

Air Force Institute of Technology

AFIT Scholar

Theses and Dissertations

Student Graduate Works

3-2003

Distributed Control of a Swarm of Autonomous Unmanned Aerial Vehicles

James T. Lotspeich

Follow this and additional works at: <https://scholar.afit.edu/etd>



Part of the [Computer Sciences Commons](#)

Recommended Citation

Lotspeich, James T., "Distributed Control of a Swarm of Autonomous Unmanned Aerial Vehicles" (2003).
Theses and Dissertations. 4205.
<https://scholar.afit.edu/etd/4205>

This Thesis is brought to you for free and open access by the Student Graduate Works at AFIT Scholar. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of AFIT Scholar. For more information, please contact richard.mansfield@afit.edu.



DISTRIBUTED CONTROL
OF A SWARM OF AUTONOMOUS
UNMANNED AERIAL VEHICLES

THESIS

James T. Lotspeich, First Lieutenant, USAF

AFIT/GCS/ENG/03-10

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright Patterson Air Force Base, Ohio

Approved for public release; distribution unlimited

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

AFIT/GCS/ENG/03-10

DISTRIBUTED CONTROL OF A SWARM OF AUTONOMOUS UNMANNED
AERIAL VEHICLES

THESIS

Presented to the Faculty of the
Department of Electrical and Computer Engineering
Graduate School of Engineering and Management
Air Force Institute of Technology
Air University
Air Education and Training Command
in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Computer Engineering

James T. Lotspeich, B.S.
First Lieutenant, USAF

March, 2003

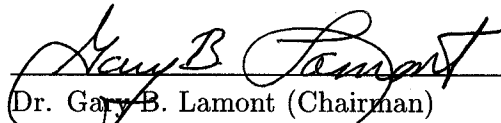
Approved for public release; distribution unlimited

DISTRIBUTED CONTROL OF A SWARM OF AUTONOMOUS UNMANNED
AERIAL VEHICLES


James T. Lotspeich, B.S.

First Lieutenant, USAF

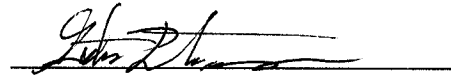
Approved:


Dr. Gary B. Lamont (Chairman)

7 Mar 03
Date


Dr. Meir Pachter (Member)

13 March 03
Date


Dr. Gilbert L. Peterson (Member)

17 MAR 03
Date

Acknowledgements

I thank the Lord Jesus Christ, son of God, who was crucified for our sins, buried, and three days later, raised from the dead. His immeasurable grace, wisdom, and love have provided a firm anchor in rough times, and a light in the darkness. “Come to me, all you who are weary and burdened, and I will give you rest. Take my yoke upon you and learn from me, for I am gentle and humble in heart, and you will find rest for your souls. For my yoke is easy and my burden is light.” (Matt 11:28-30 NIV). I also thank my beautiful fiancée, whose willingness to listen to my indecipherable rants have given me the chance to form my thoughts into coherent sentences. Her loving support has never failed me.

James T. Lotspeich

Table of Contents

	Page
Acknowledgements	iii
List of Figures	viii
List of Tables	xii
List of Abbreviations	xiv
Abstract	xvi
 1. Introduction	 1
1.1 Motivation	2
1.2 Problem Description	3
1.3 Assumptions	4
1.4 Research Goals	5
1.5 Research Sponsors	5
1.6 Organization	6
 2. Current Research in Distributed Control	 7
2.1 Terms and Definitions	7
2.2 Problem Domain Definition	10
2.3 Contemporary Research	11
2.3.1 Contemporary Swarm Architectures	14
2.4 Cooperative Munitions	21
2.5 Swarm Convergence	24
2.6 Path Planning	26
2.6.1 Path Planning Using Roadmaps	27
2.6.2 Artificial Potential Fields	28

	Page
2.7 Types of Behavior	32
2.8 Learning	35
2.9 Summary	36
3. Swarm Algorithm Design	37
3.1 Introduction	37
3.2 High Level Design	37
3.3 Swarm Model	38
3.3.1 Potential Fields	40
3.3.2 Neighborhood Determination	42
3.3.3 Reynolds' Behaviors	43
3.3.4 Peripheral Vision Weighting	46
3.3.5 Goal Seek/Threat Avoidance	47
3.3.6 Waypoints	52
3.3.7 Path Planning	56
3.3.8 Behavior Modes	56
3.3.9 Mode Transition	62
3.3.10 Communication	63
3.4 Low Level Design	64
3.4.1 Scaling and Coordinate Systems	64
3.4.2 Flight Model	65
3.4.3 Fitness Objectives	69
3.4.4 Behavior Adaptation	71
3.5 Metrics	72
3.6 Evolution Strategy	75
3.6.1 Selection Operators	76
3.6.2 Mutation Operators	76
3.6.3 Recombination	77

	Page
3.7 Low Level Design Implementation	78
3.8 Summary	78
4. Design and Analysis of Experiments	80
4.1 Overview of Experiments	80
4.2 Experimental Methodology	80
4.2.1 Evolutionary Strategy Optimization	81
4.3 Baseline	84
4.4 Validation of Coefficient Matrices	85
4.4.1 Test Landscape	86
4.5 Generality	89
4.6 Scalability	90
4.6.1 Statistical Methods	91
4.7 Summary	92
5. Results and Analysis	93
5.1 Introduction	93
5.2 Fitness Evaluation	93
5.2.1 Reconnaissance	93
5.2.2 Scan	95
5.2.3 En-route	96
5.3 Baseline	96
5.4 Reconnaissance	99
5.4.1 Saddle Map	99
5.4.2 Obstacle Map	102
5.4.3 Overlap Map	104
5.5 Scan	105
5.6 En-route	109

	Page
5.7 Matrix Values	113
5.8 Scalability	117
5.9 Summary	119
6. Conclusions	123
6.1 Introduction	123
6.2 Swarm Model	123
6.3 Fitness Evaluation	124
6.4 Noisy Swarm Model	125
6.5 Summary	126
Appendix A. Coefficient Matrix Values by Test Case	127
A.1 Baseline Matrix	127
A.2 Reconnaissance Matrices	127
A.2.1 Saddle map	127
A.2.2 Obstacle map	130
A.2.3 Overlap map	134
A.3 Scan Matrices	137
A.3.1 Saddle map	137
A.3.2 Obstacle map	141
A.3.3 Overlap map	144
A.4 En-route Matrices	147
A.4.1 Saddle map	147
A.4.2 Obstacle map	151
A.4.3 Overlap map	154
Appendix B. Behavior Matrix Comparison	158
Bibliography	161

List of Figures

Figure		Page
1.	The Israeli Scout UAV used during the Bekaa Valley Offensive [43]	2
2.	Pseudo code for agent control loop	40
3.	Shallow local minimum in a potential field	41
4.	Deep local minimum in a potential field	42
5.	Depiction of sensor shadowing in neighborhood calculation [38] . . .	44
6.	Cohesion and Separation values for distances from 0 to 5km	45
7.	Values of ω_{periph} using $\beta = 0.1, 1.0, 2.0, 4.0$	47
8.	Potential field surrounding a threat located at (350,250) with $\text{rad} = 100$	48
9.	The power received versus distance of target for a monostatic radar. The distance values are not reflective of any existing radar due to the fact that the constant coefficients were removed from the equation in order to view the affect of the dependant variable on power received	49
10.	The magnitude of repulsion for a threat versus distance from the threat. The radius is set to $\text{rad} = 50$ for this example	50
11.	The potential field around a goal located at (350,250) with $\varrho = 0.75$	51
12.	The potential field resulting from one threat with $\sigma = 100$ and one goal with $\varrho = 1.0, 0.75, 0.5, 0.25$. The location of the center of the threat is depicted by a diamond and the location of the center of the goal is indicated by a square.	51
13.	The potential field created by ten threats with radius of $\sigma = 100$ and located at (159,168), (638,439), (161,284), (539,372), (377,490), (110,280), (9,435), (161,445), (490,437), (245,196) and 1 goal with $\text{rad} = 100$, $\varrho = 0.5$ and located at (650,450). The goal is depicted by a square in the upper right corner, while the threats are depicted by diamonds	52
14.	Graphical depiction of the calculation of the waypoint intersection point	53

Figure		Page
15.	Graphical depiction of the corrected waypoint intersection point . .	53
16.	Notional example illustrating divergence behavior due to an infinitely long wavefront.	54
17.	The potential field created by a waypoint with location (350,250), direction [1,1], $\varrho = 0.75$, and width = 100	55
18.	Sensor coverage of a swarm of agents at specific moment in time. The light-gray trails represent the area that has been covered in the past 50 timesteps. The dark-gray areas are overlapping areas calculated during the current timestep. The dark line encloses the total contiguous sensor area for this example. Note that while agent 1's footprint does not overlap with agent 2's current footprint, it does overlap with agent 2's historical coverage. In this situation, the area enclosed in a dark line around agent 1 is counted as part of the contiguous area.	58
19.	A snapshot in time of agents scanning an area. Two contiguous areas are depicted with a gap between the two. Each contiguous area is emphasized by a dark outline. Note that while agent 1's footprint does not overlap with agent 2's current footprint, it does overlap with agent 2's historical coverage. In this situation, the area enclosed in a dark line around agent 1 is counted as part of the contiguous area.	59
20.	A notional example depicting a path and four agents as well as their look angles.	61
21.	Reynold's steering force envelope as restricted by thrust, braking, and steering [74]	66
22.	The amount of turning force required is calculated by representing the current direction, desired direction, and turning force as an isosceles triangle. The length of the two equal sides of the triangle is set to $ \vec{d} $. In this manner, the steering force results in a turn with no forward or reverse acceleration.	68
23.	Method for determining distance for velocity matching fitness function	70
24.	The structure of a behavior adaptation matrix	72
25.	Affect of rotation angles on the shape of the probability density for $n_\sigma = 1$ on the left, $n_\sigma = 2$ in the middle, and correlated mutations $n_\sigma = 2, n_\alpha = 1$ on the right [4]	77

Figure		Page
26.	Comparison of convergence for different values of $\sigma_i(0)$	82
27.	Convergence rate for $\sigma_i(0) = 1.0$	84
28.	Saddle landscape with broad saddle feature	87
29.	Overlap landscape with “raised” saddle feature	88
30.	Obstacle landscape with obstacle feature	88
31.	Landscape used to test the generality and scalability of the evolved coefficient matrix. The starting point is in the upper left corner of the landscape and the goal is in the lower right corner. This landscape is 2000×2000 pixels wide, which simulates a 200km wide area.	90
32.	Figure showing snapshot of baseline swarm behavior on Saddle map	98
33.	Visualization of swarm reconnaissance behavior on Saddle map . . .	99
34.	Visualization of initial response in reconnaissance behavior on Saddle map	100
35.	Visualization of swarm reconnaissance behavior on the Obstacle map	102
36.	Visualization of initial response in reconnaissance behavior on Obstacle map	103
37.	Visualization of swarm reconnaissance behavior on the Overlap map	105
38.	Visualization of swarm exhibiting scan behavior on the Saddle map	106
39.	Visualization of swarm exhibiting scan behavior on the Obstacle map	107
40.	Visualization of swarm exhibiting scan behavior on the Overlap map	109
41.	Visualization of swarm exhibiting enroute behavior on the Saddle map	110
42.	Visualization of swarm exhibiting en-route behavior on the Obstacle map	111
43.	Visualization of swarm exhibiting en-route behavior on the Overlap map	113
44.	Visualization of swarm exhibiting en-route behavior on the Overlap map	114
45.	Comparison of matrices evolved for the Reconnaissance Mode on the the Saddle map	115

Figure		Page
46.	Comparison of matrices evolved for the Reconnaissance Mode on the the Obstacle map	116
47.	Comparison of matrices evolved for the Reconnaissance Mode on the the Overlap map	116
48.	Emergent behavior of swarm with $ S = 1024$ after passing first saddle obstacle using Reconnaissance mode	118
49.	Emergent behavior of swarm with $ S = 1024$ as it encounters a set of overlapping threats while in Reconnaissance mode	119
50.	Emergent behavior of swarm with $ S = 1024$ for the Scan mode . .	120
51.	Emergent behavior of swarm with $ S = 1024$ as the swarm transitions from Scan to En-route mode	121
52.	Emergent behavior of swarm with $ S = 1024$ as it encounters a turn in the waypoint path	121
53.	Emergent behavior of swarm with $ S = 1024$ as it encounters an obstacle threat in En-route mode	122
54.	Comparison of matrices evolved for the Scan Mode on the the Saddle map	158
55.	Comparison of matrices evolved for the Scan Mode on the the Obstacle map	158
56.	Comparison of matrices evolved for the Scan Mode on the the Overlap map	159
57.	Comparison of matrices evolved for the En-route Mode on the the Saddle map	159
58.	Comparison of matrices evolved for the En-route Mode on the the Obstacle map	160
59.	Comparison of matrices evolved for the En-route Mode on the the Overlap map	160

List of Tables

Table		Page
1.	Values used for the physics model of the airframe	68
2.	A comparison of the flight performance of the simulation model and the flight performance of the RQ-1 Predator UAV [24]	69
3.	Sample Minimum, Mean, Max, and Standard Deviation for results obtained using values of $\sigma_i(0) = \{0.25, 0.5, 0.75, 1.0\}$	83
4.	Summary of experiments performed for Evolution Strategy algorithm parameter tuning. Average run time using 25 procesors is ~ 1200 minutes.	84
5.	Baseline Coefficient Matrix	85
6.	Comparison of features contained in each of the test cases used . .	87
7.	Metrics obtained using equation 48 on all three test maps	94
8.	Metrics obtained using equation 49 on all three test maps	95
9.	Metrics obtained for the baseline behavior on the Saddle map . . .	96
10.	Metrics obtained for the baseline behavior on the Obstacle map . .	97
11.	Metrics obtained for the baseline behavior on the Overlap map . . .	97
12.	Minimum, average, maximum, and standard deviation for metrics of the reconnaissance behavior on Saddle map	100
13.	Minimum, average, maximum, and standard deviation for metrics of the reconnaissance behavior on the Obstacle map	104
14.	Minimum, average, maximum, and standard deviation for metrics of the reconnaissance behavior on the Overlap map	104
15.	Minimum, average, maximum, and standard deviation for metrics of the scan behavior on the Saddle map	107
16.	Minimum, average, maximum, and standard deviation for metrics of the scan behavior on the Obstacle map	108
17.	Minimum, average, maximum, and standard deviation for metrics of the scan behavior on the Overlap map	109

Table		Page
18.	Minimum, average, maximum, and standard deviation for metrics of the en-route behavior on the Saddle map	111
19.	Minimum, average, maximum, and standard deviation for metrics of the en-route behavior on the Obstacle map	112
20.	Minimum, average, maximum, and standard deviation for metrics of the en-route behavior on the Overlap map	113

List of Abbreviations

Abbreviation		Page
RPVs	Remotely Piloted Vehicle	1
UAVs	Unmanned Aerial Vehicle	1
GMTI	Ground Moving Target Indicator	1
MAV	Micro Air Vehicle	1
APFs	Artificial Potential Fields	3
NP	Non-deterministic Polynomial	3
AFRL/IF	Air Force Research Laboratory, Information Directorate	5
VLSR	Very Large Scale Robotics System	8
SIMD	Single Instruction Multiple Data	13
MILP	Mixed Integer Linear Programming	14
SEAD	Suppression of Enemy Air Defenses	14
NP	Non-Deterministic Polynomial Time	16
FEBA	Forward Edge of the Battle Area	16
ETA	Estimated Time of Arrival	16
LAN	Local Area Network	19
RSM	Response Surface Methodology	22
PRM	Probabilistic Roadmap	27
C-free	Configuration-Free	27
RRT	Rapidly-expanding Random Search Trees	28
APFs	Aritificial Potential Fields	28
MOEA	Multi-Objective Evolutionary Algorithm	30
RCS	Radar Cross Section	49
FIFO	First-In-First-Out	55
SAR	Search And Rescue	57
TAT	Total Accumulated Threat	73

Abbreviation		Page
ES	Evolution Strategy	75
EA	Evolutionary Algorithms	75
GUI	Graphical User Interface	78
RMI	Remote Method Invocation	78

Abstract

With the increasing use of Unmanned Aerial Vehicles (UAV)s in military operations, there is a growing need to develop new methods of control and navigation for these vehicles. This investigation proposes the use of an adaptive swarming algorithm that utilizes local state information to influence the overall behavior of each individual agent in the swarm based upon the agent's current position in the battlespace. In order to investigate the ability of this algorithm to control UAVs in a cooperative manner, a swarm architecture is developed that allows for on-line modification of basic rules. Adaptation is achieved by using a set of behavior coefficients that define the weight at which each of four basic rules is asserted in an individual based upon local state information. An Evolutionary Strategy (ES) is employed to create initial matrices of behavior coefficients. Using this technique, three distinct emergent swarm behaviors are evolved, and each behavior is investigated in terms of the ability of the adaptive swarming algorithm to achieve the desired emergent behavior by modifying the simple rules of each agent. Finally, each of the three behaviors is analyzed visually using a graphical representation of the simulation, and numerically, using a set of metrics developed for this investigation.

DISTRIBUTED CONTROL OF A SWARM OF AUTONOMOUS UNMANNED AERIAL VEHICLES

1. Introduction

The introduction of unmanned and remotely piloted vehicles into the battlefield has been a goal of military organizations since before the dawn of aviation. As early as 1863, Charles Perley designed and patented an “unmanned aerial bomber” [43]. This device consisted of a hot-air balloon, an explosives device and a timing mechanism. The timer would be set based on the prevailing winds, the balloon would be launched, and when the timer went off, an explosive device would be dropped into the midst of the enemy troops [43].

Unmanned technology has progressed a great deal since the Civil War era. In the 1980’s the Israeli Air Force used remotely piloted Scout vehicles (see Figure 1) to fool Syrian radar sites into activating their radars. The Israeli bombers used this technique to locate and destroy 19 missile sites, thus achieving air superiority over Syria [33]. This use of Remotely Piloted Vehicles (RPVs) highlighted the tactical advantages obtained from the use of unmanned vehicles.

As technology progresses, Unmanned Aerial Vehicles (UAVs) are used for more applications in the battlefield environment. In support of Operation Joint Endeavor, for example, the Predator has flown “more than 350 sorties and 2,800-plus hours” [56]. More recently, the United States Air Force has pursued the development of the RQ-4 Global Hawk UAV which is capable of spot radar surveillance, wide-area search and rescue, and Ground Moving Target Indicator (GMTI) sensor modes [58].

While the 25,600 pound Global Hawk is the heaviest of the UAVs, other UAVs exist that are small enough to be carried by a soldier into the battlefield. One example of this smaller UAV is the WASP [23]. This aircraft is an example of a Micro Air Vehicle (MAV) and is capable of remaining aloft for 1 hour 47 minutes. The WASP is equipped with an

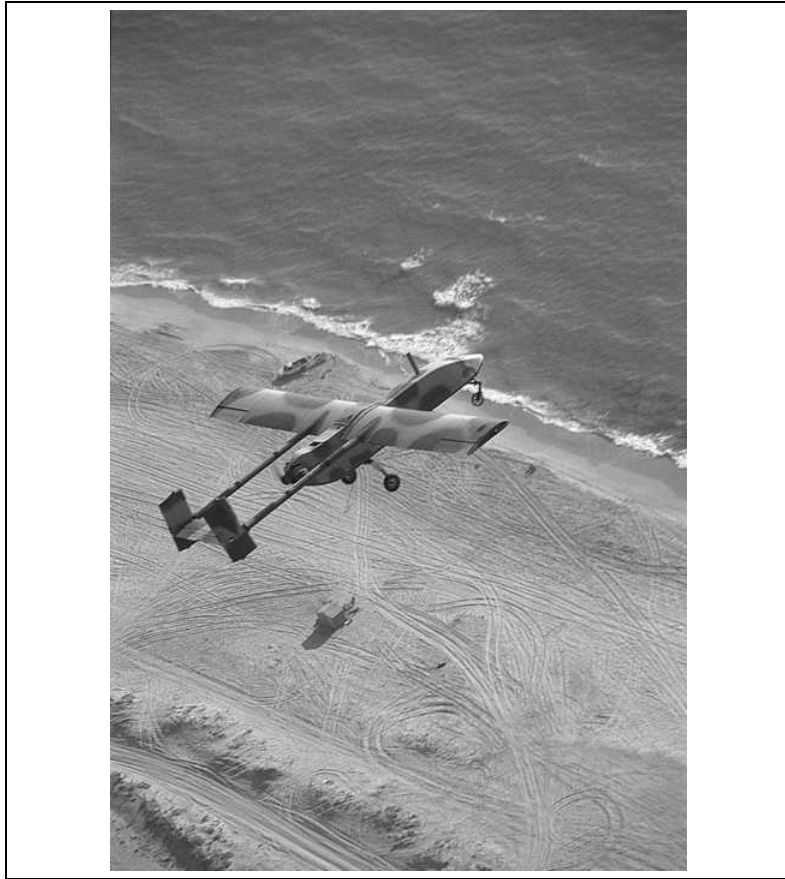


Figure 1 The Israeli Scout UAV used during the Bekaa Valley Offensive [43]

on-board color video camera, and is capable of transmitting images to the user in real-time [23].

1.1 Motivation

Missions deemed appropriate for UAVs have changed with advances in UAV technology. Attack missions that would normally be assigned to a piloted aircraft can now be performed by Predator UAVs armed with wing-mounted Hellfire missiles [87]. The United States Senate Committee on Armed Services has committed to further expand the use of this technology in the near future:

“It shall be a goal of the Armed Forces to achieve the fielding of unmanned, remotely controlled technology such that

- by 2010, one third of the operational deep strike aircraft of the Armed Forces are unmanned; and
- by 2015, one third of the operational ground combat vehicles of the Armed Forces are unmanned.” [62]

In response to this Congressional bill, the Air Force Research Lab set an objective to “address fundamental issues in distributed decision, guidance, and control for multiple UAVs” [9]. This objective is further decomposed into solving problems related to the forming of teams of UAVs, assigning each team to a task, and performing the assigned tasks in a coordinated manner [9].

While the use of UAVs in solo flight continues to be the primary application of UAV technology, it is possible to achieve gains through the use of multiple UAV platforms moving in a coordinated manner. The concept of using cooperative munitions to search for and destroy a target has become an important area of research [31] [19] [20] [35] [32]. Another area of research is the cooperative search ability of a swarm of UAVs [69]. Researchers in these fields have used path planning techniques, integer programming techniques, set covering algorithms, and Artificial Potential Fields (APFs) in an attempt to create a cooperative framework for multiple UAVs to achieve a common goal [64].

1.2 Problem Description

The problem addressed by this research is the development of algorithms to define behaviors for a team of autonomous agents. The cooperative model used for this research is analogous to a swarm of bees or a flock of birds. Rather than planning a path from start to finish for each vehicle, each adjusts its speed, heading, and orientation based upon local interactions with other members of the team. This method allows the coordination problem to be separated from the path planning and movement problem. Using this model, specific behaviors of the overall team, hereafter referred to as a “swarm”, emerge based upon the complex interactions of some basic rules [15].

The task of decomposing a complex swarm behavior into an irreducible set of rules is a non-deterministic polynomial (NP)-hard problem [71]. Due to this complexity, rather than attempting to determine the basic rules that lead to a specific, observed behavior,

this research attempts to define a behavior by providing a means of modification to a set of basic rules. This difference in approach provides a more flexible framework from which many behaviors may potentially be created.

1.3 Assumptions

In order to create a simulation of a real-world problem, it is necessary to make some assumptions to reduce the problem to a more manageable size. While these are not the only assumptions made in support of this research, they encompass assumptions that are necessary for further definition of the problem domain.

The first assumption is that only aerial vehicles are treated in this research. Since autonomous agents can consist of air or ground vehicles (and may be holonomic or non-holonomic in the case of ground vehicles [41]), it is necessary to reduce the problem domain to consist of aerial vehicles only. The behaviors developed in support of this effort are based upon a sensor-only craft as opposed to an attack aircraft. Sensors are considered to have a round footprint with the craft positioned at the center. While this assumption leads to an inaccurate representation for many sensors, more general behaviors are possible by not limiting the simulation to a specific footprint.

Another assumption is that once airborne, a swarm does not change altitude. This assumption limits the complexity of the movement problem to a two-dimensional case. This can be considered reasonable because generally it is not necessary for autonomous vehicles to change altitude often while performing a mission. While altitude changes do occur during a mission, most profiles specify flying at a constant altitude for a defined portion of the sortie. This simplifying assumption relieves the need to incorporate complex cost functions to determine the optimal altitude.

The final assumption is that the mission planner has knowledge of the area in which the flight occurs, and has a planned route for that flight. As stated in Section 1.2, the swarm movement is separated from the path planning aspect of a mission. This allows research efforts to be concentrated on the interactions of the basic rules rather than on agent path planning requirements.

1.4 Research Goals

The goal of this research is to develop a means of creating desired emergent behavior in a swarm of autonomous UAVs. In order to accomplish this goal, three objectives must be accomplished. These are:

- Develop a swarm model that is capable of moving through a given landscape along a path specified by waypoints while avoiding threats.
- Define a set of behaviors for testing.
- Develop an automated method for finding the necessary set of rule interactions to cause a desired emergent behavior.

In order to accomplish these objectives, a swarming algorithm is defined in Chapter 3 that simulates the interactions between multiple UAVs. This model utilizes a matrix of coefficients which specify the emergent behavior of the swarm. Five different behaviors are defined in Chapter 3, as well as three different landscapes which are used to assess each behavior. A baseline swarm behavior is developed which utilizes a set of basic rules to move through the landscape while avoiding threats and moving towards a goal along a path specified by waypoints. Finally, an ES is utilized to perform a search for the values of the coefficient matrix resulting in the desired emergent behavior.

1.5 Research Sponsors

This research is sponsored by the Information Directorate, Air Force Research Laboratory (AFRL/IF), Wright Patterson Air Force Base, Ohio. The mission of the Information Directorate is “the advancement and application of Information Systems Science and Technology to meet Air Force unique requirements for Information Dominance and its transition to air and space systems to meet warfighter needs.” [2]. This mission is accomplished through research and development of embedded information systems capable of delivering timely information about the battlespace to the warfighter while surviving threats. The research discussed in this thesis supports this mission by developing a control algorithm for a network of mobile sensor platforms capable of avoiding threats, moving through the battlespace autonomously, and transmitting high-resolution fused data to the warfighter. The

control algorithm developed allows maximum flexibility to the warfighter and provides the ability to get information quickly and effectively through a highly survivable distributed platform.

1.6 Organization

This document is organized in the following manner: an introduction to the problem domain and the objectives of this research project is given in Chapter 1. Chapter 2 provides a summary of the current state of research in the field of autonomous vehicles. Chapter 3 gives the design of the swarm framework used as well as the design of the behaviors that are investigated. Chapter 4 provides the design of experiments for this research. Results from the experiments, analysis of the results, and conclusions reached from that analysis are given in Chapter 5. Finally, Chapter 6 provides a summary of the research effort and suggestions for future research in the field of swarms of UAVs.

2. Current Research in Distributed Control

The concept of swarms of robots is not a new concept. In 1987, Reynolds proposed a means of simulating the movement of birds in a computer simulation [73], Beni and Wang first coined the phrase *swarm intelligence* in 1989 [12], and Parker discussed *cooperative behavior* among robots in 1993 [63]. Since these initial efforts into the field of multi-agent cooperation, research has begun to polarize into several different types of cooperation. Dudek captures the different facets of cooperative systems in his proposed taxonomy [26]. This taxonomy classifies systems based upon their processing capability, communications capabilities, number of agents, composition, and configuration flexibility. This chapter uses Dudek's taxonomy in order to categorize contemporary research efforts in the area of cooperative multi-agent systems.

This chapter is organized into eight sections. Section 2.1 discusses terms and definitions that are used throughout the remainder of this document. Section 2.2 provides a description of the problem domain addressed in this research effort. Section 2.3 summarizes contemporary research efforts in the fields of cooperative robotics, swarm intelligence, and autonomous cooperative control. Section 2.4 discusses efforts in the field of cooperative wide area search munitions control. Section 2.5 discusses recent efforts to develop a theory of convergence and stability for swarm models. Section 2.6 discusses various methods of path planning. Section 2.7 discusses different types of agent behavior that have been successfully used to implement swarm architectures. Section 2.8 discusses behavior learning techniques proposed in the literature. Finally, section 2.9 summarizes this chapter.

2.1 Terms and Definitions

In order to adequately discuss the topic of cooperative multi-agent systems, it is important to define the terminology used. Many researchers use terminology that is specific to their own field of expertise. However, since multi-agent systems are inherently interdisciplinary, many different terms are used to discuss the same topic. For example, Mataric [51] refers to a system of multiple robots as a multi-agent system and the complex

behaviors that result from interactions among the robots is referred to as group behavior. Reif and Wang [72] on the other hand, refer to a system with multiple robots as a Very Large Scale Robotics System (VLSR). Beni and Wang [12] refer to multi-robot system as a swarm. Since this research deals with a simulated system (rather than referring to robots, birds, or planes) each individual within the system is referred to as an agent. The overall collection of agents is referred to as a swarm.

In a swarm model, two types of definitions exist for the term “behavior”. The first behavior definition refers to the actions of an agent in response to its environment and internal state. For this research, this behavior is called the local behavior. The local behavior refers to an individual agent, and does not consider how other agents are affected by the action of that individual. The second type of behavior is the emergent behavior. This behavior is the result of interactions between all of the agents in the swarm.

Over time, researchers have used different definitions to describe local behavior and emergent behavior, as well as the rules or equations which lead to each type of behavior. For example, Mataric [51] refers to local behaviors as *basic behaviors*. These basic behaviors are defined as a behavior that “either achieves, or helps achieve, a relevant goal”. In [53], Mataric uses the definition given by Steels in [80] to describe the resulting systemic reaction to the interaction of basic behaviors as “emergent properties”. It is important to make the distinction that the behaviors described by Mataric are in relation to the actions of the agent. The actual rules or mapping of sensor inputs to actuator outputs result in a basic behavior, but are not in themselves, considered to be a basic behavior.

In Reynolds’ work [73], [74], the actions of individual agents are referred to as *behaviors*. Reynolds does not define the resultant systemic behavior in his work, but rather refers to the result of simple interactions. The actual interactions in Reynolds’ model come from the application of equations that define a simulated force between agents. As opposed to Mataric, Reynolds does not make a distinction between these equations and the resultant movement of the agent.

In Reif and Wang’s work [72], the mechanism used to drive group behavior is called *social potential fields*. This field consists of possibly many equations chosen *a priori* for

group interaction. Reif and Wang refer to the potential field equation used by an individual agent as a *force law*. The combination of these forces make up the social potential field. The distinction between the actions of the agents, and the equations used to specify the force laws for an agent is not clearly stated in this work. The systemic result of the social potential field is referred to in this work as a behavior.

Since each of these systems use slightly different definitions for individual agent behavior, overall system behavior, and the actual rules associated with the agent behavior, it is necessary to clarify the use of these definitions in this work. In this document, the functions which provide inputs to the agents' control algorithm are referred to as rules. These functions map sensor inputs to two-dimensional vector space in the form of a desired direction. Chapter 3 provides details of the rules used in this research.

When individual agents interact with each other through the use of rules, the swarm as a whole exhibits observable behaviors that sometimes appear chaotic, and at other times converge to a structured formation or action. This observable effect can be referred to as emergent behavior [65], swarm intelligence [12], or simply behavior [72]. Since emergent behavior, and behavior both have the same connotation when discussing the overall observed actions of the swarm, these terms are used synonymously in this discussion.

In classifying rules that are applied to individual agents, two major types of rules are found in literature - reactive and deliberative [85]. A reactive model makes decisions based upon its current state and knowledge of its domain. This knowledge can either be global or local knowledge, and can be static or dynamic. This leads to a system that is seldom capable of achieving an optimal solution, but is very robust in a dynamic and noisy environment [85]. A deliberative model on the other hand, makes decisions several steps into the future and relies heavily upon an accurate model of its environment for the entire period of time for which it is reasoning. The deliberative model also utilizes current state information, as well as either local or global knowledge of the agent's domain. Such a model is best suited for environments which are reasonably stable and well understood [3]. Many of the systems discussed in this chapter utilize a hybrid approach to the reactive/deliberative models. Rather than reacting purely to sensed knowledge at any given time, many of the systems described utilize partial knowledge of the environment in order

to provide better decision-making capabilities to the agents. This hybrid rule model is used in the framework developed in this thesis.

2.2 Problem Domain Definition

The goal of this research is the development of emergent behavior for a group of autonomous agents. This behavior is tested in a potentially hostile environment consisting of a set of threats, a set of waypoints, and a single goal. In keeping with the objectives developed, behaviors must be capable of maintaining the desired behavioral traits throughout the landscape. They must also provide flexibility for agents to avoid threats and collisions, and move towards the goal via established waypoints.

The problem domain is expressed mathematically as a set S, T, W, G, r, B such that:

- S is the set of all agents within a swarm
- T is the set of all threats $\{t \in T | \text{all threats in the environment}\}$
- W is the set of all waypoints $\{w \in W | \text{all waypoints in the environment}\}$
- G is the set containing a single goal, $|G| = 1$
- r is the set of rules utilized by agents in S for movement
- B is a set of behavior matrices δ_b such that application of the rules r to the matrix δ_b results in the emergent behavior b

The rules contained in r consist of simple rules which map sensor inputs and agent state information to a desired direction vector. Each rule in r produces a direction vector, and these vectors are combined to form a local behavior through the use of the behavior matrix, δ_b . One of the objectives of this thesis is the development of values for δ_b in order to create the desired emergent behavior b . Since the characteristics of a particular behavior can be highly subjective, this is not treated as an optimization problem, but rather as an exploration problem in which multiple answers are compared to a pedagogical baseline in order to determine how well each answer conforms to the desired behavior.

2.3 Contemporary Research

In order to discuss the qualities of existing swarm models in a meaningful manner, it is necessary to classify each architecture with others of similar assumptions and capabilities. Dudek [26] provides a taxonomy of swarm systems which classifies existing architectures. This taxonomy relies upon the assumptions and capabilities of an architecture in order to classify it in a manner that allows for direct comparison with another architecture that shares the same taxonomic labels. This taxonomy decomposes the architectures into seven different categories: size, communication range, communication topology, communication bandwidth, reconfigurability, unit processing capability, and swarm composition. Each of these categories is discussed briefly in the following paragraphs.

The size category refers to the total number of agents within an environment. While Dudek classifies all agents within an environment as being part of the same collective, in a real-world application it is possible to have multiple collectives performing multiple missions within the same environment with or without any form of collaboration between the separate collections. In order to avoid this ambiguity, this research uses the term size to denote the number of agents assigned to a particular swarm. This allows us the ability to maintain multiple swarms in the same environment, and differentiate between swarms. The different size classifications are SIZE-ALONE, SIZE-PAIR, SIZE-LIM, and SIZE-INF [26]. SIZE-ALONE refers to a swarm containing only one agent. Although, by definition, this is not a swarm, it allows us to classify single-agent systems within this taxonomy. SIZE-PAIR is used to describe a swarm containing two agents. SIZE-LIM is used to describe a swarm with a limited size greater than two. SIZE-INF describes a swarm architecture that relies upon infinitely many agents. While this final category is not possible in the real-world, Dudek gives an example of a notional search and rescue swarm in which the environment is filled with agents until either the objective is met, or every space in the environment is occupied by an agent. This problem illustrates a brute-force solution method requiring a swarm of infinite size to work in all environments.

Communication range is used to describe the distance that agents are capable of communicating. This category is only concerned with the intra-swarm communication, and does not take secondary communications devices such as satellite relays, etc into con-

sideration if they are not used to directly communicate with other agents in the swarm. Communication ranges can fall into either the COM-NONE, COM-NEAR, or COM-INF category. COM-NONE is used to refer to a situation in which no *explicit* communication takes place between agents. While agents can still communicate through *implicit* means such as behavior changes, environmental manipulation, etc, architectures falling under this category do not directly communicate between agents. COM-NEAR applies to those systems that assume some limit to the communications range. This limit can be arbitrarily large. However, it is usually constrained by physical limitations of the communications architecture. COM-INF refers to a communications system in which any agent can communicate with any other agent regardless of distance. This is a standard simplifying assumption used in many of the architectures discussed in this chapter however, it is impractical for real-world situations due to the physical limits of the communications system. This can be assumed, if the swarm is designed to always remain within the communications limit.

Communication topology is used to classify the different communications network topologies used in swarm architectures. These topologies can be classified in one of TOP-BROAD, TOP-ADD, TOP-TREE, or TOP-GRAPH. TOP-BROAD architectures utilize a broadcast communications topology whereby each agent broadcasts all communications. TOP-ADD refers to an addressing scheme that allows an agent to send messages to other agents utilizing their unique identifiers. TOP-TREE addresses architectures utilizing a static tree communication topology. Finally TOP-GRAPH describes architectures with a graph topology.

Communication bandwidth is used to classify which bandwidth assumption is made for a particular architecture. Bandwidth is considered to be either the medium used for explicit communication such as wireless transmission, or implicit communication such as proximity sensors. The possible categories for bandwidth are BAND-INF, BAND-MOTION, BAND-LOW, and BAND-ZERO. BAND-INF is used to describe architectures that assume that the cost of communication is negligible. BAND-MOTION is used to describe architectures that consider communications cost to be “of the same order of magnitude of the cost of moving the robot between locations” [26]. BAND-LOW refers to

a very high cost communication environment in which communications are seldom utilized. BAND-ZERO refers to a situation in which no communication, either implicit or explicit takes place between agents. This final category describes a system with no agent cooperation.

Collective Reconfigurability is used to classify the apparent entropy of a swarm. An example of this would be a swarm of bees reconfiguring very rapidly (high entropy) versus a flock of birds which reconfigure more gradually (low entropy). The labels used to decompose reconfigurability are ARR-STATIC, ARR-COM, and ARR-DYN. ARR-STATIC describes architectures with fixed formations that do not change over time. ARR-COM is used to describe a system in which the formation changes using coordination with other members of the swarm. ARR-DYN applies to architectures in which agents change relative positions reactively, as in Reynolds [73].

Processing capability is used to classify the computational model of individual agents. The categories for processing capability are PROC-SUM, PROC-FSA, PROC-PDA, and PROC-TME. PROC-SUM refers to a simple non-linear summation unit proposed in [36], PROC-FSA describes agents which use a finite-state automata for computation, PROC-PDA refers to agents with push-down automata computational models, and PROC-TME describes agents using a Turing Machine equivalent. This particular taxonomic category is not very relevant to the classification of the majority of contemporary swarm architectures due to the fact that PROC-TME is the predominant computation model found in contemporary research.

The last taxonomic characteristic, composition, is used to describe the heterogeneity of a swarm. The three labels used to describe composition are CMP-IDENT, CMP-HOM, and CMP-HET. CMP-IDENT describes systems in which the physical characteristics, hardware, and software of each agent is identical. This is the equivalent of a Single Instruction Multiple Data (SIMD) parallel processing scheme [29]. CMP-HOM architectures utilize agents which are physically the same, however they may have different software control routines. CMP-HET is used to describe agents which are different physically. Physical differences can be used to describe different sensor suites as well as actual form of the agent.

2.3.1 Contemporary Swarm Architectures. In order to classify existing architectures under Dudek’s taxonomy, it is necessary to define an ordering of importance for each taxonomic axis. Architectures are sorted based upon the primary axis, then within that axis, they are sorted along the secondary, and so forth. Since communication is generally the primary limiting factor in the development of real-world swarming algorithms, the communications range is chosen as the primary axis, followed by communication topology, communication bandwidth, and reconfigurability.

The communications range of an architecture has a strong impact upon the final design of that architecture. The ideal communications range is COM-INF because it allows agents to achieve semi-global knowledge through communication with agents in other parts of the environment. Richards, et. al. [75] uses this methodology in order to coordinate assignment and trajectory planning into a group of agents. This is accomplished by employing a method of “combining both assignment and trajectory design into a single Mixed Integer Linear Programming (MILP) optimization”.

Richards, et. al.’s method addresses a problem that assigns agents with differing capabilities to a set of waypoints in such a manner as to arrive at the target in a predetermined order based upon their particular capabilities. This problem domain is representative of SEAD mission profiles where a vehicle (or vehicles) arrive first to perform reconnaissance. The same vehicle or vehicles or different vehicles would then arrive to perform electronic countermeasures. Shortly thereafter, attack vehicles arrive to destroy the target, and finally surveillance vehicles assess any damage [75]. This sequence must be choreographed precisely based upon the capabilities of the different vehicles in use to effectively destroy the target with minimum threat to all participating vehicles.

In order to accomplish the requisite coordination, Richards, et. al. [75] propose a method that develops an assignment and trajectory schedule *a priori*. In order to develop the vehicle trajectories, the problem is expressed in terms of the vectors v_{\max} , ω , S , W , Z , K , Δ , t_D where

- v_{\max} is a vector of maximum velocities such that v_{\max}^i represents the i th vehicle’s maximum speed

- ω is a vector of maximum turning rates such that ω^i represents the i th vehicle's maximum turning rate
- S is a vector containing the initial positions of each vehicle
- W is a vector containing the position of each waypoint
- Z is a vector of no-fly zones such that (Z_{j1}, Z_{j2}) is the upper bottom left vertex of the j th no-fly zone, and (Z_{j3}, Z_{j4}) is the upper right vertex of the j th no-fly zone
- K is a matrix containing vehicle capabilities such that if $K_{i,j} = 1$ then vehicle i is capable of performing the necessary mission at waypoint j
- Δ is a matrix containing the time dependency for each waypoint such that if $\Delta_{i,j} = -1$ then at the time dependency at row j , waypoint i must have been passed, if $\Delta_{i,j} = 1$ then at the time dependency at row j , waypoint j is next, and if $\Delta_{i,j} = 0$, then there is no time dependency for waypoint i .
- t_D is a vector of time variances such that for time dependency j in $\Delta_{i,j}$, $\Delta t \leq t_D^j$

Using this formulation, the cost function

$$\min_{s,f,b,c} J = \bar{t} + \epsilon_1 \sum_{p=1}^{N_v} [t_p + \epsilon_2 \sum_{t=0}^{N_t-1} (|f_{x_{tp}}| + |f_{y_{tp}}|)] \quad (1)$$

is solved, where the forces f are the decision variables, and b and c represent the binary variables for waypoint visit and no-fly zone, respectively. The variables ϵ_1 and ϵ_2 are weighting factors that are small positive numbers (values not given).

In this scheme, communication bandwidth is reduced to BAND-ZERO since all of the coordination tasks are performed prior to mission execution, and the mission profile is then loaded into each agent prior to take-off. While the overall behavior of coordinated arrival at the designated target area is accomplished, this method does not address real-time requirements such as moving targets, pop-up threats, or loss of a vehicle.

While Richards, et. al.'s architecture does not perform any communication after take-off, the size of the swarm is still limited based upon the MILP computation. The waypoint assignment problem is equivalent to the set partitioning problem [21] in that

each waypoint must be covered by one and only one agent. Further constraints are added in the form of time dependencies. Since this algorithm is a Non-Deterministic Polynomial Time (NP) Complete algorithm, the time to perform the coordination task increases as a polynomial of n .

In [57], McLain utilizes a team-based control model that assumes a high level of autonomy among agents, while working to optimize a global objective. The scenario presented by McLain is a rendezvous situation in which each agent must arrive at the detection envelope of a target at the same time. Each agent is fully autonomous to plan its own route to the target, avoid threats, and minimize fuel usage. The agents coordinate based upon events in the environment, such as encountering a pop-up threat, or arriving at a predetermined boundary, such as the Forward Edge of the Battle Area (FEBA). Each coordination epoch is represented by a phase of flight: Phase I is the en-route phase, Phase II occurs when the agents reach the FEBA, Phase III occurs when an agent detects a threat, and Phase IV commences once final trajectories to the target are determined.

McLain [57] utilizes a finite state machine within each agent to control the current phase of the agent. At the transition to each phase, all the agents communicate their Estimated Time of Arrival (ETA) to the target’s detection boundary, and all ETA’s are communicated to all agents. This denotes a COMM-INF architecture with a TOP-BROAD topology and BAND-LOW communications cost. Based upon the information received from other agents, each agent calculates a “coordination function model” that represents the optimization of fuel cost and threat exposure for each feasible ETA. This vector is then passed to all agents. Once each agent knows the optimum ETAs for each individual, the optimal ETA for the team can be calculated by each individual.

Although McLain’s [57] scheme results in coordinated arrival time at the detection boundary of a given target, several disadvantages exist. For example, each agent must communicate its state to every other agent in the system at each phase transition. This results in a system that is only marginally scalable as the communication overhead increases exponentially with the addition of new agents to the system, or if global broadcasting is used, the amount of bandwidth required for communication increases as members are added. Second, coordination must occur for every pop-up threat encountered. In a dense

battle field environment where many pop-up threats occur, the communications overhead becomes a limiting factor to the size and efficiency of the system. This model cannot readily adapt to many pop-up threats as the time to communicate and optimize the trajectories does not lend itself to fast reaction times for a large system. Finally, cooperative movement techniques are not taken into account by this model. Since this model is designed specifically to solve the rendezvous problem, no effort was made to address moving agents cooperatively across the battlespace. This means that each agent is moving independently to the given target, and cannot be used effectively for sensor fusion while traversing a high-threat area.

Polycarpou, et. al. [69] [68] [67] propose a method of team control that performs a cooperative search of an enclosed region. The method proposed utilizes a path planning scheme that plans the $t + q + 1$ position at time t in order to move the agents through the landscape. In [68], this scheme is changed to an interleaved planning scheme that initially plans q steps, and then replans $t + q - r$ steps for $r = [0..k]$, $k < q$. In both cases, the path is planned using a weighted aggregate function that takes into account the certainty of the search area, the proximity to other agents, and the amount of fuel. In [68] and [67], a proximity force is applied to agents when they detect a neighbor within a certain distance and moving at a heading that sufficiently compares to its own. This force causes the agent to weigh points in the path that move away from the nearby agent as lower cost points.

This architecture is categorized as a COMM-INF architecture since each agent is capable of communicating with all other agents regardless of their distance from each other. The communication cost is determined to be BAND-MOTION since communication updates are performed at an interval c where $c \geq \Delta t$.

The overall search results of Polycarpou, et. al.'s algorithm yields a definite improvement in total area searched as compared to a Zamboni search pattern [1] and a random search pattern. Furthermore since predictive path planning is used, agents are capable of optimizing their path for several time steps at a time. This capability results in more efficient overall paths, and a lower communication frequency requirement among agents. This scheme does however, utilize a great deal of communication due to the need to share

search space certainty values for areas of the search space. Also, the predictive element of the algorithm leads to less ability to react quickly to pop-up or unknown threats.

One of the interesting aspects of Polycarpou, et. al.’s work is the ability to embed swarm-like behaviors into the predictive path planning algorithm by using a repulsion force. This force allows agents to avoid possible collisions early through small changes to the path rather than waiting until a certain proximity threshold has been passed before reacting. While this particular swarm instantiation does not concern itself with maintaining cohesion, it is possible to adjust the cost function used for the predictive path planning algorithm in order to maintain a swarm-like “cloud”. The abilities of this algorithm are very promising for further research.

Trahan [82] implements a swarm using particle-in-cell codes taken from physics simulations. These codes are used due to their well understood behavior, efficiency, and support of heterogeneous particles. Trahan shows the ability of the swarm to adapt to the tracking of two moving targets in several different situations.

Trahan [82] models each agent as an atomic particle. Each particle is governed by environmental and behavioral “forces” much like the kinetic and electrostatic forces encountered by particles at the atomic level. These forces are easily extensible and allow the particles’ behavior to be governed by many forces that sometimes conflict with each other. The main forces associated with Trahan’s model are separation, cohesion, alignment, and attraction to a target particle or particles.

While the interactions of different particles does not require implicit communication, explicit communication is required in terms of proximity, direction, and speed of neighboring particles. This suggests a COMM-NEAR classification on communications distance. Since communication is implicit in this case, the communications topology can be described as broadcast communications within a finite distance from a particle. This fits within the TOP-BROAD category. The bandwidth requirement for this system is BAND-INF since the cost of implicit communication is considered to be free.

Trahan’s [82] experiments do not address how changes to the forces applied to each particle affect the overall behavior of the system. While this method appears to hold

the best potential for investigations into swarm control, more information is required to determine the affects of various parameters on the overall behavior of the swarm. Furthermore, the actual equations used to achieve the reported results are not given, therefore this architecture could not be easily verified or tested.

Hayes, et. al. [34] performs experiments using team-based control in order to determine the advantage of using multiple agents to perform a searching function. Hayes utilizes both a real-world arena consisting of simple robots searching for a beacon, and a probabilistic simulation in order to generate his results.

In Hayes, et. al.'s [34] experiments, each agent is equipped with simple light sensors at four locations around the periphery of the robot. The beacon consists of a light beam. Each robot moves about the arena by turning between 0 and 270 degrees probabilistically. Collision avoidance is used and takes precedence over goal following behavior. When an agent senses the beacon, it moves directly to the beacon unless collision avoidance is required.

Cooperative control is accomplished by having each agent actuate a small light when it senses the beacon. Other agents then follow this light until they also see the beacon. Again, collision avoidance behavior takes precedence over all other behaviors.

While the communication in Hayes, et. al [34] is not explicit, the use of lights to attract other agents is a form of implicit communication among agents.

In his paper, "Cooperative Control of Robot Formations" [28], Fierro, et. al. proposes a means of controlling robots' formation and trajectory allowing for tasks such as "collaborative mapping and explorations, and cooperative manipulation". The control method employed utilizes a two-tiered hierarchy in which each robot generates a trajectory and then coordinates based upon a behavioral mode. This method results in a stable formation that can adapt to obstacles or threats in the planned trajectory. Communication is accomplished via wireless LAN transmissions. Due to the use of LAN transmissions as the means of communication, this architecture is classified as COMM-NEAR with TOP-ADDR topology and BAND-MOTION cost of communication.

The control aspect first performs a trajectory planning step in which each agent utilizes predetermined potential fields in order to generate a trajectory. Visco-elastic collision modelling is used to simulate a collision however, the model is developed in such a manner as to avoid any collisions in real-space. This is accomplished through the use of a virtual “safety envelope” around each robot. The envelope is calculated to be large enough for the agent to react to incursions before an actual collision can occur.

Once a trajectory is calculated using potential fields and simulated visco-elastic collisions, the robots perform cooperative movement via a second-tier controller. This controller develops cooperative behavior by switching between three different behavioral rules. The three rules used are separation-bearing control, separation-separation control, and separation distance-to-obstacle control [28]. Each behavior utilizes a specific rule set, and the agent must determine which rule set to use based upon sensor input. This allows each robot to “behave” according to its environment.

While the formation achieved using Fierro, et. al.’s [28] methodology is stable and adaptable, it suffers from the requirement to designate a lead robot. While this is not necessarily a problem in some circumstances, the inability of the formation as a whole to adapt to the loss of a leader is undesirable in a battlefield environment. The trajectory planning method proposed by Fierro, et. al. [28] however, offers a good step-off point for on-the-fly trajectory planning due to its ability to adapt to new threats easily, as well as its means of avoiding collisions.

Another disadvantage of Fierro’s approach is that each agent can only maintain a static set of rules. Since there is no provision for learning or adaptation of new rules, this limits the overall system to the set of behaviors that are attainable through the use of the defined rules.

Fax [27] develops a means of information flow that uses small amounts of information to help drive faster convergence to a stable formation. This method relies on graph and control theory and assumes that the communication network formed by the agents is a strongly connected graph (i.e. any two nodes can be joined by a path). Fax develops a proof which proves that the information flow converges for a given formation. This method

allows each vehicle to make local decisions based upon global consensus of the location of the center of the formation.

Fax [27] attempts to minimize communication among agents in his formation while still allowing each agent to have global information. For his experiments, he utilizes agents whose dynamics consist of “double integrators in the plane”. These agents are attempting to take up positions in a hexagonal formation based upon the location of its neighbors.

Fax’s [27] use of information flow allows the swarm to arrive at a consensus of the swarm’s center through the use of local communications. Using information flow in this manner allows the use of global metrics in a distributed system.

One of the disadvantages of Fax’s work is that he assumes that the communication network formed by the swarm consists of a graph structure that is strongly connected. While this assumption is reasonable for many formations and behavior dynamics, such a condition cannot be guaranteed in a real-world situation due to unforeseen dynamics in the behavior characteristics. Furthermore, Fax uses a uniform time delay model for his information flow. Such uniform time delays are not possible in a dynamic swarm environment due to the fact that communications take a finite amount of time to propagate through an entire swarm.

2.4 Cooperative Munitions

Cooperative wide area search munitions control is an area of current active research in cooperative agent systems. This field of research attempts to solve the problem of finding and engaging a number of targets with unknown locations [32]. The agents in this case consist of powered sub munitions endowed with scanning sensors (LIDAR) and capable of flying for approximately 30 minutes over the battle space. Coordination is used to improve the probability of finding, identifying, and successfully engaging targets in the landscape as efficiently and effectively as possible. Some efforts address the coordinated search task ([19] [35]) while others focus primarily on coordination of target identification and engagement ([32] [20]).

Gillen [32] addressed the cooperative munitions problem through the use of a weighted aggregate function that determines whether a particular agent engages a known target. This function accounts for the agent’s fuel state, the priority of the target, the relative motion of the agent with respect to the target, referred to as range rate, and number of agents servicing a target. Each of these values has a weight associated with it. Gillen adjusts the weight using a Response Surface Methodology (RSM).

The results obtained using Gillen’s algorithm yield a $\sim 5\%$ improvement in target engagement and kill rates as opposed to a non-cooperative search and destroy mission. This provides evidence that a cooperative framework is capable of improving overall mission effectiveness versus non-cooperative frameworks. A disadvantage to using this algorithm however, is that the results are shown to be highly specialized to the specific scenario in which they were formulated. This means that a set of weights capable of operating robustly on any given map may not be possible.

In [35] Hebert gives a method for minimizing the radar exposure of a group of agents using a hierarchical decision scheme . This method builds upon a single vehicle radar exposure model. Multi-agent coordination is achieved by each agent communicating radar exposure versus path length information to a central controller. The central controller then chooses the optimal path length that achieves the minimum cost for the team as a whole. This path length is then communicated to the agents. While a central controller is used in this case, it is possible for agents to communicate information to all agents and then solve the common table individually. While this requires a great deal of communication overhead, the central controller is eliminated. Hebert does not investigate this method and its ability to work in constrained environments with imperfect communication is unknown.

The scalability of the hierarchical decision algorithm proposed by Hebert has not been investigated. The nominal case of one radar is developed and results given, and the algorithm is expanded to the two radar case as well. Hebert claims that heuristic methods are required to deal with the increased complexity of using n radars. One of the heuristics proposed by Hebert is the use of a Voronoi diagram. Since the optimal solution tends to lie near the edges of this diagram, the heuristic can be used to obtain good, but sub optimal results. Another heuristic proposed is the minimax heuristic. In this method,

exposure to the strongest radar source is minimized. This method works well for some cases however, Hebert provides a counter example that shows that the minimax heuristic becomes trapped.

In [19] and [20], Chandler, et. al. propose a hierarchical control model. This model establishes a team leader that solves the overall task assignment problem, sub team leaders that are responsible for coordinating assigned tasks, and an agent level controller that executes the assigned task and performs trajectory optimization. The problem domain addressed in these papers is the cooperative munitions problem.

In Chandler's model, the upper level controller performs task allocation. This task allocation is discussed briefly in [19] and further defined in [20]. Several methods are proposed for solving the task allocation such as binary linear programming, iterative network flow, and iterative auction using Gauss-Seidel and Jacobi auctions. This method allows agents to be assigned to different sub teams during the execution of the search phase in order to optimize the task allocation.

Coordination in Chandler's model requires communication to pass information to other agents as it is discovered. This information consists of the probability distribution matrix that denotes the probability of a target's location. Agents coordinate based upon this probability matrix. Sub teams are formed based upon predetermined criteria in order to search high probability areas. A classification probability is also provided based upon the sensor look angle of agents. This model requires a minimum of two different scans in order to classify a target. This task is coordinated through the middle level controller.

The architecture proposed by Chandler provides a means of decomposing the problem space into task allocation, task coordination, and trajectory optimization. By decomposing this problem, agents can efficiently distribute computational tasks for a highly complex optimization task. While this eliminates guaranteed optimality, such decomposition allows the agents to quickly coordinate without requiring large amounts of computation power.

2.5 *Swarm Convergence*

In order for a specific behavior to emerge in a swarm, it must be able to converge to a specific static or oscillating formation. If the convergence properties of the swarm are unstable, a specific emergent behavior is not sustainable over time. The ability of a swarm to converge upon a desired behavior is not well understood. A general theory of swarm stability has yet to be developed, however several important steps have been taken towards this goal.

In [37], Jadbabaie uses the nearest neighbor rule for coordination of headings of autonomous agents. Using control theory, he proves that the nearest neighbor rule can “cause all agents to eventually move in the same direction despite the absence of centralized coordinated control and despite the fact that each agent’s set of nearest neighbors change with time as the system evolves”. In order for all n agents’ headings to converge, they must be “‘linked together’ via their neighbors with sufficient frequency as the system evolves.” A rigorous definition of “sufficient frequency” is not given due to the complexity of the analysis required to determine this value.

Jadbabaie’s biggest contribution to the area of swarm control is his theorem which states that “convergence of all agent’s...to a common heading is...certain provided all n agents are always linked together” via a connected graph. He goes on to prove that if a leader is used, “all n agents must converge to the leader’s [heading] provided all n agents plus their leader are linked together via their neighbors frequently enough as the system evolves”. A rigorous definition of “frequently enough” is not given as the dynamic system analysis becomes too complex to form a rigorous definition.

One of the main disadvantage of Jadbabaie’s work is the switching function used in his analysis of the communication dynamics. The models developed are limited in scope due to restrictions made to this switching function. While these models are helpful in understanding flocking behavior, they are not fully descriptive of larger systems’ actual switching functions. Jadbabaie’s work provides a foundation for understanding the need for communication among neighbors. He shows that by sufficiently frequent communications, convergence to a globally accepted value can be reached given a sufficiently long period of

time. This research effort uses neighborhoods to communicate information, and attempts to communicate as often as practicable in order to maintain pseudo global knowledge via the convergence shown in [37].

In [46] [47] and [48], an attempt is made to analyze the stability of a swarm model within a highly restrained communication topology. This work focuses on developing a set of movement rules that result in collision-free movement of each individual in the swarm in the midst of uncertainty and communication delays. In [48], the analysis focuses specifically on the ability of the swarm to converge on a (relatively) non-moving formation, and then extends the theoretical analysis to the ability of the swarm to converge on a moving formation. This work defines stability to be the ability of the control algorithm to achieve “asymptotic collision-free convergence and partial asynchronism [which] leads to finite time collision-free convergence...” [48].

The swarm model assumed in [48] consists of $N : N \geq 2$ agents with some physical dimension $\omega > 0$, and dimensionality $M : M \geq 2$. Furthermore, each agent has a proximity sensor which instantaneously senses another agent when that agent is within $\varepsilon : \epsilon > \omega$ distance. Finally, each agent has a “comfortable distance” $d : d > \varepsilon > \omega$ at which no attraction or repulsion forces are experienced with respect to other agents.

The communication topology assumed in [48] is a highly constrained chain topology where each agent i can only communication with its neighbors $i + 1, i - 1$, and the order of the agents cannot change over time. Furthermore, the movement of each agent is restricted to remain within a sector that is δ degrees wide and symmetrical about the line drawn from agent i to agent $i - 1$. This restraint is further restricted to require that agent i must remain within the overlap of all sectors formed by agents $i - 1..1$. Using these constraints in a non-mobile environment, it is shown that each agent’s movement can be restricted in such a way to provide guaranteed convergence of each agent to a comfortable distance from its neighbors $i - 1, i + 1$, even with large delays in updates of neighbors’ positions.

For a mobile environment, Liu, et. al. found it necessary to further constrain the movement such that each agent lies within a sector of size $\frac{\delta}{2}$ degrees. This restraint, coupled

with a bounded delay in position update information, allows each agent to converge to a comfortable distance in a finite time without collisions [48].

The main strength of Liu, et. al.'s approach to stability analysis is that they are able to prove that it is possible for a swarm to achieve collision-free convergence regardless of communications delays in a non-mobile environment, and that collision-free convergence is also possible in a mobile environment, provided that communication delays are bounded to be less than the position update frequency of each agent. This corresponds to the work done by [37] that states that communication within some bounded interval is required for convergence to occur.

While Liu, et. al. provide a good start in stability analysis of a swarm control algorithm, the assumptions impose unrealistic constraints on the swarm. Relaxation of these constraints causes the stability analysis to become too complex to provide any insight into the convergence characteristics of the swarm. Another weakness of this approach is that it requires a fixed communication topology in order to successfully converge. While it is possible to specify rules which results in maintaining a fixed communication topology, it is not necessarily desirable to do so.

Another area which Liu, et. al.'s work does not address is the actual motion of each agent. In their investigation, inertial motion was not taken into account, and the movements were only constrained by maximum step size, but not minimum step size. In a flying or otherwise fast moving swarm formation, these requirements cannot always be realistically met. This means that the collision-free convergence property may not be met due to violation of the assumptions necessary to guarantee such convergence.

2.6 Path Planning

In a fully reactive swarm architecture, it is not necessary for the swarm to be able to plan a path through the environment. Instead, a swarm only moves in the manner that most expediently satisfies the optimization criteria for the current time step. A well designed system can use this type of reactive behavior to create globally optimal behavior however, such a system requires *a priori* knowledge of the agent's environment. Rather

than make this assumption, several efforts have focused on the ability of an agent to act deliberately, that is, to plan one or more steps into the future in order to “choose” a path that best optimizes the agent’s movement criteria.

2.6.1 Path Planning Using Roadmaps. Much of the current research concerning path planning techniques focuses on the topic of roadmaps. Roadmaps attempt to represent a collision-free path from a starting location to a goal location using a graph structure. Due to the heavy computation required, roadmaps are created off-line using global knowledge of the environment [13]. This assumes that the agent’s domain is fully observable.

Probabilistic Roadmaps (PRM), first proposed by Kavraki [40], attempt to reduce the total search time by randomly sampling the environment for points that are within the collision-free configuration space (C-free) of the agent. This algorithm works in an iterative fashion by creating a randomly generated set of configurations that are in C-free. Each of these random configurations is then connected to its nearest neighbors if the connection does not cause a collision. This process is repeated until a sufficient termination criteria is met such as the number of edges generated in the graph, or the amount of time the algorithm has taken to create a graph [40].

Bayazit, et. al. [11] create a swarm system in which a PRM is used as a form of global knowledge to allow for the creation of behaviors in a highly constrained environment. The underlying swarm architecture utilized in this effort is that proposed by Reynolds [73]. In order to move the swarm through the environment, agents choose as the target the next node contained in a path to the goal. A goal steering behavior is then implemented to move towards that target. The standard intra-swarm interactions of separation, alignment, and cohesion are used.

Bayazit, et. al.’s use of the PRM method of integrating global path planning capability into the swarm creates a means for agents to move through a dense obstacle field towards a goal. Some disadvantages in this method exist however. For example, agents do not choose the same path within the PRM to get to the goal. This leads to a highly entropic swarm behavior in which members become separated and move singly towards the goal. While this allows members to reach the goal state, the overall effect is that agents

“trickle” in to the goal one at a time, rather than moving through the landscape as a cohesive whole. This type of behavior appears to be unstable in regards to convergence of a behavior.

Another method used for deliberative path planning is Rapidly-expanding Random Search Trees (RRTs). This path planning algorithm is proposed by LaValle [45]. This method creates paths for non-holonomic vehicles, that is, vehicles that are constrained to steering while maintaining forward motion [41]. A RRT builds a map of C-free by beginning in an initial position q_0 , and choosing a random configuration q_{rand} . Using this configuration, the nearest neighbor to q_{rand} that is within ρ distance of q_0 is added as a vertex to the tree. This vertex is set as the new q_0 and the algorithm is iterated until a termination criteria is met. LaValle defines the closing criteria to be the number of vertices in the RRT.

The RRT method proposed by LaValle provides a very fast method for determining paths through C-free. The advantage of this method is that it is strongly biased towards unexplored regions, thus filling in unexplored areas of C-free very quickly. This method works quickly enough that it allows for real-time replanning, and results are generally within a factor of 1.3 to 2.0 times longer than the optimal path [45]. Another advantage of this algorithm is that it can be tailored to work for any kind of kinematic model by adjusting the selection criteria to include constraints of the physical model.

The main disadvantage of LaValle’s algorithm is the random nature of new state generation. While using random generation allows for fast construction of the tree, it also leads to non-optimal solutions. When the optimality of the solution is not as important as the feasibility, this method provides less desirable paths for agents.

2.6.2 Artificial Potential Fields. Artificial Potential Fields (APFs) is a method of reactive path planning proposed by Latombe [44]. This method creates fields of artificial “force” with an attraction towards the goal and repulsion fields around objects and threats. Using this field, it is possible for an agent to follow the force lines to the goal. This algorithm produces one predominant problem however, in that large areas of low or zero magnitude may exist in the field where the goal attraction force and obstacle repulsion forces cancel

each other. These areas are referred to as local minimums in the potential field. Escaping these local minimum areas is a topic of on-going research.

Caselli, et. al. [18] discuss a means of parallel path planning for a potential field landscape. The algorithm proposed utilizes the Probabilistic Roadmap Algorithm first proposed in [40] in order to develop a search graph of the configuration space. The algorithm then begins searching the graph by attempting to move down the potential gradient. The main contribution of this algorithm is Caselli’s use of techniques for escaping local minima. Rather than utilizing random brownian motion in order to escape the local minimum as proposed in [10], Caselli, et. al. propose the use of two different evasion techniques dubbed “StraightLine” and “StraightLineSelect”. These two techniques attempt to escape a local minima without the high computation penalty found in brownian motion escape techniques. “StraightLine” works by selecting a direction randomly in C-free and moving in that direction until it reaches an obstacle, or the potential is less than the potential at the original position. If a new lower gradient is not found along the line of motion, the direction is thrown out and a new direction is chosen. This behavior eventually leads to a local minimum that might also be the global minimum. StraightLineSelect performs in a similar manner to StraightLine with the exception that it prunes poor direction choices early through the use of a state machine.

Both StraightLine and StraightLineSelect are shown to perform very well for “easy” local minima. However, certain circumstances can be found in which these algorithms lead to higher computation than random brownian motion before an acceptable escape is accomplished [18].

The escape forces proposed by Caselli, et. al. offer a possible means of performing local minima escape in a relatively easy local minima. Since real-world problems often lead to large, complex interactions among fields, it is not likely that many of these “simple” minima exists on a standard landscape. The amount of speedup possible from using this scheme in simple circumstances however, may allow for reasonably fast predictive path planning in real-time environments.

Vadakkepat [86] proposes a means of developing optimal potential fields via the use of a Multi-Objective Evolutionary Algorithm (MOEA). This algorithm utilizes a goal-seek function, obstacle avoidance function and minimum path function in order to develop optimal potential functions. Vadakkepat also proposes the use of an “escape force” that allows the algorithm to avoid stagnation in a local minimum. A standard path planning algorithm is applied using the evolved potential field and escape force.

For this algorithm, a reactive agent utilizing potential fields for path planning is introduced. The agent follows potential fields unless it determines that certain criteria have been met signifying that it is trapped in a local minimum. The criteria used by Vadakkepat consist of the following two equations:

$$\frac{F_a - \sum_{\forall i} F_r^i}{\sum_{\forall i} F_r^i} < b \quad (2)$$

$$\cos(\angle F_a - \angle \sum_{\forall i} F_r^i) < -\cos(c) \quad (3)$$

In these equations, F_a represents the attractive force of the goal, F_r^i is the repellant force of the i^{th} obstacle, and b and c are determined by the MOEA algorithm.

When equations 2 and 3 are satisfied, the escape force is calculated using equation 4.

$$F_e = (\frac{1}{dD_{ro}^m})(|\cos(\angle F_a - \angle \sum_{\forall i} F_r^i) - \cos(c)|) \quad (4)$$

For equation 4, D_{ro} represents the distance from the agent to an obstacle, and the parameters c , d and m are determined by the MOEA.

The cost functions consist of “goal-factor”, “obstacle-factor”, and “minimum-path-length-factor”. These functions are shown in equations 5, 6, and 7 respectively.

$$f_p = \begin{cases} 0 & \text{robot at target point} \\ F_a + \min_{t \in [0, t_g]} (\|p(t) - P_g\|) & \text{otherwise} \end{cases} \quad (5)$$

P_g is the location of the goal, $p(t)$ is the location of the agent at time t , t_g is the time to reach the goal.

$$f_o = \begin{cases} 0 & \text{no collision} \\ F_o + \max_{t \in [0, t_g]} (R_o - \|pc(t) - P_o\|) & \text{otherwise} \end{cases} \quad (6)$$

$pc(t)$ is the point at which the agent collides with an obstacle at time t , and P_o is the location of the nearest obstacle.

$$f_{\text{path length}} = \text{length of entire path} \quad (7)$$

The MOEA is designed to minimize all three equations with preference for the solutions on the Pareto front which achieve 0 for equations 5 and 6. Vadakkepat does not specify the manner in which points are chosen from the Pareto front if they do not achieve a 0 value for f_p and f_o .

Vadakkepat's algorithm achieves a means of producing smooth paths with near-optimal path lengths. Furthermore, this algorithm is capable of developing potential fields in real-time, thus allowing for adaptation to a changing landscape. Through the use of an escape force, this algorithm is capable of escaping local minima. This ability is shown using anecdotal evidence from test runs however, Vadakkepat does not rigorously show the capabilities of the escape force portion of his algorithm.

Although this algorithm is useful for developing potential fields and is robust to changing landscapes, it is only intended for the path planning of a single agent. While this problem is easily overcome, it is not known how interactions between agents affect the overall structure of the potential field. Another disadvantage to this method is that it only addresses point obstacles and a single point goal. While it is relatively simple to extend the algorithm to account for multiple goals, the ability to adapt to arbitrarily shaped obstacles is not easily developed. Another drawback of Vadakkepat's work is that a relatively small number of obstacles are used. It is unknown as to how well this algorithm scales to larger numbers of obstacles.

While the local minima problem is the primary drawback of using potential fields, several other problems are inherent in this model. One of these problems occurs when an agent's direction of movement lies along a line that passes directly over the center of a repulsive field. If the repulsive vector is exactly opposite to the attractive vector of the goal, and both vectors lie on the same line as the agent's direction vector, then no action will be taken by the agent to turn away from the obstacle [73].

2.7 *Types of Behavior*

Contemporary literature lists many different behaviors that can be applied to multi-agent systems. These behaviors generally apply to the individual agent and are executed at a local level in the system. In attempting to define multi-agent behavior, Mataric introduces the concept of a basis behavior [51]. According to Mataric, a basis behavior set should consist of only those behaviors that “either achieves, or helps achieve, a relevant goal” that cannot be reached through the use of behaviors outside of that set. Mataric further constrains this set of basis behavior such that no behavior in the basis set can be reduced to a behavior or group of behaviors outside of that set.

One of the foundational papers in swarm behavior is “Flocks, Herds, and Schools: A Distributed Behavior Model” by Reynolds [73]. In this work, Reynolds proposes a rules-based algorithm that utilizes three rules to maintain swarming characteristics. These rules are collision avoidance, velocity matching, and flock centering. In a later paper these rules are redefined as separation, cohesion, and alignment [74]. Reynolds proposes the application of these three rules as vectors with the magnitude calculated using some function $f(d)$ where d is the distance from the perceived center of a group of neighbors. The three vectors can then either be aggregated, or prioritized in some order to create a final direction for the agent.

Reynolds also proposes a means of identifying neighbors based upon biological limitations such as the field of view of a bird [73]. This limits the interaction with neighbors that are further away, and causes localized behavior to occur. The advantage of this scheme is that the size of a flock can grow to any size, but the individual agent is only aware of a small neighborhood and therefore is not directly affected by the overall size of the

flock. This also reduces communication requirements to include only those agents in a local neighborhood.

While Reynolds' primary concern in [73] is the visual appeal of the swarming behavior, the rules defined have found use in many research efforts up to the present such as [84] [81] [88] [51] [72]. This method has been shown to be robust and extensible for swarm simulation, and allows for many areas of algorithmic improvement. One of the aspects of this algorithm that shows the most promise for improvement is the weighting scheme of the rules.

In [3] [51] [52] [53], the concept of "basis" behaviors is introduced. These behaviors are defined as "1) required for generating other behaviors and 2) ... a minimal set [of behaviors] the agent needs to reach its goal repertoire" [52]. The basis behaviors defined are safe-wandering, following, dispersion, aggregation, and homing [51]. The safe-wandering behavior attempts to randomly move about the environment while avoiding collisions with obstacles and other agents. The following behavior attempts to line up with another agent using simple rules such as "If the detected agent is on the right, turn right" [51]. Dispersion and aggregation are corollaries of each other in which agents attempt to either move away from some central location, or move toward some central location while at the same time avoiding collision with other agents and inert obstacles. Homing simply causes an agent to move to a predefined location while avoiding obstacles.

In [51], experiments are performed to determine the effect of non-hierarchical distributed control on the ability of agents to accomplish the specified task. Tests were performed using up to 20 robot platforms, and testing aggregation and dispersal behaviors. For both behaviors, the non-hierarchical results closely mimicked the hierarchical results for the amount of time required for the system to converge, with the standard deviations of the two sets of results overlapping in all cases. The general trend observed was that large numbers adversely affect the convergence time, and distributed control leads to more robust behavior in terms of the system's ability to deal with agent failures. One hypothesis offered by Mataric based upon the data from these experiments is that "more complex strategies requiring individual agents to perform recognition, classification, and representation might be required to significantly improve group performance". While this

hypothesis has not been proven, the current direction of research in the field of multi-agent systems has been moving towards more complex agents.

Another experiment performed in [51] is the aggregation and switching of different behaviors in order to combine several basis behaviors to drive a specific global behavior. The global behaviors pursued were flocking, and foraging. In the flocking model, aggregation, dispersion, homing, and safe-wander are combined via a weighted sum function. The values for the weights were determined “experimentally, from the dynamics and mechanics of the agents, the ranges of the sensors, the agents’ turning radii, and their velocity” [51]. This scheme resulted in robust flocking behavior that is resistant to individual failures.

One area mentioned by Mataric as a possibly fertile area of research is the development of aggregating methods for basis behaviors. The development of robust, adaptive methods for on-line adaptation would allow emergent behaviors to change simply through correct weighting of the basis behaviors. This research investigation attempts to develop such a method of on-line adaptation.

Triani, et. al [83] propose a method to control the behavior of a group of agents using a probabilistic selection methodology. The algorithm proposed uses sensor inputs to determine the agent’s environment. Behaviors are chosen using a roulette wheel selection scheme with some probability based upon a context function. The context function evaluates sensor inputs to calculate the probability value.

The S-bot agent model proposed by Triani, et. al utilizes three different behavioral sets - movement, light, and grip. Each agent is capable of gripping its neighbor, turning on or off a light that is used as a beacon for other agents, and moving. Each of these three behavioral sets contains multiple basic behaviors. Triani, et. al describe the different movement behaviors in terms of results-based movement such as light attraction, light repulsion, robot attraction, robot repulsion, and random movement.

Triani, et. al.’s proposed method for determining the correct basic behavior at each time step is to use sensor inputs to determine the context of the robot. These sensors are then factored into a context function $h(s) \in \mathbb{N}$. The natural number returned by this

function is used to select a column of a behavior matrix. The behavior is then chosen based upon a roulette wheel selection using the probability values stored in the matrix.

This method of choosing behaviors relies upon a stochastic behavioral method. While this allows the system to be more adaptive than a deterministic rules-based control algorithm, the composition of the probability matrix used to determine behavior selection becomes very important. Triani does not suggest a means for generating this matrix, nor is any analysis performed upon the usefulness of certain behaviors in accomplishing a given task.

2.8 *Learning*

In [50], Mataric discusses a learning scheme used to develop behavioral constructs leading to accomplishing the desired goal. This learning scheme consists of a set of basis behaviors, and a set of on-line estimators. The estimators use heuristic functions in order to determine the efficacy of a particular behavior or sequence of behaviors in accomplishing some sub-portion of the overall task. The agents begin attempting to accomplish the given goal by selecting behaviors via a switching mechanism, and positively or negatively reinforcing behaviors based upon heuristic values. Reinforcement occurs when a particular sub-goal is accomplished. Behavior switching occurs based upon sensor events.

The overall learning scheme presented by Mataric [50] results in successful learning of the behavior combinations and switching scheme required to cooperatively accomplish the task of foraging for food and bringing it back to the “nest”. This learning scheme is fast (roughly 15 minutes per experiment before convergence of behaviors occurs) and capable of learning “on-line”.

Many other learning control schemes exist in the literature such as [3] [30] [49] [8] however, this research does not attempt to perform agent learning of controls, but rather the development of rule weights based upon a specified emergent behavior.

2.9 Summary

This chapter has developed terms and definitions that are used throughout this document. Also, a problem definition is given that describes the overall area of interest for this research project. Finally, contemporary research efforts have been discussed in the areas of swarm architectures, path planning techniques, and artificial potential fields. The major swarm architectures were categorized based upon Dudek's taxonomy.

Three systems discuss which show strong potential for providing on-line behavior adaptation are Reynolds' boids [73], Mataric's basis behaviors [51] and Reif and Wang's potential fields [72]. An objective of this research investigation is to develop a framework for testing swarm behaviors. This framework uses portions of these three systems in order to leverage advantageous of each one. Details of the framework design are given in Chapter 3.

3. *Swarm Algorithm Design*

3.1 *Introduction*

The design of a simulation is a complex undertaking which requires that many aspects of the real world be discretized, estimated, or ignored. As the level of detail of the simulation model increases, the fidelity of that model with the real world increases, but the computational overhead increases as well. In the design of a swarm framework for this investigation, it is necessary to consider these tradeoffs and determine a level of detail that provides acceptably accurate results, while not requiring excessive processing time.

This chapter gives a detailed description of the design of the swarm algorithm developed for this research investigation. A high level view is provided to highlight the interactions between subsystems, and each subsystem is described in detail. Finally, metrics are defined for the evaluation of the algorithm's performance, and a search algorithm is discussed for use as a means of searching for desirable weights for behavior adaptation.

3.2 *High Level Design*

Due to the need of each agent within a swarm to adapt to different behavioral characteristics based upon its environment, it is necessary to define a robust strategy for behavioral adaptation. This strategy must be computationally inexpensive in order to minimize the on-board computing requirements, must lend itself to a distributed environment with little or no explicit communication among agents, and must be capable of realizing the desired behavioral traits without requiring global knowledge of the swarm. Furthermore, the algorithm must result in a stable formation, that is, it must be capable of sustaining some steady state configuration or oscillation between several configurations given an infinite amount of time [48].

In order to develop an algorithm capable of overcoming the problems described above, we must use concepts from several different disciplines. In order to maintain a stable formation, a swarm-like behavior is used. This swarm-like behavior is derived from the system proposed by Reynolds in [73] and [74] as well as Mataric's method of multi-agent control proposed in [51] [52] [53] [50] [54] [55] and [14], and modifications to the swarming

algorithm proposed by Kadrovach in [38] [39]. This behavior results in a cohesive group that moves in a coordinated manner while avoiding threat zones and other agents in the swarm. The ability to move towards a goal and avoid threats in the landscape is provided through the use of artificial potential fields (APFs) as suggested by Reif and Wang [72] .

Different swarm behavior is maintained through the use of several different swarm modes as described in Section 3.3.8. Finally, in order to provide adaptation based upon the local environment at a given time step, multiple fitness functions are used as weight coefficients in an aggregate vector in order to adjust characteristics of the inter-swarm and intra-swarm rules. These fitness functions are formulated to describe the state of an agent based upon local information.

3.3 Swarm Model

The swarm model implemented for this research consists of characteristics found to work well in other simulations and real-world experiments [74] [72] [53] [38]. In order to work together effectively, the various algorithms used must interact in a well defined manner. Each agent moves through the environment by determining the direction vector, \vec{d} , that satisfies the following equation:

$$\vec{d} = \sum_{r=1}^{|\tau|} \tau_{r,1} \quad (8)$$

Where τ is a 4×1 matrix which is calculated using equation 9.

$$\tau = \begin{bmatrix} \omega_{\sigma} \vec{\sigma} \\ \omega_{\eta} \vec{\eta} \\ \omega_t \vec{t} \\ \omega_g \vec{g} \end{bmatrix} \quad (9)$$

In the preceding equation, the vectors $\vec{\sigma}$, $\vec{\eta}$, \vec{t} , \vec{g} denote the direction determined by the cohesion, separation, threat avoidance, and goal seek rules respectively. The manner in which these rules determine a direction is discussed in Sections 3.3.3, 3.3.4, and 3.3.5.

The variables ω_σ , ω_η , ω_t , ω_g are weight coefficients that are calculated using a behavior matrix and a set of functions that describe an agents' state. The development of a behavior matrix and the equations used to calculate the fitness functions are discussed in Section 3.4.3 and 3.4.4.

In order to simplify calculation, and to account for local knowledge rather than global knowledge within each agent, it is necessary for each agent to perform calculations using a local coordinate system. This system is a standard cartesian coordinate system with the y direction pointing in the direction of movement of the vehicle. Section 3.4.1 discusses the local coordinate system of an agent. The local coordinate system is used to calculate cohesion, $\vec{\sigma}$, and separation, $\vec{\eta}$. Also, each agent calculates the relative bearing of all neighbors using the local coordinate system.

The global coordinate system must be used for calculation of phenomena outside of the agent's immediate vicinity. The assumption of a global coordinate system does not necessarily assume global knowledge of the environment, but rather assumes that an agent is aware of its position within a global coordinate system. This is equivalent to the knowledge provided by an inertial guidance system (INS), or global positioning system (GPS). It is also possible for agents to receive broadcast telemetry data from another source without requiring an increase in bandwidth as the number of agents in a swarm increase. For these reasons, an agent's knowledge of its position within the global coordinate system does not necessarily constitute global knowledge.

The rules that must be calculated using the global coordinate system are goal seek, \vec{g} , and threat avoid, \vec{t} . The reason these values must be calculated using global knowledge is that the location of threats and the goal are not necessarily within the immediate sensor range of the agent. Instead, the locations of these points is loaded into the agent's memory prior to flight, and calculations are performed based upon the pre-loaded positions. If a threat or goal moves while it is outside of the agent's sensor range, then the agent has no way of updating that knowledge without intervention from an outside source. This investigation does not attempt to explore how agents act under uncertainty.

```

foreach timestep  $t$ 
  build list of neighbors  $N_a$ 
  foreach  $n \in N$ 
    calculate  $\omega_{\text{periph}}$ 
    calculate  $\vec{\sigma}(n), \vec{\eta}(n)$ 
     $\vec{\sigma} = \vec{\sigma} + \omega_{\text{periph}} \vec{\sigma}(n)$ 
     $\vec{\eta} = \vec{\eta} + \vec{\eta}(n)$ 
  end
  foreach  $t \in L$ 
     $\vec{t} = \vec{t} + \text{threat potential}(\text{loc}(a))$ 
  end
   $\vec{g} = \text{goal potential}(\text{loc}(a))$ 
  calculate  $\vec{dd}$ 
   $\vec{s} = \text{trunc}(\vec{dd})$ 
   $\vec{d} = \vec{d} + \vec{s}/m_a * \Delta t$ 
end

```

Figure 2 Pseudo code for agent control loop

Once the desired direction, \vec{dd} , is calculated using equation 9, a steering vector, \vec{s} , is calculated as described in [74]. The equation for this step is given as

$$\vec{s} = \vec{d} - \vec{dd} \quad (10)$$

Once \vec{s} has been determined, the agent must truncate the value of \vec{s} in order to remain within the constraints of the vehicle's actuators. A detailed discussion of this procedure is given in Section 3.4.2. Finally, the new direction, \vec{d} , is determined using a first order Euler approximation of motion using the force \vec{s}/m_a , where m_a is the mass of agent a . Figure 2 gives a pseudo code representation of the program loop for each agent.

3.3.1 Potential Fields. The standard swarm model proposed by Reynolds [74] consists of three rules that allow for swarm interaction. In order to create goal seek and threat avoidance behavior, two rules have been added to Reynolds' basic algorithm, and one rule has been removed. All four rules use artificial potential fields to determine the correct direction and magnitude. In a Reynolds swarm model, these four rules can then be combined by either using a hierarchical system that sums rule vectors until no further

actuator capability is left for the control inputs [74], or by using an aggregate form in which the importance of each rule is assigned a weight, and all the vectors are averaged [38]. The resulting vector is the new desired direction of the vehicle [74].

The major problem associated with using potential fields is the existence of local minima in the search space. These local minima are sometimes easily escapable, such as in Figure 3, and other times very difficult to escape as shown in Figure 4. In systems where the potential field is used to represent an obstacle, it is necessary to perform some form of backtracking in order to find a path that is free of obstacles. For the purposes of this investigation however, the potential fields are used to represent threats posed by a radar or danger of visual detection. It is assumed that no threat represented is impassable, such as a wall or mountain. Since our flight model assumes an altitude of 20,000 ft, this assumption holds true for most regions of the world. This means that an agent that has found a local minima can escape that minima by either backtracking, or by accepting greater threat exposure and moving through the threat creating the local minima.

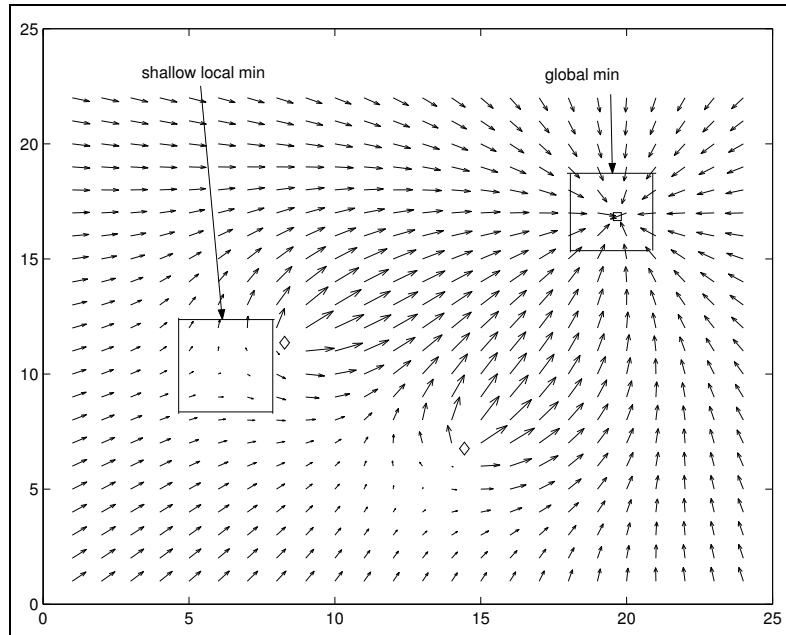


Figure 3 Shallow local minimum in a potential field

In this algorithm, the four rules are aggregated using adaptive weights. This method allows each rule to be modified based upon environment information. Another advantage

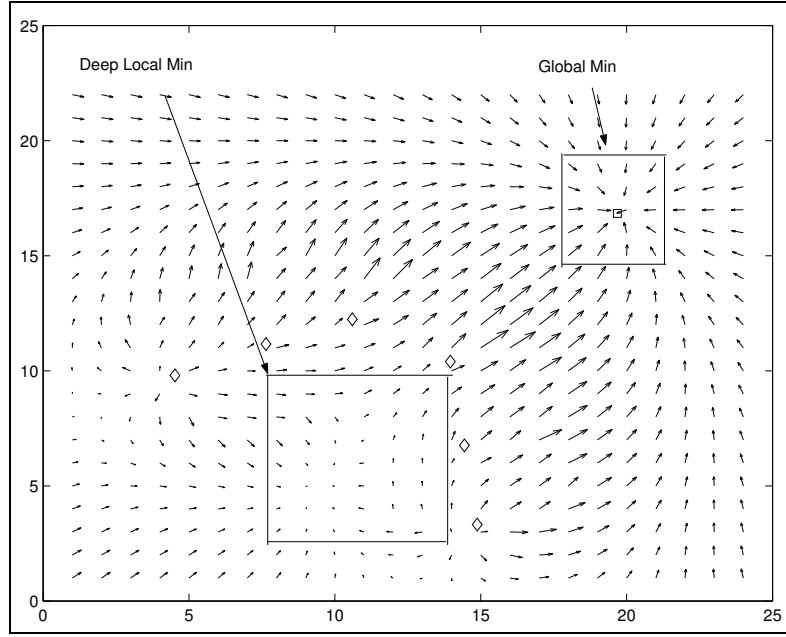


Figure 4 Deep local minimum in a potential field

to using an adaptive weighted aggregate method is that the local minimum problem is reduced by crafting fitness functions that detect local minima and allow agents to accept a less than optimal behavior in order to continue moving in the direction of the goal. This is analogous to a pilot accepting greater risk in order to reach a well-defended, high priority target.

3.3.2 Neighborhood Determination. The rules discussed in Section 3.3.3 utilize the concept of a neighborhood in order to specify interactions between neighbors. Depending upon the implicit and explicit communications assumed, the neighborhood can be static such as proposed in [48], consist of nearby agents as constrained by some distance or perception field as proposed in [51] and [38], or consist of all agents in the swarm such as implemented in [72].

The decision of what kind of neighborhood model to use depends upon several different factors such as communications assumptions, sensor capabilities, computational complexity, and desired outcome. Each of these factors can change the overall neighborhood policy.

An example of how assumptions and design decisions affect the neighborhood policy can be found in [72]. In this document, a global neighborhood policy is used where each agent is a neighbor to each and every other agent. This model assumes that the communications ability of the agents is such that each agent is aware, either through communication or sensing information, where every agent in the swarm is located in the landscape at any given time. The computational complexity of this model requires each agent to perform calculations on the position of every agent in the swarm. This means that for each timestep, $n(n - 1)$ calculations must be performed.

In [38] however, the neighborhood policy utilizes a vision blocking model, limiting the number of neighbors based upon a sensor “shadow” affect. This model assumes that each agent has a sensor capable of determining the bearing to each agent within a given distance provided that another agent is not between the sensor and the agent. This model goes on to specify a cone of shadow θ_v degrees wide, in which an agent cannot be seen. Based upon these assumptions, the neighborhood size for a steady-state swarm is reduce to $360/\theta_v$. While the simulation computation for this still results in an $O(n^2)$ growth rate, the real-world implementation does not need to check every agent to determine if it fits within the sensor shadow of another. This reduces the real-world complexity to $O(1)$.

In order to build a neighborhood, each agent first determines the agents that are within sensor range, sr . Using this information, each agent builds a list ordered by increasing distance from itself. Each agent then steps through this list starting with the nearest agent. For each agent a in the pseudo-neighborhood $N' \subset S$, $\overrightarrow{b(a)}$ is added based on equation 11. Figure 5 visually depicts this calculation.

$$N = N + a \quad \text{iff} \quad \frac{\overrightarrow{b(a)} - \overrightarrow{b(n)}}{|\overrightarrow{b(a)}||\overrightarrow{b(n)}|} > \theta_v/2, \forall n \in N \quad (11)$$

3.3.3 Reynolds’ Behaviors. Reynolds describes a control scheme whereby each agent calculates three vectors based upon a set of rules. The rules consist of the following:

- Attempt to move towards the perceived center of the neighborhood
- Attempt to move away from any nearby neighbors

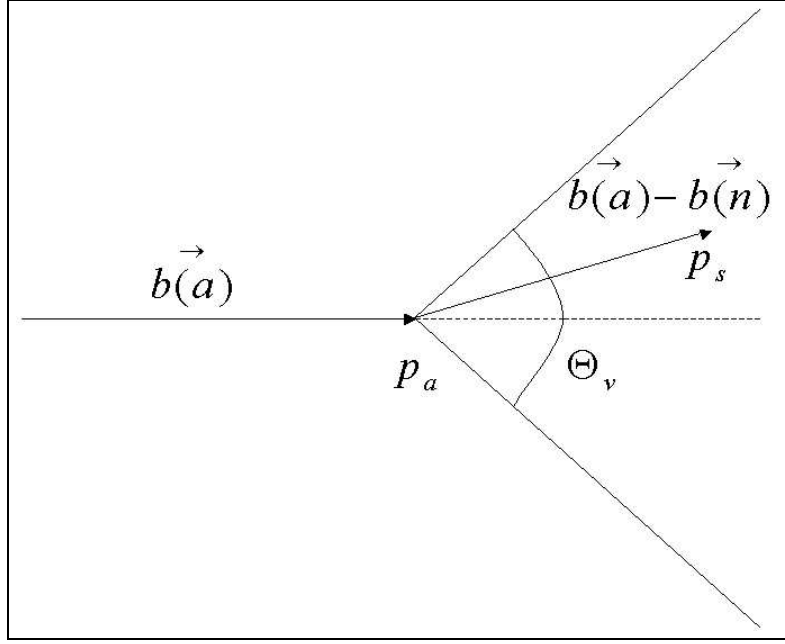


Figure 5 Depiction of sensor shadowing in neighborhood calculation [38]

- Attempt to move in the same direction and at the same speed as the neighbors

These rules are expressed mathematically in terms of three vectors: cohesion ($\vec{\sigma}$), separation ($\vec{\eta}$), and alignment ($\vec{\theta}$). Each of these vectors represents one of the rules listed in the preceding list. Cohesion is expressed as

$$\vec{\sigma} = \left[\frac{\sum_{n \in N_a} loc(n)}{|N_a|} - loc(a) \right] r(x) \quad (12)$$

Where $a \in S$ is an individual agent, $N_a \subset S$ is the neighborhood of agent a , and the function $loc(x)$ returns the location in x, y coordinates of agent x . The magnitude of $\vec{\sigma}$ is then calculated based upon an interaction function $r(x)$ that takes as its argument the current magnitude of $\vec{\sigma}$ which is the distance from agent s to the perceived center of the neighborhood N . The function $r(x)$ can be any function desired provided that it is continuous at all points $0 < x \leq \text{max distance}$. Where $0 < \text{max distance} < \infty$.

The function for $r(x)$ used in this research is a Gaussian distribution curve with mean of μ and standard deviation $sr = 50$ where sr is the range of each agent's proximity sensor. The number 50 is chosen for sr in order to represent a sensor range of 5000 m.

$$r(x) = e^{\frac{-(d-\mu)^2}{2\sigma^2}} \quad (13)$$

$$\sigma = \sqrt{\frac{(\text{sr} - \mu)^2}{2 \ln 2}} \quad (14)$$

For cohesion, $\mu = 0$. This results in a gaussian distribution centered at 0. For separation $\mu = \text{sr}$, centering the distribution at sr . Since the distance of a neighbor n cannot be more than sr units from an agent, the value of μ for each rule allows the separation magnitude to follow the “front” side of the distribution curve, while the cohesion magnitude follows the “back” side. Figure 6 depicts the magnitude of separation and cohesion over the range $0 \leq x \leq \text{sr}$.

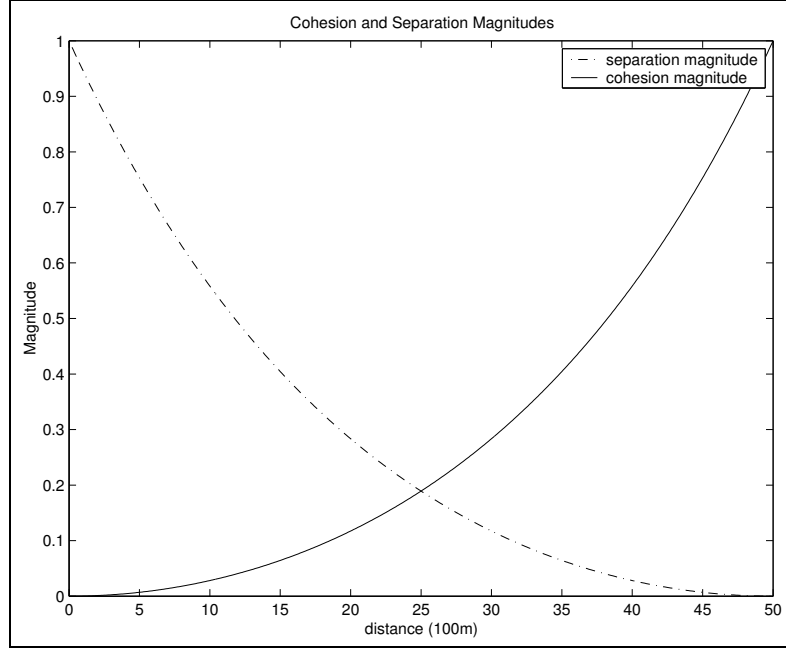


Figure 6 Cohesion and Separation values for distances from 0 to 5km

Equation 13 is chosen as the force model due to the exponential growth factor of the gaussian distribution. Since it is continuous for all values of $0 < d < \text{sr}$, it fits the requirement for $r(x)$ to be continuous. An alternative to this approach is given by Reif and Wang who suggest the use of the function:

$$r(x) = -\frac{c_1}{r^{\sigma_1}} + \frac{c_2}{r^{\sigma_2}} \quad (15)$$

where $-c_1/r^{\sigma_1}$ represents the attraction weight and c_2/r^{σ_2} represents the repulsion weight. This scheme is shown to work well in experiments performed by Reif and Wang [72] however, it suffers oscillation. Furthermore, the overall force goes to infinity as two robots approach each other. While this is desirable behavior in a simulation, such behavior in a constrained environment will rapidly overwhelm the actuator limits as well as all other rule vectors. This results in movement directly away from other agents at close range, but disregards any need to avoid a threat region or move toward a goal since the weight is so high at such close proximity.

In Reynolds' work [74], three rules are used: cohesion, separation, and alignment. Since there is no goal seeking rule in his system, these three rules cause the agents to move in an undirected coordinated fashion around the simulation area. When a goal seek rule is added however, two rules now exist that provide alignment among the agents. The goal seek rule causes agents to attempt to move towards a goal based upon the potential field lines at the agents' location. The alignment rule on the other hand, causes an agent to align with its neighbors. Since an agent and its neighbors line up on the potential field lines, this produces a strong force towards the goal. Early experiments using the alignment rule in conjunction with the goal rule suffered from over speeding, and the cohesion and separation vectors being overwhelmed by the alignment and goal seek behaviors. Due to this problem, the alignment rule is not used in the current model. Experiments performed during development of this algorithm have shown that the overall behavior of the swarm is much more desirable when the alignment rule is not used.

3.3.4 Peripheral Vision Weighting. Reynolds [73] proposes a means of weighting agent rules in his model by placing a higher weight on forces associated with neighbors in front of and to the side of an agent while reducing the weight on forces associated with neighbors behind the agent. Reynolds claims that this weighting is a more accurate depiction of how flocking birds tend to interact with their neighbors in the flock [73]. More

recently, Kadrovach [38] proposed a swarm model that uses a peripheral vision weighting scheme. This investigation utilizes Kadrovach’s weighting model.

The peripheral weight ω_{periph} is determined using the following equation:

$$\omega_{\text{periph}} = \cos^\beta\left(\frac{\theta_{a,n}}{2}\right) \quad (16)$$

where $\theta_{a,n}$ is the angle between $\vec{b}(a)$ and $\vec{b}(n)$. The value for β is chosen experimentally based upon the behavior of the model. Figure 7 shows the affects of different values for β . For this investigation, a value of $\beta = 0.1$ is used. This keeps neighbors located on the periphery from exerting too small of a separation force.

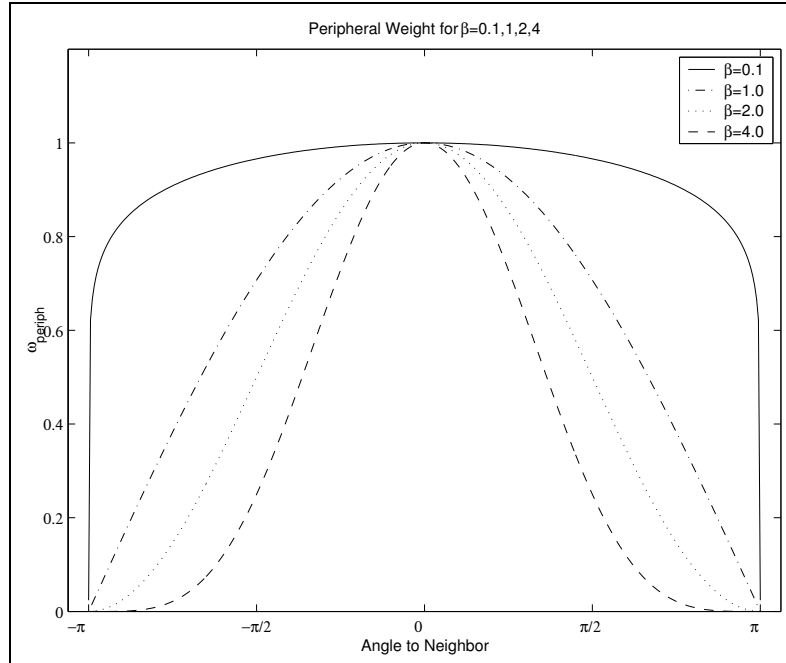


Figure 7 Values of ω_{periph} using $\beta = 0.1, 1.0, 2.0, 4.0$

3.3.5 Goal Seek/Threat Avoidance. The potential fields used to model threats in the agents’ landscape also rely on the gaussian distribution in equation 13. In this case, the mean, μ , is set to zero for all threats while standard deviation, rad is determined *a priori* based upon operator knowledge of the threat’s maximum effective range. Therefore, a threat is calculated as:

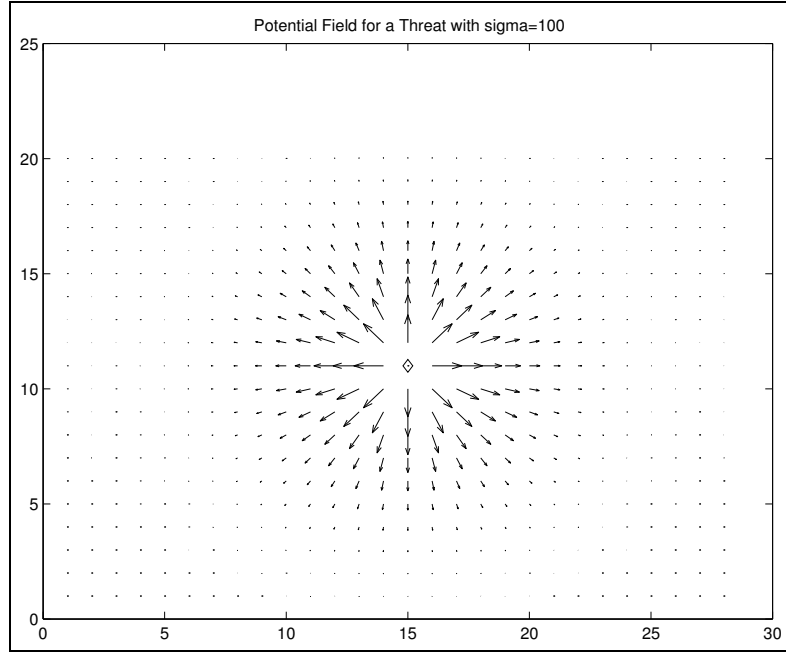


Figure 8 Potential field surrounding a threat located at (350,250) with rad = 100

$$\vec{t} = \exp\left(\frac{-d(\text{loc}(a), \text{loc}(t))^2}{\frac{2(\text{rad}/2)^2}{2\ln 2}}\right) \quad (17)$$

The function $d((x_1, y_1), (x_2, y_2))$ returns the euclidian distance between two points. The negative value returned by equation 17 is applied to an attraction vector. This causes the vector to be positive, pointing away from the center of the threat. Figure 8 shows the field surrounding a threat located at (350,250) with rad = 100.

Since the threat distribution is intended to represent the probability of detection of a fixed radar site, it is important to view the characteristics of a monostatic radar as compared to the gaussian distribution used in this investigation. The equation for a monostatic radar is given in [25] as:

$$P_r = \frac{G^2 P_t \text{rcs} \lambda^2}{64\pi^3 R^4} \quad (18)$$

The parameters for this equation are as follows:

- P_r : Power received (W)

- G : Antenna Gain (dB)
- P_t : Power transmitted (W)
- rcs : RCS of target (m^2)
- λ : wavelength (m)
- R : distance from target (m)

By eliminating the constants, it becomes clear that the R^4 term is the dependent term of this equation. The plot of this function is shown in Figure 9, and the gaussian threat model is shown in Figure 10.

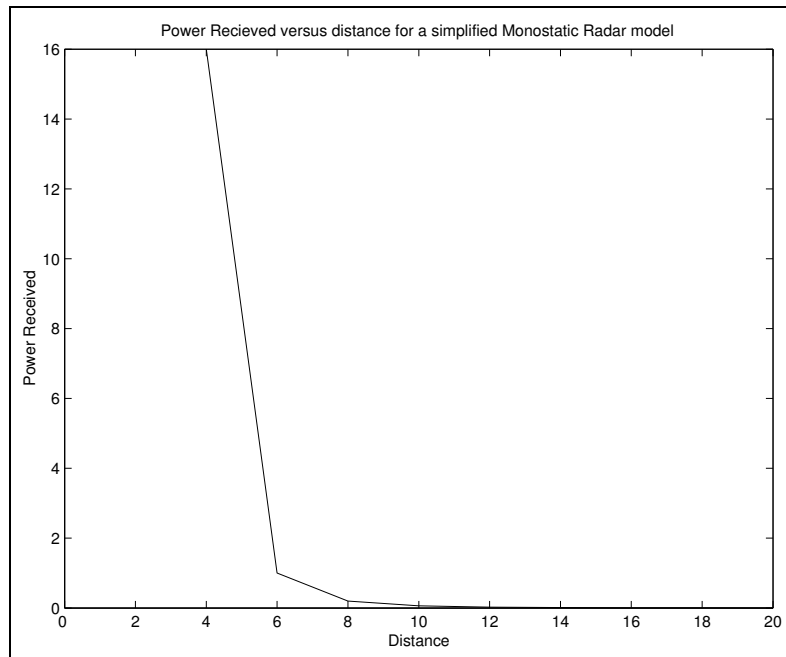


Figure 9 The power received versus distance of target for a monostatic radar. The distance values are not reflective of any existing radar due to the fact that the constant coefficients were removed from the equation in order to view the affect of the dependant variable on power received

Although the threat model is quite different from the radar model, there are several important reasons that this model is used. First of all, the argument is made in Section 3.3.3 that any value going to infinity as the distance approaches zero is not a desirable trait in a real-world system. As can be seen from equation 18, the function goes to zero as R approaches zero. Secondly, as can be seen in Figure 9, the signal gain remains very

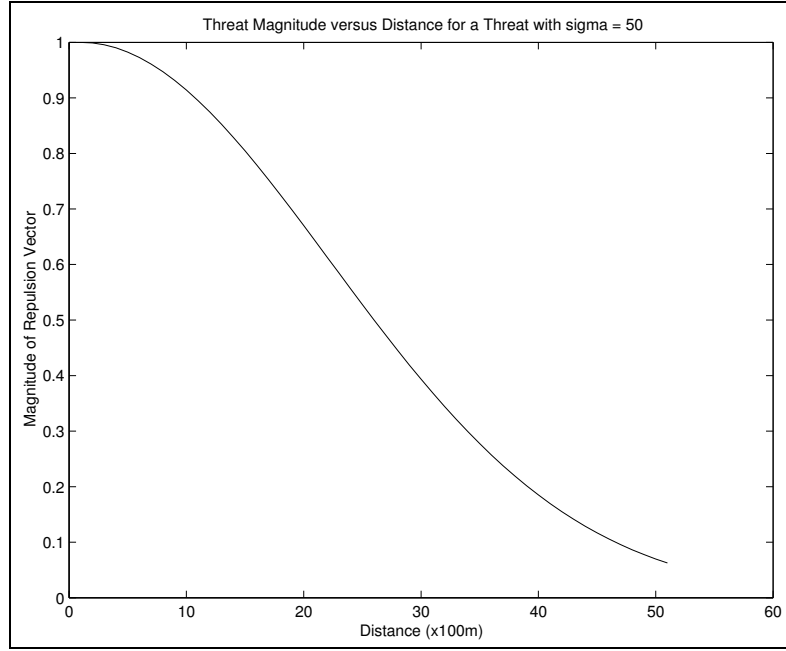


Figure 10 The magnitude of repulsion for a threat versus distance from the threat. The radius is set to $\text{rad} = 50$ for this example

low and then begins to increase rapidly. This rapid increase indicates a narrow region between the distance that a radar has a high probability of detection and a low probability of detection. This means that agents utilizing a realistic radar detection model will have little time to change course before moving from a low probability of detection distance to a high probability of detection distance. Due to these problems, a gaussian distribution with standard deviation greater than zero is used. The rad value can be changed by the operator prior to a mission and can be tailored to provide a desirable repulsion from high probability of detection areas associated with radar stations in the battle space.

The potential field of a goal is a vector field pointing towards the goal with a constant magnitude of ϱ , $0 < \varrho < 1$. This field is kept constant in order to guarantee that a critical attraction node exists at the center location of the goal. Figure 11 shows the attraction field around a goal with no threats in the landscape. The value of ϱ must be chosen based upon prior knowledge of the landscape that is being represented. Due to the force interactions, the attractive force of a goal could fully cancel out the repulsion force of a threat if $\varrho = 1$ as shown in Figure 12.

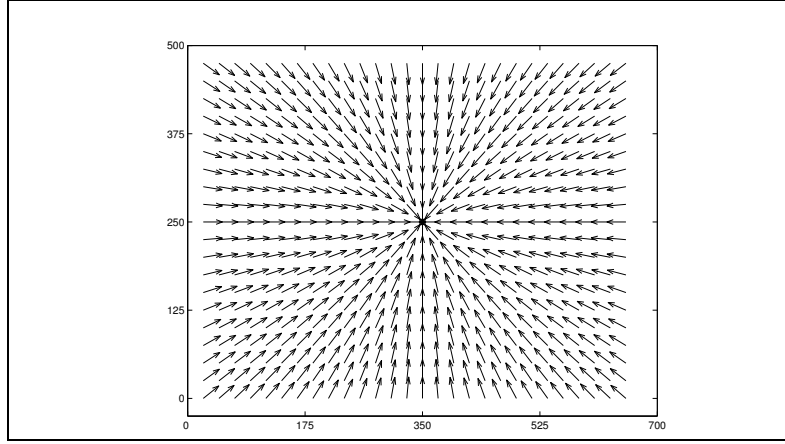


Figure 11 The potential field around a goal located at (350,250) with $\varrho = 0.75$

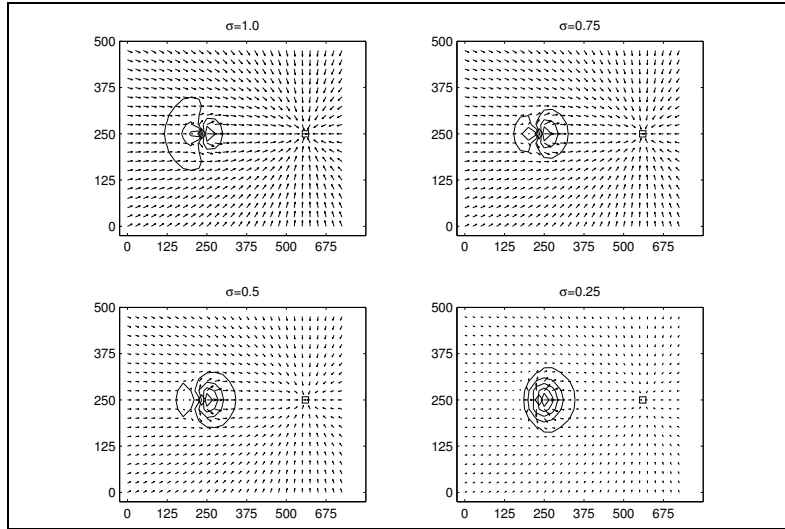


Figure 12 The potential field resulting from one threat with $\sigma = 100$ and one goal with $\varrho = 1.0, 0.75, 0.5, 0.25$. The location of the center of the threat is depicted by a diamond and the location of the center of the goal is indicated by a square.

Figure 12 shows the affects of different values of ϱ on the size of the repulsion field around a threat. For the remainder of this experiment, ϱ is set to 0.75. Figure 3.3.5 shows the resulting potential field formed by ten randomly located threats with $\sigma = 100$ and $\varrho = 0.75$.

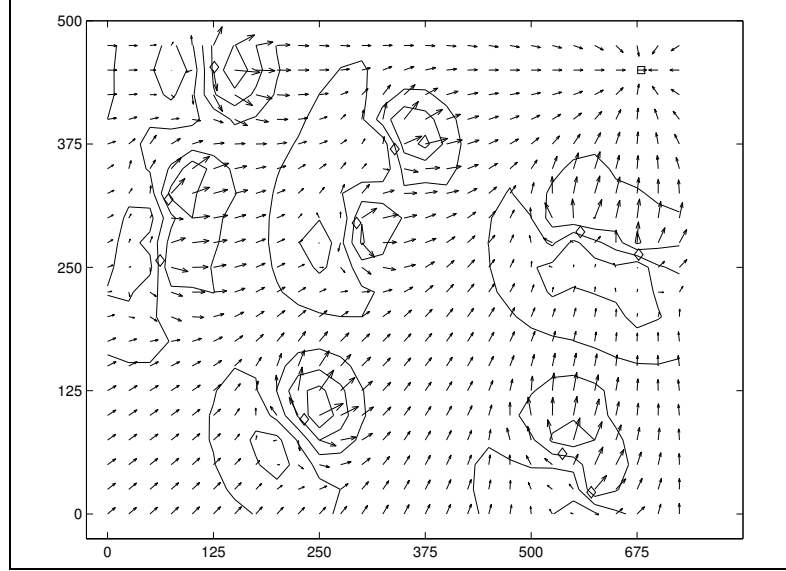


Figure 13 The potential field created by ten threats with radius of $\sigma = 100$ and located at (159,168), (638,439), (161,284), (539,372), (377,490), (110,280), (9,435), (161,445), (490,437), (245,196) and 1 goal with rad = 100, $\varrho = 0.5$ and located at (650,450). The goal is depicted by a square in the upper right corner, while the threats are depicted by diamonds

3.3.6 Waypoints. In order to reach a particular goal, an agent must follow a path through the search space. In this model, the path taken consists of a sequence of waypoints which are chosen *a priori*. In order to exert an attraction force on agents without adversely affecting the agent's swarming characteristics, it is necessary to represent a waypoint as a wavefront, rather than a finite point in space. This allows agents to approach the wavefront of a waypoint from any position without being attracted inward towards a finite point in space.

In order to allow agents to approach a waypoint without violating swarm behaviors, it is necessary to view a waypoint as a point in space with a direction. This direction, \vec{w}_i is the normal to a plane containing the waypoint, w_i . As agents approach a waypoint,

the distance and location of the waypoint are determined to be the closest point on the plane w_i^N occupied by w_i and with \vec{w} as the normal to the plane. The calculation for this point is performed by calculating the point p_w such that p_w lies in w_i^N and lies on a line perpendicular to the plane passing through $\text{loc}(a)$. This is depicted graphically in Figure 14.

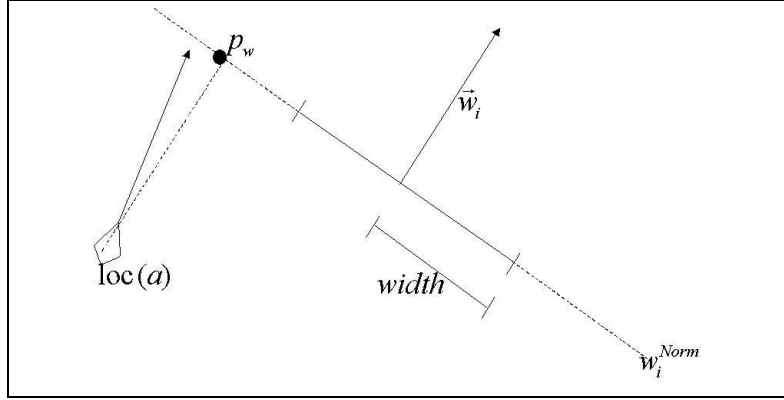


Figure 14 Graphical depiction of the calculation of the waypoint intersection point

If $d(p_w, \text{loc}(w_i)) > width$, p_w is set to be at the minimum distance from $\text{loc}(s)$ and $width$ distance from w_i while still lying on the plane w_i^N . The new location of p_w after performing this calculation is shown in Figure 15.

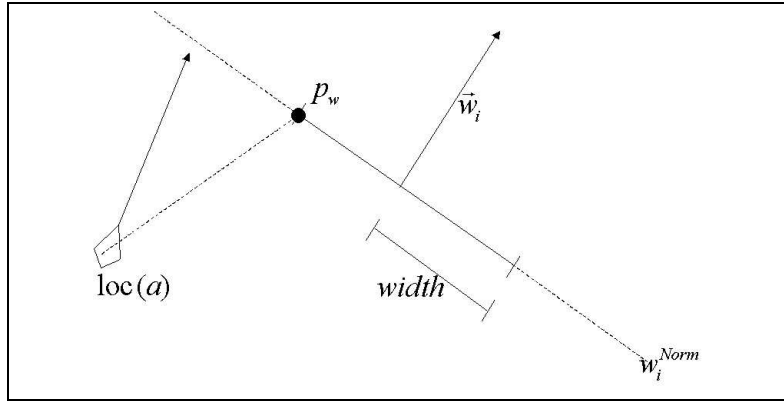


Figure 15 Graphical depiction of the corrected waypoint intersection point

While this representation works well, an infinitely long wavefront leads to a behavior in which agents separated by a threat during a simulation do not reform on the opposite side of the threat, but rather continue moving towards the next wavefront. This situation

is shown notionally in Figure 16. Due to this tendency, the length of the wavefront is set to be some value $\text{width} > 0$. Agents calculate the point of intercept on the plane occupied by w_i using equation 19.

$$\text{int}(w) = \begin{cases} x &= \frac{b_a - b_w}{m_w - m_a} \\ y &= m_w x + b_w \end{cases} \quad (19)$$

m_w is the slope of the normal plane, m_a is the slope of the line which is orthogonal to the normal plane, b_w is the point at which the normal plane intercepts the y axis, and b_a is the point at which the line from the agent to the normal plane intersects the y axis.

If $\text{int}(w)$ is more than σ distance from the center location of the waypoint, then $\text{int}(w)$ is recalculated such that $(\text{loc}(w) - \text{int}(w))^2 = \sigma$ and $\text{int}(w)$ is in the normal plane of \vec{w}_i . This causes agents to seek a corridor of attraction, rather than to create a uniform attraction field. Figure 17 depicts the potential field of a waypoint located in the center of the search space with a direction vector facing the upper right corner of the graph and $\sigma = 100$.

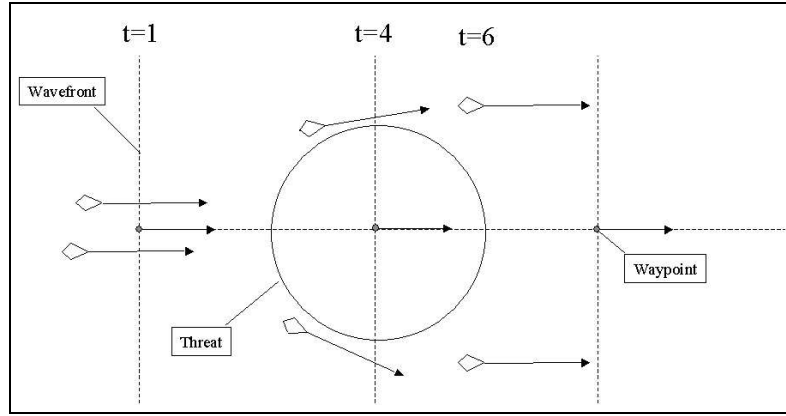


Figure 16 Notional example illustrating divergence behavior due to an infinitely long wavefront.

The magnitude of a waypoint field is calculated in the same manner as the magnitude of a goal field. Due to the need to create a critical region along the line representing the waypoint, the field is maintained at the uniform value of $\varrho = 0.75$. The reasons for this are the same as the reasons for using a uniform magnitude for the goal field.

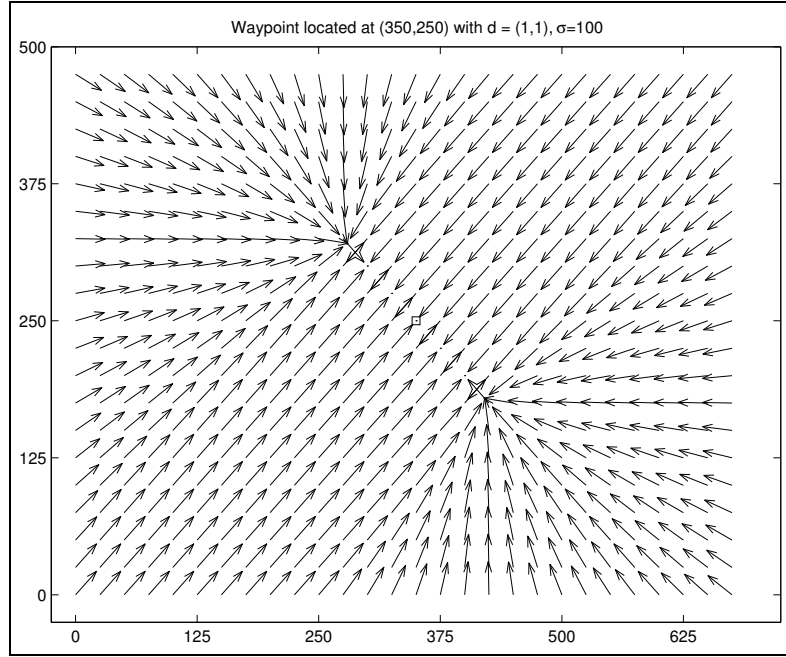


Figure 17 The potential field created by a waypoint with location (350,250), direction [1,1], $\varrho = 0.75$, and width = 100

Although multiple waypoints are used to specify a path through the landscape, as agents traverse the landscape, they only feel the force exerted by a single waypoint. An agent determines which waypoint is exerting a force by consulting a waypoint queue W . W is a FIFO queue populated *a priori* and stored in the local memory of each agent. An agent only seeks towards the waypoint at the head of the list. Once an agent has passed a waypoint, the agent pops that waypoint off the waypoint list, and seeks the next waypoint. An agent is said to have passed a waypoint if $(\text{loc}(w) - \text{loc}(a)) \cdot \vec{w}_i \geq 0$. When $|W| = 0$, the agent will then seek the goal g in the landscape. This behavior allows the agents to follow a specified path through the landscape, and then move to a predetermined rally point once the path has been fully traversed.

Waypoints also carry mode transition information. This affords the planner the ability to change modes by placing a waypoint at a specific transition point. Once an agent crosses the wavefront of a waypoint on its list, it determines the mode of the waypoint it has just passed. If the mode is different from the agent's current mode, then the agent transitions to the new mode. Next, the agent removes the waypoint from its waypoint list,

and chooses the next waypoint on the list. The agent then proceeds to the next waypoint using the mode matrix specified by the previous waypoint.

In the simulation environment developed for this research, an agent follows the list of waypoints until the queue is empty. It then seeks the goal situated in the landscape. There is only one goal in each landscape. Once an agent comes within a certain distance d_g of the goal, it signals that it has completed its mission and no further data is taken from that agent. For all experiments performed, d_g is set to 100.

3.3.7 Path Planning. In order to move the swarm through the battle-space, a path is generated *a priori* which passes over desired targets, as well as maintains desired safe distance from known threats. The actual characteristics of the path such as path length, smoothness, distance from threats, etc. are determined entirely *a priori* by the operator in order to meet desired mission requirements. This path is then input into the memory of each agent in S .

As discussed in Section 3.3.6, as each agent moves through the battle-space, it sets the current goal to be the next waypoint in its queue. The location of these waypoints is determined *a priori* by the planner, therefore the agents do not need to perform on-line path planning. This assumes that the path does not change dynamically. Dynamic path planning following is possible in this architecture by changing the location of waypoints in agents' waypoint list however, in order to maintain zero explicit communication, dynamic paths are not considered in this research.

3.3.8 Behavior Modes. The main thrust of this research is to develop a manner of controlling certain emerging behaviors by adjusting the weighting of the individual rules. The desired emergent behavior of the swarm is defined in this research as a behavior mode. Each mode uses a set of fitness functions to determine the current environment of the agent. The individual fitness functions are used to adapt the four swarm modes based upon individual knowledge of the problem domain. These functions are denoted as $f_0 \cdots f_3$. The number of functions used is dependant upon the number of environmental influences desired. For this research, the environmental influences used are:

- f_0 = rule weight
- f_1 = sensor coverage
- f_2 = sensor overlap
- f_3 = difference in velocity from neighbors

Each of these values is discussed in more detail in Section 3.4.3.

The five modes described in this section are: Reconnaissance, Scan, En-Route, Join and Hold. Each of these modes is described through the use of a behavior matrix δ_b . The construction of the behavior matrix is details in Section 3.4.4. These behavior modes are expressed by multiplying the functions $f_0 \cdots f_3$ by the matrix δ_b . The resulting 4×1 matrix contains the rule weight vectors discussed in Section 3.4.4. Each behavior mode is described in this section, and a mathematical expression for reconnaissance, scan, and en-route modes is determined experimentally in Chapter 5.

Of the five behavior modes, the join and hold modes require special characteristics and are intended to be used to initially form the swarm and to put the swarm into a holding pattern while reprogramming the mission profile, or attempting to perform human operator intervention on a single vehicle of the swarm. Since these two modes are considered to be house-keeping modes, they are only described in general terms. For this investigation, only the reconnaissance, scan, and en-route modes are tested.

The reconnaissance mode attempts to perform a low-fidelity scan of a large surface area. This scan is intended to perform a “broad stroke” approach to battlefield information gathering, and is similar in concept to the SAR strip mode [66] of the Air Force’s Global Hawk radar. A reconnaissance formation covers a large area with low sensor density in order to search for unknown threats, targets, or objectives. Assuming that a sensor fusion model is used, this means that the overall picture is broad, but not very high quality. This behavior mode provides enough information to determine details of a specific target however, for missions such as SCUD-hunting, and search and rescue this mode allows maximum sensor coverage in order to determine the general location of these desired targets.

The global emergent behaviors used to analyze the effectiveness of this mode are global sensor overlap, and contiguous area coverage. Global sensor overlap refers to the total amount of overlapping area of each agent's sensor footprint. As shown in Figure 18, it is possible for multiple agents to cover the same area over a period of time. In order to account for this behavior, each agent maintains a history of its positions $H : (p(t) \cdots p(t - 50))$. For global sensor overlap, each agent's position and history list are consulted and the total amount of overlap at time t is calculated. If any agent's sensor overlaps with a location where another agent's sensor has scanned within the last 50 timesteps, then the overlap area is included in the calculation.

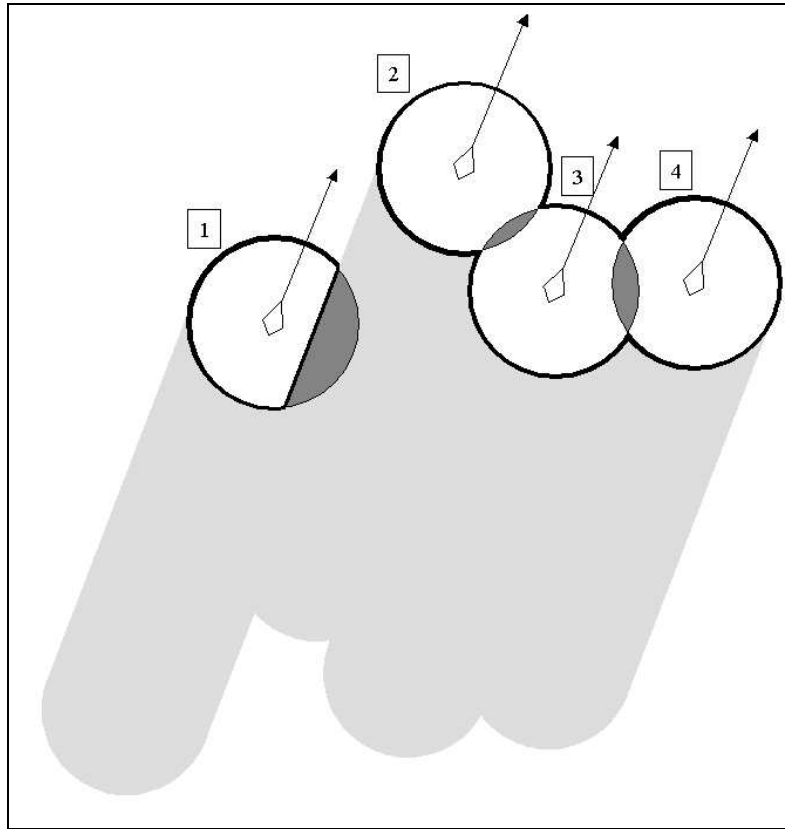


Figure 18 Sensor coverage of a swarm of agents at specific moment in time. The light-gray trails represent the area that has been covered in the past 50 timesteps. The dark-gray areas are overlapping areas calculated during the current timestep. The dark line encloses the total contiguous sensor area for this example. Note that while agent 1's footprint does not overlap with agent 2's current footprint, it does overlap with agent 2's historical coverage. In this situation, the area enclosed in a dark line around agent 1 is counted as part of the contiguous area.

Contiguous area coverage refers to the total amount of contiguous sensor coverage of the swarm. This corresponds to the area surrounded by a dark line in Figure 18. This measure represents the amount of area covered without any gaps occurring within the sensor footprint A . In the case where two large contiguous areas exist, the size $|A|$ of each area is considered and depending upon the relative size, the areas are either averaged, or the larger of the two is used for the metric. For example, if $|A_1| \approx |A_2|$, such as in Figure 19, the average of the two areas is taken. If the size of one of the areas differs by more than 10%, however, only the size of the larger area is counted. This behavior creates a non-continuous metric that attempts to measure how well the area flown over by the swarm is scanned. If a gap exists in the area scanned, some targets could be missed. With this in mind, contiguous area coverage penalizes a swarm that does not maintain contiguous coverage of an area of the landscape.

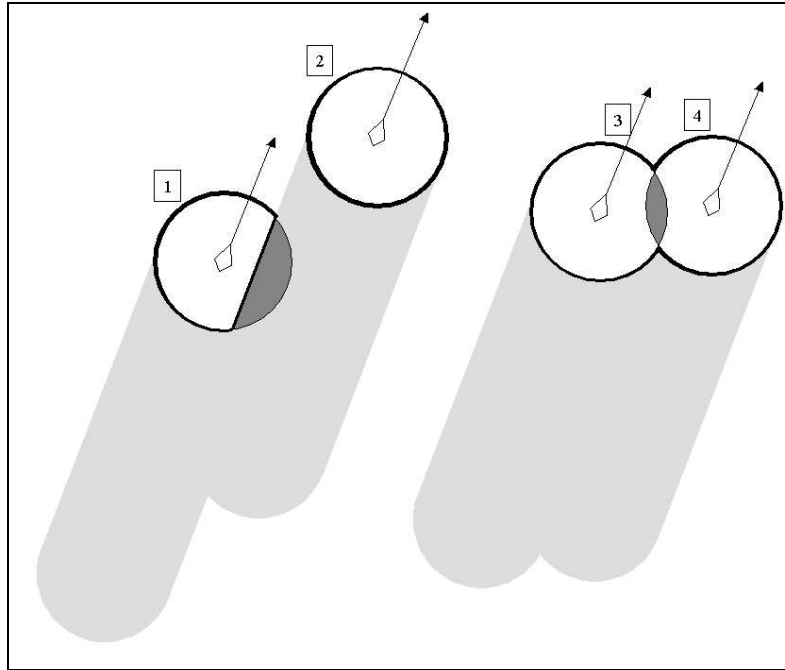


Figure 19 A snapshot in time of agents scanning an area. Two contiguous areas are depicted with a gap between the two. Each contiguous area is emphasized by a dark outline. Note that while agent 1's footprint does not overlap with agent 2's current footprint, it does overlap with agent 2's historical coverage. In this situation, the area enclosed in a dark line around agent 1 is counted as part of the contiguous area.

The scan mode performs a detailed sensor sweep along a given route. This mode performs a function analogous to the SAR spot mode [66] of the Global Hawk. Agents perform multi-angle, multi-view scans of the target location in order to provide a high-fidelity fused sensor image to the warfighter. This mode requires a very high entropy in the swarm formation, and involves a great deal of coordination and maneuvering in order to eliminate collision problems. The intent of this mode is to provide a large amount of sensor data from many different angles of a particular target or objective.

The global metrics used to measure the effectiveness of the scan mode are global sensor overlap, look angle variance, and minimum safe distance. The global sensor overlap metric is the same metric discussed previously however, rather than looking for a very small amount of overlap, a large amount of overlap is desired for the scan mode. The reason for this is that the scan mode is intended to cover a very small track along the landscape with a focused sensor coverage. The greater the overlap among sensor footprints, the greater the sensor focus is.

Look angle variance is a measure of how many different angles the desired track is being scanned from. Figure 20 depicts how each angle is measured. This metric determines the sample variance based upon a snapshot in time of look angles. The route that the planner intends to scan is laid out using waypoints. In order to ensure that agents stay within close proximity of the scan area, the waypoints must be set so that width is equal to the width of the area that must be scanned.

$$L_{s^2}(S) = \frac{\sum_{s \in S} (\overline{L(s, t)} - \overline{L(t)})^2}{t - 1} \quad (20)$$

where $\overline{L(s, t)}$ is the average look angle of the swarm S at time t , and $\overline{L(t)}$ is calculated as the average of all $\overline{L(s, t)}$ for the simulation.

The minimum safe distance metric is used in order to determine how close agents are getting to each other during a scan mission. This measure reports the minimum distance between two agents for each timestep within a slice of time such that

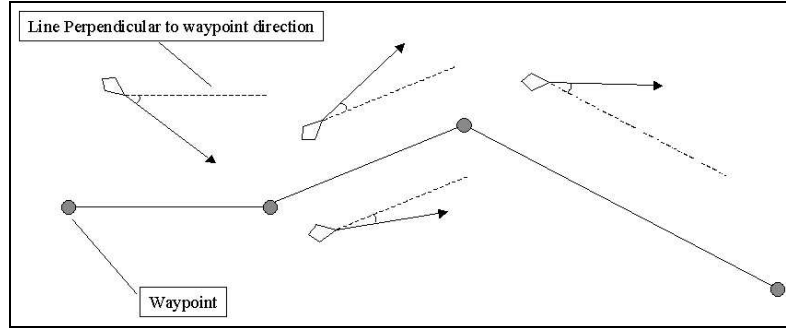


Figure 20 A notional example depicting a path and four agents as well as their look angles.

$$d_{\min} = \frac{\sum_{t_d=t-\Delta t}^t \text{mindist}(t)}{\Delta t} \quad (21)$$

Since the scan mode is intended to be highly entropic, the possibility of agents getting very close to each other during maneuvering is higher due to limits in the agents' actuators. In order to ensure that the agents are not maneuvering dangerously close to each other, this measure is used to view the average minimum distance during a scan.

The en-route mode establishes a direct flight mode where agents move from one waypoint to the next with a minimum of control inputs. This mode is characterized by a stable formation that avoids threats and maintains cohesiveness while moving in the most direct manner possible to the next waypoint. In this behavioral mode, agents maintain threat avoidance and goal seek characteristics.

Due to the basic nature of the en-route mode, it is used as our baseline for comparing to the reconnaissance and scan modes. In order to do this, all of the metrics discussed previously are applied to the en-route mode.

The Join mode is intended to be used only for formation building during the initial stage of the mission. This mode allows agents to intercept each other and form a swarm. If no agents are visible in an individual's neighborhood, then the agent does not enter into the join behavioral mode. Join can also be used during a mission to bring agents back into a swarm if they have just completed a high entropy mode such as scan and have lost sensor contact with all other agents. This helps to maintain swarm cohesiveness and

overall swarm communications however, it is more desirable for agents separated from the main swarm to continue moving towards the goal. By following the potential field lines of the waypoints, the likelihood of an agent rejoining the swarm is very high.

The Hold mode places a pre-defined racetrack route in the agent’s waypoint list. This pattern is followed until the agent is given a command from the human operator to transition to a different mode. The Join mode can be entered from the Hold mode, with all agents involved in the mode transitioning back to the hold mode until a transition is manually commanded by the human operator or the conditions for transitioning into the join mode are met. This mode is intended to aid in formation building at the beginning of a mission, and can also be used when the need arises for a swarm to remain relatively stationary over a target area while decisions are being made or data is being analyzed.

3.3.9 Mode Transition. Mode transitions are determined *a priori* by the mission planner, and are triggered by arrival at a specified waypoint. Since the mission path consists of multiple waypoints that the swarm follows in successive order, a check is made at each one to determine whether or not a mode transition is necessary. If the current waypoint’s specified mode is different from a previous one, then a mode transition occurs.

When an agent transitions modes, the behavior modification matrix for the particular mode is put into use immediately in equation 9. Since agents transition modes at different times, it takes a finite amount of time for the swarm to stabilize to the new emergent behavior. It is not known how transitions affect the overall swarm behavior, nor is the convergence time to desired behavior known. Some theoretical work has been done to establish convergence criteria of multiple agents [48] [27] however, no convergence theory currently exists that is capable of working with a coordination system as complex as the one used in this investigation.

Several modes, such as hold and join allow for transition triggers. This means that an agent can enter in the hold or the join mode when a particular environment input is encountered rather than when a particular waypoint is reached. These modes are intended to allow the operator to “build” a swarm by launching vehicles in a serial fashion, and

then allowing them to form into a swarm at a predetermined rally point. The details of this transition scheme are discussed in more detail in the following paragraphs.

From the Hold mode, an agent can enter into one of three different modes based upon operator input. These modes are En-route, Scan, and Reconnaissance. Furthermore, the Join mode can be entered from the Hold mode based upon the precondition of having encountered a new agent or group of agents, and having previously had a neighborhood $|N| = 0$. This transition is intended to allow individual agents to approach the holding position after launch, and then to join with other agents that are currently holding. This is how the swarm is initially formed after UAV launch. In order to avoid a situation where multiple agents are moving in the racetrack path but are not sensing each other, agents in the Hold mode randomly choose a direction to move in on the racetrack, and change directions at randomly chosen intervals. Provided that the random intervals are sufficiently long enough to allow agents to complete several laps of the racetrack, this results in convergence to a stable swarm formation. The racetrack is also designed so that parallel legs of the track are within sensor range of each other. This maximizes the chance that agents sense each other while in the Hold mode.

The Join mode is used to allow an agent to join with other agents already configured into a swarm formation. From this mode, agents can only reach the Hold mode. The Hold mode is reached when an agent has successfully joined the swarm formation. An agent is determined to have joined the formation when it has a neighborhood size $|N| > 0$. If the agent has reached the join location, but has not encountered any other agents, it enters the hold mode until it encounters other agents.

3.3.10 Communication. One of the goals of this research is to keep the amount of explicit communication between agents to a minimum. The reason for this is that as swarms increase in size, the communications between agents becomes the primary bottleneck. For purposes of this research, explicit communication between agents has been eliminated. In order to assume no explicit communication, it is vital to develop forms of implicit communication that provides the desired amount of functionality. These implicit communications are outlined in this section.

In order to form a neighborhood, an agent must be able to determine where its neighbors are. This can be accomplished either through a communications broadcast of an agent's position followed by responses from agents within the neighborhood range of that position, or it can be accomplished through sensors. The model used for this experiment assumes that each agent is capable of sensing other agents in the swarm. The exact nature of the sensor is not assumed however, the sensor must have the ability to accurately determine the distance and relative bearing of every agent within a finite range. For purposes of implementation simplicity, a sensor "shadow" cast by one agent does not preclude an agent in that shadow from being detected. While this means that the model used is not fully accurate to real-world circumstances, the results obtained by this model are still useful for understanding the adaptation of various swarm behaviors.

3.4 *Low Level Design*

3.4.1 Scaling and Coordinate Systems. A simulated representation of the real-world can work in many different types of numbering systems and scales. When designing a simulation environment, it is necessary to define a coordinate system for locating objects within the simulation, as well as a scale factor that allows the simulated calculations to be mapped to a real-world instantiation of the simulation. The coordinate system used in this simulation is a modified Cartesian coordinate system where the x coordinate value increases from the origin to the right of the simulation field, and the y coordinate value increases from the origin down. This coordinate system is used in order to map directly to a graphics environment that utilizes an origin in the upper left corner of the screen with y increasing in value in the downward direction. All locations in the simulation are specified using double precision floating point number representation.

The scale used in this simulation is based upon the pixel resolution of the monitor. For a monitor with low resolution, only a small portion of the overall landscape can be visualized without performing a transformation on the graphics rendering. One pixel on the screen represents a 100 m square in real space. This means that a threat with $\text{rad} = 100$ has a radius spanning 10000 m. Since double-precision values are used to represent coordinates, it is possible to represent locations to within fractions of a meter in the simulation.

Another aspect of scaling that must be defined is the timestep used. This value is the interval of time that elapses between each discrete event in the simulation. While it is possible for this interval to be dynamic, the current simulation implements a constant time interval. This interval represents a discrete amount of time that has passed since the last sequence of events occurred. For this research, the updating of an agent's position is the only event considered. At the beginning of each timestep, all agents calculate desired direction \vec{d} , and at the end of each timestep all agents update their position and direction vector \vec{d} . This allows each agent to make calculations based upon the current position of all neighbors rather than allowing some agents to perform calculations on the future location of their neighbors.

3.4.2 Flight Model. The flight model for each agent consists of a very simple inertial model that takes a steering force as its input. The direction of an agent is stored as a vector \vec{d} . When a steering force \vec{s} is applied to the agent, the new vector is calculated as:

$$\vec{a} = \frac{\vec{s}}{m_a} \quad (22)$$

$$\vec{d} = \vec{d} + \vec{a}\Delta t \quad (23)$$

The value of m_a is the mass of agent a . This calculation gives the acceleration of an object with a certain mass when the force \vec{s} is applied to that object. While this model does not accurately reflect the flight model, it is accurate for prediction of inertial motion on a large scale. Since one time step in our model is equivalent to one second of real-time, this simple model provides a reasonable level of reality without incurring large computational overhead.

The steering force \vec{s} is used to apply a force on the agent during a given timestep. This force is the equivalent of the force applied on an aircraft when the rudder is turned and the ailerons deflected in a certain manner. While the actual manner in which different

control surfaces interact is not modelled, the overall result of those interactions, specifically, the turning, acceleration, or deceleration forces are modelled by the steering force.

In order to create a more realistic model, it is necessary to limit the total amount of steering for the vehicle. This steering force depends upon the actual model that is being used. For example, an F-16 fighter can exert much more turning force than a C-141 cargo aircraft. The model proposed by Reynolds in [74] is used to accomplish this. This model applies maximum turn limits and maximum acceleration and deceleration limits to the steering force components such that $0 \leq \vec{s}_x \leq \theta_{\max}$, $a_{\min} \leq \vec{s}_y \leq a_{\max}$. Figure 21 shows how the steering limits affect overall vehicle characteristics.

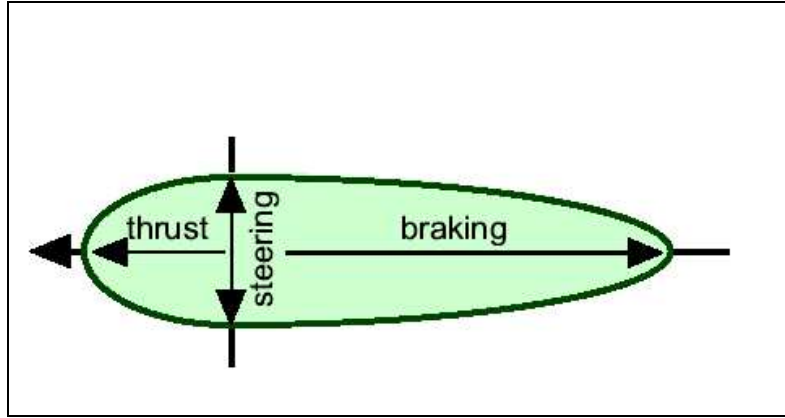


Figure 21 Reynolds's steering force envelope as restricted by thrust, braking, and steering [74]

In this investigation, the steering limits are defined as the engine thrust, and a turning actuator. In reality, the turning component of \vec{s} would be passed to an on-board control algorithm which would then determine the correct combination of control surface movements needed in order to achieve the desired turning force. Since this level of detail does not affect the overall results of this investigation, only the x component of \vec{s} is used to determine the rate of turn of the vehicle.

Actuator limits are calculated based upon the x and y component of \vec{s} with respect to the vehicle's local coordinate system. Both components of \vec{s} represent the acceleration required in order for the vehicle to be moving in the desired direction at time $t + 1$. This means that the thrust required to match the y component of \vec{s} is actually the thrust

required to accelerate or decelerate the vehicle to the desired speed within 1 timestep. The y component is mapped to thrust required using equations from [16]:

$$D = C_d \rho \frac{1}{2} v^2 s \quad (24)$$

$$T = v_y m + D \quad (25)$$

where C_d is the coefficient of drag of the vehicle, ρ is the air density, v is the current velocity of the vehicle, and s is the wing planform area. The thrust required is then truncated to the maximum available thrust of the craft T_{\max} , or to 0 if the required thrust is negative. The value of T represents the total amount of force exerted by the thrust of the engine on the vehicle, the actual acceleration experienced by the vehicle must be calculated using equation 29.

The force exerted by the turning actuator is calculated using the x component of \vec{v} . The value must be calculated as a triangle with points A , B , and C where $|AB|^2 + |BC|^2 = \vec{s}_x^2$. Furthermore, since acceleration is determined by \vec{s} 's y component, the turning force must not add any forward acceleration into the system. This means that $|AB| = |BC|$. Finally, the total turn must be limited such that $\angle BAC \leq \theta_{\max}$. This system of equations is given in 26 with a graphical depiction shown in Figure 22.

$$|AB|^2 + |BC|^2 = \vec{s}_x^2 \quad (26)$$

$$|AB| = |BC| \quad (27)$$

$$\angle BAC \leq \theta_{\max} \quad (28)$$

Once the force required to move in \vec{dd} direction at time $t + 1$ has been calculated, the acceleration in the vehicle's local x direction is calculated using equation 29.

$$\vec{a} = \frac{\vec{s}}{m} \quad (29)$$

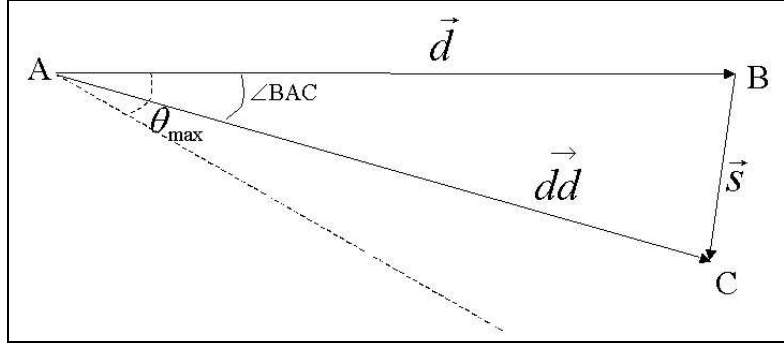


Figure 22 The amount of turning force required is calculated by representing the current direction, desired direction, and turning force as an isosceles triangle. The length of the two equal sides of the triangle is set to $|\vec{d}|$. In this manner, the steering force results in a turn with no forward or reverse acceleration.

Property	Value
C_d	0.09
ρ	0.652691 kg/m^3
S	1.858 m^2
m	1020.6 kg
T_{\max}	200.0 N
θ_{\max}	$20^\circ/\text{s}$
v_{\max}	77.16 m/s
v_s	20.577 m/s

Table 1 Values used for the physics model of the airframe

This equation is used to calculate both the x and the y components of the acceleration experienced by an agent. The value \vec{a} is what is finally returned from the vehicle's inner control loop.

The values used for C_d , ρ , and S as well as m , T_{\max} , θ_{\max} , v_{\max} , and v_s are given in Table 1, and were chosen in order to closely reflect the flight characteristics of a RQ-1 Predator UAV flying at 20,000 feet altitude [24]. The flight characteristics of the simulated UAV are compared to the flight characteristics of the RQ-1 Predator in Table 2.

Using the acceleration forces embodied in \vec{s} , the motion of the vehicle is calculated via a first order Euler equation with $\Delta t = 1$.

$$\vec{v}(t + \Delta t) = \vec{v}(t) + (\Delta t) \frac{d\vec{v}}{dt} \quad (30)$$

Characteristic	Simulated	Predator
Max Speed	117.67kts	117kts
Cruise Speed	88.25kts	70 kts
Weight	1020kg	1020kg

Table 2 A comparison of the flight performance of the simulation model and the flight performance of the RQ-1 Predator UAV [24]

It is possible to perform the simulation using a much smaller Δt however, this increases the runtime considerably. Since all agents in the simulation are subject to the same truncation error associated with using a first order approximation, a Δt value was determined to not interject enough error into the system to invalidate the data gathered from experiments.

3.4.3 Fitness Objectives. The coefficient matrix δ_b for each behavior mode $b \in B$ consists of four rows corresponding to the four rule vectors discussed previously, and four columns which correspond to four different functions $f_0 \dots f_3$ which provide a numerical value representing some aspect of the agent's current environment.

The first function, f_0 , is a weighting factor that determines how much weight the rule vector will have in the calculation of ω for each rule. The other functions are descriptive of a particular aspect of an agent's state at a given timestep. These are defined in the following paragraphs.

The function f_1 represents the sensor overlap of an agent and its neighbors. Sensor overlap calculates the total amount of overlap in sensor area of every neighbor in an agent's neighborhood. In order to work properly, this function must take into account the historical coverage as well as the current coverage. The reason for this is that, if a neighbor is behind agent a , then its sensor footprint covers the same area covered by agent a one or more time steps previously. This means that when neighbor n 's overlap is being calculated, it must take into account the historical coverage of agent a . Since the value of sensor coverage relates only to the overlap with agent a 's sensor footprint, overlaps of two neighbors' footprints are not considered in this calculation.

Sensor overlap is calculated as:

$$f_1(a) = \frac{\sum_{n \in N(a)} O(n, a)}{C(a) + \sum_{n \in N(a)} C(n)} \quad (31)$$

In the equation above, $C(a)$ is the sensor footprint coverage of agent a , and $O(n, a)$ returns the amount of overlap between neighbor n 's footprint and agent a 's footprint to include historical coverage.

The function f_2 measures the distance of neighbors from the normal plane of the primary agent. The state characteristic that this function measures is the velocity matching of an agent with its neighbors. A negative distance is assigned to those agents behind the normal plane, while a positive distance is assigned to those agents in front of the plane. Figure 23 shows a depiction of how the normal plane is represented and how the distances are determined.

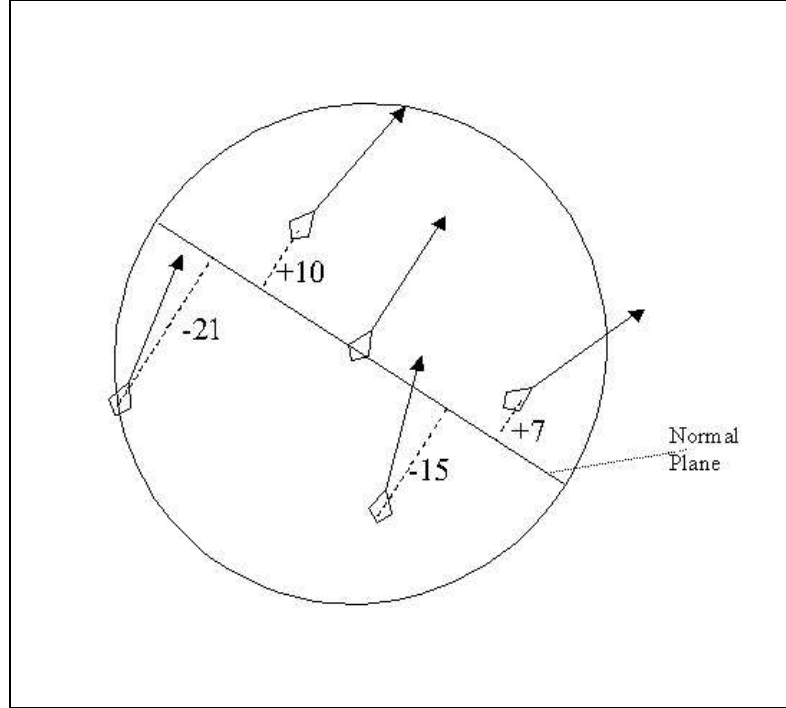


Figure 23 Method for determining distance for velocity matching fitness function

The distance from the normal plane is determined to be the distance from the point that intersects a line passing through a neighbor's position and orthogonal to the normal plane, to the position of the neighbor. The distances are used in the following equation:

$$f_2(a) = \frac{\sum_{n \in N(a)} d(n)}{|N(a)| s(a)} \quad (32)$$

where $N(a)$ returns the set of all neighbors to agent a , $d(n)$ returns the distance of neighbor n from the normal plane of a , and $s(a)$ is the sight range of agent a .

Approach angle variance, f_3 , measures the total variance of the approach angles in the swarm. The approach angle is defined relative to the target location, and is the bearing of an agent a from the target. Since the target location is known by each agent, this calculation can be made by each agent for itself, and each of its neighbors. This means that each agent has a slightly different value for this fitness function.

The variance of the different approach angles is the statistical variance normalized between 0 and 1. Using $\theta_{app}(a)$ as the approach angle of agent a , the equation is:

$$f_3(a) = \frac{(|N(a)| + 1)(\sum_{n \in N(a)} \theta_{app}(n)^2 + \theta_{app}(a)^2) - (\sum_{n \in N(a)} \theta_{app}(n) + \theta_{app}(a))^2}{(|N(a)| + 1)|N(a)|\mu_{max}} \quad (33)$$

The value of μ_{max} is determined to be 19.7392 which is the maximum variance possible from 0 to 2π . This value was determined using the equation for sample variance [60]

$$\mu_{max} = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2 \quad (34)$$

The maximum sample variance occurs when $n = 2$, $x_1 = 0$ and $x_2 = 2\pi$. This value normalizes the function.

3.4.4 Behavior Adaptation. In order to allow the four rule vectors to adapt to the agent's current environment, it is necessary to adjust the manner in which these four vectors are aggregated. This is accomplished through the use of a mode coefficient matrix δ_b for each behavioral mode $b \in B$ as well as the fitness functions $f_0 \cdots f_3$ discussed

$$\delta_b = \begin{bmatrix} \delta_{f_0,\vec{\sigma}} & \delta_{f_1,\vec{\sigma}} & \delta_{f_2,\vec{\sigma}} & \delta_{f_3,\vec{\sigma}} \\ \delta_{f_0,\vec{\eta}} & \delta_{f_1,\vec{\eta}} & \delta_{f_2,\vec{\eta}} & \delta_{f_3,\vec{\eta}} \\ \delta_{f_0,\vec{t}} & \delta_{f_1,\vec{t}} & \delta_{f_2,\vec{t}} & \delta_{f_3,\vec{t}} \\ \delta_{f_0,\vec{g}} & \delta_{f_1,\vec{g}} & \delta_{f_2,\vec{g}} & \delta_{f_3,\vec{g}} \end{bmatrix}$$

Figure 24 The structure of a behavior adaptation matrix

previously. Each row of this matrix corresponds to a specific rule vector, while each column corresponds to a specific fitness function. Figure 24 shows the structure of δ_b .

Equation 35 shows the method used to determine the coefficient for $\sigma(N_s)$

$$\omega_{\vec{\sigma}} = \frac{\sum_{n=0}^3 f_n \delta_{f_n,\vec{\sigma}}}{Z(\delta_b^{\vec{\sigma}})} \quad (35)$$

where $Z(x)$ is a function that returns the number of zeros in a vector, and $\delta_b^{\vec{\sigma}}$ represents the row of δ_b corresponding to $\vec{\sigma}$.

Using the results of equation 35, each vector is multiplied by its corresponding ω coefficient. The actual coefficients for δ_b are determined using an advanced search heuristic, discussed in Section 3.6.

3.5 Metrics

Due to the complex nature of emergent behaviors within a swarm, it is necessary to determine some measure of merit for various simulation runs. This allows for a more objective treatment of the given coefficient matrix, and reduces the need to make purely subjective assessments of the swarm's behavior. Since these metrics are only used to analyze the overall swarm behavior *a posteriori*, we utilize global knowledge for our analysis. It is important to note however, that these metrics cannot be used by agents during a simulation run because it would require global knowledge on the agent's part, and this reliance on global knowledge is undesirable in this investigation. The global metrics used to analyze the overall swarm behavior are center of mass, total accumulated threat, total overlap, total coverage, velocity variance, arrival time variance and alignment variance.

The center of mass metric treats each agent as a point with mass, and the global swarm as a single entity. The metric then calculates where the center of gravity of that object would be if all its mass were evenly distributed at each of the agent's locations. This metric allows us to see how the swarm as a whole progresses across the landscape. For all the designated modes, the center of mass of the swarm should remain close to constant (i.e. moving in the average direction of all the agents, at the average speed of all the agents), without making sudden changes at mode transition points. The ideal behavior would be for the center of mass to follow approximately the path marked out by the waypoints.

The equation for center of mass is given as follows:

$$C_m(t) = \frac{\sum_{i=0}^{|S|} p(i, t)}{|S|} \quad (36)$$

Total accumulated threat TAT is used to determine how much risk the global swarm assumes in accomplishing a particular task. This TAT provides a measure of the tradeoff made between accomplishing the desired behavior, and avoiding threats. A high TAT value denotes that the swarm incurred deeply into a high threat area, while a low value signifies that the swarm maintained a good distance from all threats. The equation for this metric is:

$$T_a(S) = \sum_{t=0}^n \sum_{a \in S} (T(a, t) - T(a, t-1)) \quad (37)$$

Total overlap and total coverage are complementary equations measuring the amount of sensor overlap in the swarm, and the amount of sensor coverage provided by the swarm as a whole. These values highlight the sensor coverage aspect of the swarm, and are most applicable when analyzing reconnaissance and scan modes. For example, in a reconnaissance mode, a high amount of sensor coverage is desired with a small, positive amount of overlap. In the scan mode however, a very high amount of overlap is required and the sensor coverage necessarily decreases.

Total overlap is calculated to be:

$$O_{\text{tot}}(S) = \sum_{t=0}^n O(S, t) \quad (38)$$

and total sensor coverage is calculated as:

$$C_{\text{tot}}(S) = \sum_{t=0}^n C(S, t) \quad (39)$$

Where $O(s, t)$ returns the total amount of overlapping sensor area in the swarm at time t , and $C(S, t)$ returns the total amount of coverage, including overlapping areas, at time t . Each of these functions only return the area in terms of what is overlapped or covered. In other words, overlapped areas are only counted once rather than each agent possibly reporting the same overlapping area.

Velocity variance, arrival time variance and alignment variance all provide information concerning the collaborative nature of the swarm. A high variance in velocity or alignment denotes very little uniformity among the swarm. Depending upon the desired mode, high variance is a desirable result. For reconnaissance however, a very low amount of variance would be desirable in order to maintain a long, spread-out line which would be the ideal formation for a large-area scan. Arrival time variance is used to discern the effectiveness of a particular mode in maintaining proximity as a swarm. Small values for arrival time variance indicate that all of the agents arrived at the goal within a relatively small period of time, while large variances indicate that the swarm has become “strung out” or has formed into one or more sub-swarms which took longer to move to the goal. Either of these behaviors, while not necessarily bad, indicate that the particular coefficient matrix for that mode should be analyzed in more detail to determine whether or not the desired behavior is being achieved.

Velocity variance is calculated as:

$$V_S = \frac{1}{t-1} \sum_{a \in S} (|\vec{v}_a|^t - |\vec{v}|)^2 \quad (40)$$

Alignment variance is:

$$A_S = \frac{1}{t-1} \sum_{a \in S} (\bar{\theta}_a^t - \bar{\theta})^2 \quad (41)$$

Arrival time variance is calculated to be:

$$T_S(t) = \frac{1}{|S|-1} \sum_{a \in S} (ETA_a^t - \overline{ETA})^2 \quad (42)$$

where $|\vec{v}_a|^t$, $\bar{\theta}_a^t$, and \overline{ETA}^t denote the average velocity, direction angle, and ETA respectively for a swarm S at a given time t , and $|\vec{v}|$, and $\bar{\theta}$ are the sample mean for $\forall t$, $|\vec{v}_a|^t$ and $\forall t, \bar{\theta}_a^t$ respectively [60].

3.6 Evolution Strategy

In order to determine a good coefficient matrix initialization, some form of search heuristic is required in order to reduce the total number of combinations tested. Normally, a search heuristic is based upon problem domain knowledge in order to lead the search towards a good solution [59]. For this problem domain however, very little is known concerning the relationships between different coefficients in the matrix, and how they affect each other. Due to this limitation, it is necessary to use a search algorithm that does not require a great deal of problem domain specific information in order to find a “good” solution. With these criteria in mind, an Evolution Strategy (ES) algorithm was chosen as the most desirable search algorithm based upon its ability to search through real-valued search spaces, as well as to modify its search parameters during the search [7]. This on-line search strategy modification is referred to as “self-adaptation” [7].

An ES algorithm is a subclass of Evolutionary Algorithms (EAs) [5]. This algorithm is capable of utilizing the mechanisms of biological reproduction in order to direct a search through a given search space. ESs have been shown to be successful on many different real-valued optimization problems including decomposing spectral data for analysis [70], designing the shape of a microchannel [61], and designing parameters for the control of jet flow [42]. The ES was originally developed to work with real-valued problem domains,

and has been shown to converge to good, although not necessarily optimal, results given enough computation time [7].

3.6.1 Selection Operators. The traditional ES utilizes one of two different selection operators, (μ, λ) and $(\mu + \lambda)$ [7]. In a (μ, λ) selection scheme, λ offspring are generated through a recombination operator (discussed in Section 3.6.3). From this λ offspring, the μ best individuals are chosen. Accordingly, in order to avoid creating a random-walk behavior, it is necessary for λ to be much greater than μ [7]. The (μ, λ) selection operator completely disposes of the parent population at each generation. This behavior allows the algorithm to escape from local minimums early in the search, and also has the added benefit of allowing the search to be conducted on dynamic as well as static fitness functions [7]

The $(\mu + \lambda)$ selection operator combines the parent and offspring populations into one population, and then chooses the best μ individuals in the combined population. This operator affords a monotonic improvement in the solution quality however, since good solutions have a tendency to remain in the population, it is possible for poorly adapted search strategies with relatively good fitness evaluations to remain in the population for a large portion of the search [7]. This results in slower convergence, and may sometimes lead to the algorithm becoming “stuck” in a local minima.

According to experiments performed by Schwefel [76] and later reported by Bäck [7], the convergence rate of the ES algorithm relies upon the ratio of size for μ/λ [7]. For a (μ, λ) selection operator, a ratio of 1/7 is recommended where $\mu \gg 1$. Bäck recommends using a value of $\mu = 15$ for most problem domains. While this ratio has been shown to work well in a general sense, no theory currently exists that allows for the derivation of the proper ratio for a given problem domain type.

3.6.2 Mutation Operators. The self-adaptation aspect of the ES is derived from the use of normally distributed mutations with an expectation value of zero and standard deviation of σ [4]. This operator works by mutating the object variable vector using a logarithmic function. It is possible to further adapt the mutation distribution through

the use of variances and rotation angles. Variances are related to object variables such that, for n object variables and n_σ variances $n_\sigma = n$, each object variable n_i is mutated using a normal distribution characterized by σ_i . Mutations can take several different forms based upon the values of n_σ . Mutations can occur uniformly across two dimensions ($n_\sigma = 1$), separately across two dimensions ($n_\sigma = 2$), and using correlated mutations, the two dimensional mutation can be rotated in two dimensions ($n_\sigma = 2, n_\alpha = 1$) [4]. Figure 25 shows how different values of n_σ and n_α affect the distribution in two dimensions.

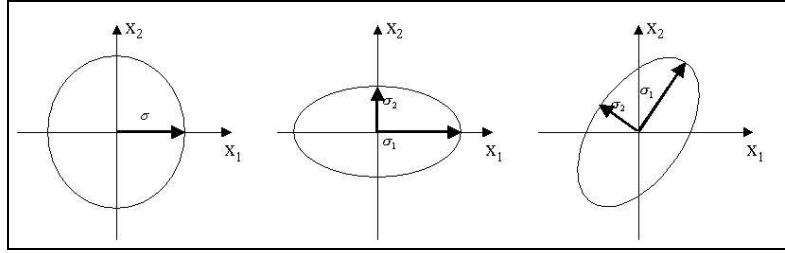


Figure 25 Affect of rotation angles on the shape of the probability density for $n_\sigma = 1$ on the left, $n_\sigma = 2$ in the middle, and correlated mutations $n_\sigma = 2, n_\alpha = 1$ on the right [4]

3.6.3 Recombination. Recombination in ES algorithms can take one of many different forms, however, the standard forms of recombination associated with ES algorithms are discrete, intermediate, global discrete, and global intermediate [6]. Recombination can be either sexual or panmictic, that is, recombination can either utilize only two parents from the population (sexual), or they can utilize multiple parents from the population (panmictic) [7]. For discrete recombination, for each position in the vector, one of the two parents is randomly chosen as the donor (for sexual recombination - panmictic randomly chooses a parent from the population). Intermediate recombination determines the arithmetic mean of the two parent values. The panmictic variant chooses a single parent from the population, and chooses the second parent at random for each position in the child vector [7]. Recombination can occur for only the object variables, only the variance vector, only the rotation angles, or any combination. Furthermore, different recombination techniques may be used for each vector. Currently, no theory exists to determine the best recombination operators to use and on which vector to use it [6].

3.7 Low Level Design Implementation

The algorithm detailed in Chapter 3 is implemented in the Java® programming language. This language was chosen for its included graphics libraries as well as ease of development. The object oriented nature of the Java programming language provide the ability to encapsulate sensor characteristics, aircraft characteristics, and rules in different classes, providing the ability to easily modify aspects of the algorithm through the use of inheritance.

For purposes of precision, all floating point numbers are implemented as IEEE-compliant double precision numbers. This allows for minimum error due to truncation, as well as higher precision calculations. A disadvantage to this design decision is the added computation overhead associated with performing math on double precision numbers.

In order to analyze the various aspects of the swarm algorithm, a visualization Graphical User Interface (GUI) was developed that provides real-time animation of agents' position and heading, threat, goal, and waypoint locations, agent neighborhood, sensor footprint, and agent sight range. This GUI is used in Chapters 4 and 5 to analyze the emergent behaviors observed for a given behavior matrix.

Due to the computation requirements of the swarm algorithm, it is necessary to use distributed computation techniques for the evaluation of behavior matrices produced by the ES. This is accomplished through the use of a Master-Slave decomposition strategy in which chromosomes are assigned to a given processor. Each processor runs the swarm simulation using the given behavior matrix encoded in the chromosome. Communication is accomplished through the use of Java's Remote Method Invocation (RMI) framework.

3.8 Summary

This chapter has discussed the design aspects of the swarm model in terms of intra-swarm interactions, and inter-swarm interactions. The potential field model used for this research has been developed and the effects of the APF parameters have been depicted graphically. A framework for behavior adaptation defined in terms of a behavior matrix is applied to the basic rules that make up the swarm interaction model. Furthermore,

the underlying physical model of the UAVs used is defined, and a brief comparison of capabilities is made to a Predator UAV. Also, the major aspects of Evolution Strategies are discussed. Chapter four provides the design of experiments, and discusses briefly the tuning of many of the ES parameters in order to obtain good results.

4. Design and Analysis of Experiments

The experimental aspect of this thesis effort involves developing behavior matrices that result in desired emergent behavior when applied to the swarm model discussed in Chapter 3. In order to obtain good results from the ES algorithm, it is necessary to “tune” the parameters for best performance. Since the evaluation time of the swarm model is large, only a small set of tuning experiments is conducted.

4.1 Overview of Experiments

In order to determine the ability of the distributed swarm algorithm discussed in Chapter 3, experiments are conducted to measure the actual emergent behavior obtained by modifying the behavioral coefficients. Due to the subjective nature of evaluating the desired behaviors, results are analyzed visually via an animation based on the simulation data. Numerical analysis is also performed by comparing the metrics (discussed in 3.5) of the behavior to metrics obtained from the baseline behavior defined in Section 4.3. Where stochastic algorithms are used, experiments are repeated in order to perform statistical analysis as discussed in Section 4.6.1.

The experiments are designed around three test maps: Saddle, Obstacle, and Overlap. These three maps are discussed in Section 4.4.1. In order to determine the robustness of the swarm algorithm, tests are also performed in a general map that contains features of all three of the test maps. Tests are also performed to test the scalability of the algorithm. These tests are conducted on the general map.

Since an Evolutionary Algorithm is used, experiments are also performed to determine good parameters for the development of the behavior matrix. Results of these experiments are analyzed and parameters are chosen for subsequent experiments.

4.2 Experimental Methodology

Experiments are conducted in multiple stages in order to create “good” coefficient matrices and evaluate their effect on emergent behaviors. The term “good” is used in this

instance to denote a coefficient matrix that provides the desired emergent behavior; a good coefficient matrix is not necessarily the optimal coefficient matrix.

4.2.1 Evolutionary Strategy Optimization. The first part of the experiment develops values for the coefficients of δ_b . Since there is no theoretical model describing the relationships between values in δ_b , it is not currently possible to determine good values *a priori*. Furthermore, the large size of δ_b coupled with a large range of possible values for each coefficient, renders an enumeration-type search impractical. With these issues in mind, an ES was chosen to evolve matrix values.

The Java Distributed Evolutionary Algorithm Library (JDEAL) is used to perform the necessary coefficient search [22]. This package is chosen due to its wide variety of evolutionary operators, as well as the ease of integration with the simulation developed for this research. Furthermore this package’s ability to perform distributed evolution using a master-slave paradigm is useful for reducing the total wall-clock time of the search. Due to the relatively long evaluation time of approximately 22s per evaluation, it is necessary to utilize the performance ability of a parallel computing paradigm in order to decrease total evolution time.

The parameters of an ES can adversely affect the performance of the search if not chosen correctly. To avoid this problem, the first step of experimentation is to tune the parameters used by the ES for the best attainable results. There are several parameters however, which have been shown to be good values for a large set of problem domains. These values are: $\mu = 15$, $\lambda = 100$. The recombination operator used is discrete crossover for the object variables and panmictic intermediate recombination for the strategy vectors. These values and operators were chosen according to Bäck [7], based upon investigations performed by Schwefel [77]. They are chosen based upon their general applicability and their ability to provide an even compromise between exploration and exploitation of the search space [6]. The parameters that are tuned through experimentation are $\sigma_i(0)$, $\iota(P(t))$, and the selection operation.

While most of the parameters used are based upon values recommended by Bäck [7], the $\sigma_i(0)$ value must be determined experimentally. This is due to the nature of the

problem domain being searched. Since each coefficient of $\delta_b \in [0, 1]$, a variance value of 3.0 causes the algorithm to take maximum steps in setting values. The variances would then be reduced through self-adaptation. This behavior could cause the algorithm to take longer to converge. While this is desirable from a quality of solution standpoint, slow convergence in this problem is too costly in computation time. Therefore the value of $\sigma_i(0)$ is tuned for exploitation rather than exploration. This value is chosen to initially allow mutation to occur across the entire range of possible values of δ_b .

Experiments are performed using $\sigma_i(0) = \{0.25, 0.5, 0.75, 1.0\}$ in order to determine the best value of $\sigma_i(0)$. The results are shown in Figure 26 with statistical results given in Table 3. Since all of the values for $\sigma_i(0)$ are statistically equivalent, any value of $\sigma_i(0)$ between 0.25 and 3.0 can be used. For this research, $\sigma_i(0) = 1.0$ is used for all subsequent experiments.

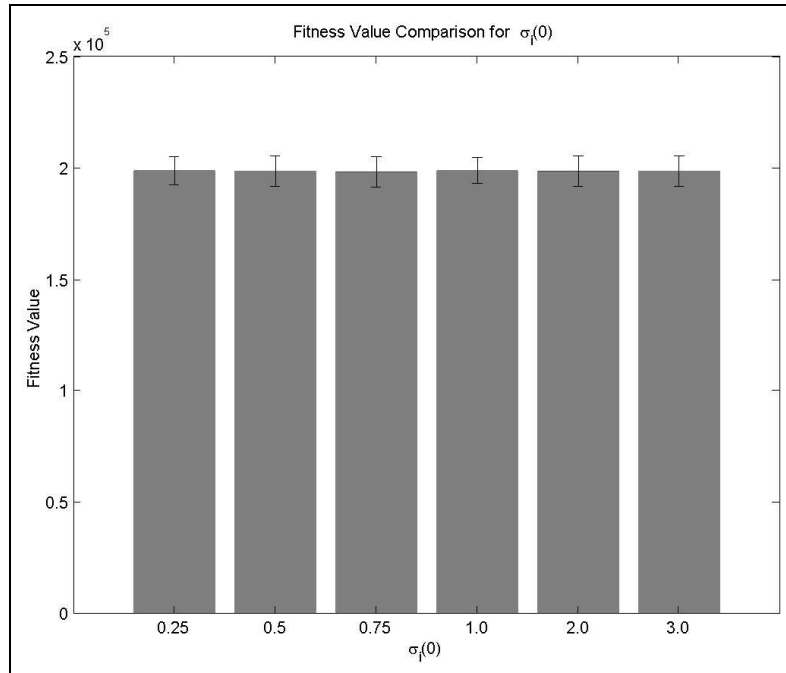


Figure 26 Comparison of convergence for different values of $\sigma_i(0)$

The results shown in Figure 26 imply that the value of $\sigma_i(0)$ does not have a great deal of influence on the convergence of the search algorithm. This means that the search space for this particular problem domain is relatively smooth and possibly unimodal. Due to this apparent smoothness, it is possible that a gradient-based search algorithm could perform

σ value	Min(s)	\bar{s}	Max(s)	s
0.25	167435.8696	199677.1712	209728.6294	6713.964459
0.5	173010.1842	198667.8616	208519.5327	6362.83784
0.75	176710.6836	198111.1415	207132.0101	6262.409196
1.0	184153.4665	197156.8133	205621.2470	5506.8813
2.0	171444.5332	198447.9914	208106.1087	6934.199872
3.0	171199.9448	198354.1836	208555.6536	6729.685135

Table 3 Sample Minimum, Mean, Max, and Standard Deviation for results obtained using values of $\sigma_i(0) = \{0.25, 0.5, 0.75, 1.0\}$

better than an ES-based algorithm. For future research in this area, it is recommended that a gradient-based search algorithm be implemented as opposed to an ES algorithm.

Another aspect of the ES operation that is considered is the use of rotation angles (see Section 3.6). Since very little is known about the shape of the fitness landscape, it is desirable to conduct tests to determine the effectiveness of the algorithm with and without rotation angles. Based upon the results in Figure 27 however, the use of rotation angles is not anticipated to affect the convergence or effectiveness of the algorithm. This is due to the fact that rotation angles on a smooth search space orient in the direction of the slope. This affect is analogous to using $n_\sigma = 1$ with sufficiently large values of σ . For this problem domain, a value of $\sigma_i(0) = 1.0$ is the largest possible value, and can be retained by the algorithm if it causes mutations to improve in a hill-climbing manner. If rotation angles are used, it is possible for the algorithm to more easily converge to a local minimum and require added generations to sufficiently evolve the strategy vector in order to escape that minimum.

Based upon the experiments performed in this section, the final values used for the ES algorithm are $n_\sigma = 1$, $\sigma_1(0) = 1.0$, $\iota(P(t)) = 100$, and $(\mu + \lambda)$ selection. The ES is run five times for each behavior on each of the test landscapes (discussed in Section 4.4.1). Since the average running time of the ES for a 100 generation run is approximately 3 hours on a high performance computing platform, it is not possible to perform a large number of runs.

Table 4 summarizes the tuning experiments performed and gives the sample mean, (\bar{s}), of each experiment.

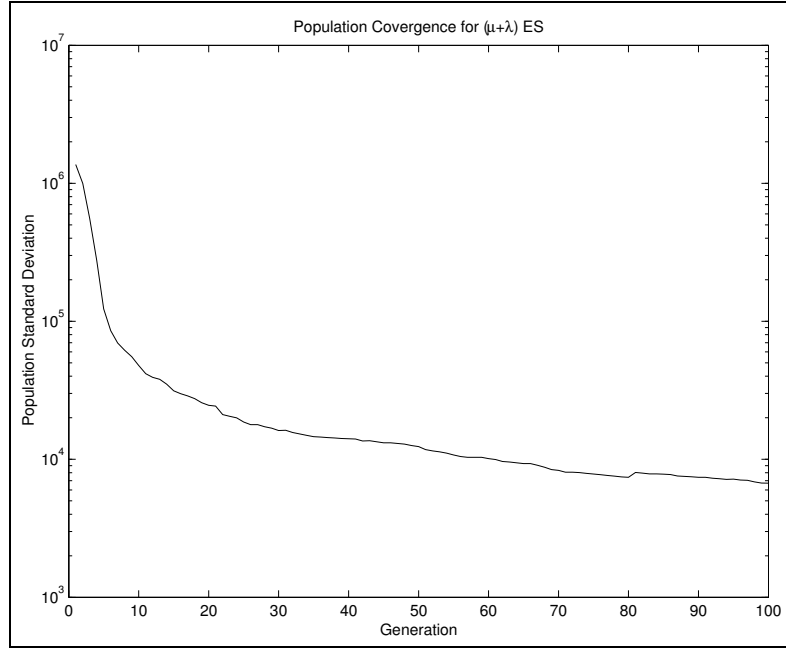


Figure 27 Convergence rate for $\sigma_i(0) = 1.0$

# of runs	$\sigma_i(0)$	$\iota(P(t))$	Selection Operator	\bar{s}
5	0.25	100	$(\mu + \lambda)$	199677.1712
5	0.5	100	$(\mu + \lambda)$	198667.8616
5	0.75	100	$(\mu + \lambda)$	198111.1415
5	1.0	100	$(\mu + \lambda)$	197156.8133

Table 4 Summary of experiments performed for Evolution Strategy algorithm parameter tuning. Average run time using 25 procesors is ~ 1200 minutes.

4.3 Baseline

Analysis of the metrics from Section 3.5 is performed by comparing results generated by a particular behavior matrix to results generated by a baseline matrix. Since no standard baseline has been developed for this problem domain, a pedagogical example is created for the purpose of comparison. The baseline matrix assumes that no weight is applied to any of the rule vectors except for the function that specifically applies to that rule. For example, the equation for ω_σ in Section 35 becomes

$$\omega_\sigma = f_0 \delta_{\sigma, f_0} \quad (43)$$

The values for $\delta_{\vec{\sigma},f_0}$, $\delta_{\vec{\eta},f_0}$, $\delta_{\vec{g},f_0}$, $\delta_{\vec{t},f_0}$ are set to 1.0, and all other δ values are set to 0.0. This is the equivalent of a standard swarm model without weighting coefficients. The baseline matrix is shown in Table 5.

	f_0	f_1	f_2	f_3
$\vec{\sigma}$	1.0	0.0	0.0	0.0
$\vec{\eta}$	1.0	0.0	0.0	0.0
\vec{g}	1.0	0.0	0.0	0.0
\vec{t}	1.0	0.0	0.0	0.0

Table 5 Baseline Coefficient Matrix

4.4 Validation of Coefficient Matrices

Upon obtaining a set of values for each behavior matrix $\delta_b, b \in B$, a real-time simulation is conducted using each evolved matrix. The real-time simulation is run in order to generate data concerning the global behaviors of the swarm for each of the behavior matrices generated by the ES (see Appendix A for results produced by the ES). These characteristics are analyzed both visually, using an animation of the simulation, and numerically using the set of global metrics discussed in Section 3.5. All five behavior matrices generated by the ES are used to gather metrics, and the final metric values are averaged across the five runs using the methods discussed in Section 4.6.1. This allows for a more balanced analysis of the results obtained.

Since the metrics designed do not completely describe the swarm behavior in an easily understandable manner, visual assessment of the behavior coefficients is necessary. Furthermore, since the mechanisms of emergent behavior are not yet fully understood, it is not possible to determine metrics which fully describe the observed behavior. This means that the metrics used may not fully capture the observed behavior. Visual inspection of the emergent behavior allows the researcher to quickly assess the nature of that behavior. This also suggests a fast method for identifying poor coefficient matrices, thus focusing analysis on why a matrix does not provide the desired behavior rather than analyzing if the desired emergent behavior was generated.

4.4.1 Test Landscape. In order to provide a consistent comparison between the baseline metrics and the test metrics, it is necessary to define test landscapes to use for evolution and testing of the coefficient matrix. This is analogous to using a training set in neural networks [59] or learning methods used in data mining approaches [17]. Each test landscape used contains threats, a goal, and a list of waypoints that the agents must follow. The path specified by the waypoints can be a straight line or a curved path, and leads from the initial position to the goal.

Threats are placed on either side of the designated path with a varying distance from each other. This creates three different situations, a saddle in the potential field as seen in Figure 28, an obstacle in the agents' path such as in Figure 30, and a "raised" saddle in between two threats as shown in Figure 29. In each of these situations agents must either accept higher risk to maintain the desired behavior, or partially compromise the desired behavior in order to reduce the total amount of accumulated risk.

Three different test cases are used with each one containing a particular feature of interest. A summary of each of these landscapes is provided in Table 6. The test landscapes, seen in Figures 28, 29, and 30 consist of 500×500 pixel wide squares containing either one or two threats with a diameter of 100 located above and below the waypoint path in the case of the Saddle and Obstacle landscapes, or on the path in the case of the Obstacle landscape. The goal also has a diameter of 100. The start location is the upper left corner of the landscape, and the goal location is the lower right corner of the landscape. The initial starting configuration for the agents is a diagonal line which is roughly perpendicular to the specified path. The waypoint path makes a direct line from the start to the end point.

The simulation terminates when one of three conditions is met: an agent leaves the map boundaries, all agents have reached the goal, or a set time limit elapses before all agents reach the goal. If the first condition occurs, the fitness value of the simulation is set to the maximum representable double value in IEEE Standard 754-1985 [79]. The second condition is the desired finish point, and the fitness function does not have a penalty applied to it when this condition is met. If the third condition is met, the fitness function is set to the maximum representable double value.

Map	Prominent Feature
Saddle	Broad saddle in between two threats with path passing through center of saddle
Obstacle	Obstacle placed in the center of the path with no other threats in the landscape
Overlap	“Raised” saddle at overlap of two threats with path passing through the overlapping area

Table 6 Comparison of features contained in each of the test cases used

The time limit used for condition three is based on the agents’ minimum speed and the size of the landscape. For example, in our experiments an agent’s minimum speed is 20.577 m/s, which translates to 74 kph. With a 500×500 landscape, the farthest an agent should have to fly to get to the goal is 1000 pixels, or 100 km (moving along the horizontal edge of the map, followed by moving along the vertical edge of the map to the goal, or vice versa). Based upon the minimum speed, the longest amount of time an agent should take is 4865s. While it is possible for an agent to fly a longer course than this by taking a serpentine course to the goal, a limit of 5000s is deemed to be adequate time to allow all agents to reach the goal.

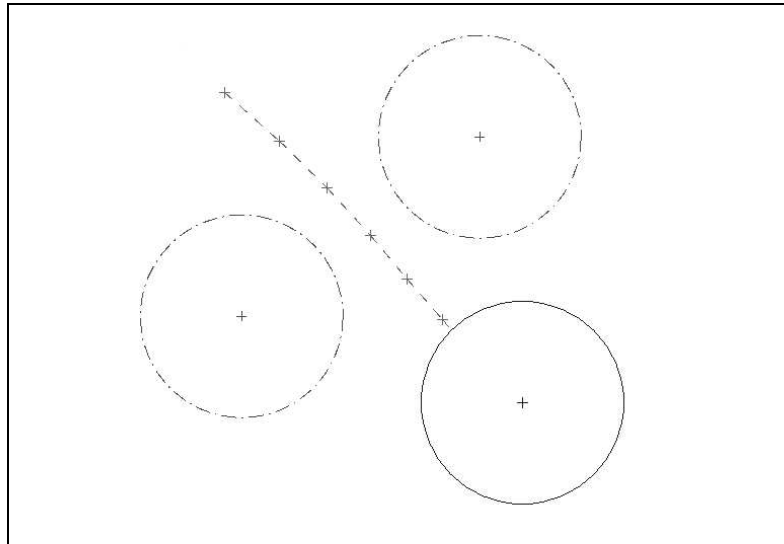


Figure 28 Saddle landscape with broad saddle feature

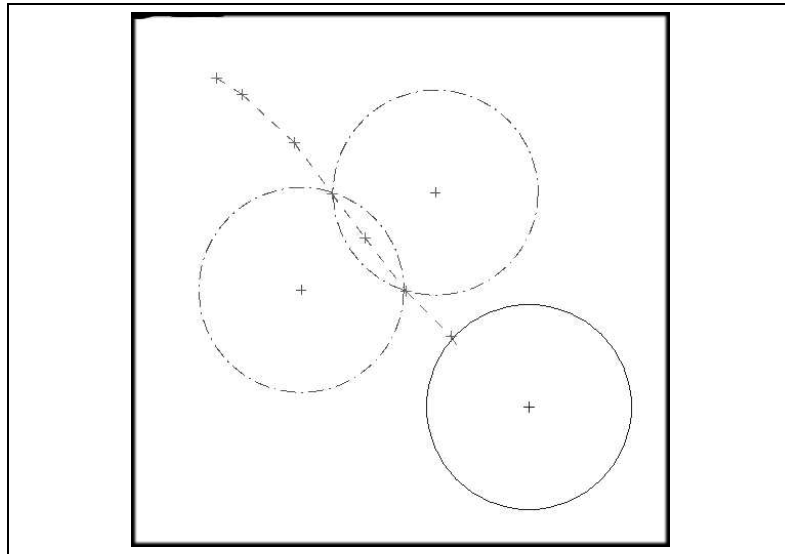


Figure 29 Overlap landscape with “raised” saddle feature

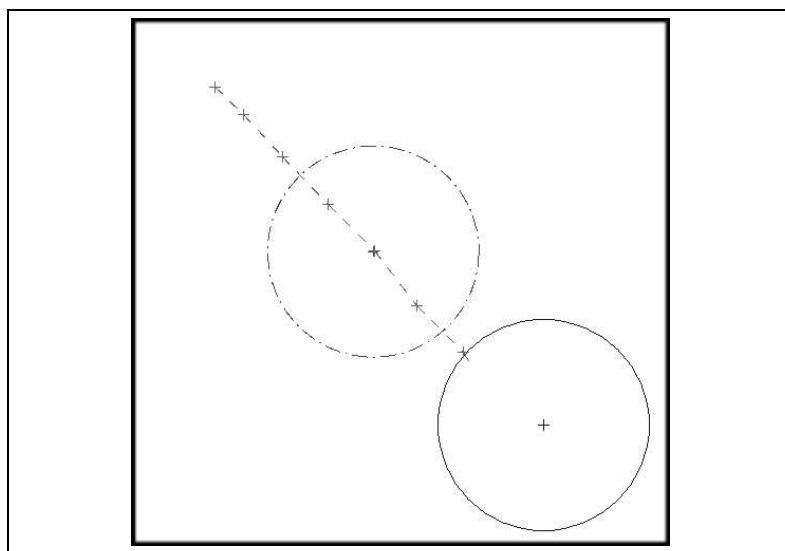


Figure 30 Obstacle landscape with obstacle feature

Small sized landscapes are used for the initial testing due to the computation time required to process a single simulation. Since the scale of our landscape is 1 pixel = 100 meters, a 500×500 pixel landscape is a representation of a 50km wide area. Since the agent's minimum vehicle speed is 74kph, it is necessary to simulate up to 5000 steps of the algorithm for an agent to reach the goal. This process takes approximately 22s on an Intel Pentium IV® dual 1.0 GHZ processor machine. Since the ES performs 100 generations and produces 100 children each generation, the total running time for a small landscape is approximately 220000s, or 61 hours. This time is reduced through the use of high performance computation techniques however, small landscapes are still necessary in order to obtain results in a timely manner.

4.5 *Generality*

Since the ES algorithm only optimizes a coefficient matrix for a specific landscape, it is important to determine whether a matrix is capable of performing well on multiple landscapes. Rather than using an ES to design the coefficient matrix for each specific mission, it is desirable for the adaptation algorithm to be capable of working in a wide variety of missions with little or no modification of the coefficient matrix. Experiments are performed for the best found coefficient matrix in order to determine whether the algorithm is capable of this. This test is performed on a landscape that incorporates all of the features of the test landscapes as well as behavior transitions in order to study all three behaviors in a single simulation run. The metrics discussed in Section 3.5 are then used to analyze how well the given coefficient matrix maintained the desired behavior in a new landscape.

The landscape used to test the generality of the algorithm is much larger than the test cases, and incorporates each of the aspects of the test cases, that is a wide saddle, overlapping threats, and an obstacle in the path. The test landscape also changes behaviors at certain waypoints in order to test the ability of the swarm to modify its behaviors during flight. Figure 31 shows the larger sized landscape used to test these aspects of the adaptive algorithm, as well as the location at which the behaviors change. The initial behavior of the swarm is set to be the reconnaissance behavior discussed in Section 3.3.8. Agents begin

the simulation at location A in Figure 31. At location B, the agents transition to a scan behavior. At location C, the agents transition to an en-route behavior.

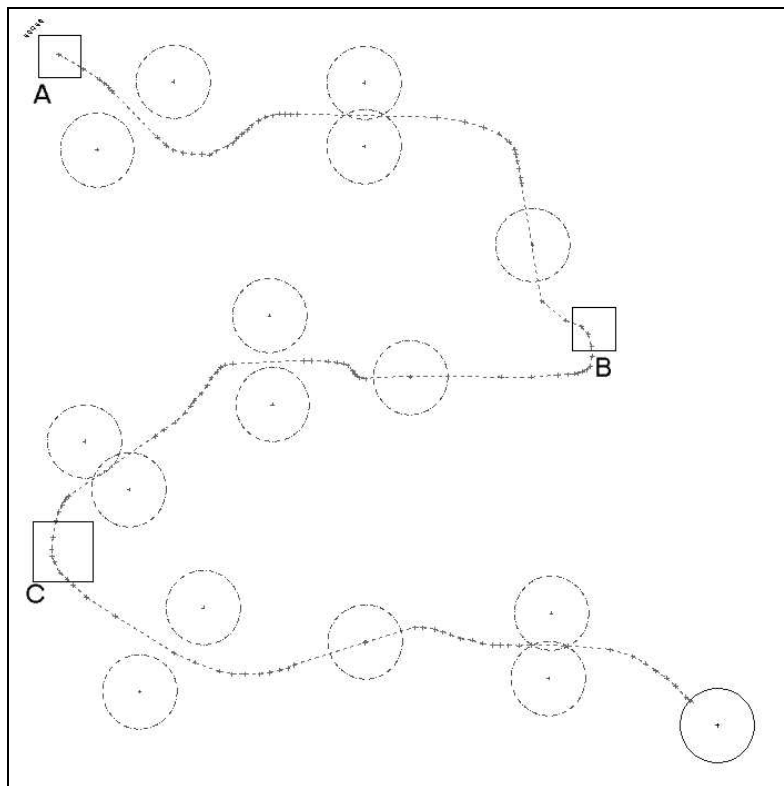


Figure 31 Landscape used to test the generality and scalability of the evolved coefficient matrix. The starting point is in the upper left corner of the landscape and the goal is in the lower right corner. This landscape is 2000×2000 pixels wide, which simulates a 200km wide area.

4.6 Scalability

Another aspect of importance to this investigation is the scalability of the adaptation algorithm. As UAVs become smaller and sensor packages are reduced in size, it becomes more desirable to utilize swarms of several hundreds or even thousands of UAVs. In this case, it is important that the control algorithm be able to scale with little or no degradation to the desired emergent behavior. To determine the algorithm’s scalability, tests are performed using $|S| = 1024$ in order to determine its scalability. The test landscape used for this larger sized swarm is the same as that used for the generality tests.

Due to the large number of agents used, the initial position of each agent affects the time to converge to a stable swarm structure. It is important that this behavior emerge prior to the swarm moving too closely to any threats that may deform the overall shape of the swarm. In order to ensure that the swarm is given sufficient time to converge, the agents are initialized in a triangular grid formation with distance between agents set to 250m. This formation is theorized by Kadrovach to be the most stable structure based upon the neighborhood model used [38]. The distance of 250m is determined by finding the point of intersection between the cohesion and separation equations (shown in Figure 6).

4.6.1 Statistical Methods. Since the Evolution Strategy algorithm described in Section 3.6 is a stochastic algorithm, it is necessary to perform a statistical analysis on the results produced in order to obtain a reasonable expectation of performance. This section discusses the statistical methods used to tune the ES's parameters.

Data gathered through experimentation does not provide a complete picture of the population distribution. Coupled with the fact that the mean and standard deviation of the distribution are unknown, it is necessary to use estimators that rely upon the central limit theorem [60]. These estimators are referred to as the sample mean \bar{s} , and the sample standard deviation s . For a sample set $X = X_1, X_2, \dots, X_n$ these values are calculated as

$$\bar{s} = \frac{\sum_{i=1}^n X_i}{n} \quad (44)$$

$$s = \sqrt{\sum_{i=1}^n \frac{(X_i - \bar{s})^2}{n-1}} \quad [60] \quad (45)$$

$$(46)$$

In order provide a statistical analysis of a sample set of data, it is necessary to determine the minimum size of the sample set in order to estimate μ to within d units with $100(100 - \alpha)\%$ confidence. This value can be determined using the equation

$$\begin{aligned}
n &\doteq \frac{(t_{\alpha/2})^2 \sigma^2}{d^2} && \text{standard deviation known} \\
n &\doteq \frac{(t_{\alpha/2})^2 \hat{\sigma}^2}{d^2} && \text{standard deviation unknown} \quad [60]
\end{aligned} \tag{47}$$

where $t_{\alpha/2}$ is area under the curve for a T distribution with $n - 1$ degrees of freedom, and $\alpha/2$ standard deviation and d is the distance between the sample mean, \bar{s} and the upper confidence interval $\bar{s} + (t_{\alpha/2})^2 \hat{\sigma}^2$. This test assumes that the underlying distribution of the sample data is a normal distribution [60]. While normality is assumed, the underlying distribution of the process may not be normal in nature. Unfortunately, due to the long computation times involved in the evaluations for the ES algorithm, it is not possible from a time standpoint to perform enough experiments to provide a reasonable expectation of the underlying distribution.

Several tests exist that indicate with some level of confidence that the underlying distribution is either normal or non-normal however, each of these tests have undesirable characteristics for this problem domain. The Shapiro-Wilks test for example, loses significance as the size of the data set decreases [78]. The Lilliefors test for normality on the other hand, works with small sample sizes. However, it is sensitive to outliers within the data set [60]. For the purpose of this investigation, minimum, maximum, and average values are given as well as sample standard deviation in order to give rough idea of the distribution of the data. While this does not provide an accurate estimate of the underlying distribution, an average value that is approximately centered between the minimum and maximum values indicates that a normal distribution is likely.

4.7 Summary

This chapter provides the experimental methodology for validating the swarm model developed in Chapter 3. Furthermore, parameters are determined for the ES in order to evolve good results for the experiments described. These parameters are determined through the use of statistical methods also defined in this chapter. Test cases are defined for the experiments performed and a baseline case is defined. Finally a general test case is developed for testing scalability and robustness of the swarm algorithm.

5. Results and Analysis

5.1 Introduction

This chapter contains data obtained through the experiments described in Chapter 4. An analysis of the data is conducted using a baseline data set as described in Section 4.3 as the comparison set. Experimental results are also discussed using visualization of the swarm’s emergent behavior. This visualization leverages the pattern matching and recognition capabilities of a human operator in order to assess the overall effectiveness of a behavior in meeting its desired goal.

5.2 Fitness Evaluation

One aspect of ESs discussed in 3.6 is that the fitness function must be carefully crafted in order to correctly capture the observed behavior resulting from a behavior matrix. Before experimentation can be performed using the different behaviors, it is necessary to analyze the results obtained using different fitness functions. This section gives the results and analyses for these tests. Since each behavior requires a separate fitness function to optimize a particular aspect of that behavior, this section discusses the fitness evaluations for each of the the three different behaviors.

5.2.1 Reconnaissance. The first fitness evaluation tested attempts to minimize the sensor overlap, f_1 , while providing a penalty for results in which one or more agents do not reach the goal, and a penalty for each agent that is separated from the swarm during the course of a run. The fitness evaluation used is given in equation 48.

$$f_{\text{recon}}(\delta_b) = O(S) * 1.1^p \quad (48)$$

where p is an accumulator which increments for each agent that has a neighborhood of zero at each timestep, as well as for each agent that has not reached the goal at time t , and $O(S)$ is the global overlap of the swarm. The motivation behind using this penalty function is to penalize solutions that sacrifice swarm cohesion for a lower overlap value. This penalty scheme also leads to solutions with shorter paths receiving lower fitness values.

This provides a selection pressure towards solutions which reach the goal quickly. Table 7 shows the results obtained using this fitness evaluation for the reconnaissance behavior. The value reported for each column is an average of five runs.

Map	Fitness Value	Overlap	Penalty
Saddle	3.64043E+783	3017992.8	18773.4
Obstacle	2.11977E+820	3942492.4	19658.8
Overlap	1.06688E+674	9478018.8	16115.2

Table 7 Metrics obtained using equation 48 on all three test maps

The results obtained using equation 48 consist of tightly grouped agents moving in seemingly erratic directions with no discernable structure to the swarm. The reason for this becomes clear when viewing the affect that the penalty function has on the overall value of the fitness function. As the penalty function increases over time, the weight of the penalty function increases exponentially. This causes the algorithm to favor population members that have low penalty values regardless of overlap because the penalty value has overshadowed the overlap value. Another problem observed using this fitness function is that there is no provision for threat avoidance.

Using the observations made in the previous paragraph, a new fitness function was designed that attempts to address the noted shortcomings of the previous function while still providing a penalty to solutions which lead to poor swarm cohesion and long paths. This new function performs an aggregate of the overlap, $O(S)$, penalty, p , and threat exposure, $T_a(S)$, values. Since the average values given in Table 7 are similar in order of magnitude, no weighting is used in the aggregate function. The new fitness function used is given as equation 49.

$$f_{\text{recon}}(\delta_b) = O(S) + T_a(S) + p \quad (49)$$

The results obtained using equation 49 are given in Table 8. The metrics reported in this table are obtained by running the swarm algorithm using the same behavior matrix used for Table 7. Rather than running the ES to determine the effectiveness of the result, the baseline behavior is used to simulate a swarm. Equation 49 is used to report the values

obtained by the baseline behavior. This same method was used to produce the results seen in Table 7. This methodology allows for a direct comparison between equations 48 and 49.

Map	Fitness Value	Overlap	Penalty
Saddle	8071329.307	8047324.8	23458
Obstacle	3329958.105	3310327.8	19255.2
Overlap	9071568.962	9052409.6	15797

Table 8 Metrics obtained using equation 49 on all three test maps

As can be seen from Table 8, the fitness values calculated from equation 49 are much lower than the those calculated using equation 48. The important point illustrated by this table however, is the affect of penalty on the overall fitness evaluation. In Table 7, a high penalty value such as for the Obstacle map leads to a disproportionate leap in the fitness evaluation. Equation 49 on the other hand, results in a much more reasonable increase in fitness evaluation. Since the overlap value now dominates the fitness evaluation, it is possible to evolve results that do not “focus” on the penalty value to the exclusion of the overlap value.

5.2.2 Scan. Using the lessons learned in the formulation of equation 49, the scan fitness evaluation is determined as:

$$f_{\text{scan}}(\delta_b) = (C(S) - O(S)) + (19.7392088 - L_{s^2}(S)) + p + T_a(S) \quad (50)$$

where $C(S)$ is the total sensor coverage of the swarm during the simulation, $O(S)$ is the amount of overlapping coverage, $L_{s^2}(S)$ is the look angle variance defined in equation 20, and p is the same penalty function described in equation 49.

This behavior attempts to maximize the amount of sensor overlap however, since the ES algorithm is designed to minimize the search function, overlap is subtracted from total coverage to provide a measure of fitness which decreases with increasing overlap. To create a decreasing value as look angle variance decreases, the constant 19.7392088 is subtracted from the look angle variance equation. This constant is derived as the maximum possible value of equation 20. Which occurs in the sample $\{360, 0\}$. The penalty measure is used to ensure that agents do not stray too closely too each other. This is accomplished by

penalizing any agent that moves within 10m of another agent. This value is chosen as the minimum safe distance possible between two agents.

5.2.3 En-route. Due to the relative simplicity of the en-route behavior, it is only necessary to encode a shortest path function and a threat exposure function into the fitness evaluation. Since the penalty function increments by the number of unfinished agents at each time step, the en-route behavior is optimized on shortest path as well as maintaining minimum safe distance, and swarm cohesion. Threat avoidance is expressed in terms of the threat exposure measure. The equation used for en-route is simply:

$$f_{\text{en-route}}(\delta_b) = p + T_a(S) \quad (51)$$

5.3 Baseline

The baseline metrics are defined first in order to provide a means of comparison to the other behaviors observed. Using the behavior matrix described in Section 4.3, the simulation is executed and results recorded. The results reported in Table 9 were obtained with the Saddle map. Table 10 gives the results for the Obstacle map. Table 11 give results for the Overlap map. Figure 32 shows the formation that emerges from the baseline matrix. Since the swarm algorithm is deterministic, it is only necessary to perform one run for each map using the baseline behavior matrix.

Metric	Value
Overlap:	3711976.0
Accumulated Threat:	11.945907268293919
Velocity Variance:	0.006179421795581918
Alignment Variance:	0.015239034592473388
Arrival Variance:	2034902.25
Look Variance:	0.00964591768459882
Penalty:	75
Coverage:	7096294.425669053

Table 9 Metrics obtained for the baseline behavior on the Saddle map

Metric	value
Overlap:	4094901.0
Accumulated Threat:	145.55451302412777
Velocity Variance:	0.0063572915968869445
Alignment Variance:	0.04588783554559924
Arrival Variance:	2092362.25
Look Variance:	0.007522928713479843
Penalty:	2140
Coverage:	7040548.7283041235

Table 10 Metrics obtained for the baseline behavior on the Obstacle map

Metric	value
Overlap:	2594326.0
Accumulated Threat:	774.4018113850954
Velocity Variance:	0.006246305714556331
Alignment Variance:	0.35681296272978075
Arrival Variance:	2689600.0
Look Variance:	0.014341440051668274
Penalty:	371
Coverage:	9155544.077023283

Table 11 Metrics obtained for the baseline behavior on the Overlap map

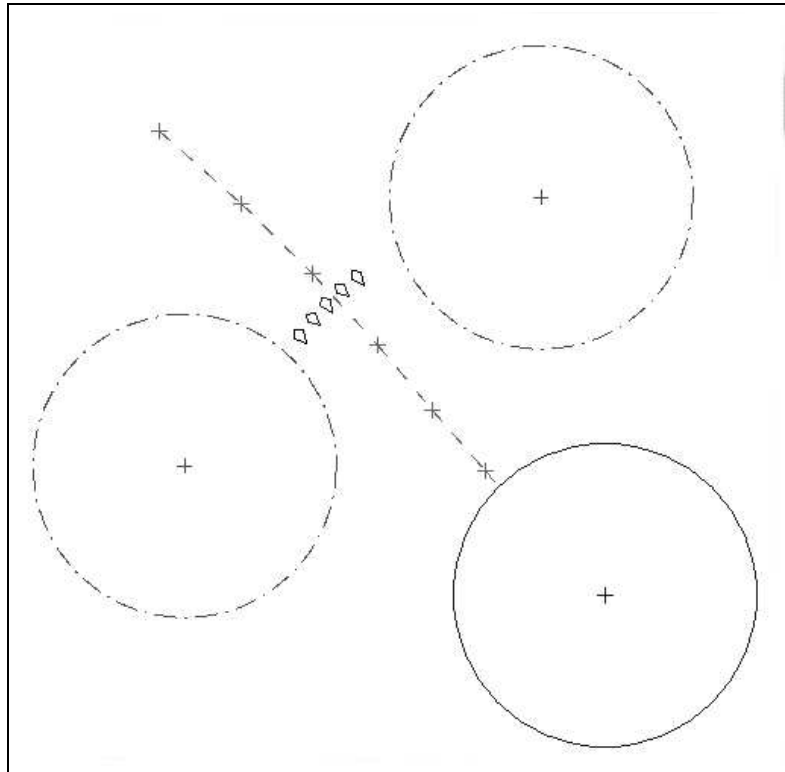


Figure 32 Figure showing snapshot of baseline swarm behavior on Saddle map

5.4 Reconnaissance

As described in Section 3.3.8, the reconnaissance behavior is intended to provide the best trade-off between area coverage and threat avoidance. As compared to the baseline metrics, the desired outcome is a low sensor overlap, high contiguous area coverage, and reasonable threat exposure value. Since reconnaissance includes determining a trade-off between threat avoidance and sensor coverage, a higher value for threat exposure is expected.

5.4.1 Saddle Map. The behavior matrices produced by the ES for the saddle map are given in Appendix A. Figure 33 shows a visualization of the swarm algorithm using behavior matrix 5. The circles around each agent in this figure represent the sensor footprint of the agent. This figure is taken after the swarm has been given ample time to converge to a stable configuration. The desired configuration for this behavior is a widely dispersed formation moving roughly in a line abreast. As the swarm approaches the saddle area, the desired formation compresses in order to reduce threat coverage. As discussed in 4.6.1, the metrics produced by the swarm algorithm using each of these matrices is averaged, and the maximum, minimum, and standard deviation are reported in Table 12.

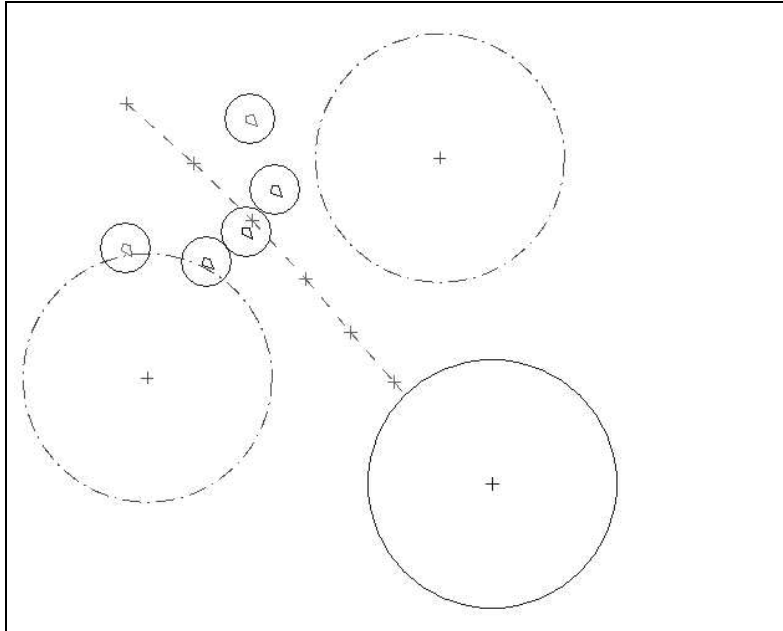


Figure 33 Visualization of swarm reconnaissance behavior on Saddle map

For all five behavior matrices generated for the Saddle map, the results are very similar. In each case, agents initially turn away from each other as much as possible in order to spread out and increase sensor overlap as shown in Figure 34. This initial behavior results in the converged behavior seen in Figure 33. The two agents on the outside “wings” of the formation are trailing far behind the other agents due to their initial reaction of turning sharply to move away from their neighbors. The total overlap of the algorithm is significantly reduced in comparison to the baseline behavior as seen in tables 12 and 9.

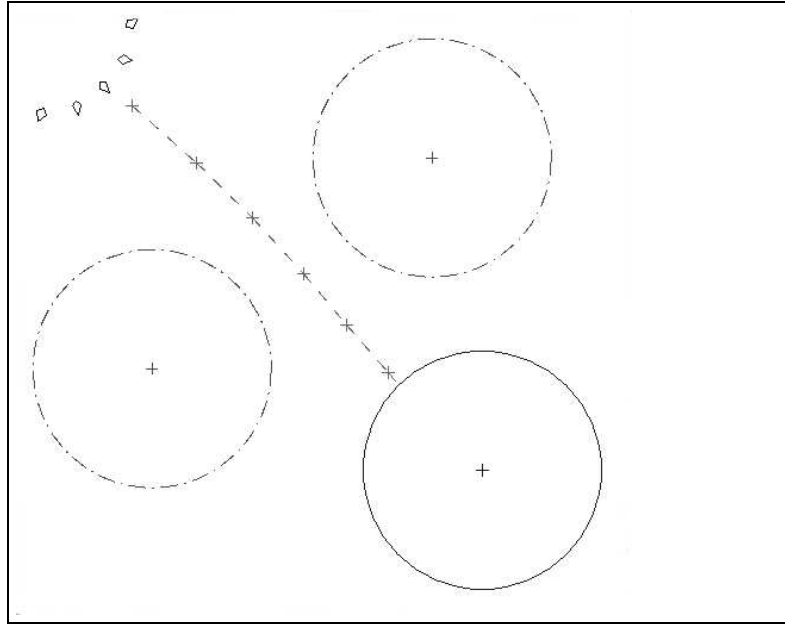


Figure 34 Visualization of initial response in reconnaissance behavior on Saddle map

Metric	min(s)	\bar{s}	max(s)	σ
Fitness	165844.1099	1165408.669	5127097.788	2214661.929
Total Overlap	164982	174128.4	183093	6487.423703
Total Accumulated Threat	54.95271419	72.10207491	105.9012332	20.28661758
Velocity Variance	0.009990875	0.014116512	0.016054478	0.002430203
Alignment Variance	0.268678109	0.340602595	0.383799294	0.047047394
Arrival Time Variance	617010.25	639123.2	703921	36503.83673
Look Angle Variance	0.022492336	0.026771268	0.028563948	0.002480376
Penalty	803	971.8	1634	370.1968395
Total Coverage	5009394.921	5082238.016	5286860.123	115955.9906

Table 12 Minimum, average, maximum, and standard deviation for metrics of the reconnaissance behavior on Saddle map

The matrices evolved by the ES algorithm point out several important features in the reconnaissance behavior. Of the five matrices evolved, two of the matrices contain zero weight for the cohesion rule and two set $\delta_b(\vec{\sigma}, f_3) = 1.0$ with all other values in the row set to zero. These weights are appropriate for this behavior because a low cohesion implies that the swarm will spread out more. Also, as agents fall behind the rest of the swarm, as shown in Figure 34, the velocity variance increases, resulting in an increase in cohesion. This strategy does not work well in practice, because the time required for the agents on the flanks of the formation to accelerate to a higher relative speed compared to their neighbors is more than the time it takes for those neighbors to move out of sight range.

For separation, the feature that appears to have the most impact on the weight is the overlap function, with three of the five matrices weighing $\delta_b(\vec{\eta}, f_2)$ at the maximum value while weighing all others in that row at zero. Again, this strategy makes sense because overlap is inversely proportional to the distance between agents. In order to decrease overlap, agents must increase separation. As can be seen in Figure 33, this strategy leads to the agents' sensor footprints touching, but not overlapping, throughout the simulation.

The goal seek behavior does not have as clear a distinction as the cohesion and separation behaviors. For the most part, the matrices weigh the goal rule, $\delta_b(\vec{g}, f_0)$ as the highest contributor to the weight, with $\delta_b(\vec{g}, f_1)$ having a small contribution. The results are inconclusive for overlap and velocity variance however, as the matrices do not have a discernible pattern in how these values are weighted.

One aspect of the reconnaissance behavior that is not addressed well is threat avoidance. During a simulation, the agents on the outsides of the formation stray into the threat areas with no discernible change in course. Three out of the five matrices set $\delta_b(\vec{t}, f_0)$ to the maximum coefficient value of 1.0, so the lack of response within threat areas is likely the result of a weak repulsion field around the threat, rather than poor matrix values.

5.4.2 *Obstacle Map.* For the Obstacle map, the desired reaction of the reconnaissance formation to the threat in the pathway is for the formation to break up into two separate lines moving in a line abreast formation while passing the threat, and then to converge back to a single line after the threat has been passed. Figure 35 shows the actual behavior of the swarm using behavior matrix 1.

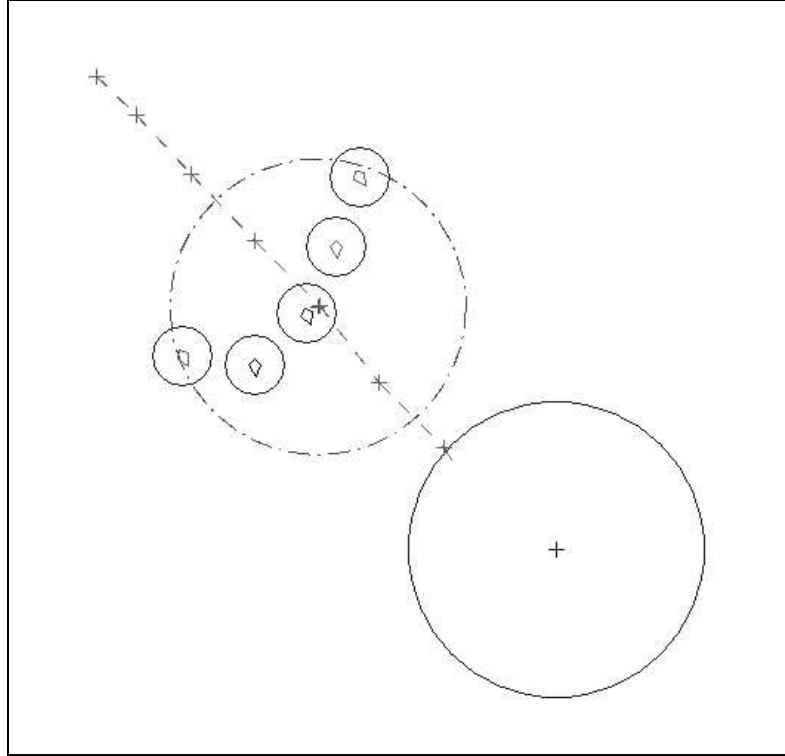


Figure 35 Visualization of swarm reconnaissance behavior on the Obstacle map

As can be seen in Figure 36, the same initial action of spreading out dominates the initial swarm behavior. Once the agents have sufficiently separated, the agents begin to move towards the goal. Unlike the behavior observed in the Saddle map, the agents all stay within sensor range of each other, with the agents on the flanks falling behind slightly, but still within sight of their neighbors. As the simulation progresses, the swarm maintains a curved “V” formation as seen in Figure 35. As can also be seen in this figure is that the swarm does not react to the threat in the path, but rather moves directly through the threat without taking any evasive action. The reason for this is likely due to the low value of threat exposure in relation to overlap in this function (see Table 13). Since threat

exposure is so low, the ES primarily optimizes on the need to maintain overlap rather than compromising between threat exposure and overlap.

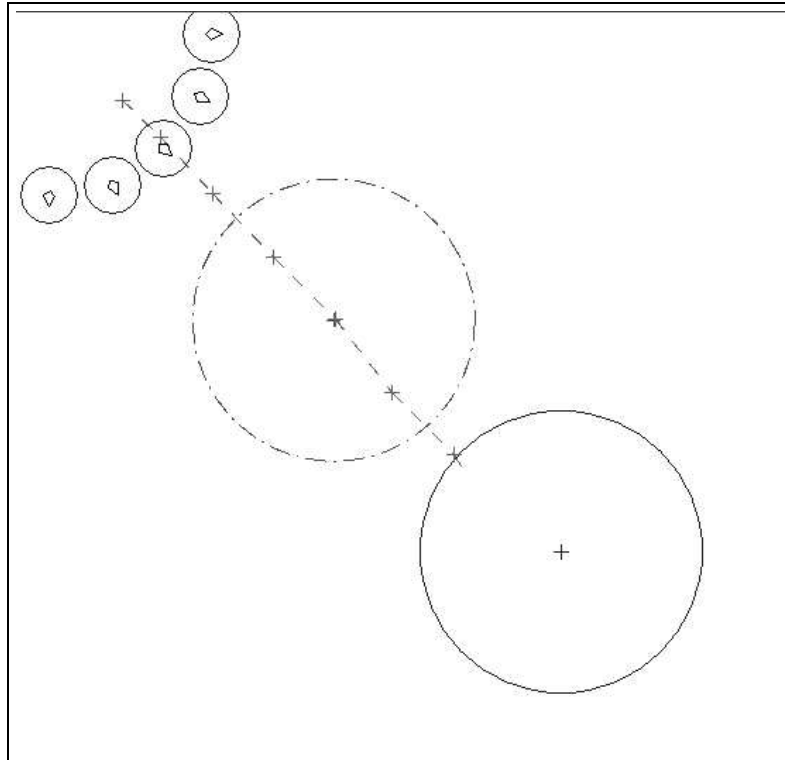


Figure 36 Visualization of initial response in reconnaissance behavior on Obstacle map

Two differences are noted between the observed behavior for the saddle map, versus the observed behavior for the obstacle map. The first of these is that the two agents on the outside edges maintain cohesion with the swarm on the Obstacle map. It is not possible based upon the data collected, to determine the exact reason that this behavior occurs. It is likely however, that this difference occurs due to the placement of the threat. In the Saddle map, the threats are on the edges, and cohesion with the swarm drives the flanking agents further into the threats. By allowing the cohesion to be broken, the two outside agents are able to move towards the center, and thereby reduce the threat exposure.

The second difference observed between these two behaviors is the gap that occurs between the agents' sensor footprint on the Obstacle map. This gap is not observed for the three agents in formation on the Saddle map. Again, while it is not possible based upon the data collected to determine why this difference exists, a plausible reason is that

Metric	min(s)	\bar{s}	max(s)	σ
Fitness	161687.8945	169775.6324	173650.7892	4890.728549
Total Overlap	160603	168695	172602	4897.116345
Total Accumulated Threat	253.7892162	272.0323801	299.1996684	17.13992304
Velocity Variance	0.013870763	0.014584391	0.015079663	0.00044511
Alignment Variance	0.322343034	0.345819599	0.377026012	0.020465469
Arrival Time Variance	629642.25	633142.15	638401	3406.660501
Look Variance	0.022412855	0.023722811	0.02493534	0.001134118
Penalty	795	808.6	823	12.89573573
Total Coverage	5043291.485	5060827.284	5087386.645	16718.40738

Table 13 Minimum, average, maximum, and standard deviation for metrics of the reconnaissance behavior on the Obstacle map

the placement of the threat drove the ES algorithm’s search space to produce minimums which contain sensor gaps.

5.4.3 Overlap Map. In the Overlap map, the desired reconnaissance behavior splits around the two overlapping threats, while also sending single or possibly multiple agents through the overlapping region. This method ensures broad sensor coverage of the area while not incurring too much threat. The actual performance of the swarm algorithm using matrix 1 is shown in Figure 37. Table 14 shows the metrics obtained for this map.

Metric	min(s)	\bar{s}	max(s)	σ
Fitness	190102.524	203821.6231	210571.1367	8238.776918
Total Overlap	189003	202198.2	208656	7899.708868
Total Accumulated Threat	313.4395557	344.823121	366.1367058	23.33513727
Velocity Variance	0.009973014	0.012212458	0.015517104	0.002346116
Alignment Variance	0.283741421	0.32814885	0.405004753	0.053375891
Arrival Time Variance	632820.25	690794.95	751689	50290.08795
Look Variance	0.017235275	0.019594231	0.024098286	0.003036264
Penalty	762	1278.6	1682	468.9715343
Total Coverage	5055705.5	5248656.198	5468255.456	171338.7044

Table 14 Minimum, average, maximum, and standard deviation for metrics of the reconnaissance behavior on the Overlap map

As with the Saddle and Obstacle maps, the reconnaissance behavior begins by spreading out as quickly as possible. For this map however, the observed behavior consists of all five members of the swarm moving out of sight range of each other, thus creating nothing more than five uncoordinated UAVs. The threat avoidance values evolved are very

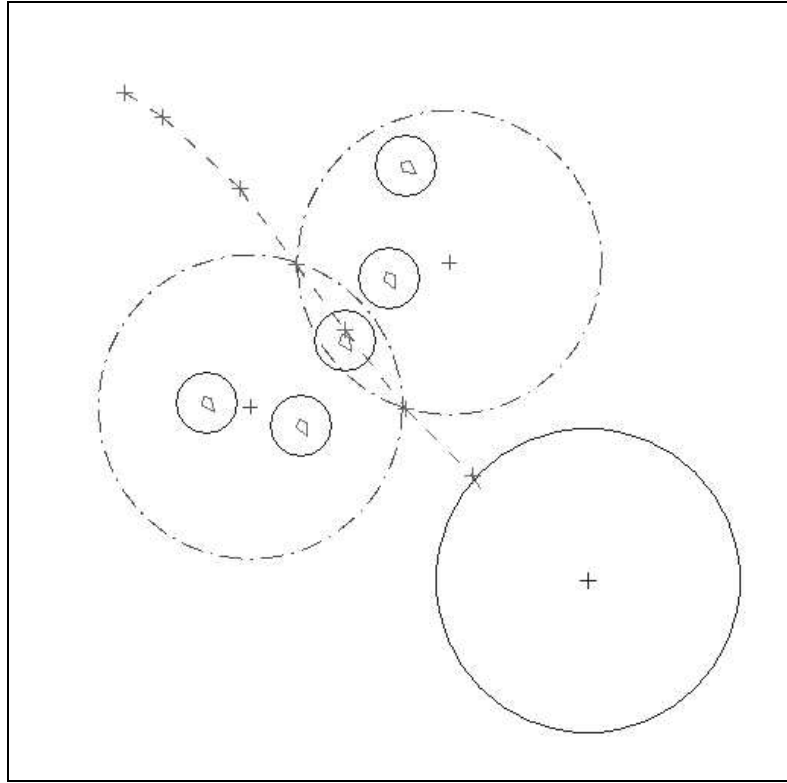


Figure 37 Visualization of swarm reconnaissance behavior on the Overlap map

low, so little reaction is observed when an agent comes into proximity with a threat. The converged formation of this behavior is shown in Figure 37.

5.5 Scan

As described in 3.3.8, the scan behavior is designed to cover a relatively small area with a large number of sensors, as well as a large variance of look angles. This behavior is designed to be tolerant of threats, exhibiting higher threat exposures in order to maximize the scanned area. As compared with the baseline metrics, scan is expected to have a high alignment variance, low arrival time variance, and higher threat exposures.

Figure 38 shows a visualization of the swarm algorithm using matrix 2 in Appendix A on the Saddle map. This figure shows the swarm after ample time has passed for convergence to a stable configuration. Since the scan behavior is designed to remain close to the path denoted by waypoints, agents are expected to pass through the saddle area

with minimal change in behavior as compared to the behavior observed prior to the saddle area. Metrics for this behavior on the Saddle map are reported in Table 15.

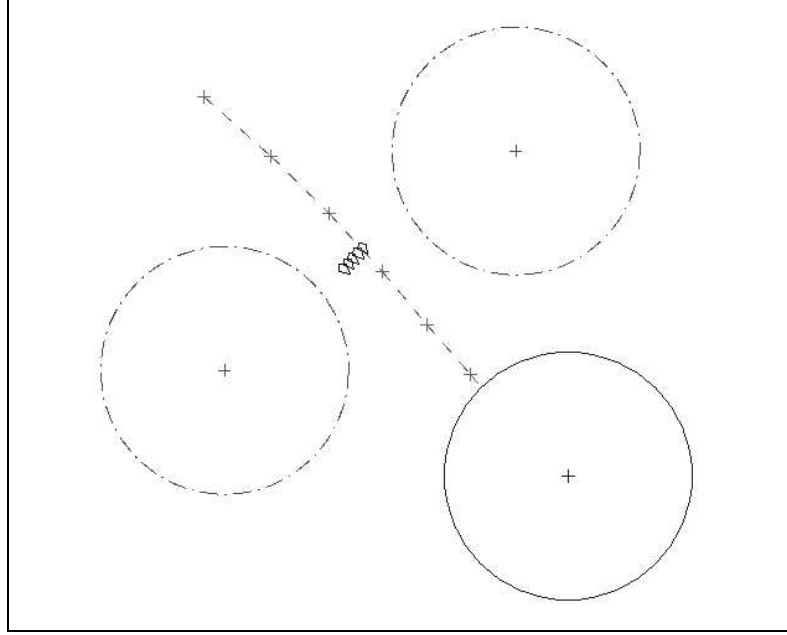


Figure 38 Visualization of swarm exhibiting scan behavior on the Saddle map

As can be seen in Figure 38, the behavior generated by the ES algorithm does not maintain a highly entropic formation as desired, but rather compresses the formation laterally. This behavior maximizes the sensor overlap, but does not maximize the look angle variance. The reason for this can be found in Table 15. The average look angle variance measured for this behavior is on the order of 1×10^{-2} . When this is subtracted from the constant 19.7392088, the resulting number is still five orders of magnitude less than the result of the overlap function, $C(S) - O(S)$. The penalty function also overshadows this value. Based upon these observations, as well as the knowledge gained from developing f_{recon} , it is very likely that a better behavior can be developed by properly scaling the different components of f_{scan} with respect to each other. A new fitness equation is not suggested here due to the costly run-time of the ES algorithm on a new fitness function.

In Figure 39, the behavior of the swarm is shown as agents encounter the threat on the Obstacle map. The desired response of the scan mode in this situation is for the agents to either split into two distinct groups and continue scanning along the desired route, or for

Metric	min(s)	\bar{s}	max(s)	σ
Fitness	2704242.448	2714284.29	2719767.31	6276.661671
Total Overlap	2108295	2175780.8	2225021	42939.68614
Total Accumulated Threat	1.556724514	1.778375504	2.136617385	0.214593837
Velocity Variance	0.001463264	0.001793917	0.002099682	0.000288784
Alignment Variance	0.019732845	0.022333729	0.029507718	0.004162825
Arrival Time Variance	543169	557445	568516	9636.441304
Look Variance	0.010687063	0.011276045	0.011761629	0.000463124
Penalty	1139	1357.2	1511	134.9544368
Total Coverage	3565250.338	3591323.362	3612444.851	19388.29496

Table 15 Minimum, average, maximum, and standard deviation for metrics of the scan behavior on the Saddle map

the agents to skirt the edge of the threat in a single group. Since this behavior is intended to accept a great deal of threat in order to scan along the intended route, the distance that the agents should begin to react to the threat is intended to be small. As can be seen in Figure 39, the agents do not react to the threat in any perceptible manner. Table 16 gives the metrics computed for the scan behavior on the Obstacle map.

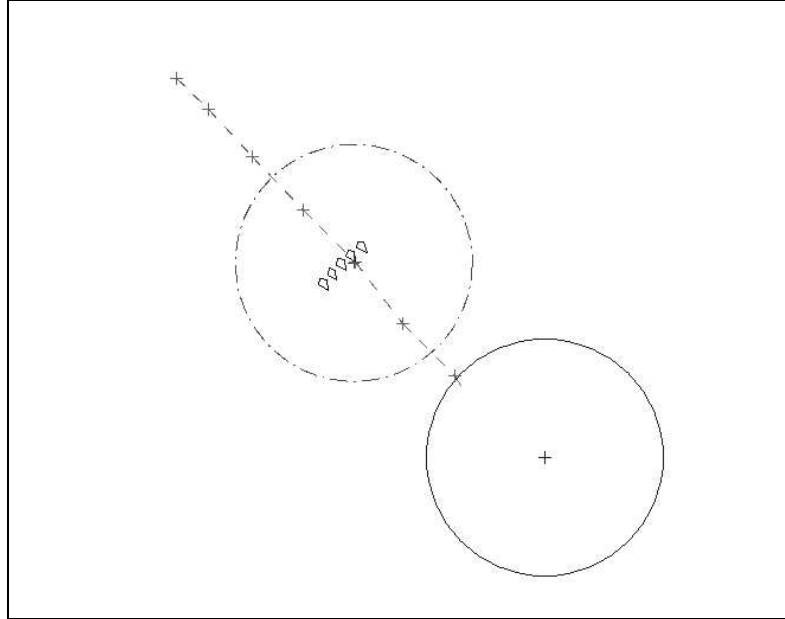


Figure 39 Visualization of swarm exhibiting scan behavior on the Obstacle map

The lack of response to the threat in the Obstacle map leads to a relatively high TAT value however, as pointed out previously, the value of $C(S) - O(S)$ is roughly 1×10^6 .

Metric	min(s)	\bar{s}	max(s)	σ
Fitness	2857384.583	2871616.892	2892713.802	14121.15814
Total Overlap	1994129	2066365	2142525	52743.09443
Total Accumulated Threat	456.0224362	472.2224194	497.4757014	16.22432782
Velocity Variance	0.001166643	0.001758434	0.002243806	0.000422752
Alignment Variance	0.0244932	0.031719869	0.045583621	0.00837767
Arrival Time Variance	553536	569493.5	592900	14415.33975
Look Variance	0.005960215	0.006540499	0.007185598	0.000441822
Penalty	676	966.8	1242	216.4178828
Total Coverage	3667572.272	3697804.709	3754223.816	33350.98588

Table 16 Minimum, average, maximum, and standard deviation for metrics of the scan behavior on the Obstacle map

This means that the TAT is overwhelmed by the $C(S) - O(S)$ component of the aggregate. Furthermore, since the penalty in this scenario is greater than the TAT, the ES is optimizing by first minimizing the sensor coverage, followed by the penalty, and finally the TAT. Since a decrease in TAT would cause an increase in p for this map due to the extra time the agents must take to circumvent the threat, the better solution is to keep penalty low. This behavior suggests that a multi-objective approach may be capable of providing a good continuum of values that offer the planner the ability to choose how much TAT and penalty the swarm should incur.

For the Overlap map, the desired scan behavior reacts to the overlapping threats by compressing the agents between the threats. This maximizes the amount of sensor coverage over the path while avoiding threat as much as possible. Figure 40 shows that the swarm reaction to this landscape feature is as desired. While the agents in Figure 40 appear to have collided, due to the map scale, the distance between agents in this scenario is actually greater than 100 m in this figure. Table 17 provides the calculated metrics for this behavior on the Overlap map.

While the swarm’s behavior on the Overlap map is as desired in terms of its reaction to the overlapping threats, the movement of the agents still produces a very low look variance value. Furthermore, the swarm reacted to the threat in this case by compressing laterally, but it did not react to the threat in the Obstacle map. This indicates that the fitness function used is sensitive to the swarm’s landscape.

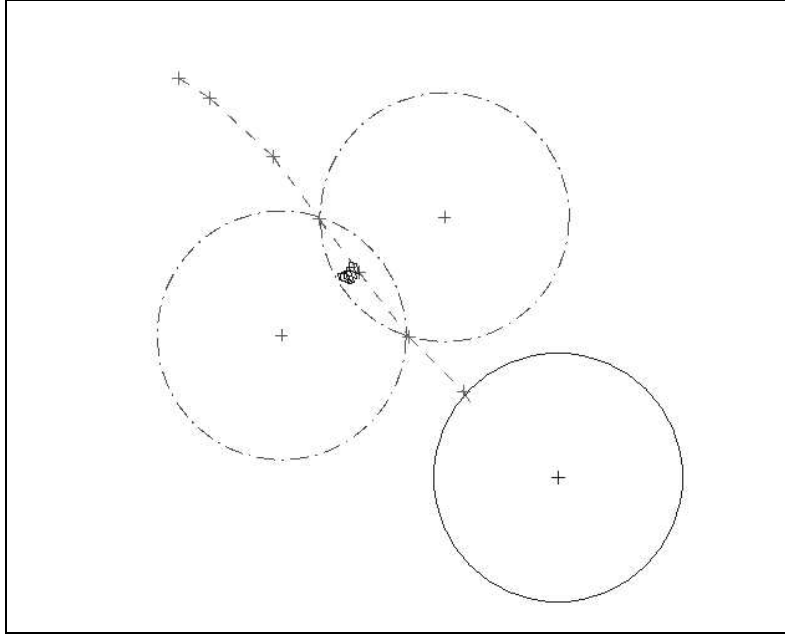


Figure 40 Visualization of swarm exhibiting scan behavior on the Overlap map

5.6 En-route

The en-route behavior is designed to be a general-purpose swarm behavior which allows for goal seek, and threat avoidance characteristics. This state only attempts to optimize the amount of time each agent takes to reach the goal. Appendix A, section A.4 gives the behavior matrices evolved for this behavior on each of the three test maps.

Metric	min(s)	\bar{s}	max(s)	σ
Fitness	2711615.316	2729204.598	2737329.385	10186.07918
Total Overlap	2121536	2159668.8	2230841	45393.59975
Total Accumulated Threat	41.89687795	49.41681811	55.4310526	5.851895292
Velocity Variance	0.001431626	0.002509129	0.003804479	0.00096784
Alignment Variance	0.035363323	0.049539765	0.074178309	0.01626828
Arrival Time Variance	547600	557010.05	574564	11402.25616
Look Variance	0.00505128	0.006176207	0.007520305	0.000991371
Penalty	947 1132.4	1472	221.4335115	
Total Coverage	3571769.531	3600384.672	3634667.592	23575.77883

Table 17 Minimum, average, maximum, and standard deviation for metrics of the scan behavior on the Overlap map

Figure 41 shows a visualization of the swarm algorithm using matrix 5 on the Saddle map. This figure shows the swarm after ample time has passed for convergence to a stable configuration. Due to the threat avoidance portion of the en-route behavior, the formation is expected to compress slightly as it passes through the saddle area of the map. This expectation is met, as shown in Figure 41. During the beginning of this simulation, the agents move in a line-abreast formation similar to the one observed in the scan mode behavior. As the agents pass between the threats however, the center agent accelerates to move forward of the other agents. This is the behavior shown in Figure 41. As the simulation continues, the swarm begins to form a circular formation however, the agents reach the goal before this shape can fully stabilize. Metrics for this behavior on the Saddle map are reported in Table 18.

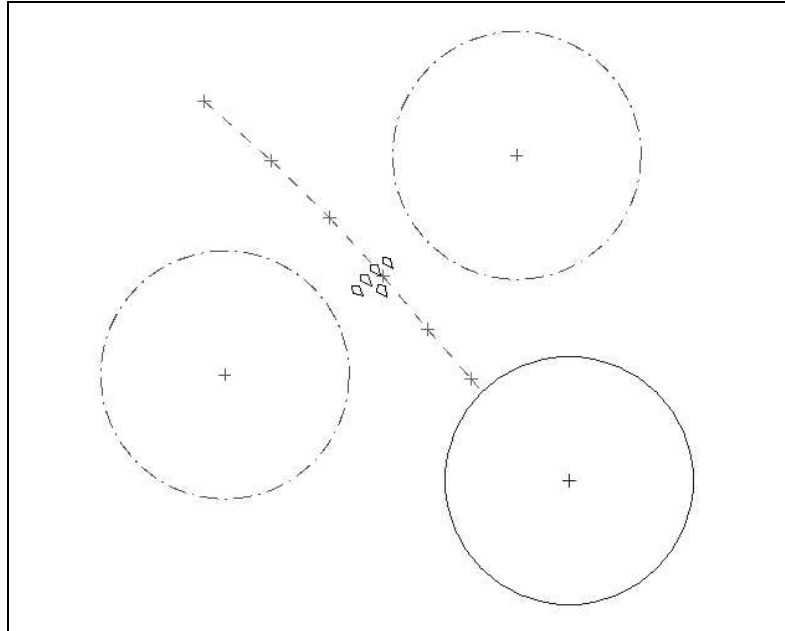


Figure 41 Visualization of swarm exhibiting enroute behavior on the Saddle map

The en-route behavior on the Obstacle map is intended to minimize threat exposure while still moving towards the goal. This means that as the agents approach the threat, their expected behavior is to split into two sub-swarms which move independently towards the goal. After passing the threat, the two sub-swarms are expected to reform into a single swarm. Figure 42 shows the actual behavior for en-route using matrix 3 on the Obstacle map. As can be seen in this figure, the agents break into two swarms before entering into

Metric	min(s)	\bar{s}	max(s)	σ
Fitness	9.394079717	10.21234458	10.54369498	0.491154993
Total Overlap	1917848	2192819	3249055	590495.1899
Total Accumulated Threat	4.394079717	5.212344577	5.543694985	0.491154993
Velocity Variance	0.002585693	0.003611108	0.005049277	0.001267287
Alignment Variance	0.028141571	0.062738053	0.150003036	0.052274759
Arrival Time Variance	537289	728292.7	1490841	426277.6164
Look Variance	0.008358835	0.012347487	0.014589385	0.002342676
Penalty	5	5	5	0
Total Coverage	3628129.871	4135221.914	6128923.169	1114532.92

Table 18 Minimum, average, maximum, and standard deviation for metrics of the en-route behavior on the Saddle map

the threat radius of the obstacle. The agents continue to skirt the threat with roughly the same distance from the center until they reach the opposite side. Once clear of the threat, the agents converge towards the waypoint path and rejoin just before reaching the goal.

Table 19 contains the metric values for the en-route matrices tested on this map.

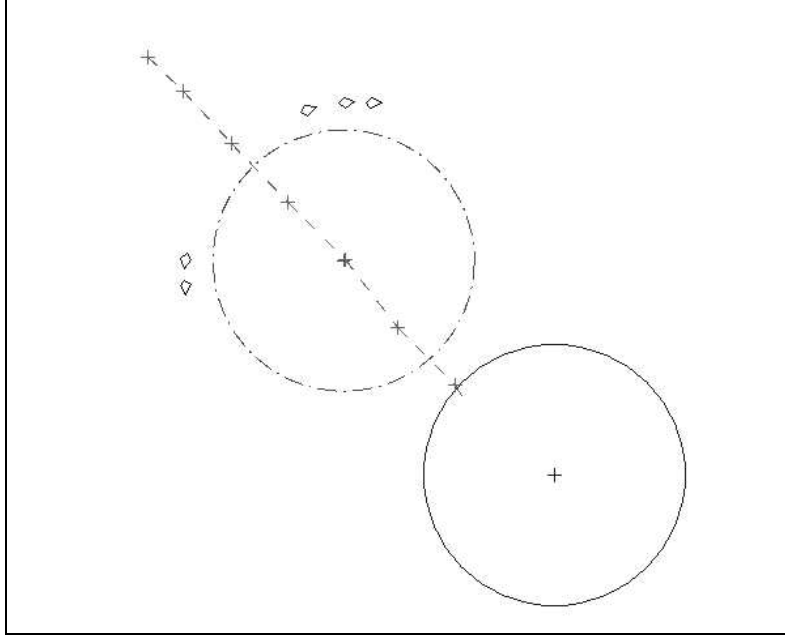


Figure 42 Visualization of swarm exhibiting en-route behavior on the Obstacle map

The exemplary behaviors observed in the enroute behavior are partially due to a fitness function that has the same order of magnitude for all components in the summation. This allows the ES the opportunity to minimize all functions simultaneously rather than

Metric	min(s)	\bar{s}	max(s)	σ
Fitness	83.74826577	111.0619455	133.9758752	22.33950394
Total Overlap	2224650	2277616.2	2372172	58759.06917
Total Accumulated Threat	22.56504892	36.86194547	50.97587517	13.88865734
Velocity Variance	0.004366112	0.007847868	0.010596205	0.002694711
Alignment Variance	0.096992087	0.192037808	0.256956526	0.072556373
Arrival Time Variance	786769	847323.3	937992.25	62860.25899
Look Variance	0.020754529	0.024205965	0.027814004	0.003187465
Penalty	60	74.2	83	9.679876032
Total Coverage	4502749.761	4692490.52	4987288.134	207213.5094

Table 19 Minimum, average, maximum, and standard deviation for metrics of the en-route behavior on the Obstacle map

focusing on the minimization of a single component at the expense of all others. This is accomplished due to the simplicity of the en-route fitness function as compared to the reconnaissance and scan functions.

The desired behavior of the en-route mode on the overlap map is to move through the center of the two overlapping threats in the most expedient manner. This can either consist of the formation compressing laterally to pass through the threats, or even better, moving through the threats in a line in order to guarantee that each agent passes through the lowest possible threat potential. The actual observed behavior is shown in Figures 43 and 44. Contrary to the expected behavior of compression as the swarm approaches the threats, the swarm spread out into a triangular grid just before the threats (see Figure 43). Immediately before entering the overlapping area however, the swarm very quickly adjusts the formation to two lines in a row (see Figure 44). This behavior is likely due to the broad local minimum just before the two threats. When the agents enter into this area, the threat repulsion is cancelled by the goal attraction, so the swarm quickly converges to the formation which results in cohesion and separation cancelling out. This is why the triangular grid is hypothesized by Kdrovach to be the most stable formation for the model used [38]. Once the agents move out of the local minimum area, the threat repulsion quickly pushes the swarm into a new formation that better minimizes the threat exposure. Table 20 gives the metrics calculated for this simulation.

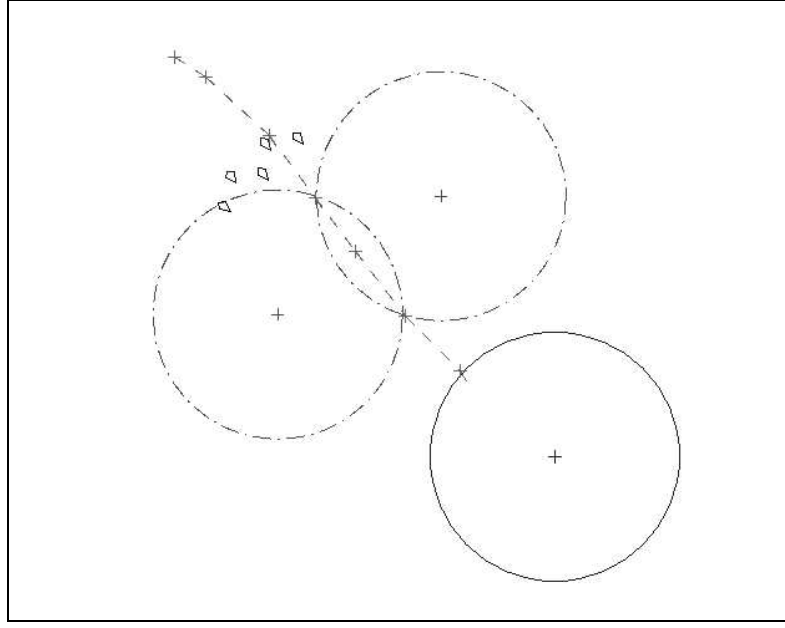


Figure 43 Visualization of swarm exhibiting en-route behavior on the Overlap map

5.7 Matrix Values

In order to understand which state functions have the greatest contributions to an observed behavior, it is necessary to look at the results obtained by the ES for each different behavior. This section provides a visual comparison of the weight coefficients obtained by the ES algorithm and develops an understanding of which weights are most important to the success of a given behavior mode in achieving the desired behavior.

Metric	min(s)	\bar{s}	max(s)	σ
Fitness	91.1228535	125.7406957	145.7105806	21.91788708
Total Overlap	1799358	2130433.6	3196202	597487.04
Total Accumulated Threat	71.71816456	96.74069571	132.5042512	25.1390764
Velocity Variance	0.005158287	0.008856643	0.010963287	0.002297082
Alignment Variance	0.165933128	0.221216519	0.278209087	0.047111729
Arrival Time Variance	588289	837080.1	1572516	417624.5226
Look Variance	0.006486697	0.011142143	0.013620281	0.002789738
Penalty	5	29	65	32.86335345
Total Coverage	3929909.169	4662534.926	6384057.614	1008730.958

Table 20 Minimum, average, maximum, and standard deviation for metrics of the en-route behavior on the Overlap map

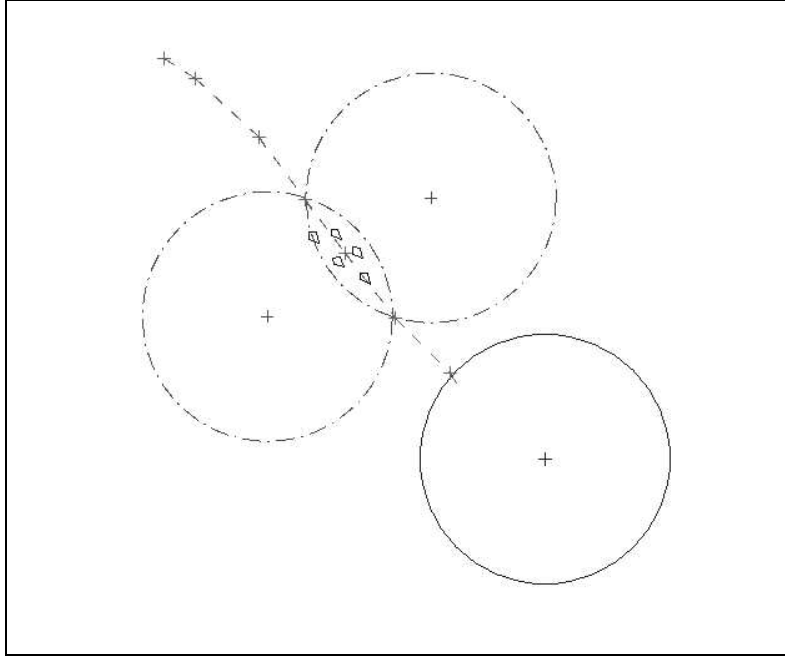


Figure 44 Visualization of swarm exhibiting en-route behavior on the Overlap map

Figure 45 depicts the coefficient weights evolved by the ES in five runs for the Reconnaissance mode on the Saddle map. Each row in the figure represents a row in the behavior matrix, and each column is numbered on the x axis of the bar graphs. The coefficient value for a particular coefficient is given as a bar with height equal to the coefficient value. The gray shading is used to depict the experiment number and ranges from experiment one through five.

The coefficient values evolved for the Reconnaissance Behavior on the Saddle map are widely distributed. Several coefficient values are strongly correlated however, indicating that the coefficient value in this portion of the matrix may contribute to this particular behavior on this map. The areas of importance in Figure 45 are in row one, columns one, two and three, row two, column three, row three, column four, and row four, column two. This indicates that it is important for the Reconnaissance mode on the Saddle map to maintain low or zero values for coefficients $\delta_{\vec{\sigma},f_0}$, $\delta_{\vec{\sigma},f_1}$, and $\delta_{\vec{\sigma},f_2}$, and $\delta_{\vec{g},f_1}$. It also appears to be equally important for values of $\delta_{\vec{\eta},f_2}$, $\delta_{\vec{t},f_0}$ and $\delta_{\vec{t},f_3}$ to be near one. One feature of interest in this figure is that all of the cohesion ($\vec{\sigma}$) coefficients with the exception of $\delta_{\vec{\sigma},f_3}$

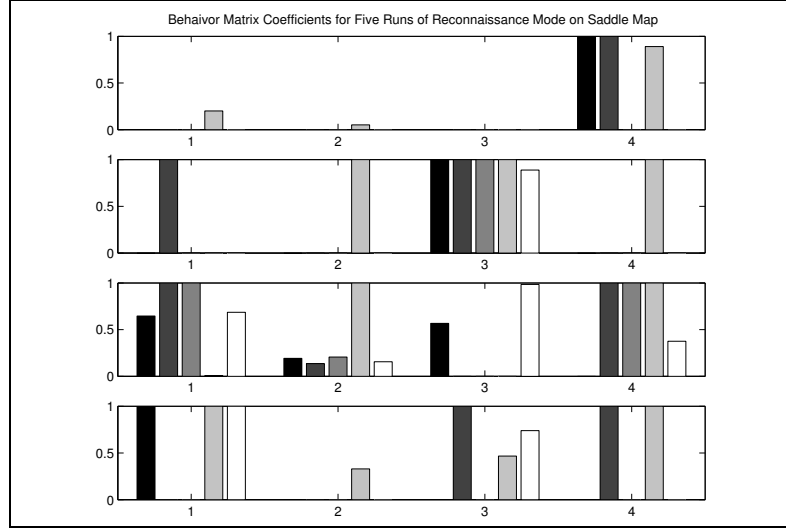


Figure 45 Comparison of matrices evolved for the Reconnaissance Mode on the the Saddle map

are zero. This indicates that a low weight for $\vec{\sigma}$ is necessary to achieve a reconnaissance behavior.

Figures 46 and 47 indicate several interesting relationships for the Reconnaissance mode across different maps. First, several features can be seen across these two figures as well as Figure 45. These features indicate that some values of the matrix are global across swarm domains. These features do not have full correlation across all five runs however, their appearance in all three behavior matrices for the same mode in different domains provides some confirmation that these feature may be important to the emergence of reconnaissance behaviors. These features are $\delta_{\vec{t}, f_3}$, which maintains a relatively high value for all three domains, and $\delta_{\vec{g}, f_0}$, which also remains high. Coefficients that remain either at or near zero for all three domains are $\delta_{\vec{\sigma}, f_1}$, and $\delta_{\vec{\sigma}, f_2}$. The fact that these values consistently remain low across all domains indicates that sensor coverage and sensor overlap are not good measures in regards to calculating $\omega_{\vec{\sigma}}$.

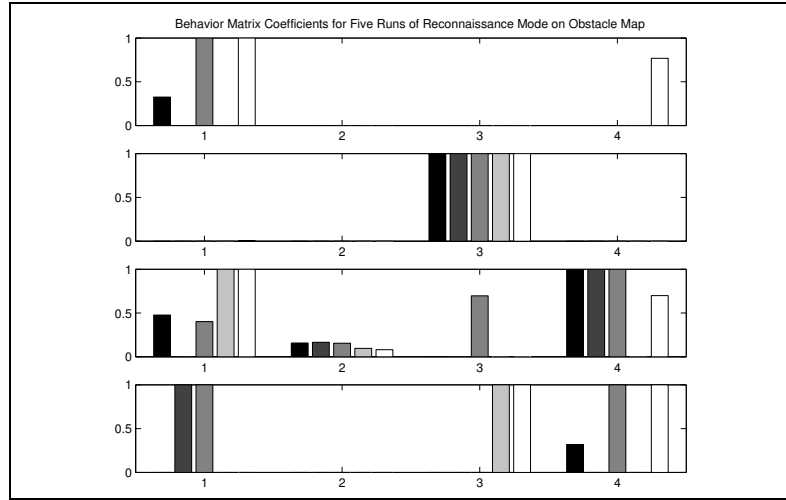


Figure 46 Comparison of matrices evolved for the Reconnaissance Mode on the the Obstacle map

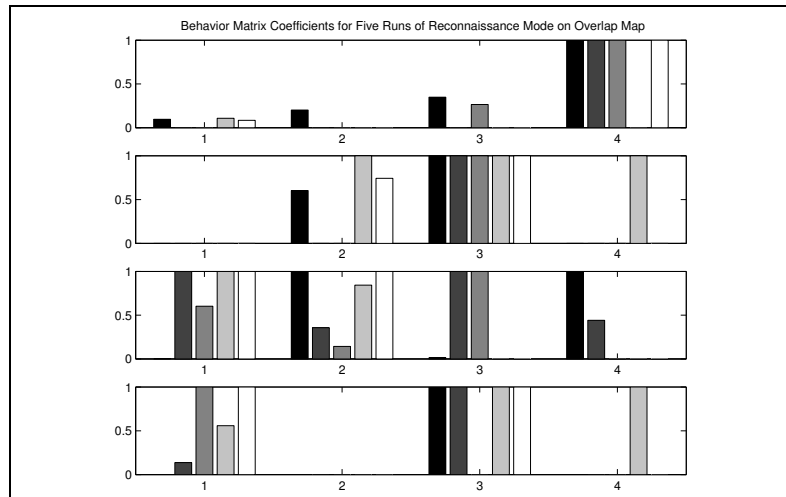


Figure 47 Comparison of matrices evolved for the Reconnaissance Mode on the the Overlap map

These analysis techniques, when applied to the scan and en-route behavior matrices reveal that similar structures exist between these behaviors. While it is not within the scope of this research, further analysis of the relationships between behavior matrices appears to be a promising area of study for further refinements of the behavior adaptation method developed for this effort. The scan and en-route behavior matrices are provided in Appendix B.

5.8 Scalability

The final experiment conducted is the scalability test. For this test, a swarm is initialized in a large map consisting of multiple behaviors and containing all of the characteristics contained in the three individual test maps. The best behavior matrices found for the Saddle map are used. The swarm is initialized with three different behavior matrices, and the landscape is designed with waypoints that change the behavior of the swarm at given locations in the landscape. The resulting behavior is analyzed visually rather than numerically because the changing behaviors each require a different measure of fitness. This test is conducted with a swarm size of $|S| = 1024$.

The five member swarm performs as expected on the larger scale map, exhibiting the same behavioral characteristics as on the smaller test maps. Since only one behavior matrix is used for all three characteristics - saddle, obstacle, and overlap - the resulting behavior is not necessarily optimal for the given landscape characteristic. One area of interest for the small swarm is the time required to converge to a new behavior after passing a mode transition point. At each transition point, the swarm quickly transitions to the observed characteristics for the new behavioral mode. This indicates that it is possible for a swarm to switch between modes within a short time spans (on the order of 30 seconds to one minute).

The results produced for a 1024 member swarm are much more interesting than the five member swarm. The first emergent behavior observed is that the swarm begins a high entropy process of attempting to converge to behavior that characterizes the current behavioral mode. Since agents in the center of the swarm do not have any space to move without increasing $\vec{\eta}$, many of the inner agents simply turn in circles until the forward edge

of the swarm has moved sufficiently far away for the agent to move forward. The affect of the waypoint width becomes apparent quickly as the swarm moves past the first set of threats (see Figure 48).

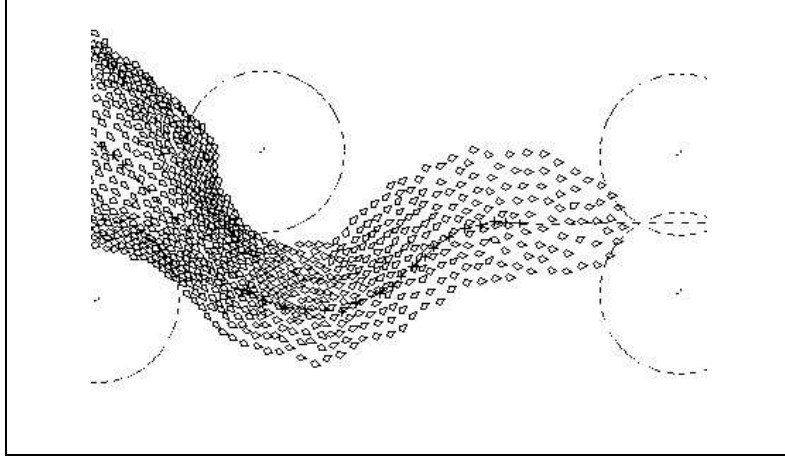


Figure 48 Emergent behavior of swarm with $|S| = 1024$ after passing first saddle obstacle using Reconnaissance mode

When the swarm encounters the first overlapping threat region, several behaviors are observed that are of interest. For example, as agents on the edge of the swarm encounter higher repulsion from one of the two threats in the overlapping region, they begin to turn in circles, first moving away from the threat, and then moving away from nearby neighbors, which leads back towards the threat. This behavior is reminiscent of separating airflow. As agents in the back of the swarm move forward, the agents on the edges eventually move around the threat. This results in the swarm splitting to flow around a threat as shown in Figure 49.

When the larger sized swarm reaches the transition between reconnaissance and scan behaviors, a different global behavior emerges than that observed for the five agent swarm. Rather than moving in a close proximity line-abreast formation, the agents form a single-file line that is tightly packed from nose to tail (see Figure 50). This behavior persists until the transition from scan to en-route.

At the en-route transition point, the single-file line dissipates into a more spread-out formation. The swarm formation of the en-route mode is very similar to the formation observed for the reconnaissance mode (see Figure 51).

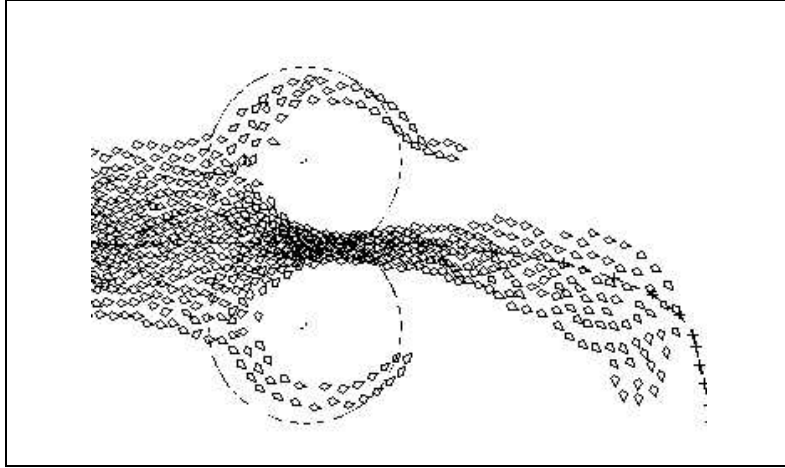


Figure 49 Emergent behavior of swarm with $|S| = 1024$ as it encounters a set of overlapping threats while in Reconnaissance mode

A swarm of size 1024 exhibits the same behavioral characteristics of the smaller 5 agent swarm with the exception of separation. Due to the high numbers of agents involved, the 1024 sized swarm becomes very densely packed at turns in the landscape. This behavior is similar to the manner in which a slinky® must compress on the inside during a turn while the outside contracts. Figure 52 shows this behavior at a turn in the landscape. As compared to a 5 agent swarm, the larger sized swarm has a much more pronounced turbulence-like behavior when encountering threats. Agents near the threat circle over a small area while neighbors pass by on the periphery. This allows the swarm to easily flow around threats as seen in Figure 53.

5.9 Summary

The results obtained using the swarm model developed in Chapter 3 indicate that it is possible to obtain emergent behavior that closely matches the desired emergent behavior. The values of coefficient matrices have been shown to hold structure among the same behavior mode for different maps. This indicates that it may be possible to develop a theoretical approach to determining the values for behavior matrices. The structures also indicate that potential exists for possibly more effective fitness functions to be formulated to better describe the state space. While the initial results using this model are encouraging,

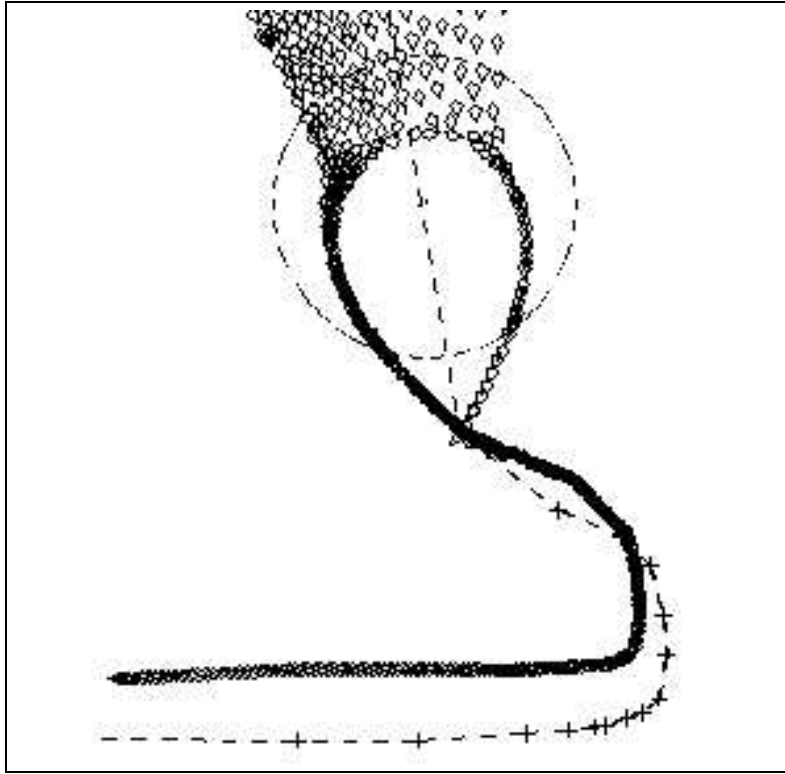


Figure 50 Emergent behavior of swarm with $|S| = 1024$ for the Scan mode

much is still unknown about the complex interactions between coefficient matrices and agent rules.

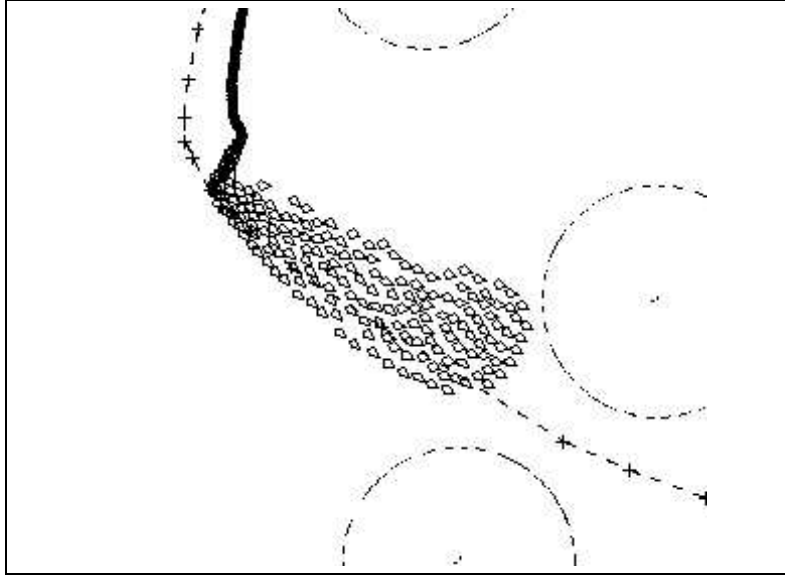


Figure 51 Emergent behavior of swarm with $|S| = 1024$ as the swarm transitions from Scan to En-route mode

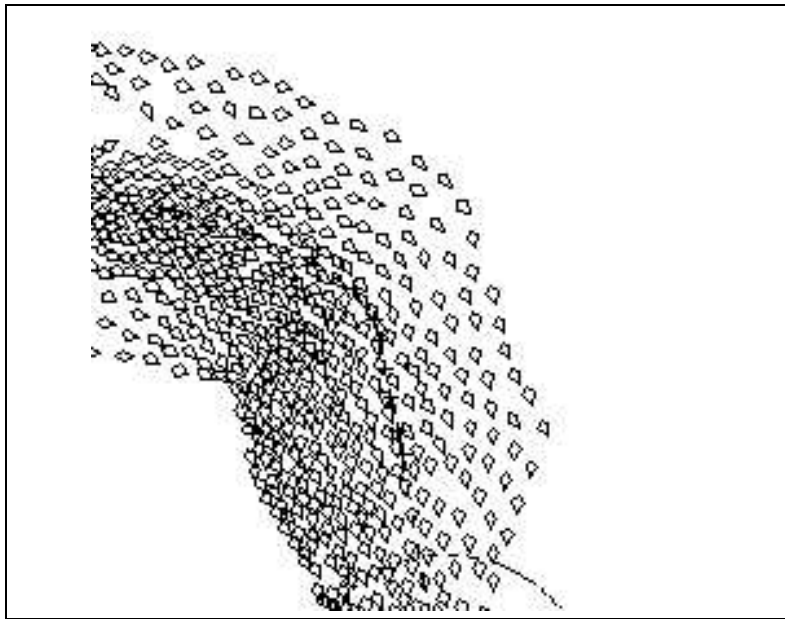


Figure 52 Emergent behavior of swarm with $|S| = 1024$ as it encounters a turn in the waypoint path

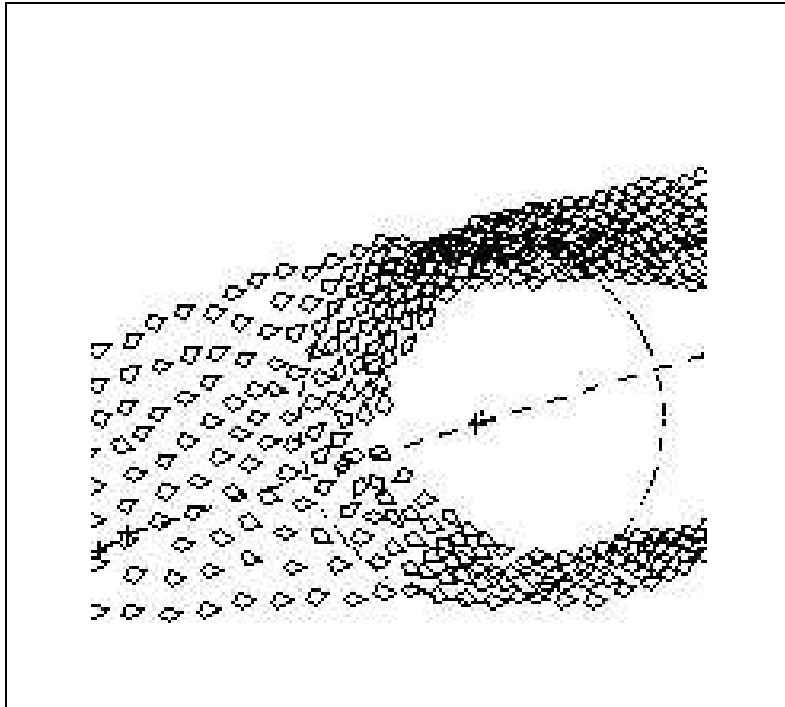


Figure 53 Emergent behavior of swarm with $|S| = 1024$ as it encounters an obstacle threat in En-route mode

6. Conclusions

6.1 Introduction

As stated in Chapter 1, the goal of this research is to develop a means of creating desired emergent behavior in a swarm of autonomous UAVs. Three objectives are given to accomplish this goal. These objectives are: 1) develop a swarm model that is capable of moving through a given landscape along a path specified by waypoints while avoiding threats, 2) define a set of behaviors for testing, and 3) develop an automated method for finding the necessary set of rule interactions to cause a desired emergent behavior.

Objective one is met in Chapter 3 where a model is developed that is capable of moving through a given landscape along a path specified by waypoints while avoiding threats. This model utilizes potential fields to represent threats, goals, and waypoints in a landscape. Agents traverse the resulting potential field based upon a set of rules.

Objective two is accomplished in Chapter 3 where three behaviors are developed. These behaviors are reconnaissance, scan, and en-route. Each of these behaviors is defined mathematically in Chapter 5. These behaviors are used to test the ability of the swarm model to achieve the desired emergent behavior utilizing a given behavior matrix.

The third objective is met through the use of an Evolution Strategy algorithm described in Chapter 3. Chapter 4 experimentally determines good parameters for the ES algorithm. These parameters are then used to automatically create behavior matrices for three different desired behaviors on three different test landscapes. The results of this automated process are presented and discussed in Chapter 5.

6.2 Swarm Model

The swarm model developed in Chapter 3 is shown in Chapter 5 to produce a swarm of agents that exhibit threat avoidance, path following, and collision avoidance between agents. Using the baseline behavior defined in Chapter 4, it is possible for all members of the swarm to reach the goal state while at the same time maintaining a cohesive swarm formation. Furthermore, as shown in Section 5.8 this model is capable of scaling from

small numbers of agents $|S| = 5$, to very large numbers $|S| = 1024$. This indicates that the model proposed and used in this research is a robust model in terms of scaling.

The use of behavior matrices for behavior modification is also shown in Chapter 5 to provide a means of modifying the overall emergent behavior of a swarm. While the results achieved do not fully meet the design criteria of the behaviors tested, the results discussed in Sections 5.4, 5.5, and 5.6 reveal that it is possible to modify the swarm's behavior through the use of behavior matrices. This indicates that the practice of modifying interaction rules has a potential to provide fruitful results in future research efforts.

6.3 *Fitness Evaluation*

The development of fitness evaluation functions for the development of behavior matrices proved to be difficult given the lack of a theoretical foundation. Without a good understanding of how a particular fitness metric relates to the desired emergent behavior, it is difficult to develop meaningful fitness functions that fully express the desired behavior. As discussed in Chapter 5, this leads to behaviors which do not fully meet the design criteria for that behavior. On the other hand, even the partially described behaviors used in this effort result in behaviors that closely resemble the desired behavior in many aspects. With a better understanding of how fitness metrics contribute to the description of a desired behavior, it may be possible to develop behavior matrices to more closely emulate the desired behavior.

An area that holds a great deal of potential for future progress in this field is the development of a framework for a formal description of swarm behaviors. This framework should be general enough to allow for possibly many different behaviors to be described. A formal behavioral description can then possibly be translated into a fitness function which more accurately describes the desired behavior. By formalizing the process of behavior definition and description, it may be possible for behaviors to be sufficiently described by the underlying fitness function.

Another approach which may prove beneficial for this area of research is the use of a multi-objective search algorithm. Since the fitness functions used in this research consists

of a weighted aggregate of fitness values, the use of a multi-objective algorithm may lead to a better understanding of the complex interactions of the fitness values, as well as lead to a set of behaviors that can be picked based upon the mission planner's desired level of safety for the swarm.

6.4 *Noisy Swarm Model*

The swarm model developed in Chapter 3 assumes that all sensor inputs provide a correct picture of the actual environment. This model does not attempt to mimic noise in sensing, or account for factors such as wind, turbulence, or hostile electro-magnetic environments. A swarm implemented in hardware must be capable of managing such environments. Since this algorithm does not account for noise, it is possible that the introduction of noise to the model can lead to diminished overall performance. It is necessary to include a noisy sensing environment as well as a dynamic landscape environment in order to determine the robustness of this algorithm to imperfect knowledge.

Another area of development for this algorithm is the migration from two-dimensional agents to three-dimensional agents. Currently, this algorithm assumes that all agents move at the same altitude, and that avoidance is only achieved by steering away from an agent or threat by turning in the plane of the swarm. A three-dimensional model introduces the ability for agents to climb or descend to avoid threats or collisions. In order to effectively function in three dimensions, the algorithm must include cost functions that determine the cost of descending, climbing, or turning for avoidance. These functions can then be used to determine what combination of turning, climbing or ascending is the most efficient for the agent.

Currently, the swarm model used does not utilize any look-ahead or predictive planning. It is possible that by providing predictive planning, more efficient overall behaviors can be achieved. For example, rather than making a large turning maneuver to avoid a collision, agents can predict that a collision is imminent if corrective action is not taken, and perform a small turn early. This may lead to more efficient individual behavior while still maintaining the desired emergent behavior.

6.5 *Summary*

This research effort provides a limited swarm model capable of performing multiple missions based upon a desired behavior for each mission. The ability to change behaviors by changing behavior matrices is tested and demonstrated. Shortcomings of the developed model are identified and discussed, and potential research areas for future efforts are also discussed.

The autonomous flight of multiple aerial vehicles promises to provide great gains to the warfighter. By utilizing resources in an efficient manner, and reducing the manpower required to conduct an operation, swarms promise to reduce the decision loop of the theater commander. Through careful design and application of engineering principles, it is possible for swarms to one day become a vital part of a world-class Air Force.

Appendix A. Coefficient Matrix Values by Test Case

A.1 Baseline Matrix

1.0	0.0	0.0	0.0
1.0	0.0	0.0	0.0
1.0	0.0	0.0	0.0
1.0	0.0	0.0	0.0

Baseline Coefficient Matrix

A.2 Reconnaissance Matrices

A.2.1 Saddle map.

0.0	0.0	0.0	1.0
0.0	0.0	1.0	0.0
0.6459361947886197	0.1921577371628287	0.5667221903585732	0.0
1.0	0.0	0.0	0.0

Landscape_Files/recon_test.lsc	
Reconnaissance Behavior	
Fitness:	173367.13691274566
Overlap:	172493.0
Accumulated Threat:	66.13691274564292
Velocity Variance:	0.01575911387760824
Alignment Variance:	0.3822069716137067
Arrival Variance:	623310.25
Look Variance:	0.028388294978885717
Penalty:	808
Coverage:	5040832.710658519

Behavior Matrix and metrics for Reconnaissance Behavior on Saddle map, result 1

0.0	0.0	0.0	1.0
1.0	0.0	1.0	0.0
1.0	0.13548603803598336	0.0	1.0
0.0	0.0	1.0	1.0

Landscape_Files/recon_test.lsc	
Reconnaissance Behavior	
Fitness:	175901.4095892962
Overlap:	175024.0
Accumulated Threat:	74.40958929621084
Velocity Variance:	0.014400291418555521
Alignment Variance:	0.33348789575073096
Arrival Variance:	629642.25
Look Variance:	0.02856394791233296
Penalty:	803
Coverage:	5057160.12951862

Behavior Matrix and metrics for Reconnaissance Behavior on Saddle map, result 2

0.0	0.0	0.0	0.0
0.0	0.0	1.0	0.0
1.0	0.20644366782753595	0.0	1.0
0.0	0.0	0.0	0.0

Landscape_Files/recon_test.lsc	
Reconnaissance Behavior	
Fitness:	165844.10992514022
Overlap:	164982.0
Accumulated Threat:	59.109925140222686
Velocity Variance:	0.01437780129763794
Alignment Variance:	0.33484070717357145
Arrival Variance:	621732.25
Look Variance:	0.02698961196827455
Penalty:	803
Coverage:	5016942.195306933

Behavior Matrix and metrics for Reconnaissance Behavior on Saddle map, result 3

0.2001623717421428	0.05108794730951009	0.0	0.8906475342082305
0.0	1.0	1.0	1.0
0.008134171592144707	1.0	0.0	1.0
1.0	0.3287670961706531	0.4666884922232355	1.0

Landscape_Files/recon_test.lsc	
Reconnaissance Behavior	
Fitness:	184832.90123318255
Overlap:	183093.0
Accumulated Threat:	105.90123318253785
Velocity Variance:	0.009990875073783295
Alignment Variance:	0.26867810867539205
Arrival Variance:	703921.0
Look Variance:	0.022492336209964825
Penalty:	1634
Coverage:	5286860.122945665

Behavior Matrix and metrics for Reconnaissance Behavior on Saddle map, result 4

0.0	0.0	0.0	0.0
0.0	0.0	0.8884442766804643	0.0
0.6861562561012117	0.15542110453480718	0.9844193547265686	0.37505901822636145
1.0	0.0	0.7393675705069607	0.0

Landscape_Files/recon_test.lsc	
Reconnaissance Behavior	
Fitness:	5127097.787714759
Overlap:	175050.0
Accumulated Threat:	54.95271418775132
Velocity Variance:	0.016054478371859325
Alignment Variance:	0.38379929362350407
Arrival Variance:	617010.25
Look Variance:	0.027422146929026312
Penalty:	811
Coverage:	5009394.921184062

Behavior Matrix and metrics for Reconnaissance Behavior on Saddle map, result 5

A.2.2 Obstacle map.

0.3244981580652232	0.0	0.0	0.0
0.0	0.0	1.0	0.0
0.4765285340963318	0.15693895293785476	0.0	1.0
0.0	0.0	0.0	0.31874080376572833

Landscape_Files/threat_in_path.lsc	
Reconnaissance Behavior	
Fitness:	173449.0388096961
Overlap:	172356.0
Accumulated Threat:	272.03880969609463
Velocity Variance:	0.014751382072223861
Alignment Variance:	0.3223430342709401
Arrival Variance:	631230.25
Look Variance:	0.023504749772358204
Penalty:	821
Coverage:	5050954.307150005

Behavior Matrix and metrics for Reconnaissance Behavior on Obstacle map, result 1

0.0	0.0	0.0	0.0
0.0	0.0	1.0	0.0
0.0	0.1653734241132664	0.0	1.0
1.0	0.0	0.0	0.0

Landscape_Files/threat_in_path.lsc	
Reconnaissance Behavior	
Fitness:	173650.7892162423
Overlap:	172602.0
Accumulated Threat:	253.78921624230546
Velocity Variance:	0.013870763137805235
Alignment Variance:	0.3343748615057222
Arrival Variance:	629642.25
Look Variance:	0.02291583439759436
Penalty:	795
Coverage:	5043291.48478139

Behavior Matrix and metrics for Reconnaissance Behavior on Obstacle map, result 2

1.0	0.0	0.0	0.0
0.0	0.0	1.0	0.0
0.4016112877656819	0.15397243041466635	0.6957668601682138	1.0
1.0	0.0	0.0	1.0

Landscape_Files/threat_in_path.lsc	
Reconnaissance Behavior	
Fitness:	169148.1996683555
Overlap:	168051.0
Accumulated Threat:	299.19966835550423
Velocity Variance:	0.01467613549122873
Alignment Variance:	0.350174059168542
Arrival Variance:	634412.25
Look Variance:	0.0224128550521048
Penalty:	798
Coverage:	5063159.903641491

Behavior Matrix and metrics for Reconnaissance Behavior on Obstacle map, result 3

0.0	0.0	0.0	0.0
0.0	0.0	1.0	0.0
1.0	0.09597191803232846	0.0	0.0
0.0	0.0	1.0	0.0

Landscape_Files/threat_in_path.lsc	
Reconnaissance Behavior	
Fitness:	161687.89454503043
Overlap:	160603.0
Accumulated Threat:	261.8945450304439
Velocity Variance:	0.01454401133034088
Alignment Variance:	0.34518002754141375
Arrival Variance:	632025.0
Look Variance:	0.024935340498184296
Penalty:	823
Coverage:	5059344.081272876

Behavior Matrix and metrics for Reconnaissance Behavior on Obstacle map, result 4

1.0	0.0	0.0	0.76769244011095
0.005564087874642465	0.0	1.0	0.0
1.0	0.08094201682674707	0.0	0.6980343304059021
0.0	0.0	1.0	1.0

Landscape_Files/threat_in_path.lsc	
Reconnaissance Behavior	
Fitness:	170942.23966118367
Overlap:	169863.0
Accumulated Threat:	273.2396611836871
Velocity Variance:	0.015079663038279312
Alignment Variance:	0.377026011922981
Arrival Variance:	638401.0
Look Variance:	0.024845272897146983
Penalty:	806
Coverage:	5087386.644870206

Behavior Matrix and metrics for Reconnaissance Behavior on Obstacle map, result 5

A.2.3 Overlap map.

0.09518888405925759	0.20128084006021982	0.34789327429321504	1.0
0.0	0.6031602801822111	1.0	0.0
0.0	1.0	0.013670185992548656	1.0
0.0	0.0	1.0	0.0

Landscape_Files/overlapping_threat.lsc	
Reconnaissance Behavior	
Fitness:	204360.09685780824
Overlap:	202320.0
Accumulated Threat:	358.09685780823577
Velocity Variance:	0.010996615801410183
Alignment Variance:	0.30375215731746175
Arrival Variance:	751689.0
Look Variance:	0.017302545243871283
Penalty:	1682
Coverage:	5468255.456283898

Behavior Matrix and metrics for Reconnaissance Behavior on Overlap map, result 1

0.0	0.0	0.0	1.0
0.0	0.0	1.0	0.0
1.0	0.35680172398684273	1.0	0.44103419063227234
0.1367141332505969	0.0	1.0	0.0

Landscape_Files/overlapping_threat.lsc	
Reconnaissance Behavior	
Fitness:	204141.43955574496
Overlap:	203066.0
Accumulated Threat:	313.4395557449655
Velocity Variance:	0.015517103914411074
Alignment Variance:	0.40500475317637424
Arrival Variance:	632820.25
Look Variance:	0.024098285500519814
Penalty:	762
Coverage:	5055705.500132979

Behavior Matrix and metrics for Reconnaissance Behavior on Overlap map, result 2

0.0	0.0	0.26547181137459785	1.0
0.0	0.0	1.0	0.0
0.6016996679371549	0.142379151084861	1.0	0.0
1.0	0.0	0.0	0.0

Landscape_Files/overlapping_threat.lsc	
Reconnaissance Behavior	
Fitness:	190102.52403723
Overlap:	189003.0
Accumulated Threat:	326.5240372300143
Velocity Variance:	0.013802972936835618
Alignment Variance:	0.36213268999035525
Arrival Variance:	650442.25
Look Variance:	0.021364852023578733
Penalty:	773
Coverage:	5118029.03083615

Behavior Matrix and metrics for Reconnaissance Behavior on Overlap map, result 3

0.10735394428731893	0.0	0.0	0.0
0.0	1.0	1.0	1.0
1.0	0.8434512810576973	0.0	0.0
0.5585311649760426	0.0	1.0	1.0

Landscape.Files/overlapping_threat.lsc	
Reconnaissance Behavior	
Fitness:	209932.9184482661
Overlap:	207946.0
Accumulated Threat:	359.91844826611526
Velocity Variance:	0.010772584768419777
Alignment Variance:	0.2861132277323101
Arrival Variance:	690561.0
Look Variance:	0.01723527530889682
Penalty:	1627
Coverage:	5231707.962856852

Behavior Matrix and metrics for Reconnaissance Behavior on Overlap map, result 4

0.08409370269055039	0.0	0.0	1.0
0.0	0.7420070753677647	1.0	0.0
1.0	1.0	0.0	0.0
1.0	0.0	1.0	0.0

Landscape_Files/overlapping_threat.lsc	
Reconnaissance Behavior	
Fitness:	210571.1367058031
Overlap:	208656.0
Accumulated Threat:	366.13670580309366
Velocity Variance:	0.009973013659392117
Alignment Variance:	0.28374142087523163
Arrival Variance:	728462.25
Look Variance:	0.017970197622180908
Penalty:	1549
Coverage:	5369583.039614783

Behavior Matrix and metrics for Reconnaissance Behavior on Overlap map, result 5

A.3 Scan Matrices

A.3.1 Saddle map.

0.0	0.11798441890756257	0.06264895540938643	0.20902382636688133
1.0	0.0	0.0	1.0
0.0	0.0	1.0	0.0
1.0	1.0	1.0	1.0

Landscape_Files/recon_test.lsc	
Scan Behavior	
Fitness:	2712329.568371471
Overlap:	2225021.0
Accumulated Threat:	1.5567245140265449
Velocity Variance:	0.00146326398956404
Alignment Variance:	0.01973284486778234
Arrival Variance:	568516.0
Look Variance:	0.01068706276020968
Penalty:	1511
Coverage:	3612444.850999159

Behavior Matrix and metrics for Scan Behavior on Saddle map, result 1

1.0	0.049355010038414464	0.05902823847405375	1.0
0.0	0.0	0.0	1.0
0.0	0.0	1.0	1.0
0.8658297405019696	1.0	0.20623589095277045	1.0

Landscape_Files/recon_test.lsc	
Scan Behavior	
Fitness:	2704242.447753567
Overlap:	2168960.0
Accumulated Threat:	1.7225345198517452
Velocity Variance:	0.0017386565005328473
Alignment Variance:	0.019905413177920654
Arrival Variance:	553536.0
Look Variance:	0.011313817371977898
Penalty:	1380
Coverage:	3577643.497927361

Behavior Matrix and metrics for Scan Behavior on Saddle map, result 2

0.0	0.0	0.06022992970560268	0.6138694216188119
1.0	0.0	0.0	0.0
0.0	0.0	1.0	0.9680048962787019
0.0	1.0	0.0	1.0

Landscape_Files/recon.test.lsc	
Scan Behavior	
Fitness:	2719767.3095446127
Overlap:	2108295.0
Accumulated Threat:	2.136617385050915
Velocity Variance:	0.0020996816741451034
Alignment Variance:	0.0224631988829693
Arrival Variance:	543169.0
Look Variance:	0.011761629385932148
Penalty:	1139
Coverage:	3565250.3378385385

Behavior Matrix and metrics for Scan Behavior on Saddle map, result 3

0.0	0.0	0.08487297544358781	0.050549628720282236
1.0	0.0	0.0	0.5696397239352176
0.0	0.0	1.0	1.0
1.0	0.683339216829228	0.0	0.0

Landscape_Files/recon_test.lsc	
Scan Behavior	
Fitness:	2716632.827185807
Overlap:	2184092.0
Accumulated Threat:	1.7246371135817147
Velocity Variance:	0.0020834869791224735
Alignment Variance:	0.02950771817462467
Arrival Variance:	559504.0
Look Variance:	0.011677409306570904
Penalty:	1368
Coverage:	3597723.5132789523

Behavior Matrix and metrics for Scan Behavior on Saddle map, result 4

0.0	0.0	0.08695057278250695	0.0
0.5407474843036476	0.0	0.0	1.0
0.0	0.0	1.0	0.0
1.0	1.0	0.12018469048229674	0.0

Landscape_Files/recon_test.lsc	
Scan Behavior	
Fitness:	2718449.294721433
Overlap:	2192536.0
Accumulated Threat:	1.75136398968718
Velocity Variance:	0.0015844982719151243
Alignment Variance:	0.020059472156064906
Arrival Variance:	562500.0
Look Variance:	0.010940304892557514
Penalty:	1388
Coverage:	3603554.60977044

Behavior Matrix and metrics for Scan Behavior on Saddle map, result 5

A.3.2 Obstacle map.

1.0	0.04470732320283517	0.0	0.0
0.0	0.0	0.0	0.44859123066533063
0.0	1.0	1.0	0.0
1.0	0.0	0.0	0.08548718801570454

Landscape_Files/threat_in_path.lsc	
Scan Behavior	
Fitness:	2876215.179889817
Overlap:	1994129.0
Accumulated Threat:	456.0224361972709
Velocity Variance:	0.002071053695207398
Alignment Variance:	0.03196743926093791
Arrival Variance:	553536.0
Look Variance:	0.006468016034148916
Penalty:	838
Coverage:	3667572.2720502354

Behavior Matrix and metrics for Scan Behavior on Obstacle map, result 1

1.0	0.1715671914916482	0.0	0.7640930013045517
0.6611709318287691	0.0	0.0	0.0
0.0	1.0	0.0	0.0
0.0	0.0	0.0	1.0

Landscape_Files/threat_in_path.lsc	
Scan Behavior	
Fitness:	2892713.801762793
Overlap:	2142525.0
Accumulated Threat:	497.47570135637466
Velocity Variance:	0.0011666430781412938
Alignment Variance:	0.02449320032800328
Arrival Variance:	592900.0
Look Variance:	0.00596021511381369
Penalty:	1242
Coverage:	3754223.815914038

Behavior Matrix and metrics for Scan Behavior on Obstacle map, result 2

1.0	0.04380753619026611	0.0	0.0
1.0	0.0	0.0	1.0
1.0	1.0	0.0	0.0
1.0	0.0	0.0	0.0

Landscape_Files/threat_in_path.lsc	
Scan Behavior	
Fitness:	2871504.479494295
Overlap:	2066153.0
Accumulated Threat:	475.33689662204694
Velocity Variance:	0.0017263369962713473
Alignment Variance:	0.030782791779351924
Arrival Variance:	569270.25
Look Variance:	0.006650775052117437
Penalty:	1035
Coverage:	3696905.399244905

Behavior Matrix and metrics for Scan Behavior on Obstacle map, result 3

1.0	0.0	0.11407237510914824	1.0
1.0	0.0	0.0	1.0
0.0	0.0	1.0	0.0
1.0	0.14420578492083094	0.0	1.0

Landscape_Files/threat_in_path.lsc	
Scan Behavior	
Fitness:	2857384.583016624
Overlap:	2071651.0
Accumulated Threat:	460.26400194269587
Velocity Variance:	0.0022438060798810728
Alignment Variance:	0.045583620581895065
Arrival Variance:	567009.0
Look Variance:	0.007185598244890342
Penalty:	676
Coverage:	3689129.8509991597

Behavior Matrix and metrics for Scan Behavior on Obstacle map, result 4

0.0	0.03190845655311623	0.0	0.0
0.0	0.0	0.0	0.0
0.0	1.0	1.0	0.0
0.35572993763855154	0.0	0.0	1.0

Landscape_Files/threat_in_path.lsc	
Scan Behavior	
Fitness:	2860266.4153340464
Overlap:	2057367.0
Accumulated Threat:	472.0130607433838
Velocity Variance:	0.0015843288172191032
Alignment Variance:	0.025772293053592826
Arrival Variance:	564752.25
Look Variance:	0.006437892779794372
Penalty:	1043
Coverage:	3681192.2062619296

Behavior Matrix and metrics for Scan Behavior on Obstacle map, result 5

A.3.3 Overlap map.

1.0	0.0	0.0	0.0
0.19974818473090017	0.0	0.0	0.0
0.0	0.0	1.0	0.7585213648013016
0.0	0.9394010698695673	0.0	0.0

Landscape_Files/overlapping_test.lsc	
Scan Behavior	
Fitness:	2737329.385060245
Overlap:	2121536.0
Accumulated Threat:	44.57505174800332
Velocity Variance:	0.0030096980009324207
Alignment Variance:	0.0565276482660111
Arrival Variance:	549822.25
Look Variance:	0.006710437555585652
Penalty:	947
Coverage:	3591302.401435874

Behavior Matrix and metrics for Scan Behavior on Overlap map, result 1

1.0	0.056035442430221245	0.0909633195107385	1.0
1.0	0.0	0.0	1.0
0.0	0.0	1.0	0.0
0.0	1.0	1.0	0.32140566075289423

Landscape_Files/overlapping_test.lsc	
Scan Behavior	
Fitness:	2730870.5105860378
Overlap:	2178680.0
Accumulated Threat:	53.16947361477701
Velocity Variance:	0.001703109106726559
Alignment Variance:	0.03583898934026881
Arrival Variance:	562500.0
Look Variance:	0.0054187184320955645
Penalty:	1239
Coverage:	3610482.60977044

Behavior Matrix and metrics for Scan Behavior on Overlap map, result 2

0.0	0.07313941650722587	0.0	0.3722445553404776
0.0	0.0	0.0	1.0
0.0	0.0	1.0	0.0
0.0	1.0	1.0	1.0

Landscape_Files/overlapping_test.lsc	
Scan Behavior	
Fitness:	2731308.667432209
Overlap:	2230841.0
Accumulated Threat:	52.011634632211226
Velocity Variance:	0.001431626010356303
Alignment Variance:	0.03536332304163298
Arrival Variance:	574564.0
Look Variance:	0.0050512795578768616
Penalty:	1472
Coverage:	3634667.5922278785

Behavior Matrix and metrics for Scan Behavior on Overlap map, result 3

1.0	0.024274831808158315	0.0	1.0
0.0	0.0	0.0	1.0
0.0	0.0	1.0	0.4166393089363333
1.0	0.7864293318097104	0.0	1.0

Landscape_Files/overlapping_test.lsc	
Scan Behavior	
Fitness:	2711615.315866287
Overlap:	2135469.0
Accumulated Threat:	55.43105259596216
Velocity Variance:	0.002596732289760873
Alignment Variance:	0.04579055588779453
Arrival Variance:	547600.0
Look Variance:	0.006180295515668748
Penalty:	1025
Coverage:	3571769.530821514

Behavior Matrix and metrics for Scan Behavior on Overlap map, result 4

1.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.6247430865512403	0.21266254906348103
0.0	1.0	1.0	0.0

Landscape_Files/overlapping_test.lsc	
Scan Behavior	
Fitness:	2734899.108759381
Overlap:	2131818.0
Accumulated Threat:	41.896877954475244
Velocity Variance:	0.003804478800219248
Alignment Variance:	0.07417830910808453
Arrival Variance:	550564.0
Look Variance:	0.0075203045034222624
Penalty:	979
Coverage:	3593701.2238044897

Behavior Matrix and metrics for Scan Behavior on Overlap map, result 5

A.4 En-route Matrices

A.4.1 Saddle map.

0.2525583434240012	0.0	0.0	1.0
0.0	0.0	0.0	1.0
0.0	0.0	1.0	0.0
0.0	0.1266232207740662	1.0	1.0

Landscape_Files/recon_test.lsc	
En-route Behavior	
Fitness:	10.543694984679489
Overlap:	1917848.0
Accumulated Threat:	5.5436949846794885
Velocity Variance:	0.002738296127229354
Alignment Variance:	0.031051194105599922
Arrival Variance:	537289.0
Look Variance:	0.01293172975925108
Penalty:	5
Coverage:	3637854.370732692

Behavior Matrix and metrics for Enroute Mode on Saddle map, result 1

0.945253519228786	0.0	0.0	1.0
0.0	0.0	0.0	1.0
0.0	0.0	1.0	0.0
1.0	0.02788409520085891	0.31263175951435773	1.0

Landscape_Files/recon_test.lsc	
En-route Behavior	
Fitness:	10.529771379030706
Overlap:	1930392.0
Accumulated Threat:	5.529771379030706
Velocity Variance:	0.004944911345479399
Alignment Variance:	0.07339358111658822
Arrival Variance:	538022.25
Look Variance:	0.014589385368367614
Penalty:	5
Coverage:	3646662.0154699236

Behavior Matrix and metrics for Enroute Mode on Saddle map, result 2

1.0	0.0	0.0	0.0
1.0	0.0	0.0	1.0
0.0	0.0	1.0	0.0
0.0	0.37696229144243276	0.0	1.0

Landscape_Files/recon_test.lsc	
En-route Behavior	
Fitness:	10.484784472332782
Overlap:	1929503.0
Accumulated Threat:	5.484784472332782
Velocity Variance:	0.00258569310892624
Alignment Variance:	0.02814157119931034
Arrival Variance:	538022.25
Look Variance:	0.012931229673316969
Penalty:	5
Coverage:	3634540.1448555635

Behavior Matrix and metrics for Enroute Mode on Saddle map, result 3

0.0	0.0	0.008414586153125958	0.7282782600156236
0.7417144645062954	0.0	0.0	1.0
0.0	0.0	1.0	0.7686947930351413
1.0	0.15815789138812916	0.0	1.0

Landscape_Files/recon_test.lsc	
En-route Behavior	
Fitness:	10.10939233184681
Overlap:	1937297.0
Accumulated Threat:	5.109392331846809
Velocity Variance:	0.002737361826112453
Alignment Variance:	0.031100883098276782
Arrival Variance:	537289.0
Look Variance:	0.012926253019575586
Penalty:	5
Coverage:	3628129.870732692

Behavior Matrix and metrics for Enroute Mode on Saddle map, result 4

0.0	0.22297542775505827	0.0	0.0
0.0	0.0	1.0	1.0
0.0	0.0	1.0	0.0
0.0	1.0	0.0	0.0

Landscape_Files/recon_test.lsc	
En-route Behavior	
Fitness:	9.394079716613561
Overlap:	3249055.0
Accumulated Threat:	4.394079716613561
Velocity Variance:	0.0050492770772587285
Alignment Variance:	0.1500030364399237
Arrival Variance:	1490841.0
Look Variance:	0.008358834809046924
Penalty:	5
Coverage:	6128923.169059371

Behavior Matrix and metrics for Enroute Mode on Saddle map, result 5

A.4.2 Obstacle map.

1.0	0.0	0.0	0.0
1.0	0.0	0.0	0.5636087241545047
0.0	0.0	0.17470260638583232	0.7048436748193876
0.0	0.0	1.0	0.0

Landscape_Files/threat_in_path.lsc	
En-route Behavior	
Fitness:	106.17521768911183
Overlap:	2372172.0
Accumulated Threat:	36.175217689111825
Velocity Variance:	0.004366112439469898
Alignment Variance:	0.09699208659483913
Arrival Variance:	856550.25
Look Variance:	0.020972073618147017
Penalty:	70
Coverage:	4649736.513308444

Behavior Matrix and metrics for Enroute Mode on Obstacle map, result 1

0.0	0.0	0.0	0.0
0.7601664183234593	0.0	0.0	0.0
0.38853449909004284	0.04507340776685054	0.0	0.0
1.0	1.0	0.0	0.0

Landscape_Files/threat_in_path.lsc	
En-route Behavior	
Fitness:	97.56504891849491
Overlap:	2224650.0
Accumulated Threat:	22.565048918494906
Velocity Variance:	0.00973273478390525
Alignment Variance:	0.2569565256035331
Arrival Variance:	866761.0
Look Variance:	0.027814004489282406
Penalty:	75
Coverage:	4811462.107608927

Behavior Matrix and metrics for Enroute Mode on Obstacle map, result 2

0.0	0.0	0.0	0.0
1.0	0.0	0.0	1.0
0.0	0.0	0.031468843755871445	1.0
1.0	0.0	1.0	0.1521563215872311

Landscape_Files/threat_in_path.lsc	
En-route Behavior	
Fitness:	83.74826577389341
Overlap:	2295228.0
Accumulated Threat:	23.748265773893404
Velocity Variance:	0.005666798968453247
Alignment Variance:	0.13488450891817733
Arrival Variance:	937992.25
Look Variance:	0.020754529023550718
Penalty:	60
Coverage:	4987288.133930129

Behavior Matrix and metrics for Enroute Mode on Obstacle map, result 3

1.0	0.0	0.0	1.0
0.0	0.0	0.0	0.46617301271822775
0.5546836564382113	0.32494560284197793	0.0	0.2796631415203416
0.0	1.0	1.0	0.0

Landscape_Files/threat_in_path.lsc	
En-route Behavior	
Fitness:	133.97587516515676
Overlap:	2248942.0
Accumulated Threat:	50.97587516515675
Velocity Variance:	0.00887748701129305
Alignment Variance:	0.216518700865528
Arrival Variance:	788544.0
Look Variance:	0.02523643823168129
Penalty:	83
Coverage:	4502749.761110109

Behavior Matrix and metrics for Enroute Mode on Obstacle map, result 4

0.3290673585466336	0.0	0.0	1.0
1.0	0.0	0.0	0.0
1.0	0.21185277365431543	0.0	1.0
0.0	1.0	0.0	1.0

Landscape_Files/threat_in_path.lsc	
En-route Behavior	
Fitness:	133.845319826933
Overlap:	2247089.0
Accumulated Threat:	50.84531982693301
Velocity Variance:	0.010596204546527292
Alignment Variance:	0.25483721986841007
Arrival Variance:	786769.0
Look Variance:	0.026252781056476042
Penalty:	83
Coverage:	4511216.083478723

Behavior Matrix and metrics for Enroute Mode on Obstacle map, result 5

A.4.3 Overlap map.

0.0	0.0	0.4581020579944739	1.0
0.027793737069750662	0.0	0.8128024777302806	0.4273907525695184
0.0	0.0	1.0	1.0
0.0	0.4116735402659337	0.0	0.0

Landscape_Files/overlap_test.lsc	
En-route Behavior	
Fitness:	137.50425122977109
Overlap:	3196202.0
Accumulated Threat:	132.50425122977109
Velocity Variance:	0.005158287497942786
Alignment Variance:	0.1659331283627535
Arrival Variance:	1572516.0
Look Variance:	0.006486697044156371
Penalty:	5
Coverage:	6384057.614240663

Behavior Matrix and metrics for Enroute Mode on Overlap map, result 1

1.0	0.0	0.0	0.7178184725515154
1.0	0.0	0.2778435755570541	0.0
0.0	0.0	1.0	0.0
0.0	1.0	1.0	1.0

Landscape_Files/overlap_test.lsc	
En-route Behavior	
Fitness:	117.64762866825414
Overlap:	1799358.0
Accumulated Threat:	112.64762866825414
Velocity Variance:	0.008191792257053718
Alignment Variance:	0.18313543470497778
Arrival Variance:	644809.0
Look Variance:	0.011136609416188378
Penalty:	5
Coverage:	4224886.93653579

Behavior Matrix and metrics for Enroute Mode on Overlap map, result 2

0.041203173485482546	0.0	0.11221027671891412	1.0
1.0	0.0	0.3179010994471243	0.040013223049817026
0.0	0.0	1.0	1.0
0.8258249564558802	1.0	0.0	0.0

Landscape.Files/overlap_test.lsc	
En-route Behavior	
Fitness:	91.12285350487511
Overlap:	1893319.0
Accumulated Threat:	86.12285350487511
Velocity Variance:	0.009949610424931535
Alignment Variance:	0.27820908654500204
Arrival Variance:	775280.25
Look Variance:	0.011550268302525857
Penalty:	5
Coverage:	4723286.921198887

Behavior Matrix and metrics for Enroute Mode on Overlap map, result 3

0.20446049553399312	0.0	0.0	1.0
0.0	0.0	0.0782366324527285	0.0
0.0	1.0	0.5191518750632823	1.0
0.0	0.6927750866211326	1.0	0.0

Landscape_Files/overlap_test.lsc	
En-route Behavior	
Fitness:	145.71058058069144
Overlap:	1846471.0
Accumulated Threat:	80.71058058069143
Velocity Variance:	0.010963287338782383
Alignment Variance:	0.25460160673653515
Arrival Variance:	604506.25
Look Variance:	0.013620281089035692
Penalty:	65
Coverage:	4050533.989163482

Behavior Matrix and metrics for Enroute Mode on Overlap map, result 4

0.0	0.0	0.0	0.0
0.35841976067492076	0.0	0.14789883559764594	0.4326835294119389
1.0	1.0	0.0	1.0
1.0	1.0	0.0	0.0

Landscape_Files/overlap_test.lsc	
En-route Behavior	
Fitness:	136.7181645619211
Overlap:	1916818.0
Accumulated Threat:	71.7181645619211
Velocity Variance:	0.01002023564955549
Alignment Variance:	0.2242033369569116
Arrival Variance:	588289.0
Look Variance:	0.01291685712238723
Penalty:	65
Coverage:	3929909.1689858367

Behavior Matrix and metrics for Enroute Mode on Overlap map, result 5

Appendix B. Behavior Matrix Comparison

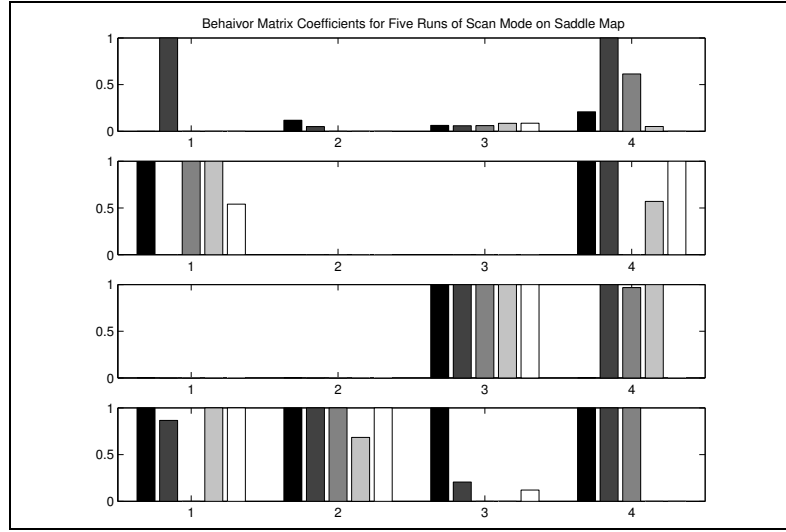


Figure 54 Comparison of matrices evolved for the Scan Mode on the the Saddle map

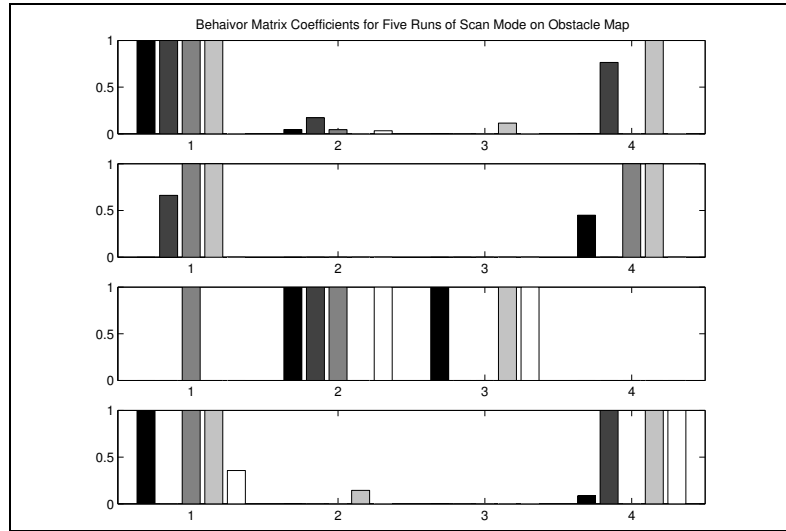


Figure 55 Comparison of matrices evolved for the Scan Mode on the the Obstacle map

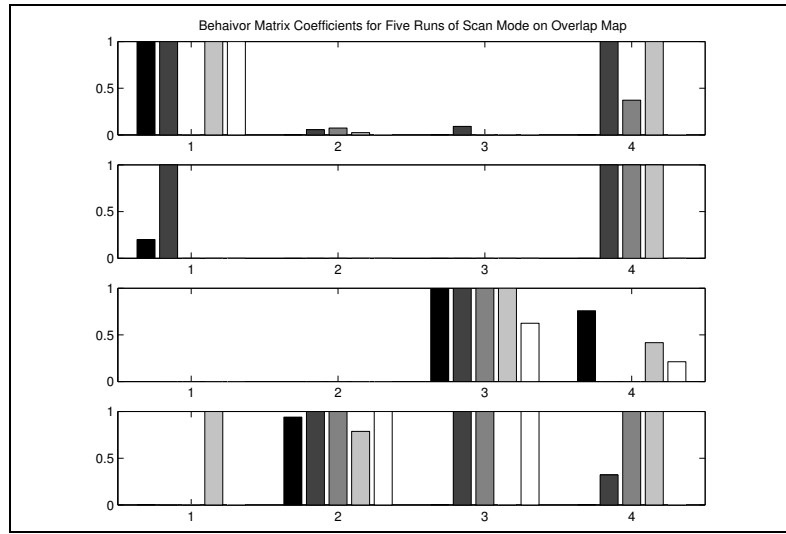


Figure 56 Comparison of matrices evolved for the Scan Mode on the the Overlap map

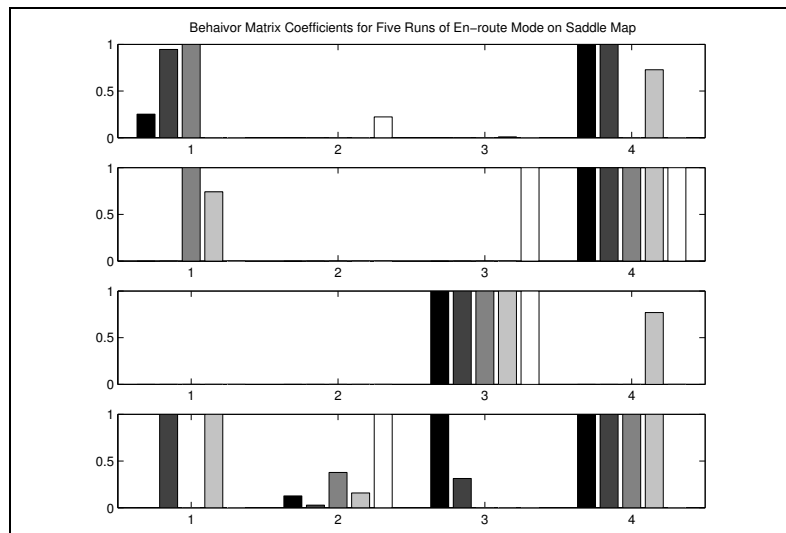


Figure 57 Comparison of matrices evolved for the En-route Mode on the the Saddle map

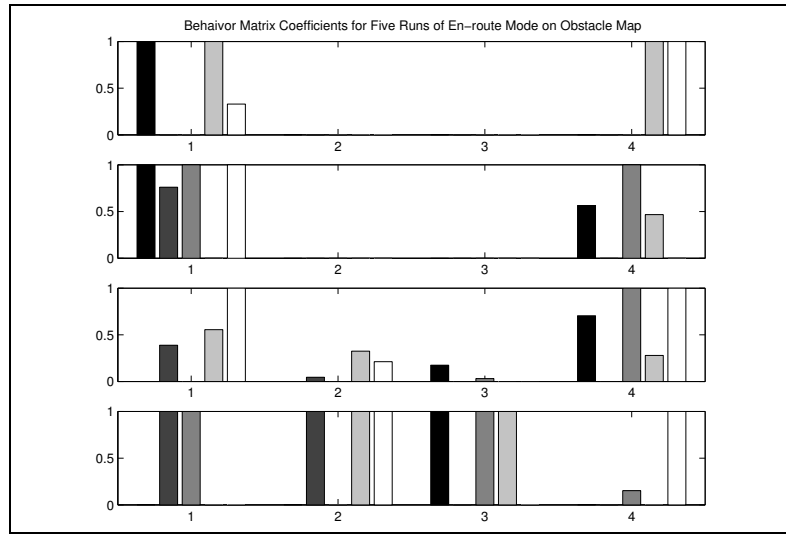


Figure 58 Comparison of matrices evolved for the En-route Mode on the the Obstacle map

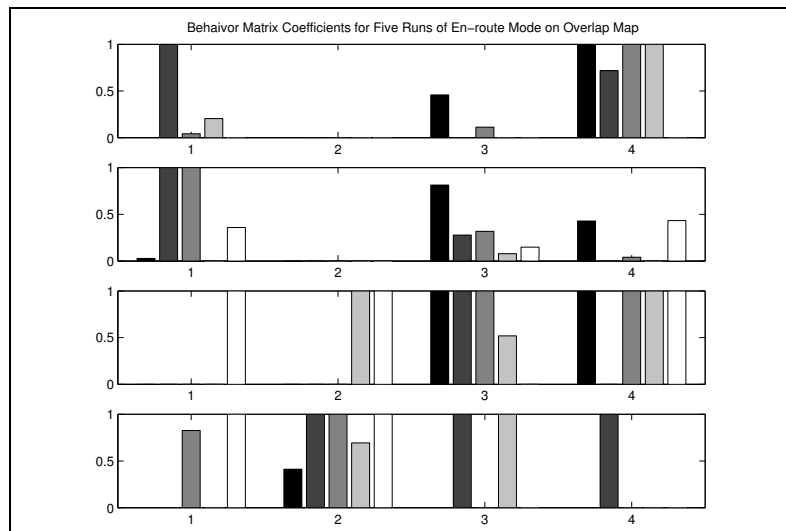


Figure 59 Comparison of matrices evolved for the En-route Mode on the the Overlap map

Bibliography

1. Ablavsky, V. and M. Snorrason. "Optimal Search for a Moving Target: A Geometric Approach." *AIAA Guidance, Navigation, and Control Conference and Exhibit*. August 2000.
2. "Air Force Research Laboratory, Information Directorate Mission Statement."
3. Arkin, Ronald C. *Behavior Based Robotics*. MIT Press, 1998.
4. Bäck, T. *Self-Adaptation*, chapter 7. Oxford University Press, 1997.
5. Bäck, T., et al. *Evolutionary Computation 1*. Institute of Physics Publishing, 2000.
6. Bäck, T., et al. "A survey of evolution strategies." *Proceedings of the 4th International Conference on Genetic Algorithms*, edited by Lashon B. Belew and Richard K. Booker. 2–9. Morgan Kaufmann, July 1991.
7. Bäck, Thomas A. *Evolutionary Algorithms in Theory and Practice*. New York - Oxford: Oxford University Press, 1996.
8. Balch, T. and R. C. Arkin. "Behavior-based Formation Control for Multiagent Robot Teams," *IEEE Transactions on Robotics and Automation* (December 1998).
9. Banda, Siva S., "Future Directions in Control for Unmanned Aerial Vehicles." Slides, April 2002.
10. Barraquand, J., et al. "Robot Motion Planning: A Distributed Representation Approach." *International Journal of Robotics Research* 10. 628–649. 1991.
11. Bayazit, O Burchan, et al. *Better Flocking behaviors in Complex Environments Using Global Roadmaps*. Technical Report TR02-003, Texas A&M University, May 2002.
12. Beni, Gerardo and Jing Wang. "Swarm Intelligence in Cellular Robotic Systems." *Proceedings of the NATO Advanced Workshop on Robotics and Biological Systems*. 1989.
13. Benkmann, Matthias S., "Motion Planning Using Random Networks."
14. Billard, Aude and Maja J Mataric. "A Biologically Inspired Robotic Model for Learning by Imitation." *Proceedings, Autonomous Agents 2000*. 3–7. June 2000.
15. Bonabeau, Eric, et al. *Swarm Intelligence: From Natural to Artificial Systems*. Santa Fe Studies in the Sciences of Complexity, Oxford University Press, 1999.
16. Bourg, David M. *Physics for Game Developers*. O'Reilly, January 2002.
17. Burges, C. "A Tutorial on Support Vector Machines for Pattern Recognition." *Data Mining and Knowledge Discovery*. 121–167. 1998.
18. Caselli, A., et al. "Parallel Path Planning with Multiple Evasion Strategies." *IEEE International Conference on Robotics and Automation, ICRA 2002*. 260–266. 2002.
19. Chandler, Phillip and Meir Pachter. "Heirarchical Control for Autonomous Teams." *AIAA Guidance, Navigation, and Control Conference and Exhibit*. August 2001.

20. Chandler, Phillip, et al. "Multiple Task Assignment for a UAV Team." *AIAA Guidance, Navigation, and Control Conference and Exhibit*. August 2002.
21. Christofides, Nicos. *Graph Theory: An Algorithmic Approach*. Academic Press, 1975.
22. Costa, João, et al., "Java Distributed Evolutionary Algorithm Library."
23. "Micro Air Vehicle Sets Endurance Flight Record." News Release, October 2002.
24. *RQ-1 Predator Unmanned Aerial Vehicle*. Fact Sheet, United States Air Force, May 2002.
25. *Electronic Warfare and Radar Systems Engineering Handbook*. Manual, Naval Air Warfare Weapons Division, April 1997.
26. Dudek, Gregory, et al. *A Taxonomy for Multi-Agent Robotics*, chapter 1, 3–22. A. K. Peters Ltd, 2002.
27. Fax, J. Alexander and Richard M. Murray. "Information Flow and Cooperative Control of Vehicle Formations." *Proceedings of the 2002 IFAC World Congress*. 2002.
28. Fierro, Rafael, et al. "Cooperative Control of Robot Formations." *Cooperative Control and Optimization 66*. Applied Optimization, edited by R. Murphy and P. Pardalos, chapter 5, 73–93, Kluwer Academic Press, 2002.
29. Flynn, M. "Some Computer Organizations and Their Effectiveness." *IEEE Transactions on Computing C-21*. 94. 1972.
30. Ghavamzadeh, Mohammad and Sridhar Mahadevan. "Continuous-time Hierarchical Reinforcement Learning." *Eighteenth International Conference on Machine Learning (ICML)*. June 2001.
31. Gillen, Daniel P. and David R. Jacques. "Cooperative Behavior Schemes for Improving the Effectiveness of Autonomous Wide Area Search Munitions." *Proceedings of the Cooperative Control Workshop*. December 2000.
32. Gillen, Daniel P. and David R. Jacques. "Cooperative Behavior Schemes for Improving the Effectiveness of Autonomous Wide Area Search Munitions." *69th Military Operations Research Society Symposium*. July 2001.
33. Grant, Rebecca. "The Bekaa Valley War," *Air Force Magazine*, 58–62 (June 2002).
34. Hayes, A., et al., "Comparing distributed exploration strategies with simulated and real autonomous robots," 2000.
35. Hebert, Jeffrey, et al. "Cooperative Control of UAVs." *AIAA Guidance, Navigation, and Control Conference and Exhibit*. August 2001.
36. Hertz, John, et al. *Introduction to the Theory of Neural Computing*. Addison-Wesley, 1991.
37. Jadbabaie, A., et al. "Coordination of Groups of Mobile Autonomous Agents Using Nearest Neighbor Rules." Center for Computational Vision Control, Yale University, 2002.

38. Kadvach, B. Anthony. *Doctoral Dissertation*. PhD dissertation, Air Force Institute of Technology, 2003.
39. Kadvach, B. Anthony and Gary B. Lamont. "A Particle System for Swarm-based Networked Sensor Systems." *Symposium on Applied Computing*. 2002.
40. Kavraki, L. E. and J.-C. Latombe. "Randomized Preprocessing of Configuration Space for Fast Path Planning." *IEEE International Conference on Robotics and Automation*. 2138–2145. 1994.
41. Kiehn, R. M. *Holonomic and Anholonomic Constraints and Coordinates, Frobenius Integrability and Torsion of Various Types*. Technical Report, University of Houston, 2002.
42. Koumoutsakos, P., et al. "Evolution Strategies for Parameter Optimization in Jet Flow Control." *Proceeding of the Summer Program 1998*. Center for Turbulence Research, 1998.
43. Krock, Lexi, "Timeline of UAVs." URL: www.pbs.org/wgbh/nova/spiesfly/uavs.html.
44. Latombe, Jean-Claude. *Robot Motion Planning*. The Kluwer International Series in Engineering and Computer Science, Kluwer Academic Publishers, 1991.
45. LaValle, Steven M. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*. Technical Report 98-11, Iowa State University, October 1998.
46. Liu, Yang, et al. "Stability Analysis of One-dimensional Asynchronous Swarms." *Proceedings of the 2001 American Control Conference*. 716–721. June 2001.
47. Liu, Yang, et al. "Stability Analysis of One-dimensional Asynchronous Mobile Swarms." *Proceedings of the 40th IEEE Conference on Decision Control*. 1077–1082. Dec 2001.
48. Liu, Yang, et al. "Stability Analysis of M-dimensional Asynchronous Swarms with a Fixed Communication Topology." *IEEE Transactions on Automatic Control* 48. 76–95. Jan 2003.
49. Makar, Rajbala, et al. "Hierarchical Multi-Agent Reinforcement Learning." *Fifth International Conference on Autonomous Agents*. 2001.
50. Mataric, Maja J. "Reward Functions for Accelerated Learning." *Machine Learning: Proceedings of the Eleventh International Conference*, edited by William W. Cohen and Haym Hirsh. 181–189. Morgan Kaufmann Publishers, 1994.
51. Mataric, Maja J. "Designing and Understanding Adaptive Group Behavior." *Adaptive Behavior* 4. 51–80. December 1995.
52. Mataric, Maja J. "Issues and Approaches in the Design of Collective Autonomous Agents." *Robotics and Autonomous Systems* 16. 321–331. Dec 1995.
53. Mataric, Maja J. "Behavior-Based Control: Examples from Navigation, Learning, and Group Behavior." *Journal of Experimental and Theoretical Artificial Intelligence, special issue on Software Architectures for Physical Agents* 9, edited by H. Hexmoor, et al. 323–336. 1997.

54. Mataric, Maja J. "Learning Social Behavior." *Robotics and Autonomous Systems* 20. 191–204. 1997.
55. Mataric, Maja J. "Reinforcement Learning in the Multi-Robot Domain." *Autonomous Robots* 4. 73–83. March 1997.
56. McKenna, TSgt Pat. "Eyes of the Warrior," *Airman Magazine* (July 1998).
57. McLain, Timothy W., "Coordinated Control of Unmanned Air Vehicles," 1999.
58. McPherson, Col Craig, "Global Hawk System Program Overview." Slides, January 2000.
59. Michalewicz, Zbigniew and David B. Fogel. *How to Solve It: Modern Heuristics*. New York: Springer-Verlag, 2000.
60. Milton, J.S. and Jesse C. Arnold. *Introduction to Probability and Statistics: Principles and Applications for Engineering and the Computing Sciences* (3 Edition). McGraw-Hill Series in Probability and Statistics, McGraw Hill, 1995.
61. Mueller, S. D., et al. "Evolution Strategies for the Optimization of Microdevices." *Proceedings of the Congress on Evolutionary Computation (CEC 2001)*. 302–309. 2001.
62. on Armed Services, Senate Committee. *National Defense Authorization Act for Fiscal Year 2001*. Bill, United States Senate, 12 May 2000.
63. Parker, Lynn. "Designing Control Laws for Cooperative Agent Teams." *Transactions of the IEEE on Robotics and Automation*. 582–587. 1993.
64. Parker, Lynn. "Current State of the Art in Distributed Autonomous Mobile Robotics." *Distributed Autonomous Robotic Systems* 4, edited by L. E. Parker, et al. 3–12. Springer-Verlag, 2000.
65. Parunak, H. Van Dyke and Raymond S. VanderBok. "Managing Emergent Behavior in Distributed Control Systems." *ISA-Tech '97*. 1997.
66. Pike, John, "RQ-4A Global Hawk (Tier II+ HAE UAV)."
67. Polycarpou, Marios, et al. *Cooperative Control: Models, Applications and Algorithms*, chapter Cooperative Control Design for Uninhabited Air Vehicles. Kluwer Academic Publishers, 2003. to appear.
68. Polycarpou, Marios M., et al., "Cooperative Control of Distributed Multi-Agent Systems." submitted June 2001, revised July 2001.
69. Polycarpou, Marios M., et al. "A Cooperative Search Framework for Distributed Agents." *Proceedings of the 2001 IEEE International Symposium on Intelligent Control*. 16. 2001.
70. Ramírez, J. Frederico and Olac Fuentes. "Spectral Analysis Using Evolution Strategies." *IASTED International Conference on Artificial Intelligence and Soft Computing*. July 2002.

71. Reif, J. and S. R. Tate. "The Complexity of N-body Simulation." *Proceedings of the Twentieth annual Colloquium on Automata, Languages and Programming*. 162–176. July 1993.
72. Reif, J.H. and H. Wang. "Social Potential Fields: A Distributed Behavioral Control for Autonomous Robots." *Workshop on Algorithmic Foundations of Robotics (WAFR'94)*. February 1994.
73. Reynolds, Craig W. "Flocks, Herds, and Schools: A Distributed Behavioral Model." *Computer Graphics*21. 25–34. July 1987.
74. Reynolds, Craig W. "Steering Behaviors for Autonomous Characters." *need title*. need address: need publisher, 1999.
75. Richards, Arthur, et al., "Co-ordination and Control of Multiple UAVs," 2002.
76. Schwefel, H. P. "Numerische Optimierung von Computer-Modellen mitele der Evolutionenstartegie," 26 (1977).
77. Schwefel, H. P. "Collective Pheonomena in Evoluationary Systems." *Preprints of the 31st Annual Meeting of the International Society for General System Research*2. 1025–1033. June 1987.
78. Shapiro, et al. "A Comparative Study of Various Tests for Normality," *Journal of American Statistical Association*, 63:1343–1372 (1968).
79. Society, IEEE Computer. *IEEE Standard for Binary Floating-Point Arithmetic*. IEEE Standard 754-1985, IEEE, 1985.
80. Steels, L. "The Artificial Life Roots of Artificial Intelligence." *Artificial Life*1. 75–110. 1994.
81. Terzopoulos, Demetri, et al. "Artificial Fishes with Autonomous Locomotion, Perception, Behavior, and Learning in a Simulated Physical World." *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*. 17–27. July 1994.
82. Trahan, Michael W., et al. "Swarms of UAVs and Fighter Aircraft." *Proceedings of the Second International Conference on Nonlinear Problems in Aviation and Aerospace*2. 745–752. 1998.
83. Trianni, V., et al. *Modeling Pattern Formation in a Swarm of Self-Assembling Robots*. Technical Report TR/IRIDIA/2002-12, IRIDIA, May 2002.
84. Tu, Xiaoyuan and Demetri Terzopoulos. "Artificial fishes: Physics, locomotion, perception, behavior." *Computer Graphics*28. Annual Conference Series. 4350. 1994.
85. Usher, John M. and Yi-Chi Wang. "Intelligent Agents Architectures for Manufacturing Control." *Industrial Engineering Research Conference 2000*. May 2000.
86. Vadakkepat, Prahlad, et al. "Evolutionary artificial potential fields and their application in real time robot path planning." *Congress of Evolutionary Computation (CEC2000)*. 256–263. July 2000.

87. Wegerbauer, Cyndi, "RQ-1 Predator Hellfire Missile Tests 'Totally Successfull'." News Release, Febuary 2001.
88. Zaera, Nahum, et al. "(not) Evolving Collective Behaviours in Synthetic Fish." *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior (SAB96)*. 635644. MIT Press Bradford Books, 1996.

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.					
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) 25-03-2003		2. REPORT TYPE Master's Thesis		3. DATES COVERED (From - To) Aug 2002 - Mar 2003	
4. TITLE AND SUBTITLE DISTRIBUTED CONTROL OF A SWARM OF AUTONOMOUS UNMANNED AERIAL VEHICLES				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
				5d. PROJECT NUMBER	
6. AUTHOR(S) Lotspeich, James, T., 1st Lieutenant, USAF				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way, Building 640 WPAFB OH 45433-7765				8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GCS/ENG/03-10	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFMC AFRL/IFTA Attn: Mr. Robert Ewing 2241 Avionics Circle WPAFB OH 45433-7334 DSN 785-6653 x3592 robert.ewing@wpafb.af.mil				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT With the increasing use of Unmanned Aerial Vehicles (UAV)s in military operations, there is a growing need to develop new methods of control and navigation for these vehicles. This investigation proposes the use of an adaptive swarming algorithm that utilizes local state information to influence the overall behavior of each individual agent in the swarm based upon the agent's current position in the battlespace. In order to investigate the ability of this algorithm to control UAVs in a cooperative manner, a swarm architecture is developed that allows for on-line modification of basic rules. Adaptation is achieved by using a set of behavior coefficients that define the weight at which each of four basic rules is asserted in an individual based upon local state information. An Evolutionary Strategy (ES) is employed to create initial matrices of behavior coefficients. Using this technique, three distinct emergent swarm behaviors are evolved, and each behavior is investigated in terms of the ability of the adaptive swarming algorithm to achieve the desired emergent behavior by modifying the simple rules of each agent. Finally, each of the three behaviors is analyzed visually using a graphical representation of the simulation, and numerically, using a set of metrics developed for this investigation.					
15. SUBJECT TERMS Cooperative Behavior, Multi-Agent System, Autonomous Control, Cooperative Agents, Swarming, Swarm, Evolutionary Computation, Evolution Strategy					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT UU	18. NUMBER OF PAGES 185	19a. NAME OF RESPONSIBLE PERSON Dr. Gary Lamont, ENG
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U			19b. TELEPHONE NUMBER (Include area code) (937)255-3636 x4718; gary.lamont@afit.edu