2019-12-01

# Trajectory Optimization and Design for a Large Number of Unmanned Aerial Vehicles

Jenna Elisabeth Newcomb
*Brigham Young University*

# Trajectory Optimization and Design for a Large Number of Unmanned Aerial Vehicles

Jenna Elisabeth Newcomb

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Andrew Ning, Chair
Cammy Peterson
Christopher Mattson

Department of Mechanical Engineering

Brigham Young University

ABSTRACT

Trajectory Optimization and Design for a Large Number of Unmanned Aerial Vehicles

Jenna Elisabeth Newcomb
Department of Mechanical Engineering, BYU
Master of Science

An unmanned aerial vehicle (UAV) swarm allows for a more time-efficient method of searching a specified area than a single UAV or piloted plane. There are a variety of factors that affect how well an area is surveyed. We specifically analyzed the effect both vehicle properties and communication had on the swarm search performance. We used non-dimensionalization so the results can be applied to any domain size with any type of vehicle. We found that endurance was the most important factor. Vehicles with good endurance sensed approximately 90% to 100% of the grid, even when other properties were lacking. If the vehicles lacked endurance, the amount of area the vehicles could sense at a given time step became more important and 10% more of the grid was sensed with the increase in sensed area. The maneuverability of the vehicles was measured as the vehicles' radii of turn compared to the search domain size. The maneuverability mattered the most in the middle-range endurance cases. In some cases 30% more of the grid was searched with improving vehicle maneuverability. In addition, we also examined four communication cases with different amounts of information regarding vehicle location. We found communication increased search performance by at least 6.3%. However, increasing the amount of information only changed the performance by 2.3%. We also studied the impact the range of vehicle communication had on search performance. We found that simulations benefited most from increasing the communication range when the amount of area sensed at a given time step was small and the vehicles had good maneuverability. We also extended the optimization to a multi-objective process with the inclusion of target tracking. We analyzed how the different weightings of the objectives affected the performance outcomes. We found that target tracking performance dramatically changes based on the given weighting of each objective and saw an increase of approximately 52%. However, the amount of the grid that was sensed only dropped by approximately 10%.

Keywords: trajectory optimization, UAV swarms, design optimization, communication

# ACKNOWLEDGMENTS

I would like to express my sincere gratitude for my committee chair and mentor Dr. Andrew Ning for all his help and guidance. He has helped me grow tremendously both academically and individually from his example and dedication to my learning. I would also like to show my appreciation for Dr. Cammy Peterson and Dr. Chris Mattson for their assistance and time.

I would like to acknowledge the funding I received from the Center for Unmanned Aircraft Systems. C-UAS graciously funded my research including tuition and a research assistant salary for the past two years.

Lastly, I would like to thank my parents for their guidance and encouragement when I was doubting myself and helping me achieve my full potential.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

| | |
|---|---|
| $\gamma$ | Flight Path Angle |
| $\psi$ | Yaw angle |
| $\psi_t$ | Target heading angle |
| $\phi$ | Roll angle |
| $\sigma^2$ | Variance of grid value distribution |
| $A_r$ | Area ratio |
| $d_i$ | Distance between $i^{th}$ cell and closest vehicle |
| $E_r$ | Endurance Ratio |
| $g$ | Acceleration due to gravity |
| $J$ | Potential reward for planned path |
| $L$ | Domain length |
| $l_c$ | Length of grid cell size |
| $N_g$ | Number of grid cells |
| $N_v$ | Number of vehicles |
| $N_t$ | Number of targets |
| $N_{ts_i}$ | Number of targets seen at the $ith$ time step |
| $n_g$ | Number of guesses for multi-start |
| $n_i$ | Number of initial starting locations |
| $n_t$ | Number of time steps |
| $n_{roll}$ | Number of planned roll angles |
| $O$ | Three dimensional matrix of optimized grid values |
| $P_g$ | Grid performance |
| $P_t$ | Target track performance |
| $p_e$ | Inertial east position of the vehicle |
| $p_n$ | Inertial north position of the vehicle |
| $R_s$ | Sensing radius |
| $R_t$ | Turning radius |
| $R_c$ | Communication radius |
| $r_i$ | Value of $i^{th}$ cell |
| $r_l$ | Reward from $l^{th} target$ |
| $r_m$ | Maximum value of grid cell |
| $r_{p_i}$ | Potential value of $i^{th}$ cell |
| $t$ | Length of time step |
| $T$ | Total duration of simulation |
| $Te$ | East position of target |
| $Tn$ | North position of target |
| $T_r$ | Turning ratio |
| $V_a$ | Airspeed |
| $V_{cr}$ | Cruise speed of vehicle |
| $V_g$ | Ground speed |
| $V_{max}$ | Maximum speed of vehicle |
| $V_s$ | Stall speed of vehicle |
| $V_t$ | Target velocity |

| | |
|---|---|
| $W_g$ | Grid weighting |
| $W_t$ | Target weighting |

# CHAPTER 1. BACKGROUND

Unmanned Aerial Vehicles are increasing in popularity due to the recent improvements in weight, cost, and operational complexity. Fig. 1.1 shows the increase in millions of dollars of the Commercial UAV market [1].

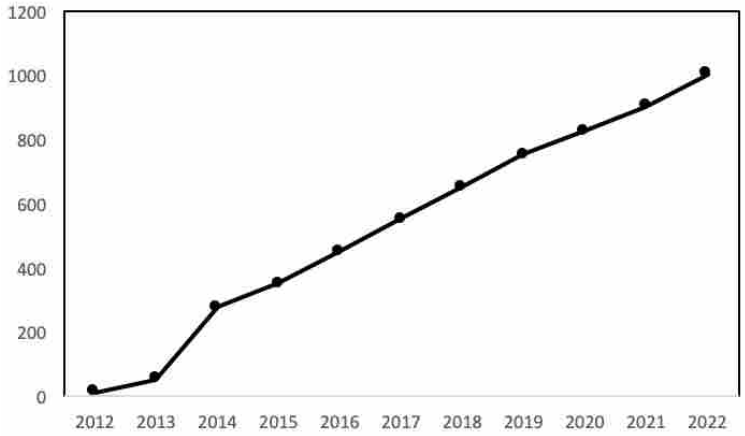



Figure 1.1: The size of the UAV market by million of USD over one decade.

The global UAV market is estimated to have a compound annual growth rate of 17%[1] and many companies are using UAVs as means to complete tasks more efficiently. New uses include delivery, surveillance, law enforcement, environmental studies, infrastructure, and disaster management. Because of this, new technology and research is needed to satisfy the rapidly expanding market.

For this research, we explored aerial vehicle swarms and multi-objective missions. There are many characteristics that can affect the performance of a large swarm. We analyzed vehicle properties, communication, and multi-objective weighting to determine the effect each had on swarm performance.

[1] https://www.grandviewresearch.com/industry-analysis/commercial-uav-market

## 1.1 Mission-Based Design

We analyzed different mission parameters to determine the respective effect each has on swarm diversity. These parameters included the endurance of the aircraft, sensing properties, vehicle agility, and communication distance. We used nondimensionalization, so the results are applicable and generalizable to a wide range of situations.

Many people have created trajectory optimization methods subject to dynamic vehicle constraints. Sujit and Ghose created a path optimization with vehicle endurance as a constraint [1]. They only allowed the UAVs to travel a specified amount before needing to return to refuel. Blasi, Barbato, and Mattei used particle swarm optimization to optimize flight paths subject to specific operational constraints including minimum turning radius, range and endurance, maximum climbing rate, maximum payload capacity, and maximum and minimum speed [2].

Nigma, Bieniawsi, Kroo, and Vian account for dynamic constraints in their proposed control policy [3]. Using a 3 degree of freedom motion model, they analyzed how dynamics affected the performance of the UAVs. Their performance metric was the cell age based on when the cell was last explored. Their analysis showed that when using the Actual Distance Policy, where the actual distances reflect the dynamic constraints, the maximum cell ages increased from 79 to 106.6 when increasing the turning radius from 1.67 to 5 m and increasing $C_{Lmax}$ from 1.03 to 1.67. Although Nigma, Bieniawsi, Kroo, and Vian looked into how dynamics affected search performance, their primary motivation was to compare the Actual Distance Policy to the Euclidean Distance Policy under dynamic constraints.

Much of the published research, including the above, sought to find a way to optimize paths with vehicle operational constraints, whereas we wanted to know how changing these operational properties affect search performance. In essence, we were not concerned with how to optimize paths constrained to a certain vehicle but how to design or choose a vehicle which is optimal for a given search mission. Our end goal was to know what operational properties affected the search performance most significantly. Therefore, we non-dimensionalized the operational properties to find the impact they had on mission performance. The results are generalizable and applicable to a wide range of domains.

## 1.2 Communication Effects

One question that arises when considering vehicle coordination is the amount of information shared between vehicles. When operating simultaneously, aerial vehicles must communicate to remain dispersed and accomplish missions. Without communication, an UAV swarm will operate inefficiently. However, noise and network connectivity interfere with vehicle communication. In addition, the amount of information shared between swarm members increases computational cost, which is especially detrimental to vehicles flying real-time. It is important to distinguish how shared information alters the vehicles' chosen paths because of the dramatic effect different types of information can have on mission completion. In addition, some shared information may only have a minimal effect and by eliminating this information computational costs are reduced.

Researchers have explored UAV swarm path planning and mission assignment [4–13], and how vehicles communicate [14, 15]. However, little has been done to change and analyze different types or amounts of communication. Waharte and Rigoni compared the effect sharing data has on a search and rescue mission through analyzing the results of different optimization methods [16]. Their goal was to minimize the time required for the UAVs to locate a target. Partially observable Markov decision process based, potential-based, and greedy processes were used for the search algorithms. The study concluded that the partially observable Markov decision process and look ahead methods were the most effective when it came to reducing the time to find the target. These results showed that the most suitable paths came from methods in which vehicles shared and used information about predicted locations of a target. Although these methods were effective they were computationally expensive. This study was primarily focused on the effectiveness of the optimization methods. Some methods shared data and others did not. Since communication was not the only varying factor, it is not known how much the communication contributed to increasing the swarm's performance. For part of this research, we examined how both the type and amount of information shared between vehicles affects swarm performance.

Krieger and Billeter used a group of robots to gather food items for energy [17]. They utilized a decentralized task distribution method based on insect swarm behavior. They performed the experiment both with no coordination and with a simple coordination approach in which robots could guide other members to the discovered food source. In the end, they concluded that coordinated communication increased total group performance by 13%. Arkin, Balch, and Nitz showed

that communication can increase performance and productivity in a robotic swarm [18]. They assessed task completion for two scenarios: the first with no communication, and the second with a collective memory where robots can share and access information pertaining to the other members. If a robot was unable to locate a goal, it used the collective memory to find a robot that had found a goal. They measured performance by the average distance traveled by the robots. The results showed that communication decreases the overall total distance traveled. The number of steps decreased, showing that communication allowed more efficient behavior.

Trianni and Dorigo used common behavior strategies found in insect colonies to create swarm algorithms [19]. They analyzed both direct and indirect communication as well as direct interactions for a group of s-bots attempting to avoid holes in a specified area. They concluded that using direct communication resulted in far fewer hole encounters. This study demonstrated that for a group of robots with a common goal, communication is necessary.

These studies do focus on the effect of communication alone; however, they are all-or-nothing approaches: the robots performed either with, or without, communication. There was no testing to determine how much information needed to be shared. In contrast, we tested a range of communication amounts to determine the increase in group performance. As a result, we likewise know that communication does improve search performance and how much different amounts improve performance. Therefore, we cannot only show the importance of information sharing, but we can limit the information to that which is necessary. In addition, we coupled communication with design. We ran two communication cases with different communication ranges and a no communication case across a range of vehicle properties. As a result, we can conclude based on the vehicle type and mission characteristics how much communication will benefit the swarm performance.

### 1.3 Vehicle Diversity

Previous work has been performed with heterogeneous swarms. Modified genetic algorithms, Ant-Colony, and Artificial Potential Function based strategies have been used to assign vehicle tasks within the diversified swarms [5, 8]. Many have explored the concept of superagents where a more sophisticated vehicle oversees and guides the actions of other swarm members [4–6].

In addition Howard, Parker, and Sukhatme have found that the system cost was reduced by close to an order of magnitude by using a heterogeneous swarm.

There are different types of aerial vehicles, each with its own unique set of capabilities. Using a non-homogeneous swarm increases the effectiveness of a searching and tracking mission through utilization of these differing capabilities. For example, some vehicles are able to obtain a faster flight speed. We created vehicle diversity through varying the velocities at which vehicles can fly. We added velocity as a design variable subject to upper and lower bounds. As a result, the optimization determines what velocity is best suited for that vehicle at that particular moment. This enhanced performance because vehicles could decrease or increase velocity to maintain course with a target or change velocity dependent on the location of search boundaries.

# CHAPTER 2. FORMULATION OF THE PATH OPTIMIZATION

The end goal of this research was to determine how to coordinate a group of vehicles to best search a grid space. We analyzed how both communication and swarm properties affected mission completion. To achieve maximum performance, it was important to have a way for vehicles to use available information to make the best decision possible about where to travel. We created a process in which vehicles optimize their paths based on their current location and the predicted location of vehicles in close proximity. A high-level overview of this process is shown below. The number of time steps of the simulation is $n_t$, the number of vehicles is $N_v$, and $n_i$ is the number of initial starting locations for each optimization.

**Algorithm 1** Path Optimization

```
1:  procedure PATH OPTIMIZATION
2:      Initialize Locations
3:      for i in n_t do
4:          for j in N_v do
5:              for k in n_i do
6:                  Optimize future location
7:              end for
8:              Compare objective outputs
9:              Determine roll angles and velocity
10:         end for
11:         Propagate vehicles
12:         Propagate targets
13:         Share information
14:     end for
15:     Calculate performance
16: end procedure
```

We began the process with the initialization of the vehicles and grid. We set the grid area as a square based on the user-directed domain length, $L$. Since searching a grid was one of the missions our research aimed to address, we needed a way to determine if the vehicles searched all

sections of the grid. Thus in the domain area, we created grid points that were evenly spaced by a distance of $l_c$. We later used these points in the reward function. In addition, we spaced vehicles evenly through the grid area. We set the number of vehicles equal to a perfect square so there could be a symmetric distribution of vehicles. We set the heading angle of each vehicle to a random value around the unit circle. See section 2.1 for more information on the initialization process.

After the initialization, we optimized each vehicle's trajectory with $n_i$ different starting trajectories. The trajectories consisted of $n_{roll}$ roll angles. We used gradient-based optimization to determine the optimal trajectory and velocity of each vehicle based on current information from other vehicles. In addition, we used a receding horizon controller to continuously re-plan the vehicle path. After the path was planned, the vehicles proceeded only one step, using the first optimized roll angle. We then re-optimized the path with new available information. See section 2.2 for more information on the optimization process. To simulate the flight of a UAV, we used a simple coordinated turn model to predict motion based on the input roll angles and velocity. See Sec. 2.3 for a more in-depth look at the motion model.

For the objective function used in the optimization, we summed all the grid point values for each predicted path based on the vehicle's predicted location. Rather than provide a constant reward value across the point, we used a Gaussian distribution to allow the reward to vary continuously. As a result of this continuous reward, we could use gradient-based optimization. The realized reward was the largest value obtained for each point. Later, we added target tracking to create a multi-objective problem. The target rewards were based on the same Gaussian distribution as the grid cell reward. Sec. 2.4 and section 2.5 give more information on the objective function.

Gradient-based optimization works well with convex problems and can be extended to non-convex problems. If the objective function is noisy or discontinuous, a local minimum or maximum may be found instead of the global minimum or maximum. To increase the likelihood of finding a global maximum, we used a multistart. In other words, we used different initial guesses, $n_i$, for the roll angles and velocity for the optimization function. We then compared the results of the different guesses to determine which roll angles and velocity to use. Section 2.6 discusses the multistart more in-depth.

Since the convergence time of the optimization scales with the number of design variables, we used a decentralized approach where we determined each vehicle's path separately based on

available information. In a centralized case, we would determine every roll angle and velocity in unison which is beneficial for swarm distribution and increasing search performance. However, centralized approaches are extremely computationally expensive due to the large number of design variables. As a result, we simulated the decentralized movement by allowing the optimization for each vehicle to finish before we propagated the vehicles forward and simulated communication.

After, we simulated communication through sharing the predicted paths of the vehicles from the previous time step. We performed calculations to determine which vehicles were within each vehicle's communication radius, $R_c$. If two vehicles were within the communication radius, we simulated communication by sharing the predicted grid point values from the vehicles' future planned paths. In this way, we coordinated vehicles and adjusted the vehicles accordingly to remain dispersed and search the grid. See section 2.7 for more detail on the communication aspect of the algorithm.

Finally, we calculated the total performance of the grid search. The total vehicle search performance determined at the end of the simulation is given by Eq. 2.1. It was the fraction of the grid that was seen after the simulation was completed.

$$P_g = \frac{\sum_{i=1}^{N_g} r_i}{r_m N_g} \tag{2.1}$$

In this equation, $r_i$ is the $ith$ cell's reward, given in Eq. 2.16. The maximum reward of the grid cell is one and is represented by $r_m$. The number of grid cells is $N_g$. The cell reward is summed across every cell then divided by $r_m$ times $N_g$, which is the maximum attainable reward for the entire grid.

For some simulations, we added target tracking. To calculate the performance of the target tracking, we slightly modified the above performance equation to give the fraction of the number of targets seen at each time step over the course of the simulation. The equation for the target track performance is given by Eq. 2.2 below.

$$P_t = \frac{\sum_{i=1}^{T} \frac{N_{ts_i}}{N_t}}{T} \tag{2.2}$$

The total duration of the simulation is given by $T$. The number of seen targets at the $ith$ iteration is given by $N_{ts_i}$ and $N_t$ is the number of targets.

## 2.1 Initialization

We began the simulations with a three part initialization process. For the first part of the process, we calculated the different vehicle parameters based on the desired endurance ratio ($E_r$), turning ratio ($T_r$), and area ratio ($A_r$), given in Chpt. 3. We calculated the turning radius ($R_t$) to be the turning ratio times the domain length ($L$).

$$R_t = T_r L \tag{2.3}$$

We calculated the sensing radius ($R_s$) given by Eq. 2.4 below.

$$R_s = \sqrt{\frac{L^2 A_r}{N_v \pi}} \tag{2.4}$$

Where $N_v$ is the number of vehicles. We set the communication radius ($R_c$) equal to the sensing radius.

$$R_c = R_s \tag{2.5}$$

We calculated the stall speed ($V_s$) by Eq. 3.10 below.

$$V_s = \sqrt{g R_t \tan(\phi)} \tag{2.6}$$

Where $\phi$ is the roll angle and $g$ is the acceleration due to gravity of 9.81 m/s. We calculated the maximum velocity ($V_{max}$) through the following equation discussed in Chpt. 3.

$$V_{max} = 2V_s \tag{2.7}$$

We then computed the length of each time step, $t$.

$$t = \frac{R_s}{n_{roll} V_{max}} \tag{2.8}$$

Where $n_{roll}$ is the number of look ahead steps. This calculation comes from the Look Ahead ratio discussed in Chpt. 3. Lastly, we calculated the total simulation time, T.

$$T = \frac{E_r L^2}{2V_{max} R_s} \tag{2.9}$$

Where $E_r$ is the endurance ratio discussed in Chpt. 3.

For the next part of the initialization, we initialized the vehicle starting locations. As stated previously, we set the number of vehicles equal to a perfect square. This allowed the equal distribution of vehicles among the grid. We set the heading of the vehicle to a random value between 0 and $2\pi$. The below process shows the function we used for the initialization of the vehicles.

**Algorithm 2** Initialization of Vehicles

**procedure** INITIALIZE($N_v, L, l_c$)
    rowvehicle $\leftarrow \sqrt{N_v}$
    spacing $\leftarrow L/(\text{rowvehicle} + 1)$
    position $\leftarrow (N_v \text{ by } 3)$ matrix
    count $\leftarrow 1$
    **for** $i = 1$ to rowvehicle **do**
        **for** $j = 1$ to rowvehicle **do**
            position(count,:) $\leftarrow$ [getrandom $* 2 * \pi$, $i *$ spacing, $j *$ spacing]
            count $++$
        **end for**
    **end for**
    **return** Position
**end procedure**

We recorded the position of each vehicle in a $N_v$ by 3 matrix. Each row of the matrix contained the heading ($\psi$), east location ($p_e$), and north location ($p_n$) of a vehicle. We updated these values after each optimization.

For the next and final part of the initialization process, we set the grid point centers. The centers were evenly spaced by the length $l_c$. We set the grid values to an initial value of 0, meaning the grid points had not yet been explored. The code is shown in appendix A.1.

**Algorithm 3** Initialization of Grid Points

**procedure** INITIALIZE$(L, l_c, N_v)$
    $N_{rows} \leftarrow L/l_c$
    $N_g \leftarrow N_{rows}^2$
    count $\leftarrow 1$
    **for** $i = 1$ to $L$ increments of $l_c$ **do**
        **for** $j = 1$ to $L$ increments of $l_c$ **do**
            gridposition(count,:) $\leftarrow [j, i]$
            count$++$
        **end for**
    **end for**
    gridvalues $\leftarrow N_g$ by $N_v$ matrix of zeros
    **return** gridposition, gridvalues
**end procedure**

## 2.2 Optimization

We used gradient-based optimization to determine the vehicles' optimal paths through varying the vehicles' roll angles. Upon analyzing initial testing with the multi-objective target tracking and grid search missions, we saw some vehicle behavior that helped us reach the conclusion that we needed to add velocity as a design variable in the optimization. The vehicles began tracking a target and two different outcomes would occur. The first situation we saw was a vehicle would begin tracking a target then begin a back-and-forth lawnmower-type pattern to continue to track the target. This occurred when the target's velocity was slower than that of the aerial vehicle. The other scenario we witnessed was when a aerial vehicle could not keep up with a target it initially began tracking. As a result, we added vehicle velocity to the optimization design variables. It was

assumed that the vehicle would fly at this optimized velocity for the entire planned path at that respective time step. See chapter 3 for more information on velocity bounds and calculations.

We created an objective function comprised of only grid searching. We also developed a multi-objective function consisting of both grid searching and target tracking. The multi-objective optimization problem is stated below in Eq. 2.10.

$$\begin{aligned} \max \quad & \text{search reward} + \text{target reward} \\ \text{w.r.t} \quad & \phi_1 \ldots \phi_{n_{roll}},\ V \\ \text{s.t.} \quad & -\phi_{max} \leq \phi \leq \phi_{max} \\ & V_s \leq V \leq V_{max} \end{aligned} \tag{2.10}$$

Where $V_s$ is the vehicle stall speed, $V_{max}$ is the vehicle's maximum speed, and $\pm\phi_{max}$ is the bound for the roll angles. We calculated the stall speed through Eq. 2.6 above based on the vehicle's turning radius ($R_t$), see Chpt. 3. We set the maximum speed to two times the stall speed, Eq. 2.7. We further discuss the reward function in section 2.4 for the grid search reward and section 2.5 for the multi-objective target tracking and grid search reward.

We began by initially using MATLAB's function fmincon. However, to increase computational efficiency we switched to Julia and used SNOPT, which stands for Sparse Nonlinear OPTimizer. The use of SNOPT was made available for Julia by the repository Snopt.jl, a Julia interface to SNOPT, which was created by Dr. Andrew Ning and Taylor McDonnell, a PhD candidate in the FLOW Lab[1].

Developers designed Julia as a programming language built for high performance. Julia demonstrates performance comparable to statically-typed languages but is also a flexible dynamic language. In fact, the performance is close to languages such as C and C++ while still having the benefits of a dynamic language[2].

Researchers use SNOPT for constrained optimization. SNOPT uses sequential quadratic programming to find the optimal solution of a linear or nonlinear function with variable bounds

[1] https://github.com/byuflowlab/Snopt.jl
[2] https://docs.julialang.org/en/v1/

and sparse linear or nonlinear constraints. SNOPT calculates the search directions from quadratic programming subproblems[3]. See appendix A.8 for the optimization function.

### 2.2.1 Receding Horizon Controller

Since vehicles could potentially receive new information at every time step, we used a receding horizon controller [9–11, 13]. A receding horizon controller involves solving a constrained optimization problem repeatedly using future predicted costs or rewards with a moving time horizon [4]. This type of controller is beneficial for problems which continuously receive new information.

To implement the receding horizon controller, we optimized a vehicle's full path out to its sensing radius. However, we only directed the vehicle to proceed one step before re-optimizing out to the sensing radius again with the new information received from vehicles within its communication radius.

### 2.2.2 Automatic Differentiation

We used forward mode automatic differentiation through the use of the Julia package ForwardDiff. We implemented this package to calculate the gradients of the reward functions discussed in Sec. 2.4 and 2.5. We considered using the Julia package ReverseDiff to use reverse mode automatic differentiation. Typically, reverse mode differentiation is more computationally efficient when the number of outputs is less than the number of inputs. However, the documentation stated that "ForwardDiff is often faster than ReverseDiff for lower dimensional gradients" when the input dimension length is less than 100[5].

Automatic differentiation utilizes the fact that any computer function is a sequence of primitive functions. Each of these primitive functions' derivatives can be taken relatively easily. Automatic differentiation uses the chain rule on the primitive functions to calculate the total derivative of the more complex function. Forward and Reverse differentiation are based on how the chain rule is applied. Forward differentiation involves sequentially solving derivatives of operations in

[3] https://web.stanford.edu/group/SOL/snopt.htm
[4] https://web.stanford.edu/ boyd/papers/code_gen_rhc.html
[5] https://github.com/JuliaDiff/ReverseDiff.jl

terms of their parent functions[6]. See the following example. Given Eq. 2.11 below, we desire to find the derivative of the following function.

$$f(x,y) = x + \cos(y)\sin(x) \tag{2.11}$$

To take the derivative of this function we proceed as follows. First, we simplify the calculations to a series of primative operations.

$$w_1 = x$$

$$w_2 = y$$

$$w_3 = \sin(x)$$

$$w_4 = \cos(y)$$

$$w_5 = w_4 * w_3$$

$$w_6 = w_1 + w_5$$

We then take the derivative of each of these nodes with respect to some variable, $t$.

$$\frac{dw_1}{dt} = \frac{dx}{dt}$$

$$\frac{dw_2}{dt} = \frac{dy}{dt}$$

$$\frac{dw_3}{dt} = \frac{d}{dt}\sin w_1 = \cos w_1 * \frac{dw_1}{dt}$$

$$\frac{dw_4}{dt} = \frac{d}{dt}\cos w_2 = -\sin w_2 * \frac{dw_2}{dt}$$

$$\frac{dw_5}{dt} = \frac{d}{dt}(w_4 w_3) = w_4 * \frac{dw_3}{dt} + w_3 * \frac{dw_4}{dt}$$

$$\frac{dw_6}{dt} = \frac{d}{dt}(w_1 + w_5) = \frac{dw_1}{dt} + \frac{dw_5}{dt}$$

To calculate the derivative of the function with respect to x, we can plug in $t = x$. We can initialize $dw_1/dt$ equal to 1 and $dw_2/dt$ equal to 0. These are called the seed values. By choosing these

[6]http://www.columbia.edu/ ahd2125/post/2015/12/5/

seeds, $dw_6/dt$ will calculate the value of the derivative with respect to the x value. Vice versa, we can plug $t = y$ to get the derivative with respect to y.

This is the process used for a specific function. The process is made automatic through using a set of rules to translate a set of expressions into a program to calculate the derivatives. For forward-mode differentiation, the order of the program is preserved and the derivative of the expressions are evaluated sequentially from the first expression onwards [7].

## 2.3 Vehicle Motion Model

We modeled the UAVs as point masses using a simple coordinated-turn approach. The turning radius is given by Eq. 2.12. To simplify analysis, we assumed the sideslip angle, $\beta$, is zero and that there was no wind. Therefore, the equation for the change in yaw, $\dot{\psi}$, is given by Eq. 2.13. The inertial north and east position of the vehicle are then given by Eq. 2.14 and 2.15 respectively.

$$R_t = \frac{V_g^2 \cos\gamma}{g \tan\phi} \tag{2.12}$$

$$\dot{\psi} = \frac{g}{V_{cr}} \tan\phi \tag{2.13}$$

$$\dot{p_n} = V_{cr} \cos\psi \tag{2.14}$$

$$\dot{p_e} = V_{cr} \sin\psi \tag{2.15}$$

Where $V_g$ is the velocity of the vehicle with respect to the ground, $\gamma$ is the flight path angle, $\phi$ is the roll angle in examination, and $V_{cr}$ is the cruise velocity. See appendix A.2 for the motion model function.

## 2.4 Search Reward

We used the roll angles and velocity of the vehicles as the design variables used in the optimization. The reward function used Julia's DifferentialEquations.jl package [8] to solve equations

[7] https://rufflewind.com/2016-12-30/reverse-mode-automatic-differentiation
[8] https://docs.juliadiffeq.org/latest/

2.13, 2.14, and 2.15 for the heading angle, north position, and east position after each roll angle for the current vehicle. We then calculated the distance from the vehicle's entire planned path to every grid center. After tabulating these distances, we recorded the minimum distance to each reward point from the vehicle over the entire path.

The grid points contained an associated reward value, $r_i$, which began at a minimum value of 0. A value of 0 meant the point had not yet been seen and vehicles should have proceeded to that location to increase the point's value. Rather than provide a constant reward value across the cell, we used a Gaussian distribution to allow the reward to vary continuously and permit the use of gradient-based optimization. Fig. 2.1 shows the value associated with the distance between a vehicle and a grid center. The realized reward was the largest value obtained in each cell. See Eq. 2.16 for how we updated the grid value of the $ith$ cell.

$$r_i = e^{-\frac{d_i^2}{2\sigma^2}} \tag{2.16}$$

The distance between a vehicle and the $ith$ reward point is given by $d_i$ . We set the value of $\sigma$ to the vehicle's sensing radius divided by three because we wanted most of the reward contained within the sensing radius. In addition, the cell reward increased significantly as the vehicle approached that cell's center. We assigned a reward array to every vehicle that contained a reward value for each grid point. This reward array only reflected the movement of the current vehicle unless communication was simulated with another vehicle. We only updated the reward for a specific grid point if the vehicle came closer than it had previously. Thus, we compared the new Gaussian grid values to the previous values for that respective vehicle. If a grid point received a higher reward based on the vehicle's path, we updated the grid point's value. If this was not the case, we maintained the old point's value. We summed the maximum value for all grid points among all vehicles for the final reward.

The total reward for the $jth$ iteration for a specific vehicle is calculated as:

$$J_j = \sum_{i=1}^{N_g} r_{i,j} \tag{2.17}$$

The number of grid cells is represented as $N_g$ and $r_i$ represents the reward of the $ith$ cell resulting from the potential flight path. See appendix A.3 for the reward function code.

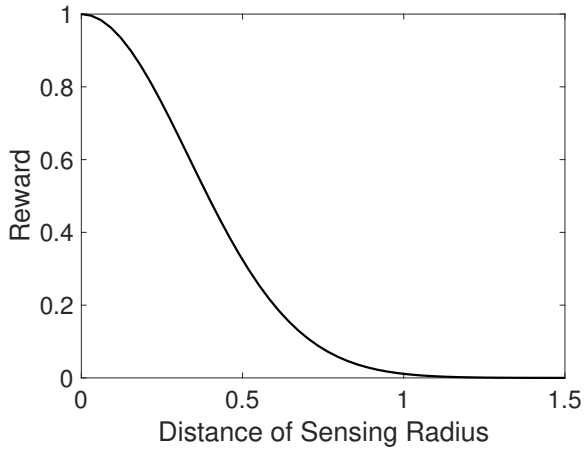



Figure 2.1: The reward distribution as a function of the distance, $d_i$ compared to the sensing radius, $R_s$.

## 2.5 Target Tracking

We combined searching a grid space and finding and tracking a target to create multi-mission level objectives. We initialized five targets to begin at predetermined locations and random orientations within the grid. We initialized the heading angle of each vehicle to any angle from 0 to $2\pi$. The locations were set so one target began in the middle of the grid and the other four were in each corner of the grid, positioned one third of the domain length away from the closest boundary. In addition, we initialized the targets with an initial velocity of 10 $m/s$. The initialization code is shown in appendix A.5. Initially, we began with the vehicles beginning in random locations and with random velocities. However, this gave too much variability in the results for the pareto front, discussed in Chpt. 6. At each time step, we changed the targets' headings by a small random value. We calculated the new locations by using the respective target velocities and new heading angles.

$$\psi_{t_{i,j}} = \psi_{t_{i-1,j}} + rand(1)\frac{\pi}{6} - \frac{\pi}{12} \tag{2.18}$$

$$Te_{i,j} = Te_{i-1,j} + tV_{t_j}\cos(\psi_{t_{i,j}}) \tag{2.19}$$

$$Tn_{i,j} = Tn_{i-1,j} + tV_{t_j}\sin(\psi_{t_{i,j}}) \tag{2.20}$$

We used eq. 2.18 to calculate the new target heading angle, $\psi_{t_{i,j}}$, for the $ith$ iteration and $jth$ target. We calculated the new heading angle as the previous step's heading angle plus a random

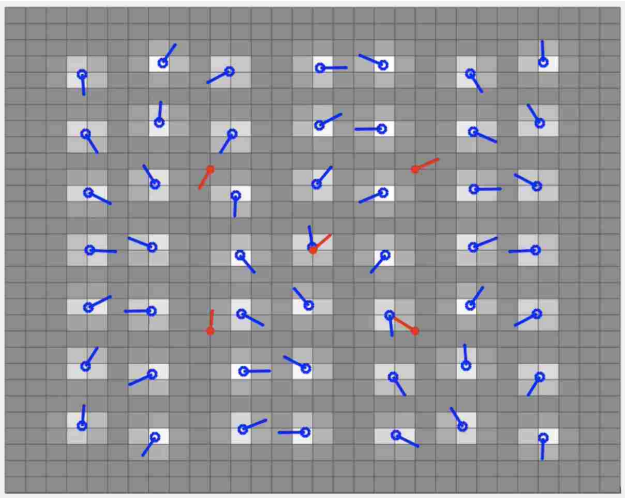


Figure 2.2: The targets began in evenly-spaced locations with random headings. The target locations are represented by the read circles with heading directions shown by the attached red quivers.

number between 0 and 30 degrees minus 15 degrees. This enabled the new heading to vary from the previous heading by a value of plus or minus 15 degrees. We calculated the new east location, $Te_{i,j}$, of the $jth$ target for the $ith$ iteration from eq. 2.19. We calculated the new north location, $Tn_{i,j}$, of the $jth$ target for the $ith$ iteration from eq. 2.20. In the above equations, $t$ is the time step, $V_{t_j}$ is the velocity of the $jth$ target, and $\psi_{t_{i,j}}$ is the current heading angle of that target. Appendix A.6 shows the function used to update the locations of the targets.

We used a multi-objective reward where we computed the target reward similarly to the grid search reward. The optimization used each roll angle, $\phi$ and the vehicle's velocity, $V$, to determine the ending location of the vehicle after each time step from equations 2.13, 2.14, and 2.15. We then calculated the distance from each of these points to each target. We used the minimum distance along the entire path to each target to calculate the reward from the Gaussian distribution given by Fig 2.1 and eq. 2.16. We then summed the values from this Gaussian distribution for all five targets for the final target reward. Reward was only given if a target was within a vehicle's sensing radius to simulate that the vehicles did not know the locations of the targets.

We varied the weighting of the target reward to determine the change in performance of both the grid search and target track missions with the different weightings. The reward was normalized by the maximum attainable reward. Thus, the total reward for the $jth$ iteration from

the $kth$ vehicle is:

$$J_{j,k} = \frac{W_g \sum_{i=1}^{N_g} r_{i,j} + W_t \sum_{i=1}^{N_t} r_{t_i}}{W_g N_g r_m + W_t N_t r_m} \tag{2.21}$$

The number of grid cells is represented as $N_g$ and $r_{i,j}$ represents the predicted reward of the $ith$ grid cell for the $jth$ iteration. The number of targets is given by $N_t$ and $r_{t_i}$ is the predicted reward that vehicle receives from the position relative to the $ith$ target. $W_t$ is the weighting we used for the target track mission and $W_g$ is the weighting we used for the grid search mission. The maximum attainable reward for each grid point and target is represented as $r_m$ and was equal to 1.0. We assumed the vehicles did not know the locations of the targets until a target entered the vehicle's sensing radius. See appendix A.7 for the updated reward function. Fig. 2.3 displays the mission of vehicles both tracking targets and searching a grid area.

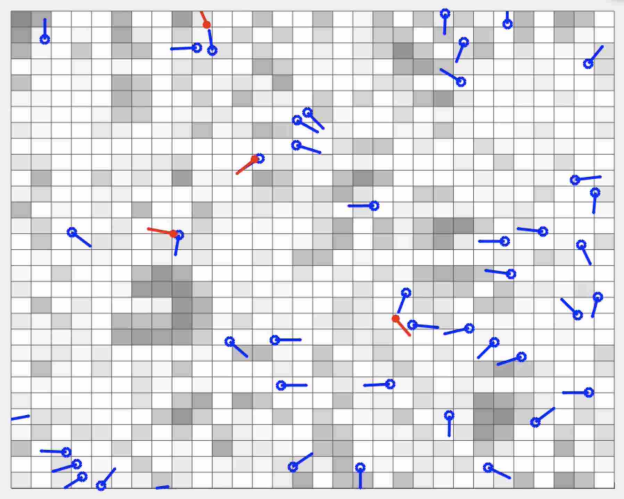


Figure 2.3: The vehicles both searched the grid and tracked targets. Note: one of the targets went out of bounds and is not pictured

## 2.6 Multistart

Additionally, because gradient based optimization can get stuck in local maximums or minimums, to increase the chances of finding the global maximum we used a multistart. To determine how many different starting locations were needed we performed a convergence study, iteratively increasing the number of initial guesses. The initial guesses were composed of the roll angles. We set all angles of each guess to the same value. We evenly spaced the values of the guesses between

the bounds of the roll angles. See the below pseudocode included in algorithm 4. The number of guesses is given by $n_g$. The number of roll angles we used in the optimization is given by $n_{roll}$. It

**Algorithm 4** Multistart

```
procedure MULTISTART
    spacing ← 60/(n_g − 1)
    guesses ← (n_g by n_roll)
    guesses[1, :] ← −30π/180
    for i in 2 to n_g do
        guesses[i, :] ← −30 + spacing(i−1)π/180
    end for
end procedure
```

was surprising to see the number of multistarts had little effect on performance. The performance plateaued and stayed relatively constant after three initial guesses. Therefore, for the simulations we used three initial starting guesses with all roll angles of the respective guess set to either -30, 0, or 30 degrees.

### 2.7 Data Sharing

After all optimizations were completed, we performed calculations to determine which vehicles would communicate with one another. This communication was based on if the location of a vehicle was within the specified sensing radius, $R_s$, of another vehicle. We created a three dimensional matrix, $M$. If the $jth$ vehicle was within the specified communication radius of the $kth$ vehicle we assumed perfect communication would occur. If this was the case, we updated all of the $i$ rows in the $jth$ column of the $kth$ dimension of matrix $M$ to that of the grid cells from the predicted path of the $jth$ vehicle. See the below piecewise defined function for clarification.

$$M_{i,j,k} \begin{cases} r_{p_{i,j}} & d_{j,k} < R_c \\ r_{i,k} & d_{j,k} \geq R_c \end{cases} \tag{2.22}$$

If the distance between the $jth$ and $kth$ vehicle, $d_{j,k}$, was smaller than the specified communication radius, $R_c$, communication would occur. If this was the case, the $ith$ row of the $jth$ column

of the $kth$ dimension of the M matrix was equal to the predicted $ith$ grid value of the $jth$ vehicle, $r_{p_{i,j}}$. If the distance was larger than the communication radius, the $ith$ row of the $jth$ column of the $kth$ dimension of the M matrix was equal to the actual $ith$ grid value of the $kth$ vehicle, $r_{i,k}$. Note that only the vehicle's predicted path was communicated and not any of the vehicle's past path information. We used this $M$ matrix in the next path optimization. We then used the maximum grid values achieved across all rows in the $kth$ dimension as the grid values for the $kth$ vehicle's optimization. See algorithm 5 below for psuedocode and appendix A.4 for the actual code used.

**Algorithm 5** Vehicles' sharing grid values

**procedure** COMMUNICATION
  **for** $k$ in $N_v$ **do**
    **for** $j$ in $N_v$ **do**
      **if** $\sqrt{(p_{e_j} - p_{e_k})^2 + (p_{n_j} - p_{n_k})^2} < R_c$ **then**
        $M[:, j, k] \leftarrow r_p[:, j]$.
      **else**
        $M[:, j, k] \leftarrow r[:, k]$.
      **end if**
    **end for**
  **end for**
**end procedure**

# CHAPTER 3. NONDIMENSIONALIZATION OF VEHICLE AND SEARCH AREA PROPERTIES

## 3.1 Non-dimensionalization

Since search missions vary by size, duration, and number and type of vehicles, we wanted to find the relationships between vehicle properties and search performance. In addition, determining the effect of a single mission parameter on search performance can be difficult to find because the performance also depends on other parameters. Because of this, we non-dimensionalized the properties and analyzed how the search performance, given by Eq. 2.1, changed with varying the non-dimensional numbers. This generalized the results and gave us insight into which parameters are critical as opposed to parameters that may not have an effect on performance and can be relaxed. The properties included turning radius, communication radius, sensing radius, maximum velocity, stall speed, time step, flight time, number of vehicles, acceleration due to gravity, domain length, and grid cell length. We had seven main non-dimensional parameters.

### 3.1.1 Endurance Ratio

The first parameter we define is the endurance ratio and is shown below.

$$E_r = \frac{V_{max} T 2R_s}{L^2} \tag{3.1}$$

where

$$T = n_t t \tag{3.2}$$

Where $n_t$ is the total number of time steps the vehicles fly throughout the simulation and $t$ is the length of one time step. The maximum velocity is given by $V_{max}$, the sensing radius is $R_s$, and the domain length is $L$. This ratio gives the amount of times a vehicle can cross the domain to the amount of times required to sense the entire grid with a given sensing radius if traveling in a

straight lawnmower-type pattern. The maximum distance a vehicle can travel is given in Eq. 3.3.

$$V_{max}T \tag{3.3}$$

Dividing this value by the length of the domain, $L$, gives the number of times the vehicle can fly across the domain length. This is modeled by Eq. 3.4 below and show in Fig. 3.1.

$$\frac{V_{max}T}{L} \tag{3.4}$$

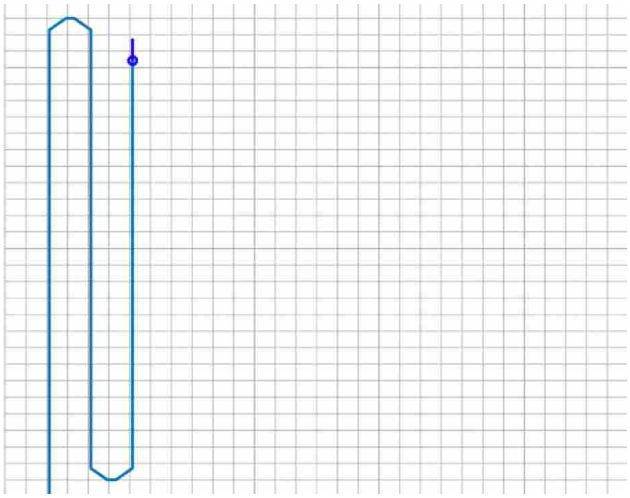


Figure 3.1: A depiction of the parameter in Eq. 3.4. How far a vehicle can travel based on its endurance ($T$) and maximum velocity ($V_{max}$) compared to domain size ($L$).

We then divide this number by the number of times it takes for a vehicle to travel back and forth across the domain to see all the grid. This was based on the ratio of the domain length to the sensing diameter. This number is given by Eq. 3.5. Figure 3.2 gives a depiction of this parameter.

$$\frac{L}{2R_s} \tag{3.5}$$

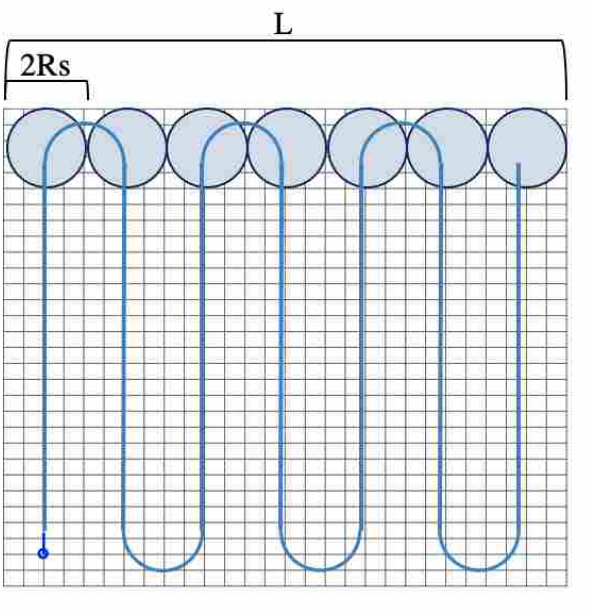



Figure 3.2: A depiction of the parameter in Eq. 3.5. How many times a vehicle needs to travel back and forth across the domain to sense all of the grid based on the sensing radius, $R_s$

Vehicles flying for a longer duration can potentially search more of the grid and see unexplored areas. The ratio demonstrates how suitable the endurance of an aircraft is based on the sensing properties for a particular domain size. Therefore, vehicles searching large domain sizes with poor sensing properties will require a high endurance. To test a wide range of ratios we ranged the endurance ratio from 0.025 to 1.0. A value of 1.0 meant that a vehicle could potentially sense all of the grid based on the endurance and sensing properties. A value of 0.025 meant a vehicle could only sense 2.5% of the entire grid throughout the duration of the flight.

### 3.1.2 Area Ratio

The following parameter is the area ratio and is defined as:

$$A_r = \frac{N_v R_s^2 \pi}{L^2} \tag{3.6}$$

This gives the ratio between the amount of area sensed by all vehicles, $N_v$, to the total area of the domain. For the analysis we varied this parameter from 0.2 to 1.0. A graphic of the parameter is shown in Fig. 3.3 below.

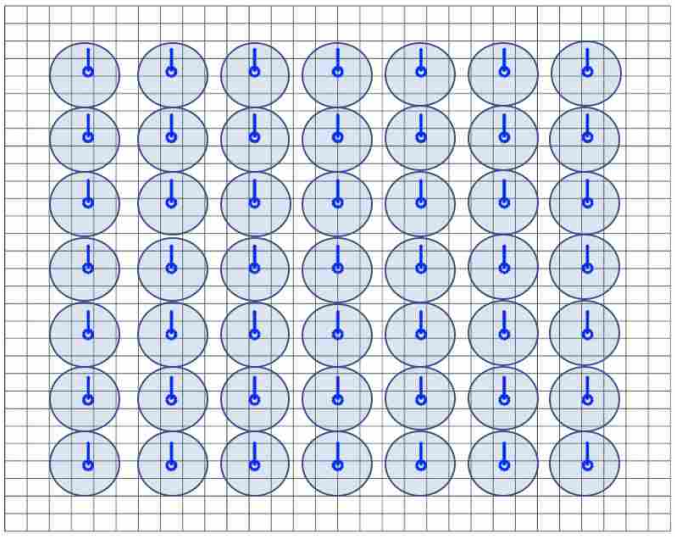


Figure 3.3: A depiction of the parameter in Eq. 3.6. How much of the grid the vehicles can sense at a single time step. The blue circles represents the numerator of Eq. 3.6.

### 3.1.3 Look Ahead Ratio

If we set a vehicle's event horizon further than the sensing radius, there would be insufficient information to plan to this distance. On the other hand, if we had the vehicles planning to a distance less than the sensing radius we would not be utilizing all possible information, possibly resulting in less optimal path optimizations. Therefore, the look ahead ratio gives the fraction of the sensing radius distance to how far the vehicle can plan.

$$L_r = \frac{R_s}{V_{max} t n_{roll}} \tag{3.7}$$

We defined the look ahead ratio as the sensing radius, $R_s$ divided by the maximum velocity, $V_{max}$, times the duration of one time step, $t$, and the number of planned roll angles or look ahead steps, $n_{roll}$. This look ahead ratio should be equal to one, enabling vehicles to plan out to what they can sense.

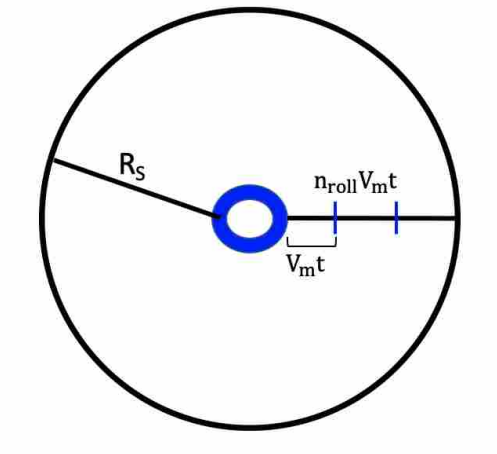



Figure 3.4: A depiction of the parameter in Eq. 3.7. Each section labeled $V_m t$ represents a different roll angle step. In this figure, the vehicle planned a path going east. The look ahead ratio compares the length of the sensing radius ($R_s$) to the length of the planned path ($n_{roll} V_{max} t$).

### 3.1.4 Communication Ratio

The next parameter we used is the communication ratio and is defined as:

$$C_r = \frac{R_c}{R_s} \tag{3.8}$$

It shows how far the vehicles can communicate compared to what the vehicles can sense. We kept this value constant at a value of 1 for the non-dimensional properties study. This allowed us to examine the other vehicle properties. However, we do vary this ratio between 1 and 2 for the communication studies.

### 3.1.5 Turning Ratio

The next parameter we used is the turning ratio and is defined as:

$$T_r = \frac{2R_t}{L} \tag{3.9}$$

It is a ratio of the vehicle's turning radius, $R_t$, to the domain size, $L$. It represented how many completed turns a vehicle could make within the domain space. The turning radius is defined in

Eq. 3.10.

$$R_t = \frac{V_s^2}{g \tan \phi} \tag{3.10}$$

A graphic of this parameter is shown in Fig. 3.5. To determine the effect the turning ratio had on vehicle performance, we varied Eq. 3.9 from 0.05 to 0.45 to test a wide range of turning radii by changing $V_s$, the stall speed. A turning ratio of 0.05 allowed 20 turns to be completed in the domain while a turning ratio of 0.45 allowed around 2 completed turns. This range allowed us to see the full effect turning ratio had on search performance.

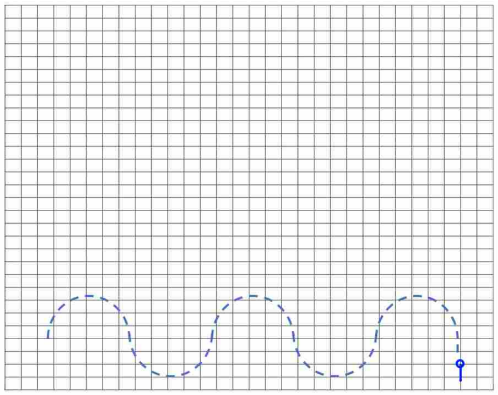


Figure 3.5: A depiction of the parameter in Eq. 3.9. How many times a vehicle can turn with a specified turn radius ($R_t$) within the domain size ($L$).

### 3.1.6 Velocity Ratio

We created the velocity ratio to keep the ratio of the maximum velocity, $V_{max}$ to the stall speed, $V_s$, consistent when varying other parameters. The velocity ratio is defined in eq. 3.11.

$$V_r = \frac{V_{max}}{V_s} \tag{3.11}$$

We kept this ratio at a constant value of two for all simulations. Thus, the maximum speed, $V_{max}$, was always twice the amount of the stall speed, $V_s$.

### 3.1.7 Misc.

The parameter:

$$\frac{l_c}{L} \tag{3.12}$$

represents the length of a grid cell to the length of the domain size. This parameter was not physically meaningful and was only an artifact of the discretization. Therefore, we kept it constant across the different studies.

There were seven non-dimensional numbers we created from the parameters used in the simulations. For simplicity, we included table 3.1 below to show the four non-dimensional ratios we discuss in the results section below.

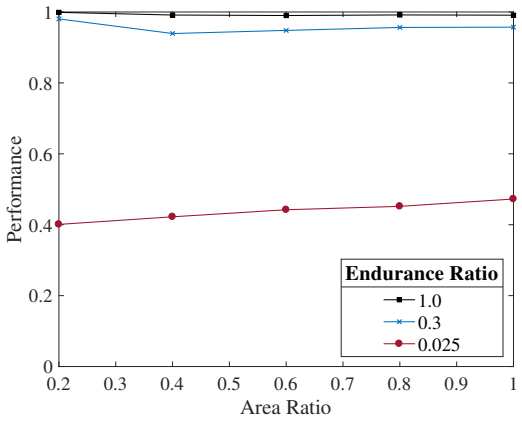
Table 3.1: Non-dimensional parameters studied

| Endurance Ratio | Area Ratio | Turning Ratio | Communication Ratio |
|---|---|---|---|
| $\frac{V_m T 2R_s}{L^2}$ | $\frac{N_v R_s^2 \pi}{L^2}$ | $\frac{2R_t}{L}$ | $\frac{R_c}{R_s}$ |

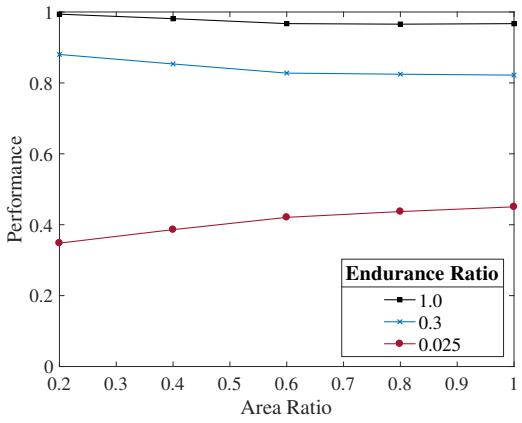
## 3.2 Results

There were seven primary non-dimensional parameters of interest. We kept the communication ratio constant at a value of 1, look ahead ratio constant at a value of 5, and velocity ratio constant at a value of 2 to reduce the number of necessary optimizations. We also kept the length of a grid cell to the length of the domain size constant. Therefore, we varied three parameters across a range of combinations and multistarts. We ran simulations across five different area ratios, five different endurance ratios, and five different turning ratios. This resulted in a total of 375 total simulations. Figs. 3.6, 3.7, and 3.8 all use the same data gathered from these optimizations. The only difference between the plots is what ratio is used on the x-axis and what parameter is changed from plot to plot.

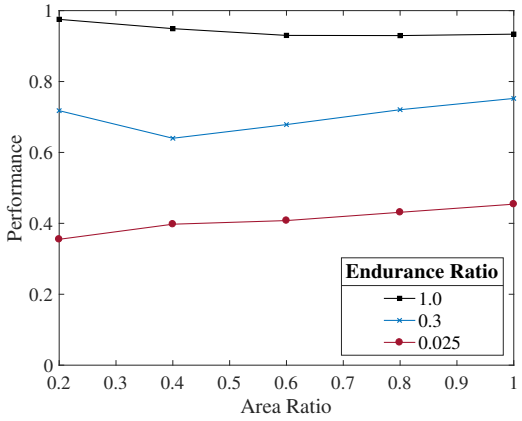
Each plot in Fig. 3.6 shows the performance verses area ratio. Each line represents a different endurance ratio and each plot is plotted at a different turning ratio. We can see from these figures that changing the area ratio actually does not have a large impact on performance. Furthermore, upon analyzing Fig. 3.6, we see that when vehicles have a high endurance, increasing the area ratio actually decreases performance. This is because the endurance ratio gives how many times the vehicles can cross the domain to how many domain crosses are needed based on the




(a) turning ratio = 0.05




(b) turning ratio = 0.25



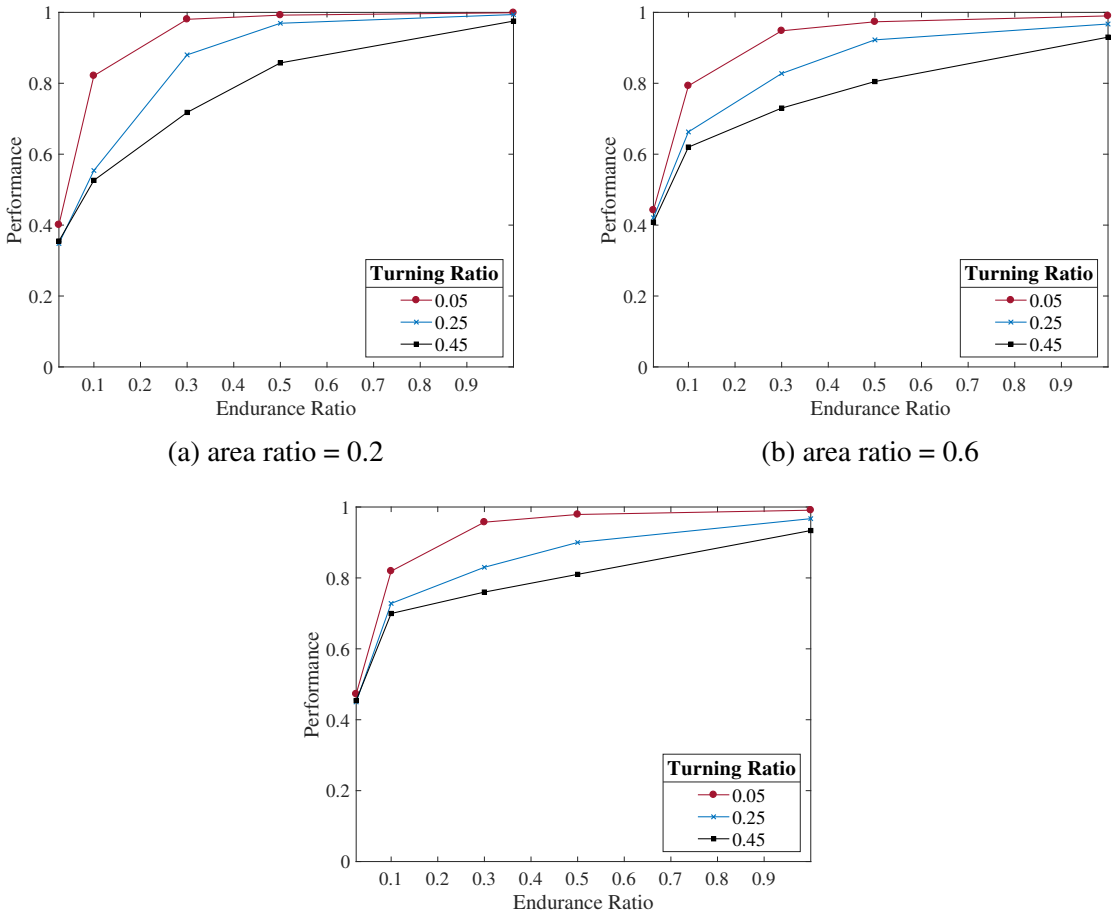

(c) turning ratio = 0.45

Figure 3.6: Each figure shows the performance verses area ratio. Each line represents a different endurance ratio and each different plot corresponds to a different turning ratio.

vehicles' sensing properties. Upon analyzing the endurance ratio in Eq. 3.1, we see that when there is a smaller area ratio, a vehicle needs to have more endurance to still see the same amount of area. As a result, the vehicles have more opportunity to search more of the grid area. In addition, since the grid point values are based on a Gaussian distribution, less value is given to points on the edge of the vehicles' sensing radii and more value is given to grid points closer to the location of the vehicle. Thus, vehicles are able to get closer to grid points when they travel longer and thus increase reward.

There is little improvement from increasing the endurance ratio from 0.3 to 1.0 when the turning ratio is 0.05, as shown in Fig. 3.6a. This means that performance plateaus and does not

improve. From analyzing all three plots in Fig. 3.6, we see that as the turning ratio increases, meaning the vehicles lack agility, the endurance ratio becomes more important at high endurance ratios. We can see as the turning ratio increases, the gaps between the endurance ratios of 0.3 and 1.0 enlarge. However, at the endurance ratio of 0.025, the performance values stay relatively constant among the plots with increasing the turning ratio. We further discuss the endurance ratio trends and provide figures with the endurance ratio on the x-axis in Fig. 3.7.




(a) area ratio = 0.2

(b) area ratio = 0.6

(c) area ratio = 1.0

Figure 3.7: Each figure shows the performance verses endurance ratio. Each line represents a different turning ratio and each different plot corresponds to a different area ratio.

In Fig. 3.7 the performance is plotted across the endurance ratio where the lines are plotted at different turning ratios. Each plot has a different area ratio. From Fig. 3.7, we see that the endurance ratio has a large impact on the performance. Increasing the endurance ratio from 0.025

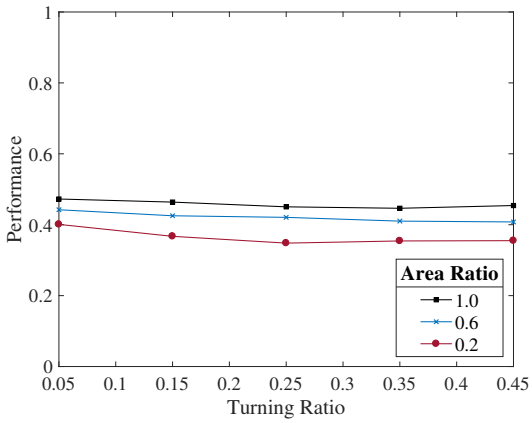
to 1.0 increases the amount of sensed domain by as much as 60% for select cases. In addition, vehicles can perform well at any area ratio with good endurance. Similar to the results from Fig. 3.6, it can been seen from Fig. 3.7 that the turning ratio alters the mission performance outcome. Figs. 3.7b and 3.7c demonstrate that vehicle agility is important for good performance even if the vehicles can sense a large area of the grid. Fig. 3.7 shows at any area ratio, the vehicles can coordinate and search 100% of the grid if the turning ratio and endurance ratios are high. Thus, it seems the endurance ratio and turning ratio are more important than the area ratio.

In addition, each plot shows that the performance plateaus after a certain endurance ratio and performance does not further improve. For low turning ratios, this plateau happens sooner, as shown by the red line in each figure. As we analyze the different figures, we also see the general slope of the lines from each figure decreasing as the area ratio increases. This is because the performance starts higher at a small endurance ratio with a large area ratio. Thus, the performance rate of change is smaller with a higher area ratio. Also, there is a steep increase in performance from changing the endurance ratio from 0.025 to 0.1 with all area ratios. The rate of change of the performance then decreases past this point.

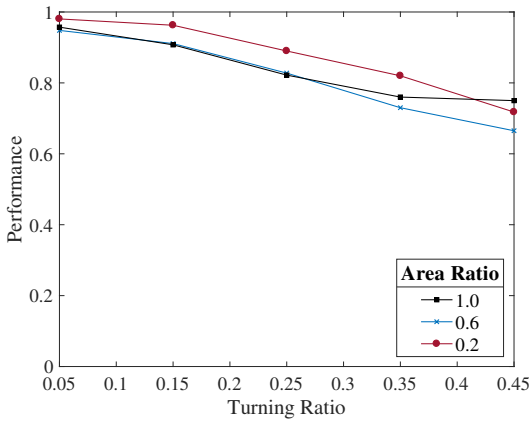
When vehicles lack endurance, it appears that the amount of area the vehicles can sense is more important than their agility. When the endurance ratio is 0.025, the performance improves by approximately 7% from increasing the area ratio from 0.2 to 1.0. This makes sense because if vehicles are unable to travel very far, it is more important for the vehicles to sense a large amount of the grid with their limited movement.

In Fig. 3.8, the performance ratio is plotted across the turning ratio with each line corresponding to a different area ratio. Each figure has a different endurance ratio. Note that the lower the turning ratio, the better the vehicle's agility. From Fig. 4.2a there is little improvement for decreasing the turning ratio. However, the area ratio improves the performance. As shown from the previous plots, to overcome poor endurance the vehicles need to sense a large portion of the grid.

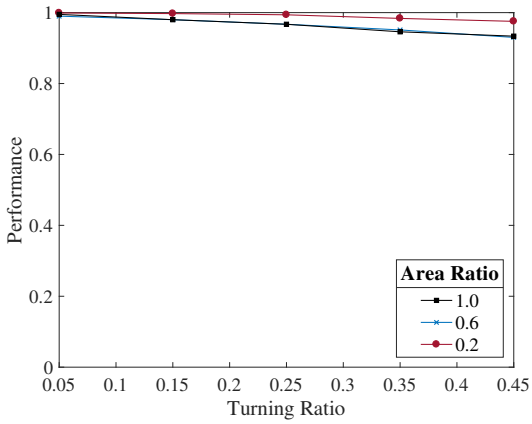
The other two plots, Figs. 4.2b and 4.2c, show the turning ratio has an effect on performance. The endurance ratio of 1 shows a small change in performance. This shows that when the endurance is high, the agility matters less. In addition, as stated previously, vehicles that sense less area at higher endurance ratios perform better. The reason was previously explained. In addition,




(a) endurance ratio = 0.025




(b) endurance ratio = 0.3




(c) endurance ratio = 1.0

Figure 3.8: Each figure shows the performance verses turning ratio. Each line represents a different area ratio and each different plot corresponds to a different endurance ratio.

Fig. 4.2b shows that at a high turning ratio, the area ratio becomes more important. Thus, when vehicles lack agility and have low endurance, sensing a larger area of the grid is more important.

## 3.3 Applying Results to Different Swarm

For the results obtained in section 3.2, we used 49 vehicles, 900 cells, a domain length of 1500, a grid spacing of 50, and 5 look ahead steps. See table 3.2 below.

The plots given in Figs. 3.6, 3.7, and 3.8, use the above parameters. However, you can use these plots to estimate the search performance for different domain areas and vehicle types. For example, let's suppose that the swarm now uses a domain size of 1000 meters instead of 1500

meters. Using the nondimensional parameters given earlier, we can calculate new values for the needed parameters to achieve the same levels of performance. For our simulations we had:

$$\frac{l_c}{L} = \frac{50}{1500} = 0.0333$$

To keep this value constant with the new domain length of 1000, the following calculations are performed.

$$0.033 = \frac{l_c}{1000}$$

Thus,

$$l_c = 33.33$$

Now, we will just use one plot as an example for the rest of the calculations. The same process can be repeated with any of the points on the plots. Let's assume that we know the new swarm are agile but are unable to sense a large portion of the grid. If we wanted to search 80% of the grid we can use Fig. 3.7a. From the plot, to achieve a performance of at least 80% with good agility (turning ratio of 0.05) and poor sensing properties (area ratio of 0.2), an endurance ratio of at least 0.1 is needed. Note that this is only the process we used to calculate the mission parameters. These individual parameters can be set to any number and calculated in any order as long as the previously discussed non-dimensional parameters stay constant. For example, there could be a different stall speed but the other values would need to change to keep the nondimensional values constant. To calculate the necessary swarm properties based on the number of vehicles and domain size, the following calculations are made. Using Eq. 3.6, we calculate the sensing radius the vehicles would require.

$$0.2 = \frac{N_v R_s^2 \pi}{L^2}$$

$$R_s = \sqrt{\frac{0.2L^2}{N_v \pi}}$$

$$R_s = \sqrt{\frac{0.2(1000)^2}{49\pi}}$$

$$R_s = 36.04m$$

In addition, since we kept the communication ratio, Eq. 3.8 equal to 1, we have the communication radius equal to the sensing radius.

$$R_c = 36.04m$$

The turning radius is calculated from the turning ratio of 0.05 and domain length of $1000.0m$.

$$R_t = \frac{0.05(1000.0)}{2}$$

$$R_t = 25.0m$$

Then, the stall speed is calculated through Eq. 3.10.

$$V_s = \sqrt{gR_t \tan(\phi)}$$

$$V_s = \sqrt{9.81(25)\tan(\frac{\pi}{6})}$$

$$V_s = \sqrt{9.81(25)\tan(\frac{\pi}{6})}$$

$$V_s = 11.90m/s$$

Now, using Eq. 3.11, we can calculate the maximum velocity.

$$V_{max} = 2V_s$$

$$V_{max} = 23.80m/s$$

After, we can use the look ahead ratio, given by Eq. 3.7 to calculate the new time step.

$$L_r = \frac{R_s}{V_m t n_{roll}}$$

$$t = \frac{R_s}{V_m L_r n_{roll}}$$

$$t = \frac{36.04}{23.8(1)(5)}$$

$$t = 0.30s$$

Finally, we calculate the total duration of the simulation through using the Endurance Ratio in Eq. 3.1.

$$E_r = \frac{V_m T 2R_s}{L^2}$$

$$T = \frac{E_r L^2}{V_m 2R_s}$$

$$T = \frac{(.1)1000^2}{23.8(2)(36.04)}$$

$$T = 58.29s$$

Thus, using these plots with a different domain size results in the need to change other parameters. Table 3.2 gives the different parameters needed to search 80% of the area with a turning ratio of 0.05, endurance ratio of 0.1, and area ratio of 0.02 for both a domain size of 1000 m and 1500 m.

Table 3.2: Parameter values used for cases with different domain lengths to achieve the same search performance

| $L$ | $l_c$ | $n_{roll}$ | $t$ | $R_t$ | $V_s$ | $V_m$ | $T$ | $R_s$ |
|---|---|---|---|---|---|---|---|---|
| 1500 | 50.0 | 5 | 0.25 | 37.5 | 14.57 | 29.15 | 107.09 | 54.07 |
| 1000 | 33.333 | 5 | 0.30 | 25.0 | 11.9 | 23.8 | 58.29 | 36.04 |

# CHAPTER 4. COMMUNICATION EFFECTS

## 4.1 Varying Amount of Communication

To test how the amount of information shared affected performance, we developed four communication cases in our original Matlab simulation: a decentralized no communication case, a decentralized case in which current and past locations were shared, and both a decentralized and centralized case where the current, past, and future locations were shared. In the centralized case the path of all vehicles were optimized in unison. In the decentralized cases each vehicle optimized its own path based on available information. For these cases, we assumed there was perfect communication and every vehicle instantaneously communicated with the other vehicles. In addition, a linear time-based decay was added. We subtracted a value of 0.01 from every grid reward point at each time step. As a result, cells would require continuous surveillance. We averaged the performance metric in Eq. 2.1 across all time steps to determine how the vehicles performed at searching throughout the entire simulation. Since the performance was averaged over the entire search mission and there was a time-based decay, the performance values were significantly lower than the non-dimensional results. We kept the vehicle properties constant among all four cases so the only factor that was changing from case to case was the amount of communication. The performance of all cases is provided in Table 4.1. Since the difference between communication strategies is difficult to see in static plots, we provided a video that shows the trajectories of all four communication cases: **https://youtu.be/GkR6KVicwPI**

For the no communication case, we optimized each vehicle's path as if it was the only vehicle performing the mission. For each vehicle's optimization, we set the grid to replicate only what each respective vehicle sensed. We only changed the grid cell values based on the location of each individual vehicle. Vehicles primarily proceeded in a straight line except when approaching a grid boundary. Since the vehicles did not communicate, each vehicle did not know which cells were previously explored. In addition, since the domain was large compared to the vehicles' sensing

radius, the potential reward was only decreased when a vehicle approached a grid boundary. As a result, the vehicles did not come close to the edge, so reward could be maximized without the search radius being outside of the domain. In addition, there was vehicle clustering with some cells not being searched at all.

In the second case, it was assumed that there was perfect communication of the vehicles' past and current locations within their sensing radius. For each optimization, we optimized the vehicles' paths based on grid values that reflected the motion of every other vehicle. We used a decentralized approach in which vehicles optimized paths based on the previous step of all other vehicles to replicate the behavior of the simultaneous movement of vehicles. The vehicles remained more dispersed, however, there was still vehicle clustering where vehicles searched the same cell.

In the third case, in addition to the communication of the current and past location, vehicles also communicated where they were planning on traveling. Each vehicle optimized based on grid values that were reflective of each vehicle's future planned path. For this case, we also used decentralized path planning. Again, we optimized each path based on the planned path from the previous time step of all other vehicles. This case had the most equal distribution of vehicles out of the decentralized cases.

In the final case case, a centralized approach was used where we used one function to optimize the path of all vehicles. Thus, this case represented in-sync decision making. This approach was very costly compared to the other cases. The centralized approach took approximately two and a half days to complete whereas the decentralized took approximately two hours. However, in the centralized case vehicles remained dispersed and never searched the same cell at the same time. However, Table 4.1 shows only a 1.7% increase from the decentralized to centralized case.

Table 4.1: Performance percentages of different communication cases

| **No Communication** | **Current Location** | **Full Decentralized** | **Full Centralized** |
|---|---|---|---|
| 35.9% | 42.2% | 44.5% | 46.2% |

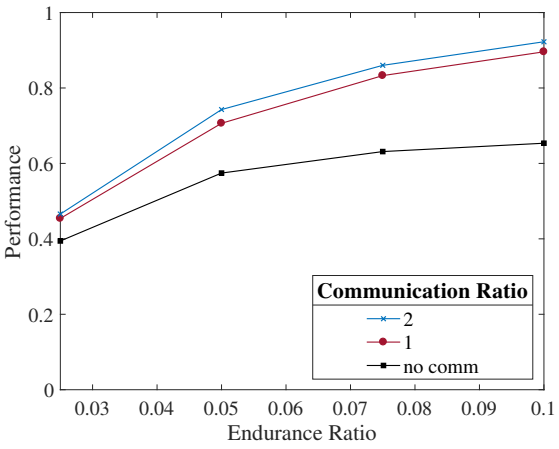
Sharing only the vehicles' current locations improved performance by 6.3%. The type of communication varied performance by 2.3% and the centralized case improved performance by 1.7%. This shows that any type of communication improves performance. However, the type of communication has less impact on the search performance. In addition, this shows centralized communication is, for the most part, not worth the large increase in computational cost.

## 4.2 The Effect of Swarm Properties on Importance of Communication

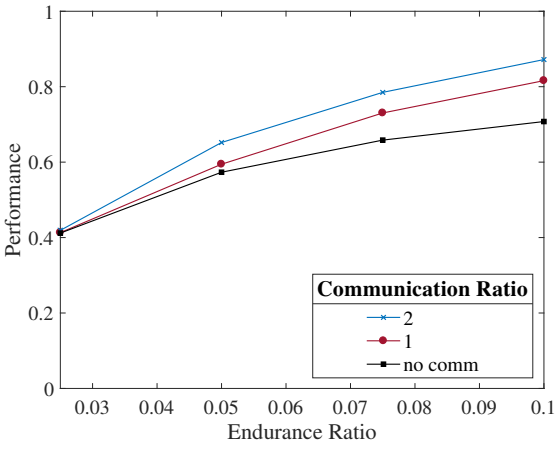
In addition, we ran the full communication case and the no communication case across a range of endurance ratios at two separate area ratios and turning ratios to determine when communication mattered the most. We only examined the lower end of these ratios because we inferred that communication would matter most when vehicles are unable to travel far or see the majority of the grid. Thus, vehicles would need to communicate to maximize the amount of seen grid space. In addition, we used both a communication ratio, given in Eq. 3.8, of 1 and 2. We performed these cases without the time decay and performance was the amount of the grid that was seen, given by Eq. 2.1. Below are the plots showing the performance results across endurance ratios from 0.025 to 1.0. We set the area ratio as 0.1 and 0.3 and the turning ratio was 0.025, 0.05, and 0.15.

The plots displayed in Fig. 4.1 show that when the turning ratio is small, communication is more important. This is because the vehicles are able to adjust their course more quickly and thus adapt to the incoming information from other vehicles. In contrast, when vehicles posses a high turning ratio, they are unable to alter the paths because of lack of agility. Thereby, the vehicles cannot make the necessary course adjustments to benefit from the sharing of information.

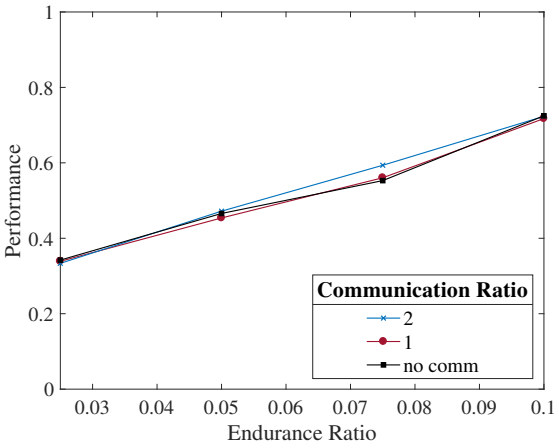
The plots from Fig. 4.2 show the performance verses endurance ratio at an area ratio of 0.3 and turning ratios of 0.025, 0.05, and 0.15. Similarly to the previous plots, when we set the turning ratio to a lower value, there was an increase in performance when communication was used in the swarm. However, there is a smaller difference in performance between the different communication cases at this larger area ratio. Thus, we can conclude that when vehicles have a limited range of sight, communication is more important. When there was a larger turning ratio of 0.15, there seems again to be no improvement for adding communication, which was previously discussed.




(a) turning ratio = 0.025

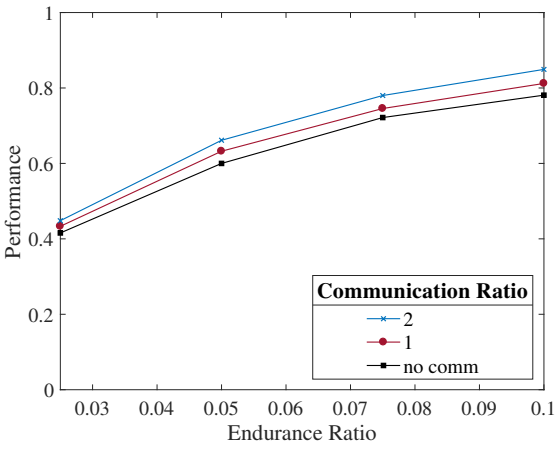



(b) turning ratio = 0.05

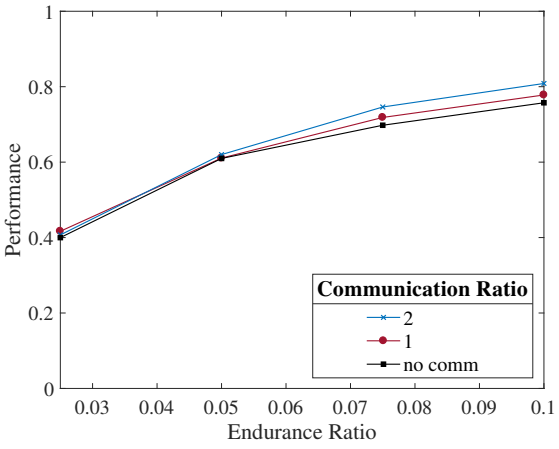



(c) turning ratio = 0.15

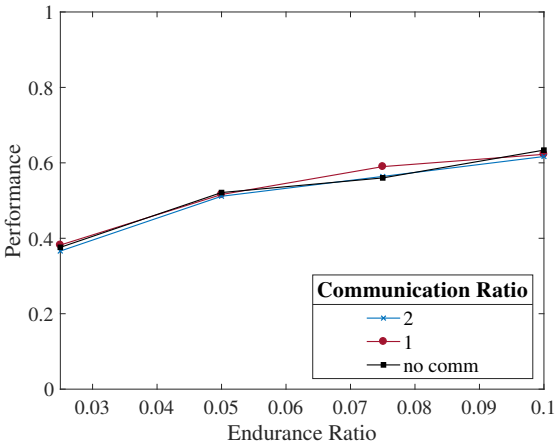
Figure 4.1: Each figure shows the performance verses endurance ratio for communication ratios of 1 and 2 and no communication. Each plot is a different turning ratio. The area ratio is 0.1.




(a) turning ratio = 0.025




(b) turning ratio = 0.05




(c) turning ratio = 0.15

Figure 4.2: Each figure shows the performance verses endurance ratio for communication ratios of 1 and 2 and no communication. Each plot is a different turning ratio. The area ratio is 0.3.

## CHAPTER 5. MULTI-OBJECTIVE PARETO FRONT

We created a Pareto front to find the trade-offs between the grid search performance and the target track performance. The reward function we used for the multi-mission objectives is given by Eq. 2.21. The weightings we used for the Pareto front plot are shown in table 5.1 below.

Table 5.1: The weighting of grid and target searching for the multi-objective reward

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $W_g$ | 1.0 | 0.9 | 0.8 | 0.7 | 0.6 | 0.5 | 0.4 | 0.3 | 0.2 | 0.1 | 0.75 | 0.5 | 0.25 | 0.0 |
| $W_t$ | 0.0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 9.25 | 9.5 | 9.75 | 1.0 |

We added additional points between the target weighting of 0.9 and 1.0 and the grid weighting of 0.1 and 0.0. This was because there was a large shift in the performance values when $W_g$ was assigned to 0.0 and $W_t$ was assigned to 1.0. We ran each weighting for multiple simulations with the targets beginning with random heading angles and using a random new heading direction for each time step. For every simulation, the targets began in the same initial location. We set each target's velocity to $10m/s$ since performance showed a large degree of variation based on the velocity of the targets. See Fig. 5.1 for the starting location orientation of the vehicles and targets. The total number of simulation runs for each weighting varied from 6 to 11 based on the amount of variation with each weighting combination. The Pareto front is shown in Fig. 5.2 below.

We obtained these results by using 5 targets. We expect results would change based on the number of targets and the velocities of the targets, but the general trend would remain constant. From the plot, we can see that increasing the target weighting dramatically alters the target performance. Target tracking performance increased by approximately 52% when adjusting the weight from 0.0 to 1.0. On the other hand, the grid performance shows little change until $W_g$ decreased to 0.0. Even then, the performance only dropped by approximately 10%. This is because even

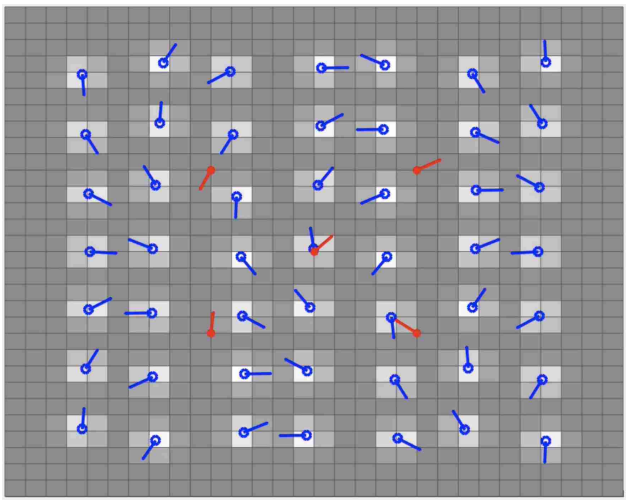


Figure 5.1: Initial starting locations of the UAVs and targets.

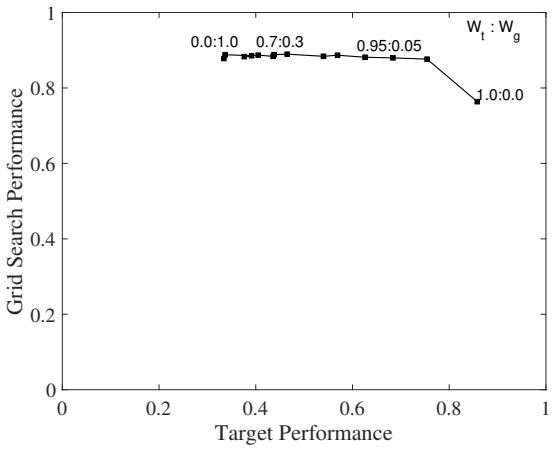



Figure 5.2: Pareto front showing the trade-offs between the weighting of target search and grid search. The labels give the ratio of the target search weighting to the grid search weighting.

when the vehicles are tracking the targets, they are still seeing and searching the grid. In addition, vehicles could only sense targets within their sensing radius. As a result, all vehicles that were not close to a target continued to search the grid, even if this reward was small. These results suggest to have good performance in both areas we should set $W_t$, the target weighting, to a large value and set $W_g$, the grid weighting, to a small value.

# CHAPTER 6. CONCLUSIONS AND FUTURE WORK

## 6.1 Conclusions

We sought to find how different types of communication and vehicle parameters affected the search performance for a swarm of UAVs and to create an efficient optimization process for multi-objective missions. Unlike other research which compares different optimization algorithms, we analyzed communication alone. Additionally, unlike previous research, we did not set the vehicle properties as constraints, but we varied these properties to determine the trends in search performance for different properties. We used non-dimensionalization so our results could be applied to a variety of scenarios. We also explored extending the grid search reward function to a multi-objective reward with the inclusion of target tracking.

These results convey that for maximum performance it is desirable to have a high endurance and low turning and area ratios. However, if the simulation has a high turning ratio, high performance can still be achieved if the endurance ratio ratio is sufficiently large. We determined that even if vehicles could only make approximately two turns within the domain length, 97% of the area could be seen by using a vehicle with good endurance. However, when a vehicle has poor endurance, having a larger area ratio is more important and can increase performance by approximately 7%. Although having a larger area ratio helped, vehicles with poor endurance ratios lacked good performance. This suggests that ensuring a vehicle's endurance is well-suited for the domain is especially vital. A poor area ratio and turning ratio can be overcome, but poor endurance always resulted in a lacking search performance. Thus, endurance matters much more than the other parameters.

The communication study showed that when vehicles communicated the current location and past history, performance improved by 6.3%. When vehicles communicated the planned future location as well, performance improved by 8.6%. When we used a centralized method in which vehicles used an in-sync decision making process, performance increased by 10.3% and the

optimizations were computationally expensive. This was only a 1.7% increased beyond the decentralized future location case. Therefore, for the majority of search missions this small increase in performance is not worth the large computational cost. In addition, ranging the communication studies across a range of parameters with no time decay showed communication mattered most at low area ratios and turning ratios. Specifically, when we set the area ratio to 0.1, the turning ratio to 0.025, and the endurance ratio to 0.075, performance improved by approximately 23% from the no communication case to the communication case with a communication ratio of 2. This suggests that communication is most important when vehicles have limited vision of a search space. In addition, good agility is necessary to quickly make adjustments to a planned path. Communication had less of an impact for higher area ratios and higher turning ratios. In addition, communication did not matter for very low endurance ratios because vehicles could not travel far enough to encounter another vehicle.

## 6.2 Future Work

We only analyzed the effect of four of the non-dimensional parameters. Future work would analyze further with the other non-dimensional numbers and see how performance changes with the variation of these parameters. This would be difficult because we already varied parameters across three dimensions and it would be challenging to show the results from a higher-dimension parameter variation study.

Little work has been performed regarding high density search swarms. In the future, optimization methods should be explored with swarms consisting of hundreds to thousands of vehicles. This presents many challenges involving the large number of trajectory optimizations. Efficient methods would need to be devised to coordinate this many vehicles while trying to minimize the large computational cost. We used this optimization algorithm with 400 vehicles for 16 time steps. With 28 CPUs the wall time was several hours. Therefore we feel this algorithm is suitable for high density swarms. However, further research is necessary because this simulation was a demonstration of scaling the number vehicles. In actuality, a real case study using this scaling would have a larger domain area. We did not continue to run simulations with this large number of vehicles.

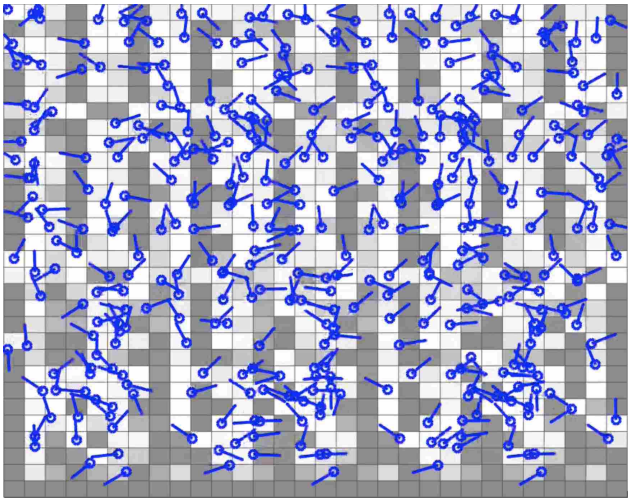


Figure 6.1: Image displaying 400 vehicles searching a small area.

In addition, search missions can benefit from heterogeneous swarms. Adding diverse vehicles to a swarm can increase search performance by capitalizing on the different vehicle strengths. Diverse fixed wing vehicles such as vehicles with different velocities, turning radii, or sensing radii can be added to see how performance changes. Furthermore, the swarm could benefit from the use of quadrotor UAVs which have different properties and characteristics from fixed wing UAVs. This would thereby increase overall swarm capabilities. However, adding diverse vehicles would require the use of mission allocation methods to assign missions based on a variety of factors.

In addition, trajectory optimization algorithms should be tested with real-time flight trials. This would require uploading the code to an embedded processor to be flown on fixed wing UAVs. While hundreds of vehicles may not be practical, several vehicles can be used to test this algorithm. However, a collision avoidance constraint should be applied to prevent the loss of expensive hardware.

## REFERENCES


[1] Sujit, P., and Ghose, D., 2004. "Search using multiple uavs with flight time constraints." *IEEE Transactions on Aerospace and Electronic Systems,* **40**(2), April. 2

[2] Blasi, L., Barbato, S., and Mattei, M., 2013. "A particle swarm approach for flight path optimiation in a constrained environment." *Aerospace Science and Technology,* **26**(1), April-May, pp. 128–137. 2

[3] Nigma, N., Bieniawski, S., Kroo, I., and Vian, J., 2012. "Control of multiple uavs for persistent surveillance: Algorithm and flight test results." *IEEE Transactions on Control Systems Technology,* **20**(5), September. 2

[4] Howard, A., Parker, L. E., and Sukhatme, G. S., 2006. "Experiments with a large heterogeneous mobile robot team: Exploration, mapping, deployment, and detection." *The International Journal of Robotics,* **25**(5-6). 3, 4

[5] Gade, S., and Joshi, A., 2013. "Heterogeneous uav swarm system for target search in adversarial environment." *Control Communication and Computing*. 3, 4

[6] Jaimes, A., Kota, S., and Gomez, J., 2008. "An approach to surveillance an area using swarm of fixed wing and quad-rotor unmanned aerial vehicles uav(s)." *System of Systems Engineering,* **IEEE**. 3, 4

[7] Zhang, B., Liu, W., Mao, Z., Liu, J., and Shen, L., 2014. "Cooperative and geometric learning algorithm (cgla) for path planning of uavs with limited information." *Automatica,* **50**(3), March, pp. 809–820. 3

[8] Deng, Q., Yu, J., and Wang, N., 2013. "Cooperative task assignment of multiple heterogeneous unmanned aerial vehicles using a modified genetic algorithm with multi-type genes." *Chinese Journal of Aeronautics,* **26**(5), October, pp. 1238–1250. 3, 4

[9] Shen, C., Shi, Y., and Buckham, B., 2017. "Integrated path planning and tracking control of an auv: A unified receding horizon optimization approach." *IEEE/ASME Transactions on Mechatronics,* **22**(3), September, pp. 1163–1173. 3, 13

[10] Schouwenaars, T., Feron, E., and How, J. "Safe receding horizon path planning for autonomous vehicles.". 3, 13

[11] Richards, A., Schouwenaars, T., How, J. P., and Feron, E., 2012. "Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming." *Journal of Guidance, Control, and Dynamics,* **25**(4), July-August. 3, 13

[12] Keviczky, T., Borrelli, F., Fregene, K., Godbole, D., and Balas, G. J., 2007. "Decentralized receding horizon control and coordination of autonomous vehicle formations." *IEEE Transactions on Control Systems Technology,* **16**(1), January, pp. 19–33. 3

[13] Bellingham, J., Richards, A., and How, J., 2002. "Receding horizon control of aautonomous aerial vehicles." *Proceedings of the 2002 American Control Conference*. 3, 13

[14] Huang, Y., Xu, J., Qiu, L., and Zhang, R., 2018. "Cognitive uav communication via joint trajectory and power control.". 3

[15] Alshbatat, A. I., and Dong, L., 2010. "Performance analysis of mobile ad hoc unmanned aerial vehicle communication networks with directional antennas." *International Journal of Aerospace Engineering*, p. 14. 3

[16] Waharte, S., and Trigoni, N., 2010. "Supporting search and rescue operations with uavs." In *Emerging Security Technologies*. 3

[17] Krieger, M. J., and Billeter, J.-B., 2000. "The call of duty: Self-organised task allocation in a population of up to twelve mobile robots." *Robotics and Autonomous Systems,* **30**(1-2), January, pp. 65–84. 3

[18] Arkin, R. C., Balch, T., and Nitz, E., 1993. "Communication of behavioral state in multi-agent retrieval tasks." *IEEE ICRA,* **3**, pp. 588–594. 4

[19] Trianni, V., and Dorigo, M., 2006. "Self-organisation and communication in groups of simulated and physical robots." *Biological Cybernetics,* **95**(3). 4

# APPENDIX A. CODE

## A.1 Grid Initialization

```
function initializegrid(nVehicles, NorthBoundary, EastBoundary,gridspacing)

  # initialize starting locations of the vehicles
  rowVehicle = sqrt(nVehicles)
  vehicleSpacing = EastBoundary/(rowVehicle + 1)
  vehiclePosition = SharedArray{Float64}(nVehicles,3)
  count = 1

  for i = 1:rowVehicle
      for j = 1:rowVehicle

          vehiclePosition[count,:] = [rand(1)*2*pi i*vehicleSpacing j*
              vehicleSpacing]
          count = count + 1
      end
  end

  # initialize grid
  count = 1
  grid_point = Array{Float64}(undef,900,2)
  for x = gridspacing/2:gridspacing:EastBoundary
      for y = gridspacing/2:gridspacing:NorthBoundary
          grid_point[count,:] = [y x]
            count = count + 1
        end
  end

```

```
    # initialize grid values
    nPsuedoTargets = count-1
    grid_value = SharedArray{Float64}(nPsuedoTargets,nVehicles)
    grid_value .= 0
    return vehiclePosition, grid_point, grid_value
end # initializegrid
```

Listing A.1: Initialization of grid and vehicle locations

## A.2 Coordinated Turn Model

```

  function getlocation(dy,y,p,t)
    roll = p[1]
    g = p[2]
    V = p[3]
    dy[1] = g/V*tan(roll) #yaw
    dy[2] = V*sin(y[1]) #east position
    dy[3] = V*cos(y[1]) #north position
end
```

Listing A.2: Equations of motions used in conjunction with ODE solver

## A.3 Reward Function

```
function calcreward(optimizevalues,paramsint,paramsfloat,endlocation,
    gridvalues,gridcenters)
    # unpack vectors and initialize

    R = eltype(optimizevalues)
    nRoll = paramsint[1]
     nVehicles = paramsint[2]
    nCells = paramsint[3]
    t = paramsfloat[3]
    radius = paramsfloat[1]
    g = paramsfloat[2]
    vehicleEast = Array{R}(undef,1,nRoll)
    vehicleNorth = Array{R}(undef,1,nRoll)
```

```
    y0 = Array{R}(undef,3)
    y0 .= endlocation


# get location of vehicles after each roll angle
 for i = 1:nRoll

        params = [optimizevalues[i],g,optimizevalues[end]]
        tspan = (0.0,t)
        prob = DifferentialEquations.ODEProblem(getlocation,y0,tspan,params)
        sol = DifferentialEquations.solve(prob,save_everystep=false)
        y0=sol(t)
        vehicleEast[i]= y0[2]
        vehicleNorth[i] = y0[3]

 end

endlocation = [y0[1] y0[2] y0[3]];
# create vector of vehicle locations to determine distances to all grid
# centers using matrix math
EastVehicle = repeat(vehicleEast,nCells,1)
NorthVehicle = repeat(vehicleNorth,nCells,1)
Northcenters = repeat(gridcenters[:,1],1,nRoll)
Eastcenters = repeat(gridcenters[:,2],1,nRoll)

# calculate distance to each grid center
centerdistance = sqrt.((NorthVehicle-Northcenters).^2 + (EastVehicle-
    Eastcenters).^2)
 # Calculate smallest distance from path to each grid center
distance,_= findmin(centerdistance,dims=2)
# Use gaussian distribution to calculate new grid value
gaussian = exp.(-(distance).^2/(2*(radius/3)^2))

# update grid values
updatelocation = gaussian.>gridvalues
grid_values = gridvalues-updatelocation.*gridvalues + updatelocation.*gaussian
```

```
48 cost = sum(grid_values)/900
49 return cost, grid_values, endlocation
50 end #calcreward
```

Listing A.3: Julia reward function

## A.4 Data Sharing

```
1     for k = 1:nVehicles
2         for j = 1:nVehicles
3
4             # Calculate how close the two vehicles are
5             raddistance = sqrt((vehiclePosition[k,2]-vehiclePosition[j,2])^2 +
                  (vehiclePosition[k,3]-vehiclePosition[j,3])^2)
6
7             # The jth vehicle shares its grid values if within communication
                  radius
8             if raddistance<commrad && k!= j
9                 O[:,j,k] = lookahead[:,j]
10            # If not within communication radius, update that column to be kth
                  vehicle's grid values
11            else
12                O[:,j,k] = gridvalues[:,k]
13            end
14        end
15
16        # Use highest perceived grid values from shared data and vehicle's own
              prediction
17        optimizevals[:,k] = max(O[:,:,k], dims = 2)
18    end
```

Listing A.4: If vehicles entered the communication radius, they would share their grid values. The highest value for each grid cell shared with the vehicle would be the grid value the vehicle received for the next optimization.

## A.5 Initialization of Targets

```
function initializetarg(ntargs,northbound,eastbound,nvehicles)

    # Set starting location and orientation of targets
    targPosition = [rand(1)*2*pi eastbound/3 northbound/3;
    rand(1)*2*pi eastbound/3 2*northbound/3;
    rand(1)*2*pi eastbound/2 northbound/2;
    rand(1)*2*pi 2*eastbound/3 northbound/3;
    rand(1)*2*pi 2*eastbound/3 2*northbound/3
    ]

    # Set starting velocities of targets
    velocity = Array{Float64}(undef,1,ntargs)
    velocity[:] .= 10.0
    targvalues = SharedArray{Float64}(ntargs,nvehicles)
    return targPosition,velocity,targvalues

end
```

Listing A.5: Initialization of initial target locations, orientations, and velocities

## A.6 Propagation of Targets

```
function propogatetargs(ntargets, targPosition,velocity,timestep)

    # Propogate target positiion based on velocity and heading
    targPosition[:,1] .= targPosition[:,1].+rand(ntargets)*pi/6 .- pi/12
    targPosition[:,3] = targPosition[:,3].+(timestep*velocity.*(sin.(
        targPosition[:,1]))')'
    targPosition[:,2] = targPosition[:,2].+(timestep*velocity.*(cos.(
        targPosition[:,1]))')'
    return targPosition
end
```

Listing A.6: Propagate targets forward in time based on velocities and heading angles

## A.7 Combined target tracking and grid search function

```
function calcreward(optimizevalues,paramsint,paramsfloat,endlocation,
    gridvalues,gridcenters,targposition,targvalues,ntargs,weight,gridweight)
    # unpack vectors and initialize

    R = eltype(optimizevalues)
    nRoll = paramsint[1]
    nVehicles = paramsint[2]
    nCells = paramsint[3]
    t = paramsfloat[3]
    radius = paramsfloat[1]
    g = paramsfloat[2]
    vehicleEast = Array{R}(undef,1,nRoll)
    vehicleNorth = Array{R}(undef,1,nRoll)
    y0 = Array{R}(undef,3)
    y0 .= endlocation

# get location of vehicles after each roll angle
 for i = 1:nRoll

        params = [optimizevalues[i],g,optimizevalues[end]]
        tspan = (0.0,t)
        prob = DifferentialEquations.ODEProblem(getlocation,y0,tspan,params)
        sol = DifferentialEquations.solve(prob,save_everystep=false)
        y0=sol(t)
        vehicleEast[i]= y0[2]
        vehicleNorth[i] = y0[3]

 end
endlocation = [y0[1] y0[2] y0[3]];

# create vector of vehicle locations to determine distances to all grid
# centers using matrix math
EastVehicle = repeat(vehicleEast,nCells,1)
NorthVehicle = repeat(vehicleNorth,nCells,1)
Northcenters = repeat(gridcenters[:,1],1,nRoll)
Eastcenters = repeat(gridcenters[:,2],1,nRoll)
```

```
36 EastVehicletarg = repeat(vehicleEast,ntargs,1)
37 NorthVehicletarg = repeat(vehicleNorth,ntargs,1)
38 vehicletargnorth = repeat(targposition[:,3],1,nRoll)
39 vehicletargeast = repeat(targposition[:,2],1,nRoll)
40
41 # calculate distance to each target
42 targdistance = sqrt.((vehicletargnorth-NorthVehicletarg).^2 + (vehicletargeast
       -EastVehicletarg).^2)
43
44 # calculate smallest distance froom path to each target
45 mintargdis,_= findmin(targdistance,dims=2)
46
47 # use gaussian to calculate target values
48 targgaussian = exp.(-(mintargdis).^2/(2*(radius/3)^2))
49
50 # calculate distance to each grid center
51 centerdistance = sqrt.((NorthVehicle-Northcenters).^2 + (EastVehicle-
       Eastcenters).^2)
52
53  # calculate smallest distance from path to each grid center
54 distance,_= findmin(centerdistance,dims=2)
55
56 # use gaussian distribution to calculate new grid value
57 gaussian = exp.(-(distance).^2/(2*(radius/3)^2))
58
59 # update grid and target values
60 updatelocation = gaussian.>gridvalues
61 updatetarg = targgaussian.>targvalues
62 targreward = targvalues - targvalues.*updatetarg + updatetarg.*targgaussian
63 grid_values = gridvalues-updatelocation.*gridvalues + updatelocation.*gaussian
64 gridcost = sum(grid_values)
65 targcost = sum(targreward)
66
67 # calculate the final cost based on the weightings
68 finalcost = (gridweight*gridcost+weight*targcost)/(gridweight*900+weight*5)
69 return finalcost, grid_values, endlocation, targreward, gridcost
```

```
end #calcreward
```

Listing A.7: Combined reward function with target tracking and grid search

## A.8 Optimization Function

```
function optimizeswarm(x0,lb,ub,paramsint,paramsfloat,Endlocation,gridvalues,
    gridcenters)

    # Function to be optimized with constraints
    function objcon(x)

        # Call calcreward as the objective
        function reward(x)
            obj,_,_ = calcreward(x,paramsint,paramsfloat,Endlocation,
                gridvalues,gridcenters)
            # We want to maximize the reward so negate objective for the
                minimizer snopt
            return -obj
        end
        f = reward(x)
        # call forward diff to calculate gradients
        dfdx = ForwardDiff.gradient(reward,x)
        # return empty arrays as constraints
        c = Array{Float64, 1}(undef, 0)
        dcdx = Array{Float64, 1}(undef, 0)
        fail = false
        return f, c, dfdx, dcdx, fail
    end
    options = Dict{String, Any}()
    options["Major optimality tolerance"] = 1e-6
    options["Derivative option"] = 1
    options["Verify level"] = 1
    # Call snopt
    xopt, fopt = snopt(objcon, x0, lb, ub, options)
     return xopt,fopt
```

```
28      end
```

Listing A.8: If vehicles entered the communication radius, they would share their grid values. The highest value for each grid cell shared with the vehicle would be the grid value the vehicle received for the next optimization.