Cognitive UAS Path-Planning for Large Spatial Search

By

Thomas Le Pichon
B.S., University of Kansas, 2017

Submitted to the graduate degree program in Department of Aerospace Engineering and the Graduate Faculty of the University of Kansas in partial fulfillment of the requirements for the degree of Master of Science.

____

Chair: Dr. Shawn Keshmiri

____

Dr. Mark Ewing

____

Dr. Ray Taghavi

Date Defended: 23 July 2020

The thesis committee for Thomas Le Pichon certifies that this is the
approved version of the following thesis:

Cognitive UAS Path-Planning for Large Spatial Search

Chair: Dr. Shawn Keshmiri

Date Approved: 23 July 2020

Abstract

Search and Rescue/Destroy missions are some of the most high-risk situations in modern engineering. Every mission nearly always presents a life or death scenario for one or more individuals, with the penalty for failure often being human lives. Modern Search and Rescue/Destroy missions implement the use of autonomous systems in the form of giving an unmanned autonomous aerial system(s) the task of searching a given area in the attempt of discovering one or more objects of interest. Though this ingenuity has already benefited the line of work, these unmanned systems are still not being used to their full potential. Some means of planning how to search the area must be developed, with the most basic means of accomplishing this task being creating a predefined path which is guaranteed to cover all known areas. To increase the rate of success and decrease necessary search time, a pseudo-random search method, known as meta-heuristics, is used to develop a new path planning algorithm to search the field in an intelligent manner. This work develops a means of turning meta-heuristic optimization into a cognitive navigation with autonomous path-planning algorithm that is decoupled from apriori information, with minimal requirements for initiation. To account for the higher performance requirements of such a method, novel guidance methods were developed to meet said demands. Simulations suggest that the proposed search method performs better on average than the current accepted basis.

Acknowledgments

This work was done in direct collaboration with Dr. Shawn Keshmiri and Jeffrey Xu of the KUFRL. This outlines in greater details the portion that the author had the most involvement in. L_N guidance in particular is primarily the work of Jeffrey Xu and Dr. Shawn Keshmiri. This work could not have been completed without the support and groundwork laid by the KU Flight Research Lab (KUFRL).Though not all members contributed directly to this research, all were vital in providing the necessary tools in its completion. Furthermore, this work was completed with funding by Heising-Simons foundation and NASA and DARcorporation project # 18CDA067L. Authors greatly appreciate HeisingSimons foundation, NASA, and DARcorporation support.

Table of Contents

List of Figures

1 Introduction

Search and Rescue (SaR) and Search and Destroy (SaD) missions exist as one of the highest risk missions, with failure resulting in the loss of lives. In such a time sensitive situation that is a Search and Rescue (SaR) or Search and Destroy (SaD) missions, efficient mission planning is of the utmost importance. When the risk of failure results in the loss of lives, not guaranteeing optimal performance is tilting the scales of “risk versus reward” towards a catastrophic risk. Despite their significant differences in terms of end goal, SaR and SaD missions both are built off a core of finding one or multiple points of interest. As such, this work will adopt standard SaR terminology as generic terms, and not as mission specific terms. Studies have found that SaR missions in the late 20th/early 21st have a mean success rate of 95%, with the average successful search taking 7-8 hours, and the average failed search being abandoned after 30 hours, as shown in Ref. [1]. Achieving such a number requires a full team that is fully dedicated to the task, focusing all of their efforts on a single area.

The fundamental theory of SaR missions can be found in References [1] and [2]. The primary baseline for this method is a series of sequential, linear sweeps of a strictly defined rectangular area, such as is shown in Fig. 1.

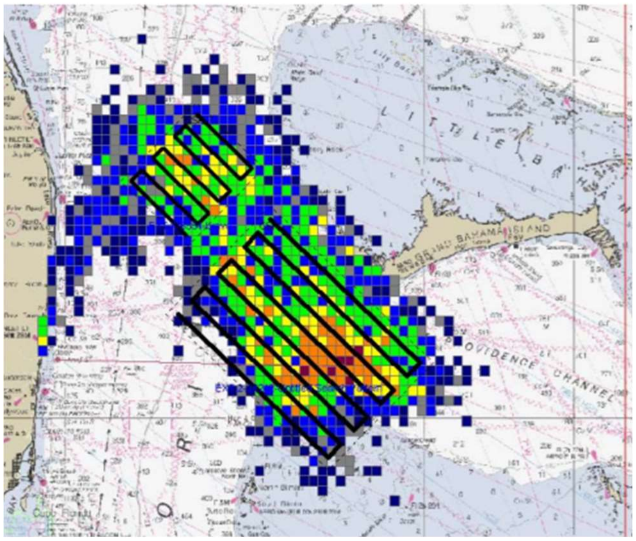

Figure 1 Example of Search and Rescue Map [2]

Though this method may seem simple when only observing the final product, the mathematics that hold up the statistical advantage of this method are far from trivial. However, this does not exempt it from risk or error. This method is developed from the ground up for search personnel and manned systems, with an emphasis on accounting for human error. This work instead proposes the use of unmanned aerial systems (UAS) and potentially unmanned ground vehicles (UGV), reducing the coupling to human error, allowing for more cost-efficient methods. Note that this work presents primarily only the use of UAS, however the base theory is applicable to all forms of autonomous robots as the navigation is entirely decoupled from its environment. Furthermore, the author would like to emphasize the difference between cognitive autonomy and autonomous control. While robot guidance and autonomous guidance and control have been extensively developed, this work is on cognitive navigation with autonomous path-planning to produce a system that can make intelligent decisions on its own. The desired result of such a system is one that acts on the same level as its human handlers. That is the unmanned system can make cognitive and intelligent decisions without any human interaction needed. Elevating the position of a robot from being a follower to being a leader or collaborator, or corobot, expands the capabilities of both the humans and their machines. In an environment in which robots can lead humans, the capabilities of both will be maximized. The primary method proposed to reduce the search time is through meta-heuristic path-planning, which can be considered a subset of optimal path planning.

Methods that seek only to find the single most correct answer to complex problems are termed optimizers. The main issue with existing aircraft trajectory optimization methods is that it requires a known target spatial position (termina or transitional), i.e. the aircraft must know its receding horizon. When an optimization problem has no explicitly known end goal, an infinite number of options exist and it becomes exponentially more time consuming to solve, as it will need to observe all known possibilities to assure optimality. One example of this would be a series of equations in which a system of equations has more variables than equations. These optimization problems become even further complex when applied to situations such as SaR, in which environments can be rapidly and chaotically changing and have an unknown moving terminal.

Path-planning can be defined as the mathematical process of producing a desired position for a UAS or UGV over time. Standard Guidance, Navigation and Control (GNC) systems are reliant on apriori information in the form of predefined waypoints such as those presented in reference [3]. Most algorithms referred to as optimal path-planners prioritize the production of easier paths (found in reference [4]) to follow and/or improved tracking [5 & 6]. Furthermore, prior works make significant assumptions to simply the physics of the system, whereas this work remains constrained to aircraft dynamics, which becomes particularly important with high inertia and high-speed vehicles. The proposed method decouples itself from the necessity of waypoints and/or pre-defined paths while also adaptively converging onto an unknown Point of Interest

(PoI). To avoid confusion with the technical definition of point (an entity with no mass or volume), the term Object of Interest (OoI) will be used instead to not imply a lack of dynamics of the search target. This is achievable through the aforementioned branch of optimal path-planning dubbed meta-heuristic path-planning.

Meta-heuristics originate from the field of computer science, a little under three decades ago, with the purpose of finding an acceptable solution as quickly as possible, instead of a single optimal solution being discovered in as much time as necessary. Prior to meta-heuristics, the field of optimization concentrated solely on absolute optimization. One example of an optimal solver is the Hungarian Assignment Method developed in 1955 [7] in which a set of n agents are tasked with completing some specified number of jobs as optimally as possible. Though this method can produce an optimal solution quickly, it is also possible that the method will have to be repeated/looped multiple times, reducing the computational efficiency of the algorithm, especially as it approaches the eventuality of checking every possibility, due to it being constrained by its dimensionality. Although this is sometimes acceptable in the field of computer science when time is not a critical factor, it is not acceptable in situations such as SaR in which lives are on the line. Therefore meta-heuristics are used instead, as the entire purpose of the field of meta-heuristics is to avoid such time-consuming scenarios regardless of the complexity of the problem or the size of the data set.

Metaheuristic methods are mainly modeled off natural phenomenon such as animal behavior or evolution. The first widely spread of these methods is the Cuckoo-Search (CS) algorithm, found in reference [8], which mimics the brood parasitic behavior observed in cuckoos. Some bio-inspired methods such as Marriage in Honey Bees Optimization (MBO), Artificial Fish-Swarm Algorithm (AFSA) and Monkey Search existed prior to CS, however CS was largely the most influential of early meta-heuristic methods becoming a foundation off of which subsequent algorithms would build from. Bio-inspired methods developed post CS include Ant Colony Optimization and Whale Optimization, both of which have seen some widespread use, as well as less influential algorithms such as the Bat Algorithm. These works can be found references [9 – 14]. The major algorithm of note post CS is the Gray Wolf Optimizer (GWO) and the subsequent MultiObjective Gray Wolf Optimizer (MOGWO) shown in references [15 & 16], which rapidly overtook both its predecessors and successors, including generic methods such as the Predator-Prey Optimization method presented in reference [17]. The original work of reference [18] compares its own results to several benchmarks, leaving a defining mark on the world of meta-heuristics. The authors behind GWO would later go on to be inspired by the paper on hawk behavior [19] and produce their own Harris Hawk Optimizer [20] which lends itself incredibly well to the field of path-planning. Of these algorithms, GWO and HHO were applied to the fields of SaR and SaD.

This work develops path-planning algorithms based on some of the more famous meta-heuristic methods, primarily GWO and HHO (dubbed GWSaR and HHSaR), and fusing them with the fundamental theory of SaR to produce a highly cognitive and adaptive means of path-planning for unmanned systems into unknown and dynamic territory. A general overview of this topic can be found in prior work developed by the author in reference [21]. The use of meta-heuristic methods in navigation and path-planning of autonomous systems has become increasingly popular in recent times, such as remote-controlled carts and small robots similar to those presented in reference [22]. Even dating back to the late 20th century, genetic algorithms were implemented in the field of robotics for path-planning purposes in small, low-speed systems [23]. Genetic algorithms are a form of meta-heuristic that use a process mimicking evolution and iterating upon the most successful solutions. These methods tend to be more computationally intensive than their non-genetic counterparts. Some examples limit the full range of robot movement to reduce the amount of computations at the cost of restricting control of the system [24]. This computational cost has not dissuaded the use of genetic algorithms in aircraft, such as those presented in references [25-29]. Low computational costs and few constraints are largely absent in these methods. HHSaR aims to work past these boundaries, presenting a far less intensive method while also requiring less initial information. Additionally, the nature of HHSaR allows every individual member of a party to only perform its own calculations and not have to concern itself with its compatriots, instead receiving only the necessary relevant data, such as the position of other agents.. Non-genetic meta-heuristics have seen much stronger use due to their lower complexity at no cost to efficiency or applicability. Such meta-heuristics have been used in a variety of applications ranging from traffic navigation in reference [30] to optimizing submarine propeller designs in reference [31]. The published research available that comes closest to performing as well/similarly to HHSaR is a pseudo logic-tree based heuristic developed by Elodie Chanthery in her 2005 PhD.Dissertation at the Ecole Nationale Supérieure of Air & Space Engineering in France, which can be found in reference [32]. In her work, Chanthery proves her algorithms capability of taking large quantities of complex missions at once and providing a steam-lined and optimized mission plan with no overlap or redundant paths. However, the presented method requires apriori information and knowledge of the space between missions, which HHSaR can act entirely independent of at no loss of efficiency. Furthermore, despite in depth simulation testing using multiple aircraft models in an exhaustive range of situations, no further work can be found beyond that of the dissertation.

A dynamic path-planning algorithm such as HHSaR does not come without its own faults. Frequent turns and aggressive maneuvers force the necessity of high-performance and safe guidance and control methods. As such, a series of novel longitudinal and lateral-directional guidance algorithms were developed with dynamic constraints in mind [33]. These produced paths may be more varying than usually, they are still practical and achievable due to the dynamic constraints placed on the system. This also allows for the use of heterogeneous UAS teams due to the physics-based constraints on guidance and system decoupling in the navigation.

Lateral-directional guidance is meant to produce state commands for the system to follow. Although straight-line and loiter guidance can be relatively simple to design to produce reasonable commands, a path with frequent turns and maneuvers planned can often result in highly aggressive commands that become impossible to achieve, resulting in large tracking error, or worst, putting the UAS in high risk positions.. The desired result of lateral-directional guidance is to calculate a ψ_{cmd} or ϕ_{cmd} based on aircraft position data to send to the system controller to best follow the desired path. An in-depth comparison of various guidance methods for fixed-wing UAS can be found in [34]. One such guidance is the Nonlinear Guidance Law (NLGL) [35], which finds the intersection point of a circle of radius L centered on the agent and the desired path to be followed from which an a_{ycmd} is calculated based on V_T,ψ, and L. Other examples of lateral-directional guidance algorithms include Pure Pursuit and Line of Sight-Based Path Follow (PLOS) [36]. As the name suggests, this method makes use of both Pure Pursuit and Line of Sight-Based guidance algorithms to find a target to converge onto the desired waypoint line. Linear Quadratic Regulator (LQR) control theory is used to create a guidance in [37] to produce an optimized command based on position and heading. Higher level mathematics have also been applied beyond LQ guidance in the form Vector Field (VF) Path following which produces a vector field defining a desired direction for the UAS at all points.

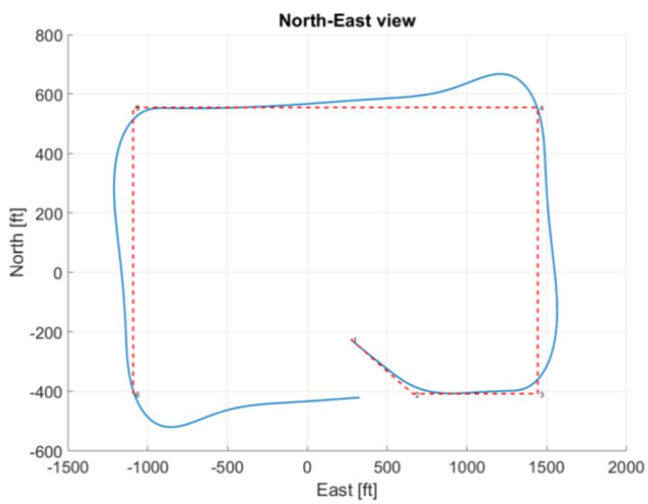

Figure 2 Simulation of LQR guidance on UAS

Despite all these methods having proven they can produce desired results, they are all flawed in their own way, making them incompatible with the developed path-planning algorithm. All the following observations are based on the results of [34]. NLGL is reliant on being near enough to the desired path, else it will become increasingly inaccurate. PLOS displays a high level of sensitivity to both tuning and type of path to be followed. That is, a set up for PLOS that works for a straight-line path can be several orders of magnitude too large for a path with a curve/turn. VF struggles to converge on turns, maintaining a steady state error in loiter patterns. LQ Guidance solves many of these issues, however it can have aggressive overshoot in tight corners, as seen in Fig. 2. Such overshoot on a path with frequent turns will result in inevitable and frequent oscillatory maneuvers. These issues in LQ guidance were solved in [38] through a pseudo adoption of Dubins Paths [39], however this required making some modifications to path planning, which is best avoided in this scenario in which guidance is being chosen based on path-planning.

Conversely, less complex algorithms are often used as industry standards, such as L_1 [40] which finds the point at distance L from the UAS on the line to follow. Subsequently, L_2 and L_2^+ [41] use two reference distances and takes the mean of their guidance commands as the final command. The lateral-directional guidance method described in [42] is developed using the popular guidance methods of L_1, L_2, and L_2^+. The primary goal of the newly developed guidance is to produce a cognitive guidance that follows multiple "'look-ahead" points to eliminate overshoot while optimizing tracking accuracy. Such a guidance method was originally developed with travel through cluttered environments and tight turns (such as urban environments) in mind, though has also been repurposed for use in a meta-heuristic based GNC.

Longitudinal guidance for the most part consists of altitude and velocity commands or holds, that has been simplified and augmented using a PI outer controller. In general, most altitude hold algorithms are effective in correcting small perturbations, however, will produce an aggressive response to large errors or sudden perturbations, as seen in Fig. 3, which depicts a PID guidance reacting to a large initial error. Such a maneuver can easily result in stall for the majority of UAS. Such a rapidly changing and high magnitude output occurring every time a somewhat aggressive maneuver is requested spells disaster for the desired path-planning.

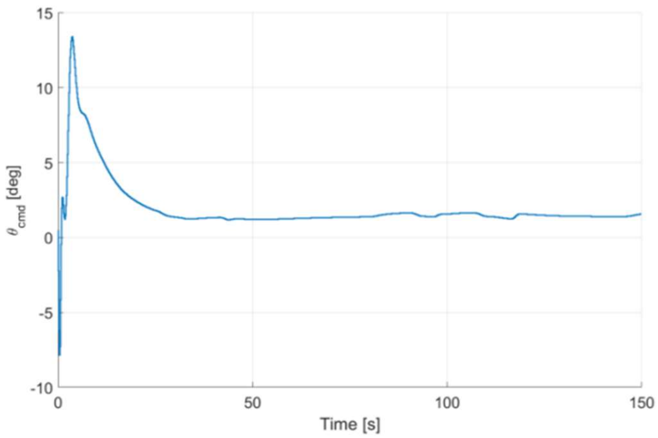

Figure 3 Simulation of longitudinal PID guidance on UAS.

There exist longitudinal guidance methods such as in [43] that can account for this through the use trajectory optimization with piecewise-affine (PWA) control and Lyapunov based techniques, unfortunately many of these suffer from potentially dangerous assumptions, such as angle of attack always being zero. As such, a novel longitudinal guidance was derived entirely from aircraft dynamics, resulting in a guidance that not only keeps aircraft trim conditions salient, but also produces conservative outputs to avoid self-induced stall and loss of control.

The use of unmanned systems such as UAS and UGV has become commonplace in modern search and rescue missions. Assigning autonomous beings to take on the same task of searching an area for possibly multiple OoI requires an intelligent means of rapid objective completion. The standard methods used by manned teams such as patrol routes are not optimal for a unit of unmanned systems. As such a pseudo-optimal search algorithm/path-planning method is developed using the field of meta-heuristics. A set of novel guidance methods for both longitudinal and lateral-directional were developed alongside the path-planning algorithm to account for the increased complexity of potential commanded maneuvers. Results shown later in this work suggest that such a method reaches completion sooner than the standard methods.

2 Theory

2.1 Search and Rescue Science

The work presented in this section uses [1] and [2] as primary sources. These works were intended to be used as a global coverage of the topic and are officially published by the United States Coast Guard.

The skill, and moreover, the size of the search party directly correlates to the probability of mission success. In technical terms, probability of success (POS) is quantified as the product of probability of containment (POC) and probability of detection (POD) (i.e. POS = $\text{POC} \times \text{POD}$) [1]. POC represents the likelihood of the object(s) of interest (OoI) being present within said targeted area. In theory, the larger this search area grows, the closer a 100% POC is obtained. The size of the designated area, however, is limited to the size and available equipment of the search party. As such, the global space is instead divided into smaller subsections to be targeted. POD represents the likelihood of the OoI being found if the specified sub-area is searched. This can be seen in a probabilistic formulation in Eq (1), in which $P(A)$ represents the likelihood of the OoI being in the given search area (i.e. POC), and $P(B)$ represents the likelihood of the POI being detected by the rescue party. Accordingly $P(A \cap B)$ represents the possibility that the POI will be in the given area and will be detected by the party (i.e. POS), and $P(B|A)$ represents the probability that the POI will be detected if they search the area in which it is contained (i.e. POD).

$$P(A \cap B) = P(A) \times P(B|A) \tag{1}$$

Since detection of the OoI is heavily reliant on sensor accuracy and situational awareness of the search party, it is possible for the OoI to go unseen in searched areas. As such, passing over the same area repeatedly increases the POD for the mission. The resultant cumulative POD can be calculated based on the net product of all POD of each subsequent search, as seen in Eq (2). For the purpose of this work, the active sensor will be a radar.

$$POD_C = 1 - \prod_{i=1}^{N}(1 - POD_i) \tag{2}$$

In this equation that calculates cumulative POD (Subscript *C*) , *i* represents the current number of completed searches performed of a designated area, and *N* the total number of completed searches that will be performed of the same designated area. Based on the previously established theory, POS can be increased in one of two manners. The first would be increasing POC by increasing the size of the search area. This is not a practical means of improving success as doing so increases the amount of logistical and expense issues of covering the entire area. Although highly accurate means of defining the search area already exist for search and rescue missions, as one may discern from the 95% mission success rate [1] , no such data can be found on search and destroy missions, which presents a significantly more complex task with the possibility of OoI moving at missile speeds. One example of such a scenario would be the September 14, 2019 drone strikes on two key Saudi oil facilities, which consisted of an assault of systems so small and so fast they could not be intercepted by Saudi Arabia's anti-missile defenses. The existence of such a scenario should place doubt on a 95% success rate when being applied to high-speed entities. If an algorithm seeks to be generic enough to be used in both scenarios, having a predefined means of determining search area or attempting to improve success via area increase would not be optimal in the slightest. As such the alternative means of increasing POD to drive up POS is preferable, as POD can be reasonable pushed closer to a value of one (100%) by repeatedly searching the same area. This however does have a limit to how effective it can be, as every repeated search of an explored area reduces the POC. The possibility of the OoI having been in an already searched area is non-dismissible as no sensor can achieve 100% accuracy, however the sensors must still be trusted to a certain level. If one search of an area returns no results, then it should be implied that there is a higher probability that the OoI is in another unexplored area. To model this phenomenon and capture the concept of trust in ones sensors, references [1] and [2] designed a relation for POC and POD, similar to the relation of kinetic energy and potential energy in that as one increases, a proportional of the other should decrease. That is, as POD increases with repeated searches, POC should decrease. This is defined by the relation presented in Eq (3).

$$POC_{i+1} = POC_i \times (1 - POD_i) \tag{3}$$

As such evaluating the worth of repeated searches is highly important to avoid losing time in the form of an attempt to synthetically driving up the POS.

More advanced SaR mathematics exists; however, these delve into observing specific geography and conditions of a search area (e.g. altitude) as opposed to observing a generic case. Since this work seeks to function on a generic level and require no prior or concrete knowledge of the area, the theory and mathematics defined beyond what is presented in this work is not considered, despite having its own merit.

2.2 Meta-Heuristic Optimization

This section presents the theory behind *GWO* and *HHO*. Meta-heuristic algorithms have very little overlap in terms of theory, outside of minor concepts. The one major overlap across heuristic methods is that of exploration/exploitation phases. Exploration is defined as the portion of the algorithm that takes broad, random sweeps of the possible space, with little correlation between steps. Conversely, exploitation is the portion of the algorithm in which individual steps are much smaller and far more calculated.

The final subsection of this section details how to initiate such an algorithm, for both computer science and physics based scenarios.

2.2.1 Gray Wolf Optimization

Gray Wolf Optimization developed by Mirjalili et. al in references [31 & 32] uses the hierarchical pack structure and encircling hunt behaviors of gray wolves to model the solution to a cost function optimization problem, with each wolf/pack member representing one solution to the cost function. The lower the cost of the wolf, the higher in the hierarchy they are placed. In nature, this hierarchy (Fig. 4) consists of one or two leaders (dubbed alpha, given symbol α), one secondary leader with veto power over the alpha (dubbed beta, given symbol β), a group of stronger or more skilled wolves (dubbed deltas, given symbol δ), and all other wolves are placed into the subservient category of omega (ω). Such a hierarchical method might also be referred to as horizontal organization. *GWO* adapts this structure as a means of determining which wolves are considered relevant, and how relevant they should be. In the context of optimization, the cost of a wolf represents how close to the solution (or to an error of 0) the individual input is. That is, if Y is our desired solution and y is the solution produced using the input suggested by the observed wolf, the wolfs cost can be defined as $Y - y$. In the context of reality and aircraft path-planning, the wolfs cost can be defined as the distance between the predator and the prey (or the aircraft and the observed reference point). The lowest cost wolf, is given the rank of alpha, the second is given the rank of beta, the third is given the rank of delta, and all remaining wolves are given the rank of omega and are ignored in the current iteration of the decision making process.

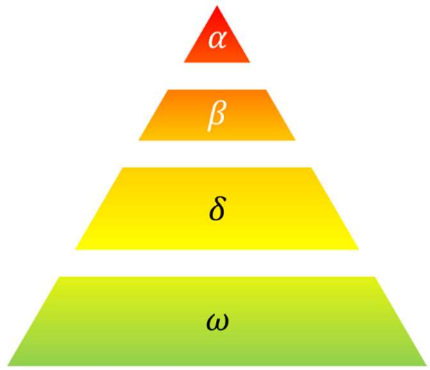

Figure 4 Visual Representation of GWO Hierarchy [33].

Observing a generic cost function $J(x) = y$ in which x is the current control/input to the system and y is the resultant state/output, n number of wolves are initiated for the algorithm with each wolf being given a random x at which to evaluate their own respective y to establish a hierarchy.

If the terminal state, termed the prey, is known, then the wolves begin the encircling behavior, which is the *GWO* equivalent of the exploitation phase that is modeled in the mathematical definitions found in Eqs (4-7). In this set of definitions $\vec{A}$ and $\vec{C}$ are vectors based on random vectors $\vec{r}_1$ and $\vec{r}_2$ with range of [0,1] and $\vec{a}$ is a vector that decreases linearly in time from a value of two to zero. No detailed reason for how any of these (listed prior to and beyond this sentence) ranges were defined is given in [31], which presents potential theory issues when abstracting this computer science concept to the movement of an unmanned system. It is likely that these ranges are to preserve the 0-1 bounded positive values for the output as such magnitudes would bound A by zero and one. $\vec{X}(t)$ is the position of the observed wolf at time t, and $\vec{X}_p(t)$ is the position of the prey at time t. $\vec{D}$ represents a directional vector for wolves to follow as a function of the randomized vector $\vec{C}$ and the observed wolfs current position.

$$\vec{D} = |\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)| \tag{4}$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D} \tag{5}$$

$$\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a} \tag{6}$$

$$\vec{C} = 2 \cdot \vec{r}_2 \tag{7}$$

Of immediate note in this model is the fact that the terminal/prey position is already known. Such an optimization problem is already trivial and does need additional steps to be solved. It is at this point that the hierarchical behavior is instated, and exploration behavior is defined. Each agent will evaluate their cost and treat said cost as their distance to the unknown prey. The closest three wolves are ranked the α, β, and δ wolf respectively, and the modifications shown in Eq (8-10) are made to the directional vector $\vec{D}$ and position update $\vec{X}(t+1)$. These modifications exist under the assumption that a known cost function to be evaluated is available. If no such known cost function exists, such as in a SaR/SaD scenario, a new one must be made. This is addressed in the immediately subsequent section II-B-2. If there is not enough information to properly initiate an X_W, further modifications must be made, which can be found in section II-B-5.

$$\vec{D}_W = \left|\vec{C}_W \cdot \vec{X}_W - \vec{X}\right| \tag{8}$$

$$\vec{X}_{W_{new}} = \vec{X}_W - \vec{A} \cdot \vec{D}_W \tag{9}$$

$$\vec{X}(t+1) = \frac{\sum_{i=1}^{H} \vec{X}_i}{H} \tag{10}$$

In these equations, the subscript *W* represents the current observed members of the hierarchy and *H* represents the total number of relevant members to the hierarchy. In the base *GWO* algorithm, all the agents are entirely randomly initiated, which is not logistically doable in real world scenarios. The capability of functioning in any and all conditions is demonstrated throughout the results presented following, however a real world scenario would have all agents being released from designated initial positions, such as in standard *SaR* theory. Omega wolves are always omitted from the relevant hierarchy as their information is only relevant for themselves, and not for anyone else, whereas the other pack members influence the next position of the entire pack. This level of difference in importance further goes to emphasize the need for a constantly and immediately evaluable cost function. Using a standard hierarchy of one alpha, one beta, one delta, and any number of omegas, *H* will be three. This updates the position of each wolf based on the positions of the relevant solutions/wolves, which is presented in a visual manner in Fig. 5.

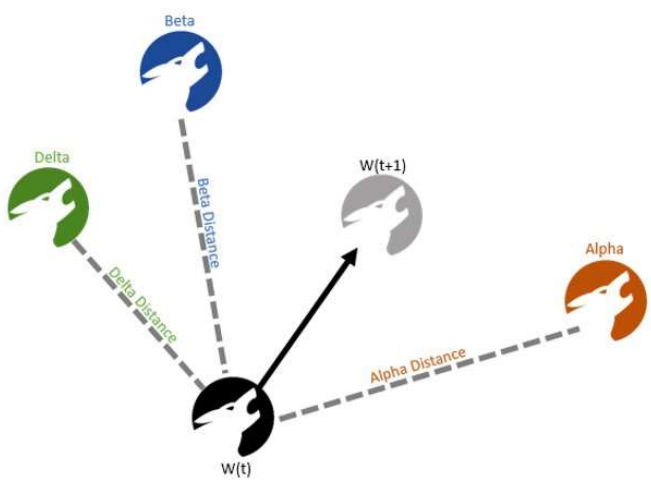

Figure 5 Visual Representation of GWO Progression.

The pseudocode presented in Fig. 6. is to help summarize the overall process in a more program-oriented manner, as this method has no physics or mathematical derivation to follow for better understanding. This pseudo-code also completely disregards the aforementioned issue of situations in which a cost function does not exist and will instead act as the base to build off of.

- Initiate w_n number of wolves with random values x_{w_n}
- Evaluate the cost of each wolf using cost function.
- IF more than H wolves have the same lowest cost
    - Randomly choose H of the lowest cost wolves to enter hierarchy
- END IF statement.
- The H wolves with the lowest cost are assigned to the hierarchy.
- FOR a = 2:0
    - Calculate directional vector $\vec{D}$ for each member of the hierarchy. (equation 8)
    - Calculate a new suggested position $\vec{X}_N$ (equation 9)
    - Update wolf position (equation 10)
    - Assign the H wolves with the lowest cost to the hierarchy.
    - IF more than H wolves have the same lowest cost
        - Randomly choose H of the lowest cost wolves to enter hierarchy
    - END IF statement.
    - Update time ($t = t + 1$).
- END FOR loop

Figure 6 *GWO* Pseudocode

2.2.2 Exploration Adaptations – Peuco Search

At this point, the major flaws present in the available built models is that of how to evaluate the cost of an agent in a scenario in which no prey or an unknown prey is present (i.e. no cost function), as well as the lack of dynamic constraints being applied to the agents. To remedy this, the mathematics and science behind *SaR* is applied to generate a pseudo-heat map of the environment supplement the meta-heuristic methods. This concept is largely inspired by the Seahawk *SaR* methodologies that can be seen in Navy and Coast Guard missions [2]. This method is being termed "Peuco Search" for the time being with Peuco being the Spanish word for Harris' Hawk.

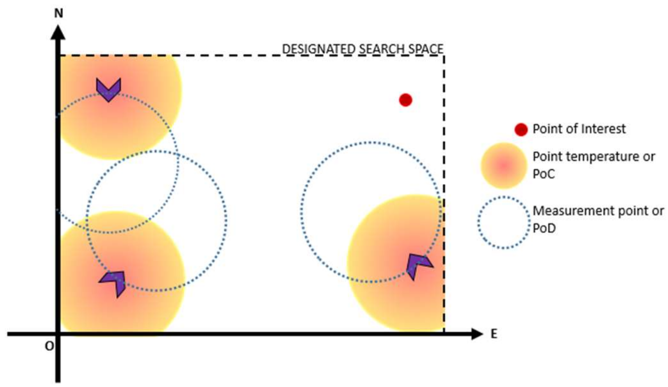

Figure 7 Visual Representation of SaR heat mapping.

Following basic thermodynamics/physics, Eq (25) represents a change in heat over time, In which q is the heat transfer coefficient, S is the surface area of the object being observed, c is the heat capacitance, and Ts and T are the surrounding temperature and the observed objects temperature.

$$\frac{dT}{dt} = \frac{qS}{c}(T_s - T) \tag{25}$$

With this, given an initial condition, temperature can be defined as a function of time. Since q and c are both unknown constants to us, we will replace them with a singular coefficient K. This K can thus be observed as a constant that directly control decay rate. Suggestions on what values to use for K can be found in section II-B-5. Additionally, time step dt will be replaced by a division of frequency 1/f to instead represent the systems update rate. The result of this can be seen in Eq (26).

$$T(t+1) = T(t) + \frac{KS}{f}(T_S - T) \tag{26}$$

Relating the thermodynamic equation presented in Eq. 26 to the concept of information tracking, the equation can be treated as a level of confidence in currently tracked knowledge (T), and how quickly one loses trust in said knowledge (KS/f) based on the knowledge it has of surrounding areas (T_S). The base goal of such a model is to have a variable value that can be attributed to this described level of knowledge of an area. With this model, as a specific location continues to be searched, it will increase in temperature, representing an increase in knowledge/certainty of what exists within the designated space. Conversely, as an area goes untouched by the pursuing agents, the heat will gradually decay over time, representing the decay of how relevant old knowledge is. On initiation, the search field will be generated with an initial minimum temperature of 0 (i.e. no knowledge). The collaborating agents will act as heat sources, warming areas (with no upper limit on temperature) within their area of effect. Each agent is instructed to seek out the coldest temperatures possible, to maintain as much knowledge of the total area as possible. The desired result of such a model is that the agents will search independently of one another while also prioritizing lesser known sweeps before returning to an already searched portion.

T can be defined as a matrix of temperatures consisting of the temperature measurements at every dx distance, where dx is any length we define based on how refined the desired mesh is. Next, a Gaussian filter is applied to matrix T to obtain the average of the surrounding temperatures, or Ts. Based on this, higher temperatures are equivalent to having more knowledge of that position, and lower temperatures as the lack thereof. However, the resultant modifications to this equation results in a dimensional output, which is not desirable as it can produced ill-conditioned results depending on the units used. As such, to change all parameters to be unitless for the purpose of algorithm scale, the area is normalized by the total area observed by the pack as shown in Eq (27), in which m represents the number of collaborative agents, and R being their vision/radar range. In other words, the total area S is replaced with ratio of the total search area to the amount of area that can be covered at any point in time by the pack. The remaining variable K thus becomes a tunable variable that defines the rate of decay of temperature, or how long a specified area can retain trust that the information is still accurate.

$$T(t+1) = T(t) + K\frac{S}{m\pi f R^2}(T_s(t) - T(t)) \tag{27}$$

In this work and in the later presented results, K is tuned such that the area sees an average decay rate of 0.3, which usually requires a K value of approximately 1 or 2 for a UAS that flies at 50ft/s. Systems that are significantly faster would want to use lower numbers to avoid near immediate decay. Alternatively, if prey information is available, K becomes the ratio of prey velocity to tracker velocity to scale decay rate based on the preys escape potential. As an example of this, if it is known that the prey move at three times the speed of the predators, K should be set to 3, resulting in rapid decay of information. Fig. 8 shows the MATLAB function peaks in its original form (left) and after 50 s of decay with no external heat source/searching agent (right). As one can see, the overall temperature progressively decays to zero as desired. The unit used for temperature in this work will be generically referred to as an $°T$.

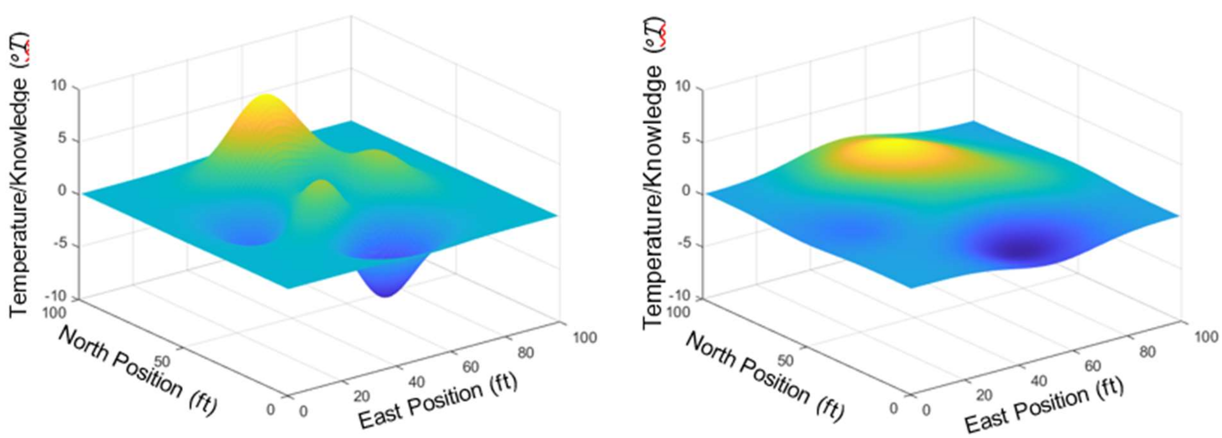

Figure 8 Peaks Undergoing Temperature Decay at K = 2.

However, this method lacks foresight in that it will only observe immediate results without considering future fallout. To further extend the reach of each individual and have them perform their missions cognitively, the gradient of the heat map is used in addition to the heat map to condition the trackers to consider not only observe the temperature of their observed target, but also to observe whether the point is leading the agent towards an increase in temperature or a decrease in temperature. Fig. 9 presents an example scenario to better demonstrate the importance of the gradient. These two points are determined as a function of radar range, R, and maximum turning radius ψ_{max} . The image on the left shows an agent exploring a 100 x 100 ft2 field in a position in which it is observing two points of equal temperature. However, one of the two points is located near a cold sink, whereas the other is surrounded by a flat temperature outside of the peak it exists on. Practically, it would be preferable for the agent to head in the direction of the cold sink(s) over the neutral field. By letting the agent also use the heat map gradient (shown on right), the agent observes the exact same points but can now anticipate that one point is in an area of declining temperature, whereas the other is in a mostly level field. This will lead the agent to head in the direction of the cold sink as desired.

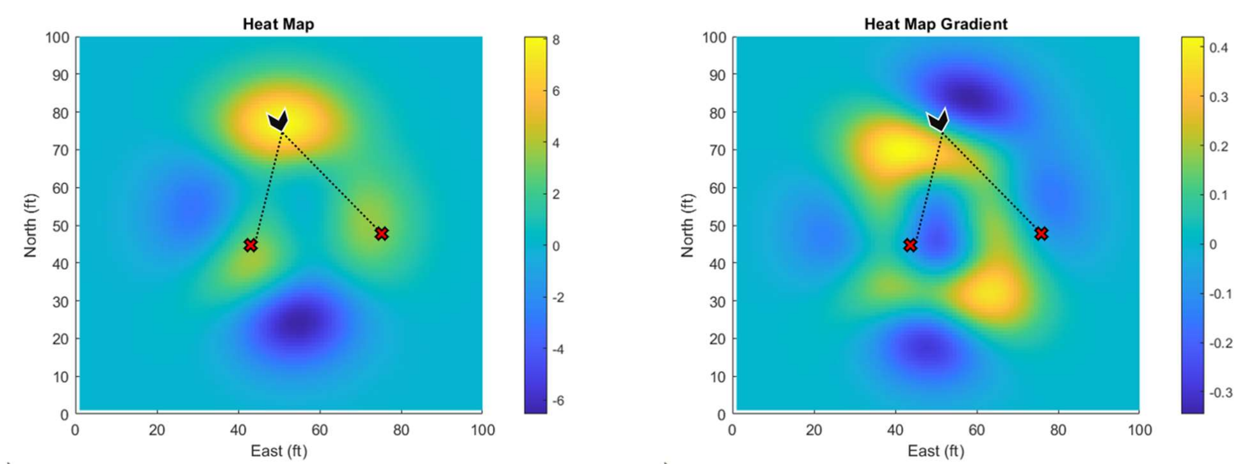

Figure 9 Visual Representation of Peuco Search heat mapping.

With this modification, a new cost function can be attributed to the system based on what the system has and has not tried. This cost function requires absolutely no prior knowledge outside of being given a designated search space it cannot leave. This method will be used throughout the remainder of this work, and X_p will be used to define the position of the prey if it is known or the position of a targeted exploration point as shown above.

The two observation points are defined as a function maximum turning rate as $\dot{\psi}_{max} = \frac{g}{V}\sin\phi_{max}$ and the amount of time the aircraft needs to cover a distance equivalent to R_{radar}, producing a cone of achievable positions.

2.2.3 Gray Wolf Adaptations

With the knowledge heat mapping providing a means of producing a cost function for a system that has none, *GWO* can be adjusted to properly function in its desired manner. The path-planning adaptation of *GWO* is straight forward conceptually. Each agent in the sky is a wolf and their cost are the distance between the observed UAS and the prey (if known). Alternatively, if multiple known prey exists only the closest is observed. In the scenario in which a prey is not known, an alternative/modified search algorithm is applied in the form of Peuco Search. This modified method is detailed in Fig. 10, in which the entire new modified algorithm is presented within a green area, and the old base algorithm presented in blue. The area portioned off in red is a situation that is not discussed in detail in the original work [31], in which there are a greater number of viable candidates for the hierarchy than there are available positions in the hierarchy. This issue is addressed using the gradient once again to determine which candidate has the most future potential. The base algorithm (blue) consists of initiating *W* number of wolves, randomizing their optimization parameters, and isolating the *H* most optimal solutions. This is expanded (green) to also function when no cost function is present through the use of *PS,* which also possesses dynamic constraints that the base (blue) did not. Additionally it intelligently accounts for the possibility of their being more optimal agents then there are optimal solutions by feeding said agents back into a *PS* loop. If absolutely no information is provided such that even a heat map cannot be generated, initial hierarchy will be assigned randomly, which is further explained in section II-B-5.

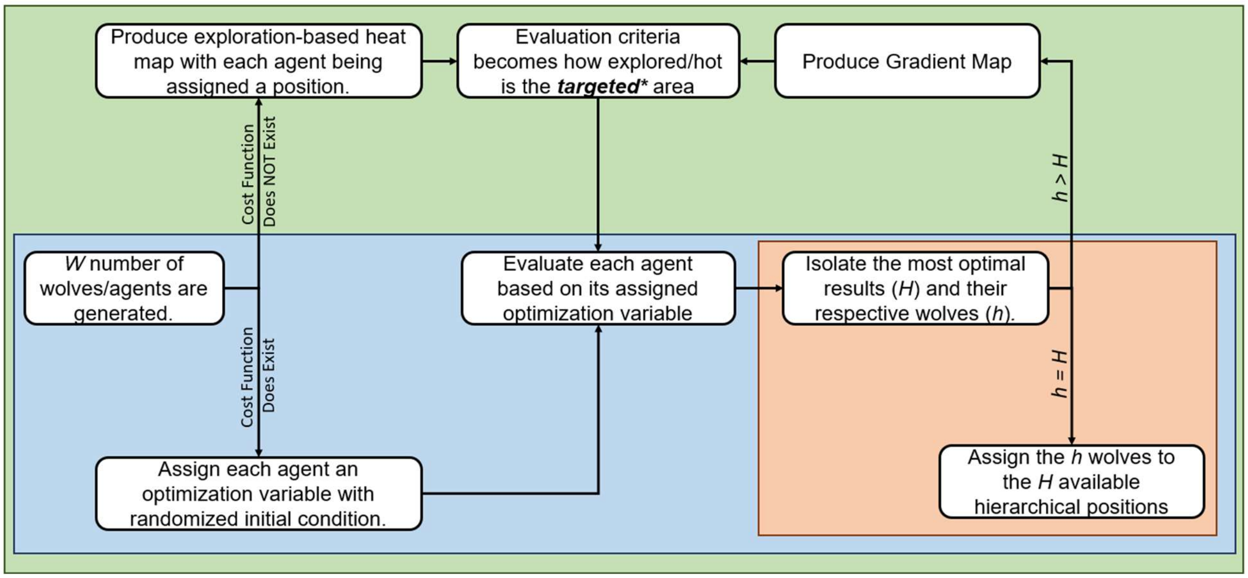

Figure 10 Gray Wolf Flowchart.

The primary flaw of a theoretical *GWO* path-planning (termed *GWSaR*) that Peuco Search resolves is that such an algorithm requires at least two agents to function in the intended manner otherwise *GWO* will update its position based on a single individual. If only one agent is used, it will always hold 100% of the weight resulting in it updating based on its own position with no reference. The lack of reference in such a method is equivalent to an entirely random method, removing all the intelligence from the metaheuristic. With Peuco search, the algorithm will no longer randomly decide its next position, but rather compare a range of nearby achievable points and adapt appropriately.

2.2.4 Harris' Hawk Optimization

HHO, as presented in reference [34], differentiates itself from *GWO* by forgoing the hierarchical structure which only uses the information of *H* out of *N* wolves and instead uses the information of every individual to make a consensus decision as opposed to a weighted decision based on a select few out of the pack deemed more important. This is done to mimic the hunting behavior of Harris' Hawks, which work as a cohesive pack alternating rolls between spotters (exploration) and hunters (exploitation) to minimize the total effort exerted by each member. This presents itself as a more democratic option when compared to a authoritarian search such as *GWO*, resulting in higher performance.

The exploration phase of *HHO* follows the model presented in Eq (28) that is similar to *GWO* in terms of complexity, observing the position of every hawk $\vec{Y}_j(t)$, with subscript j representing the current observed hawk, at time t and updating its position within the defined desired search area bounds *LB* and *UB*, where *LB* represents the furthest south-west constraint and *UB* the furthest north-east (assuming North and East as positive). The subscripts z, p, and m represent a random predator pack member, the prey, and the centroid of the pack, respectively. This model is dependent on the random variables k_1, k_2, k_3, and k_4 with range (0,1), similar to the r_1 and r_2 variables of *GWO*. Once again, these ranges are given no mathematical or logistical explanation, only proof that they work when used in a virtual setting. Such a range would allow for results akin to that of fuzzy logic, in which the output is a full range of possibilities (i.e. 0, 1, and all numbers between the two) as opposed to binary results (0,1). This already presents one major issue with the exploration portion of *HHO*. The second can be observed in Eq (28) in which one of the two definitions of $\vec{Y}(t)$ being independent of its previous state, presenting the possibility of sudden jumps, which may not be problematic in virtual scenarios such as computer science, they become impossible (and dangerous to even attempt) for a physical system such as an aircraft with dynamic constraints.

The final concern with the presented model is the reliance on a known prey position in the same manner as *GWO*. The major improvement in *HHO* over *GWO* is that not only the best hawks are observed for updating individual hawk positions, but also the centroid of the pack $\vec{Y}_m$ for consensus hunt, and other random hawks $\vec{Y}_z$ for smaller group hunts which can be seen in both its exploration and exploitation. However, the exploration behavior is based on the perching parameter *Q*, being randomized every time step with range (0,1) which suffers the same downfall as prior randomized variables defined. All these issues can be resolved by using Peuco Search over the base exploration. As such Harris' Hawk exploration is not used and is replaced with *PS* while preserving exploitation. With the use of *PS*, initial prey position can be replaced with the observed coordinate of the heat map if desired, or exploration can be maintained until a known prey is identified.

$$\vec{Y}_j(t+1) = \begin{cases} Y_z - k_1\left|Y_z - 2k_2Y_j(t)\right|, if\ Q \geq 0.5 \\ \left(Y_p(t) - Y_m(t)\right) - k_3\left(LB + k_4(UB - LB)\right),\ \ if\ Q < 0.5 \end{cases} \tag{28}$$

Based on this exploration phase, it is arguable whether *HHO* provides any significant improvement over *GWO*. However, the spotter behavior seen in Harris' Hawks is not what defines them as such excellent hunters, but rather their hunting packs ability to manipulate their prey into favorable positions, i.e. the exploitation behavior is far more significant than the exploration for the designed scenario. This behavior is modeled with a dependency on the prey's energy levels as seen in Eq (29) and is used in the *HHO* exploitation behavior. τ and t represent the total hunt time and current time, respectively. Total hunt time can be assigned as a designated design parameter.

$$E = 2E_0(1 - \frac{t}{\tau}) \tag{29}$$

In this model E is the current unitless energy of the prey, with E_0 being the initial energy which randomly changes in the interval (-1,1). For the scenarios presented in [34] negative and positive energy are treated as the same (i.e. E is an even function), and no physical meaning is given to the variable. Instead of using this randomized nature entirely new means of relating system dynamics to previously random energy are developed. These novel developments are presented in Eqs (30-35). The first of these changes is the system energy is now modeled as a parallel to the concept of physical energy, presented in Eq (30).

$$E = \frac{1}{2}\bar{m}\bar{V}^2 + \bar{m}\bar{g}\bar{h} \tag{30}$$

The aim of such an equation is to provide relevant physical meaning to each variable while also preserving the fact that E has no units. The lack of dimensionality in this parameter is to avoid any constraints that are accompanied with high dimension variables, such as the dimensionality constraints seen in Hungarian Assignment. Furthermore, the random nature of E0 is best abandoned once dynamic constraints have been applied and instead be treated as zero for the first time step after discovery of the prey. This work seeks to characterize the system as a function of known and system defining parameters for determining success rate of capture. These parameters are velocity, number of members, system area of effect, and distance between predator and prey. The final chosen representative weightings to encompass these parameters for a moving prey are shown in Eq (31). The energy of a stationary prey (Vp = 0) is treated as zero since it has no means of escaping. Note this applies to static points such as waypoints as well.

$$\bar{m} = \frac{\eta_p}{\eta_u}, \;\; \bar{V} = \frac{V_p}{V_{pm}}, \;\; \bar{g} = \frac{V_{pm}}{V_{pmax}}, \bar{h} = \frac{d_{pu}}{R_{radar}} \tag{31}$$

In this model, η represents the number of members of a party (u for predator and p for prey) with no apriori information, V_{pm} and V_{pmax} represent the prey mean (cruise) velocity and max velocity respectively. The variables d_{pu} and R_{radar} represent the distance between predator and prey. The result of this model is a characterization of prey as a function of its numbers (mass), how much it is currently exerting itself beyond its norm (velocity), how much it exerts itself on average with respect to its known maximum limit (gravity), and a distance upon which gravity will act (height). On initiation, all of these variables, as well as E and e are kept at zero and are only updated with the discovery of new information on the prey. This is discussed in further detail in section II-B-5.

Based on this model, the prey's energy will decay to exhaustion as time passes. Beyond this modeling of prey characteristics, *HHO* also models the preys escape attempts with the variable e, which randomly changes in the range (0,1), which must again be given a non-heuristic physical definition.

Escape potential, e, is defined as a function of the differences in capabilities between trackers (subscript u) and prey (subscript p). In an attempt to relate the concept of escape potential to the maneuverability of a fixed-wing UAS, the ability to accelerate and perform sharper turns is given significant weighting. These weights should be based on the dynamic constraints of the aircraft and its propulsion system. To do this, the capabilities to be modeled are again system velocity and distance in a similar manner to energy, but also with system maneuverability (ϕ_{max}) taken into consideration. This is modeled using Eq (32), in which $\bar{M}$, $\bar{U}$, $\overline{U_s}$ and $\bar{E}$ represent maneuverability comparison, velocity comparison, velocity weighting and error weighting respectively.

$$e = \bar{M} \cdot \bar{U} + \overline{U_s} + \bar{E} \cdot \bar{U} \qquad (32)$$

The primary goal of this model is to provide a bounded and weighted means of comparing the predator and prey in a relative manner. As the current algorithm treats all individuals as fixed-wing UAS, maneuverability is defined as a systems ϕ_{max} as seen in Eq (33).

$$\overline{M} = \frac{\phi_{\mathrm{maxp}} - \phi_{maxt}}{\phi_{maxt}} \quad (33)$$

The velocity comparison can also be modeled in a similar manner as shown in Eq (34).

$$\overline{U} = \frac{V_p - V_\tau}{V_\tau} \quad (34)$$

To add realistic weighting to the system, a velocity weighting is applied to each term that does not have a velocity dependency, as velocity will always be one of the largest magnitudes. Finally, error must also be considered in the same fashion as in the energy model as represented in Eq (35).

$$\overline{U_s} = \frac{V_p}{V_\tau}, \quad \overline{E} = \overline{h} = \frac{d_{pu}}{R_{radar}} \quad (35)$$

Based on these prey characteristics, the hawks will enter one of four complex exploitation behaviors.

2.2.4.1 Soft Besiege

When $e \geq 0.5$ and $|E| \geq 0.5$ the hunters perform a low energy cost encircling of the prey to exhaust it further than its current high-escape potential. This is modeled by Eq (36) in which $\Delta Y(t)$ is the distance between the observed hawk and the prey at time *t* and *J* is a random variable in range (-2,2) modeling the preys ability to perform sudden escape maneuvers (e.g. large jumps). This *J* parameter also has no mathematical explanation on its range provided, and none were developed by the author of this work. Future contributions to this work would benefit from defining this heuristic variable, possibly as a function of preys maximum acceleration to imitate the concept of a rabbits jumping escape maneuvers.

$$Y_j(t+1) = \Delta Y_j(t) - E\left|JY_p(t) - Y_j(t)\right| \quad (36)$$

This behavior is intended to represent the conservation of energy in situations in which the observed prey still has enough energy and options for escape. Rather than expending energy in such scenarios, the hawks will focus more on exhausting the prey over trying to capture while the prey is at its strongest. Furthermore, it is possible that if the hunters attempt to immediately converge onto the prey they risk the prey performing an escape maneuver, thus putting the hunters in a more difficult position than originally.

2.2.4.2 Hard Besiege

When $e \geq 0.5$ and $|E| < 0.5$, the prey has been exhausted to the point that it can be assailed, however it still has potential for escape. In this situation, the hawks will perform sudden unexpected dives onto the prey to catch it off guard. This is modeled in Eq (37).

$$Y(t+1) = Y_p(t) - E|\Delta Y(t)| \tag{37}$$

This behavior is intended to represent the increase after they have exhausted a prey through a soft besiege and now begin to bait the prey into an unfavorable position. Such maneuvers allow the hunters to gain a level of control over the situation now that the prey has had further resources exhausted while also not fully committing to any maneuvers that could lose the hunters said control in situations in which the prey attempts an escape in an unfavorable direction.

2.2.4.3 Soft Besiege with Progressive Rapid Dives

When $e < 0.5$ and $|E| \geq 0.5$, the hunt has entered a "it can run but it can't hide" scenario in which the prey has a large amount of energy, but has no means of reliable escape. At this point the hawks perform a soft besiege to conserve energy while also performing frequent dives onto the prey to capture it, as modeled in Eqs (38-40). The Levy Function, *LF,* is a value calculated using a 1x2 vector of random values with range [0,1] use a series of complex equations which can be found in [6]. However, going through and calculating *LF* will reveal that no matter what value within the range of [0,1] is chosen, the result will always be significantly smaller than one.

Since this algorithm will be applied using real world measurements with much large magnitudes (e.g. ft, m, mi, etc.) the final value of Eq (39) will be nearly identical to the result of Eq (38). As such, Eq 38 becomes unnecessary and only Eq (38) is used.

$$Y_{j_A}(t+1) = Y_p(t) - E\left|JY_p(t) - Y_j(t)\right| \tag{38}$$

$$Y_{j_B}(t+1) = Y_j(t) + S \times LF \tag{39}$$

$$Y_j(t+1) = \begin{cases} Y_{j_A}(t+1)\ if\ Y_{j_A}(t+1) < Y_{j_B}(t+1) \\ Y_{j_B}(t+1)\ if\ Y_{j_B}(t+1) < Y_{j_A}(t+1) \end{cases} \tag{40}$$

Comparing this Soft Besiege with Progressive Rapid Dives behavior with the standard Soft Besiege behavior, the only difference to appear is that the $\Delta Y_j(t)$ is replaced with $Y_p(t)$, shifting the focus from encircling a prey to converging onto the prey. This is intended to represent the conservation of energy seen in Soft Besiege behavior to account for the prey having a large amount of energy, while also taking advantage of the prey not having a reliable escape route.

2.2.4.4 Hard Besiege with Progressive Rapid Dives

Finally, in scenarios in which $|E| < 0.5$ and $e < 0.5$ the prey is considered to have next to no chance at escaping the hunters. At this point a hard besiege is performed in tandem with a series of rapid dives to exploit the scenario and end the hunt. This follows the same modeling presented in previously in Eq (38). As such the model based on *LF* is dropped for the same reason and only Eq (41) is observed.

$$Y_j = Y_p(t) - E\left|JY_p(t) - Y_m(t)\right| \tag{41}$$

This behavior is highly aggressive, essentially consisting of the entire pack attempting to rapidly converge on the target. In a real-world physical scenario this will likely put the unmanned systems at risk of colliding. As such, this behavior is likely best applied in a sequential manner. That is, apply the behavior to one system at a time, and do not apply the behavior to a different agent until the first has completed its dive, similar to how the hawks would take turns in rapid succession and not all at once so as to avoid collision. Despite this level of aggression, the systems relation to observation points defined as a function of maximum turning rate ($\dot{\psi}_{max} = \frac{g}{V}\sin\phi_{max}$) guarantees the these points are feasible for the aircraft to achieve.

2.2.5 Meta-Heuristic Initiation

One of the major issues with the presented meta-heuristic methods is their heavy reliance on apriori information, which this work aims to remain independent from. To remedy this the following series of modifications were made.

2.2.5.1 Exploration Initiation – Heat Mapping

To decouple exploration from the necessity of information on the area, the existence of prey and their positions, *PS* is designed to replace both *GWO* and *HHO* exploration. The only mandatory input to *PS* is to provide an area or search space for the system to explore. Beyond this, all other system inputs may be considered tunable parameters. The decay rate *K* should be a function of predator radar range and search area size. Some examples of standard ranges to area size ratios would be a search space of 2000 x 2000 ft^2 for a radar range of 100 ft, or 5000 x 5000 ft^2 for a radar range of 500 ft. Both examples would typically use a *K* value within a range of (1,3) depending on how confident the user is on how much to trust older information, with 1 representing high confidence, and 3 representing low confidence.

For computer science applications, the initial agent positions can be entirely randomized within the search space. However, this is not feasible for physics constrained systems. In the case of UAS agents, it is preferable to initiate the algorithm near a corner solely as it will result in the immediate coverage of a corner, which the algorithm seems to prioritize the least and is often the last region to be explored. This is however only a suggestion, as the algorithm has been designed and tested such that the system can be successful at all initial conditions.

For *GWO*, if the algorithm is initiated in such a manner that there is no reasonable means of obtaining the desired *H* agents to place within the hierarchy, then the hierarchy should be randomized to provide an initial condition. The system should appropriately adjust the hierarchy in the following time step.

2.2.5.2 Exploitation Initiation

Since exploitation only occurs when a known prey position exists, there is no need add additional directions for such cases as was done for exploration. *HHSaR* however, needs not only prey position but also certain prey states, which cannot be ascertained with only a single known position. To account for this, the prey is assumed to be stationary with an *E* and *e* of zero until sufficient prey positions over time are found and states such velocity and bank angle can be calculated.

2.3 L_N Lateral Guidance

The following section on lateral directional guidance is a full development of the L_N guidance law, which can also be found in reference [7]. The full conference document from which the patent is based can be found in the Appendix.

The L_N guidance law is built off the concept L_1 of using a reference point that is a constant distance L from the system, projected onto the desired path. An acceleration command is calculated using this information based on Eq. (42), in which η represents the angular error between the waypoint line and the aircraft heading angle and V_{T_i} is the system total velocity.

This acceleration command is then converted into a roll angle command through the relation between centripetal force and turn radius (L_1). With this a lateral guidance command that can be fed to the controller is obtained. A geometric representation of this concept can be found on the left in Fig. 11 provided by reference [47].

The use of an N number of reference points for lateral guidance was developed as a means of accounting for both the overshoot issues present in L_1 and the introduction of complicating variables in L_2^+ as developed in reference [7]. L_N operates as series of N many L_1 guidances being run in parallel with different L_1 lengths, and then taking the average command of all outputs as opposed to having a direct feed to the output. This allows smooth transitions with little to no overshoot in tighter turns without the need for additional tuning parameters in the form of lead times. A visual representation of this can be seen in Fig. 11.

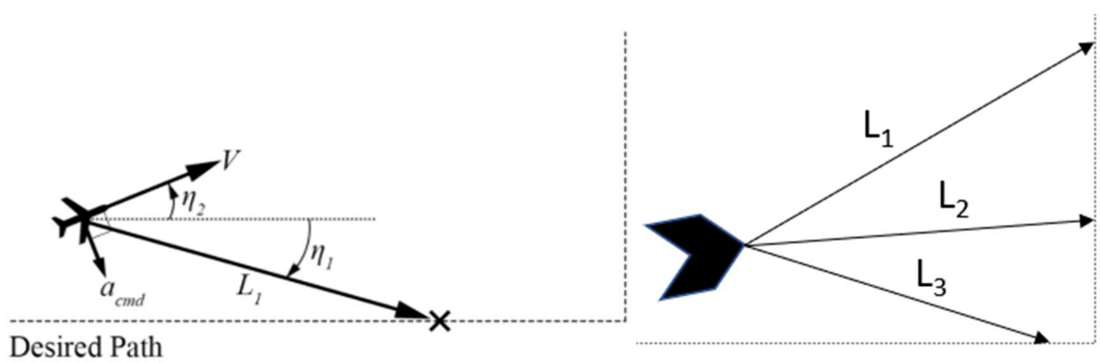

Figure 11 Visual Representation of L_1 from Reference [47] versus LN guidance where N = 3.

L_N generates the additional $N - 1$ reference lines by taking the input line and proportionally increasing and decreasing its lengths until all desired lines are produced. Each of these lines will have their own guidance command generated as derived in reference [47] and shown in Eq (42). These lines then have their guidance commands averaged to produce a command with information about both near and far points along the desired trajectory. Despite the larger a_{cmd} generated from the longer line, the system will never actually respond to such a large command, as it will be proportionally offset by the smaller a_{cmd} generated by the shorter lines. This allows for the longer line(s) to provide information on necessary future actions, and shorter line(s) to provide information on necessary actions for quality tracking. This can be seen as a behavior similar to the lead time used in L_2^+ guidance, without any of the necessary complexities. The smooth transition of reference points moving from one line to the next allows for smooth transitions between lines with next to zero overshoot.

$$a_{cmd_i} = 2\frac{V_T^2}{L_i}\sin(\eta) \tag{42}$$

2.4 Θ_1 Longitudinal Guidance

The following section on longitudinal guidance is a summary and extension of the theory described in [8]. The full conference document from which the patent is based can be found in Appendix A.

Starting from the z-direction equation of motion in the inertial frame, as seen in (43), and then using perturbation theory to create perturbed model about a steady state condition, onto which trigonometric identities are applied alongside the small angle assumption to produce Eq (44). Worth noting is that the assumption of a small β is also made, however this is a less significant assumption as it is rare for smaller UAS to exceed six or seven degrees in standard conditions.

$$\dot{H} = \sin(\Theta)\,U - \cos(\Theta)\sin(\Phi)\,V - \cos(\Theta)\cos(\Phi)\,W \tag{43}$$

$$-\dot{h} = u\sin(\Theta_1) - U_1\theta\cos(\Theta_1) \tag{44}$$

From (30), it is possible to isolate a desired command state (such as u or θ), as shown in Eq (45) and Eq (46), in which θ and u represent perturbed pitch angle and forward body velocity respectively, and U_1 and Θ_1 represent trim pitch angle and velocity respectively.

$$\theta_{cmd} = \frac{\dot{h} + u\sin(\Theta_1)}{U_1\cos(\Theta_1)} \tag{45}$$

$$u_{cmd} = \frac{-\dot{h} + U_1\theta\cos(\theta_1)}{\sin(\Theta_1)} \tag{46}$$

Assuming $\dot{h}$ can be assigned a constant value (i.e. maximum ascent/descent rate), the output of Eq (45) can be used treated as an altitude hold guidance. Rather than directly deciding upon an ascent/descent rate, the desired $\dot{h}$ is instead calculated as in Eq (47), in which Γ is vertical flight path angle.

$$\dot{h} = U_1 \sin\Gamma \tag{47}$$

At this point, the guidance is further augmented with the purpose of intelligently choosing an ascent/descent rate as well as modifying it dynamically if needed. The first step in doing this is deciding upon a distance over which to damp the ascent/descent rate over. This distance is defined in Eq (48), in which Δt_0 is the desired minimum amount of time required for the aircraft to transition from maximum ascent rate, $\dot{h}$, to an altitude hold (i.e. inertial z-deceleration). That is as Δt_0 increases, the aircraft will take longer to decelerate as well as it will start the deceleration process appropriately earlier. Inversely, as Δt_0 decreases, the aircraft will attempt to go from max ascent/descent to an altitude hold in a short amount of time and over a smaller distance. In addition to this an additional altitude damping, which is defined in Eq (49), is developed to saturate the incoming value as the agent approaches the desired altitude, as is modeled in Eq (50).

$$d_{damp} = |\dot{h}| \cdot \Delta t_0 \tag{48}$$

$$h_{damp} = \frac{|h_{desired} - h_{measured}|}{d_{damp}} \tag{49}$$

$$\dot{h}_{desired} = |\dot{h}_{desired}| \cdot sign(h_{desired} - h_{measured})h_{damp} \tag{50}$$

In other words, when the aircraft is a distance d_{damp} away from the desired altitude, the ascent rate does not change. However as it approaches the desired altitude, the rate is progressively decreased to 0. This allows us to isolate the first term of the guidance laws numerator $(u \cdot sin(\Theta_1))$ from the second term ($\dot{h}$), allowing us to control a desired ascent/descent rate without altering the velocity portion of the guidance. The significance of this is that if the aircraft is flying at the desired altitude and undergoes a change in velocity:

- $u \uparrow \rightarrow \theta_{cmd} \uparrow \rightarrow \theta \uparrow \rightarrow u \downarrow$
- $u \downarrow \rightarrow \theta_{cmd} \downarrow \rightarrow \theta \downarrow \rightarrow u \uparrow$

The guidance implicitly improves an aircraft's longitudinal stability (assuming an already stable system), in that it will alter the pitch command to take advantage of the coupling between u and θ to damp u back to zero when the desired altitude is already met. In other words, the guidance law works not only to meet a certain altitude requirement, but to also meet trim velocity when the desired altitude is achieved.

3 UAS Testbed

The results presented subsequently in this work all use a SkyHunter UAS model for simulation developed in [22] at the KUFRL. Currently the SkyHunter UAS is one of the baseline system on which KUFRL experiments are conducted. The aircraft consists a foam fuselage, wing set and empennage, which are supported using carbon fiber rods. The aircraft is 4.9 ft long with a 13 ft wingspan. The system is flown at a variety of weight settings, however the base aircraft with all avionics accounted for has a mass of 4.4 kg. The system cruises at 50 ft/s and has a supposed stall speed of 29.8 ft/s. Six servos are used to control the elevator, two ailerons, and two rudders, with a final servo coupling the rudder to the nose wheel, all of which are controlled via an electronic speed controller (ESC). The six degrees of freedom flight simulator uses a physics based dynamic model developed using Advanced Aircraft Analysis (AAA) [48]. An image of this testbed undergoing inhouse testing and simulation can be found in Fig. 12.

Figure 12 Skyhunter UAS Advanced Simulation [50].

4 Results and Discussion

4.1 Meta-Heuristic Path-Planning

4.1.1 GWSaR

GWSaR was simulated for three agents with a hierarchy of three tiers. This is done so that the system can first be tested in ideal situations where all information is used. Example results for *GWSaR* with heat mapping (but not the full *PS* algorithm) can be found in Fig. 13. Where the major flaws of attempting to use *GWO* as a path-planner. The first major issue is that all three agents quickly convene to a uniform direction, which then proceeds to dominate the directional portion of the *GWO* algorithm. This results in the equivalent of all agents searching a field in a poorly maintained formation. However, this is an exploration issue, meaning that the developed *PS* alternative should help remedy this issue. Yet, *PS* is not designed to exist during exploitation, which as one can see in Fig. 13. Also has some major issues. First and foremost, it becomes apparent that *GWSaR* does not have boundary constraints during exploitation. Beyond this issue, the encircling behavior modeled in the *GWO* and *GWSaR* can be seen occurring, not as an expected loiter type shape, but instead as large locks or petals branching out from the prey.

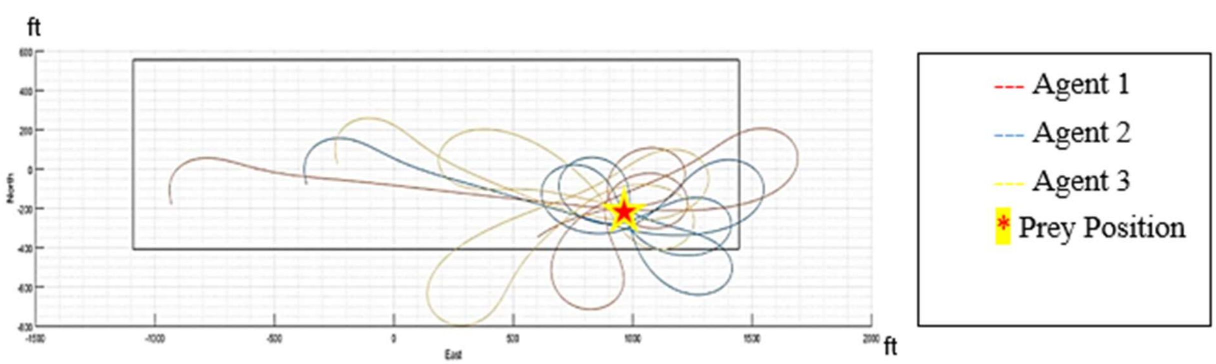

Figure 13 Example results of GWSaR.

Considering the issues present in both exploration and exploitation, it is necessary to make further modifications to the *GWSaR* algorithm to better transition major concepts from the original computer science based theory to a physically constrained aircraft model.

4.1.2 HHSaR + Peuco Search.

The Peuco Search heat map performed as expected, producing a function that can be evaluated as a function of position. Fig. 14 shows the base heat map being applied to a system moving at 50ft/s and following a outward square spiral path. This is assuming a default gain *K* of 1. Fig 15 shows the exact same scenario, however now *K* has been tripled to 3. These two images show the algorithms ability to define how much past information is trusted as time goes on. A low *K* suggests that very little decay happens over time and that all collected data can still be trusted, whereas a higher *K* suggests that older data becomes less reliable over time, as can be seen by the significant decay occurring at the origin of the path.

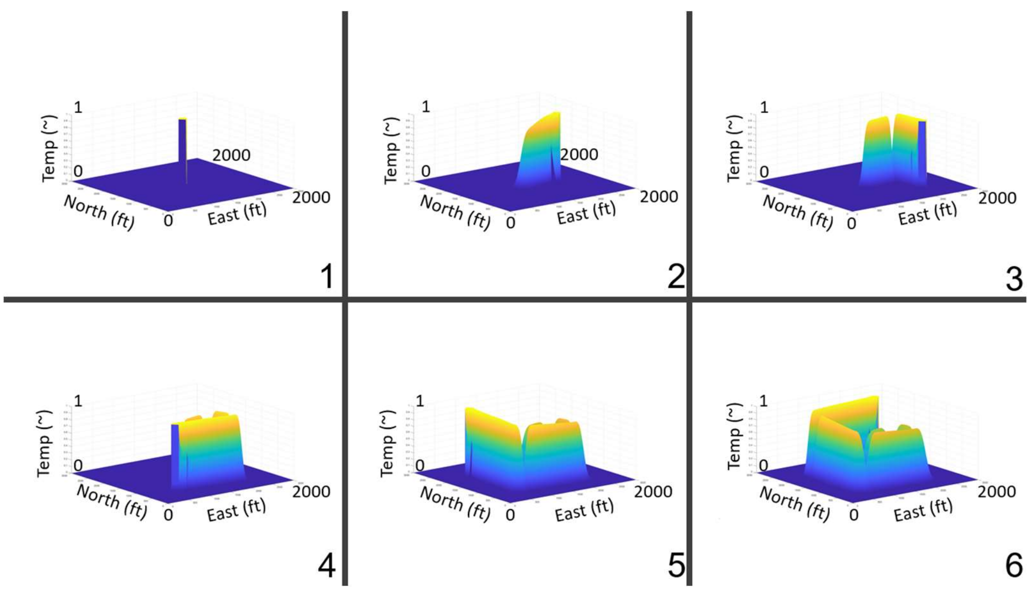

Figure 14 Example of Heat Mapping With Low Decay (K = 0.8)

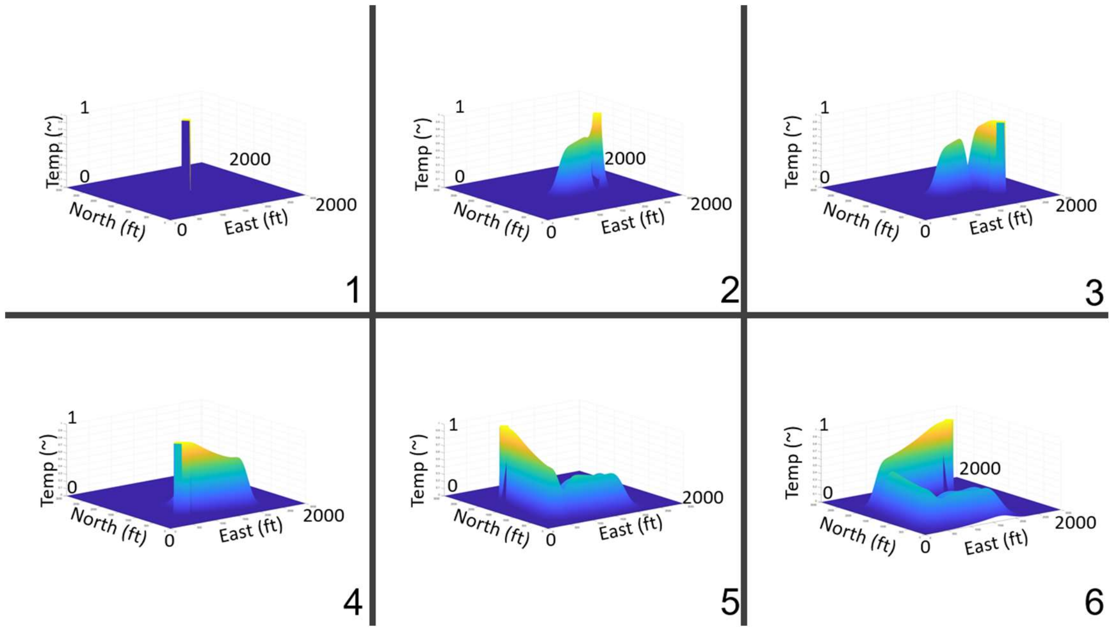

Figure 15 Example of Heat Mapping With High Decay (K = 2.8)

HHSaR faired significantly better than *GWSaR* with the modifications that were made. To have a benchmark to compare this work to, a standard SaR grid search is initiated in parallel with every HHSaR search. One example of such a search can be seen depicted in Fig 16.

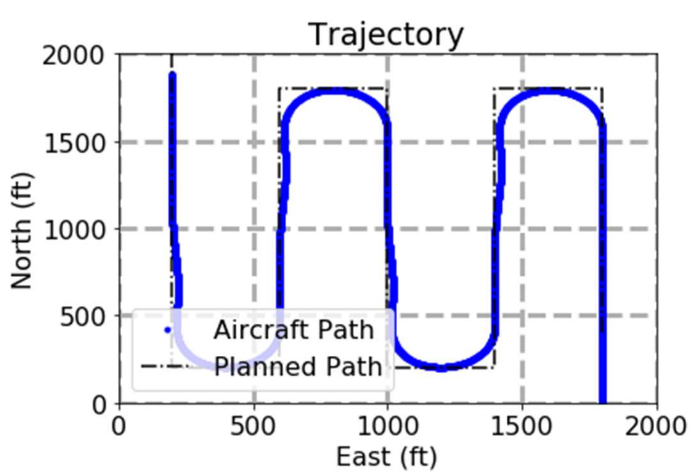

Figure 16 Example results of GWSaR.

Fig. 17 depicts a typical example simulation of the final algorithm using a single agent. This agent has an R_{radar} of 100 ft to search a 2000 ft by 2000 ft field at a cruise velocity of 50 ft/s. The radar size is taken to be so small to account for the smaller field size. In this figure the aircraft trajectory (blue line) is plotted from its initial condition (red '×') until the point it has detected the prey (black '×'). The algorithm tends to opt to search from the outer most reaches and then work its way inwards if the field is not exceptionally large with respect to the system velocity (e.g. a 10,000 x 10,000 ft field for a single UAS traveling at 50ft/s). In the presented scenario, the HHSaR system was able to find the prey in 85 s. When given the same condition and being told to search in a standard method (see Fig. 1) from east to west, a standard SaR procedure found the prey in 136 s. The maximum amount of time that a standard SaR in such a scenario could have needed to find the OoI is 184 s.

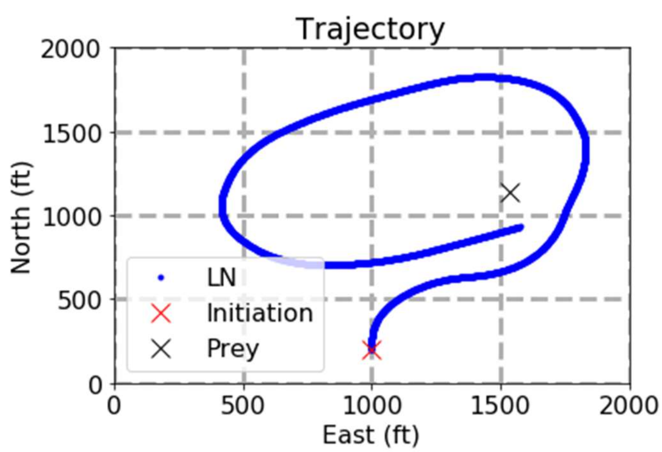

Figure 17 HHSaR Example Simulation.

The agent consistently remains within the defined bounds of the field while also not returning to already explored locations and moving in the direction of unexplored regions over attempting to cover smaller portions that were missed. In terms of success rate, out of 108 trials on the 3000 ft by 3000 ft field, each with randomized initial conditions for both the OoI and the searcher, only one failed (0.9% fail rate). This fail condition was a result of the OoI being in nearly on top of one of the corners of the boundary, resulting in the searcher to be less likely to search such a small portion. The fail criterion for this field type was if the searcher could not find the OoI within 2000 seconds. It is possible if given enough time that the searcher would find the OoI, however such inefficient results are not suited for the task at hand. The probability distribution is presented in Fig. 18 in histogram form. Each bin for this histogram contains all the scenarios that completed the mission in a specific one minute time frame (i.e. bin 1 contains all scenarios that were completed in a range of 0-60 s, bin 2 contains all scenarios that were completed in a range of 60-120 s, etc. with the last bin containing any scenario that took 900 s or more to complete. The red line delineates the amount of time a standard SaR patrol would take to find an OoI in the worst possible scenario (i.e. the PoI is in the last region they explore). This time was simulated to be 292 seconds for a 3000 ft by 3000 ft field. It is important to note that this calculation is treating the search party as a point that can move at the same velocity as the UAS and has the same visibility range as the UAS, both of which are optimistic assumptions. With that said it remains an important benchmark, and as such Fig. 19 presents the same results in histogram form but with only two bins, one for results that were faster than the standard search party, and the other for the results that were not.

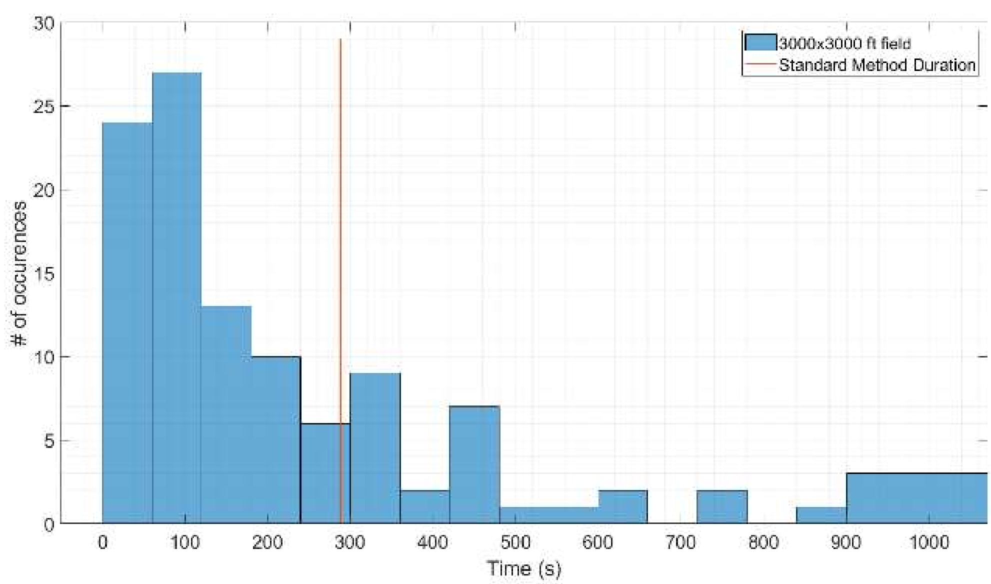

Figure 18 Histogram of number of successful searches as a function of time.

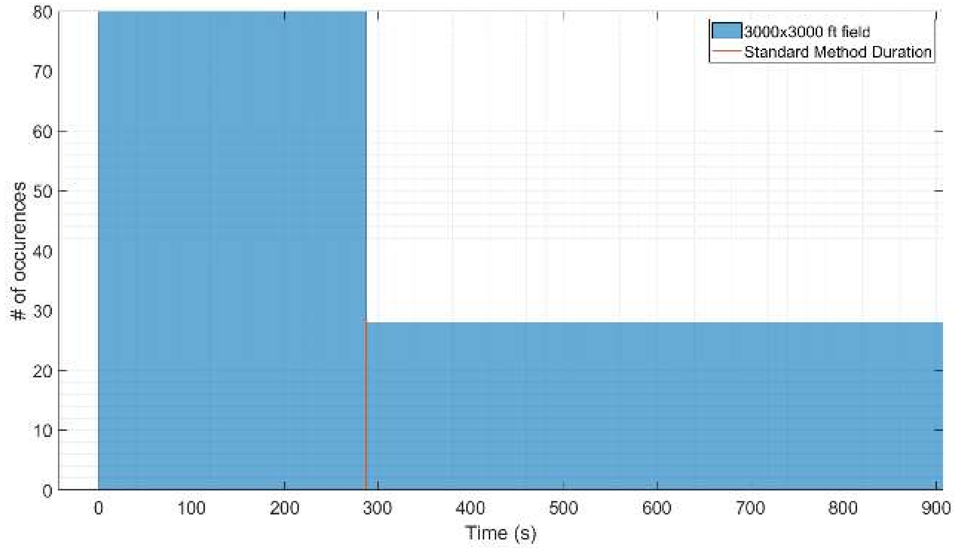

Figure 19 Histogram of number of acceptable searches as a function of time.

The data presented thus far has been solely for missions with singular hawks/agents. Fig. 20 depicts a scenario with two agents searching for an OoI that is just barely outside of radar range (100 ft) from their initial positions, while also being initiated oriented such that they will not immediately find it. In theory such a scenario should take by far the longest, since it requires the system to return to an area that is adjacent to an explored region. However, the agents are still able to find the OoI within 63 s, presenting a significant improvement over a single agent. Two agent missions tend to follow a rather specific trend, which is overlaid on Fig. 20. The two agents will start by moving away from one another to stay away from the others heat source. Then will start searching closer to the borders, moving in opposite directions (clockwise/counterclockwise) until they once again begin to approach one another. At this point they will both once again prioritize staying apart, resulting in both veering off to explore the inner regions of their lap until they decide to focus on searching near the borders once again, repeating the process until an OoI is found.

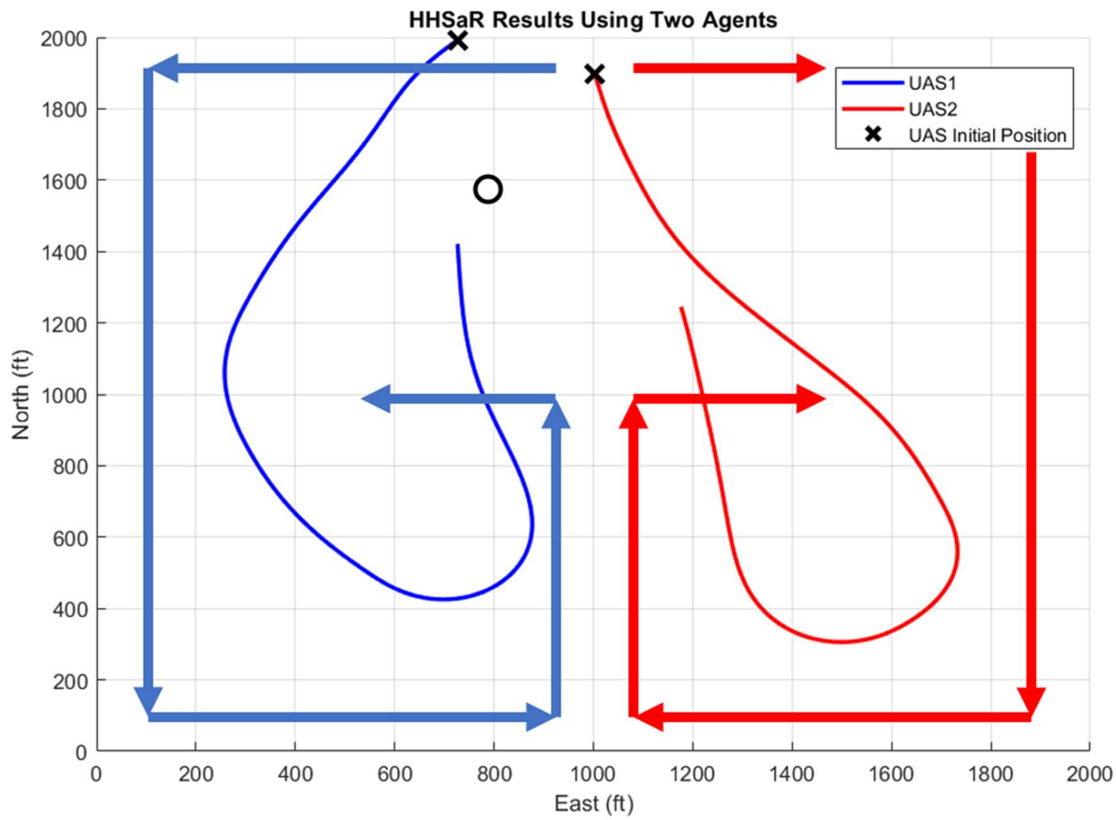

Figure 20 Depiction of HHSaR Using Two Agents

A similar scenario is then tested for three agents, with the OoI being place behind all three agents. The agents were able to find the prey in 46 s. Based on the prior scenarios, one agent completed a difficult mission in 85 s, two agents completed a difficult mission in 63 s, and three agents completed a difficult mission within 46 s. These results may suggest an inverse linear relationship between number of agents and time of a successful search.

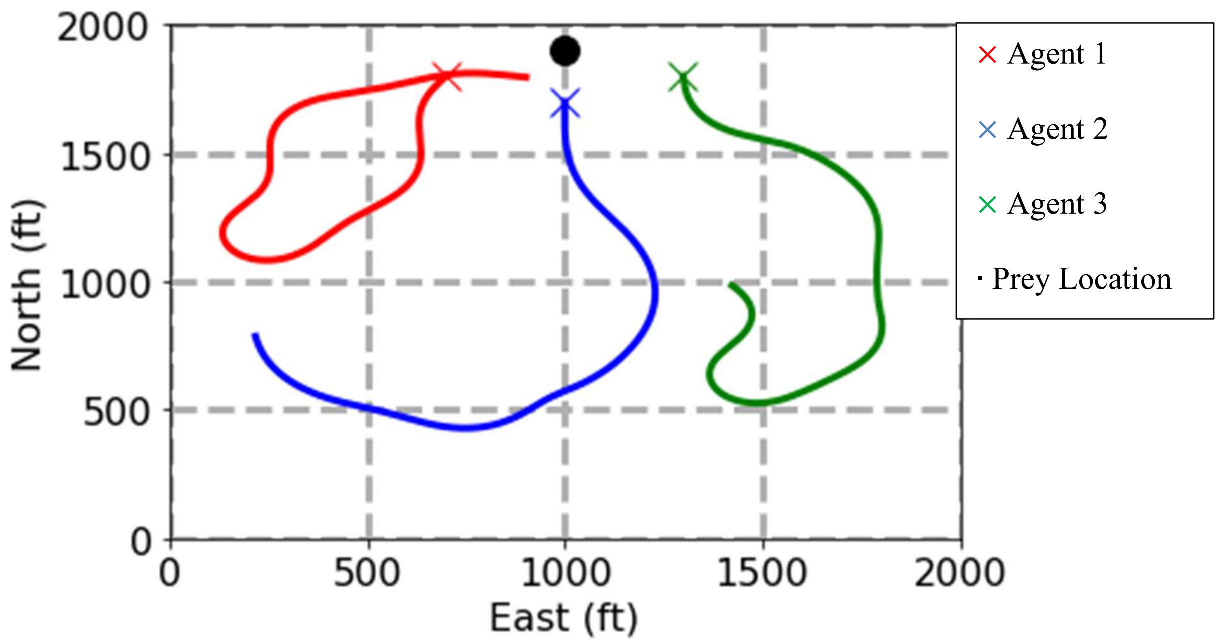

Figure 21 Depiction of HHSaR with Three Agents

In terms of exploitation, *HHSaR* also shows improved performance over *GWSaR* as seen in Fig. 21, which depicts a team of three tackers following a moving prey. The prey is traveling at a velocity of 40 ft/s and following a straight line heading due east. The three predators all fly at 50 ft/s with a radar range of 500 ft. Two of the three predators are initiated in opposite directions, with the third being given a more ideal position to quickly find the prey and initiate exploitation. As soon as the prey has been detected by one of the three, the other two immediately change course to the prey based on the shared information. This simulation also provides a strong visual representation of the concept of progressive rapid dives, as one can see the individual agents following parallel to the pray before making sudden turns onto the preys path. The individual dives may present some risk if not properly space apart as mentioned in Section II.

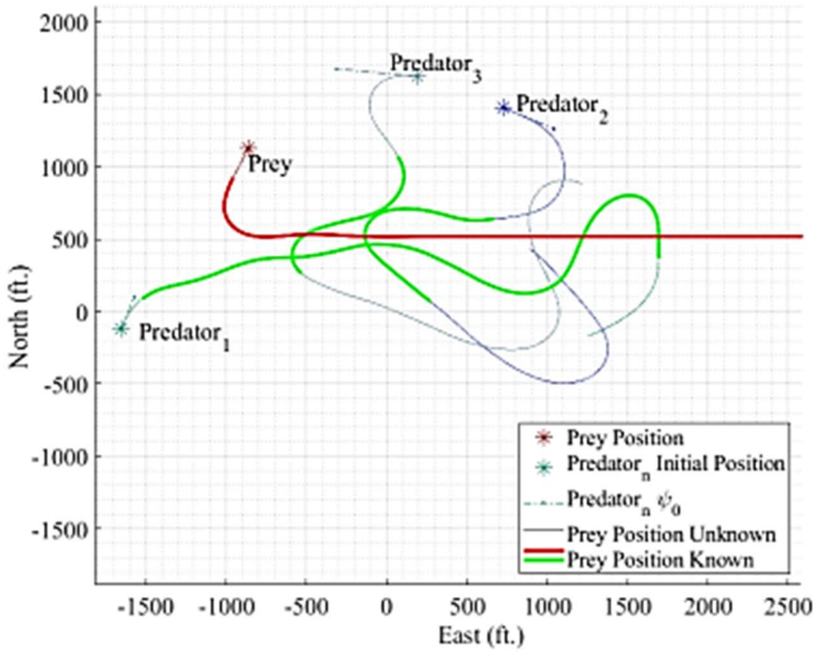

Figure 22 HHSaR in an Exploitation Phase.

4.2 L_N Lateral Guidance

The L_N lateral guidance was validated through a series of simulation and flight tests. Simulation results are presented first in subsection 1 followed by real world flight test results in subsection 2. All work in this section is a result of direct collaboration with Dr. Shawn Keshmiri and Jeffrey Xu, which resulted in the patent source in the Appendix.

4.2.1 Simulation Results

The developed L_N guidance proved to be incredibly potent. In the proceeding Fig. 22-29, the results of a thoroughly tested L_2 guidance (blue) is compared with the L_N guidance (red). Following this analysis of a general case of the guidance, the two will be used in an HHSaR setting and scrutinized accordingly.

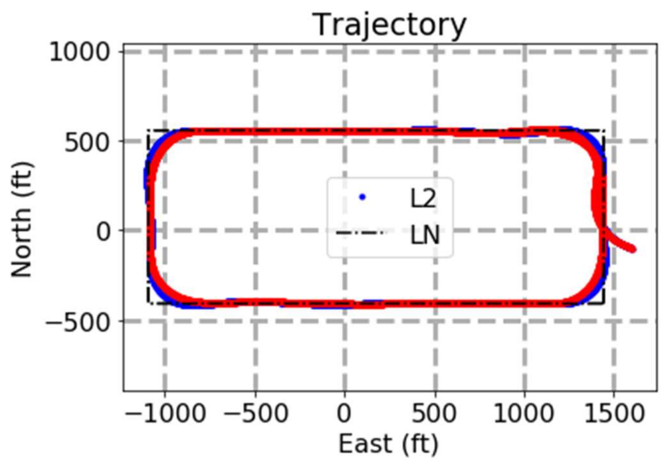

Figure 23 Tracking Comparison of L2 and LN.

The tracking of L_N saw noticeable improvement over the already accurate L_2 in the form of less overshoot and little to no oscillations.

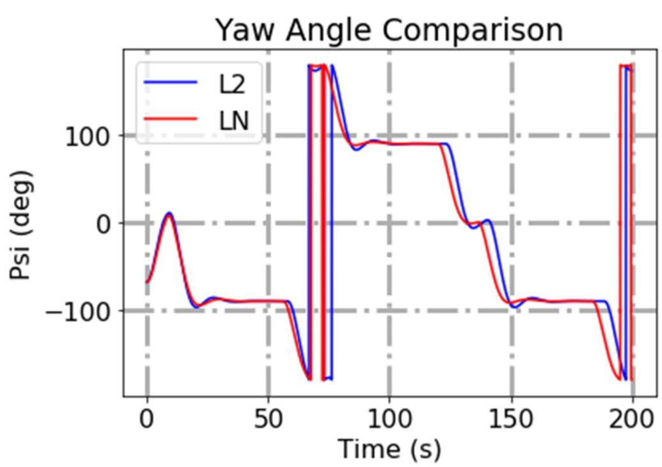

Figure 24 Yaw Angle Comparison of L2 and LN.

Noticeable when comparing yaw angles is that L_N has less oscillations and smoother transitions in turns.

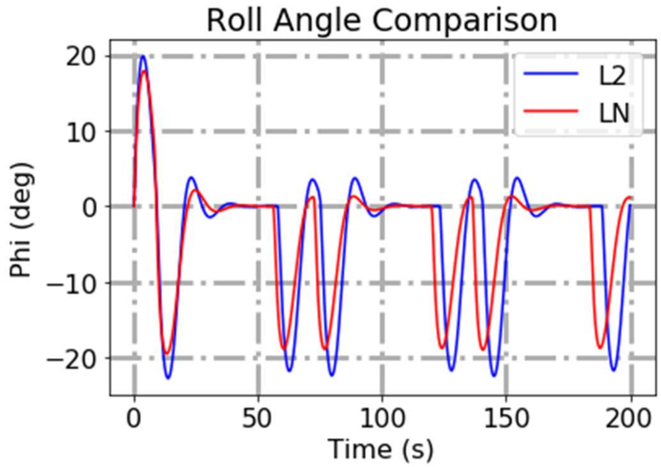

Figure 25 Roll Angle Comparison of L2 and LN.

In addition to smoother turns and improved tracking, Fig. 24 shows L_N achieves said results with less roll than its predecessor.

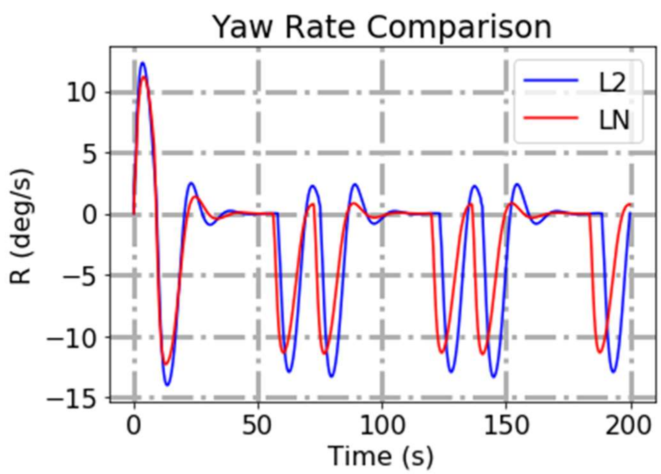

Figure 26 Yaw Rate Comparison of L2 and LN.

The same observation can be made from Fig. 25 for yaw rate, which is to be expected when performing smoother turns.

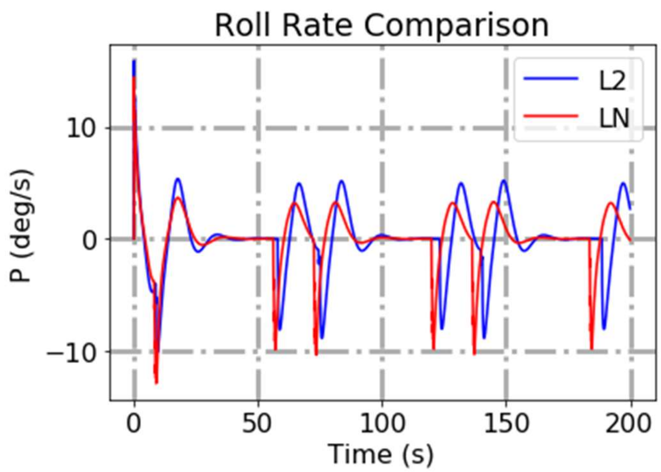

Figure 27 Roll Rate Comparison of L2 and LN.

Fig. 26 shows that roll rate sees occasional higher spikes in magnitude in L_N over its predecessor, however neither guidance produces troublesome or questionable rates for this to be an issue.

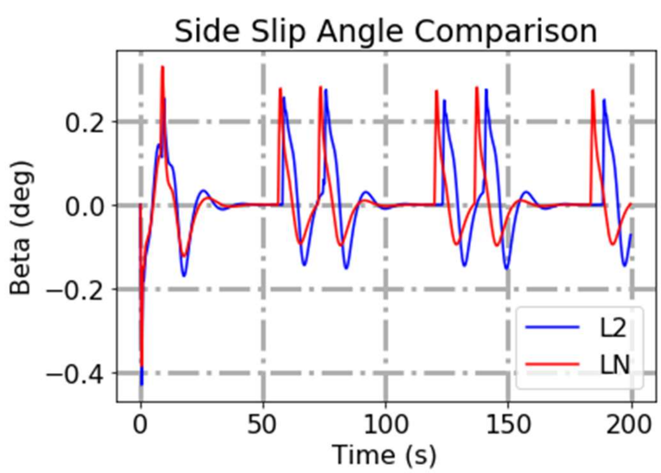

Figure 28 Side Slip Comparison of L2 and LN.

Sideslip shows results similar to roll rate in that it sees the occasional higher spike, though the magnitudes are still nowhere near large enough to cause concern for either guidance.

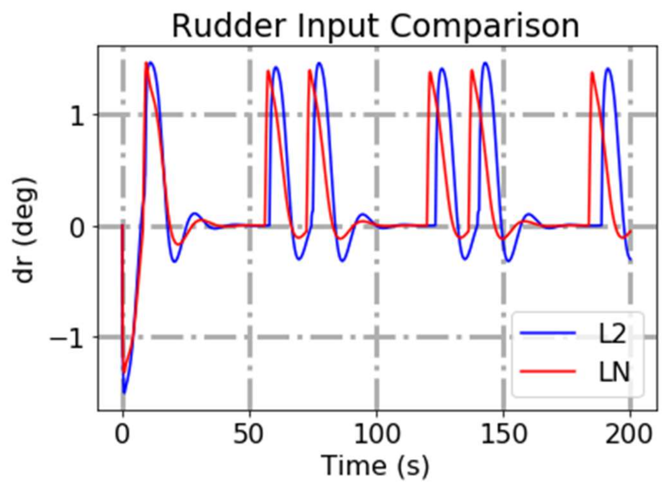

Figure 29 Rudder Input Comparison of L2 and LN.

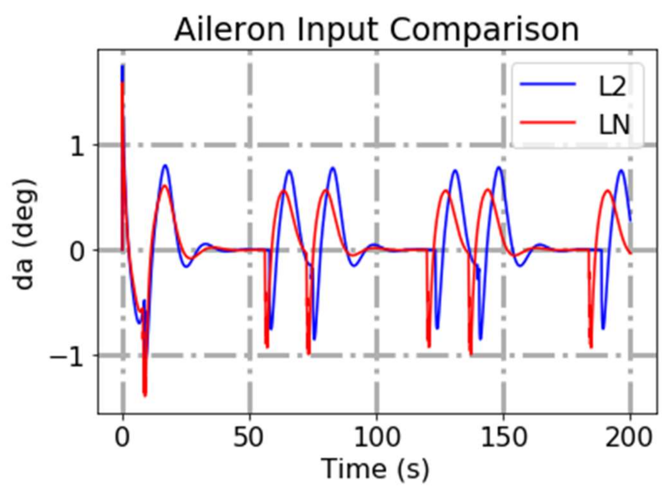

Figure 30 Aileron Input Comparison of L2 and LN.

Fig. 28 and 29 show the control inputs produced by the same LQR-NZSP controller based on the input guidance commands. Aileron is used to control ϕ and rudder is used to control β. Though both guidances produce similar results, with L_N control inputs having fewer oscillations, the first arguable criticism towards L_N is present here in the form of higher control rates.

Next, the two guidance methods were implemented for *HHSaR*. Due to the heuristic nature of the algorithm, the exact same pathing results cannot be guaranteed. As such, the system was run a single time, after which the path produced by *HHSaR* was extracted, and then provided to both methods to run independently, providing a 1 to 1 comparison of the two guidances. The results of this can be seen in Fig. 30, in which one can see L_N has similar improvements in tracking as seen in the prior generic case.

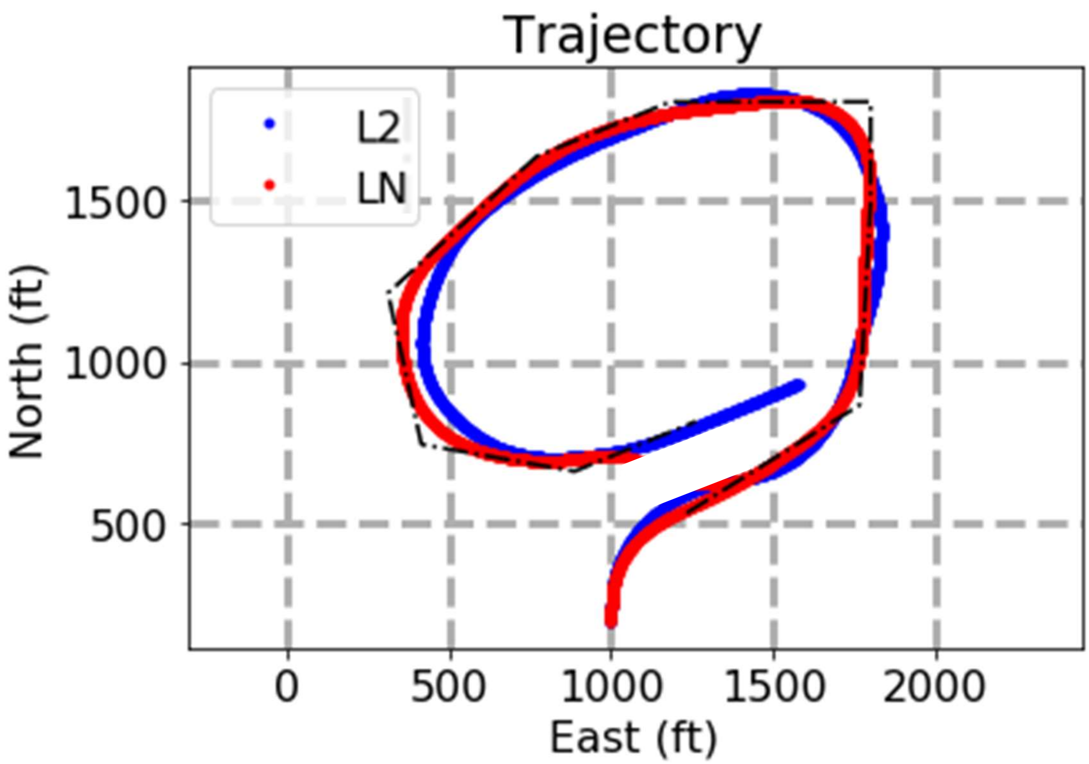

Figure 31 Tracking comparison of Comparison of L2 and LN on an HHSaR Path

4.2.2 Flight Test Results

The flight testing of the L_N lateral guidance proved to be highly successful, with a very noticeable improvement in tracking. Fig. 30 presents a tracking comparison of an L_N flight test and an L_2 flight test, each given the same path and initiated along the path. These results make it clear that in terms of tracking accuracy, L_N is significantly better in terms of tracking.

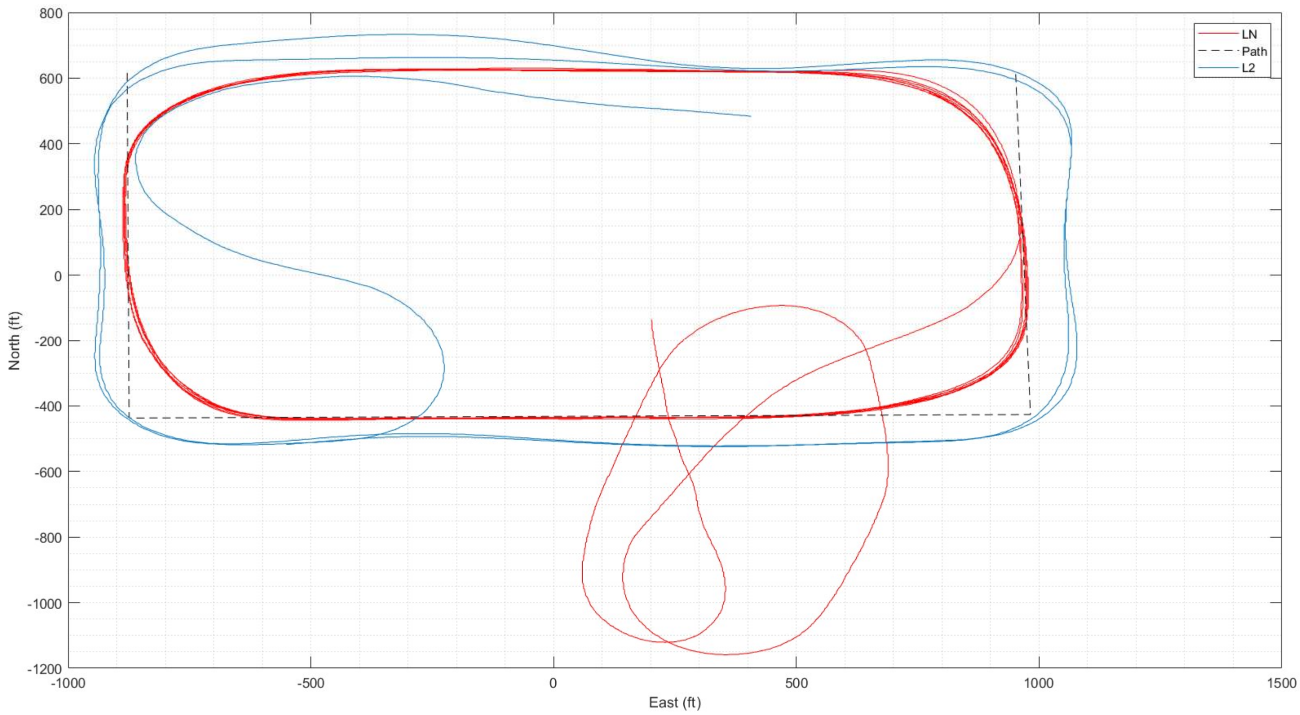

Figure 32 Tracking comparison of Comparison of L2 and LN on a standard waypoint path

To ensure that this increase in tracking accuracy did not come with unreasonable state or control conditions, the systems state space was also analyzed. Fig. 32 shows aircraft roll angle during a 200 s span of flight using L_N guidance. The roll angle never exceeds its imposed limit of 45 deg and overall presents acceptable magnitudes. The system does see the occasional sudden spike in roll angle, and as such state rates were also observed.

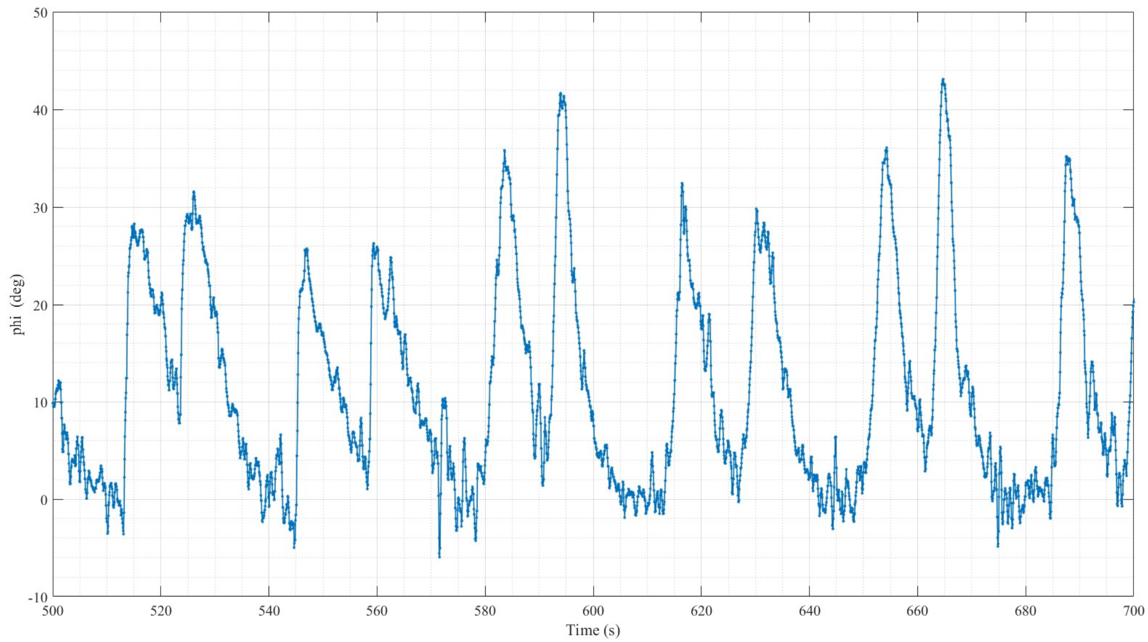

Figure 33 Aircraft roll angle during LN flight test

Figures 33 and 34 contain the roll and yaw rates during the same 200 s time span. For most of the duration, the roll rate is reasonable in magnitude, though as expected does produce the expected occasional spikes. However, these spikes are not excessive or dangerous to the system, and such rates are not outside the realm of reason in an unstructured environment. The same cane be said for yaw rate, which remains within ± 20 deg for the majority of the flight.

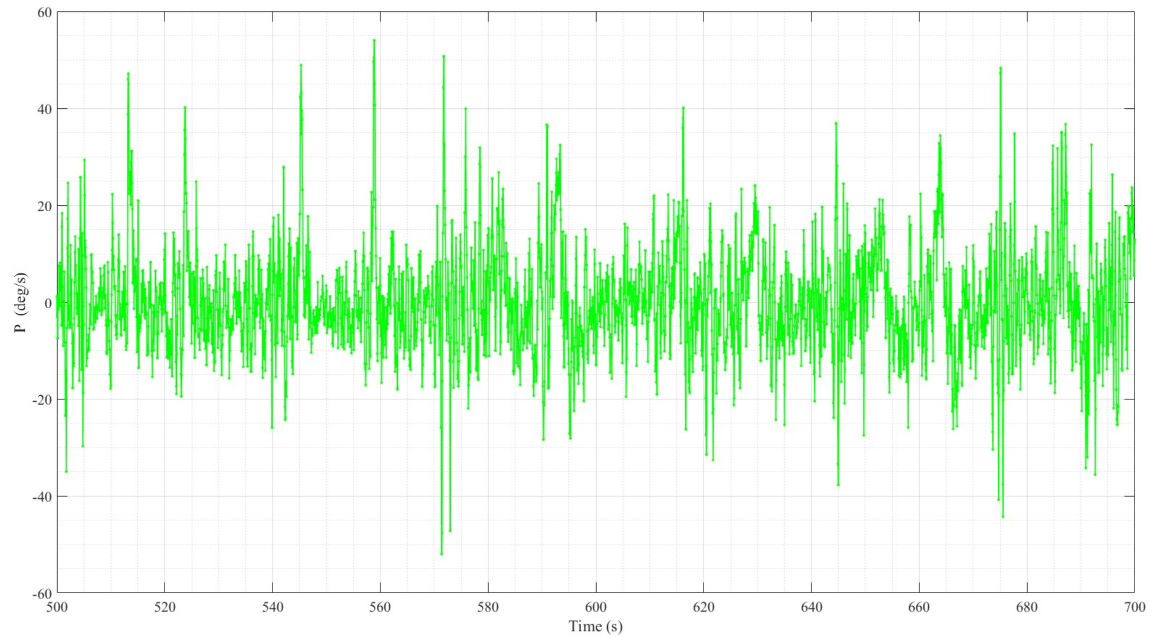

Figure 34 Aircraft roll rate during LN flight test

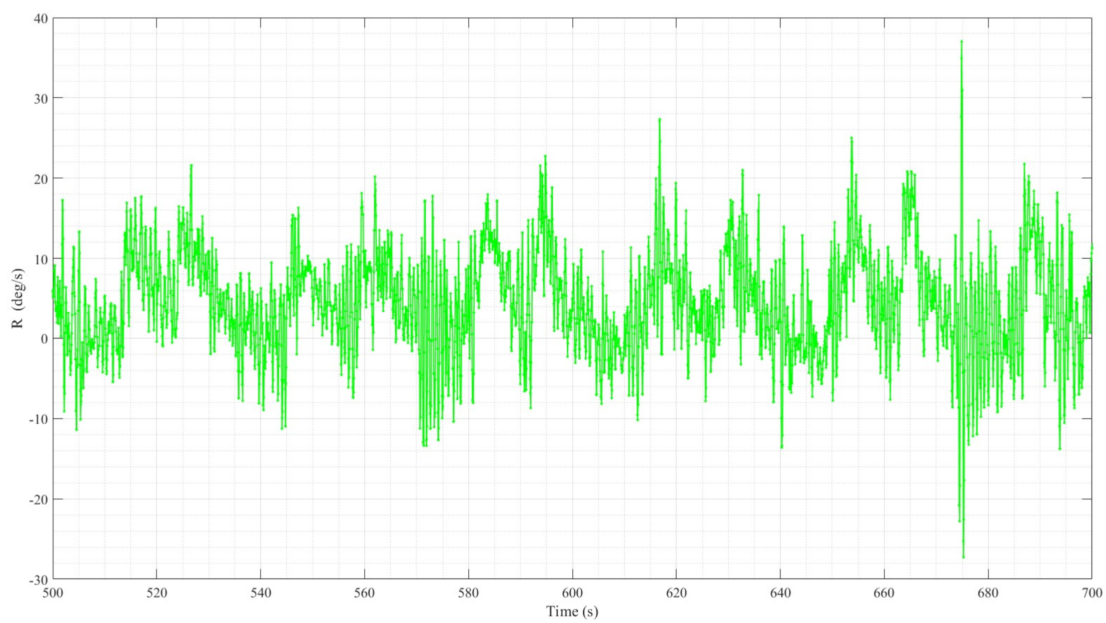

Figure 35 Aircraft yaw rate during LN flight test

Finally, the control surfaces were observed in the same 200 s time frame, as seen in Fig. 35 (aileron) and Fig. 36 (rudder). The aileron deflection never reached any concerning magnitudes. The occasional jump in magnitude is present, for example at 560 s, however even the absolute largest aileron rate of change does not exceed 3.1 deg/s. Similarly, the rudder never produced any concerning inputs, in terms of either magnitude or rate of change.

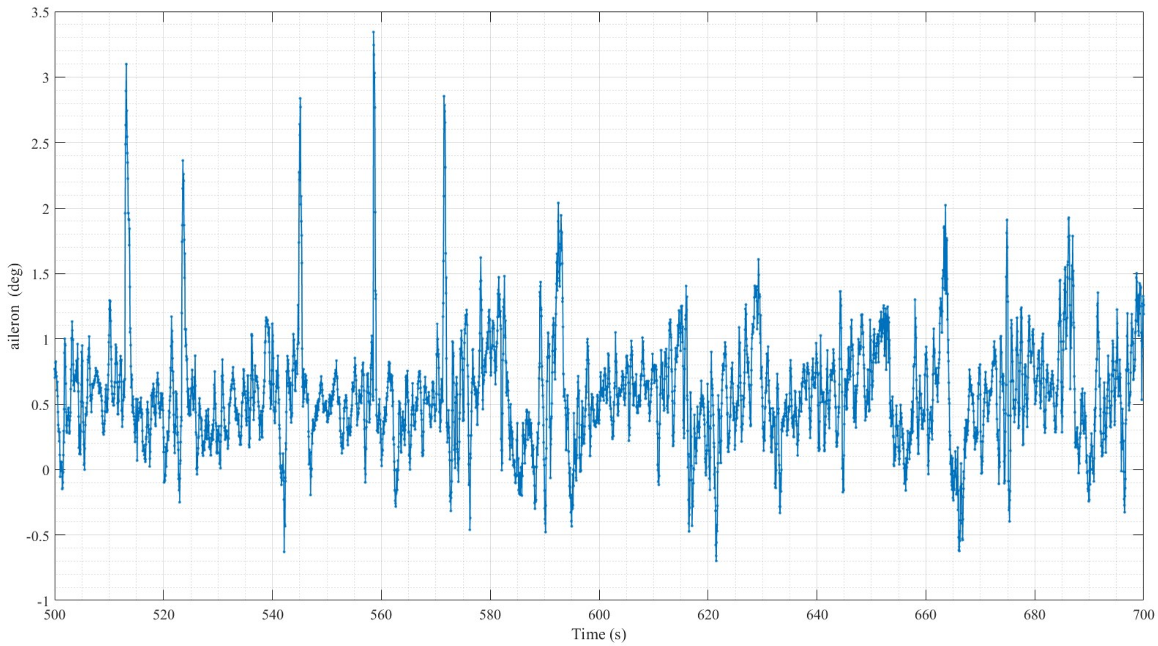

Figure 36 Aircraft aileron input during LN flight test

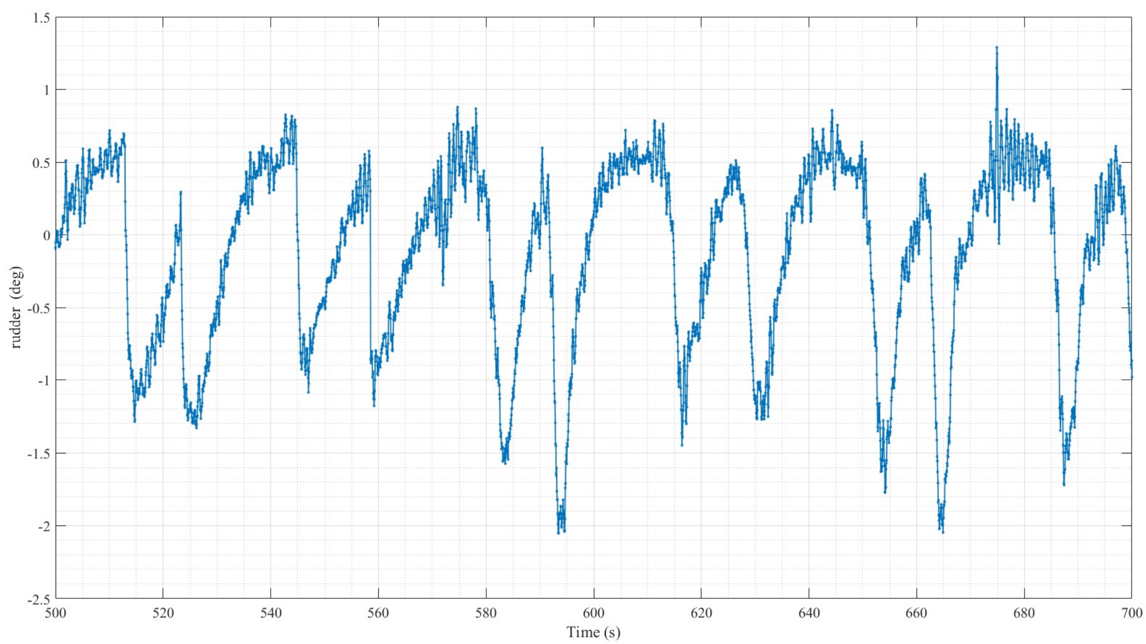

Figure 37 Aircraft rudder input during LN flight test

4.3 Θ_1 Longitudinal Guidance

The primary goal of the developed longitudinal guidance is to prevent erratic maneuvers and inputs. Less aggressive control will lead to lower rise times, as can be seen in Fig. 31-36, however the rate at which the longitudinal guidance converges is not the primary concern, only that the end result keeps the aircraft safe even in sharp turns and regular maneuvers.

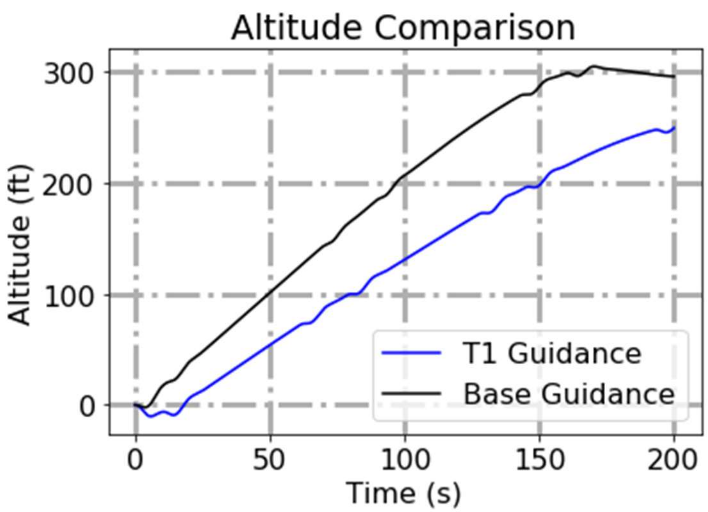

Figure 38 Altitude Comparison of Θ_1 and base guidance.

As expected, the new guidance takes longer to converge to the desired altitude (300 ft) however one should note the nearly linear increase in altitude, originating from the guidances behavior to command a constant Γ. Though it does take a longer period of time to recover from an off-trim condition, the length of time and magnitudes to be recovered are not unlike traditional altitude holds. The main benefit to Θ_1 is best seen when observing states and controls.

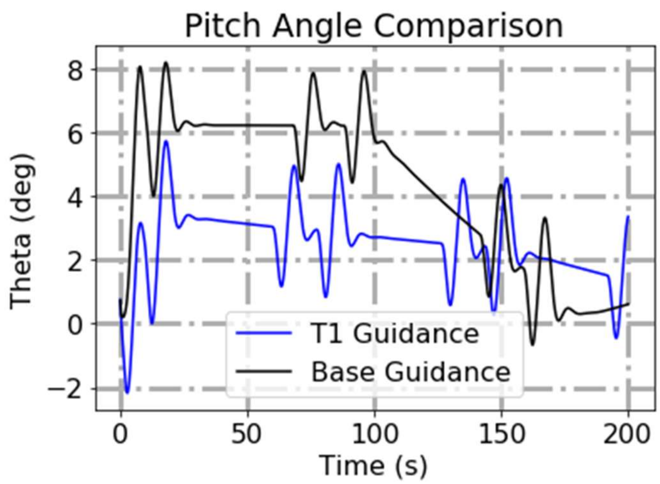

Figure 39 Pitch Angle Comparison of Θ_1 and base guidance.

From Fig. 32, the less aggressive behavior of the new guidance becomes immediately evident. Despite its longer recover times it keeps the aircraft at a significantly lower pitch angle, eventually achieving the same result while using half as much pitch angle.

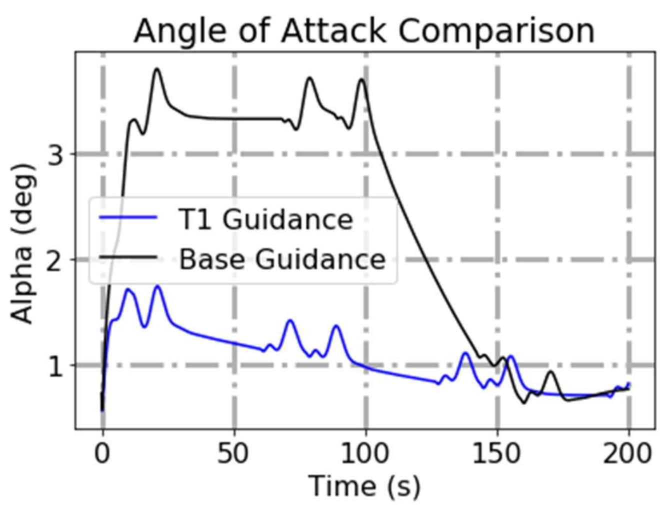

Figure 40 Angle of Attack Comparison of Θ_1 and base guidance.

This is further emphasized when observing angle of attack, in which Θ_1 guidance can hold α to nearly a third of the magnitude of the base autopilot. Such behavior is critical when designing a guidance with the aim of stall prevention.

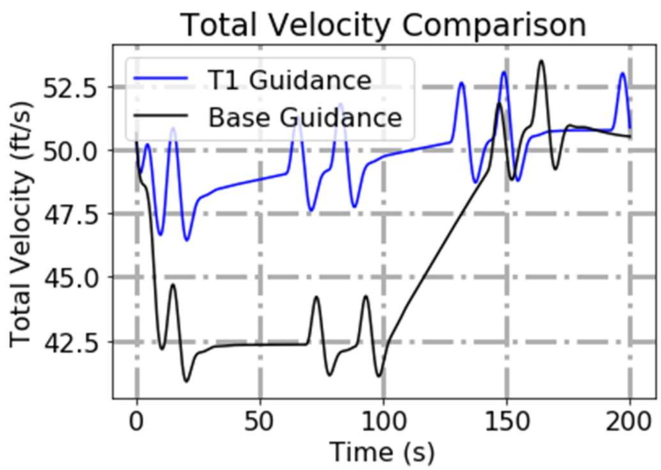

Figure 41 Velocity Comparison of Θ_1 and base guidance.

Once again emphasizing the anti-stall capabilities of Θ_1 guidance, cruise velocity is nearly maintained during ascent while the base autopilot sees a significant dip.

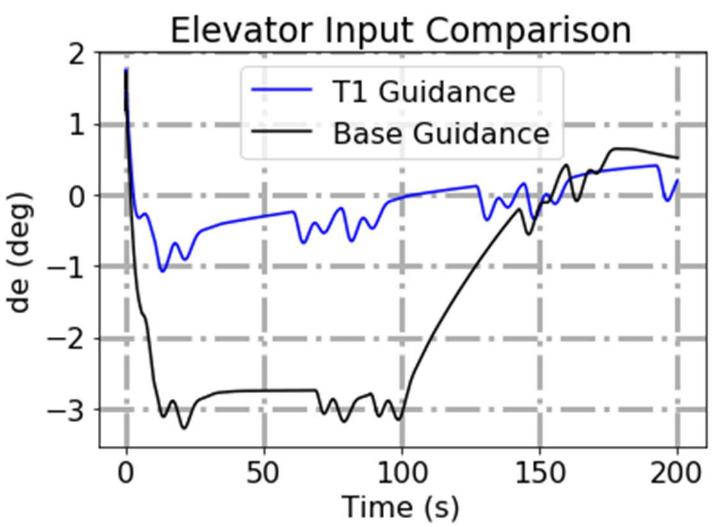

Figure 42 Elevator Input Comparison of Θ_1 and base guidance.

Finally, when observing control inputs, particularly the elevator being used to control the commanded state, one can see the significant reduction in magnitude between the two guidances. Large inputs are generally undesirable as they can lead to loss of control in an unstructured environment.

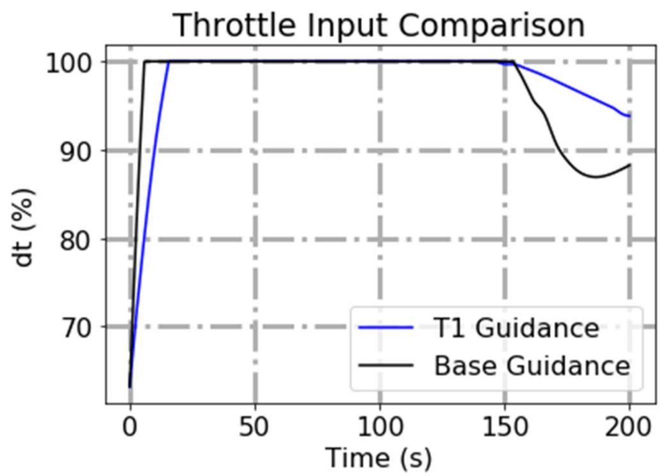

Figure 43 Throttle Input Comparison of Θ_1 and base guidance.

Though throttle controls velocity, which is not an output command of the developed guidance, a comparison of throttle input is still performed. Both of the presented throttle inputs are using the same velocity guidance and throttle controller, only the pitch guidance is different. The primary potential concern of note here is the further extended use of maximum or near maximum throttle levels seen in the new guidance. This is a result of the longer ascent time.

5 Conclusions

The field of Search and Rescue is a rapidly evolving science that exists in a constant state of high risk. Autonomous systems such as UGV and UAS have already seen some use in the field as a means of supporting the human search party. The standard search method employed in these missions is designed for a manned search team without any additional emphasis placed on how to better use their unmanned systems. This work aims to resolve this through the means of a cognitive and intelligent path-planning algorithm that does not require any prior knowledge of the region. This algorithm is developed using bio-inspired meta-heuristic based optimization methods to reduce search time.

The developed algorithm was able to outperform the standard search method 75% of the time when using a single agent, and outperformed the standard search method in every simulation, though significantly fewer trials were run in the multi-agent setting due to computational load. Though the system was tested for up to three agents, the feasibility of higher orders of agents remains to be tested, as well as accounting for the communication network that would need to be set up for such a path-planner. The resultant path-planner will often request sudden maneuvers and sharp turns which do not lend well to traditional guidance algorithms. As such new longitudinal and lateral directional guidance schemes are developed as a means of supporting the new path planning. Both guidances succeeded in achieving their goals, with the lateral L_N guidance outperforming its predecessor, L_2, and Θ_1 producing safer paths and commands than the base KUFRL guidance. All aspects of this research are simulated within a six degrees-of-freedom flight simulator, which presents a noticeable improvement in search times over traditional methods.

6 References

[1] Soza & Company, Ltd. And Office of Search and Rescue U.S. Coast Guard (1996). "The Theory of Search: A Simplified Explanation" DON: 96-F-HNG040

[2] U.S. Coast Guard (2013). "U.S Coast Guard Addendeum to the Unites States National Search and Rescue Supplement to the International Aeronautical and Maritime Search and Rescue Manual (IAMSAR)" COMDTINST M16130.2F

[3] Sujit, P., Saripalli, S. and Borges Sousa, J. (2014). Unmanned Aerial Vehicle Path Following: A Survey and Analysis of Algorithms for FixedWing Unmanned Aerial Vehicless. IEEE Control Systems, 34(1), pp.42- 59.

[4] S. Park, J. Deystt, and J.P. How, "Performance and Lyapunov stability of a nonlinear path-following guidance method," J. Guidance, Control, Dyn., vol. 30, no. 6, pp. 1718-1728,2007.

[5] M. Kothari and I. Postlethwaite. "A probabilistically robust path-planning algorithm for UAVs using rapidly-exploring random trees," J. Intell. Robot. Syst., vol. 71, no.2, pp 231-253, Aug 2013.

[6] A. Ratnoo, P. B. Suit, and M. Kothari, "Optimal path following for high wind flights," in Proc. IFAC World Congr., Milan, Italy, Aug. 28-Sept. 2,2011, pp. 12,985,-12,990.

[7] D. R. Nelson, D. B. Barber, T. W. McLain, and R. W. Beard, "Vector field path following for miniature air vehicles," IEEE Trans. Robot., vol. 23 no. 3, pp. 519-529, 2007.

[8] Bares, P., Lazarus, S., Tsourdos, A. and Savvaris, A. (2013). Adaptive guidance for UAV based on Dubins path. AIAA Guidance, Navigation, and Control (GNC).

[9] Park, S., Deyst, J. and How, J. (2004). A new nonlinear guidance logic for trajectory tracking. AIAA Guidance, Navigation, and Control Conference and Exhibit.

[10] R. Curry, M. Lizarraga, B. Mairs and G. H. Elkaim, "L+2, an improved line of sight guidance law for UAVs," 2013 American Control Conference, Washington, DC, 2013, pp. 1-6. doi: 10.1109/ACC.2013.657980.

[11] S. Shehab and L. Rodrigues "Preliminarly Results on UAV Path Following Using Piecewise-Affine Control," Proceedings of the IEEE Conference on Control Applications, Toronto, Canada, 2005, pp. 4906- 4913, doi:10.1109/ACC.2006.16574978

[12] Stastny, T., Garcia, G. and Keshmiri, S. (2014). Collision and Obstacle Avoidance in Unmanned Aerial Systems Using Morphing Potential Field Navigation and Nonlinear Model Predictive Control. Journal of Dynamic Systems, Measurement, and Control, 137(1), p.014503.

[13] Blevins, A., Benyamen, H., Godfrey, G., Shukla, D. and Kim, B. (2018). Analysis and Verification of Cost-Effective Design Modifications to Commercially Available Fixed-Wing Unmanned Aerial Vehicle to Improve Performance, Stability and Control Characteristics, and Structural Integrity. AIAA Information Systems.

[14] S. Park, J. Deyst, and J. P. How, "Performance and Lyapunov stability of a nonlinear path-following guidance method," in Journal of Guidance, Control, and Dynamics, vol. 30, no. 6, pp. 1718-1728, Dec. 2007, doi: 10.2514/1.28957.

[15] R. Curry, M. Lizarraga, B. Mairs and G. H. Elkaim, "L2 + , an improved line of sight guidance law for UAVs," 2013 American Control Conference, Washington, DC, 2013, pp. 1-6. doi: 10.1109/ACC.2013.657980.

[16] A. Ratnoo, P.B. Sujit, and M, Kothari, "Adaptive optimal path following for high wind flights," in IFAC World Congress, Milano, Italy, 28 Aug, 2011, pp. 12985-12990.

[17] P.B. Sujit, S. Saripalli, J.B. Sousa, "Unmanned aerial vehicle path following," in IEEE Control Systems Magazine, vol. 34, no. 1, pp. 42-59. doi: 10.1109/MCS.2013.2287568.

[18] S. A. Isaev, P.A. Baranov, Y. V. Zhukova, A. A. Tereshkin, and A. E. Usachov, "Simulation of the wind effect on an ensemble of high-rise buildings by means of multiblock computational technologies," in Journal of Engineering Physics and Thermophysics, vol. 87, no. 1, pp. 112-123, Jan. 2014, doi: 10.1007/s10891-014-0991-7

[19] M. Z. Shah, R. Samar and A. I. Bhatti, "Guidance of Air Vehicles: A Sliding Mode Approach," in IEEE Transactions on Control Systems Technology, vol. 23, no. 1, pp. 231-244, Jan. 2015.

[20] M. Y. Hui, C. Q. Yang, H. Z. Xi and G. Zheng, "An improved nonlinear guidance law for unmanned aerial vehicles path following," 2015 34th Chinese Control Conference (CCC), Hangzhou, 2015, pp. 5271-5276. doi: 10.1109/ChiCC.2015.7260462.

[21] F. Balampanis, A. P. Aguiar, I. Maza and A. Ollero, "Path tracking for waypoint lists based on a pure pursuit method for fixed wing UAS," 2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS), Linkoping, 2017, pp. 55-59. doi: 10.1109/REDUAS.2017.8101643

[22] Benyamen, H. (2018) "Stability and Control Derivatives Identification for an Unmanned Aerial Vehicle with Low Cost Sensors Using an Extended Kalman Filter Algorithm," University of Kansas, Lawrence, KS, USA.

[23] Kuhn, Harold W. "The Hungarian method for the assignment problem." Naval research logistics quarterly 2.1-2 (1955): 83-97.

[24] Yang, Xin-She, and Suash Deb. "Cuckoo search via Lévy flights." 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC). IEEE, 2009.

[25] Abbass, Hussein A. "MBO: Marriage in honey bees optimization-A haplometrosis polygynous swarming approach." Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546). Vol. 1. IEEE, 2001.

[26] Li, X. L. "A new intelligent optimization-artificial fish swarm algorithm." Doctor thesis, Zhejiang University of Zhejiang, China (2003).

[27] Mucherino, Antonio, and Onur Seref. "Monkey search: a novel metaheuristic search for global optimization." AIP conference proceedings. Vol. 953. No. 1. AIP, 2007.

[28] Dorigo, Marco, and Gianni Di Caro. "Ant colony optimization: a new meta-heuristic." Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406). Vol. 2. IEEE, 1999.

[29] Mirjalili, Seyedali, and Andrew Lewis. "The whale optimization algorithm." Advances in engineering software 95 (2016): 51-67.

[30] Yang, Xin-She. "Bat algorithm for multi-objective optimisation." arXiv preprint arXiv:1203.6571 (2012).

[31] Mirjalili, Seyedali, Seyed Mohammad Mirjalili, and Andrew Lewis. "Grey wolf optimizer." Advances in engineering software 69 (2014): 46-61.

[32] Mirjalili, Seyedali, et al. "Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization." Expert Systems with Applications 47 (2016): 106-119.

[33] Le Pichon, Thomas. "Gray Wolf Hierarchy Pyramid." 2020. PNG.

[34] Heidari, Ali Asghar, et al. "Harris hawks optimization: Algorithm and applications." *Future generation computer systems* 97 (2019): 849-872.

[35] Deb, Kalyanmoy. "Investigating predator-prey algorithms for multiobjective optimization." (2005).

[36] Xu, J., Le Pichon, T., Spiegel, I., Hauptman, D. and Keshmiri, S. (2019). Bio-Inspired Predator-Prey Large Spacial Search Path Planning. IEEE Aeroconference.

[37] Le Pichon, T., Xu, J. and Keshmiri, S. (2019). Optimized Guidance Methods for Smooth Transitions in UAS Path Following. IEEE NAECON.

[38] Xu, J., Le Pichon, T., and Keshmiri, S. (2019). Multi-Eye Guidance Method for UAVs Path Following. IEEE NAECON.

[39] Kuroki, Yuki, George S. Young, and Sue Ellen Haupt. "UAV navigation by an expert system for contaminant mapping with a genetic algorithm." Expert Systems with Applications 37.6 (2010): 4687-4697.

[40] Allaire, François CJ, et al. "FPGA implementation of genetic algorithm for UAV real-time path planning." Unmanned Aircraft Systems. Springer, Dordrecht, 2008. 495-510.

[41] Pehlivanoglu, Y. Volkan, Oktay Baysal, and Abdurrahman Hacioglu. "Path planning for autonomous UAV via vibrational genetic algorithm." Aircraft Engineering and Aerospace Technology (2007).

[42] Nikolos, Ioannis K., et al. "Evolutionary algorithm based offline/online path planner for UAV navigation." IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 33.6 (2003): 898-912.

[43] Huang, Hsu-Chih, and Ching-Chih Tsai. "Global path planning for autonomous robot navigation using hybrid metaheuristic GA-PSO algorithm." SICE Annual Conference 2011. IEEE, 2011.

[44] Sugihara, Kazuo, and John Smith. "Genetic algorithms for adaptive planning of path and trajectory of a mobile robot in 2d terrains." IEICE TRANSACTIONS on Information and Systems 82.1 (1999): 309-317.

[45] Mohanty, Prases Kumar, and Dayal R. Parhi. "Cuckoo search algorithm for the mobile robot navigation." International Conference on Swarm, Evolutionary, and Memetic Computing. Springer, Cham, 2013.

[46] Ferguson, Dave, Maxim Likhachev, and Anthony Stentz. "A guide to heuristic-based path planning." Proceedings of the international workshop on planning under uncertainty for autonomous systems, international conference on automated planning and scheduling (ICAPS). 2005.

[47] Meng, Zhenyu, Jeng-Shyang Pan, and Abdulhameed Alelaiwi. "A new meta-heuristic ebb-tide-fish-inspired algorithm for traffic navigation." Telecommunication Systems 62.2 (2016): 403-415.

[48] Mirjalili, Seyedali. "Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems." Neural Computing and Applications 27.4 (2016): 1053-1073.

[49] Chanthery, Elodie. Planification de mission pour un véhicule aérien autonome. Diss. 2005.

[50] Le Pichon, Thomas. "HiTL Simulation" 2020. PNG.