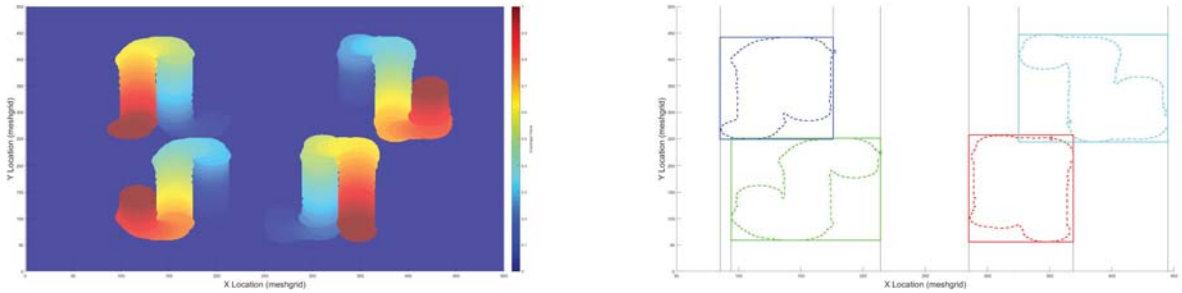

Figure 16: *A visualization of how a search area is broken into new cells around recently covered areas.*

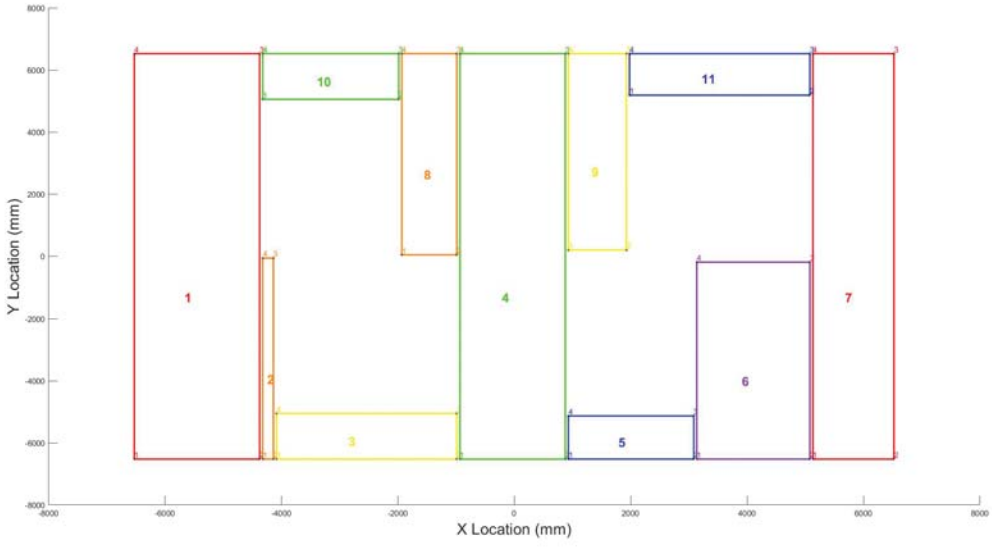

Figure 17: *A visualization of the repartitioned output with over 15 cells.*

6 Results

Simulation results verify functionality of MATLAB simulation execution which allows for rapid prototyping and iteration of algorithm components to increase overall effectiveness. Iterative runs of the simulation allow for further development and tuning of the swarm to achieve desired performance, which is discussed in the follow section. Completed simulations include 4, 6, 8, 10, and 12 agents searching for 1, 2, and 3 targets in all configuration. All simulations ran for 60 min. All figures in this section focus discussion on the 4 and 12 agents searching for 3 targets because their greater disparity offers higher resolution in the results comparison. The remaining agent cases, 6, 8, and 10, are included as a subset for comparison to show general trends. Note, 4 agent figures use blue lines, where 12 agent figures use red lines.

Figure 18 is a visualization of a full scale swarm simulation during an experiment, operating with 12 agents searching for 3 random targets. In the figure, black numbers are the search agents with light gray sensor cones and trace of their path; red numbers are untracked targets and black letters are tracked targets, all targets leave behind a trace of their path. The color patches are search cells occupied by agents assigned to the search mode. These simulation results, over 70 experiments in total, are intended to aggregate a large data set on swarm performance which will influence further developments to improve overall functionality.

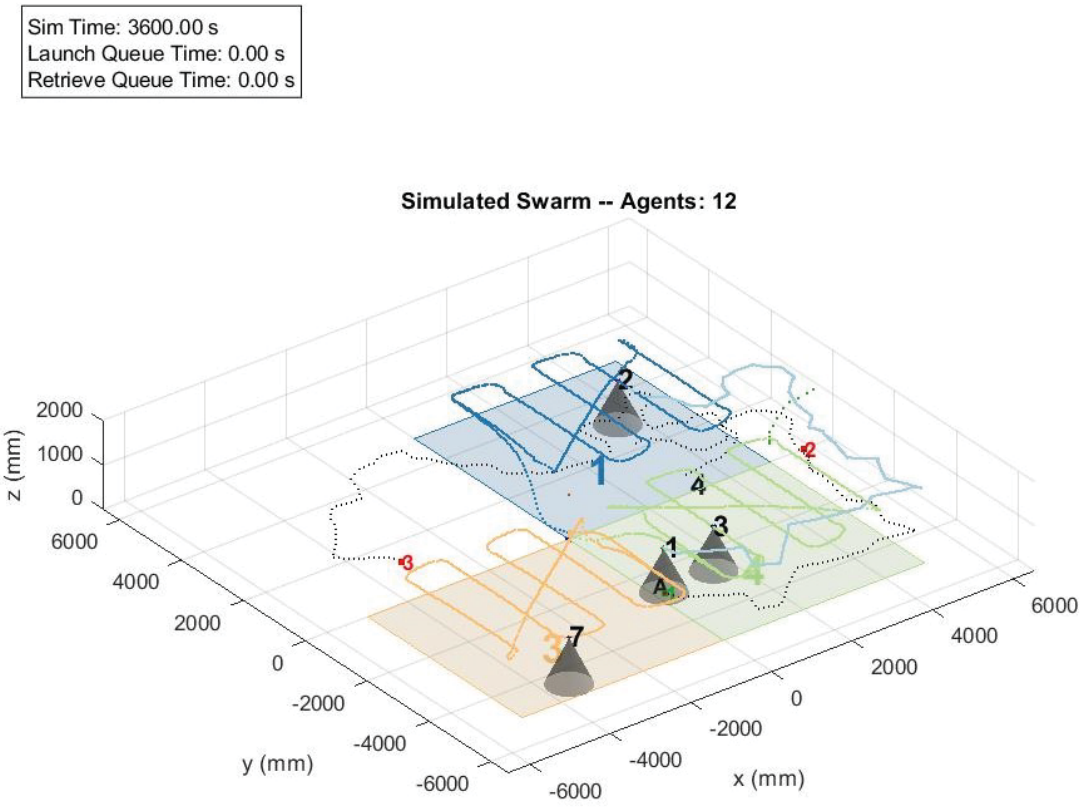

Figure 18: *A visualization of full swarm simulation in progress.*

6.1 Coverage Results

Figures 19 through 22 describe the coverage performance of the swarm in the experiment. As discussed in Section 5.2, the coverage value is a spatiotemporal measure that describes the time since last visit by a sensor. The coverage metric is designed to spike high when a space is covered, limiting the minimum measure to 50 seconds since last coverage; extended simulation results suggest that a measurement that increases with time, rather than decreases from a maximum value, will give a more comprehensive measure of search space coverage. Coverage metrics are a focus of ongoing work to accurately characterize the performance of the swarm and compare results with literature in the field.

First, Figure 19 is a coverage heat map that shows the number of replanning iterations a region was below the minimum coverage threshold, which is currently defined 47.5 seconds since the last visit. Dark red areas a locations that need attention, where dark blue areas are locations that have been recently covered. Again, the ideal case is for the entire search space to be deep blue. This metric depends on the number of replanning iterations called by the algorithm, if the number of replans is high and there was an instance where one area had not been covered in a long time this figure can be skewed or biased by an agent in track mode for a long period of time. However, the edges of the search boundary are the darkest red, as expected. Since all agents propagate from center of the map, the central base station, we

expected the boundaries to be on the lower end of coverage. Improved coverage metrics and integration of the repartitioning, discussed in Section 5.4.1, could increase the utility of this figure during experimental analysis to help determine the effectiveness of overall coverage.

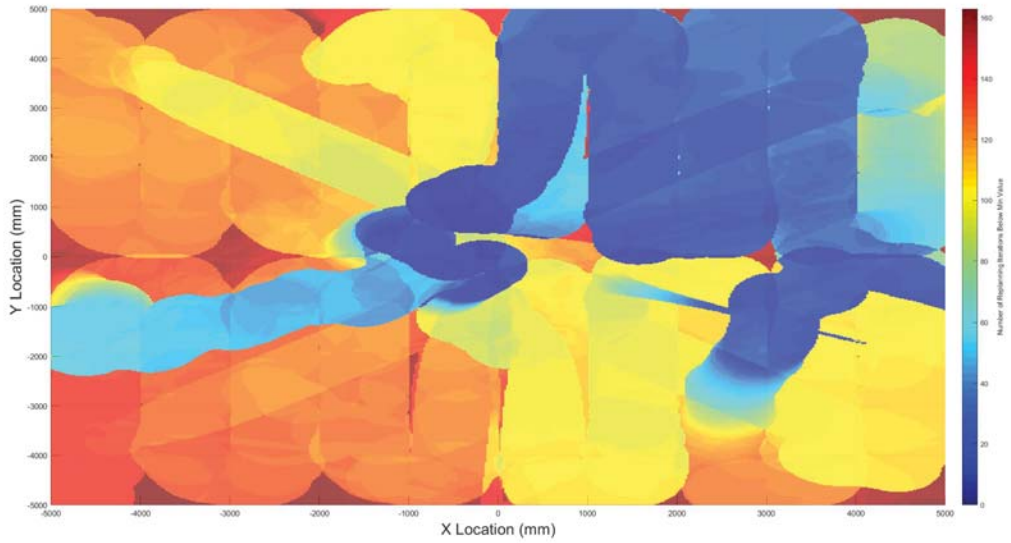

Figure 19: *A visualization of the "holes" in coverage. Deep red areas are not covered enough.*

Figures 20 through 22 show the average coverage value for the entire search space for all time. The light colored dotted lines are the values for individual experiments, the green line in the maximum value, the solid line is the average for all trials at each time step, and the dashed magenta line is the overall average of all runs for the entire experiment. Recall from Section 5.2 that the ideal case, although unrealistic, is '1' for all time, 100% coverage, or 0 seconds since last revisit. However, as shown in the figures, the general coverage value tends to increase as the number of available agents increase.

In Figure 23 the average coverage value, approximately 0.13, relates to 43 seconds since last revisit across the entire search space.

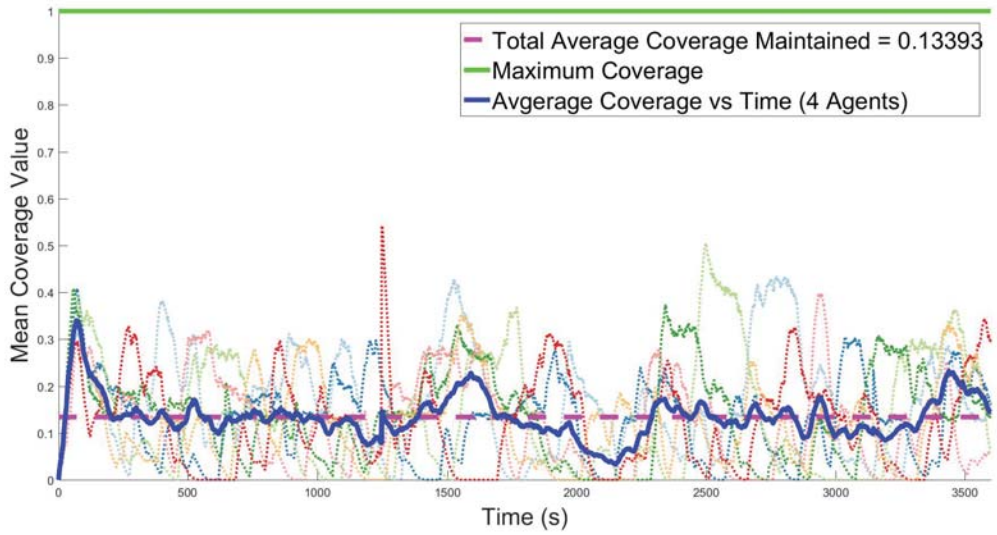

Figure 20: *Mean coverage over all time for four agents.*

Figure 21 shows the expected increasing coverage trend for the intermediate agent cases between 4 and 12.

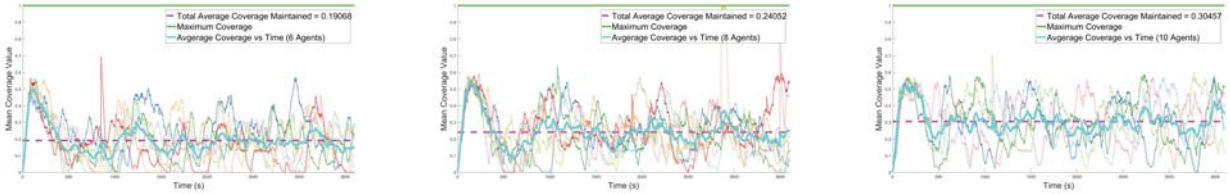

Figure 21: *Coverage results for intermediate test cases, from left to right: 6, 8, 10 agent simulations*

Finally, in the 12 agent case the average value hovers near 0.37. Relative to the spatiotemporal coverage value, 0.37 means that on average every point in the search space was visited within the last 32 seconds by an agent's sensor field.

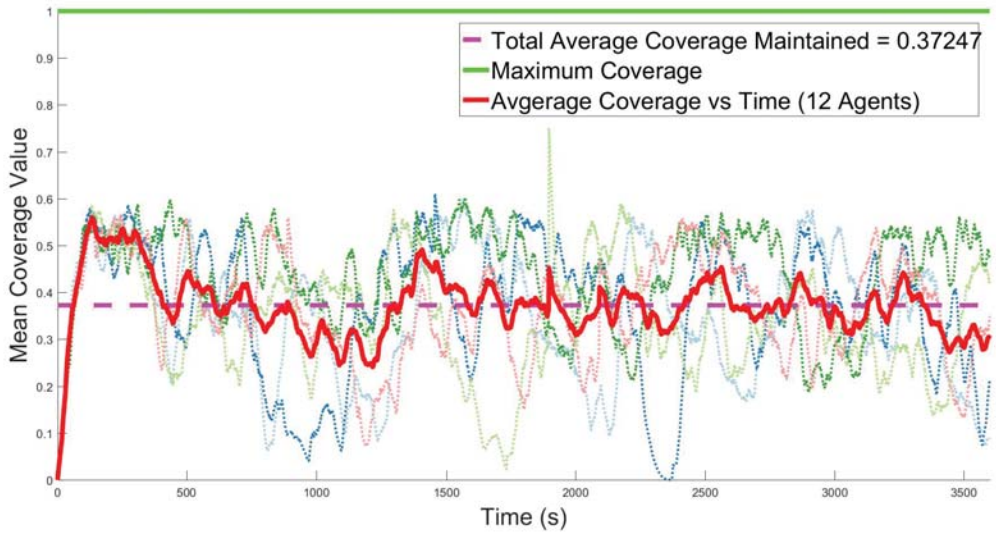

Figure 22: *Mean coverage over all time for twelve agents.*

Notice in each of the figures that each experiment tends to start with an initial spike in coverage before any targets are found the all agents are available for launch before dropping off to hover near an average value. This trend suggests that the maximum coverage value for that agent case is the initial spike and further manipulation of the swarm task assignments can raise the average coverage value to maintain this maximum value. Generally, each experiment has no momentary lapses in coverage (i.e. 0) and displays acceptable acceptable performance levels that offer opportunities for improvement.

6.2 Tracking Results

Figures 23 through 25 describe the overall tracking performance of the swarm for all time. The light colored dotted lines are the values for individual experiments, the green line shows the maximum number of targets is 3, the solid line is the average for all trials at each time step, and the dashed magenta line is the overall average of all runs for the entire experiment. Recall from Section 2.5.1 that the targets walk randomly about the space and are reset to a new location if they move beyond a set boundary just beyond the search space boundary, which ensures a constant density of targets in the space. The ideal case, although unrealistic, is all targets tracked for all time. However, as shown in the figures, the more agents available the more targets they are generally able to track, an expected outcome of adding more agents to the experiment.

In Figure 23 4 agents are able to track 0.5 out of 3 targets on average.

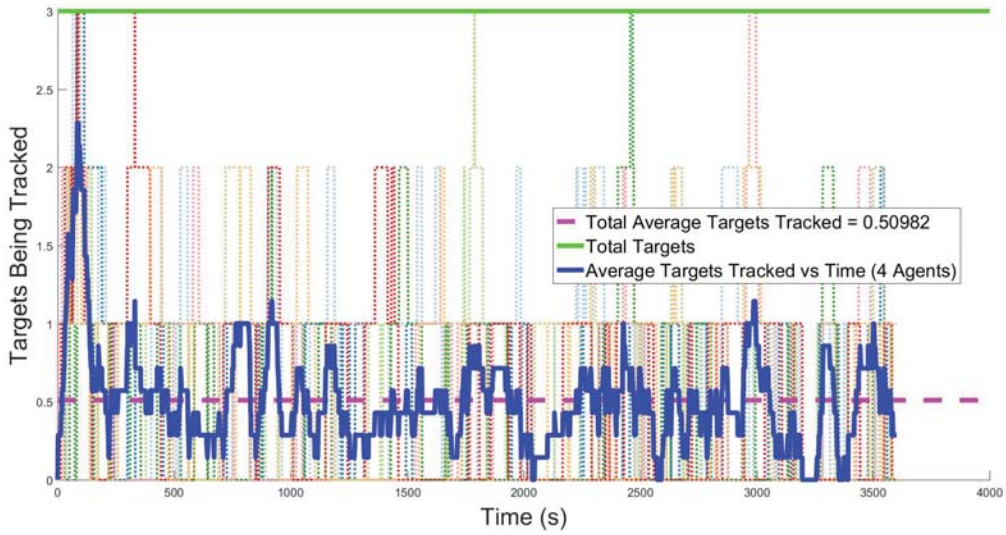

Figure 23: *Average targets tracked by four agents for all time with three available targets.*

Figure 24 shows the expected increasing targets tracked trend for the intermediate agent cases 6, 8, and 10.

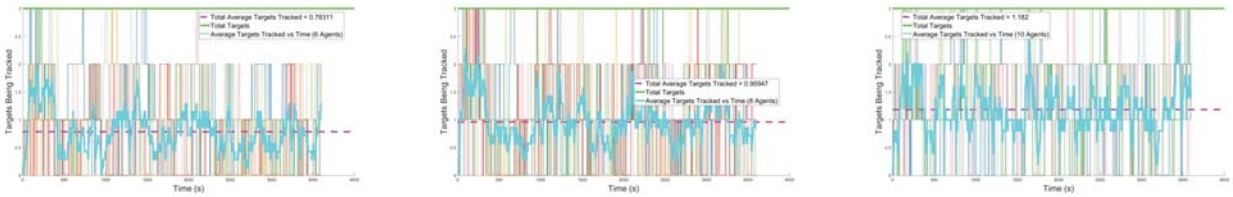

Figure 24: *Tracking results for intermediate test cases, from left to right: 6, 8, 10 agent simulations*

Figure 25 shows that on average, 12 agents are able to track nearly 1.5 targets for all time.

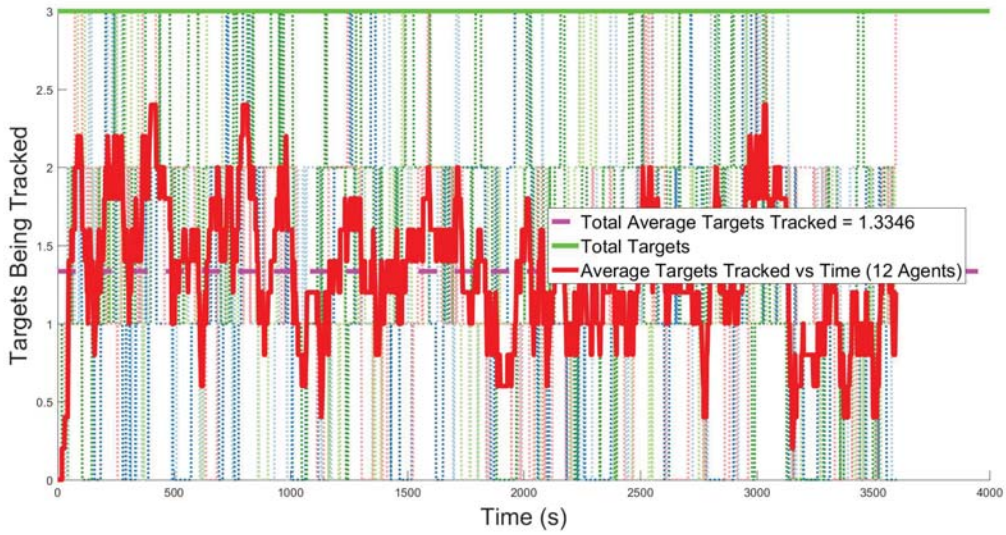

Figure 25: *Average targets tracked by twelve agents for all time with three available targets.*

Notice that no matter the number of agents there is a large degree of variability in the number of targets tracked from experiment to experiment, which is due to the resetting of targets as they leave the search space. The target reset effectively acts as a lost target even though the agent may still have the capability to continue tracking. Future experiments may consider implementing a different target model to 'bounce' around the search space, similar to a billiards ball on a table, but never leave and reset to allow targets to be tracked for longer.

6.3 Swarm Power Results

Figures 26 through 28 describe the overall power of the swarm for all time. The light colored dotted lines are the values for individual experiments, the green line shows the maximum number of agents available, the solid line is the average for all trials at each time step, and the dashed magenta line is the overall average of all runs for the entire experiment. This metric is intended to measure the effect recharging has on the operation of the swarm. The ideal case is to have as many agents flying as possible, but there may a 'sweet spot' for number of agents related to number of targets to achieve maximum performance. However, as shown in the figures, the more agents available the more agents stay airborne, an expected outcome of adding more agents to the experiment.

In Figure 23 4 agents are able to maintain approximately 1.5 agents airborne throughout the whole experiment.

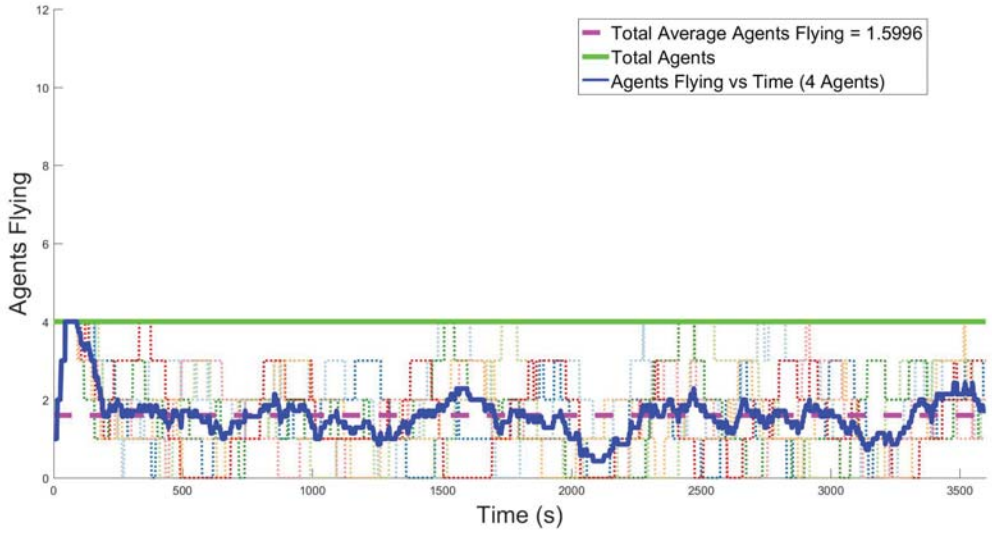

Figure 26: *Average swarm power for hour long experiment with four available agents.*

Figure 27 shows the expected increasing agents airborne trend for the intermediate agent cases 6, 8, and 10.

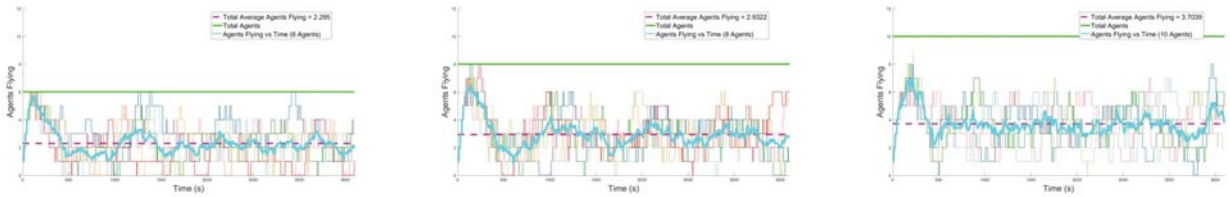

Figure 27: *Swarm power results for intermediate test cases, from left to right: 6, 8, 10 agent simulations*

Figure 28 shows that on average, 12 agents are able to keep nearly 4.5 agents airborne throughout the experiment.

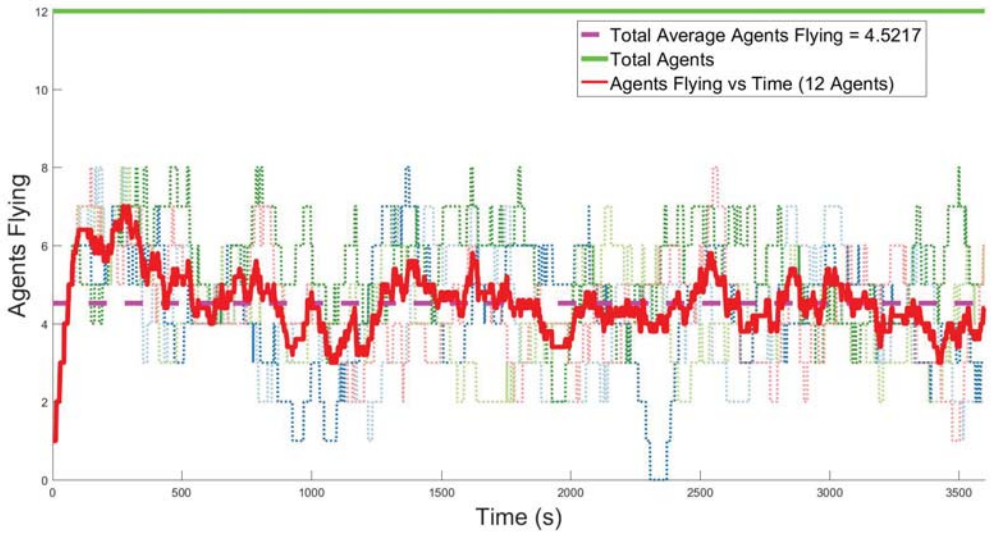

Figure 28: *Average swarm power for hour long experiment with twelve available agents.*

Battery status for each of the agents directly relates to the swarm power metric because recharging is the highest priority operational mode. Any agent will drop its task to go recharge if needed, regardless of the task it is accomplishing. Further, each agent is looping though approximately three battery life cycles, 7 min as discussed in 4.2.1, over the course of a 60 minute experiment. Overall, the increase in average swarm power as number of available agents increases gives numerical validation to an expected behavior of the swarm and motivates the investment in having as many available agents as possible.

6.4 Monte Carlo Style Results

An underlying goal of the swarm simulation is to collect as much performance data as possible, similar to a Monte Carlo Simulation, to inform future developments. The following Figures 29 and 30 depict the coverage value trends for 1 target and 3 target experiments across an increasing number of agents. As expected, both figures show an increase in coverage value as the number of agents increases. In both figures, the red stars are individual data points for each experiment; the y axis is normalized from 0 to 1, identical to the coverage figures in Section 6.1; and the trend line has associated 95% confidence interval error bars at each agent case 4 through 12.

Figure 29 shows an increasing coverage trend for the 1 target case. The error bars are very small, as the number of simulations run for each agent case is smaller than that of the 3 target scenario. Additionally, only 1 target presents less random uncertainty for agents covering the search space in track mode. Generally, the swarm is able to maintain coverage and tracking of one target, especially with larger numbers of agents. In fact, the 12 agent case as a higher coverage value for 1 target than for 3 targets, more agents are able to focus on coverage rather than tracking.

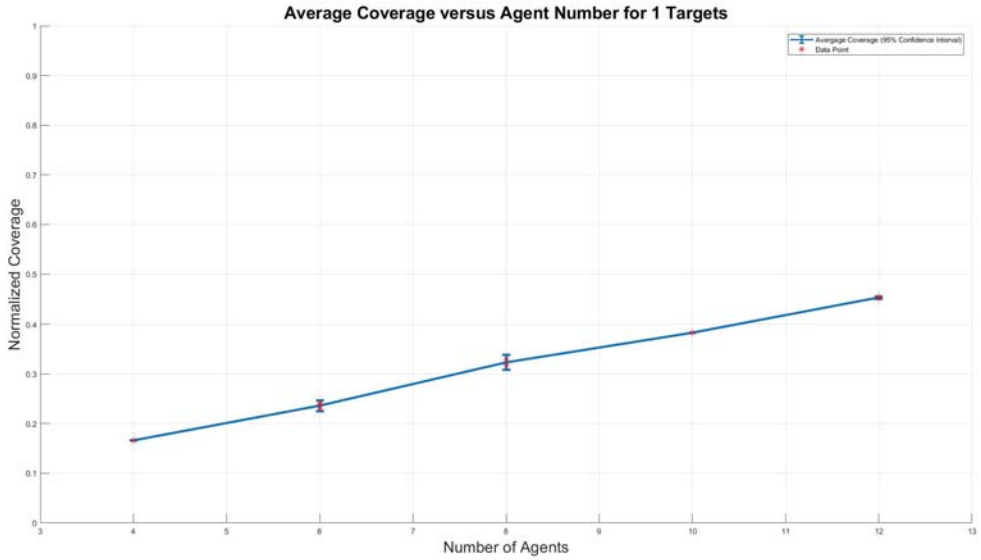

Figure 29: *Coverage trend results for 1 target, all agent cases.*

Figure 29 shows an increasing coverage trend for the 3 target case.

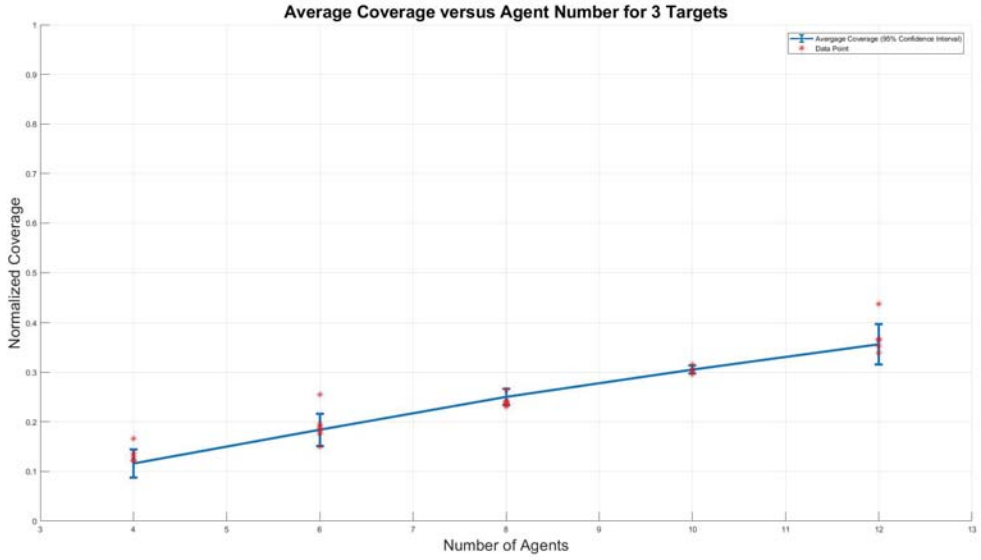

Figure 30: *Coverage trend results for 3 targets, all agent cases.*

We have run over 70 simulation experiments and the collection of more data will only increase our understanding of swarm performance. A focus of ongoing work is to continue running experiments to continue collecting data for analysis.

7 Conclusions and Suggestions for Future Work

The goal of this project is to develop, simulate, and experimentally validate a multivehicle control structure that accounts for the limited battery life of individual agents accomplishing a persistent search and track task. The control structure is developed independent of the hardware system employed for validation, but aims at mapping the relationships between control modes to increase the overall ability of a multivehicle team, or swarm. Therefore, this approach allows for scalability. Additionally, employing small, cost effective autonomous vehicles as a team increases their overall utility beyond the implementation of a single agent. Finally, current work completed supports the accomplishment of these goals and opportunities for future development.

The following subsections summarize the contributions of this work outlined in Section 1.

7.1 Designing Search Trajectories

In order to accomplish the search and track mission we designed search trajectories to enable reliable coverage of the entire search space, fully discussed in Section 3. These search trajectories took the form of lawn mower patterns sweeping back and forth across the search area. They trajectories are designed to be smooth, continuously differentiable, and evolve at a constant velocity. Each of these elements of the search trajectory enable the assumption that all sensor measurements are reliable.

7.2 Incorporating Endurance Requirements

To enable persistent swarm operations for realistic platforms we designed return to base and recharging operation modes to accounted for the limited battery life of small UAV's, fully discussed in Sections 4.1 and 4.2. These computationally simple modes enable the swarm to replenish itself over time to accomplish missions beyond the single battery life of an individual agent.

7.3 Coordinating Multiple Vehicles

We created a multi-modal control strategy to drive agents through four main modes of operation: Search, Track, Return, and Recharge. Incorporating each of these task modes into a larger experiment with multiple targets and vehicles requires a high level of coordination. In order to accomplish the mission we prioritized tasks as follows: maintain battery to operate, track detected targets, assign all search cells. The coordination techniques are further discussed in Section 5. Coordination boiled down to a heuristic decision making model to achieve the task priorities that accomplish the search and track mission. Future developments of this work will focus on improving the swarm's task coordination to boost performance.

7.4 Future Work

The development of this project is ongoing. Additional functions and diagnostic tests can be completed using the available foundation simulation. Additional heuristic tests include coverage efficiency mapping, task effectiveness, and sustainability rate to analyze the effectiveness of the algorithm. Future papers will include these additional results for further discussion.

Incorporating the vertical cell decomposition to execute dynamic spatial reconfiguration at replanning iterations is the immediate focus of follow on research. The coverage value can be used to inform dynamic spatial reconfiguration of the search area based on the coverage values across the entire search space. During a swarm re-planning phase the search area is repartitioned to ensure coverage of areas that were not previously covered by other search agents. This reconfiguration would enable new search agents to be as efficient as possible and maximize the coverage value across the entire search area. We can also consider implementing more advanced mathematical models for the decay of the coverage value supported by other work in the field. The current steps taken have been focused on setting up the capability which will be followed by functional updates. Another focus for future work is to implement a more accurate battery discharge model which will inform the health status of each agent. We have found in both research and experimentation that the voltage level of a quadrotor battery greatly fluctuates according to the load drawn from the four motors. This fluctuation makes voltage a difficult state to measure for battery health status. Instead, work cited in [25] makes endurance and range estimates for UAVs based on the system's battery specifications, flight characteristics, and velocity. Endurance estimation based on agent velocity is viable candidate for implementation because we designed the search pattern to travel at a constant speed.

Finally, an original stretch goal of this work was to experimentally validate the simulated swarm on hardware existing within the department. Work completed on this project supports future integration with hardware for further development and testing.

References

[1] M. Valenti, B. Bethke, G. Fiore, and J. P. How, "Indoor Multi-Vehicle Flight Testbed for Fault Detection , Isolation , and Recovery," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, no. 8, pp. 1–18, 2006.

[2] S. Bouabdallah, "Design and Control of Quadrotors With Application To Autonomous Flying," *École Polytechnique Fédérale De Lausanne, À La Faculté Des Sciences Et Techniques De L'Ingénieur*, vol. 3727, no. 3727, p. 61, 2007.

[3] M. Valenti, B. Bethke, D. Dale, A. Frank, J. McGrew, S. Ahrens, J. P. How, and J. Vian, "The MIT indoor multi-vehicle flight testbed," in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2758–2759, 2007.

[4] V. Kumar, "Architectures, abstractions, and algorithms for large teams of robots," *2009 Second International Conference on Robot Communication and Coordination*, vol. 66, pp. 409–419, 2009.

[5] B. Michini, T. Toksoz, J. Redding, M. Michini, J. How, M. Vavrina, and J. Vian, "Automated Battery Swap and Recharge to Enable Persistent UAV Missions," *Infotech@Aerospace 2011*, no. March, pp. 1–10, 2011.

[6] F. Bourgault, T. Furukawa, and H. F. Durrant-Whyte, "Decentralized Bayesian Negotiation for Cooperative Search," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, no. 1, pp. 2681–2686, 2004.

[7] T. Furukawa, F. Bourgault, B. Lavis, and H. F. Durrant-Whyte, "Recursive Bayesian search-and-tracking using coordinated UAVs for lost targets," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2006, pp. 2521–2526, 2006.

[8] B. Lavis, T. Furukawa, and H. F. Durrant Whyte, "Dynamic space reconfiguration for Bayesian search and tracking with moving targets," *Autonomous Robots*, vol. 24, no. 4, pp. 387–399, 2008.

[9] AeroVrionment, "RQ-11B Data Sheet," 2016.

[10] Northrop Grumman, "RQ-4 Block 30 Global Hawk Datasheet," 2012.

[11] US Department of Defense, "Department of Defense Announces Successful Micro-Drone Demonstration > U.S. DEPARTMENT OF DEFENSE > News Release View," 2017.

[12] L. C. Mak, M. Kumon, and M. Whitty, "Design and Development of Micro Aerial Vehicles and their Cooperative . . . ," . . . *Micro Air Vehicles*, vol. 4, no. 1, pp. 91–98, 2009.

[13] Bitcraze AB, "CrazyFLie 2.0," 2016.

[14] R. L. Stroup, "Feedback Controller Design for an Autonomous Robotic Manipulator with Applications to Quadcopter Capture, Recharge, and Re-Launch." 2016.

[15] M. Kutzer, "Scalable Mobile Swarm Testbed," tech. rep., United State Naval Academy, 2016.

[16] S. Johnshon, R. Stroup, J. J. Gainer, L. D. DeVries, and M. D. M. Kutzer, "Design of a Robotic Catch And Release Manipulation Architecture (CARMA)," in *ASME 2017 International Mechanical Engineering Congress and Exposition*, pp. 1–8, 2018.

[17] D. Mellinger, *Trajectory Generation and Control for Quadrotors*. PhD thesis, University of Pennsylvania, 2012.

[18] G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin, "Quadrotor Helicopter Trajectory Tracking Control," *Electrical Engineering*, vol. 44, no. August, pp. 1–14, 2008.

[19] N. Sydney, D. A. Paley, and D. Sofge, "Physics-inspired motion planning for information-theoretic target detection using multiple aerial robots," *Autonomous Robots*, pp. 1–11, 2015.

[20] L. Stone, "Likelihood Ratio Detection and Tracking," no. July, p. 10, 2002.

[21] L. D. Stone, C. A. Barlow, and T. L. Corwin, *Bayesian Multiple Target Tracking*. Boston: Artech House, 1999.

[22] E. M. Wong, F. Bourgault, and T. Furukawa, "Multi-vehicle Bayesian search for multiple lost targets," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2005, no. April 2005, pp. 3169–3174, 2005.

[23] M. Kutzer and L. DeVries, "Testbed for Multilayer Conformal Additive Manufacturing," *Technologies*, vol. 5, no. 2, p. 25, 2017.

[24] J.-C. Latombe, *Robot Motion Planning*. Boston, MA: Springer US, 1991.

[25] L. W. Traub, "Range and Endurance Estimates for Battery-Powered Aircraft," *Journal of Aircraft*, vol. 48, no. 2, pp. 703–707, 2011.

List of Figures