3-3-2008

A Potential Field Based Formation Control Methodology for Robot Swarms

Laura E. Barnes
University of South Florida

A Potential Field Based Formation Control Methodology for Robot Swarms

by

Laura E. Barnes

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Major Professor: Kimon Valavanis, Ph.D.
Lawrence Hall, Ph.D.
Rangachar Kasturi, Ph.D.
Rafael Perez, Ph.D.
Stephen Wilkerson, Ph.D.
Maryanne Fields, Ph.D.

Date of Approval:
March 3, 2008

Keywords: swarm formation, mobile robots, robot control, multirobot systems, intelligent systems

Note to Reader

The original of this document contains color that is necessary for understanding the data. The original dissertation is on file with the USF library in Tampa, Florida.

Dedication

This work is dedicated to my family and friends without whom this work would not have been possible. I would like to specifically thank my parents for their ongoing support. I would also like to thank Richard Garcia for his constant friendship and technical assistance throughout our journey the past years. Last but not least, I would like to thank Sakis for his love and encouragement.

Acknowledgments

This research was supported in part by an appointment to Student Research Participation Program at U.S. Army Research Laboratory administered by the Oak Ridge Institute for Science and Education through interagency agreement between the U.S. Department of Energy and US ARL. This work was also supported partially by two grants ARO W911NF-06-1-0069 and SPAWAR N00039-06-C-0062. I would specifically like to thank Dr. MaryAnne Fields from the Army Research Lab for all her time and help.

Table of Contents

List of Tables

List of Figures

A Potential Field Based Formation Control Methodology for Robot Swarms

Laura E. Barnes

ABSTRACT

A novel methodology is presented for organizing swarms of robots into a formation utilizing artificial potential fields generated from normal and sigmoid functions. These functions construct the surface which swarm members travel on, controlling the overall swarm geometry and the individual member spacing. Nonlinear limiting functions are defined to provide tighter swarm control by modifying and adjusting a set of control variables forcing the swarm to behave according to set constraints, formation and member spacing. The swarm function and limiting functions are combined to control swarm formation, orientation, and swarm movement as a whole. Parameters are chosen based on desired formation as well as user defined constraints. This approach compared to others, is simple, computationally efficient, scales well to different swarm sizes, to heterogeneous systems, and to both centralized and decentralized swarm models. Simulation results are presented for a swarm of four and ten particles following circle, ellipse and wedge formations. Experimental results are also included with a swarm of four unmanned ground vehicles (UGV) as well as UGV swarm and unmanned aerial vehicle (UAV) coordination.

Chapter 1

Introduction

1.1 Motivation

Multirobot and swarm systems are of mounting importance as robotics research progresses. Swarms and other multi-agent systems have been suggested as a means to accomplish tasks on the battlefield and in other environments. Applications of swarm intelligence are clear when tasks are well defined and redundancy among swarm members is the most important factor. Swarm robots are generally equipped with few sensors, limiting the tasks they can perform alone. Utilizing a group of robots to complete a task is sometimes necessary. These systems have been used for a variety of tasks including Robocup soccer, containing spills, finding and neutralizing mines, foraging, reconnaissance, and surveillance.

The need for formation control of multi-robot systems performing a coordinated task has lead to the development of a challenging research field. The formation control problem is defined as finding a control algorithm ensuring that multiple autonomous vehicles can uphold a specific formation or specific set of formations while traversing a trajectory and avoiding collisions simultaneously. Formation control of the group as well as the dynamic response and reconfiguration of a group in varying environments is of particular interest. In this thesis, the concern is to develop a simple control algorithm, which will allow multiple robots to traverse a trajectory in formation while avoiding collisions.

The motivating area for the proposed approach is convoy protection where a swarm of robots assigned to protect a convoy of vehicles requires a specific formation around the vehicles. Thus, given a convoy of assets to protect, determining the configuration of the swarm and the arrangement of the support vehicles around the convoy poses a particularly interesting problem.

1.2 Problem Statement

This research focuses on *the ability to modify formation of a heterogeneous swarm given a set of parameters and limiting functions*. The swarm should be able to recover and reconfigure itself in the event of loss of a team member, loss of configuration or the addition of a new team member. It is also imperative that the formation be reconfigurable based on dynamically changing parameters. The formation control problem is defined as *finding a control algorithm to ensure that multiple autonomous vehicles can maintain a specific formation or specific set of formations while traversing a trajectory and avoiding collisions simultaneously*.

1.3 Method of Approach

The main contribution is the proposed swarm formation control method that is based on troop movement models [1, 2] satisfying scalability (to varying numbers of swarm members) and supporting multiple formations. The central objective is *the overall group formation and behavior, not control of an individual swarm member*.

The proposed solution is based on potential fields generated by bivariate normal functions that are used to control swarm geometry and inter-member spacing, as well as manage obstacle avoidance. The rationale behind using potential fields for formation control is that they facilitate, in a simple and straightforward way, the representation of multiple constraints and goals in a swarm system. Repulsive and attractive forces are utilized to control the overall swarm formation. Bivariate normal functions are used to create the vector fields that control the velocity and heading of the robot swarms, with the swarm located on the mathematical surface. Potential fields are not only used in the classical sense for path planning, but also to hold robots in a specified formation while in motion. Fields generate desired velocities and headings to maintain a path and to dynamically avoid obstacles while maintaining a formation. Although potential field methods do face the local minima problem [3], the vectors are dynamically changing at each time step so the chances of hitting local minima are greatly reduced.

1.4 Research Contributions

The contribution of this research is a fault-tolerant, scalable swarm formation control methodology. This approach is able to support loose ellipse-based configurations and is highly

flexible in varying environments. The advantage and difference of this approach compared to similar ones is the simplicity of the vector generation. Control parameters are easily modified and adjusted to change formation and relative distance between swarm members, while other methods are rigid in formation constraints [4]. Further, in the presented approach, since control parameters may be tuned off-line and used on-line, changing formation is computationally inexpensive (compared to, for example, the approach in [5] that suffers from computational complexity).

The proposed method scales well to different swarm sizes as well as to heterogeneous swarms because the vector field generation is independent of the specific robot vehicle platform, (as opposed to approaches in [6, 7] that are scalable in size but not in heterogeneity). The robot vehicle controller receives the generated vector as input and translates it to the corresponding motion.

This presented method does not add communication overhead because a central controller is not a necessity. Depending on the mission, robots may intercommunicate on an as needed basis. Although the scope of this work is not on communication issues but on actual control and manipulation of swarm formations, a simple broadcast communication model is implemented in which robots only exchange information when predefined points are reached.

1.5 Summary of Results

The presented method is demonstrated by simulation using both robot vehicles modeled after an RC-car with Ackerman steering and a simple particle model. A model is derived and simulations using up to ten (heterogeneous) such vehicles are presented, including two formations to demonstrate applicability and performance of the approach. Simulations are performed with up to ten robots, and real-time experiments are run with up to four custom built UGVs. Ellipse, circle, line, and arc formations are demonstrated. The real-world experiments are presented with either three or four autonomous UGVs. In addition, autonomous UAV-UGV swarm coordination is demonstrated. Dynamic formation change is demonstrated as well as continued performance with loss of team members. The results demonstrate that the proposed method can be successfully utilized to control robot swarm formation.

1.6 Thesis Outline

This work consists of eleven chapters each explaining the swarm formation control methodology used. Chapter 2 provides the necessary background information from mobile robot systems to swarm formation control. Chapter 3 provides the literature review. The most relevant formation control methods are discussed and a comparative study is given.

Chapter 4 presents the problem formulation and details how the vector fields are generated to achieve formation. In Chapter 5, the simulation system and methods are discussed. In Chapter 6, the robot hardware platform and sensors are described. In Chapter 7, the software system is discussed. In Chapter 8, the simulation results on simulated and particle robots are presented. In Chapter 9, field experiments with the actual UGVs are presented. In Chapter 10, the approach to formation control is expanded utilizing deformable ellipses to achieve more formations. Finally, Chapter 11 concludes this research and discusses future work.

Chapter 2

Background Information

In order to provide the necessary background knowledge for this research, a brief description of mobile robots is given Section 2.1. Multirobot systems and multirobot system architectures are described in Section 2.2 and 2.3, respectively, and finally robot swarms and swarm formation control are described in Section 2.4 and Section 2.5.

2.1 Mobile Robot Systems

A robot, in the loosest sense, may be defined as a physical agent who conveys by its appearance and motion that it has some level of autonomy. An autonomous robot is one that can perform a specific set of tasks without human supervision as opposed to a teleoperated robot which requires constant human supervision. A robot that has full autonomous capabilities will embody some level of artificial intelligence so the robot can 'choose' the right actions and in some cases adapt and/or learn from its changing environment or its own choices.

For the purpose of this work, the concern is in mobile robotics. Robots are used for civilian and military applications, especially those that are dull, dirty or dangerous. They can come in many shapes and sizes and can be classified into 3 groups: unmanned ground vehicles (UGVs); unmanned air vehicles (UAVs); and unmanned underwater vehicles (UUVs). Figure 2.1a and Figure 2.1b are examples of robots used in military applications. Figure 2.1a depicts the Army Research Lab (ARL) UGV Lynchbot which is used for detection of improvised explosive devices (IEDs) in war zones. In Figure 2.1b, the ARL UAV fixed wing drone target is shown. Figure 2.1c is an example of a domestic robot used for autonomous vacuuming, and Figure 2.1d is the Mitsubishi service robot, Wakamaru, which can provide services such as care-taking, managing schedules, and reporting unusual activities.

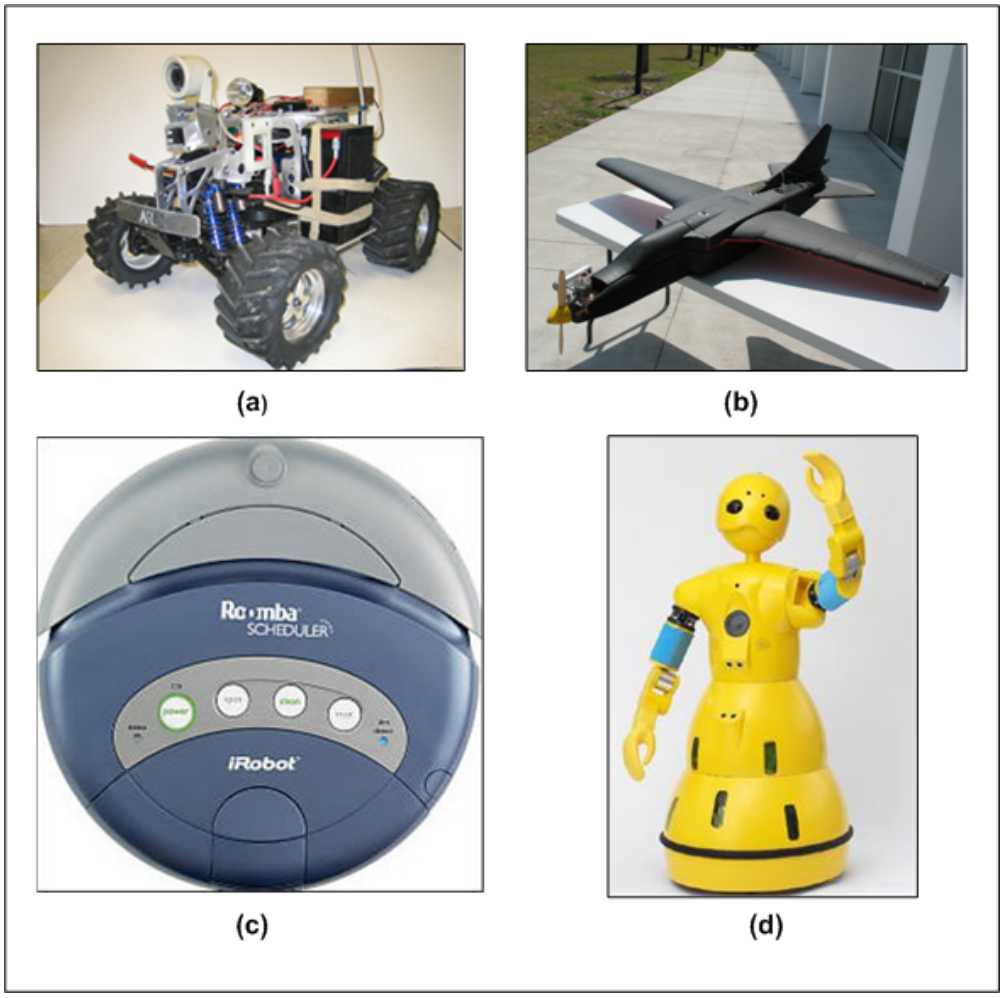

Figure 2.1. Examples of robots. (a) Army Research Lab (ARL) Lynchbot; (b) U.S. Army fixed wing target drone; (c) iRobot Roomba robotic vacuum cleaner; (d) Mitsubishi service robot.

Single mobile robot systems have been used for security [8] , medicine [9], and domestic tasks [10] [11], but in some cases a single robot is not sufficient. For example, in the case of planet exploration or pushing objects [12], the use of a single robot would be both unrealistic and inefficient. For these tasks, several mobile robots can be utilized to accomplish a task that would otherwise be very difficult or impossible for a single robot.

2.2 Multirobot Systems

Multirobot systems are of mounting importance as robotics research progresses. Many new systems and experimental platforms have been developed. Recently, several large scale multirobot systems have appeared in the literature. Multirobot systems are of interest for tasks that are inherently too complex for a single robot or simply because using several simple robots can be better than having a single powerful robot for each task [13].

Groups of mobile robots are used for studying issues such as group behavior, resource conflict, and distributed learning. These systems have been used for a variety of tasks including but not limited to Robocup soccer [14]; search and rescue [15]; terrain coverage [16]; foraging [17]; and cooperative manipulation [12]. Since these application domains and tasks are of increasing complexity, the ability of the robots to cooperate and coordinate is sometimes necessary. The coordination of multiple robot systems remains a challenging problem.

A multirobot system is any robotic system of two or more robots. These robots may be identical or they may contain a multitude of varying systems ranging from slightly different sensors to entirely distinct hardware and/or software platforms.

Multirobot systems can further be classified into two groups shown in Figure 2.2: (i) cooperative robot teams and (ii) swarms of robots [18].

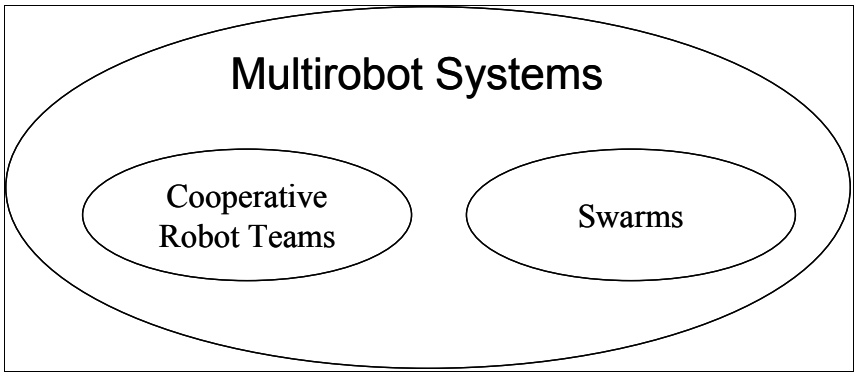

Figure 2.2. Multirobot system groups.

In the first group, cooperative robot teams, each robot generally has different capabilities and control algorithms which when combined can be used to complete a task. In a swarm of robots each robot generally has identical function and capabilities with the goal being the overall group behavior. The focus here is on swarms of robots including UAV-UGV swarm coordination.

2.3 Multirobot Architectures

In order to give the full background necessary, a brief discussion of group architectures in relation to multirobot systems is necessary. The group architecture of a multirobot system provides the framework upon which missions are implemented, and determines the system functionality and boundaries. The key architectural aspects of a group system include: (i) type of control, (ii) team composition, and (iii) communication structures.

The most basic decision that is made when designing group architectures is defining the type of control the system will utilize. Group control techniques of autonomous robots include: i) centralized-control in which individuals receive commands from a central controller and ii) decentralized control where local control laws operating in individual robots produce a desired global, *emergent behavior*.

In centralized control methods [19-24], a single computational unit oversees the whole group and plans the motion control of the group accordingly. Figure 2.3 shows a simplified model of a centralized architecture with the block in the middle being the main control block, all robots communicating through this block. This central control unit might be an external computer or it might be one of the robots in the group. In centralized control methods, the entire multirobot system is dependent on one controller, so these methods are not very tolerant to failure.

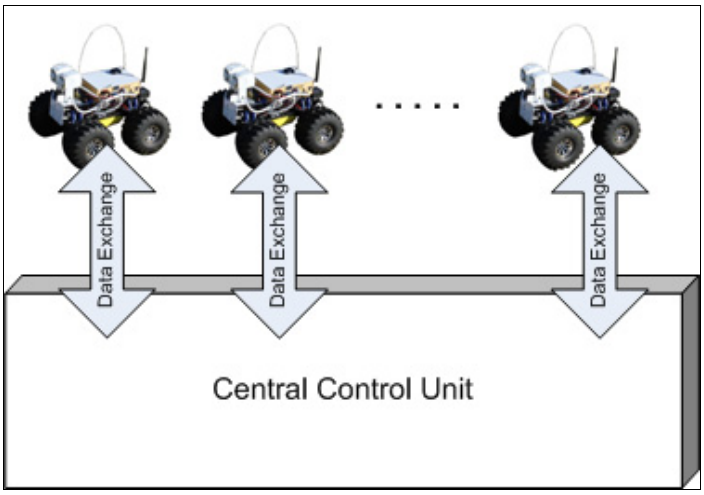

Figure 2.3. Centralized control model.

Decentralized control methods [25-35] lack a central control unit and follow two forms: (i) distributed control and (ii) hierarchical control. In distributed control, all robots are equal with

respect to control; in the hierarchical method, control is locally centralized. In distributed, decentralized control methods, the desired behavior is produced using only local control laws operating on individual robot members. These local control laws depend on the specific model and methodology used. A key feature of centralized control is that each of the robot members communicates directly with each other as shown in Figure 2.4. Decentralized control methods are advantageous over centralized ones in that they are more fault tolerant, scalable, and reliable.

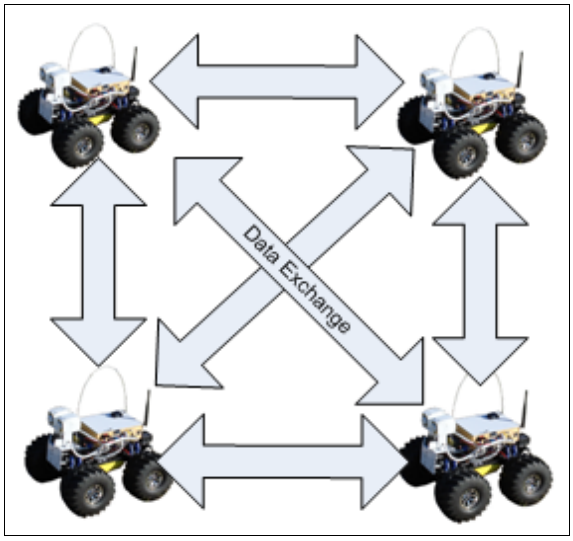

Figure 2.4. Distributed, decentralized control model.

Conventionally, most systems do not conform exactly to either a centralized or decentralized architecture. Instead, the emergent architecture tends to be a hybrid of the two, utilizing, for example, a hierarchical leader structure or even virtual leaders.

In addition to the type of control used in a multirobot system, the team composition and size of the system is also important. A multirobot system may either be made up of homogeneous or heterogeneous units. A group of robots is said to be homogeneous if the capabilities of the individual robots are identical and heterogeneous otherwise. Any difference in software or hardware can make a robot different from another. Higher levels of heterogeneity introduce more complexity in task allocation since robots have different capabilities. This requires agents to have a greater knowledge about each other. In a heterogeneous system, it is necessary to prioritize a robot's tasks based on its capabilities, whereas in a completely homogeneous system all robots have equal capabilities and priorities. The team size can either hinder or help depending on the task and team composition.

Within a multirobot system, robots may communicate following several information structures, including: (i) interaction via the environment, (ii) interaction via sensing, and (iii) interaction via communications [13]. Interaction by means of the environment involves using the surrounding features as the communication medium. An example of interaction via the environment would be some type of landmark navigation [36]. Interaction via sensing involves using sensory data such as range measurements to sense other robots. Finally, the third form of interaction is explicit communication through either directed or broadcast messages. This communicated information could be any necessary data such as position information or images necessary to the mission.

All of these architectural aspects will determine the abilities of both the robots and the entire multirobot system. The type of control and level of communication in the system will determine how cohesive and coupled the system is, and how dependent each member is on the other team members. Only in a fully distributed system can you have the lowest level of agent dependency and highest level of fault tolerance.

2.4 Robot Swarms

The swarm-type approach to multirobot cooperation traditionally deals with large numbers of homogeneous robots with limited sensors. Swarm robot systems have been defined to include ten or more robots, but this bound is relaxed in this work to include any system with two or more robots. Swarms are usually made up of robots with similar controllers, and instead of a centralized controller, they employ one or more supervisors or leaders. Further, swarm control systems are usually focused around decentralized control methods which make them more robust to loss of team members. For the purpose of this work, the definition of a swarm is relaxed and extended to include *any group of robots, homogeneous or heterogeneous, in which the goal is some global emergent group behavior*.

Research on robotic swarms [37] typically involves mimicking nature [38], [39] and observing animal groups accomplishing tasks an individual cannot accomplish alone [40], [41]. Figure 2.5 represents examples of swarms in nature, a school of fish, a flock of birds and a swarm of bees.

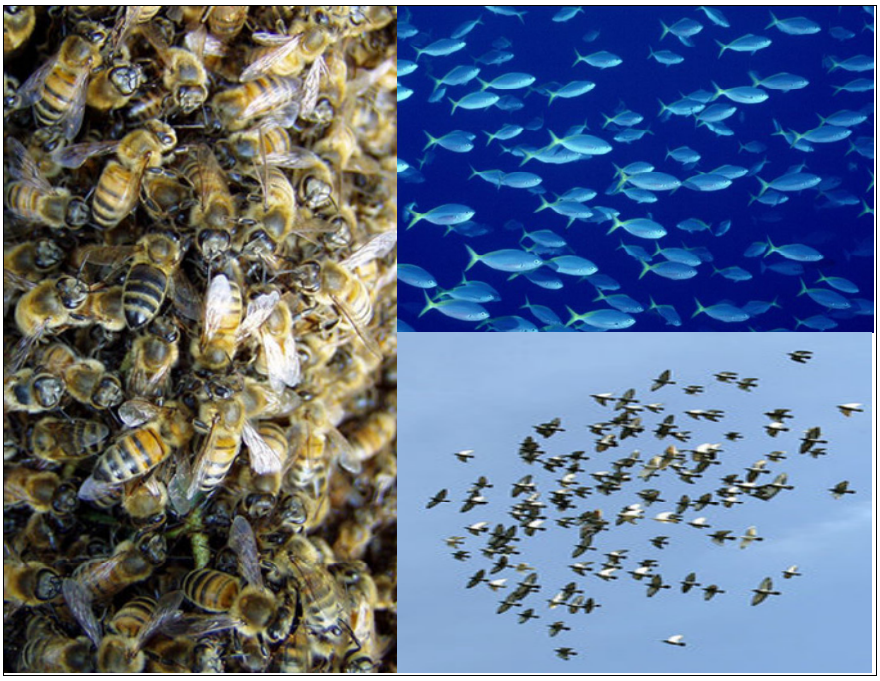

Figure 2.5. Examples of swarms in nature.

When studying swarms of robots, many factors must be considered [37] including: (i) swarm composition and size; (ii) communication issues; and (iii) swarm reconfigurability.

When a swarm system is designed, swarm composition and size must be taken into account. Both the heterogeneity of the swarm system and the number of robots that will be in the environment are important issues which are dependent on the task and operating environment. A larger swarm is not always better and could actually hinder task achievement, and vice versa. Determining the optimal number of swarm members for a particular task or mission is a problem by itself.

Communication issues are a huge concern especially as the number of robots in the swarm increases ($N \rightarrow \infty$). Communication should be used conservatively. The idea is to use a minimal amount communication given the mission. The robots do not necessarily have to obtain all information through explicit communication, because it is possible robots to obtain information about the swarm via other sources of information such as sensory data or the environment. The following aspects of communication must be considered in swarm communication design:

- The *communication range* of the system is finite. In a swarm of robots, the range of direct communication to any single robot is limited. Both the communication medium and the spatial distribution of the swarm will affect the communication range.

- The *communication topology* of the system will determine how robots can communicate with each other regardless of their proximity. Depending on the communication topology chosen robots may not be able to communicate directly with any arbitrary member of the swarm; or they may be able to communicate directly with any member of the swarm.
- The *communication bandwidth* of the system is also finite. Communication can be cheap in terms of a robot's processing time, if there is a dedicated channel, or it could be computationally expensive keeping the robot from doing other necessary work.

Swarm reconfigurability is the rate at which the swarm is able to spatially reorganize itself. Depending on the method, the swarm can have a static arrangement or a dynamic arrangement. How members move among each other and spatially distribute is a problem alone and is the main focus of this work. This aspect will further be discussed in next section.

2.5 Swarm Formation Control

The problem of formation control of multi-robot systems performing a coordinated task has become a challenging research field. The pattern formation of a multirobot system is the organization of the robots into a particular shape such as a wedge or circle. In addition to a specific geometric pattern formation, an approximate or loose geometric pattern is referred to flocking. In this work, the focus is both on specific and loose geometric formations. The shape or formation of the swarms is highly task and environment dependent. Current applications of pattern formation include convoy support [42-45], chemical source localization [46], and unmanned aerial vehicles [20, 47, 48].

Pattern formation can be observed directly in biological systems. Flocks of birds, schools of fish, and swarms of bees form complex dynamical spatial patterns that emerge when each animal in the group follows a simple set of rules exhibiting emergent swarm behaviors [49]. Ants are a particularly good example of emergent swarm behaviors. In coordination between groups of ants, the control is not hierarchical or central, but the ants coordinate locally to achieve colony level goals [49].

Various approaches to ground-based formation control are behavior-based control [25, 50, 51], graph theoretic [52, 53], algorithmic [35] and virtual structures [54]. In Chapter 3, a detailed survey and comparative study of formation control and flocking methodologies are presented.

The particular type of methodology used in formation control is highly dependent on many factors including [55]:

- The formation *stability* or *accuracy* of the system will determine if a very tight formation control methodology or a looser formation control method will suffice. The stability of a formation refers to the propagation of errors through the system. In some applications, it is necessary to have a very low error bound so a very tight, stable formation control method is necessary.
- The *scalability* of a formation refers to a swarm's ability to reconfigure in the event of removal or addition of a swarm member. In anonymous robot swarms where specific robots are not distinguishable, scalability is much better. In swarms utilizing a leader-follower based methodology, the positions of the added robots are in relative to one or more other members. In the case of robot removal, reorganization needs to occur. This type of system needs to be very well designed to tackle the scalability problem.
- The formation control task can be divided into *establishing a formation* and *maintaining a formation*. While the task of establishing the initial formation may be trivial, the act of maintaining a formation while traversing a territory or performing a particular mission is a large challenge.
- The ability to *dynamically modify a formation* from one shape or another for any reason such as obstacle avoidance increases the difficulty of formation control. In the event of anonymous swarms, this task is much simpler than in the leader-follower formation structures.

All of these factors will contribute to the design of the swarm architecture and formation control mechanism. The operating environment and types of missions will factor in when making design decisions.

Chapter 3

Literature Review

In this section, current formation control methodologies will be evaluated and compared with the proposed approach. A comparative study on existing multirobot and swarm formation methodologies is presented. Formations discussed and compared include both specific geometric formations and flocking formations. Only multirobot systems focusing on formation control are included in this comparison. The formation control strategies are analyzed from their control methods and shape representations.

The most fundamental and key aspect of formation control is the method of control used in the multirobot or swarm system. The control method follows three different types, including (i) centralized, (ii) completely decentralized, and (iii) hybrid. The majority of the approaches are hybrid. The robots need to maintain a specific *formation shape* while maintaining the correct formation position among other robots. This formation position is determined based on the particular reference method used. Robotic control is used to get to the correct formation position. In Section 3.1 formation control and shape representations are discussed. Section 3.2 discusses the foundation and basis for the approach in this research, and finally Section 3.2 summarizes the literature review.

3.1 Formation Control Methods

In this section different control and shape representation methods are surveyed. In formation control for a group of robots, different control topologies can be adopted depending on the specific scenarios and/or missions. There may be one or more leaders in the group, while the other robots follow one or more leaders in a specified way. Each robot has onboard sensing and computation ability. In some applications, robots only have limited communication ability. In general, global knowledge about the system is not available to each robot. A centralized

controller is not utilized, and in this case the design of each robot controller has to be based on local information. If there is no assigned leader, then each robot must coordinate with the others by relying on some global consensus to achieve the common goal.

Various types of shapes have been employed in formation control. The specific shape might be scenario or mission dependent. The more common formation shapes are column, line, wedge, triangle, and circle. Some of these shapes are shown in Figure 3.1 where individual robots are represented by the circles.

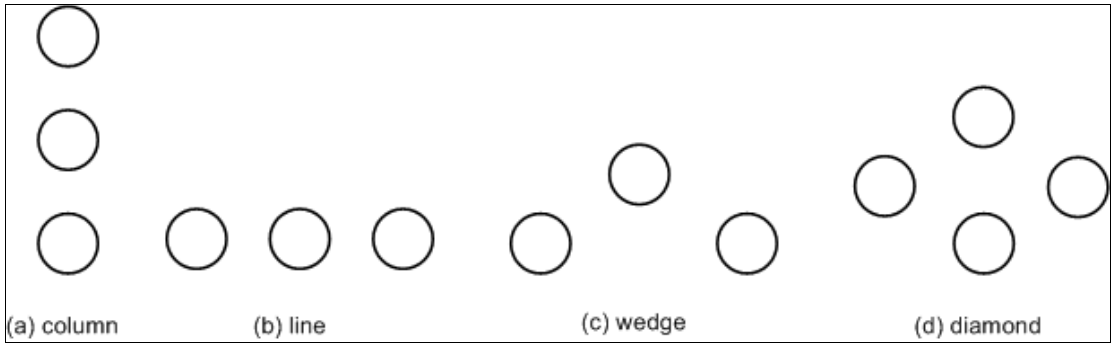

Figure 3.1. Examples of formation shapes with three robots.

The concept of formation control has been studied extensively in the literature with applications to the coordination of multiple robots [56-63] , unmanned air vehicles (UAVs) [20, 47, 64, 65], autonomous underwater vehicles (AUVs) [66, 67], and spacecraft [68-70]. Four main control frameworks have emerged to address the multi-agent formation problem including the behavior-based and potential fields, leader-following, graph-theoretic, and virtual structure approaches. Tables 3.1, 3.2 and 3.3 and the following sections summarize relevant work in formation control. Table 3.1 shows formation control methodologies with implicit robotic control. Table 3.2 shows formation control methodologies where control is explicit and leader-follower based approach. Table 3.3 shows the remainder of explicit robotic control formation methodologies.

In Section 3.1.1, behavior-based and potential field formation control strategies will be discussed, followed by leader-follower and graph-theoretic techniques in Section 3.1.2, and finally virtual structures in Section 3.1.3. In Section 3.1.4, other formation control strategies are discussed.

Table 3.1. Formations with implicit robotic control.

System	Shape Representation	Formations	Reference Type	Robotic Control
Sugihara [35]	Implicit	geometric	shortest or farthest neighbors	algorithmic
Yun [71]	Implicit	line, circle	neighbor-based	algorithmic
Balch [25]	Implicit	column, line, diamond, square	Attachment sites	behavior-based

Table 3.2. Formations with explicit robotic control utilizing leader-follower reference.

System	Shape Representation	Formations	Reference Type	Robotic Control
Balch [50]	Hardcoding	line, wedge, column	unit-center, leader, or neighbor	behavior-based
Egerstedt [22, 72]	formation constraint function	triangular	virtual leader and reference points	tracking
Fredslund [28]	chain of friendships	line, column, diamond, wedge	friend robot (leader-follower)	desired angle keeping
Das [52]	Graph	triangle, line, circular arc	directed edge (leader-follower)	control algorithms
Hsu [73]	Graph	line, column, wedge	directed edge (leader-follower)	potential field and behavior-based
Gazi [74]	potential function	aggregations	local interactions	sliding-mode
Ren [75]	virtual structure	geometric configurations	virtual center, neighbors	trajectory tracking
Elkaim [76]	predefined points	ring, line, pyramid, box	virtual leader	potential functions
Chuang [63]	Ball	flocking	virtual leader	pairwise potentials

Table 3.3. Formations with explicit robotic control utilizing other reference types.

System	Shape Representation	Formations	Reference Type	Robotic Control
Barfoot [58]	formation offsets	geometric	point-referenced	motion planning
Lewis [54]	virtual structure	geometric configurations	virtual structure points	bidirectional
Chaimowicz [62]	shape functions	convergence to a given 2D curve	local sensing (closest-neighbors)	gradients
Hsieh [77]	shape functions	convergence to the boundary of a specified shape	relative neighbors	artificial potential functions
Ji [78]	Graph	not specified	interaction graph	nonlinear control
Ge [79]	queues and formation verticies	wedge, column, line, circle	queue status	potential trenches

3.1.1 Behavior-Based and Potential Field Formation Control Strategies

Behavior-based systems integrate several goal oriented behaviors concurrently in order to reach a goal. In the behavioral approach to formation control [19, 27, 50, 80-85], each agent has several desired behaviors, and the control action for each swarm member is defined by a weighting of the relative importance of each behavior. In addition there are many formation control strategies which utilize potential fields [25, 74, 76, 86-90]. Behavior-based methods and potential fields are often combined in formation control of mobile robot systems.

In [19], a behavior based formation control method is used in which a leader is referenced to determine formation position. Each of the robots is equipped with some primitive motor behaviors. The behaviors have control parameters which are tuned using a genetic algorithm. During the motion, the leader decides its next position based on its knowledge about the goal and environment and then broadcasts its anticipated position to the followers. The use of genetic algorithms for optimizing the formation control is interesting, but the drawback is that the system

requires almost global knowledge about the environment and is dependent on receiving this via broadcast communication.

In [50], the behavioral approach is applied to formation-keeping for mobile robots, where control strategies are consequent of several simultaneous behaviors. In this approach, line, column, diamond and wedge formations are presented. For each formation, each robot has a specified position based on an identification number.

In [91], complex formation maneuvers are broken down into a sequence of behaviors to achieve formation patterns. A bidirectional ring topology is used to maintain the formation of the system. The advantage of this approach is that it can be implemented when only neighbor position information is available. Because of the way formation patterns are defined, this approach is limited in directing rotational maneuvers for the group.

In [69], a behavioral-based approach is used to obtain formation control laws to maintain attitude alignment among a group of spacecrafts. The approach utilizes velocity feedback and the other passivity-based damping.

Behavior-based methods and potential fields are often combined in formation control of mobile robot systems [50, 82]. In these approaches, each robot has basic motor schemas which generate a vector representing the desired behavioral response to sensory input. These motor schemas include behaviors such as obstacle avoidance and formation keeping.

In [25], a strategy to arrange a large scale, homogeneous team in a geometric formation utilizing potential functions is presented. The method is inspired by and is similar to the process of molecular covalent bonding. Various robot formations result from the usage of different attachment sites. Attachment sites are constructed relative to the other agents in the team. Formation is maintained in the presence of obstacles. Local sensing is sufficient to generate and maintain formation. Robots are not assigned specific locations, but attracted to the closest teammates or attachment sites. Behaviors such as “move to a goal” and “avoid an obstacle” are utilized for robotic control.

In [27], formation control is achieved via a group formation behavior based on social potential fields. The robot’s behavior is based on a subsumption architecture where individual behaviors are prioritized with respect to others. This work extends the work in [85] using the social potential fields method by integrating agent failure and imperfect sensory input. This method uses only local information and is scalable to very large groups of robots.

In [92], the behavior-based formation control is modeled by a non-linear dynamic systems for trajectory generation and obstacle avoidance in unknown environments. The desired formation

pattern is given through a matrix which includes parameters to define the leader, desired distance, and relative orientation to the leader. These parameters are then used to shape the vector field of the dynamical system that generates the control variables.

In [79, 93], the desired formation pattern is represented in terms of queues and formation vertices. The desired pattern and trajectory for the group of robots is represented by artificial potential trenches. Each robot is attracted to and moves along the bottom of the potential trench, automatically distributing with respect to each other.

Although the behavior-based approach has the advantage of formation feedback through neighbor-based communication and it is highly decentralized; it is extremely difficult to analyze mathematically and has limited ability in precise geometric formation keeping. If a very precise formation is required, another method, such as a virtual structure method, should be used.

3.1.2 Leader-Follower and Graph Theoretic Formation Control Strategies

The leader-follower [94, 95] and graph-based [26, 34, 53, 96] approaches are decentralized, but each robot is assigned a unique name. Also, the robots typically try to maintain some desired distance to some of their neighbors and/or virtual points. Some robot members are designated as leaders while other robots are designated as followers. The structure is dependent on the architecture utilized. There can be as few as one leader, or there can be several leaders with a hierarchical structure. The leaders are generally tracking a predefined trajectory, and the followers are tracking their leader(s) in some manner dependent on the approach used. There are many approaches for maintaining formations utilizing a leader-follower or graph-based method. The objective could be to maintain a desired length and/or desired relative angle from leader robot(s).

In [28, 97], local sensing and minimal communication between agents is used to maintain a predetermined formation. Each robot in the group keeps a single ‘friend’ robot which it knows via a special sensor, and maintains a specific angle at all times in relation to this ‘friend’. The angle is calculated locally per robot. Broadcast communication is utilized but is minimal, only passing robot IDs, directional changes, and formation messages. Each robot has access to the number of robots in the system as well as the type of formation. With each formation, each robot has a specified angle to keep between its friend and the frontal direction. This algorithm is limited to only formations which can adhere to the chain of friendship which limits it to no more than two loose ends or frontally concave formations.

In [22, 72], a coordination strategy for maintaining formation over a given trajectory is presented. The formation control is achieved through the tracking of virtual reference points. The leader of the path acts as a reference point for the robots to follow. The robots move in a triangular formation avoiding obstacles, and if the tracking errors are bounded then the formation error is stabilized.

In [32], artificial potentials define the interaction control forces between adjacent vehicles and define the desired inter-vehicle spacing. The approach is inspired by biology considering attraction and repulsion to neighbors as well as velocity matching. Virtual leaders or beacons (not an actual vehicle) are used to manipulate group geometry and motion direction. Constant prescribed formations of schooling and flocking are demonstrated, but the approach is only applicable to homogeneous formations. Closed-loop stability is proven with Lyapunov function using kinetic and potential energy of the robot system.

In [31], an approach is provided for multirobot formations including obstacle avoidance using local communication and sensing. The approach is behavior-based but integrates social roles representing positions within the formation using local communication to improve performance. New agents are allowed to join the formation by role changes when necessary. The local communication is fixed, and the locally information travels to the leader which knows the entire shape of the current formation and decides on necessary role changes. This information is then passed back to the necessary followers, updating the information. The roles or positions for the robots are decided dynamically and changed as the formation grows.

In [98], leader-follower patterns are used for formation control. In this approach, it is assumed that only local sensor based information is available for each robot. In [99], a fuzzy-logic based leader-follower approach is presented for formation control. Maintaining correct formation position while avoid collisions is investigated here. Separate fuzzy-logic controllers are developed for formation position control and internal collision avoidance. Circle, wedge, line, and column formations are presented.

In [34], a graph theoretic approach is described which allows multiple vehicle formations to be defined as rigid graphs. In [89], the authors also show a graph theory called graph rigidity which is very helpful in representation and control of formations of multiple vehicles. These rigid graphs identify the shape of the formation and the interconnections lead to automatic generation of potential functions. The basis is that performing graph operations allows the creation of larger rigid graphs to be formed by combining smaller rigid sub-graphs. This work

has specific applicability in the area of dynamic reformation as well as splitting and merging of vehicles in a distributed manner.

In [26], a graph theoretic framework for formation control of moving robots in an area containing obstacles is presented. Control graphs are used to determine the behavior and transitions that are possible between different formations or control graphs. Each team's model consists of a lead robot, shape variables containing relative positions of robots, and a control graph that describes the behaviors of the robots in the formation. This method is scalable to large groups despite the computational complexity of growing control graphs due to its decentralized design.

In [100], another graph-based approach consisting of a four-layer hierarchical modular architecture for formation control is proposed. The group control is at the highest level layer generating a desired trajectory for the whole group to move. The next layer manages formation control implementing a physical network, a communication network, and a computational network (control graph). The formation is maintained by using only local communication and relative position. Next, there are two layers, one to control robotic kinematics and one to handle robot dynamics. This system is very scalable to heterogeneous systems because of the layered approach with the adaptable kinematics and dynamic layers. This method also allows for various formations, and both centralized and decentralized methods of control graph assignment are described.

In [30], nearest neighbor rules are used to control the motion of the robots updating each robot's heading based on the average of its heading plus its neighbors' headings. Undirected graphs are used to represent robot interactions. This method claims that all robots, despite the absence of a centralized coordination mechanism and the dynamic neighbor changes, that there will be an overall emergent coordinated motion. No particular formation is exhibited but overall robot motion is in the same direction.

While the leader-follower and graph-theoretic approaches are logical and easily implemented, there are limitations. Each leader is a single point of failure for the formation so this makes these systems weaker than completely decentralized systems. Reassigning leadership and information flow in the event of a failure can be difficult and computationally expensive. In addition, if there is no explicit response from the followers to the leader, and if the follower has a difficulty, the formation cannot be maintained.

3.1.3 Virtual Structure Formation Control Strategies

In the virtual structure approach [29, 54, 101], the entire formation is treated as a single rigid body. The concept of the virtual structure was first introduced in[101]. The virtual structure approach is typically used in spacecraft or small satellite formation flying control [68]. The virtual structure can adapt its shape expanding in a specified direction while maintaining a rigid geometric relationship among multiple agents. These approaches were proposed to acquire high precision formations control for mobile robots.

In [29], a virtual structure method is proposed for self-organizing formation control in which it is assumed that elements are not connected to each other and can move in a continuous space. The goal is to arrange the elements into the spatial pattern of a crystal using virtual springs to keep neighboring elements within close proximity. Each pair of elements within a certain range is connected with a virtual spring. The elements form triangular lattices with random outlines. In order to determine the desired outline, virtual springs are broken with a certain probability. The candidate springs for breaking are chosen based on the connections of its neighbors. Elements interact locally and have no global information, but the tuning of parameters for different formations and number of robots is computationally expensive.

The main advantage of the virtual structure and graph theoretic approaches to formation control is that it is simple to prescribe the behavior for the entire group. The formation structure is generally very tight and precise in these methods during tasks. The main disadvantage is in the computation complexity of some these methodologies, as well as the centralized nature which make these systems less robust to failure.

3.1.4 Other Control Strategies in Formation Control

There are also many other formation control strategies which do not easily fit into the categories previously discussed. In [33], a distributed coordination algorithm is presented for multirobot systems in which a particular method of navigation function with Vornoi partitions is used. The robots navigate, maintaining a flocking formation, while avoiding obstacles. Inter-vehicle communication is achieved by using a global list of positions where every single vehicle can only get a list of its neighbors within a specified radius.

In [24], the formation problem is solved for a group of autonomous vehicles by providing inter and intra vehicle constraints as well as a time limit for reconfiguration. The nominal input

trajectory for each vehicle is determined so that the group begins in the initial position and ends in the final position in the specified amount of time. The information is represented as a particular form of input signals so the formation problem can be reformulated as an optimization problem and solve more efficiently especially for large groups of vehicles. This method suffers from single point failure, though, since it utilizes a central controller.

In [102], the stabilization and maneuvering of vehicles is achieve through model predictive control. Each individual vehicle may be governed by nonlinear and constrained dynamics. The vehicles are stabilized to acceptable equilibriums rather than precise locations for each individual. The individual trajectories of autonomous vehicles moving in formation were generated by solving an optimal control problem at each time step. This is computationally demanding and hence not possible to perform in real-time.

In [103], a distributed control scheme for multirobot systems is presented. Each robotic vehicle has its own coordinate system, and it senses its relative position and orientation in reference to others in order to create a group formation. Despite the presence of a supervisor, the robot vehicles are stabilized. The stability of the vehicles is proven only for symmetrical formations.

In [35], approximate pattern formation is achieved by sharing position information to other robots. In this method, it is assumed that robots have global position information. An algorithm is developed for each pattern formation which includes circles, polygons, lines, filled circles, and filled polygons. Robots can also be split into an arbitrary number of equal or near equal group size. Although this method is decentralized, the information sharing of the global position of each group member to the whole group is a significant drawback.

In [23], a target assignment strategy for formation building of multiple robots scattered in the environment is presented. The algorithm first begins with assigning each robot a target point in the desired formation. Trajectories, including collision avoidance, are generated by a central planner. Priorities of areas around the robots are integrated so robots will avoid each other when in a certain threshold. Sensing is global and the method is dependent on a central controller.

In [104], a formation control methodology based on generalized coordinate system is presented. The generalized coordinates characterize the vehicle's location, orientation and its shape of the formation. This allows the group to be controlled as a single entity. Force-based and velocity-based controls are developed. Similar ideas utilizing coordinated systems for shape representations are presented in [105, 106].

In [64], a hierarchical, centralized planning method to achieve a desired formation for a group of UAVs is presented. The desired flight trajectories for each UAV are determined by a leader which is more capable than the other team members. Control laws are designed based on the desired flight trajectories. To achieve flight formation according to a given scenario, each UAV independently takes off towards its corresponding trajectory and locks onto it in finite time. Only the leader is equipped with sensors, and it communicates to the other team members what trajectories to track via a communication channel. This method is very prone to failure, is very risky in dynamic environments, and scales very poorly with growing team sizes.

In [20, 21] swarms of unmanned ground vehicles (UGVs) are coordinated with the use of unmanned aerial vehicles (UAVs). In [21], a hierarchy is formed between the UAV and the UGVs. The UAV is in charge of determining the grouping and merging of swarms as well as the swarm distribution and motion of the group. The UGVs are at the lowest level of the hierarchy. At the second level, there is one UAV blimp for the tracking of each group of UGVs, and at the highest level there is a centralized planner for the whole system, a single UAV. This system is centralized with each robot communicating to its central planner. The central planner broadcasts information to the robots as well as sending information to the UAVs. The shape of the formation is determined by the central planner in the form of a directed graph. Due to the dependence on a central planner, this method is prone to failure, but the UGV swarm-UAV coordination proves interesting.

There are also many other methods in the application of formation control. In [107], genetic algorithm and reinforcement learning are used for robot formation control and obstacle avoidance. In [108], neural networks and radial basis functions are used to achieve formation control. Vision is used for formation control in [52, 94, 109-112].

3.2 Foundation of the Proposed Approach

In this work, a new approach for swarm formation control is set forth. Many limitations that other methods suffer from can be solved utilizing this approach. This approach most closely falls into the potential field and leader-follower control for swarm formation. The architecture utilized is best classified as hybrid or hierarchical, rather than a purely decentralized. Most of the formation methods discussed assume that the swarm members begin there activities in the mission space, and do not address how the robots travel to that mission space. This assumption is not acceptable in scenarios where groups of unmanned vehicles need to travel autonomously from

one region to another of the mission space while remaining a cohesive unit. In this approach, no assumptions are made about where the swarm members begin activities. This approach utilizes ellipsoid contours for shape representation and artificial potential fields and limiting functions for control.

Although this approach does not achieve the precision that is seen in virtual structure methods in [54, 75, 101], robots adhere to set formation parameters, including an approximate shape and dispersion. The robots adhere to a semi-static formation while moving. They are also able to change formation dynamically to respond to the loss of a team member or environmental change. The contribution of this work is a swarm formation control methodology which is scalable to varying swarm sizes, computationally efficient, supports multiple formations, dynamic formation switching, decentralized and centralized control techniques, and heterogeneous swarms.

This framework and methodology for swarm formation control allows heterogeneous robots of ground vehicles to maintain formation while avoiding collisions and handling dynamic changes in the environment. The design is not dependent on the size of the swarm or the type of platform and is transferable to multiple formation types including line, wedge, circles, and ellipses. Two types of formation reference are utilized in this work. On utilizes unit centers to define the formation. The other utilizes a hierarchical structure of leaders.

Artificial potential fields are the basis for the swarm formation control, obstacle avoidance, and overall swarm formation movement. This work is based on the troop movement models discussed in [1, 2]. In [1], reaction diffusion equations (RDEs) are used to describe behavior and control the troop as a group. Troop behavior is described with RDEs, also describing motion and diffusion of the troop. In [2] troop movement model is extended by combining Variable Resolution Terrain (VRT) model [113] and RDEs. VRT was developed to represent the battlefield as a continually differentiable surface. When VRT is combined with RDE, it creates a simulation tool to model troop movement on simulated battlefields.

3.3 Summary

This chapter has provided a survey of the research in multirobot formation control. There are a variety of approaches to the multi-robot formation control problem. Depending on the task or mission, one method might be better than another. This work is most similar to [77, 79, 93] but it is much more applicable across control platforms. It can be utilized with a centralized

architecture design, hierarchical architecture, or purely decentralized depending on communication requirements for the group. Obstacle information can either be locally sensed or broadcast through the group. Unlike the approach in [51], this approach is also platform independent.

In this work, [45] is expanded to achieve tighter formation control with fewer potential fields utilized. In a static formation, the stability and convergence of all the robots to the boundary of a specified ellipse is guaranteed and can be shown mathematically. Swarm member can traverse a trajectory while avoiding other swarm members and static and dynamic obstacles. The main contributions of this work are the concept of utilizing a sum of multiple vector fields based on a simple surface, the bivariate normal, with few parameters for formation control for a swarm of arbitrary size. The ability to change formation, and redistribute the robots makes this approach very dynamic. Conventional methods which utilize potential fields using predetermined points of attraction limit the formation's scalability. Utilizing the bivariate normal function as a basis, a variety of formations can be achieved by robots dispersing about the contours.

Chapter 4

Proposed Formation Control Approach

4.1 Swarm Surface

The main objective the proposed approach is to attract elements of a swarm into a bounded formation and allow the swarm to stay in that formation as it moves around. A vector field is used to attract swarm members to an ellipse with desired parameters. The minimum distance between swarm members is controlled using an additional vector field.

At any instant in time, the robots can be visualized as particles moving in a potential field generated from a bivariate normal 'hill' that controls the velocity and heading of the swarm members. A bivariate normal function with form given in:

$$f(x,y) = e^{-\alpha((x-x_c)^2 + \gamma(y-y_c)^2)} \tag{4.1}$$

produces an oval/ellipsoid shaped function. Assuming that the current robot location is at *(x, y)*, the center of the function in (4.1) is represented by (x_c, y_c) with respect to the world reference frame. The control variable γ determines the ratio of the minor axis (y-direction) to the major axis (x-direction) affecting the eccentricity of the swarm. Note that the center (x_c, y_c) could be a function of time allowing the swarm to move along a path.

The *x* and *y* partial derivatives create the velocity vectors that are used to determine the heading and velocity of each member of the swarm as shown in:

$$\begin{aligned} d_x &= -2\alpha f(x,y)(x-x_c) \\ d_y &= -2\alpha\gamma f(x,y)(y-y_c) \end{aligned} \tag{4.2}$$

Just as with a single robot, the swarm formation, treated as a single shape, has both a local reference and a world reference frame. For the swarm to follow a trajectory in the world reference frame, an axis rotation is required. The heading, ϕ, between the swarm formation's x-axis and the center (x_c, y_c) must be found; the rotated coordinates for (x, y) and (x_c, y_c) can be found using:

$$\begin{aligned} x_{rot} &= \cos(\phi)(x - x_c) - \sin(\phi)(y - y_c) \\ y_{rot} &= \sin(\phi)(x - x_c) + \cos(\phi)(y - y_c) \end{aligned} \tag{4.3}$$

The rotated coordinates are then substituted to find d_x and d_y.

4.2 Formation Problem

In order to describe the general formation problem, it is discussed in reference to convoy protection. Suppose that a swarm of robots needs to accompany a convoy of vehicles and surrounding them in a particular formation. In the general case, the convoy can be enclosed in some geometric shape, defined loosely by dimensions, direction of travel, and the center of mass as shown in Figure 4.1. The length of the convoy along the axis of travel is 2A. The width of the convoy with respect to the axis of travel is 2B.

A field needs to be designed to attract the swarm members to surround the convoy in a designated formation. The swarm members need to be close enough to the convoy to offer protection, but far enough to allow the convoy to move safely.

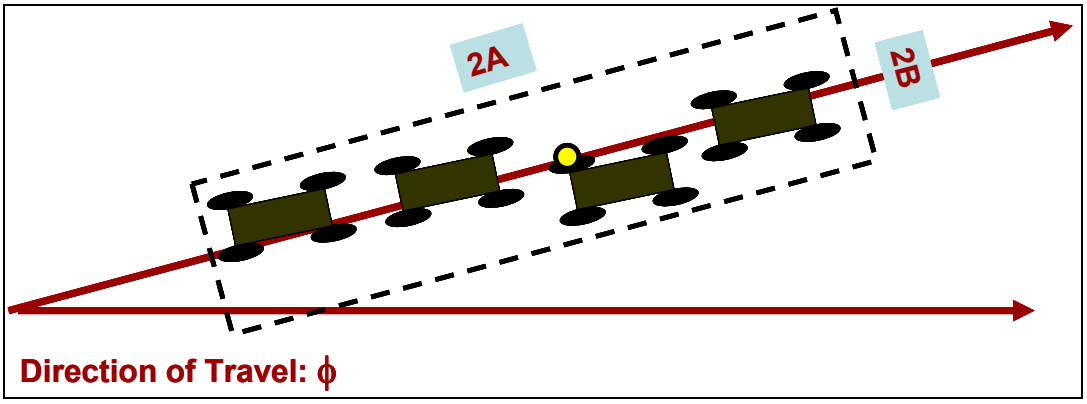

Figure 4.1. Convoy description.

Suppose the positions of each of the convoy vehicles are known and that the centroid of the convoy is (x_c, y_c). It is possible to enclose the convoy within a sequence of concentric ellipses with center (x_c, y_c). Figure 4.2 depicts three elliptical rings with center (x_c, y_c), semi-major axis A, and semi-minor axis B, surrounding a convoy of vehicles. By attracting swarm members to the center elliptical ring described as the set of points $(x,y) \in \Re^2$ satisfying:

$$R^{*2} = (x - x_c)^2 + \gamma(y - y_c)^2 \tag{4.4}$$

where (x_c, y_c) is the center and γ is the axis ratio B/A. the swarm can be closely associated with the convoy without endangering the convoy vehicles. For a fixed value of γ, we will refer to the set of points (x, y) satisfying (4.4) as the R^* ellipse.

The general form of the swarm controller is described by:

$$V(x, y, t) = \sum_{1}^{N} w_i(x, y, t) V_i(x, y, t) \tag{4.5}$$

where $V(x,y,t)$ gives the velocity of the swarm at a particular time and place. Each of the vectors $V_i(x,y,t)$ is associated with different fields and $w_i(x,y,t)$ are weights of the overall contribution of the ith vector. In general, the field $V(x,y,t)$ is the weighted sum of N different vectors, each of which is acting on the swarm. In this case, three different vector fields are utilized: one attracts robots to the elliptical band from points outside the elliptical region; one pushes robots away from the center towards the desired band; and one controls the movements of the robots within the band).

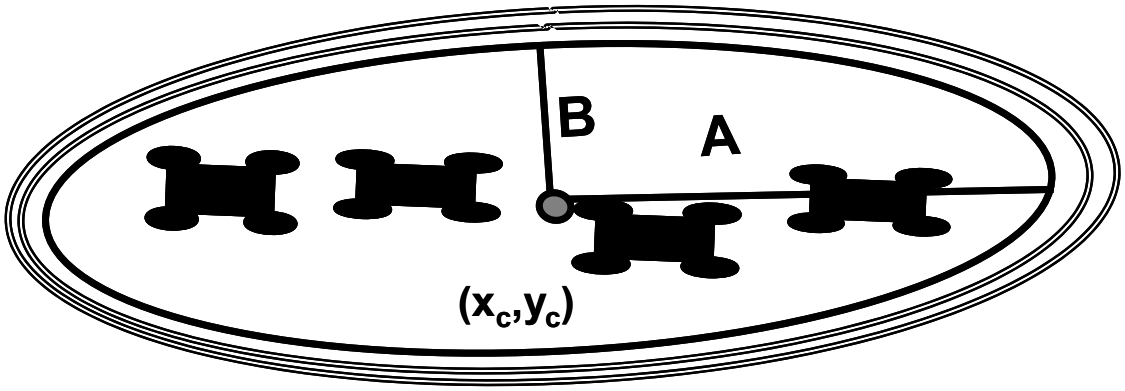

Figure 4.2. Convoy of vehicles surrounded by concentric ellipses.

The challenge is to develop a potential field based controller using a small number of physically relevant weights, w_i, and vectors v_i that attract particles to a neighborhood of the R^* ellipse. This neighborhood is shown in Figure 4.3. The parameters R_{in} and R_{out} denote the inside and outside boundaries of the R^* neighborhood, respectively, as shown in Figure 4.3. The desired vector fields will 'trap' the robots in these bands. Typically, this is a very narrow band of allowable space for the robots with a controllable width of $\Delta R_{in} + \Delta R_{out}$ where:

$$R_{in} = R^* - \Delta R_{in} \tag{4.6}$$

$$R_{out} = R^* + \Delta R_{out} \tag{4.7}$$

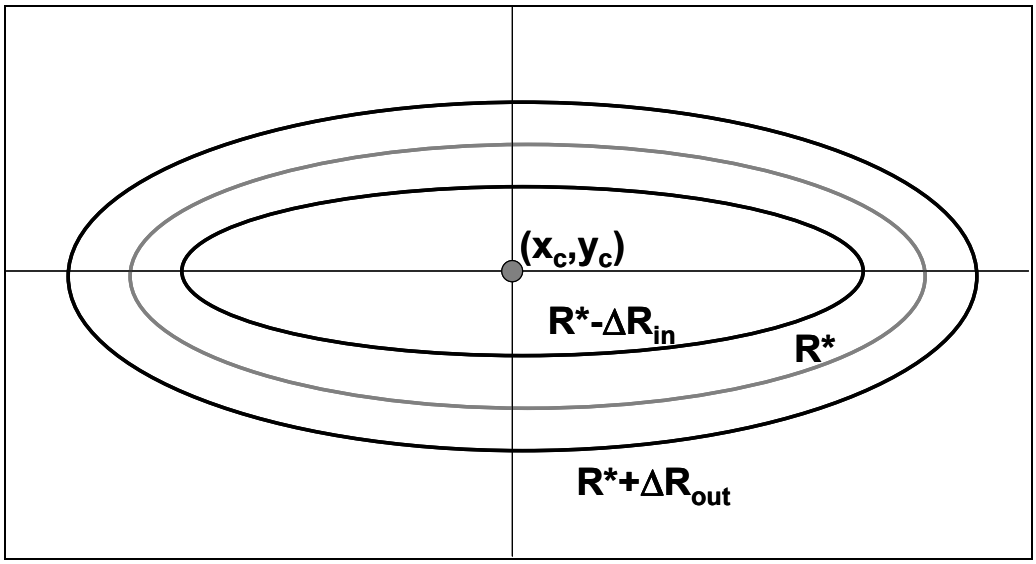

Figure 4.3. Elliptical attraction band for the swarm robots.

The vector field will be constructed utilizing the normalized gradient from Equation (4.2). For every *(x, y)*, let the gradient field vector have the form:

$$V_i(x,y) = \begin{cases} W_i(x,y)\dfrac{1}{L(x,y)}\begin{pmatrix}(x-x_c)\\ \gamma(y-y_c)\end{pmatrix} for\ (x,y) \neq (x_c,y_c) \\ \begin{pmatrix}0\\0\end{pmatrix} for\ (x,y) = (x_c,y_c) \end{cases} \tag{4.8}$$

where:

$$L(x,y) = \sqrt{(x-x_c)^2 + \gamma^2 (y-y_c)^2} \tag{4.9}$$

The vector $\frac{1}{\mathcal{L}(x,y)} \begin{pmatrix} (x-x_c) \\ \gamma(y-y_c) \end{pmatrix}$ is a unit vector that provides the direction of the vector at *(x, y)*. The function *w(x, y)* provides the magnitude of the vector at that point. Notice that for any *(x, y)*, this vector points *away* from the center of the ellipse.

In the defined vector field, particles starting within the $R^* - \Delta R$ ellipse with:

$$R^* = \sqrt{(x-x_c)^2 + \gamma^2 (y-y_c)^2} \tag{4.10}$$

move out from the center until they reach the R^* neighborhood. Particles starting outside the $R^* + \Delta R$ ellipse move toward the center until they reach the R^* neighborhood. Eventually all the robots will be trapped within the neighborhood given by:

$$(R^* - R_{in}) \le R \le (R^* + R_{out}) \tag{4.11}$$

4.3 Generation of Vectors and Vector Field

4.3.1 Description of Vector Fields

In order to generate the desired vector fields to hold the robots inside the R^* neighborhood, three fields are needed. One attracts robots to the elliptical band from points outside the elliptical region. One pushes robots away from the center towards the desired band. One controls the movements of the robots within the band. The first two fields utilize the gradient vector field discussed in the previous section, $G^- = -(d_x, d_y)$ points away from the center as shown in Figure 4.4. Vector calculus dictates that the gradient vector field, $G^+ = (d_x, d_y)$ points in the direction of greatest increase of the function *f(x,y)*, which is towards the center as illustrated in Figure 4.5. The third vector field utilizes a vector that is perpendicular to the gradient vector. The vectors

(d_x, $-d_y$) and ($-d_x$, d_y) are perpendicular to the gradient; Figure 4.6 shows such a perpendicular field.

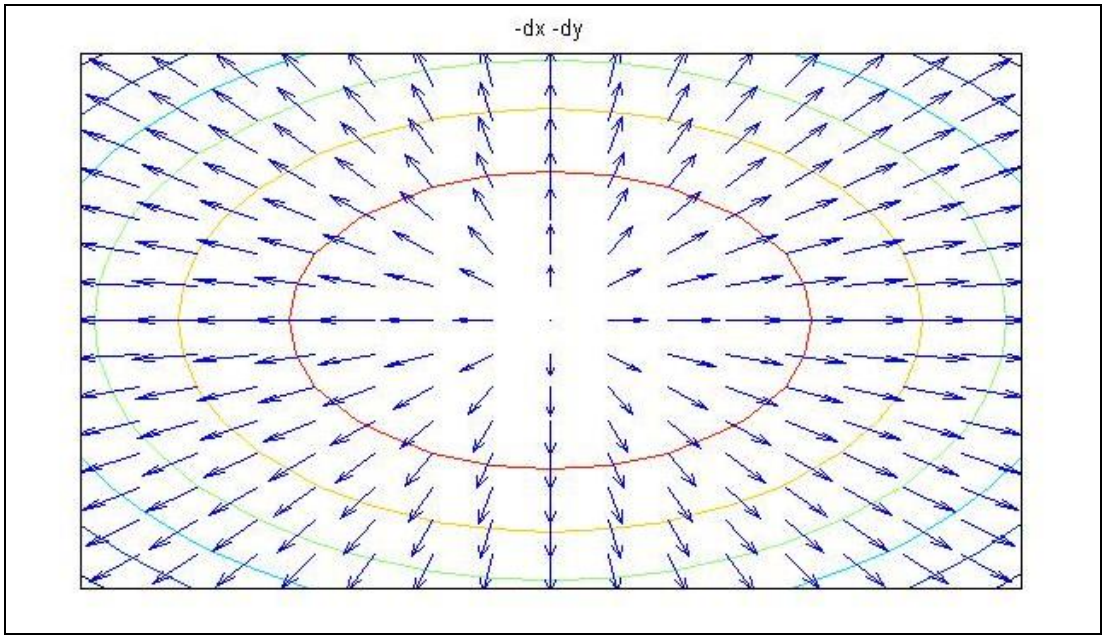

Figure 4.4. Vector fields directed away from the center (G^-).

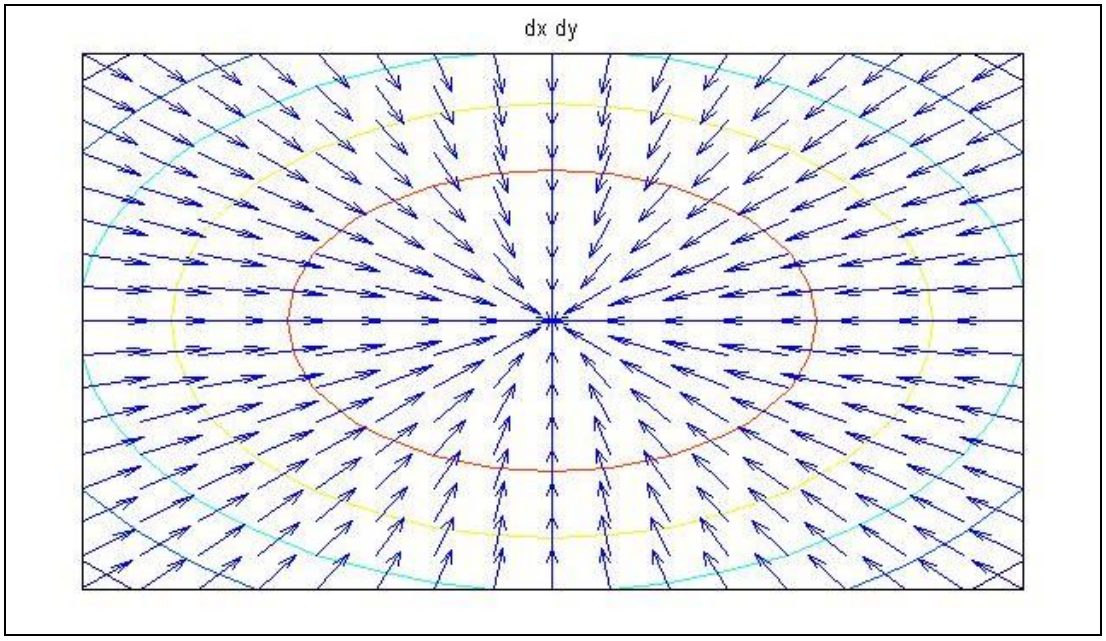

Figure 4.5. Vector fields directed towards the center (G^+).

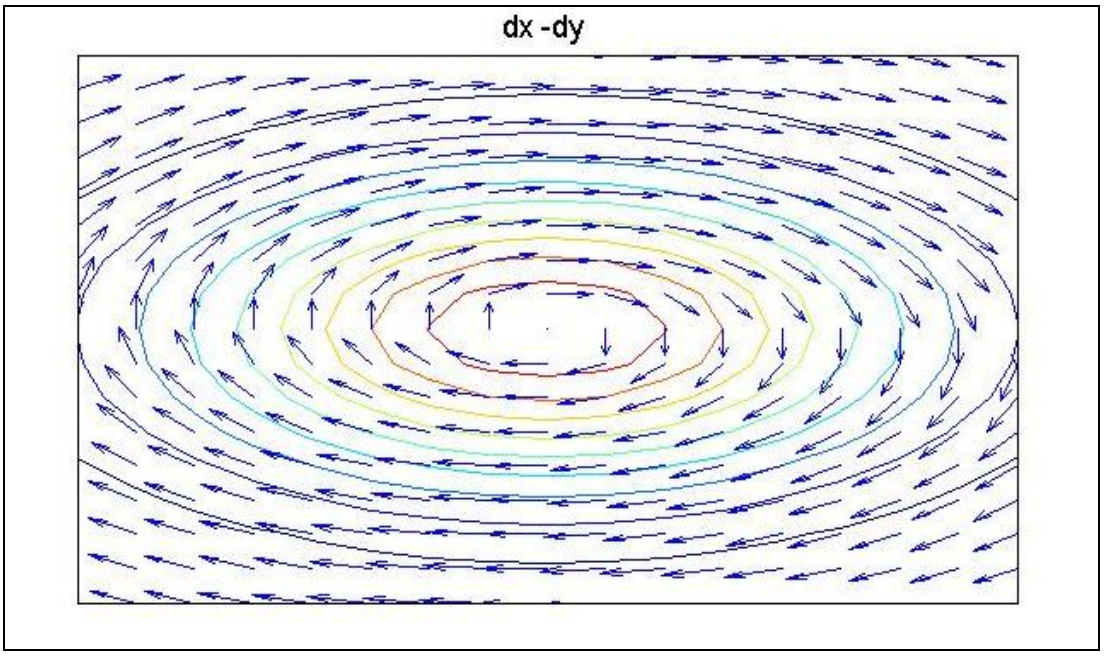

Figure 4.6. Vector fields directed perpendicular to the center ($G^{\perp}$).

4.3.2 Description of Limiting Functions

Tighter swarm control may be accomplished when restricting the influence of the vector fields to a small region of the *x*-*y* plane by multiplying each of the fields by a limiting function. This limiting function controls the influence of the vector field in various regions of R^2. For instance, the limiting function can determine the distance from the center at which the vectors in the field 'die out' or become smaller than some number ε.

4.3.2.1 S_{in} and S_{out} Sigmoid Limiting Functions

In order to create the desired field, the G^- and G^+ fields as shown in Figure 4.4 and Figure 4.5 must be limited to end at the appropriate boundaries. These fields will be limited with *sigmoid* functions or S-shaped functions of the form:

$$S(t) = \frac{1}{1+e^{-t}} \tag{4.12}$$

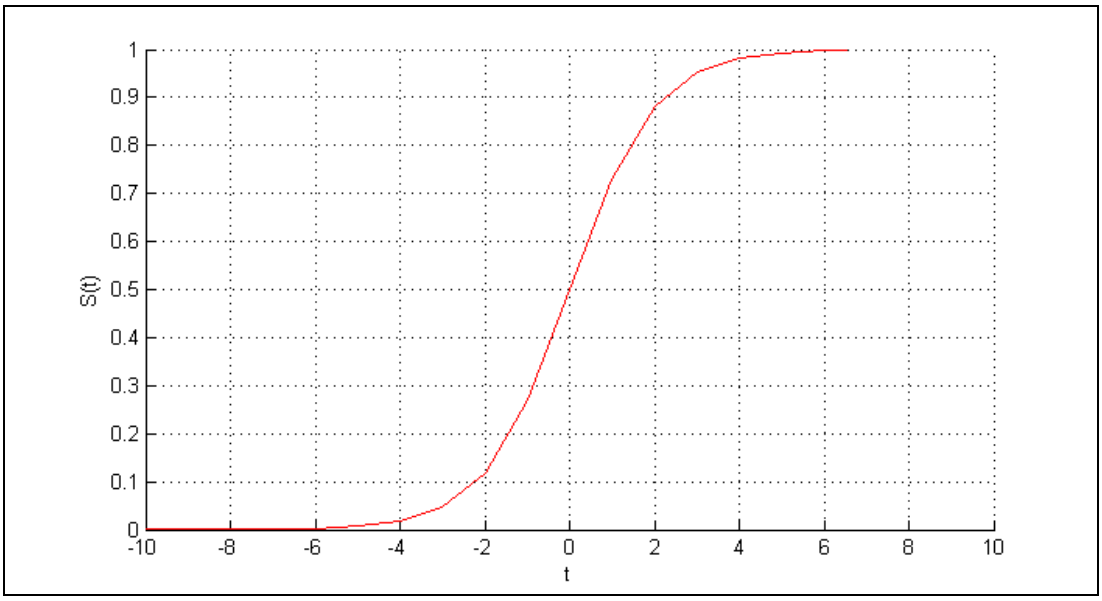

Figure 4.7. General sigmoid function.

Figure 4.7 shows the general case of a sigmoid function. The value of the sigmoid function ranges between 0 and 1 and has one inflection point at 0.5.

Vector fields 'moving away' from the center (the vectors inside of the ellipse) require a limiting function that approaches zero as the distance from the center increases; such a limiting function is given by:

$$S_{in}(\alpha_{in}, r, R^*_{in}, \Delta R_{in}) = 1 - \frac{1}{1 + e^{\alpha_{in}(r-(R^*-\Delta R_{in}))}} \tag{4.13}$$

Gradient vector fields directed towards the center (those vectors outside of the ellipse) are required to approach zero as the vectors 'move towards' the center; this is achieved using the limiting function by:

$$S_{out}(\alpha_{out}, r, R^*, \Delta R_{out}) = 1 - \frac{1}{1 + e^{-\alpha_{out}(r-(R^*+\Delta R_{out}))}} \tag{4.14}$$

The G^- field should die out at R^*-ΔR_{in}, and the G^+ field should die out at R^*+ΔR_{out}. This creates the field shown in Figure 4.8.

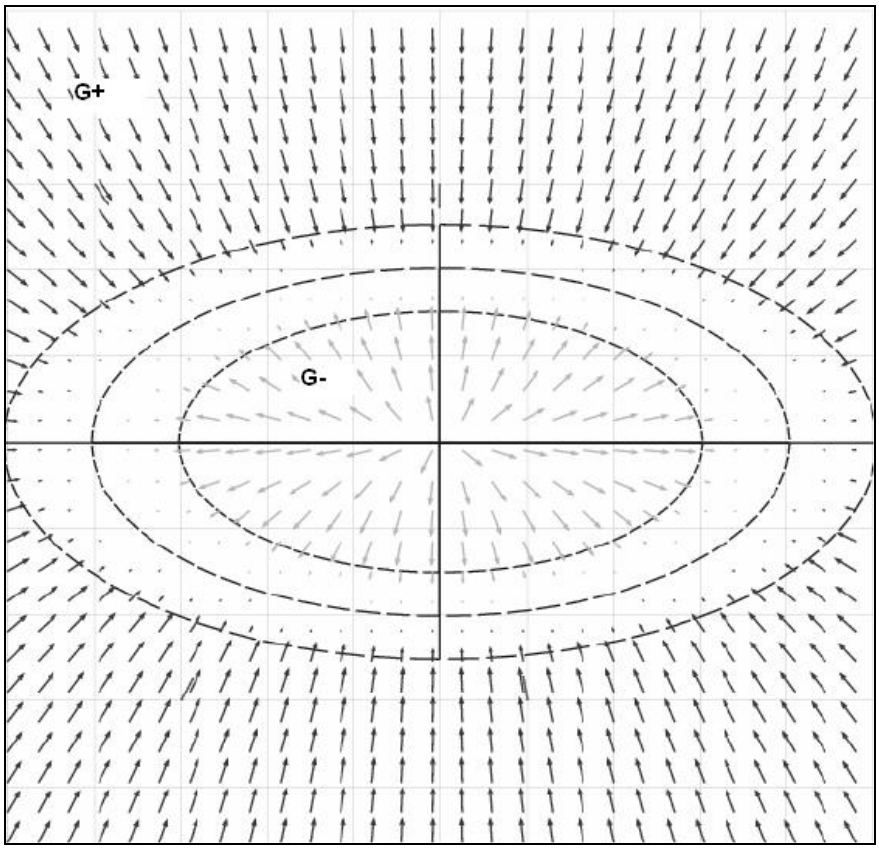

Figure 4.8. Combined in (G^+) and out (G^-) fields.

Although Figure 4.8 illustrates a symmetric case where $\Delta R_{out} = \Delta R_{in}$, the weight, $W(x,y)$, can be written as:

$$W(x,y) = S_{in}(x,y) - S_{out}(x,y) \tag{4.15}$$

so that the inside and the outside of the R^* ellipse can be considered separately. As a simplification, a modified distance function, r, will be used where:

$$r = \sqrt{(x_{rot})^2 + \gamma^2 (y_{rot})^2} \tag{4.16}$$

which can be simplified (using basic trigonometry) to:

$$r = \sqrt{(x - x_c)^2 + \gamma^2 (y - y_c)^2} \tag{4.17}$$

So *W(x,y)* becomes:

$$W(r) = S_{in}(r) - S_{out}(r) \tag{4.18}$$

Notice that *r* is never negative. The plot of the functions S_{in} and S_{out} as a function of r is provided in Figure 4.9. S_{out} has its largest influence at points whose distance from the center of the ellipse is small. S_{in} has its greatest influence at points whose distance from the center is large. Neither function has much influence within the R* band. Convergence of the S_{in} and S_{out} limiting functions to the R^* band is shown mathematically in Section 4.3.2.1.3.

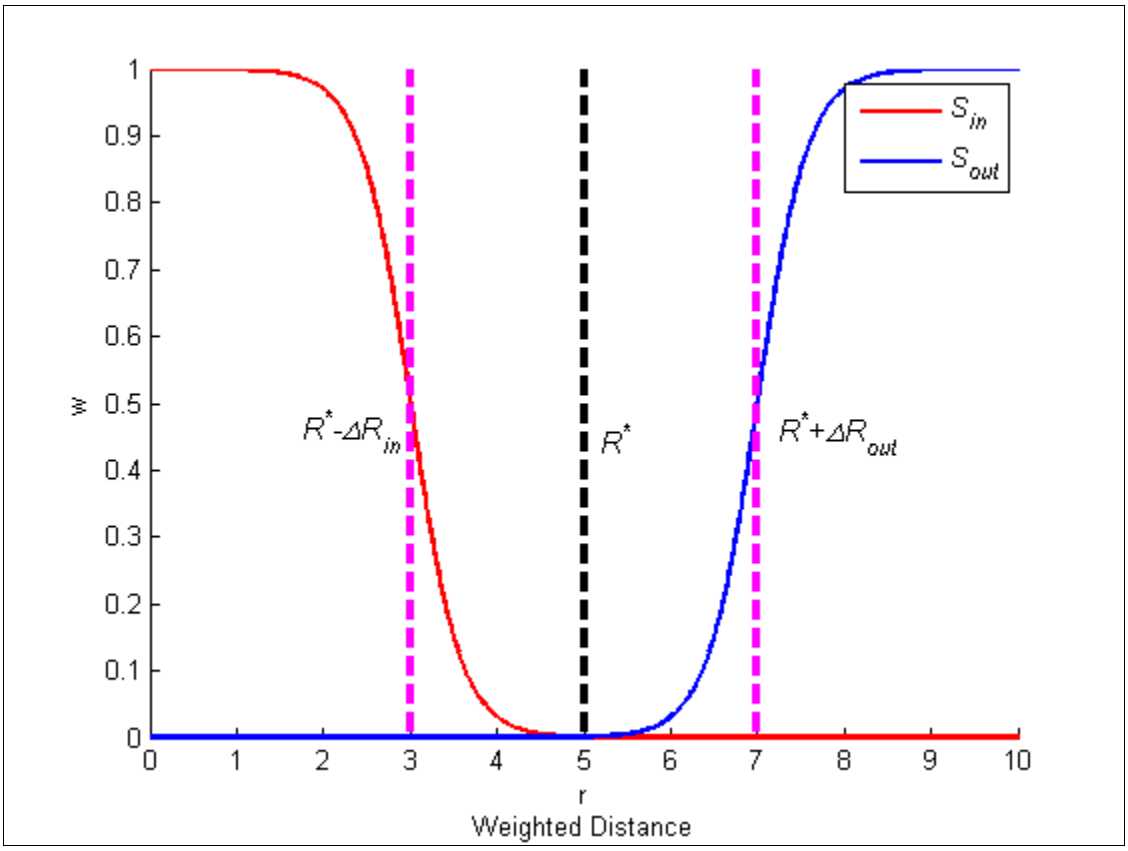

Figure 4.9. The weighting functions S_{in} and S_{out} as a function of the weighted distance r defined in (4.17).

Each of the limiting functions in (4.13) and (4.14) contains *tuning parameters* that may be used as vector field control variables. For each function, one tuning parameter determines how quickly the function approaches zero. The parameters α_{in} and α_{out} control the slope of $S_{in}(r)$ and $S_{out}(r)$, respectively, for r in the set $R - \Delta R_{in} < r < R + \Delta R_{out}$. These parameters will be defined so that the value of $S_{in}(R^*)$ and $S_{out}(R^*)$ can be made arbitrarily small.

Derivation of the α-control variables from the S_{in} and S_{out} limiting functions is discussed in Section 4.3.2.1.1 and Section 4.3.2.1.2. Selection and tuning of the R-values is discussed in Section 4.4.

4.3.2.1.1 Mathematical Solution for α_{in}-Control Variable

In the designed vector field, particles starting within the R^* ellipse will move out from the center until they reach the boundary of the R^* ellipse. Particles starting outside the ellipse will move toward the center until they reach the boundary of the R^* ellipse. The weights will be in the form of Equation (4.18). For particles within the R^* ellipse, the magnitude of the field vector is dominated by:

$$S_{in}(r) = 1 - \frac{e^{\alpha_{in}(r-K_{in})}}{1+e^{\alpha_{in}(r-K_{in})}} \tag{4.19}$$

Equation (4.19) has two parameters α_{in} and K_{in}. The function $S_{in}(r)$ is monotonically decreasing with a limit value of *0*. The function K_{in} is the value of r such that $S_{in}(r) = 1/2$. The function should have a small value at $r = R^*$, so $K_{in} < R^*$. Since the area of interest is the behavior of the vector field near R^*, set $K_{in} = R^* - \Delta R_{in}$ where $0 < \Delta R_{in} < R^*$ then $S_{in}(r)$ can be written as shown in:

$$S_{in}(r) = 1 - \frac{e^{\alpha_{in}(r-(R^*-\Delta R_{in}))}}{1+e^{\alpha_{in}(r-(R^*-\Delta R_{in}))}} \tag{4.20}$$

which can also be rewritten as in Equation (4.13).

The parameter α_{in} controls the slope of $S_{in}(r)$ for r in the set $R^* - \Delta R_{in} < r < R^* + \Delta R_{in}$. Since the desired value of $S_{in}(R^*)$ is very small, let ε be an arbitrarily small number greater than *0* such

that $S_{in}(R^*) = \varepsilon$. The control parameter α_{in} can be determined as shown in Equations (4.21) through 4.26.

$$\varepsilon = \frac{1}{1+e^{\alpha_{in}(R^* - (R^* - \Delta R_{in}))}} = \frac{1}{1+e^{\alpha_{in}\Delta R_{in}}} \tag{4.21}$$

$$= \varepsilon + \varepsilon e^{\alpha_{in}\Delta R_{in}} = 1 \tag{4.22}$$

$$= \varepsilon e^{\alpha_{in}\Delta R_{in}} = 1 - \varepsilon \tag{4.23}$$

$$= \frac{1-\varepsilon}{\varepsilon} = e^{\alpha_{in}\Delta R_{in}} \tag{4.24}$$

$$= \ln\left(\frac{1-\varepsilon}{\varepsilon}\right) = \alpha_{in}\Delta R_{in} \tag{4.25}$$

$$\alpha_{in} = \frac{1}{\Delta R_{in}} \ln\left(\frac{1-\varepsilon}{\varepsilon}\right) \tag{4.26}$$

4.3.2.1.2 Mathematical Solution for α_{out}-Control Variable

In the designed vector field, particles starting outside the ellipse will move toward the center until they reach the boundary of the R^* ellipse. The weights will be in the form of Equation (4.18). For particles outside the R* ellipse, the magnitude of the field vector is dominated by:

$$S_{out}(r) = \frac{e^{\alpha_{out}(r-K_{out})}}{1+e^{\alpha_{out}(r-K_{out})}} \tag{4.27}$$

which can also be rewritten as in Equation (4.14). Equation (4.27) is a monotonically increasing function with a limit value of 1. Picking $K_{out} > R^*$, let $K_{out} = R^* + \Delta R_{out}$ so $S_{out}(r)$ is:

$$S_{out}(r) = \frac{e^{\alpha_{out}(r-(R^*+\Delta R_{out}))}}{1+e^{\alpha_{out}(r-(R^*+\Delta R_{out}))}} \tag{4.28}$$

The parameter α_{out} controls the slope of $S_{out}(r)$ for r in the set $R^* - \Delta R_{out} < r < R^* + \Delta R_{out}$. Since the desired value of $S_{out}(R^*)$ is very small, let ε be an arbitrarily small number greater than 0 such that $S_{in}(R^*) = \varepsilon$. The control parameter α_{out} can then be determined as shown in Equations (4.29) through (4.37).

$$\varepsilon = \frac{e^{\alpha_{out}(R^*-(R^*+\Delta R_{out}))}}{1+e^{\alpha_{out}(R^*-(R^*+\Delta R_{out}))}} \tag{4.29}$$

$$= \frac{e^{-\alpha_{out}\Delta R_{out}}}{1+e^{-\alpha_{out}\Delta R_{out}}} \quad = \varepsilon \tag{4.30}$$

$$= e^{-\alpha_{out}\Delta R_{out}} = \varepsilon + \varepsilon e^{-\alpha_{out}\Delta R_{out}} \tag{4.31}$$

$$= (1-\varepsilon)e^{-\alpha_{out}\Delta R_{out}} = \varepsilon \tag{4.32}$$

$$= e^{-\alpha_{out}\Delta R_{out}} = \frac{\varepsilon}{1-\varepsilon} \tag{4.33}$$

$$= -\alpha_{out}\Delta R_{out} = \ln\left(\frac{\varepsilon}{1-\varepsilon}\right) \tag{4.34}$$

$$= \alpha_{out} = -\frac{1}{\Delta R_{out}}\ln\left(\frac{\varepsilon}{1-\varepsilon}\right) \tag{4.35}$$

$$= \alpha_{out} = \frac{1}{\Delta R_{out}}(\ln(1-\varepsilon) - \ln(\varepsilon)) \tag{4.36}$$

$$= \alpha_{out} = \frac{1}{\Delta R_{out}} \ln\left(\frac{1-\varepsilon}{\varepsilon}\right) \tag{4.37}$$

4.3.2.1.3 Mathematical Proof for Convergence of S_{in} and S_{out} Limiting Functions

Recall Equation (4.14), so *W(r)* can be written as:

$$W(r) = \frac{e^{\alpha_{in}(r-(R^*-\Delta R_{in}))}}{1+e^{\alpha_{in}(r-(R^*-\Delta R_{in}))}} - \frac{1}{1+e^{\alpha_{out}(r-(R^*+\Delta R_{out}))}} \tag{4.38}$$

Since $S_{in}(R^*) = \varepsilon$ and $S_{out}(R^*) = \varepsilon$ then $W(R^*) = 0$. Further it can be shown that for $r < R^*$, $W(r) > 0$ and that for $r > R^*$, $W(r) < 0$.

$$W(r) = \frac{e^{\alpha_{in}(r-(R^*-\Delta R_{in}))}\left(1+e^{\alpha_{out}(r-(R^*+\Delta R_{out}))}\right) - \left(1+e^{\alpha_{in}(r-(R^*-\Delta R_{in}))}\right)}{\left(1+e^{\alpha_{in}(r-(R^*-\Delta R_{in}))}\right)\left(1+e^{\alpha_{out}(r-(R^*+\Delta R_{out}))}\right)} \tag{4.39}$$

The denominator is always positive so the sign is controlled by the numerator. Simplifying the numerator results in:

$$= e^{\alpha_{in}(r-(R^*-\Delta R_{in}))}\left(1+e^{\alpha_{out}(r-(R^*+\Delta R_{out}))}\right) - \left(1+e^{\alpha_{in}(r-(R^*-\Delta R_{in}))}\right) \tag{4.40}$$

$$= e^{\alpha_{in}(r-(R^*-\Delta R_{in}))} + e^{\alpha_{in}(r-(R^*-\Delta R_{in}))}e^{\alpha_{out}(r-(R^*+\Delta R_{out}))} - 1 - e^{\alpha_{in}(r-(R^*-\Delta R_{in}))} \tag{4.41}$$

$$= e^{\alpha_{in}(r-(R^*-\Delta R_{in}))}e^{\alpha_{out}(r-(R^*+\Delta R_{out}))} - 1 \tag{4.42}$$

$$= e^{\alpha_{in}(r-(R^*-\Delta R_{in}))+\alpha_{out}(r-(R^*+\Delta R_{out})} - 1 \tag{4.43}$$

Since $e^0 = 1$, the simplified numerator (and *W(R)*) will be positive if the argument in the exponent is positive. Similarly *W(R)* will be negative if the argument in the exponent is negative.

Substituting from Equation (4.26) and Equation (4.37) into the expression of the exponent:

$$= \alpha_{in}(r-(R^{*}-\Delta R_{in}))+\alpha_{out}(r-(R^{*}+\Delta R_{out}) \tag{4.44}$$

$$= \frac{1}{\Delta R_{in}}\ln\left(\frac{\varepsilon}{(1-\varepsilon)}\right)(r-(R^{*}-\Delta R_{in}))+\frac{1}{\Delta R_{out}}\ln\left(\frac{\varepsilon}{1-\varepsilon}\right)(r-(R^{*}+\Delta R_{out}) \tag{4.45}$$

$$= \ln\left(\frac{\varepsilon}{(1-\varepsilon)}\right)((\frac{1}{\Delta R_{in}}+\frac{1}{\Delta R_{out}})(r-R^{*})+\frac{\Delta R_{in}}{\Delta R_{in}}-\frac{\Delta R_{out}}{\Delta R_{out}}) \tag{4.46}$$

$$= \ln\left(\frac{\varepsilon}{(1-\varepsilon)}\right)(\frac{1}{\Delta R_{in}}+\frac{1}{\Delta R_{out}})(r-R^{*}) \tag{4.47}$$

The expression in Equation (4.47) is positive for $r > R^{*}$ and negative for $r < R^{*}$. Figure 4.10 and Figure 4.11 show symmetric ($\Delta R_{in} = \Delta R_{out}$) and asymmetric ($\Delta R_{out} > \Delta R_{in}$) cases respectively.

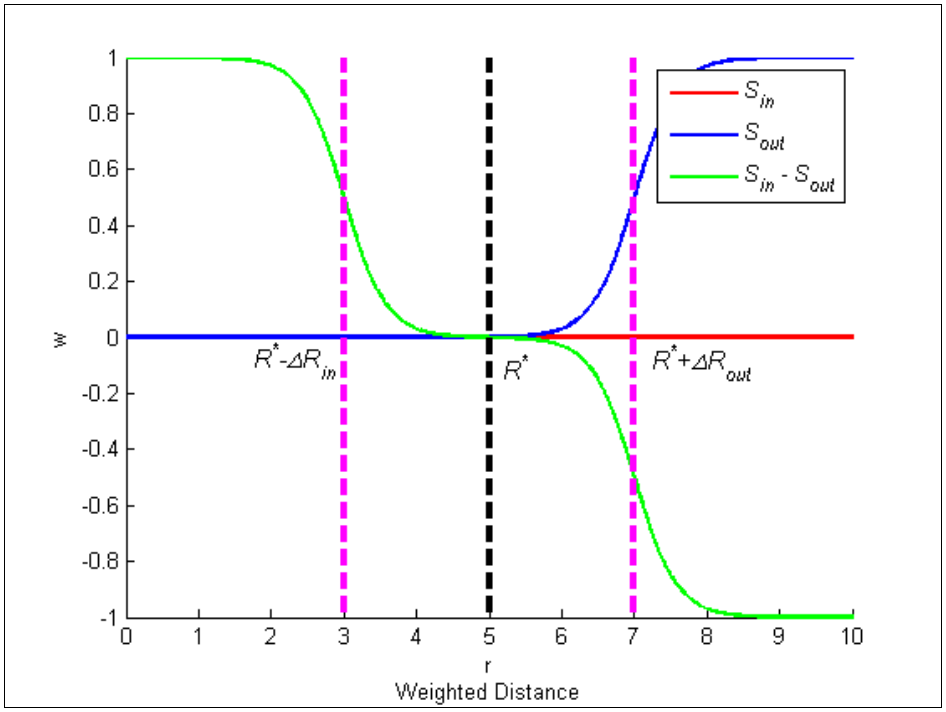

Figure 4.10. Weighting function W(r) (shown in green) when $\Delta R_{in} = \Delta R_{out}$.

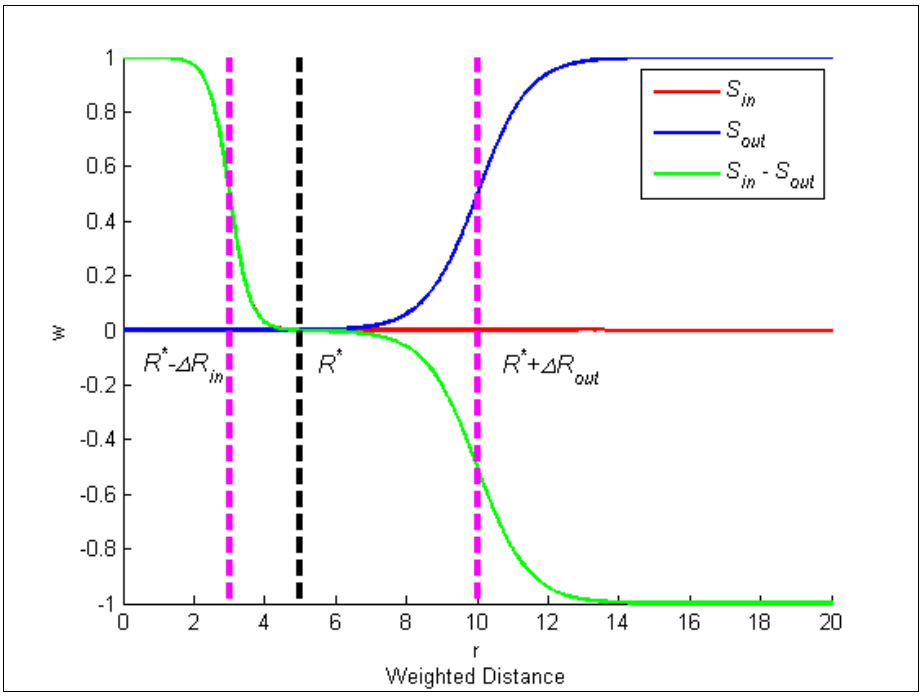

Figure 4.11. Weighting function W(r) (shown in green) when $\Delta R_{out} > \Delta R_{in}$.

4.3.2.2 $N_\perp$ Normal Limiting Function

Attracting the robot to the R^* neighborhood specified in Equation (4.11) is the first step in the construction of the final vector field. Another vector field is needed to control the robots once they are in the elliptical band. In this field, the robots need to move along the ellipse in a field perpendicular to the previously described gradient fields. The influence of these perpendicular fields must be restricted to a narrow band, similar to that described by Equation (4.11). Vectors in this field must die off outside this narrow band. A limiting function accomplishes this is given by:

$$N_\perp(\alpha_\perp, r, R^*) = e^{-\alpha_\perp (r-R^*)^2} \tag{4.48}$$

Figure 4.12 graphs the $N_\perp$ limiting function.

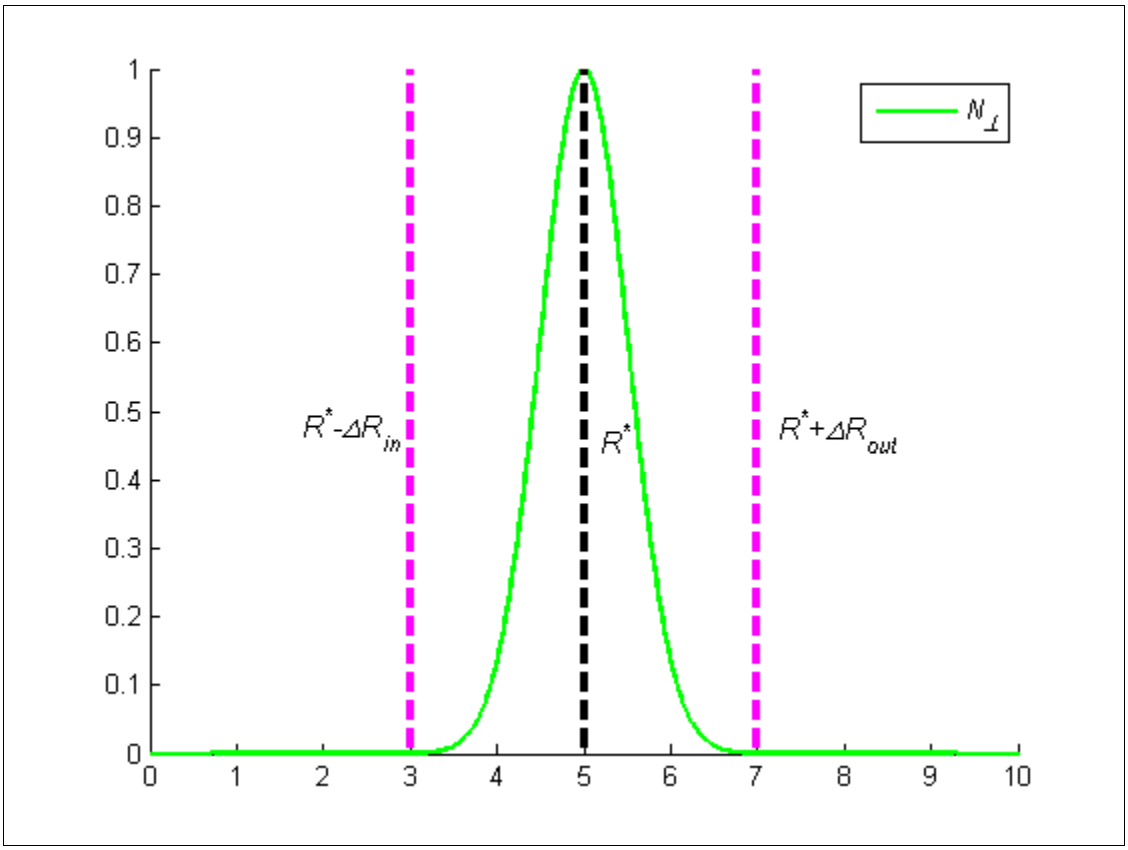

Figure 4.12. The weighting function $N_\perp$ as a function of the weighted distance r defined in (4.17).

In addition, another multiplier to the perpendicular field must be added so the robots do not circle around the elliptical bands as in Figure 4.6. In order for the perpendicular field to change directions, the field perpendicular to the gradient is multiplied by a function which changes the direction of the perpendicular field about the x-axis:

$$SGN(\alpha_\perp, y_{rot}) = 1 - 2.0\left(\frac{1}{1+e^{-\alpha_\perp (y_{rot})}}\right) \tag{4.49}$$

Function $N_\perp$ in Equation (4.48) includes one tuning parameter, $\alpha_\perp$. The parameter $\alpha_\perp$ controls the slope of $N_\perp(r)$ for r in the set $R - \Delta R_{in} < r < R + \Delta R_{out}$. In this case, the parameter will be defined so that the value of $N_\perp(R^* + \Delta R_{out})$ and $N_\perp(R^* - \Delta R_{in})$ can be made arbitrarily small. The resulting formula for $\alpha_\perp$ is shown in Equation (4.52). The same technique is used in the other limiting functions. For the symmetric case ($\Delta R_{in} = \Delta R_{out}$), solving for $\alpha_\perp$:

$$e^{-\alpha_{\perp}\left((R^{*}+\Delta R_{out})-R^{*}\right)^{2}} = \varepsilon \tag{4.50}$$

$$-\alpha_{\perp}\left(\Delta R_{out}\right)^{2} = \ln(\varepsilon) \tag{4.51}$$

$$\alpha_{\perp} = \frac{-1}{\left(\Delta R_{out}\right)^{2}} \ln\left(\varepsilon\right) \tag{4.52}$$

The vector field is depicted in Figure 4.13 is the sum of the three vector fields discussed in this section. Functions S_{in}, S_{out} and $N_{\perp}$ impose additional restrictions and constraints on top of and in addition to the initial swarm function *f(x, y)*. These limiting functions provide a much tighter level of control by limiting and restricting where the vector fields begin and end. The limiting functions, along with vector fields created by the bivariate normal function, may be summed to create swarm movement in formation as a group. When combined, these equations form the velocity and direction of the swarm movement with respect to the center of the swarm, as shown in:

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = (S_{in} - S_{out}) \begin{bmatrix} d_x \\ d_y \end{bmatrix} + SGN * N_{\perp} \begin{bmatrix} d_x \\ d_y \end{bmatrix}_{\perp} \tag{4.53}$$

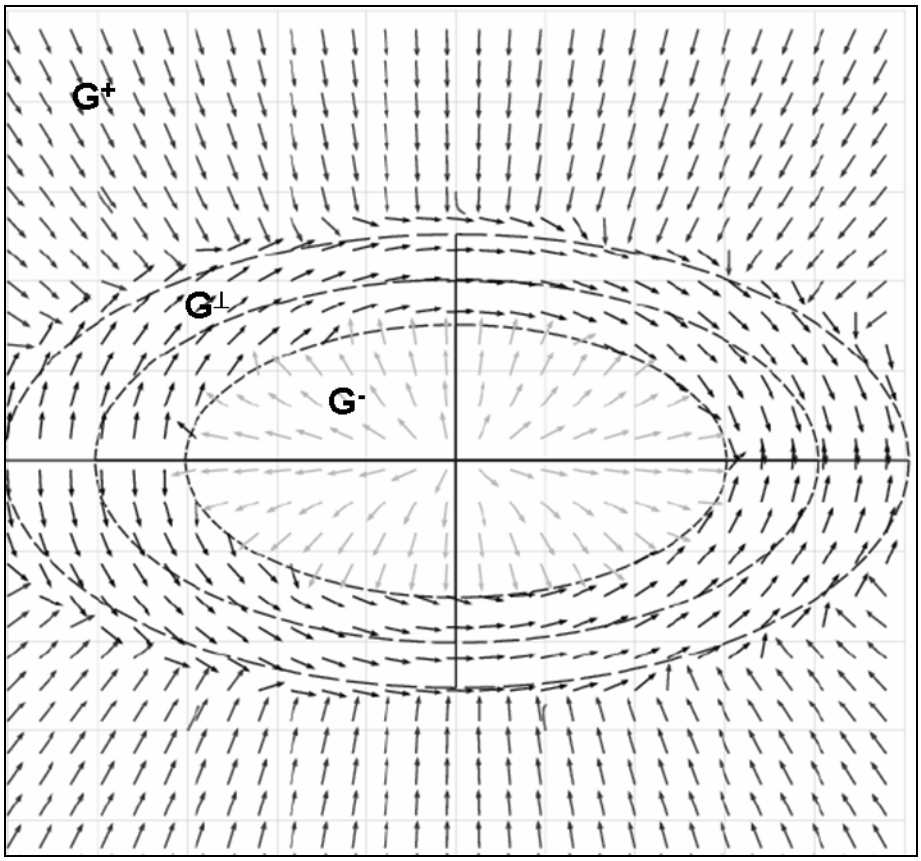

Figure 4.13. Vector field with S_{in}, S_{out} and $N_{\perp}$ limiting functions.

4.3.3 Controlling Swarm Member Dispersion within Bands

Vector fields weighted with sigmoid functions may be used for obstacle avoidance as well as controlling member spacing by creating vectors moving away from the center of the obstacle's or other swarm member's location (x_{co}, y_{co}). For the purposes of this work, the concern is formation including member spacing, so for the purposes of describing the formation control methodology, it is assumed that the only obstacles are other members of the swarm. The same form of limiting function as S_{in} may be used. Obstacle avoidance between members is accomplished using Equations (4.54) to (4.56):

$$r_{avoid} = \sqrt{(x - x_{co})^2 + (y - y_{co})^2} \tag{4.54}$$

$$S_{avoid}(\alpha_{avoid}, r_{avoid}, \Delta R_{avoid}) = 1 - \frac{1}{1 + e^{\alpha_{avoid}(\sqrt{r_{avoid} - \Delta R_{avoid}})}} \tag{4.55}$$

$$\begin{bmatrix} d_{x\_avoid} \\ d_{y\_avoid} \end{bmatrix} = \begin{bmatrix} S_{avoid}(x - x_{co}) \\ S_{avoid}(y - y_{co}) \end{bmatrix} \tag{4.56}$$

Notice that r_{avoid} is similar to r from Equation (4.17) except that instead of distance from the center, the distance to the swarm member is used. The ΔR_{avoid} parameter determines the distance from other members. This parameter determines the dispersion of swarm members in formation. The S_{out} and S_{in} get the robots to the band, but do not control their dispersion.

Avoidance of individual robot swarm members including their dispersion is controlled by the range of influence for the avoidance vector field. The α_{avoid} parameter in Equation (4.24) controls how quickly vector fields die out near obstacles. As α_{avoid} decreases, the influence range of the avoidance vector field increases. By controlling the α_{avoid} parameter, different types of formations can be made within the ellipse bands. Selection and tuning of the ΔR_{avoid} parameter is discussed in Section 4.4.1.

The α_{avoid} parameter is solved for in the same way as the other sigmoid limiting functions in Equations (4.13) and (4.14). The ΔR_{avoid} parameter specifies the minimum distance between robots. Solving for $S_{avoid}(\Delta R_{avoid}) = \varepsilon$ *gives:*

$$\alpha_{avoid} = \frac{1}{\Delta R_{avoid}} \ln\left(\frac{1 - \varepsilon}{\varepsilon}\right) \tag{4.57}$$

The S_{avoid} function can be used to prevent swarm members from colliding with each other. A combination of all of the above fields creates a static formation for the robots; shifting the center of the ellipse as a function of time, creates an overall movement of a group of swarm members as a whole.

The swarm may move from one waypoint to another by moving the center of the ellipse (x_c, y_c). The general equation to create the vector to follow a trajectory with member avoidance by summing the vector fields is given by:

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = (S_{in} - S_{out}) \begin{bmatrix} d_x \\ d_y \end{bmatrix} + \sum_{1}^{\#size-1} S_{avoid} \begin{bmatrix} d_{x\_avoid} \\ d_{y\_avoid} \end{bmatrix} + SGN * N_{\perp} \begin{bmatrix} d_x \\ d_y \end{bmatrix} \tag{4.58}$$

The computational requirements for an individual swarm member are very low, *O(n)*. The computational complexity of the vector generation depends on the number of obstacles because this is the only factor in the equations which has potential to be continuously growing. The complexity will grow in denser environments as well as when the size of the swarm increases because more avoidance vectors must be calculated. This complexity is due to the fact that at each time step, an avoidance vector for *n* obstacles and/ or robots within a certain range must be generated. It is important to note that swarm members do not compute the entire field. They compute a single vector from that field which depends on the center of the ellipse, (x_c, y_c), and the four vectors (in, out, perpendicular, and avoidance) and their corresponding weights.

4.4 Parameter Selection

4.4.1 Logical and Static Parameter Selection for *R*-Parameters

In order to select control parameters, some logic and basic mathematics must be used. The formation must be feasible given the swarm characteristics. These swarm characteristics might include the number of team members; the desired length of the minor and major axes; and the average or maximum length of the robots. The precision at which these *R*-values are chosen will determine how tight and accurate the formation will be. The first step necessary is determining which formation is desired. It is important to note that there is some allowable margin of error when selecting parameters because the swarm members can lie in the area described by Equation (4.11). Parameter selection guidelines for different formations are discussed in the following sections.

4.4.1.1 Ellipse Formation

If the desired formation is an ellipse, the vectors are generated as described in Equation (4.58). The only necessary requirement is that the chosen major and minor axis fit the swarm characteristics. In order to create a circle formation, an equal minor and major axis must be

chosen. In addition, with ellipse and circle formations the ΔR_{avoid} parameter must allow for equal dispersion along the ellipsoid perimeter to actually create an ellipse or circle figure with the swarm. If not, then the robots will tend towards the front or back of the formation shape. A general estimate of the perimeter of the ellipse can be utilized as guideline for choosing ΔR_{avoid}. Given R^*, γ, and N, denoting the number swarm members, the ellipse perimeter can be used as the upper bound in estimating the ΔR_{avoid} parameter. ΔR_{avoid} should adhere to the following at the very least to achieve equal dispersion:

$$\Delta R_{avoid} \le \pi\sqrt{2(R^{*2}+\gamma R^{*2})-(R^{*2}-\gamma R^{*})^2/2}\,/\,N \tag{4.59}$$

In addition, it is also necessary to make sure that ΔR_{avoid} is chosen large enough to avoid the other swarm members. This factor is highly dependent on the obstacle avoidance sensor used.

4.4.1.2 Arc Formation

If the desired formation is an arc or wedge, the formation is as described in Equation (4.27). The parameters are chosen in the same way as the ellipse formation but in order to force the swarm members to the front of the formation it is necessary to choose R* and ΔR_{avoid} so approximately half of the perimeter is empty.

4.4.1.3 Line Formation

4.4.1.3.1 Line Formation with Skinny Ellipse

The line formation still uses the ellipse as the basis but with a slight modification. The S_{in} and $N_{\perp}$ limiting functions are removed in order to trap the robots inside a narrow or skinny ellipse as shown in Figure 4.14 and described by:

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = (-S_{out})\begin{bmatrix} d_x \\ d_y \end{bmatrix} + \sum_{1}^{\#size-1} S_{avoid}\begin{bmatrix} d_{x\_avoid} \\ d_{y\_avoid} \end{bmatrix} \tag{4.60}$$

The length or major axis of the ellipse needs to be long enough to hold all the swarm members, and the width or minor axis needs to be wide enough for the swarm member. If the γ parameter is chosen to small, the swarm members will have a zigzag pattern as they continually overshoot the desired path. If the γ parameter is chosen to large, the swarm members will have an offset line pattern.

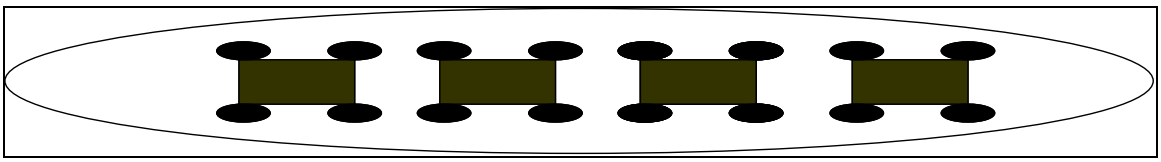

Figure 4.14. Skinny ellipse with swarm members trapped inside.

4.4.1.3.2 Leader-Follower Line Formation

Tighter line formation can be achieved by combining a similar method as in Section 4.4.3.1 with a hierarchical leader-follower approach. In this approach, each robot takes the roll of a leader with the exception of the robot with the tail position in the line. The first robot or highest leader robot is in control of where the swarm travels. Robot 1 follows the same approach discussed for trajectory following in the other formation approaches. Robots 2 to *n* simply follow their leaders as depicted in Figure 4.15 by tracing their paths.

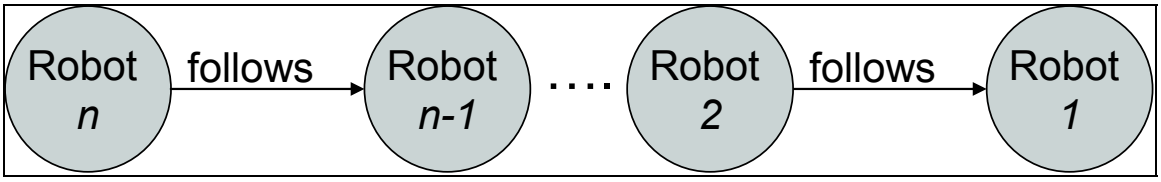

Figure 4.15. Leader-follower line formation approach.

4.4.2 Fuzzy Speed Control and Parameter Selection

Since the swarm members follow the desired behavior to the bands of the ellipse, it is necessary to consider factors such as speed as well as member dispersion. Avoidance of individual robot swarm members and obstacles is accomplished in two ways: i) by controlling the speed at which robots move away when approaching an avoidance vector field, and, ii) by controlling the range the vector field is allowed to exist utilizing the ΔR_{avoid} parameter. The

first method discussed in Section 4.4.2.1 can be achieved via a fuzzy speed controller based on relevant parameters. The second method discussed in Section 4.4.2.2 can be achieved using a fuzzy method to make the static parameter selection dynamic. This fuzzy method makes the ΔR_{avoid} parameter 'tunable'. It is important to note that the swarming methodology discussed above still works without these fuzzy methods, but tuning the parameters with a fuzzy method can result in more optimal swarm behavior over time.

4.4.2.1 Fuzzy Speed Control

In addition to controlling the swarm formation, a desirable feature is speed control based upon distance from elliptical bands as well as distance from other robot members. In order control the speed of the members efficiently, it is only necessary to modify the magnitude of the vectors with a speed parameter which will be denoted by S_{speed}. For this purpose, a Mamdani type fuzzy logic speed controller with two inputs was developed as shown in Figure 4.16.

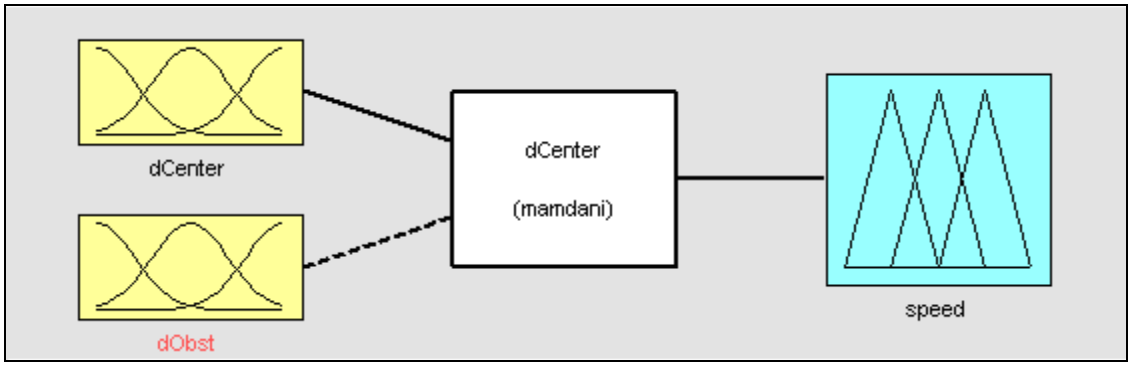

Figure 4.16. Fuzzy speed controller.

Figure 4.16 is the fuzzy speed controller with two inputs: distance from center, *dCenter* and distance to nearest obstacle (including other members), *dObst*. Each swarm member will have an identical but independent speed controller. The output of the speed controller will be a multiplier, the S_{speed} variable, for the final v_x and v_y as shown by:

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = S_{speed}\left[(S_{in}\text{-}S_{out})\begin{bmatrix} d_x \\ d_y \end{bmatrix} + \sum_{1}^{\#obstacles} S_{avoid}\begin{bmatrix} d_{x\_avoid} \\ d_{y\_avoid} \end{bmatrix} + N_{\perp}\begin{bmatrix} -d_x \\ d_y \end{bmatrix}\right] \tag{4.61}$$

The first input, *dCenter*, will be controlled based upon the defined rings of the elliptical bands, R^*, R_{in}, and R_{out}. A generalized description of this input follows in Figure 4.17. The second input, desired distance from obstacles is a user-defined characteristic. The user defines a reasonable range from R_{short} to R_{long} of acceptable distances from obstacles and/or other swarm members. Figure 4.18 describes this input. The values for *a*, *b*, *c*, *d*, and *e* are logically and arbitrarily chosen.

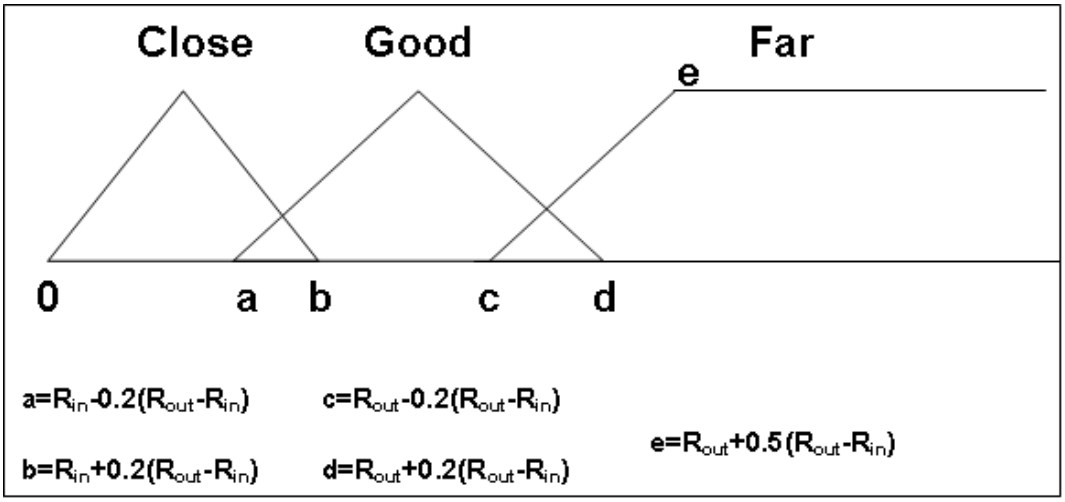

Figure 4.17. Distance from center (*dCenter*) input.

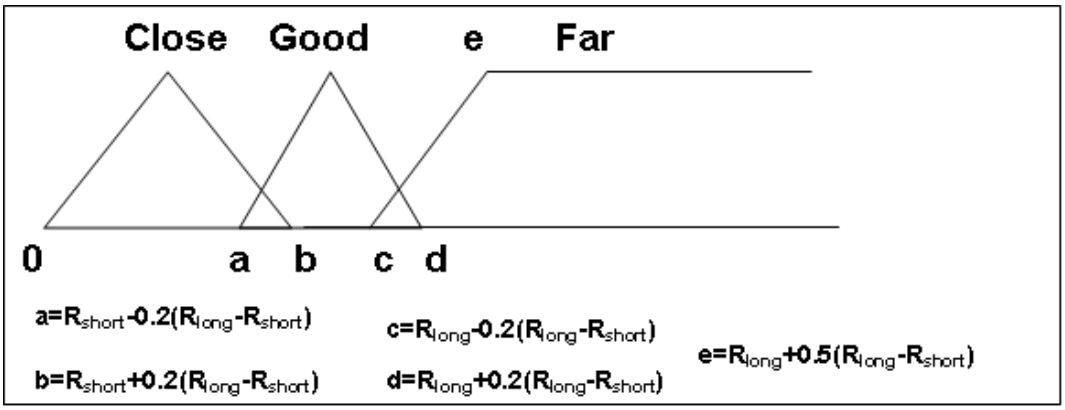

Figure 4.18. Distance from obstacles (*dObst*) input.

In order to describe the control speed controller, the possible speeds are normalized to between 0 and 100 units. Figure 4.19 shows the fuzzy speed output. Figure 4.20 shows the fuzzy rules taken directly from Matlab. Figure 4.21 depicts the surface of the fuzzy function.

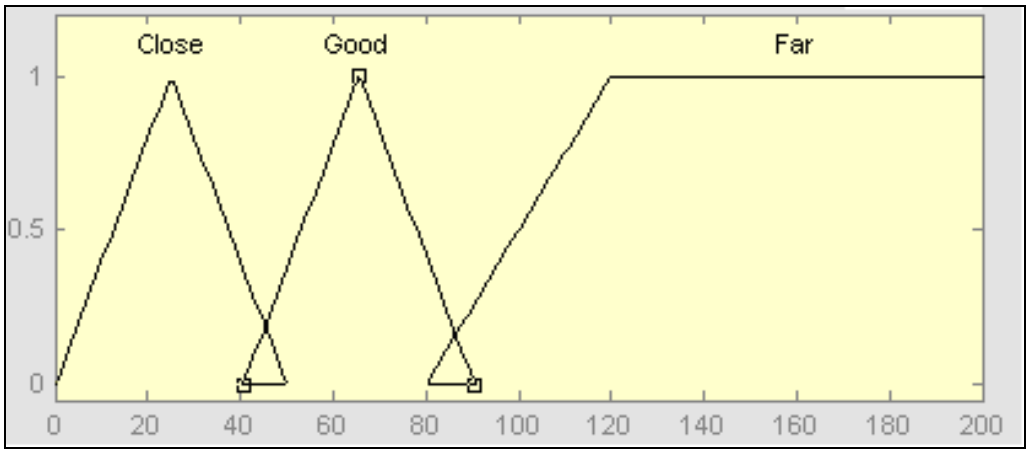

Figure 4.19. Fuzzy speed output.

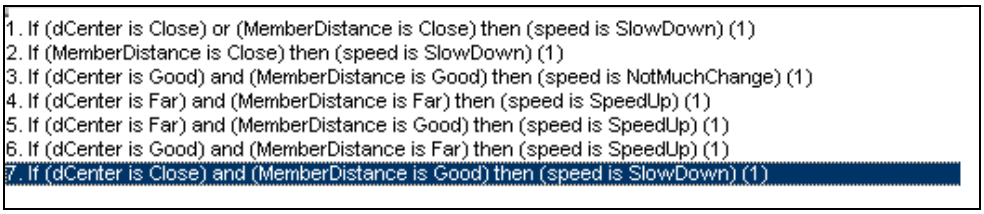

Figure 4.20. Fuzzy rules for speed controller.

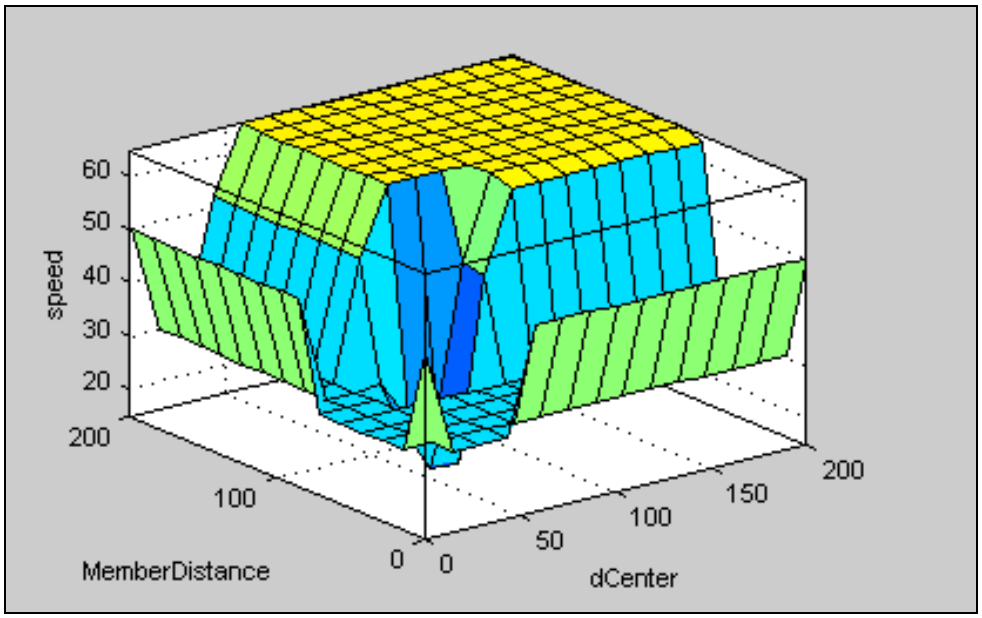

Figure 4.21. Surface function for speed controller.

4.4.2.2 Fuzzy Parameter Selection of ΔR_{avoid}

In order to equally disperse the robots along the swarm surface, the ΔR_{avoid} parameter can be controlled and optimized via a fuzzy function. Although it is the same idea as the S_{in} limiting function, the ΔR_{avoid} parameter needs to be controlled via fuzzy function to get the desired spacing. This parameter needs to be dynamically changing until the robots are at the desired space apart. The user can make a good estimate as to what this parameter should be a priori, but the vector field use in this work is a highly non-linear sum of four dynamically changing vectors so the behavior is not always what is expected. To alleviate this error, a simple fuzzy controller is designed. It is possible to tune the ΔR_{avoid} parameter within reason and still hold to the desired formation.

Only one input is needed for this fuzzy controller, and that is the distance to the nearest neighbor. In order to determine the membership values a user defined desired spacing must be given to the fuzzy controller. Since the desired formation comes in some sort of ellipse, one possible method of selection is via the perimeter of the ellipse and number of robots as described before in Equation (4.59). A similar input to the *dObst* input is used except now instead of controlling the speed near the robots; only the dispersion between the robots is to be controlled. This new variable is called *dMembers*. Figure 4.18 is a generalization of the *dMembers* input. The user again is supposed to give a reasonable range from R_{begin} to R_{end} of acceptable distances from each robot.

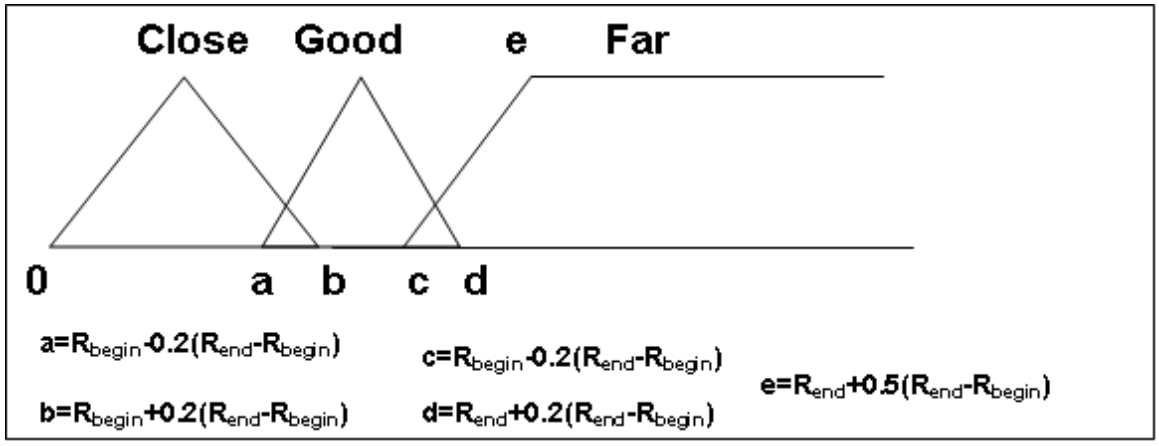

Figure 4.22. Distance to nearest neighbor (*dMembers*).

Figure 4.23 shows the fuzzy output function for the ΔR_{avoid} parameter. Figure 4.24 shows the fuzzy rules taken directly from *Matlab*, followed by the fuzzy surface depicted in Figure 4.25.

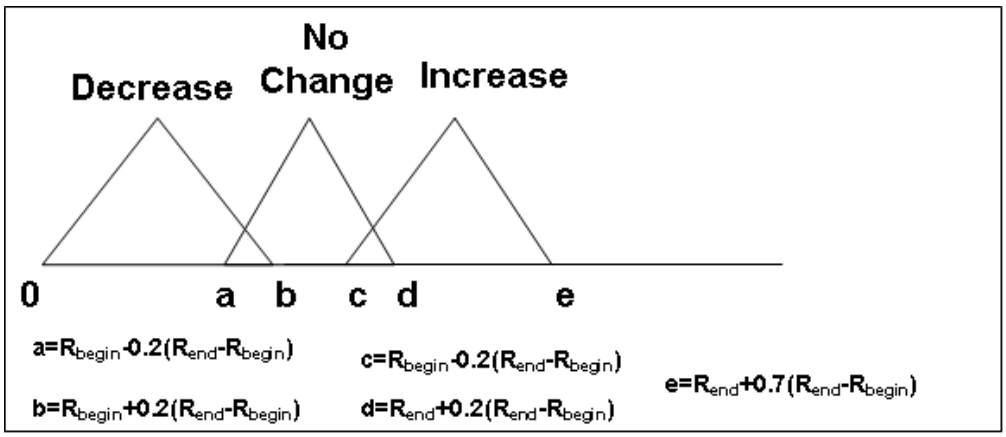

Figure 4.23. Fuzzy output for the ΔR_{avoid} parameter.

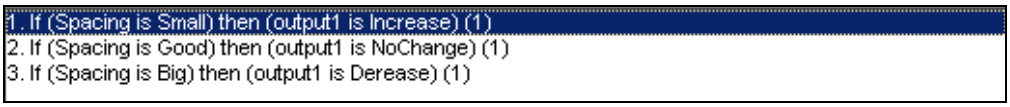

Figure 4.24. Fuzzy rules for the ΔR_{avoid} parameter selection.

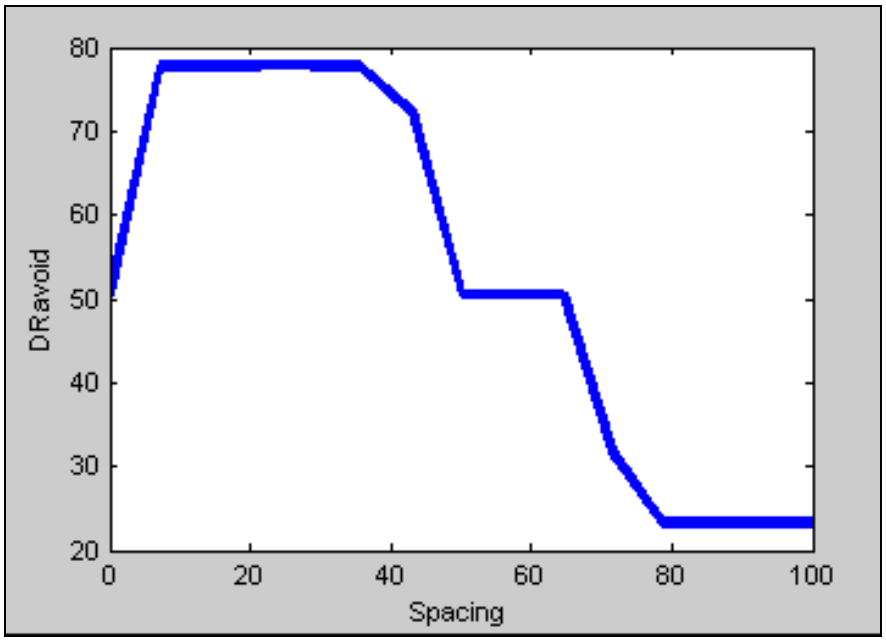

Figure 4.25. Surface plot for the ΔR_{avoid} parameter.

Chapter 5

Simulation Software

In order to test the swarming theory and algorithms before trying them on the actual hardware, rigorous simulations are performed. These simulations allowed for the discovery of bugs in software, algorithms, and the mathematical theory. The robot swarm was simulated in *Matlab* version 7.4 utilizing the *Simulink* toolbox. In order to demonstrate the fuzzy speed control and parameter selection methods, the *Fuzzy Logic* toolbox was also utilized.

5.1 Matlab Simulink Model

Simulink was used to model the robot swarm. *Simulink* is a toolbox developed by Mathworks for modeling, simulating, and analyzing multi-domain dynamic systems. Both actual robot models as well as models of particles are used as vehicles in the swarm model. Ten different robot models are used in combination with vector generation modules to simulate the overall swarm behavior. The vehicle models are discussed in Section 5.1.2.2. Particles are also used to get more precise simulations which are discussed in Section 5.1.2.1. Four and ten particle/robot simulations are run for demonstration of the concept. The swarm formation controller, which is identical for each robot/particle, is programmed in C. Each individual robot's vector generating controller is implemented as a MEX S-function with the different control parameters fed in as well as a position vector with nearby member locations. MEX functions have a return type void and work as an interpreter between MATLAB functions and C code. The vector controller will be described in Section 5.1.1. The overall Simulink model of the swarm system is shown in Figure 5.1.

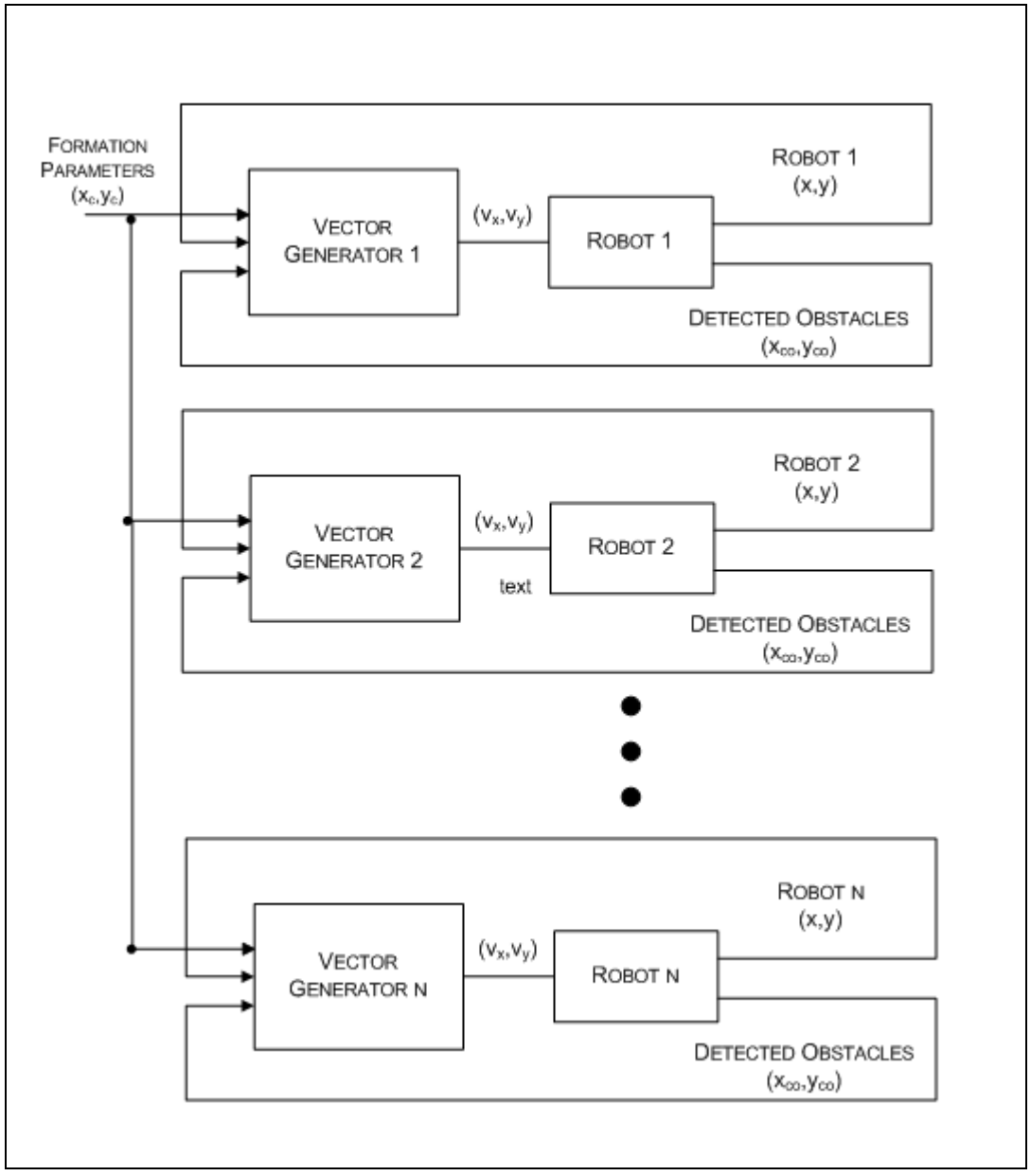

Figure 5.1. Matlab Simulink swarm simulation with *n* robots / particles.

5.1.1 Vector Generator Block

Each of the *n* swarm members has an identical block of C-code for vector generation. Each of the C-MEX S-functions generates the desired vector presented in Chapter 4 at each time step which is then fed into the vehicle model block. The formation parameters can either be set inside of C-MEX S-function or outside in an *m-file*. These parameters can also dynamically change if necessary. Formation changes are easily made by determining which fields will be included in the final vector summation equation and updating just a few tuning parameters. Parameters are selected based on the desired formation and dispersion using methods described in Chapter 4. The center of the swarm, (x_c, y_c) is then broadcast into the vector generating S-functions at the appropriate time steps. In addition, nearby obstacles and swarm member locations are also broadcast in order to compute the avoidance vector. How much information is transferred depends on what type of model is used. If it is assumed that all member locations are known, then global knowledge is broadcast. If it is assumed that only obstacles / swarm members at a reasonable 'line of sight' is known, then this is the only information known to each swarm member.

It is important to note, that knowledge of the other member locations are not a necessity except for the dispersion aspect of the swarm. The way this knowledge is shared could be done in numerous ways making a case for both centralized and decentralized robotic systems, but this is not the relevant point of this work. The robots, assuming identical formation parameters, will hold to the bands of the ellipse regardless of knowledge of each other.

5.1.2 Vehicle Model Block

5.1.2.1 Particle Model

Models of particles are used to simulate the robot swarm vehicles. The particles follow the vectors perfectly whereas robots are limited in their motion. The vectors are integrated to obtain position based only on vectors as shown in (5.1) and (5.2).

$$X(t) = \int_0^t d_x dt \tag{5.1}$$

$$Y(t) = \int_0^t d_y dt \tag{5.2}$$

This allows for verification of the math theory with excellent accuracy because the particles will follow the generated vector fields exactly.

5.1.2.2 Robot Model

A physical robot has much more limited movement than a particle; therefore, a robot model may be implemented with the vector field generator to validate the applicability of the proposed approach for swarm control.

A working model of an RC-car with Ackerman steering has been derived to demonstrate simple controller design (see Fig. 5.2) used in conjunction with the vector fields. Model simplifications relate to neglecting both the losses in the drive train and motor, as well as slippage of the wheels in the kinematics model.

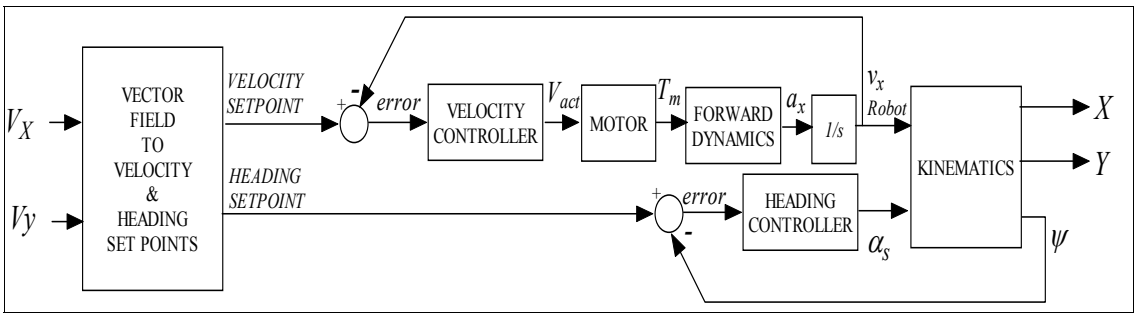

Figure 5.2. Block diagram of car model with feedback.

5.1.2.2.1 Forward Body Reference Dynamics

The primary force on the car is the forward motion due to the torque generated by the motor. Other forces acting on the car such as ground resistance, wind or uneven ground, have not been included in the model. These forces can be overcome by a well designed velocity controller and, therefore, may be neglected in the context of this study. The equation used to calculate the velocity in the forward direction of the robot is given by Equation (5.3). This represents the mass on wheels portion of the model. The force acting on the vehicle is the torque, $T_m(t)$, produced by

the motor increased by the gear ratio between the motor, N_{mw}, the wheels and decreased by the radius of the wheels, r_w, and the mass of the vehicle which is the weight of the vehicle, W, divided by the force of gravity, $32.2\ ft\ s^{-2}$.

$$v_{x\_robot}(t) = \int_0^t \frac{N_{mw} T_m(t)}{(W/32.2) r_w} dt \tag{5.3}$$

Unlike simulated models, robots with the exact same design will not behave identically. In order to reflect this in simulation, some of the robot's constants relating to the physical characteristics were changed between robots. These are given in Table 5.1. Notice that the some of the characteristics have been changed to a large enough degree that the swarm could be considered a heterogeneous swarm.

Table 5.1. Robot physical parameters.

Robot	N_{mw}	r (m)	W (kg)	V_{sp} limit (m/s)	d_w (m)
1	10	0.01524	1.814	6.096	0.1524
2	15	0.02134	3.629	6.096	0.1829
3	18	0.0274	4.536	6.096	0.2134
4	20	0.0305	6.350	6.096	0.2438
5	23	0.0366	7.257	6.096	0.2743
6	26	0.0457	7.711	6.096	0.3048
7	29	0.0549	9.072	6.096	0.3353
8	32	0.0610	11.340	6.096	0.3658
9	35	0.0671	13.608	6.096	0.3962
10	40	0.0762	14.969	6.096	0.4267

5.1.2.2.2 Motor Model

Equations (5.4), (5.5) and (5.6) are used to model the electric motor of the RC-car. The input variable (control variable) is the motor voltage. The output of the motor model is the torque applied to the drive train of the RC-car model. The constants related to motor specifications are listed in Table 5.2.

Table 5.2. Motor parameters.

Symbol	Description	Value
R	Electrical resistance	0.26 mW
L	Electrical inductance	0.1 uH
K_t	Motor torque constant	0.000162 Nm/s
B	Motor dampening ratio	0.0005 Nms
K_v	Motor voltage constant	11.67 radian/Volt-s
J	Motor inertia	0.002 kg-m^2/s^2

The motor variables are the current $i(t)$, the angular velocity $w_m(t)$, the input voltage $V_{in}(t)$ and the output torque $T_m(t)$. The output of the motor is converted to lbs-ft to match the units in the summation of forces in the x-direction.

$$\frac{di(t)}{dt} = -\frac{R}{L}i(t) - \frac{1}{K_v L}w_m(t) + \frac{V_{in}(t)}{L} \tag{5.4}$$

$$\frac{dw_m(t)}{dt} = \frac{K_t}{J}i(t) - \frac{b}{J}w_m(t) \tag{5.5}$$

$$T_m(t) = K_t i(t) \tag{5.6}$$

5.1.2.2.3 Kinematic Calculations

Kinematic equations have been used to convert the motion along the x-body axis to car's position in the world reference frame. Equations (5.7), (5.8) and (5.9) give the kinematics, where $v_{x\_robot}(t)$ is equal to the velocity in the body reference frame, d_w is the distance between the center of the front and back wheels, value given in Table 5.1, $\alpha_s(t)$ is the steering angle of the front tires, and $\psi(t)$ is the heading in the world reference frame. In addition, the steering angle of the car has been limited to +/-30 degrees due to the physical limitations of Ackerman steering:

$$X(t) = \int_0^t v_{x\_robot}(t)\cos(\alpha_s(t))\cos(\psi(t))dt \tag{5.7}$$

$$Y(t) = \int_0^t v_{x\_robot}(t)\cos(\alpha_s(t))\sin(\psi(t))dt \tag{5.8}$$

$$\psi(t) = \int_0^t \frac{v_{x\_robot}(t)}{d_w}\sin(\alpha_s(t))dt \tag{5.9}$$

Control has been implemented by first converting the *x* and *y* vector inputs to velocity and heading set points, then implementing feedback with proportional controllers to maintain the set points.

$$v_{sp}(t) = \kappa\sqrt{v_x^{\,2}(t) + v_y^{\,2}(t)} \tag{5.10}$$

Equation (5.10) is used to calculate the velocity set point. For simplicity of design, a scale factor κ has been set along with a limit given in Table 5.2. The primary objective of calculating the velocity set point from the generated vectors is to slow the movement of the car as the length of the generated vectors decrease. As the robots approach the way point, $v_x(t)$ and $v_y(t)$ approach zero reducing the velocity set point. The scale factor allows for further control over the velocity of the car without altering the vector field generation. For the simulations shown in this study, κ = 10. This parameter increased the velocity of the robot by a factor of ten without the necessity of recalculating the vector field tuning parameters.

The heading set point is controlled by calculating the angle between the *x* and *y* velocity vectors, $v_x(t)$ and $v_y(t)$, generated by the bivariate and normal functions shown in Equation (5.11):

$$\psi_{sp}(t) = \tan^{-1}(v_y(t)/(v_x(t)) \tag{5.11}$$

It is important to note that, the mathematical value for the inverse tangent of infinity is ninety. Some programming languages will not allow this condition. If this is the case, then additional code is required to prevent division by zero.

The velocity control has been implemented with a proportional controller in the body reference frame. This is given in Equation (5.12), where K_P is the controller constant, $V_x(t)$ is the velocity in the body reference frame and $v_{x\_robot}(t)$ is the velocity set point calculated from the generated vector fields:

$$V_{act}(t) = K_P\left(v_{x\_robot}(t) - V_x(t)\right) \tag{5.12}$$

The steering angle, $\alpha_s(t)$, is set to the difference between the vehicles desired heading, $\psi_{sp}(t)$, and the actual heading of the vehicle, $\psi(t)$, in the world coordinate frame:

$$\alpha_s(t) = \psi_{sp}(t) - \psi(t) \tag{5.13}$$

Chapter 6

Hardware Architecture and Platform

The hardware platform for the swarm of robots is four custom-built (in-house) RC-cars equipped with a custom computer control system equipped with GPS and IMU sensors, stereo vision and encoders. The overall hardware system and interconnections is depicted in Figure 6.1. In addition a radio controlled helicopter is utilized in several experiments to demonstrate UAV-UGV coordination. These radio-controlled vehicles will be discussed in 6.1. The sensors used for the swarm experiments will be described in Section 6.2. In Section 6.3 the computer system will be described.

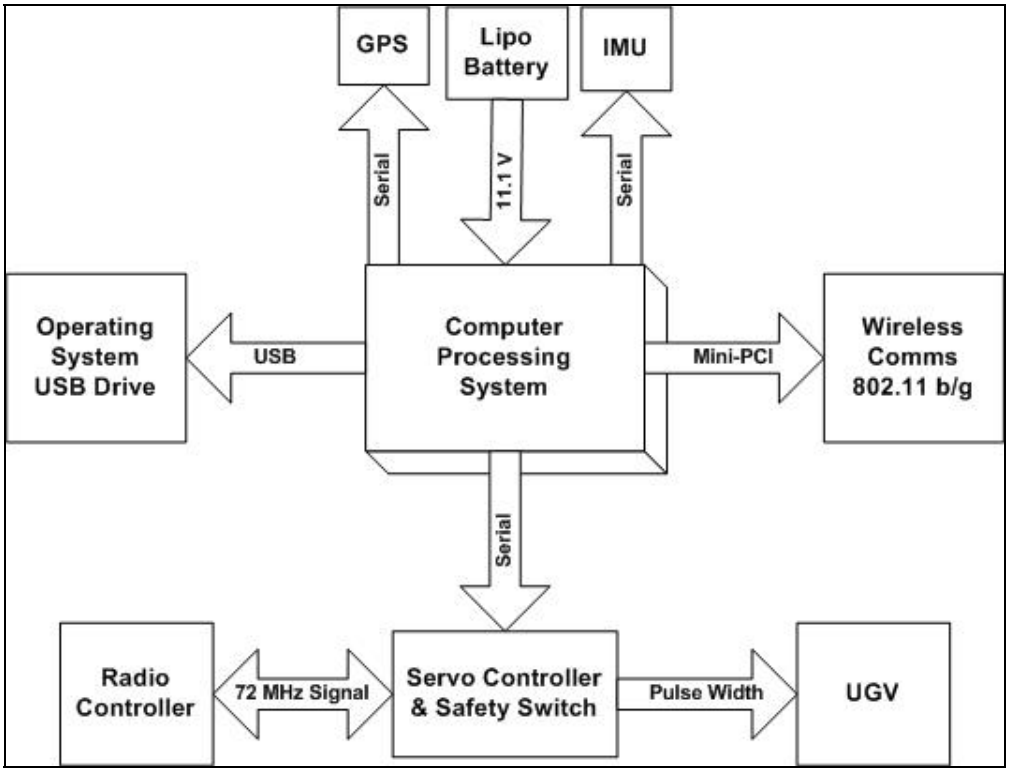

Figure 6.1. Overall hardware system for UGVs.

6.1 Radio-Controlled Vehicles

6.1.1 Radio-Controlled Ground Vehicles

The RC-cars chosen for the swarm vehicles are TRAXXAS E-Maxx Cars. The Emaxx vehicles are Ackerman steered. Figure 6.2 shows the fully upgraded USL unmanned ground vehicles. The vehicles were painted different colors in order to run vision experiments from an aerial camera. These vehicles are also equipped with the USL second generation controller box described in [114, 115] and sensors including a GPS (global positioning system) an IMU(inertial measurement unit). The vehicles are powered by two 7.4V 4200 mAh lithium polymer batteries while the control box and sensors are powered through an 11.1V 4200 mAh battery. The vehicle can run anywhere from *45* minutes to *2* hours depending on the level of usage. Each vehicle platform includes upgraded brushless motors and an upgraded suspension system capable of handling the weight of the control box, sensors, cameras, and pan/tilt unit. The vehicle also includes a stereo pair of Sony block cameras. Figure 6.3 shows the vehicles with labeled components.

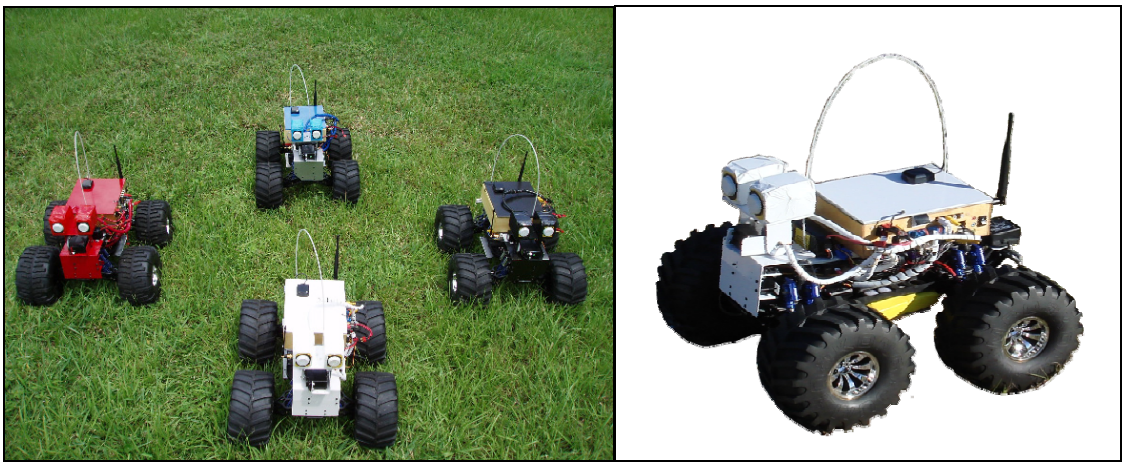

Figure 6.2. Custom-built RC-cars.

The RC–car is controlled via Pulse Width Modulation (PWM) servos. In addition, the swarm vehicles are equipped with the Microbotics Servo/Switch Controller (SSC). Since it is expected that software and/or hardware systems may fail during the development process, the vehicles are equipped with a safety switch. This is a very important safety feature especially when you are

working with several vehicles at a time and need to be able to control them all with a single switch. The hardware component allows the ground vehicles to be taken out of autonomous operation any time for any reason. When the switch is reset, control is transferred back to the user. The radio control receiver transmits control signals from the operator to the SSC and control whether a vehicle is in autonomous or manual operation.

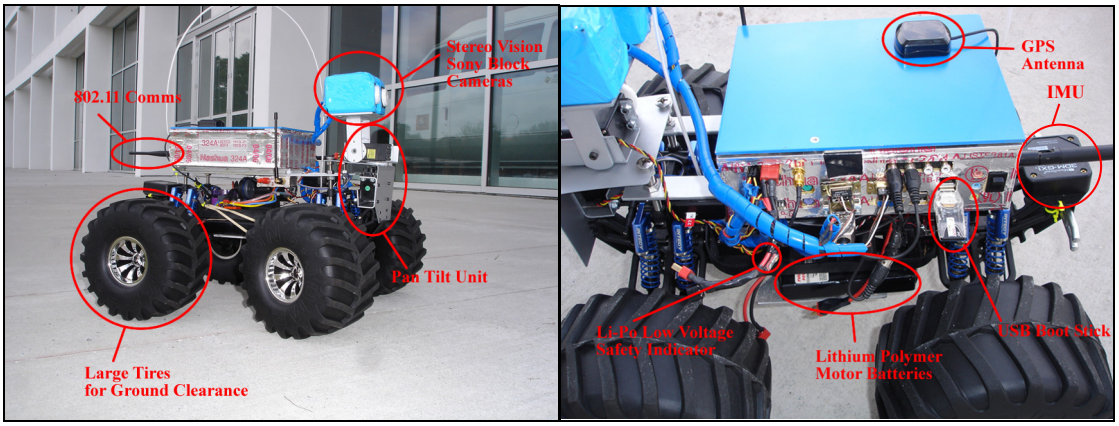

Figure 6.3. RC-car components.

6.1.2 Radio-Controlled Helicopter

In order to demonstrate UAV-UGV swarm coordination an autonomous RC helicopter is also utilized. The Maxi Joker 2 shown in Figure 6.4 is an electric helicopter capable of lifting approximately 10 pounds of payload and flights of between 10 and 20 minutes. The Joker is powered by lithium polymer batteries with separate batteries powering the same second generation controller box and safety switch that is on the ground vehicles. The Joker utilizes a custom set of skids with an incorporated pan tilt unit for a Sony block camera. Sensors include a GPS unit, an IMU, and laser.

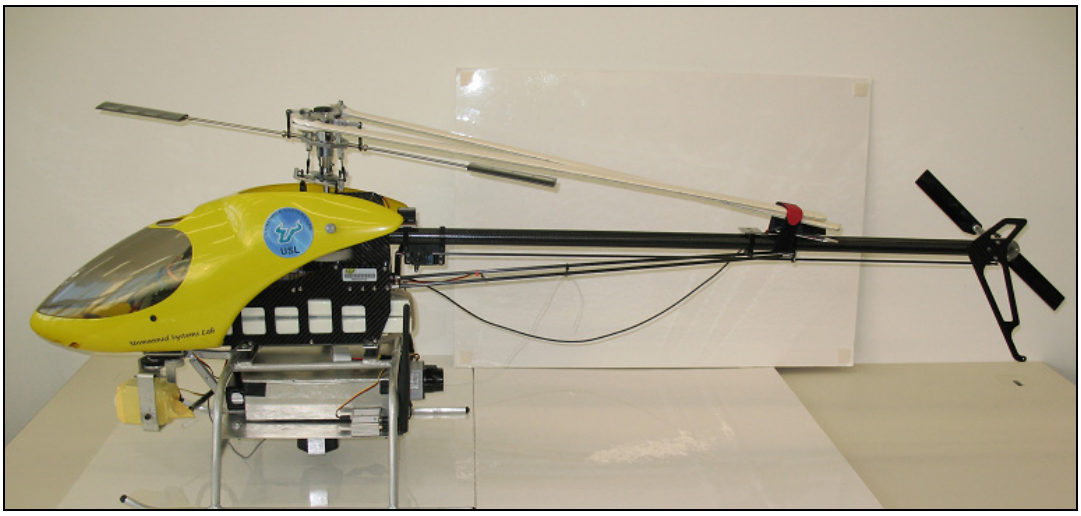

Figure 6.4. Maxi Joker 2 helicopter.

6.2 Sensors

In order to obtain vehicle state data, particularly orientation and position, two sensors are utilized. Orientation or the vehicle heading is collect with an inertial measurement unit. Positional Information is gathered utilizing a global positioning system to get latitude and longitude coordinates. In Figure 5.5a the Microstrain 3DMG-X1 IMU is shown, and in Figure 5.5b the Superstar II GPS receiver is shown. The IMU collects data at the rate of 100 Hz and the GPS at *5* Hz.

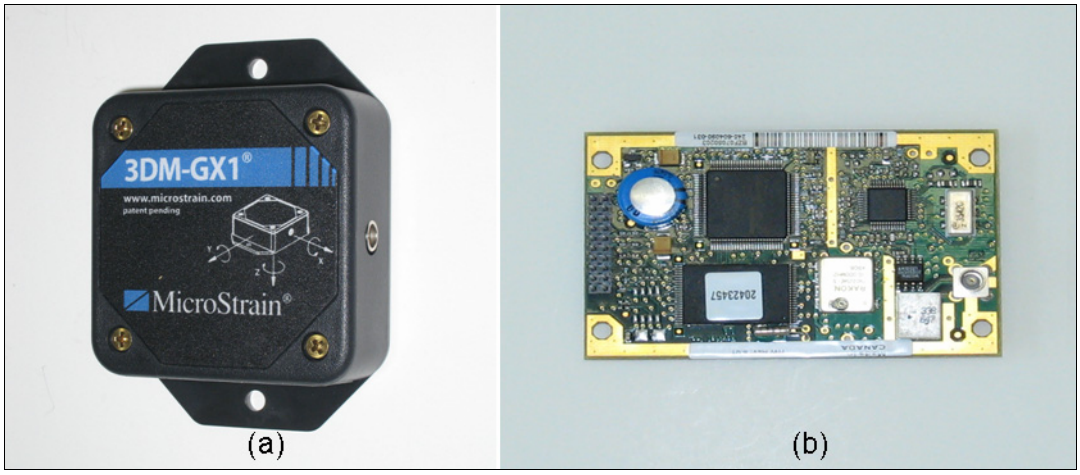

Figure 6.5. Sensors. (a) Microstrain 3DMG-X1 IMU and (b) Superstar II GPS.

6.3 Computer System

The USL generation II control box (Figure 6.6) incorporates features to allow autonomous operation on aerial and ground platforms. The system, weighing *2.5* pounds, is composed of a 2 GHz Pentium mobile chip, mini-ITX motherboard, 2 GB of memory, Superstar II GPS receiver unit, Microbotics safety switch, Intel wireless mini PCI card, and 4 port video capture card. The control box is power by a single 11.1 V, 4200 mAh lithium polymer battery. At full processing power, this battery will power the control box for approximately *45* minutes.

The position sensors can be easily interfaced to the box on any platform by integrating the IMU and attaching the GPS antenna. The GPS and IMU both have a serial interface.

The system is booted from a USB memory stick, which can be removed after the operating system is loaded into memory. This USB stick has a compressed version of Slackware Linux.

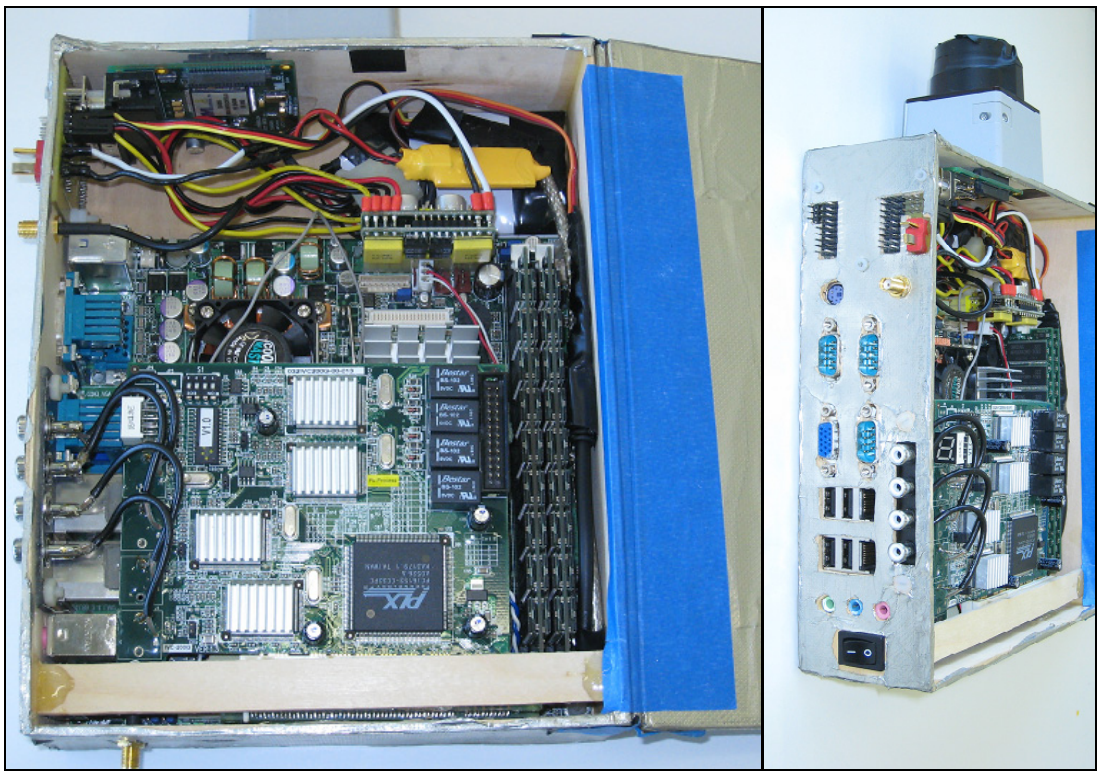

Figure 6.6. On-board computer processing system.

Chapter 7

Software System Architecture

In addition to having a custom built hardware platform, this robotic system is equipped with a highly customizable and modular software system. Most of the components can also be utilized on the UAV described in Chapter 6. This chapter will cover the software from the ground up. First, the operating system will be briefly described. Then the software architecture will be described including robotic controllers, sensor modules, the communication module, and finally the swarm formation controllers.

7.1 Operating System

The operating system provides the basis for any software architecture and can determine how well software runs or how it is written. For this reason, a Linux platform was chosen for all vehicles. The particular distribution is Slackware Linux version *10.0* with the *2.6* kernel. The actual setup of the operating system was performed on a desktop development system. After development was complete, the software was ported to the USB drive that is in the vehicle.

The Linux installation was minimized as much as possible to include only the necessary support for devices and needed software. This is because the operating system is booted into RAM from the USB drive discussed in Chapter 6, and there must be adequate space left for operating processes. In addition, the kernel also had to be recompiled to allow support for the on-board computer's wireless network card.

The software also needs to be specifically configured for communication. An ad-hoc communication scheme was chosen. Mobile ad-hoc networking allows the network nodes or vehicles to exchange information in a wireless environment without the need for a fixed infrastructure. An open source package called *Mobile Mesh* was selected. This software automatically determines the best route to any node on a dynamic network as shown in Figure 7.1. More information on the operating system and installation can be found in [115].

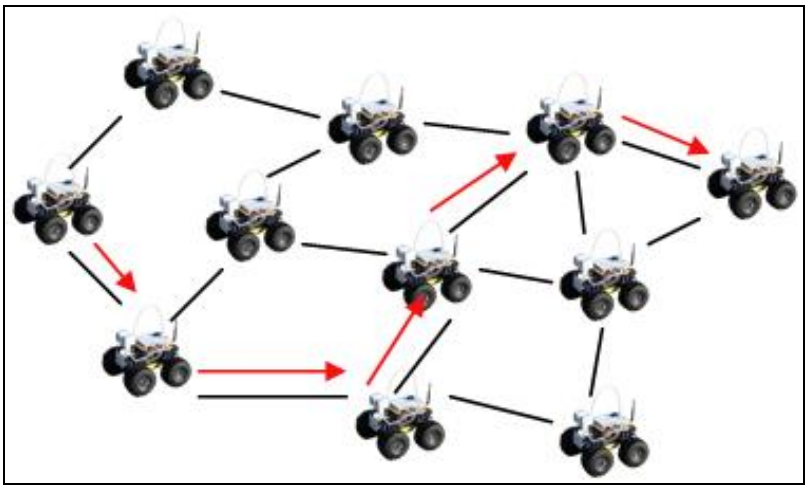

Figure 7.1. Ad-hoc communication network utilizing Mobile Mesh.

7.2 Software Architecture

The section describes the structure of all the software used to interface with the swarm vehicles. All software was written by USF personnel. A high level depiction of all software components and their interactions is shown in Figure 7.2. All components will be discussed in more detail in the following sections.

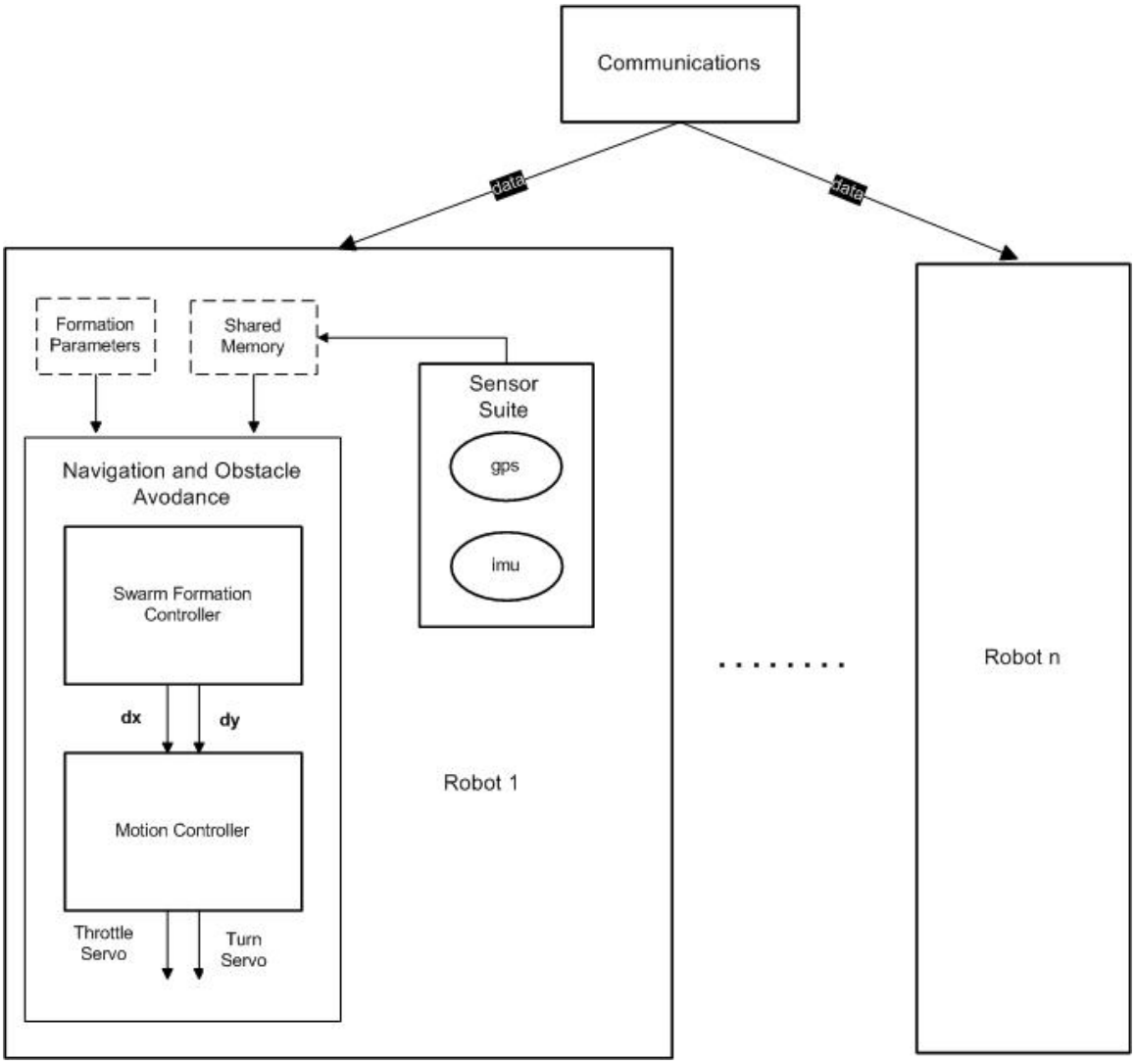

Figure 7.2. Overall software system architecture.

All of the source code is developed in the C programming language. The software is designed to be modular so it can be easily ported to other types of operating systems and vehicles. The software modules are easily reused and integrated. Many of the components have also been used the UAV as well as other UGVs. The software is a set of processes that run concurrently in the background. Information is passed via shared memory structures. Since each process is a single entity, it can be started and stopped as needed. This modular design is also more failsafe than a single module design. If a single process fails, it does not cause all processes to fail.

The actual source code for the unmanned vehicles is created and compiled on the ground station laptop, a Dell Latitude *D820*. This laptop has Fedora Core 6 Linux distribution installed. Once the source code is compiled, the executables are uploaded with the *secure copy* function (*scp*) to the unmanned vehicles. Figure 7.3 shows a detailed diagram of the source code. Each box denotes a different folder of source. Level 1 contains the root folder, the compilation file and all executables. At Level 3, each folder is responsible for a single process.

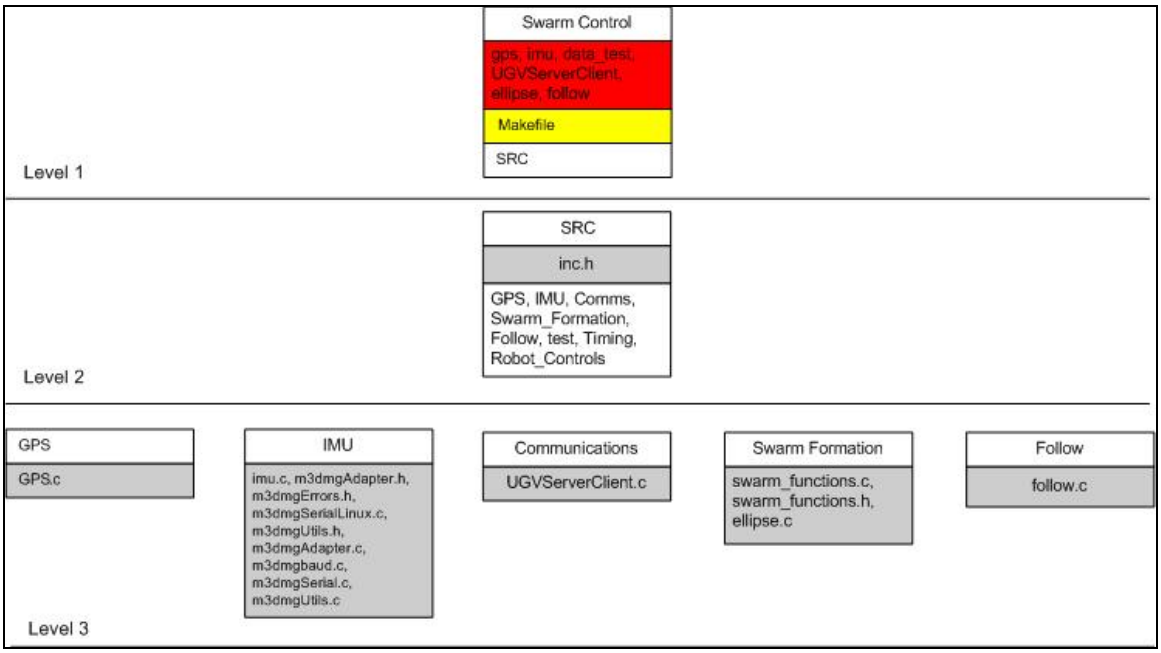

Figure 7.3. Source code directory and file structure.

7.2.1 Sensor Suite

7.2.1.1 GPS Sensor

The GPS process is responsible for reading and parsing data from the GPS receiver and placing the data in shared memory. The GPS process creates a serial connection with the GPS receiver. After the connection is established, the GPS process creates a shared memory location for the GPS data structure. In addition, a corresponding semaphore for controlling mutual exclusion must be created. A semaphore is a protected variable which restricts access to shared resources. The shared resource in this case is the GPS data. The GPS process then enters an infinite loop which will continuously read, parse, and store the GPS data. The GPS receiver is configured with StarView (a setup tool that came with the GPS unit) to continuously output the NMEA GPGGA message at 5 Hz to the serial port.

The GPS process continually updates shared memory. Table 7.1 shows the GPS structure stored in shared memory. The final value recorded to shared memory, *Count*, is an integer variable which increments every time it receives a new string of data from the GPS receiver. The counter variable allows all processes accessing the GPS shared memory structure to determine if the information available is older or newer than the data they already have.

Table 7.1. GPS shared memory data structure.

Shared Memory Variable	Type
Latitude	double
Longitude	double
Direction Latitude	char
Direction Longitude	char
Altitude	Float
Satellites	Integer
Lock	Integer
Count	Integer

The GPS process is also responsible for informing the operator about the state changes of the GPS positional data. The state of the GPS data is determined by the type of lock acquired. The *lock* refers to the level of accuracy of the data being received. The state of lock can be (1) no lock, (2) lock without WAAS correction, or (3) lock with WAAS correction. This notification is simply printed to the screen.

7.2.1.2 IMU Sensor

The IMU process is responsible for accessing the IMU sensor. The IMU process creates a serial connection to the IMU. After the connection is established, the IMU process creates a shared memory location for the IMU data structure. In addition, a corresponding semaphore for controlling mutual exclusion must be created. Then, the process then enters and infinite loop which is responsible for gathering stabilized Euler angles, angular rates, and accelerations. These values are gathered at the rate of 80 Hz and written to shared memory. Table 7.2 shows the IMU structure stored in shared memory.

Table 7.2. IMU shared memory data structure.

Shared Memory Variable	Type
angles [3] roll,pitch,yaw	Float
accel [3] xyz	Float
angRate[3] xyz	Float
Count	Integer

7.2.2 Navigation and Obstacle Avoidance

Navigation and obstacle avoidance are achieved by taking the swarm formation control methodology described in by taking the simulation source code discussed in Chapter 5 and making slight modifications to make it run on the actual robot.

As shown in Figure 7.2, the formation parameters, sensor data, and communicated data are input into the navigation and obstacle avoidance block. After all the shared memory is created and all necessary connections are made the main navigation loop for formation control runs. Pseudo code for the navigation and obstacle avoidance block is shown in Figure 7.4.

NavigationController(my_id, num_robots, x_c, y_c, *formation_parameters[])

```
while(swarm!@goal)
{
        check_dead_robots();
        compute_weights(formation_parameters);
        get_gps(&latitude, &longitude);
        get_imu(&yaw);

        if (gps_lock < 1)
                robot_drive_command(neutral_servo);

        convert_gps_utm(lat, lon, &northing, &easting);

        for(i=0;i< num_robots; i++)
        {
                get_data(vehicle_id, &vehicle[i].lat, &vehicle[i].lon);
                convert_gps_utm(vehicle[i].lat, vehicle[i].lon, & vehicle[i].northing, &vehicle[i].easting);
        }

        get_swarm_center(&xc, yc)
        compute_vectors(num_robots, northing, easting, xc, yc, *vehicledata, &dx, &dy);
        compute_desired_heading(dx, dy, yaw, &des_heading);
        compute_turn_servo(dx, dy, &turn_servo);
        compute_throttle_servo(dx, dy, &throttle_servo);
        robot_drive_command(turn_servo, throttle_servo);
}
```

Figure 7.4. Pseudo-code for navigation of a single swarm member.

7.2.2.1 Swarm Formation Controller

While the vehicles have not reached the goal, the navigation loop will continue to run. Each loop iteration checks to see if there has been a failure with the function *check_dead_robots*. The failure of a robot in these experiments is signified by loss of communication. A failure is simulated by killing a swarm member's communication server. After checking for failure, the weights are computed from the input formation parameters and sensor data is read from shared memory.

GPS coordinates are converted from Latitude/Longitude to Universal Transverse Mercator (UTM) coordinates. UTM is a rectilinear mapping system in map coordinates are represented as Cartesian coordinates and distance is calculated using Euclidian distance measures. Once this conversion has been made, the UTM values are utilized in the vector field generation. The mathematical equations for the conversion from GPS to UTM can be found in [116].

After the sensor data is read and converted, the swarm center is read, and the vectors are computed.

For obstacle avoidance in all experiments it is assumed that the only obstacles are the other swarm members. Each swarm member location is utilized to create the obstacle avoidance vector. Global knowledge of other swarm members' positions is not a necessity but since GPS is the only sensor available for obstacle avoidance, it was necessary for the field experiments.

At the end of the loop iteration, the vectors d_x and d_y are fed into the robot motion controller. This robot motion controller is described next.

7.2.2.2 Robot Motion Controller

The robot motion controller block is responsible for converting the generated vector, d_x and d_y, from the navigation and obstacle avoidance block and converting it into servo values that will perform the desired motion on the robot.

Each robot has different pulse width limits for the servos which define the minimum, maximum, and neutral values for the throttle and turn servos. These values are set in a definitions file and loaded at run-time. The generated vector is converted into a value in the allowable range. Once the throttle and turn servo values are generated, a command is sent to the robot servos.

7.2.3 Communication Server / Client Model

In order to exchange data between the swarm members, a communication protocol is necessary. Mobile Mesh takes care of the routing of the data, but a server and a client still need to be setup to send and receive data. The communication model for the swarm of unmanned vehicles is a simple server / client model utilizing the UDP protocol.

The communication module runs in the background like the IMU and GPS processes. Each vehicle runs a single server. The server is a single process which listens, sends, and receives all incoming data from the other swarm members. A single thread is created inside the server which is responsible for sending the other swarm members its positional data.

Data storage is handled in the same was at it is for the GPS and IMU. Each robot has a shared memory storage location for each of the other swarm member. When a data packet is sent, an identification tag denoting which vehicle it came from is attached. When this identification tag is read, the data will be stored in the appropriate shared memory location.

In addition to listening, sending, and receiving data, the server also keeps track of how long it has been since it has received data from each swarm member. In the shared memory structure, each swarm member has an integer bit denoting whether that particular member is *dead* or *alive*. This bit can carry the value 0 or 1. A value of 0 denotes that that particular member is *alive* and 1 denotes that it is *dead*. If the server has not received data in a specified amount of time, then it marks this value as 1. This bit allows the other swarm member to determine if that vehicle has failed, and the swarm can move on without this member.

Chapter 8

Simulation Results

The proposed method is demonstrated by simulation using both robot vehicles modeled after an RC-car with Ackerman steering and a simple particle model. The simulation method is described in Chapter 5. Simulations are performed with up to ten robots. Ellipse, circle, line, and arc formations are demonstrated.

8.1 Simulations with Robot Model

A set of simulations are run on homogenous and heterogeneous swarms to demonstrate the validity of the approach including the effects of the limiting functions. In addition, the diversity of the swarm demonstrates that the proposed method is platform independent. For experiments focusing on a heterogeneous swarm of robots, the robot parameters are listed in Chapter 5, Table 5.1.

8.1.1 Ten Heterogeneous Robots Circling a Point

Figure 8.1 demonstrates a swarm of ten heterogeneous robots circling around a stationary point, with the robots starting at points close to the center and moving to achieve the desired trajectory. Table 8.1 summarizes the values used for the control variables. Figure 8.1 demonstrates that all robots eventually followed the same circular path, but because of the obstacle avoidance fields, they were at different points of the circular path at the same time instant.

Table 8.1. Control variables with ten robots.

Control Variables	Circling Center Point	Following Trajectory: No Limiting Functions
R*	30	30
γ	1	0.5
ΔR_{out}	0.2	-
ΔR_{in}	0.001	-
ΔR_{avoid}	5	5
ε	0.001	.001

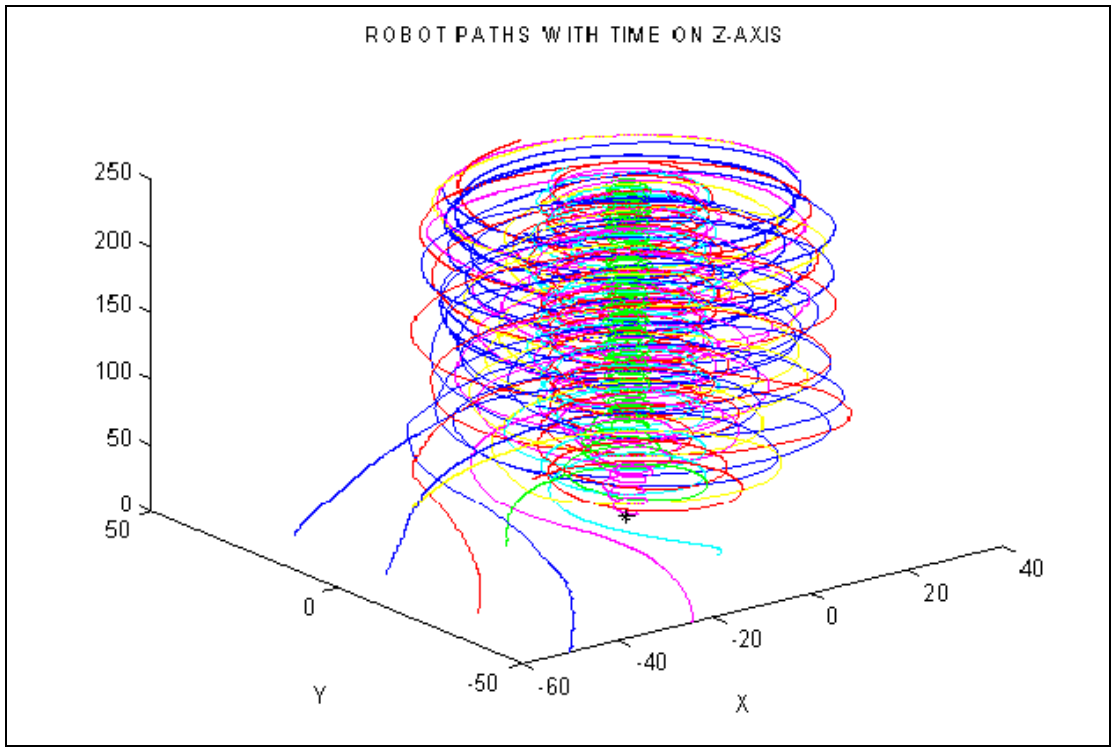

Figure 8.1. Ten heterogeneous robots circling a fixed center point.

8.1.2 Ten Homogeneous Robots Following a Straight Trajectory without Limiting Functions

Another experiment relates to implementing control of ten homogeneous robots following a straight trajectory without utilizing the limiting functions to control the swarm. This is essential to show the importance of the limiting functions in formation control and the necessary change in obstacle avoidance parameters when swarm members are closer. The robots start initially at spread distances and approach the center, coming together into a cluster with different distances apart depending on the α_{avoid} parameter.

Figure 8.2 shows the ten robots following a straight trajectory with time plotted on the *z-axis*. The robots avoid each other and follow the moving bivariate function with center $x_c(t)$ and $y_c(t)$ given at different time steps. Figure 8.3 shows the emergent flocking behavior that results when the limiting functions are not used. Figure 8.4 shows the robots' paths with stationary obstacles placed throughout the mission. The robots avoid each other as well as the obstacles throughout navigation.

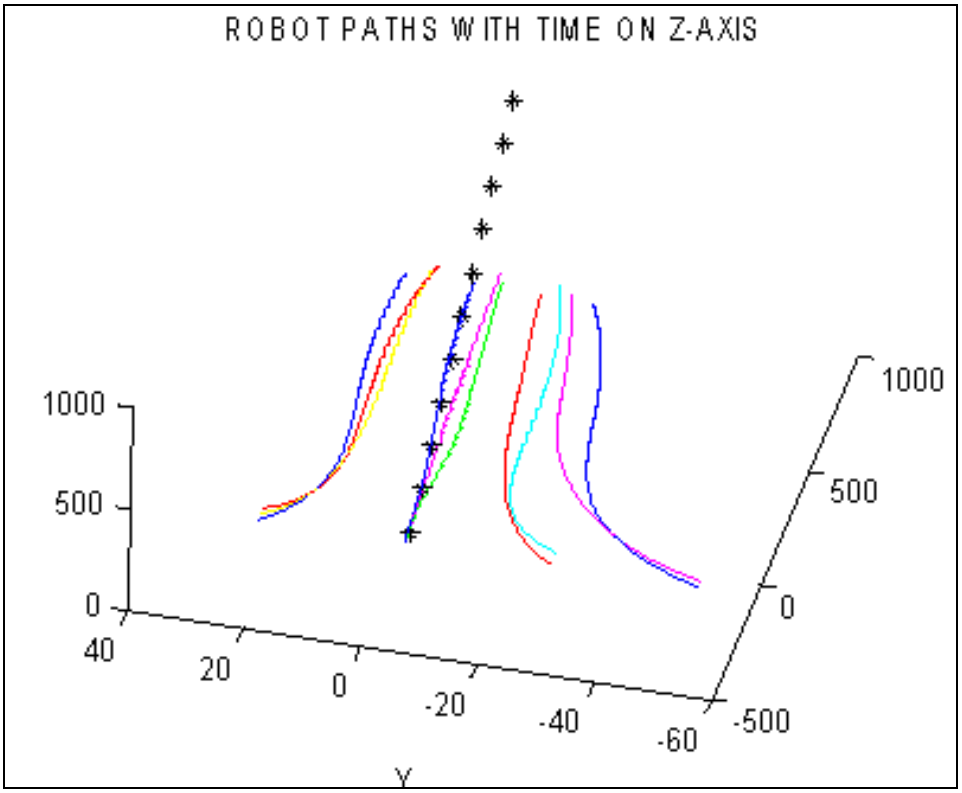

Figure 8.2. Ten robot swarm following a trajectory with time on the *z*-axis without limiting functions.

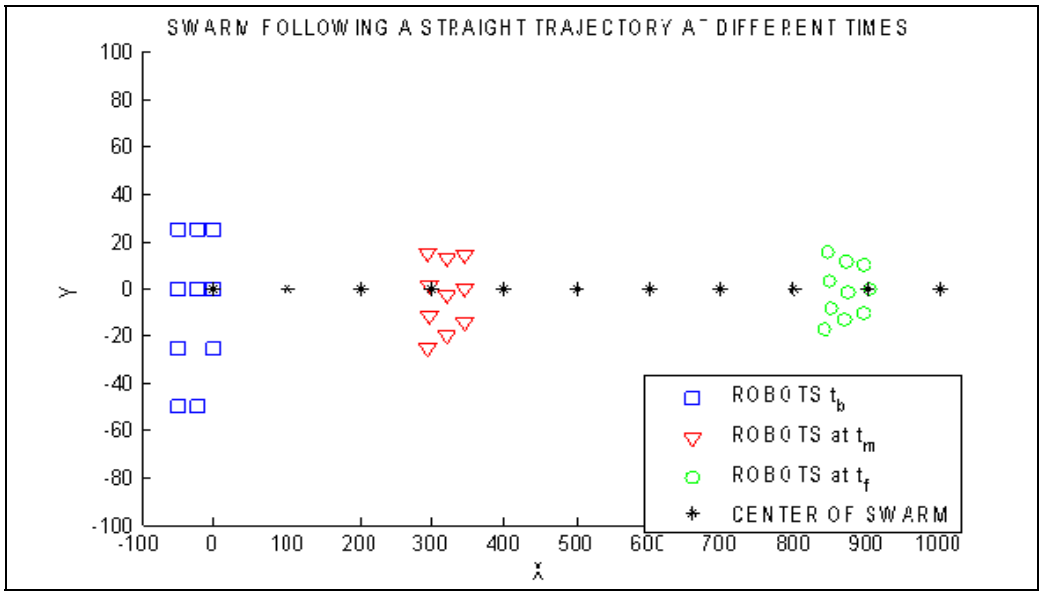

Figure 8.3. Ten robot swarm at beginning (t_b), middle (t_m), and end (t_f) of mission without using limiting functions.

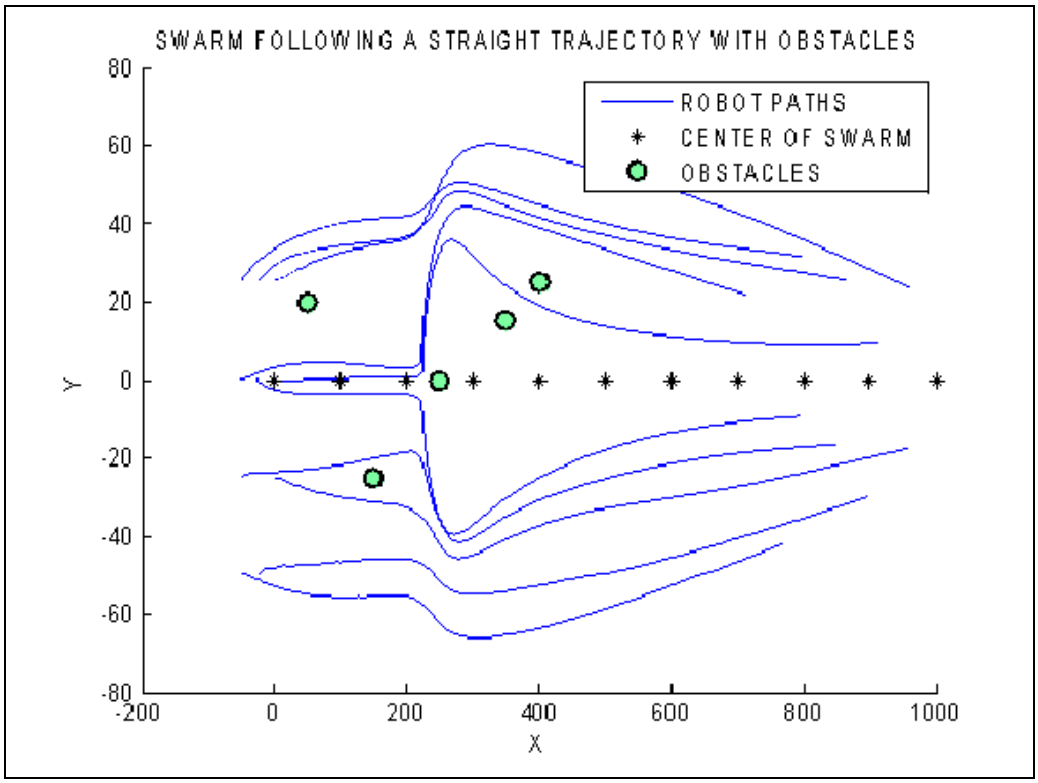

Figure 8.4. Ten robot swarm following a trajectory avoiding fixed obstacles without limiting functions.

8.1.3 Ten Heterogeneous Robots in a Line Formation

Simulations demonstrating a swarm of ten heterogeneous robots moving into a line formation are presented. In order to force the robots into a line formation γ must be very small so the surface of the ellipse function from Equation (4.1) is long and skinny. In order to force this formation Equation (4.60) is utilized. The fields then force the robots to line up to adhere to the function parameters. The parameters for the swarm function as well as the limiting functions are given in Table 8.2.

The robots form a line avoiding each other and successfully followed the desired trajectory. All ten robots were slightly different but used the identical vector generation code. The other parameters were fixed. Figure 8.5 demonstrates the swarm moving into line formation.

Table 8.2. Control variables with ten heterogeneous robots.

Control Variables	Line Formation	Ellipse Formation
R*	100	30
γ	0.1	0.5
ΔR_{out}	30	30
ΔR_{in}	20	20
ΔR_{avoid}	10	10
ε	0.001	.001

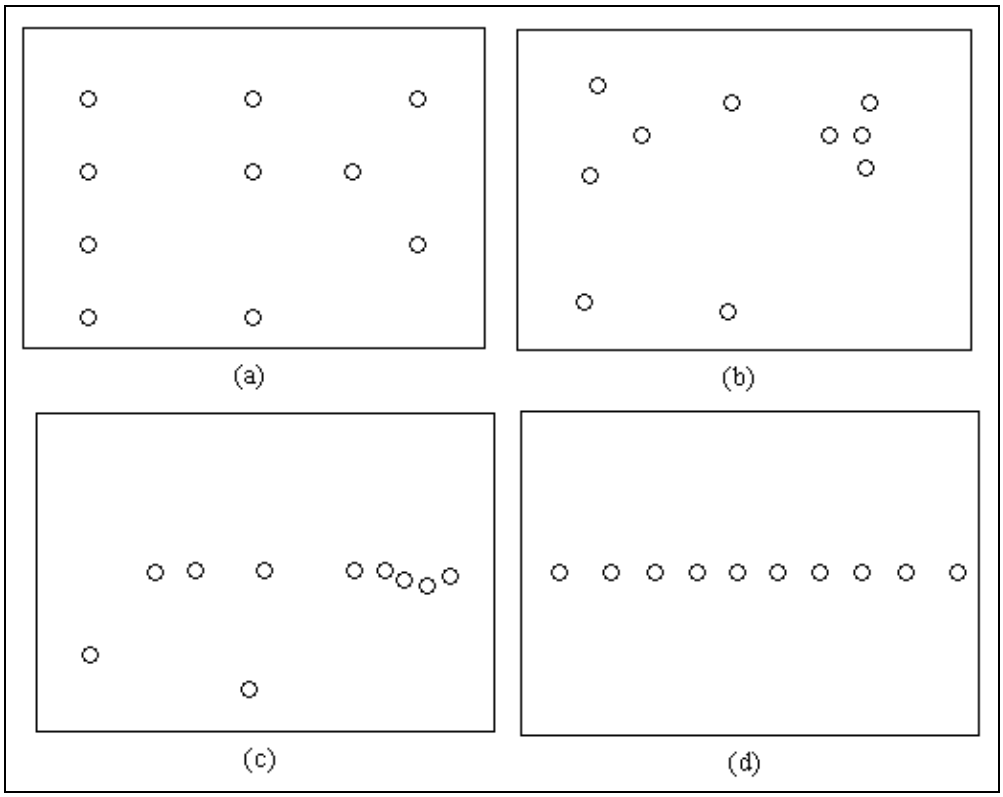

Figure 8.5. Line formation with ten robots at different time steps. (a) t=1, (b) t=25, (c) t=50, and (d) t=100.

8.1.4 Ten Heterogeneous Robots in an Ellipse Formation

Another simulation demonstrates control of ten robots following a straight trajectory and a sine trajectory. In this case, $\gamma<1$ in order to force a narrower ellipse configuration along the path. Each robot avoided other robots in the swarm and generated an avoidance field at the correct location to avoid a collision between swarm members. The center of the swarm is a function of time and is incremented as the members move. All ten robots used the same vector generation code and the control variables as in Table 8.2.

Figure 8.6 illustrates the swarm following a straight trajectory along the *x*-axis at different time steps. The robots start in random positions and align themselves onto the band of the ellipse. The robots take longer to get into the elliptical formation versus the line presented in section 8.1.3 but successfully maintain the formation while following a trajectory.

Figure 8.7a illustrates the robots making an elliptical formation while following a sine wave trajectory. The formation is still tight even when the robots are forced to reorient constantly.

Figure 8.7b shows the trajectories of each of the robots along the center. The robots exhibit the oscillating pattern because they do not all reach the center at the same time, and they rotate around the center area until the center is incremented.

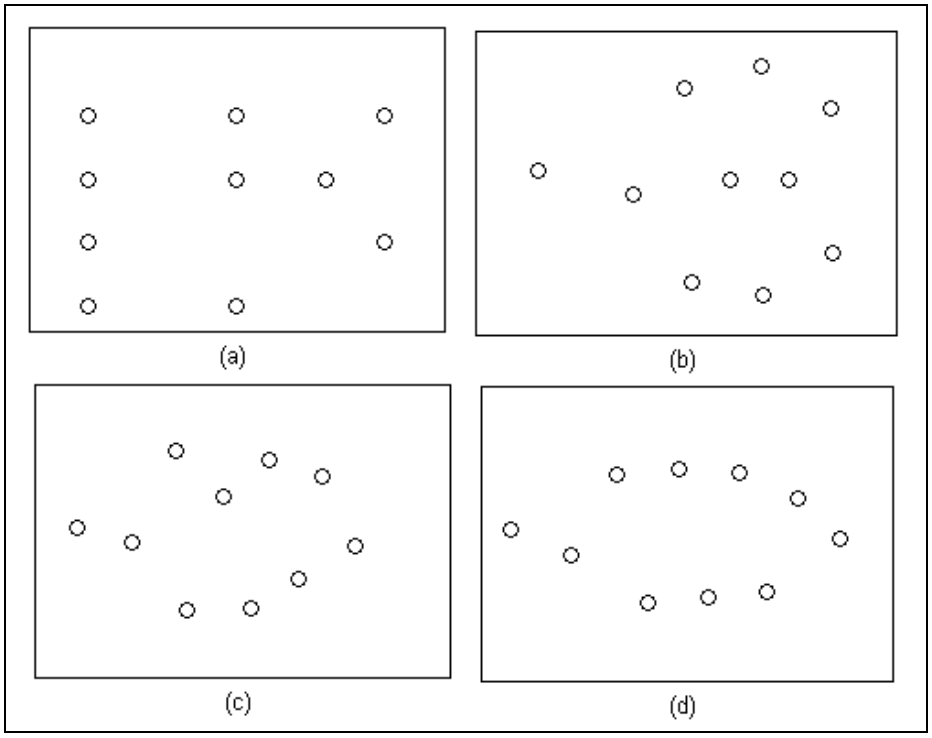

Figure 8.6. Ellipse formation with ten robots at different time steps. (a) t=1, (b) t=50, (c) t=100, and (d) t=200.

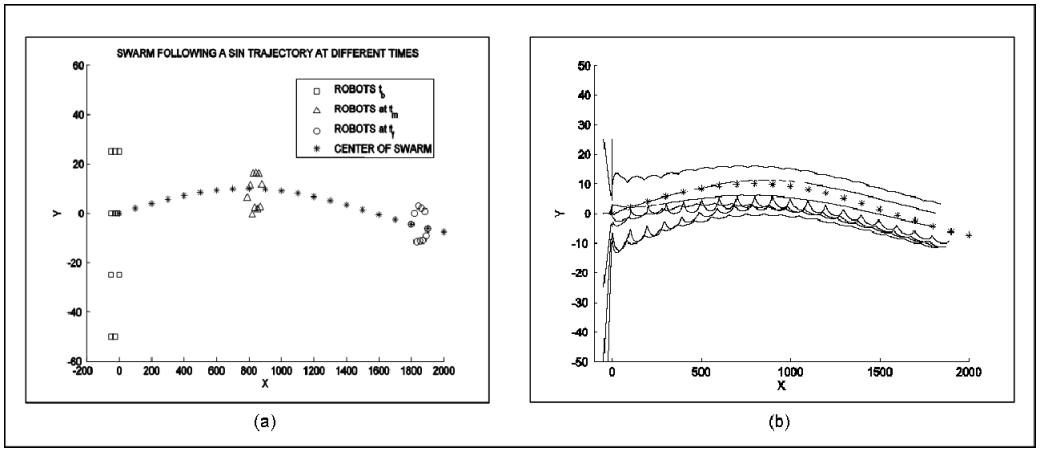

Figure 8.7. Ellipse formation with sine wave trajectory. (a) robots at different time steps, (b) trajectories.

8.2 Simulations with Particle Model

In order to demonstrate the precision of this method, simulations are run with four particle models. Parameters are selected based on desired formation and dispersion using methods described in Section 4.4. The center of the swarm is broadcast into the vector generating S-functions. The swarm center, (x_c,y_c), is left fixed in order to show the formation shape.

8.2.1 Simulations with Particle Model and Static Variable Selection

8.2.1.1 Four Particles in Arc and Circle Formations

Simulations are run with four particles to make an arc and a circle formation. Table 8.3 shows the parameter values used for each formation. Notice that the only factor to change the formation from an arc to a circle is a modification of the ΔR_{avoid} parameter.

Table 8.3. Control variables with four particles.

Control Variables	Arc Formation	Circle Formation
R^*	100	100
γ	1	1
ΔR_{in}	30	30
ΔR_{out}	20	20
ΔR_{avoid}	50	75
ε	0.001	0.001

Figure 8.8 shows the paths from initial positions into formation. Figure 8.9 shows the beginning swarm formation to the final swarm formation in the arc. Figure 8.10 shows the final swarm formation in circle/square.

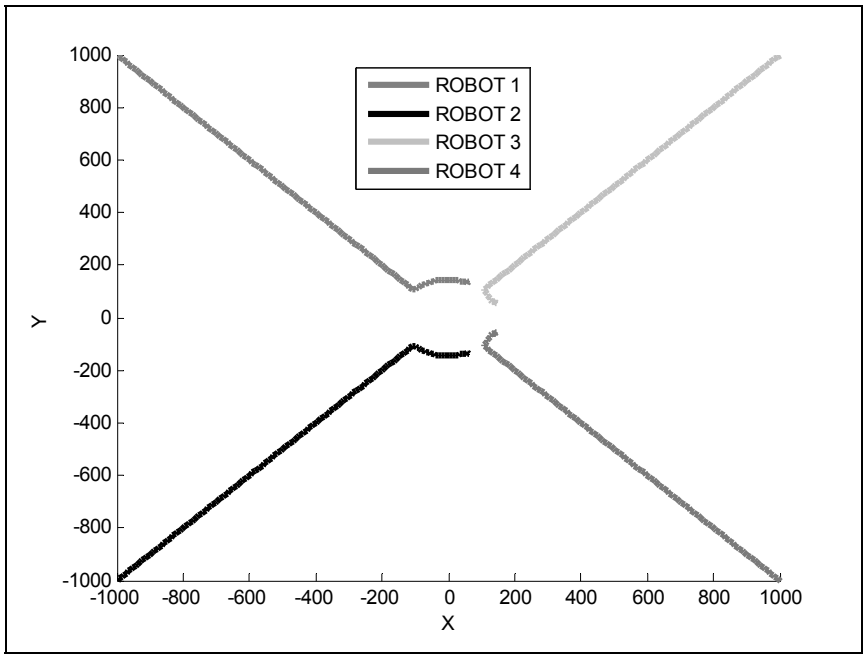

Figure 8.8. Particle paths from initial position into arc formation.

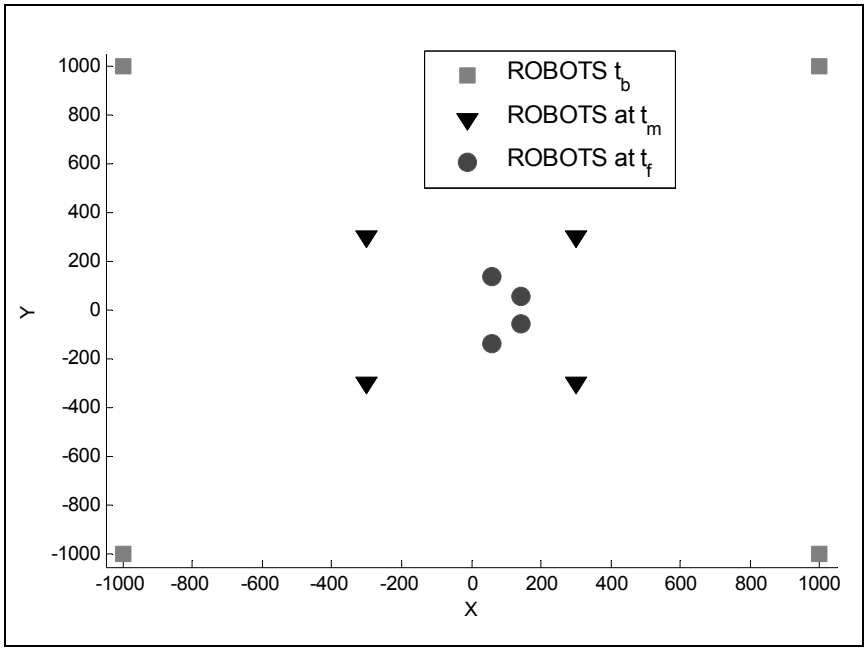

Figure 8.9. Particle arc formation at beginning (t_b), middle (t_m), and end (t_f).

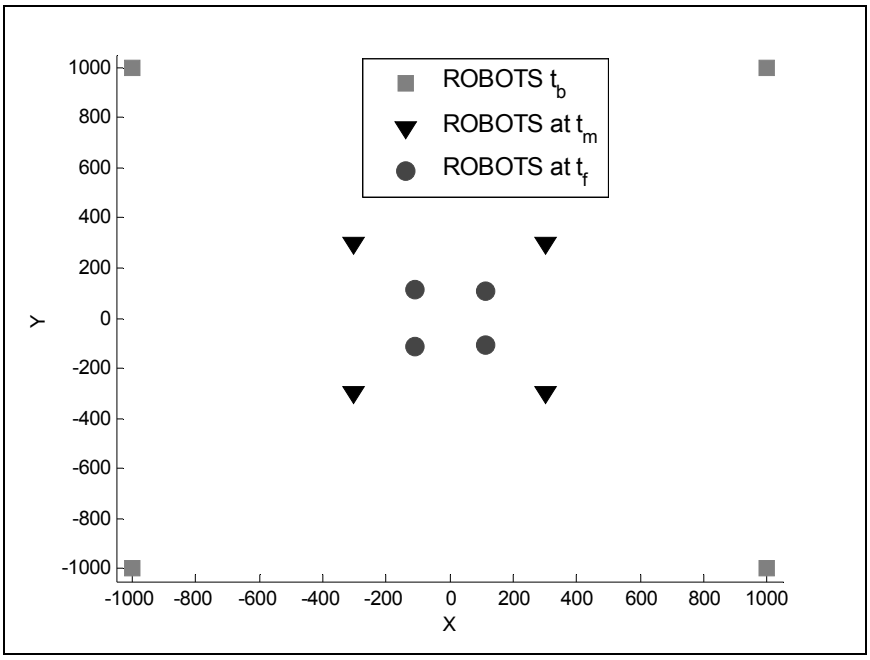

Figure 8.10. Particle circle formation at beginning (t_b), middle (t_m), and end (t_f).

8.2.1.2 Ten Particles in Circle and Ellipse Formations

Simulations with ten particles in circle and ellipse formation are also run. Table 8.4 shows the parameter values used for each formation. Notice that the only factor to change the formation from a circle to an ellipse is a modification of the γ parameter to make the *y-axis* skinnier.

Table 8.4. Control variables with ten particles.

Control Variables	Circle Formation	Ellipse Formation
R^*	200	200
γ	1	0.5
ΔR_{in}	60	30
ΔR_{out}	40	20
ΔR_{avoid}	75	75
ε	0.001	0.001

Figure 8.11 shows the particles in circle formation. In addition, the ellipse bands are plotted. The central band is R^*. The outer band is $R^* - \Delta R_{in}$ and the inner band is $R^* + \Delta R_{out}$. The particles stay within the acceptable bands and disperse themselves. Figure 8.12 is similar to Figure 8.11 except it is in ellipse formation. The robots are on the outer band in order to adhere to the ΔR_{avoid} parameter which remained the same between the circle and the ellipse although the area for formation decreased.

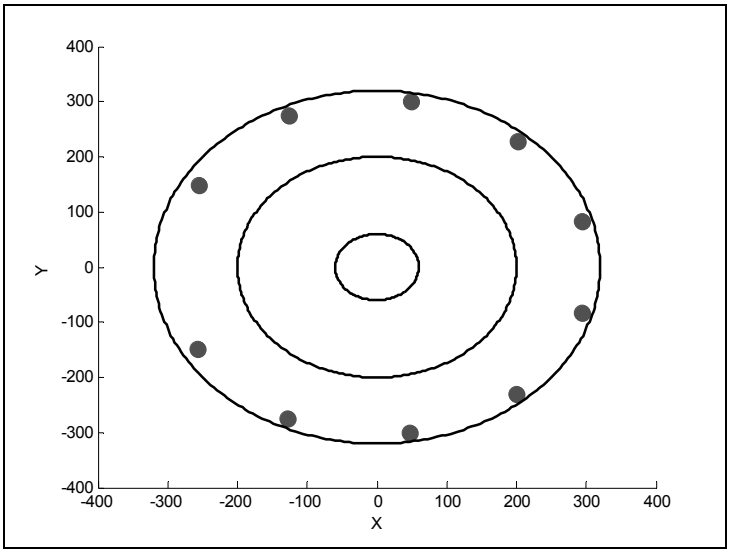

Figure 8.11. Particle circle formation.

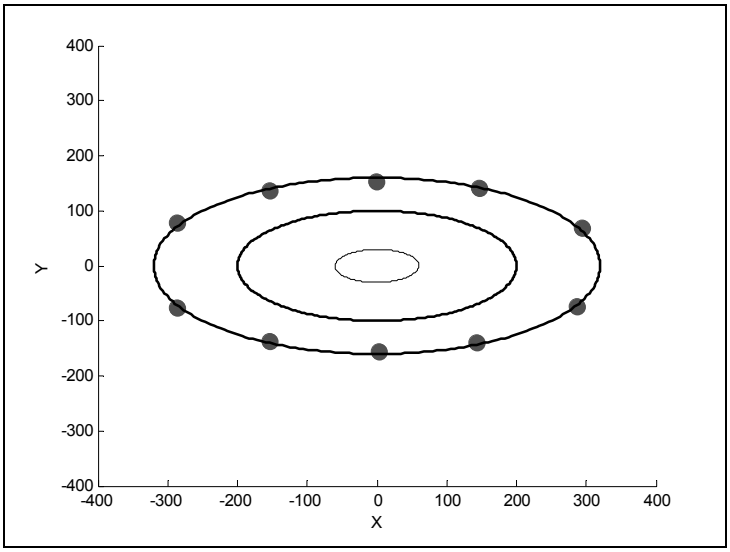

Figure 8.12. Particle ellipse formation.

8.2.2 Simulations with Particle Model and Fuzzy Variable Selection

Simulations are run with particles utilizing the fuzzy parameter selection method discussed in Section 4.4.2 to get an arc or wedge formation. These experiments demonstrate that tuning can help with dispersing the members about the formation function.

8.2.2.1 Four Particles in Arc / Wedge Formation

Simulations are run with swarm members to get an arc or wedge formation. Table 8.5 shows the control parameter used for the two experiments. Figure 8.13 and 8.14 show the plot from initial to final formation for the first and second experiment respectively. The only change between the two experiments is the R_{begin} and R_{end} ranges. In the first experiment the particles stay approximately 71 units apart. In the second experiment they stay approximately 88 units apart.

Table 8.5. Control variables for arc formation utilizing fuzzy parameter selection.

Control Variables	Value Experiment 1	Value Experiment 2
R^*	50	50
γ	1	1
R_{in}	70	70
R_{out}	30	30
R_{short}-R_{long}	10-20	10-20
R_{begin}-R_{end}	50-70	60-90
ε	0.001	0.001

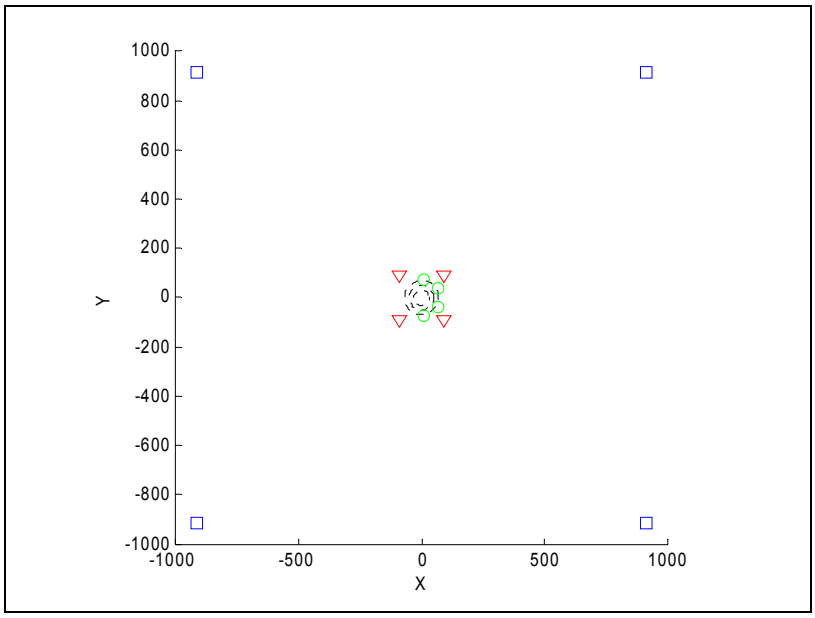

Figure 8.13. Experiment 1: Particle arc formation at beginning, middle, and end.

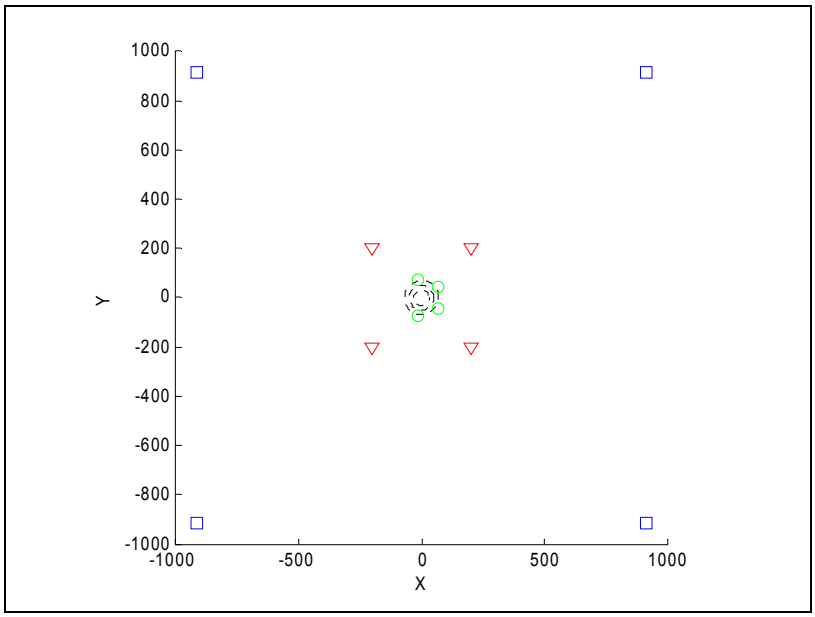

Figure 8.14. Experiment 2: Particle arc formation at beginning, middle, and end.

8.2.2.2 Four Particles in Circle / Square Formation

Simulations are run with swarm members to obtain a circle or square formation. Table 8.6 shows the control parameters used for the two experiments. Figure 8.15 and 8.16 show the plot from initial to final formation for the first and second experiment respectively. The only change between the two experiments is the R_{begin} and R_{end} ranges. In the first experiment the particles stay approximately 97 units apart. In the second experiment they stay approximately 117 units apart.

Table 8.6. Control variables for circle formation utilizing fuzzy parameter selection.

Control Variables	Value Experiment 1	Value Experiment 2
R^*	50	50
γ	1	1
R_{in}	70	70
R_{out}	30	30
R_{short}-R_{long}	10-20	10-20
R_{begin}-R_{end}	70-100	80-120
ε	0.001	0.001

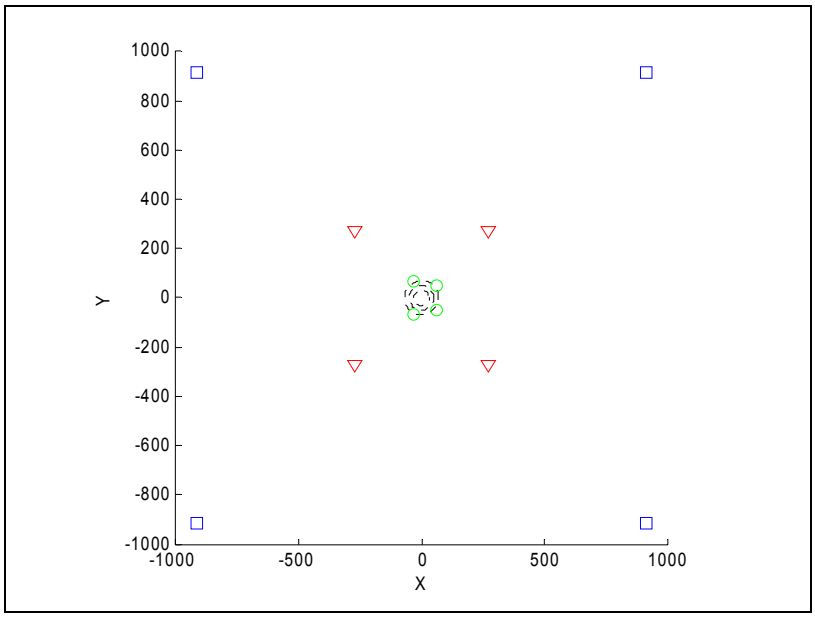

Figure 8.15. Experiment 1: Particle circle formation at beginning, middle, and end.

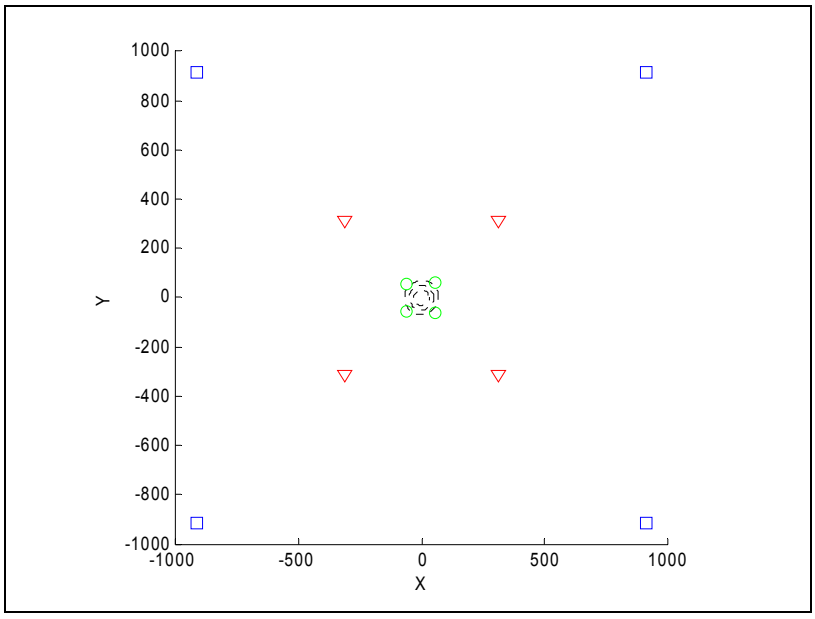

Figure 8.16. Experiment 2: Particle circle formation at beginning, middle, and end.

8.2.2.3 Four Particles in Ellipse / Rectangle Formation

Simulations are run with swarm members to get an ellipse or rectangular formation. Table 8.7 shows the control parameters used for the experiment. Figure 8.17 shows the particle paths to the final formation and Figure 8.18 show the plot from initial to final formation. The particles stay approximately 108 units apart.

Table 8.7. Control variables for ellipse formation utilizing fuzzy parameter selection.

Control Variables	Value
R^*	100
γ	0.5
R_{in}	60
R_{out}	140
R_{short}-R_{long}	10-20
R_{begin}-R_{end}	80-120
E	0.001

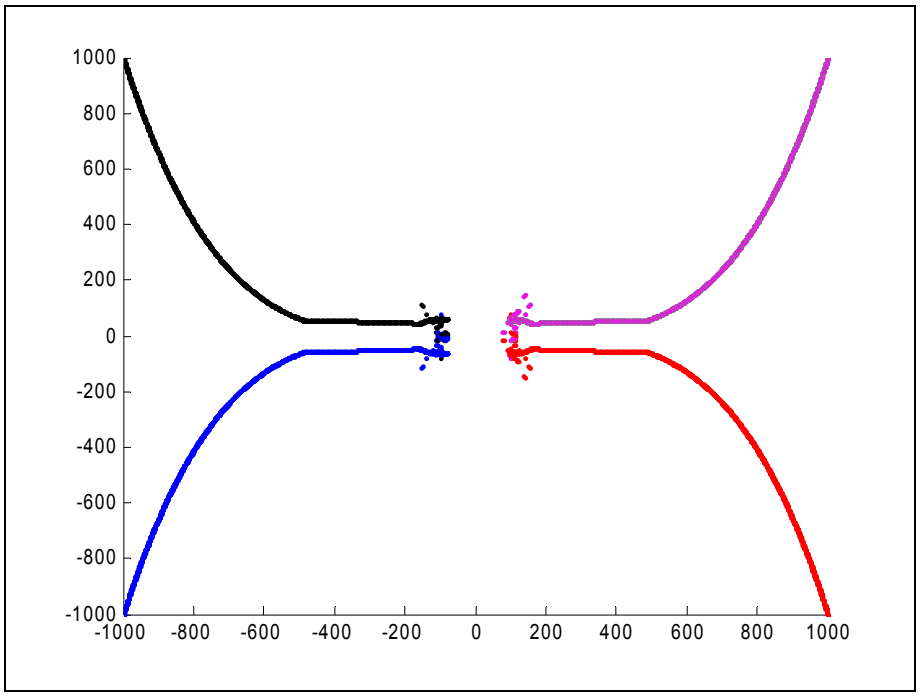

Figure 8.17. Particle paths to ellipse formation.

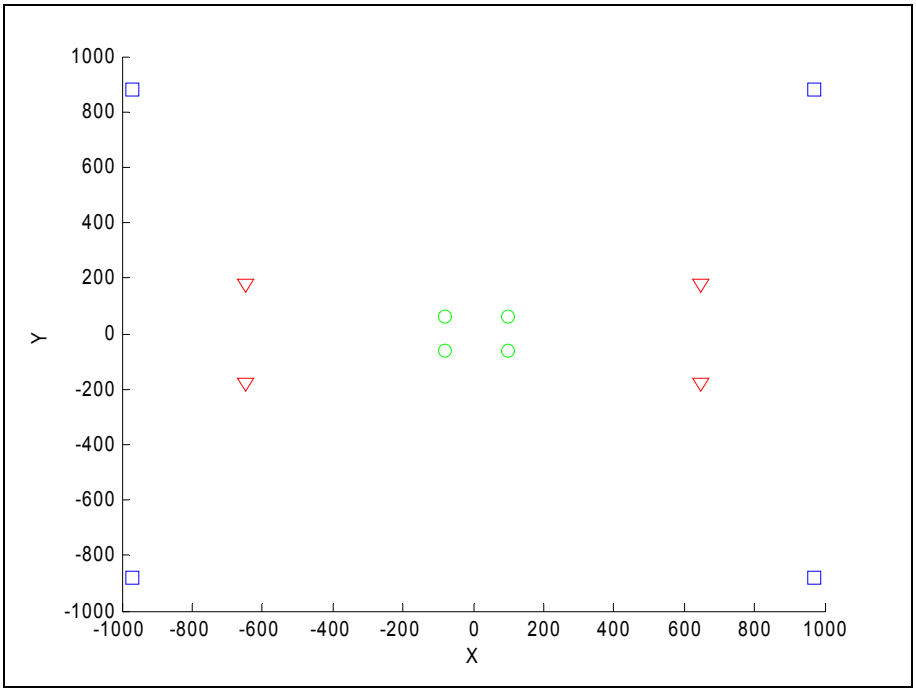

Figure 8.18. Particles at different time steps to ellipse formation.

Chapter 9

Field Experiments

Six different sets of field experiments have been performed. The test field is approximately *70* meters in width and *100* meters in length. Experiments were performed with three and four UGVs. In addition, an experiment demonstrating UAV-UGV swarm coordination was performed. The hardware described in Chapter 6 was utilized for all experiments. In some experiments a virtual coordinate was used for the swarm center, (x_c,y_c). This virtual coordinate is a function of time which all robots receive periodically. In the experiments not utilizing a virtual coordinate for the swarm center, another vehicle (either another UGV or a UAV) is defined as the center of the swarm. This vehicle has a trajectory to follow and the other swarm members receive its position periodically. At each time step, the robots compute their vectors based on the current position, the current center, and other swarm members locations. Based on the output of the vector fields, a desired speed and a desired heading are computed. For all experiments, time is in seconds, coordinates are in UTMs, and distance measurements are in meters.

9.1 Experiment 1: Four Robots in an Ellipse Formation with a Virtual Center

In experiment one, four UGV vehicles traveled in an ellipse formation surrounding a virtual center. The four UGVs travel surrounding each center point and staying at a minimum specified distance away from one another. Table 9.1 shows the control parameters used for this experiment. The units for R^*, ΔR_{in}, ΔR_{out}, and ΔR_{avoid} are all in meters. Figure 9.1 shows each swarm member's distance from the center over time. The jaggedness in these lines is due to GPS update rate. From this plot, the robots travel 3 to 4 meters outside the acceptable range from the center. This travel outside of range from Equation (4.11) could be because the formation parameters were slightly too small for the swarm size. Another possibility is that virtual center

was moving to quickly for the swarm members to keep up. This error is minor and could be improved with tuning.

Table 9.1. Control variables for experiment 1.

Control Variables	Experiment 1 Parameters
R^*	7
γ	1
ΔR_{in}	3
ΔR_{out}	4
ΔR_{avoid}	5
ε	0.001

Figure 9.2 shows the swarm formation at the beginning (t_b), middle (t_m), and end (t_f) of mission. The swarm members were started at random places at the beginning of the mission and moved into formation over time. The robots continued to hold formation with only slight deviation from t_m to t_f. It is important to note that the center is guiding the robots and 'pulls' and 'pushes' them along in formation. Unfortunately, on the actual robots the center tends to pull the robots along very well but the pushing ahead (to the front of the formation) does not work as it does in simulation and the robots fall behind the center (x_c,y_c). In Figure 9.2, the robots are in formation but they are behind the center throughout the mission. Therefore, the most predictable control of the swarm is obtained when the movement of the center 'leads' the movement of the swarm members. Chapter 10 proposes an alternate approach which alleviates this inconsistency and sharpens the formations.

Figure 9.3 show the robot paths and the centers with respect to time. The robots avoid each other and follow the formation function with center x_c and y_c given at different time steps. Figure 9.4 demonstrates that the swarm members stay an acceptable distance away from one another throughout the mission.

A video demonstrating the first experiment can be found at http://www.csee.usf.edu/USL/Videos/4bot-clip1.wmv.

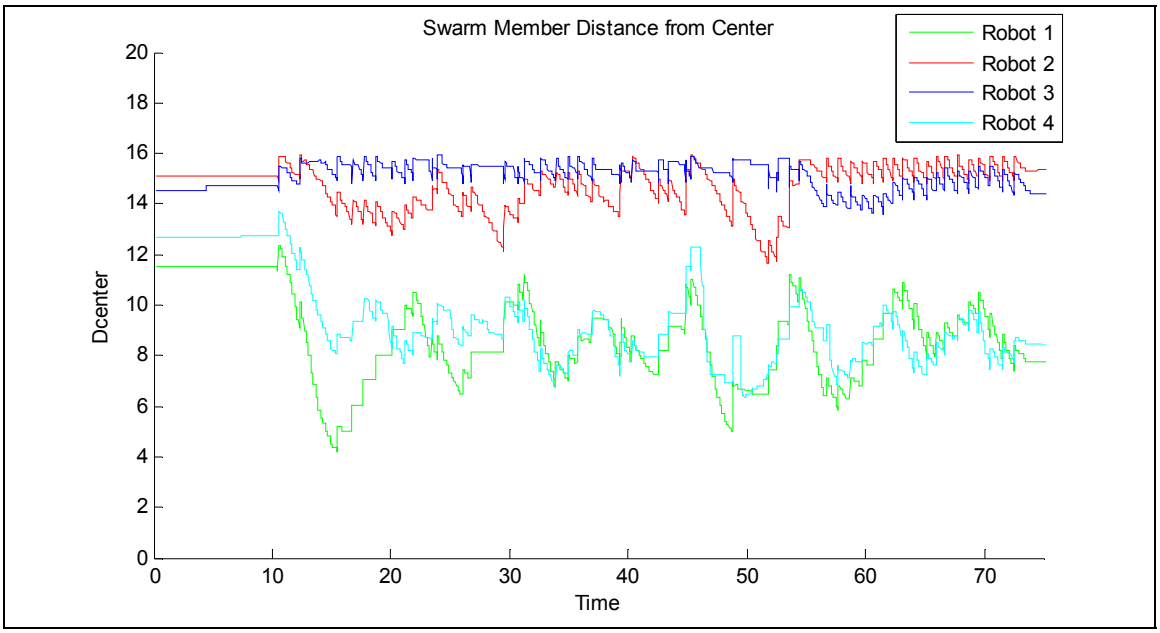

Figure 9.1. Experiment 1: Robot distance from center of swarm (x_c, y_c).

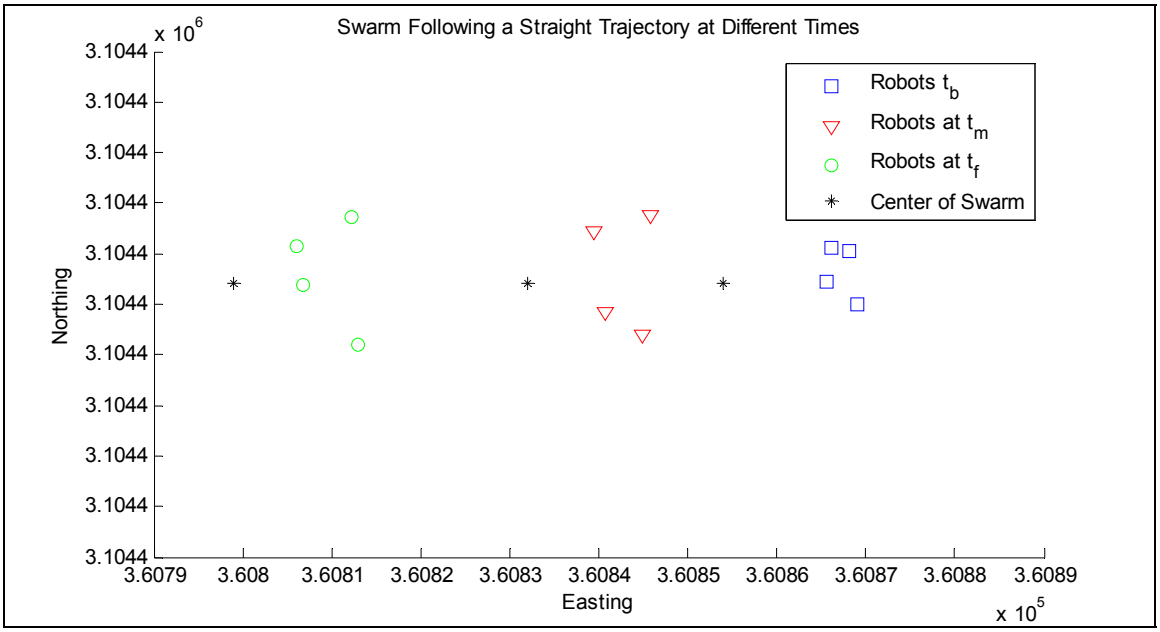

Figure 9.2. Experiment 1: Robot formation at beginning (t_b), middle (t_m), and end (t_f) of mission.

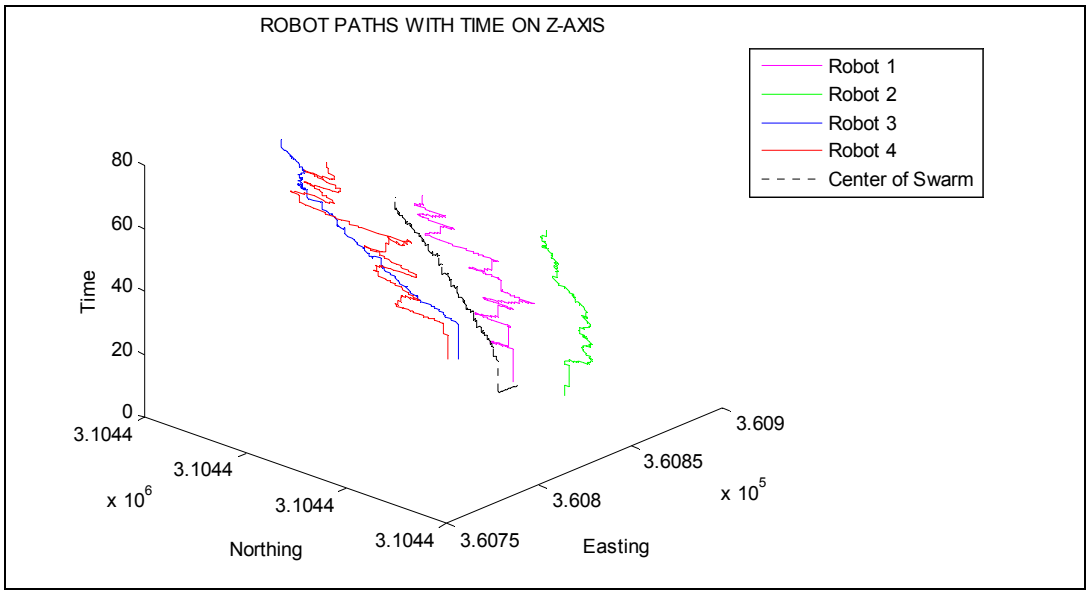

Figure 9.3. Experiment 1: Robot paths with respect to center (x_c, y_c) with time on z-axis.

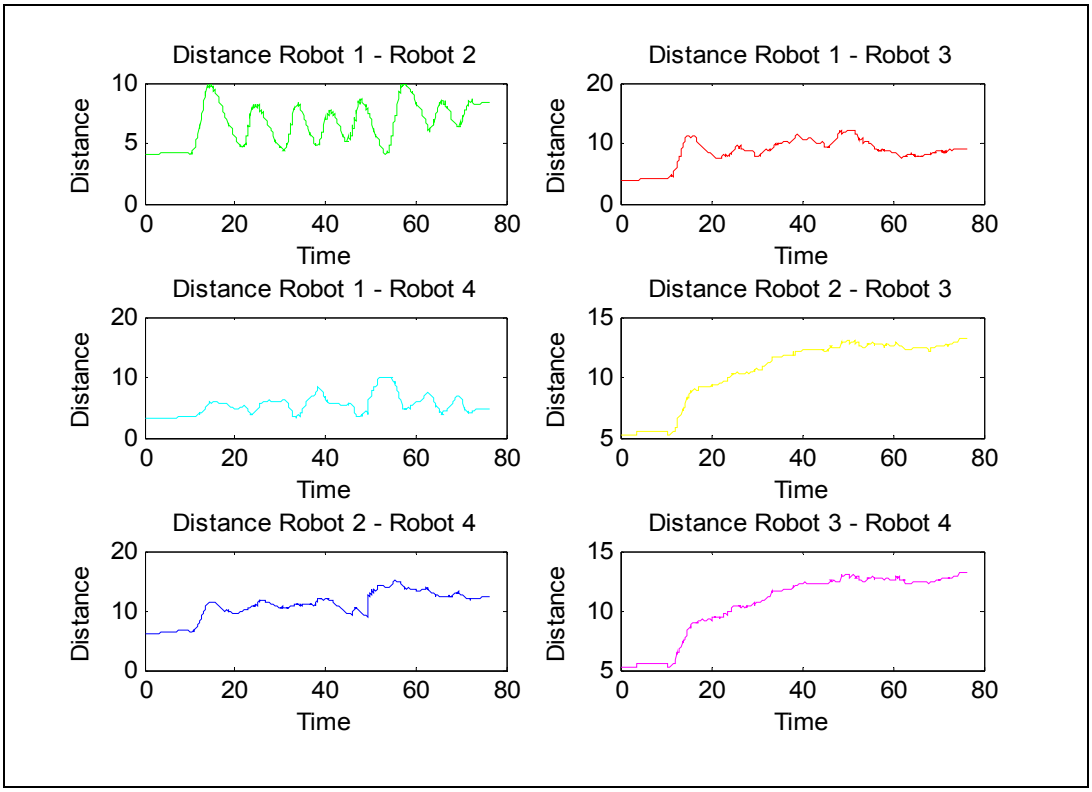

Figure 9.4. Experiment 1: Distance between swarm members.

9.2 Experiment 2: Three Robots in an Ellipse Formation with a Virtual Center

In experiment two, three UGV vehicles traveled in an ellipse formation surrounding a virtual center. The three UGVs travel surrounding the center point at each time step and staying at a minimum specified distance away from one another. Table 9.2 shows the control parameters used for this experiment. Figure 9.5 shows each swarm member's distance from the center over time. From this plot, as in experiment 1, the robots travel 3 to 4 meters outside the acceptable range from the center.

Table 9.2. Control variables for experiment 2.

Control Variables	Experiment 2 Parameters
R^*	10
γ	1
ΔR_{in}	4
ΔR_{out}	6
ΔR_{avoid}	5
ε	0.001

Figure 9.6 shows the swarm formation at the beginning (t_b), middle (t_m), and end (t_f) of mission. The swarm members were started at random places at the beginning of the mission and moved into formation over time. The robots continued to hold a tight wedge formation with only slight deviation from t_m to t_f. Again, the center is guiding the robots and 'pulls' them along in formation.

Figure 9.7 show the robot paths and the centers with respect to time. The robots avoid each other and follow the same formation function with center x_c and y_c given at different time steps. Figure 9.8 demonstrates that the swarm members stay an acceptable distance away from one another throughout the mission.

A video demonstrating experiment two can be found at http://www.csee.usf.edu/USL/Videos/3bot-clip1.wmv.

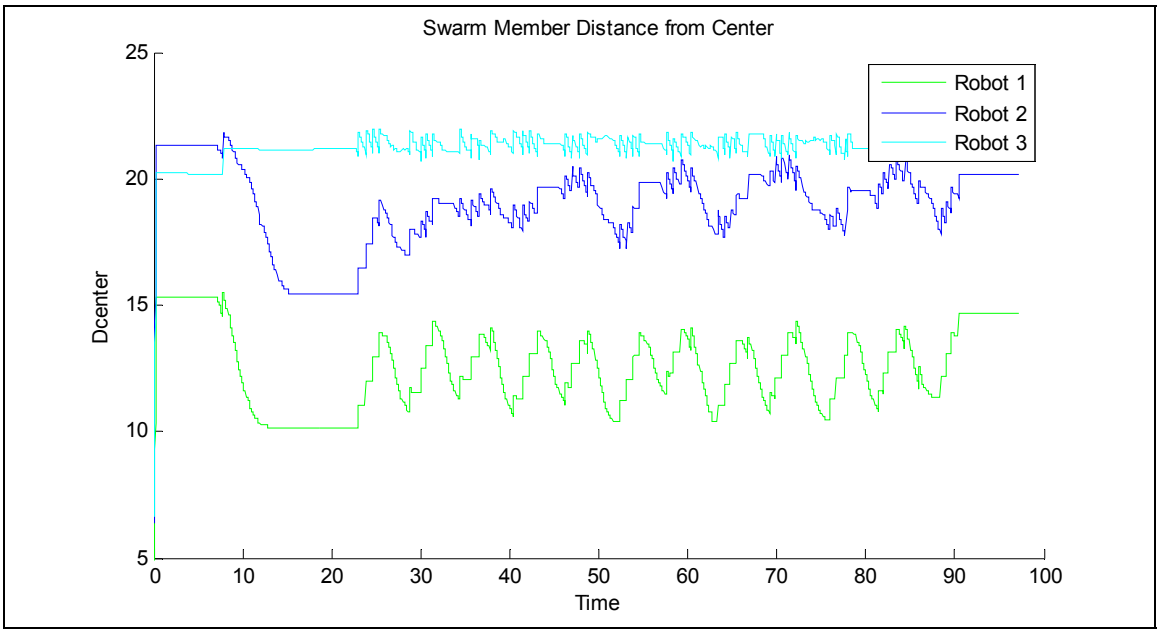

Figure 9.5. Experiment 2: Robot distance from center of swarm (x_c, y_c).

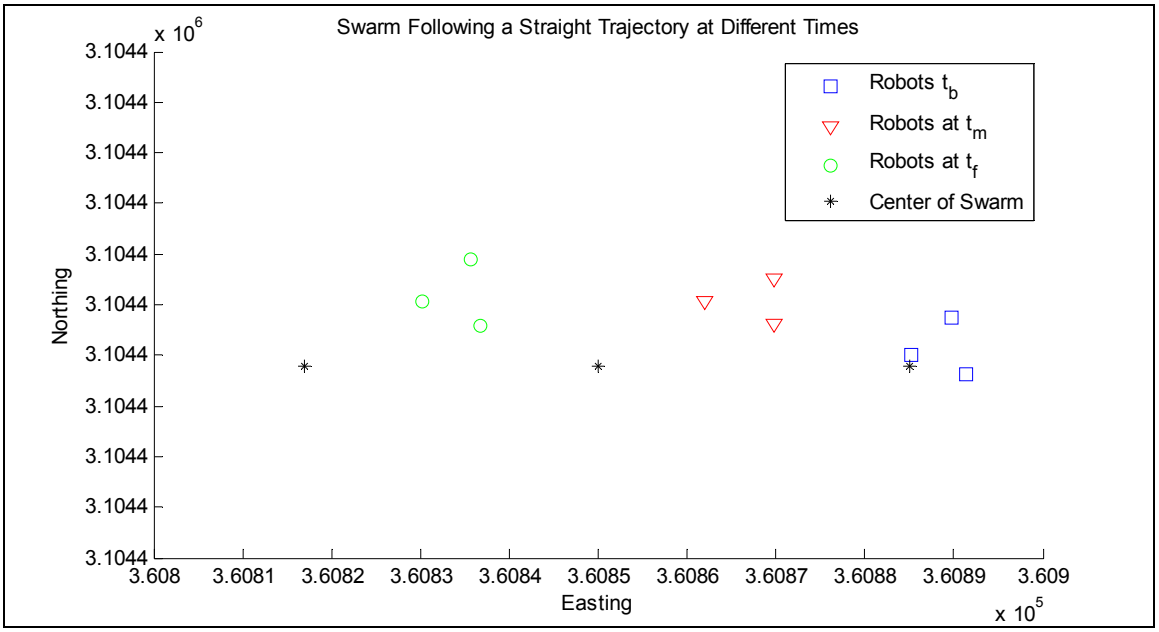

Figure 9.6. Experiment 2: Robot formation at beginning (t_b), middle (t_m), and end (t_f) of mission.

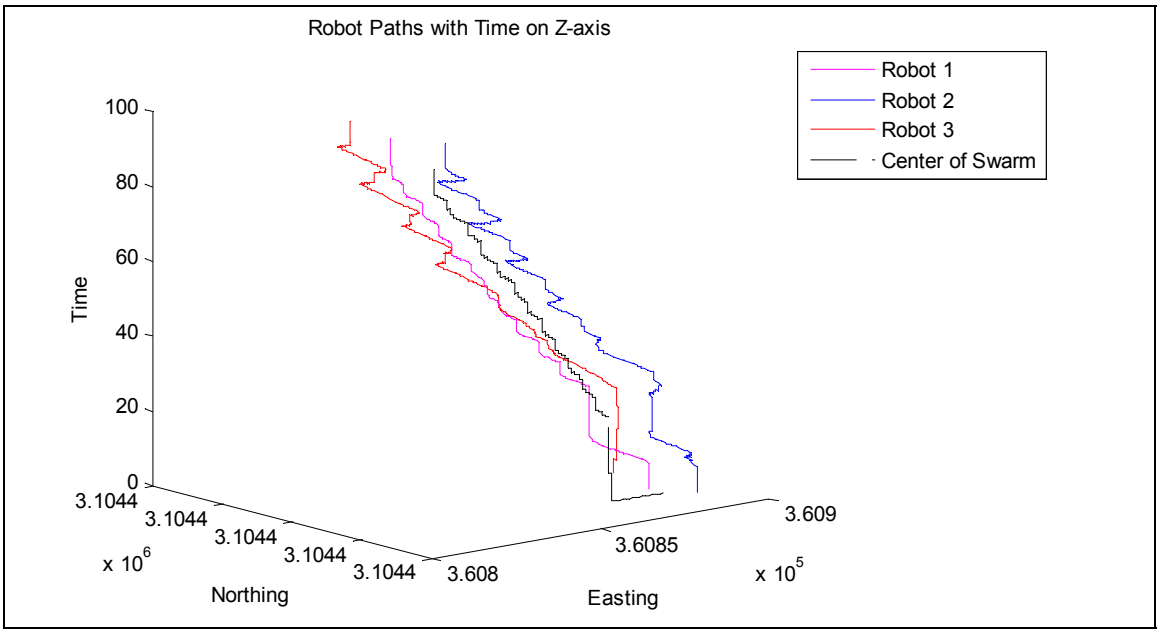

Figure 9.7. Experiment 2: Robot paths with respect to center (x_c, y_c) with time on z-axis.

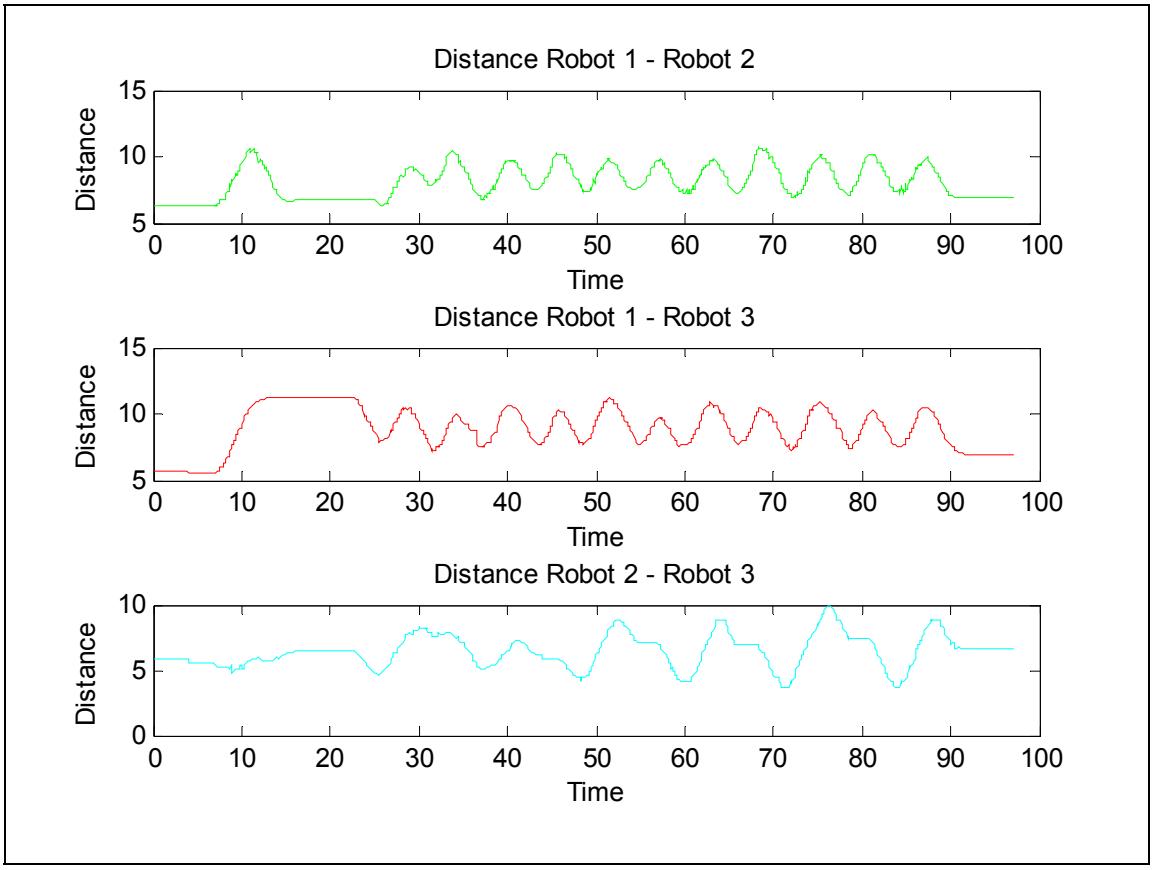

Figure 9.8. Experiment 2: Distance between swarm members.

9.3 Experiment 3: Three Robots in an Ellipse Formation with a Robot Center

In experiment three, three UGV vehicles travel in an ellipse formation. One of these UGVs, the alpha robot, acts as the formation center (x_c,y_c). Two UGVs, the beta robots, travel surrounding the alpha UGV and stay a minimum specified distance away from one another. Table 9.3 shows the control parameters used for this experiment. Figure 9.9 shows each beta swarm member's distance from the center over time. From this plot, as in experiment 1, the robots travel between *7* and *11* meters from the center. The error is approximately *3* to *4* meters as in experiment 1 and experiment 2.

Table 9.3. Control variables for experiment 3.

Control Variables	Experiment 3 Parameters
R^*	5
γ	1
ΔR_{in}	2
ΔR_{out}	3
ΔR_{avoid}	5
ε	0.001

Figure 9.10 shows the swarm formation at the beginning (t_b), middle (t_m), and end (t_f) of mission. The swarm members were started at random places at the beginning of the mission and moved into formation following the leader robot. The robots continued to hold a tight wedge formation with only slight deviation from t_m to t_f. The alpha robot is guiding the robots along in formation.

Figure 9.11 show the robot paths and the centers with respect to time. The black line is the path of the alpha robot. The robots avoid each other and follow the same formation function with center x_c and y_c given at different time steps. Figure 9.12 demonstrates that the two beta swarm members stay an acceptable distance away from one another throughout the mission.

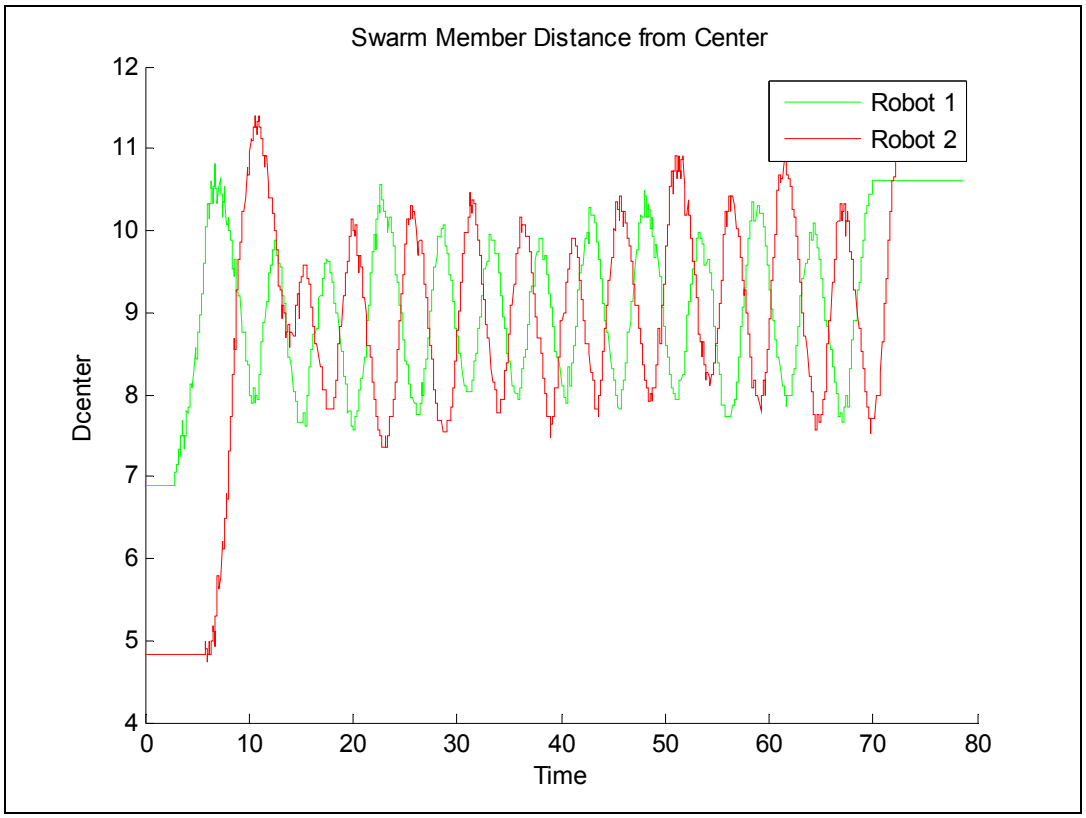

Figure 9.9. Experiment 3: Robot distance from center of swarm (x_c, y_c).

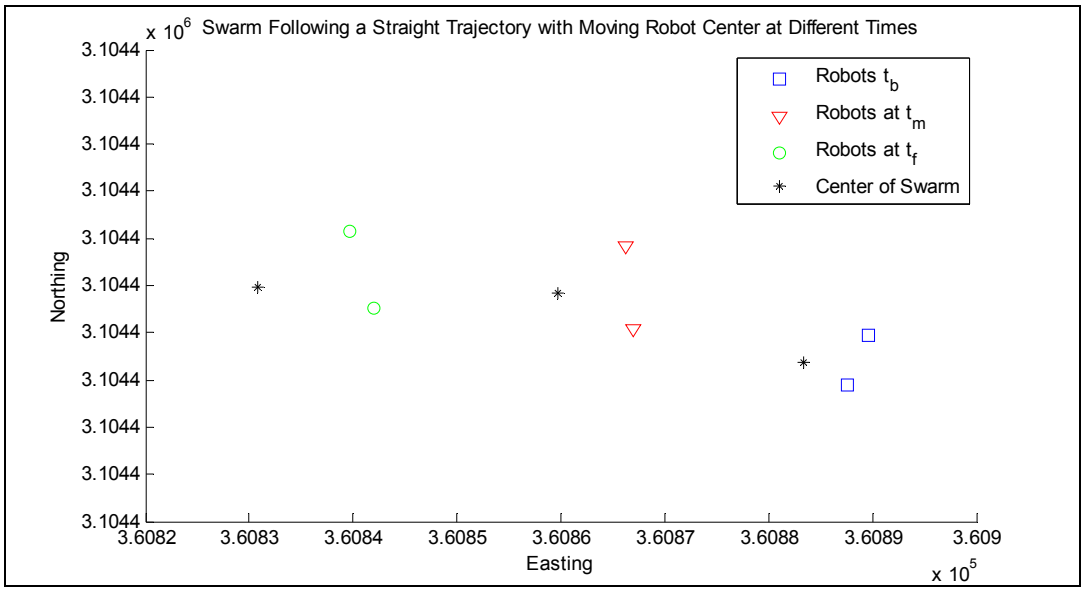

Figure 9.10. Experiment 3: Robot formation at beginning (t_b), middle (t_m), and end (t_f) of mission.

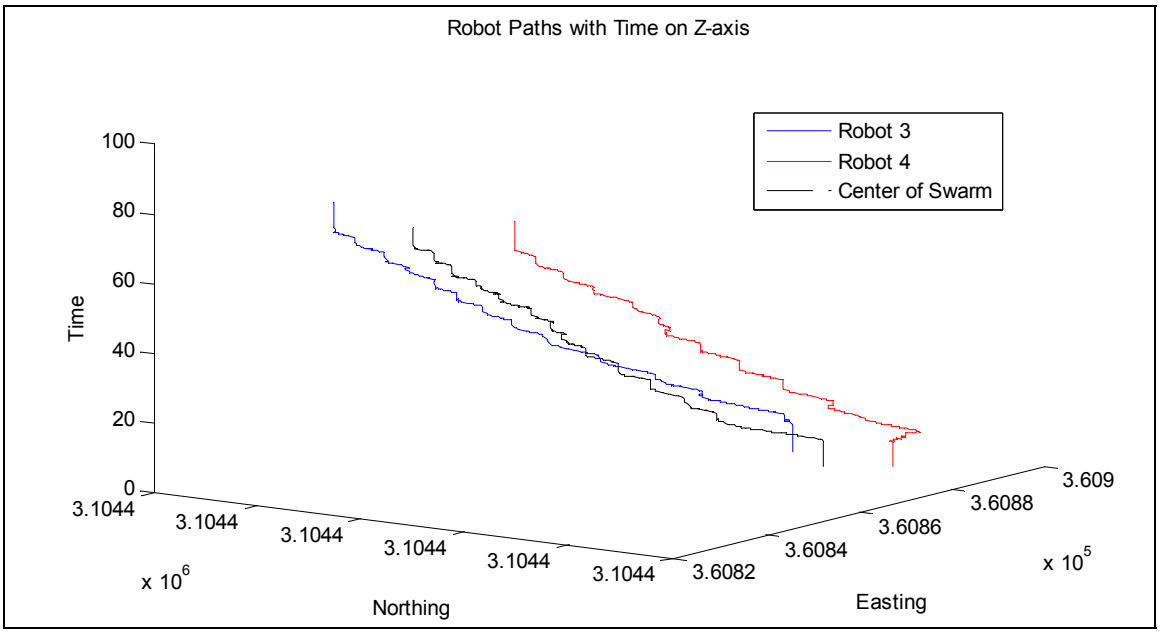

Figure 9.11. Experiment 3: Robot paths with respect to center (x_c, y_c) with time on z-axis.

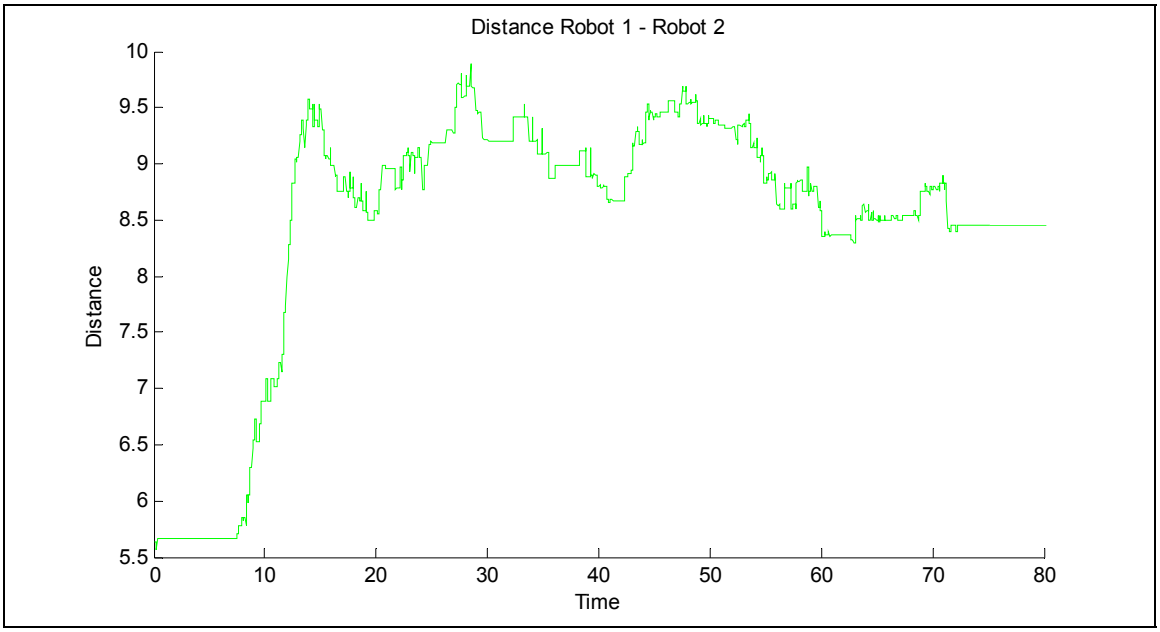

Figure 9.12. Experiment 3: Distance between swarm members.

9.4 Experiment 4: Three Robots in a Line Formation

In experiment four, three UGV vehicles travel in a line formation. The leader-follower method discussed in Section 4.4.1.3.2 is used. One of these UGVs, the alpha robot, follows a formation utilizing a virtual formation center (x_c,y_c). The alpha robot is at the top of the line hierarchy. The next UGV, beta robot 1, follows the alpha UGV staying a minimum specified distance away. The next UGV, beta robot 2, follows beta robot 1. Table 9.4 shows the control parameters used for the alpha robot in this experiment. Figure 9.13 shows each swarm member's distance from the other swarm members over time. The robots are evenly distributed approximately 10 meters apart in a line formation.

Table 9.4. Control variables for experiment 4.

Control Variables	Experiment 4 Parameters
R^*	3
γ	1
ΔR_{in}	1
ΔR_{out}	1
ΔR_{avoid}	5
ε	0.001

Figure 9.14 shows the swarm formation at the beginning (t_b), middle (t_m), and end (t_f) of mission. The swarm members were started at random places at the beginning of the mission and moved into formation following the alpha robot. The robots continued to hold a tight line formation from t_m to t_f.

Figure 9.15 show the robot paths and the centers with respect to time. On the legend, Robot 1 is the path of the alpha robot. Robot 2 is the path of beta robot 1, and Robot 3 is the path of beta robot 2. The dark black line is the center of the swarm which the alpha robot follows.

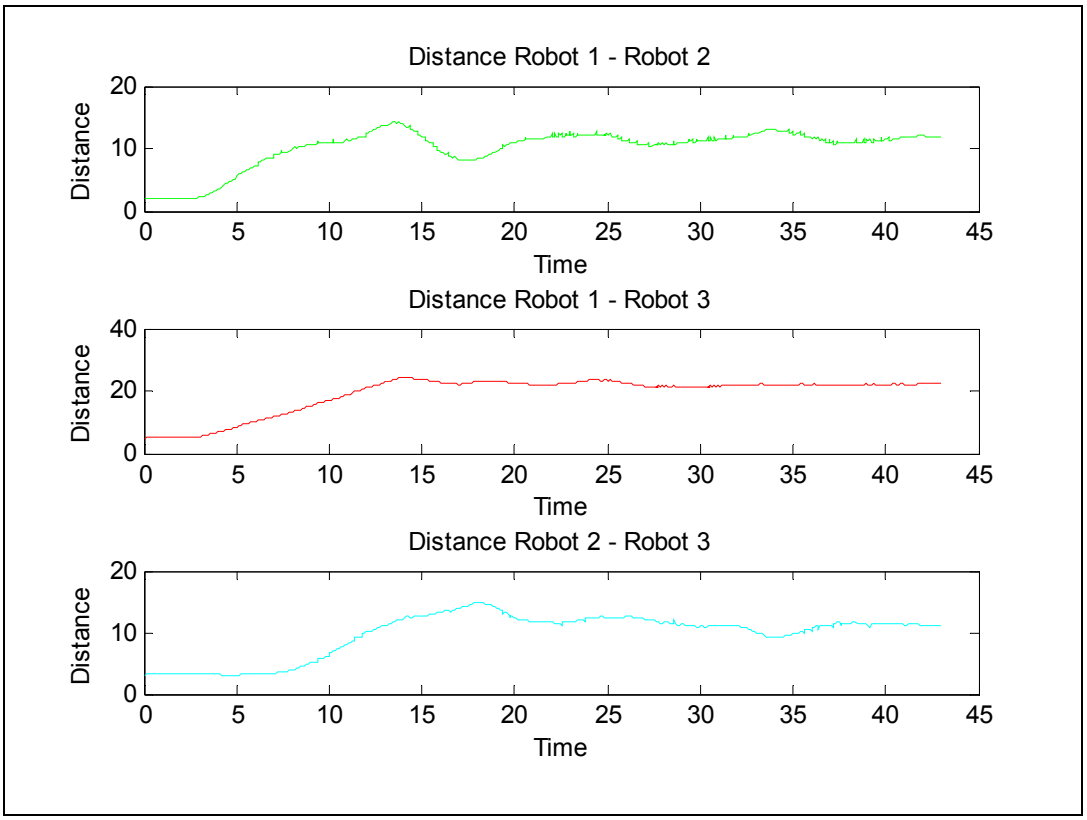

Figure 9.13. Experiment 4: Distance between swarm members.

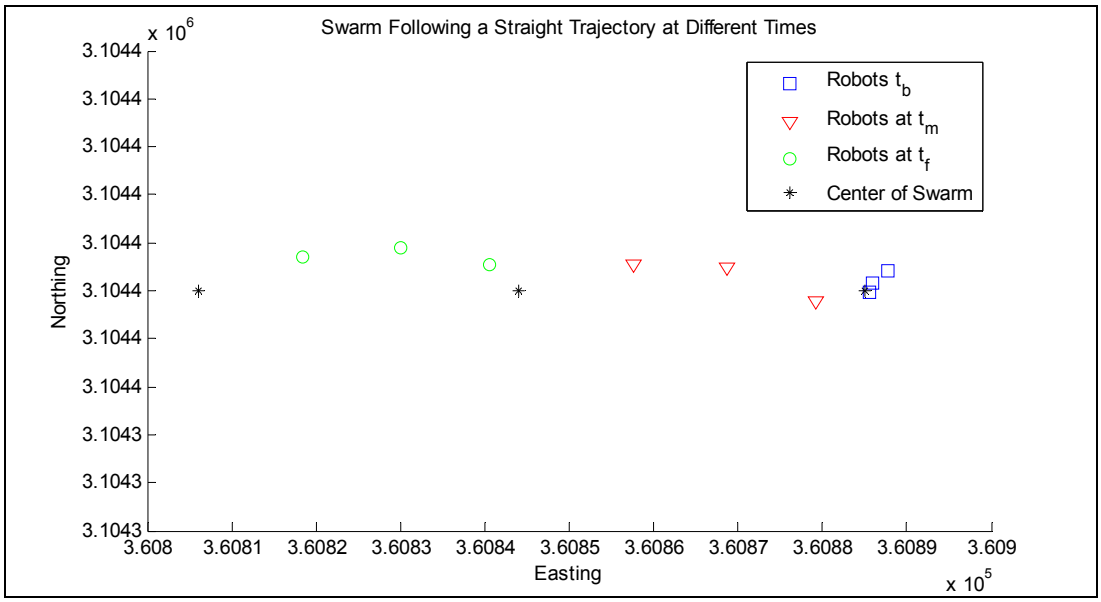

Figure 9.14. Experiment 4: Robot formation at beginning (t_b), middle (t_m), and end (t_f) of mission.

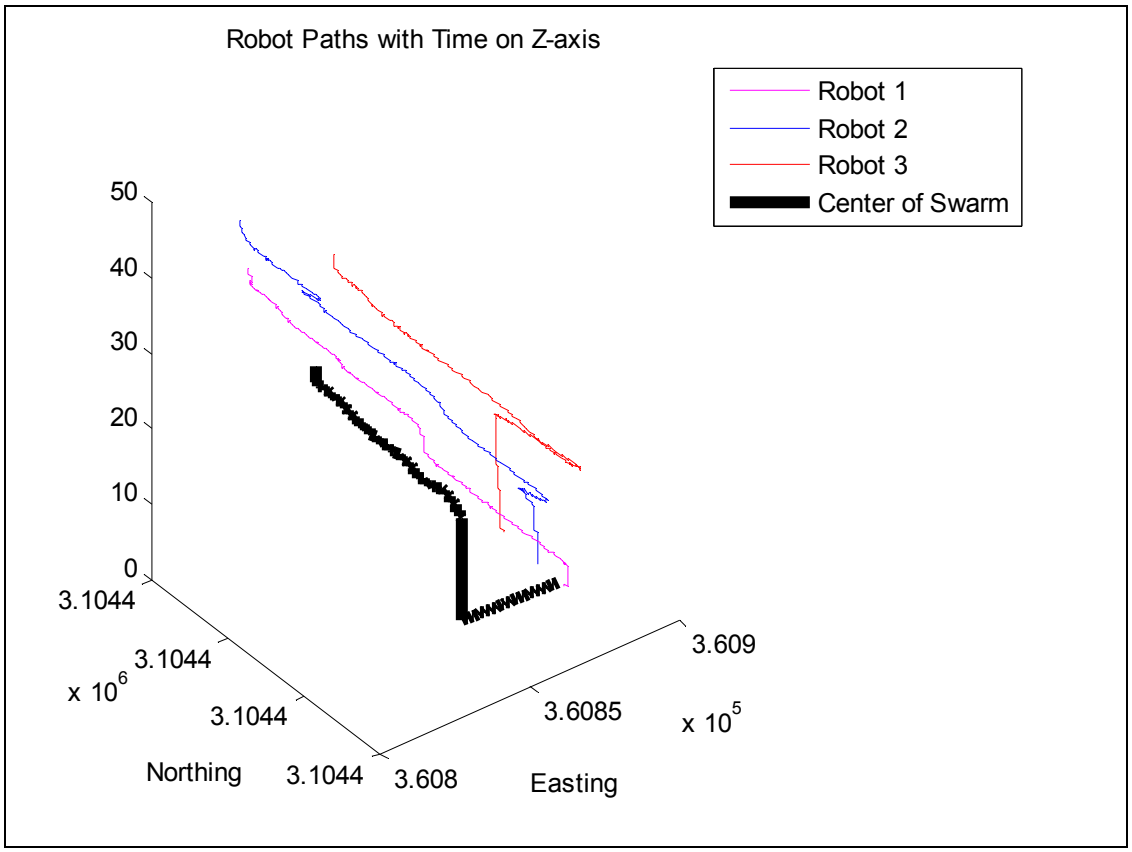

Figure 9.15. Experiment 4: Robot paths with respect to center (x_c, y_c) with time on z-axis.

9.5 Experiment 5: Three Robots in an Ellipse Formation with a Failure

In this experiment, as in experiment three, UGV vehicles travel in an ellipse formation. One of these UGVs, the alpha robot, acts as the formation center (x_c,y_c). Two UGVs, the beta robots, travel surrounding the alpha UGV and stay a minimum specified distance away from one another.

A UGV failure is integrated into this experiment. One of the beta robots will fail during the mission. This UGV does not actually fail, but the communication server dies, and the robot's navigation function is suspended. When the two other UGVs recognize that the communication link is broken, they dynamically change their parameters from those in column one of Table 9.5 to those in column two. This dynamic formation change results in a circle with a smaller radius. The UGVs seamlessly make the transition and continue after the failure.

Table 9.5. Control variables for experiment 5.

Control Variables	Experiment 5 Parameters Before Failure	Experiment 5 Parameters After Failure
R^*	7	4
γ	1	1
ΔR_{in}	3	1
ΔR_{out}	4	2
ΔR_{avoid}	5	5
ε	0.001	0.001

Figure 9.16 shows the swarm formation at the beginning (t_b), middle (t_m), and end (t_f) of mission. The swarm members were started at random places at the beginning of the mission and moved into formation following the alpha robot. After t_m, one of the beta robots fails. When the other robots realize there has been a failure, they modify their formation and continue to t_f. Figure 9.17 show the robot paths over time. The failure can be seen on the graph.

Figure 9.18 shows each swarm member's distance from the other swarm members. Robot 2 is the failed robot which can be seen on the plots when the distance from the other robots begins to grow.

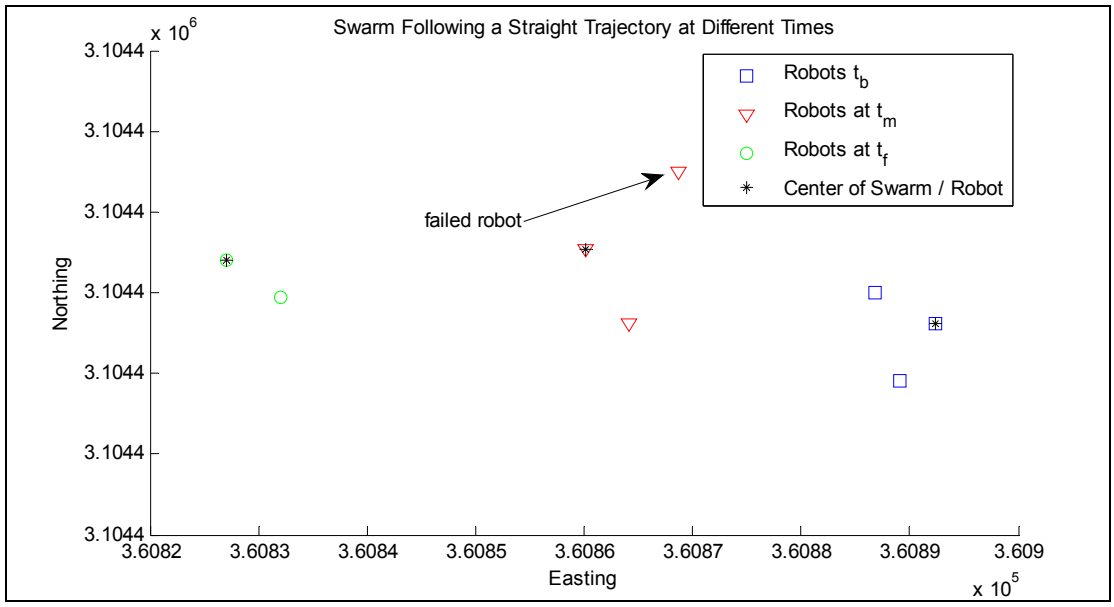

Figure 9.16. Experiment 5: Robot formation at beginning (t_b), middle (t_m), and end (t_f) of mission.

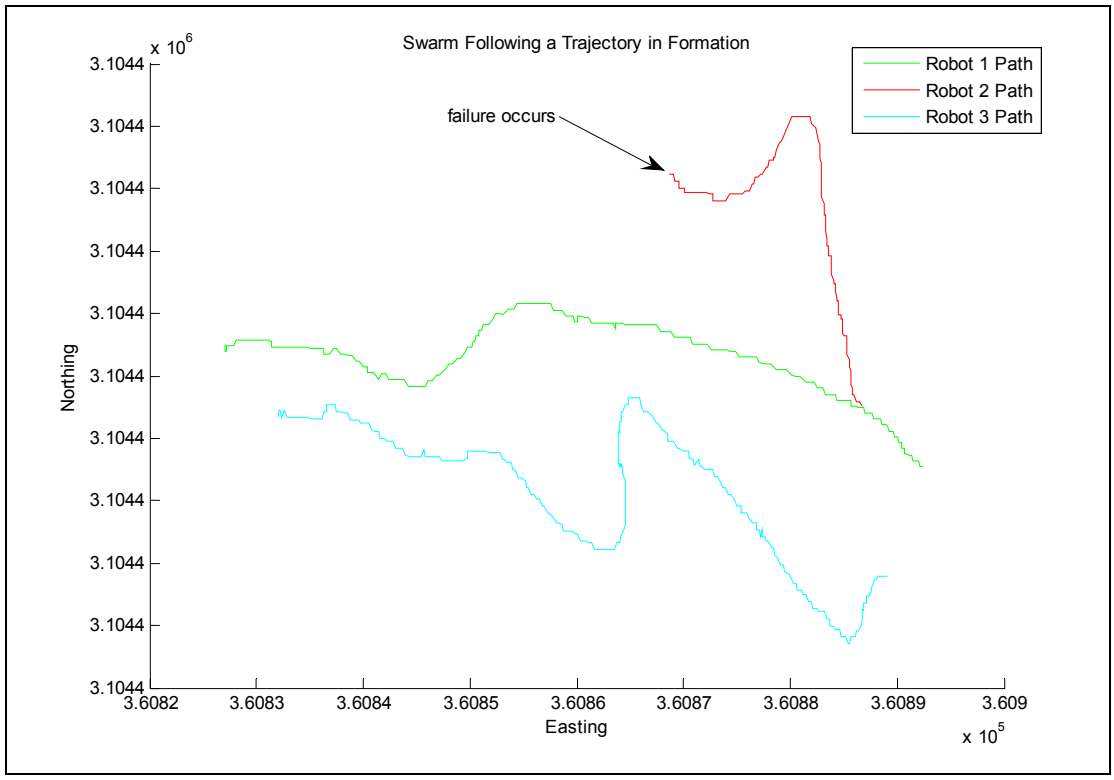

Figure 9.17. Experiment 5: Robot paths with respect to center (x_c, y_c).

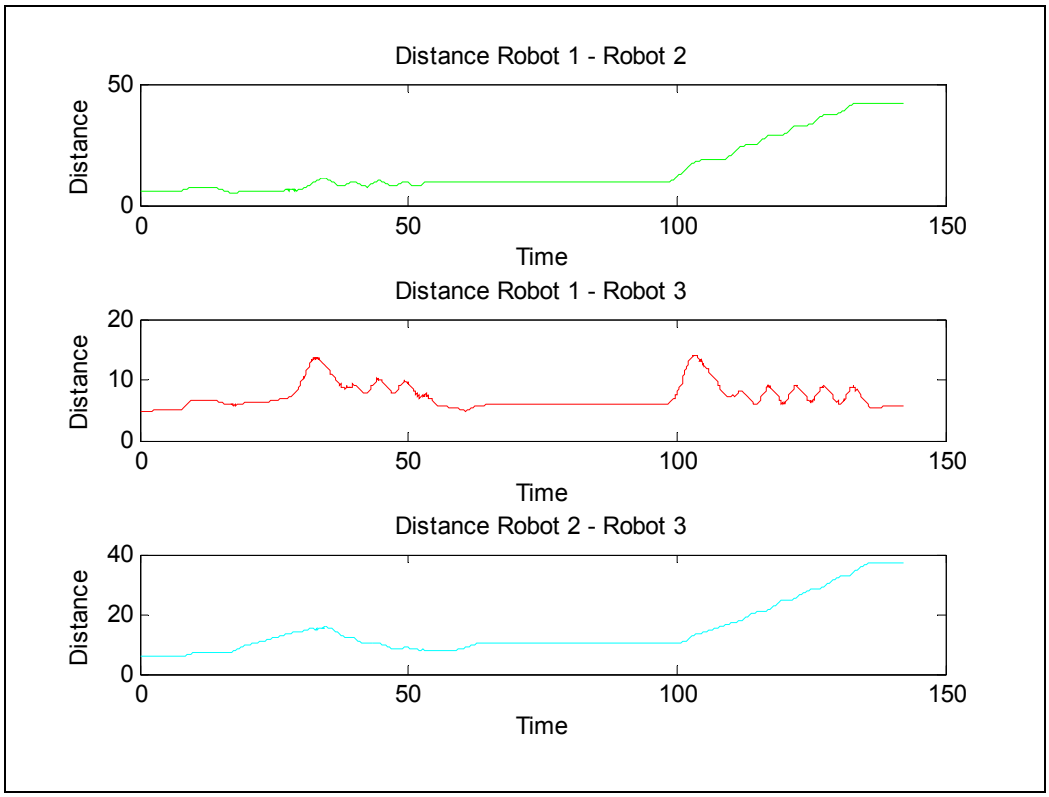

Figure 9.18. Experiment 5: Distance between swarm members.

9.6 Experiment 6: UAV-UGV Swarm Coordination

In experiment six, a helicopter is utilized as the alpha robot. Controllers from [115] are used on the Maxi Joker 2 helicopter. Figure 9.19 shows a diagram of the experiment with the helicopter and the UGVs.

Three UGV vehicles travel in an ellipse formation surrounding the helicopter. The helicopter or the alpha robot acts as the formation center (x_c,y_c). Three UGVs, the beta robots, travel surrounding the alpha UAV and stay a minimum specified distance away from one another. Table 9.6 shows the control parameters used for this experiment. Figure 9.20 shows each beta swarm member's distance from the center over time.

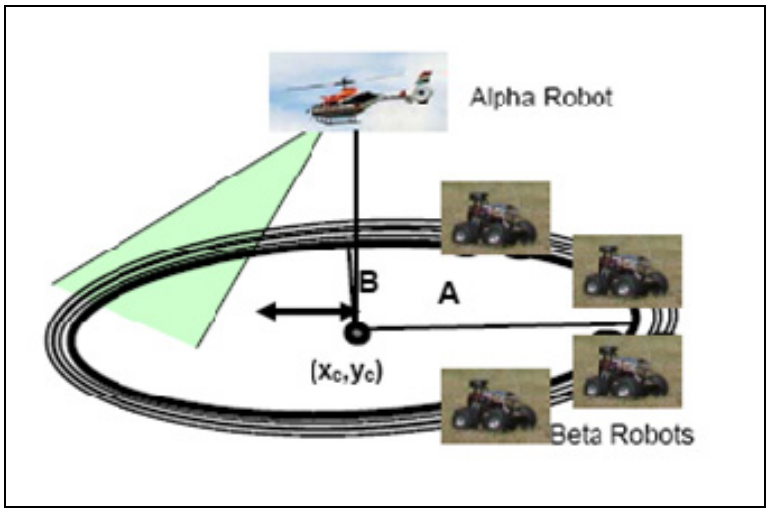

Figure 9.19. UAV-UGV swarm coordination.

Table 9.6. Control variables for experiment 6.

Control Variables	Experiment 6 Parameters
R^*	7
γ	1
ΔR_{in}	3
ΔR_{out}	4
ΔR_{avoid}	5
ε	0.001

Figure 9.21 shows the swarm formation at the beginning (t_b), middle (t_m), and end (t_f) of mission. The swarm members were started at random places at the beginning of the mission and moved into formation following the alpha robot. The robots continued to hold an inverted 'vee' like formation with only slight deviation from t_m to t_f.

Figure 9.22 show the robot paths and the centers with respect to time. The black line is the path of the alpha robot. The robots avoid each other and follow the same formation function with center x_c and y_c given at different time steps. Figure 9.23 demonstrates that the three beta swarm members stay an acceptable distance away from one another throughout the mission.

A video demonstrating this experiment can be found at http://www.csee.usf.edu/USL/Videos/UAV-UGV-Swarm.wmv.

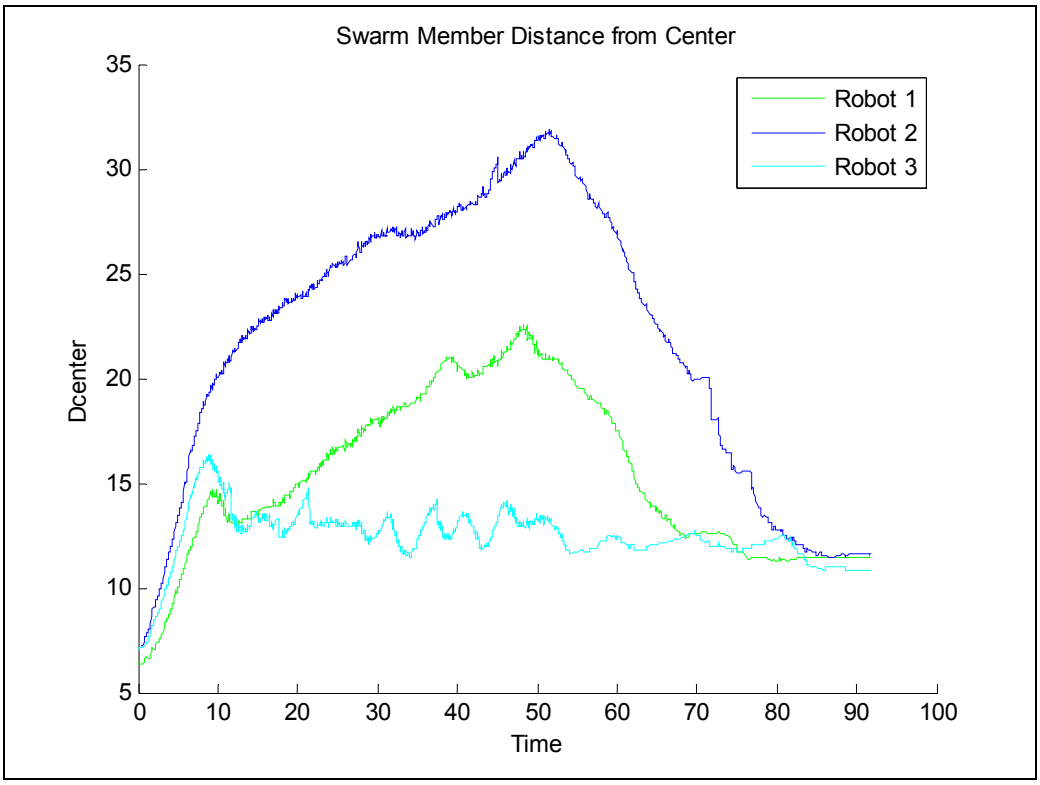

Figure 9.20. Experiment 6: Robot distance from center of swarm (x_c, y_c).

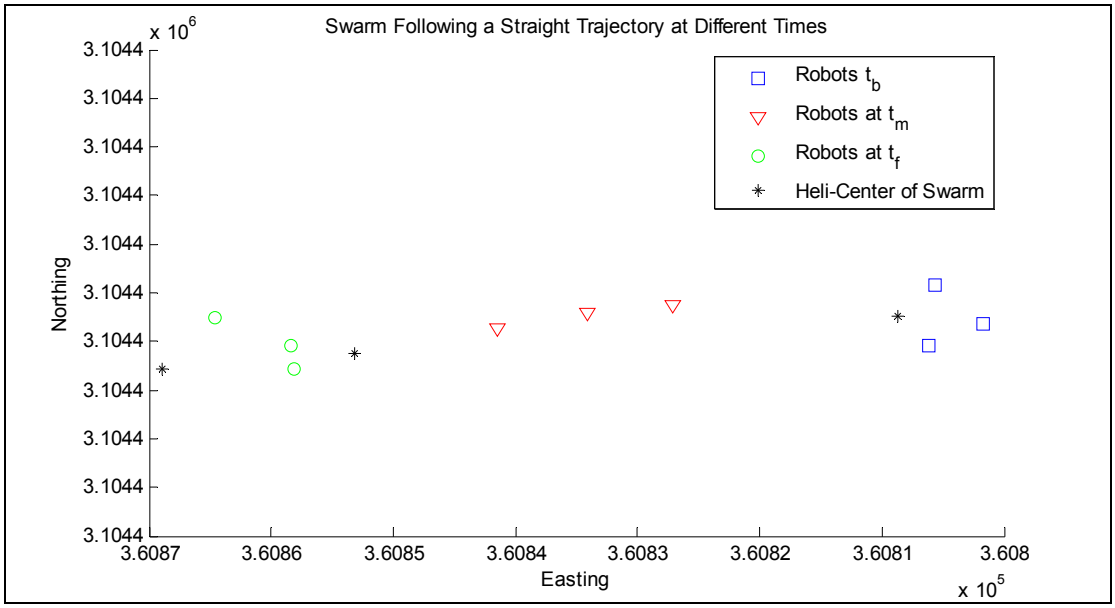

Figure 9.21. Experiment 6: Robot formation at beginning (t_b), middle (t_m), and end (t_f) of mission.

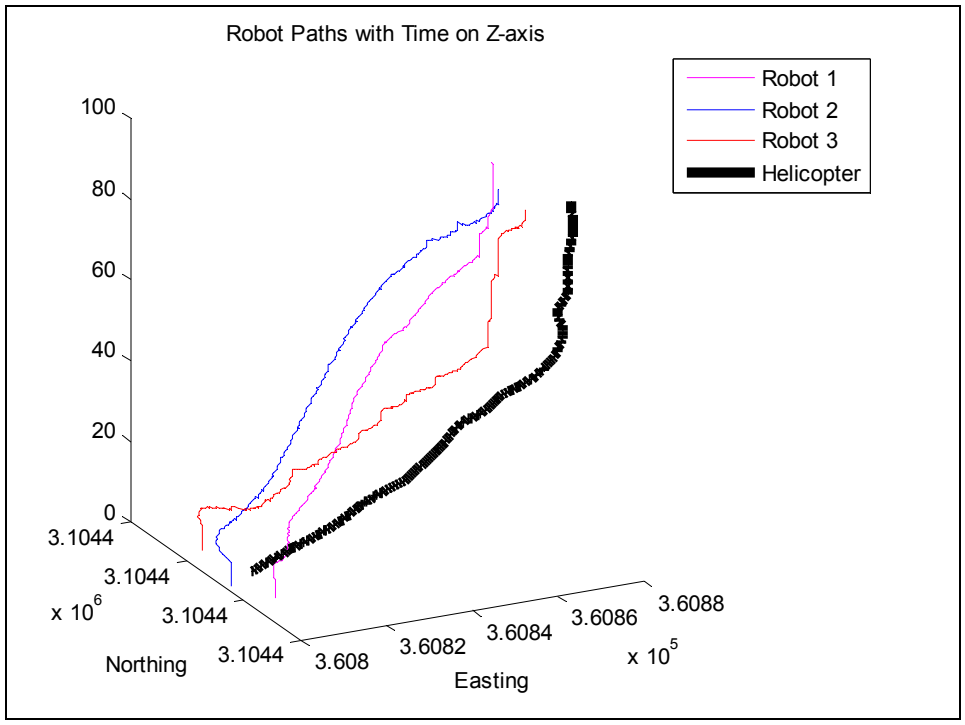

Figure 9.22. Experiment 6: Robot paths with respect to center (x_c, y_c) with time on z-axis.

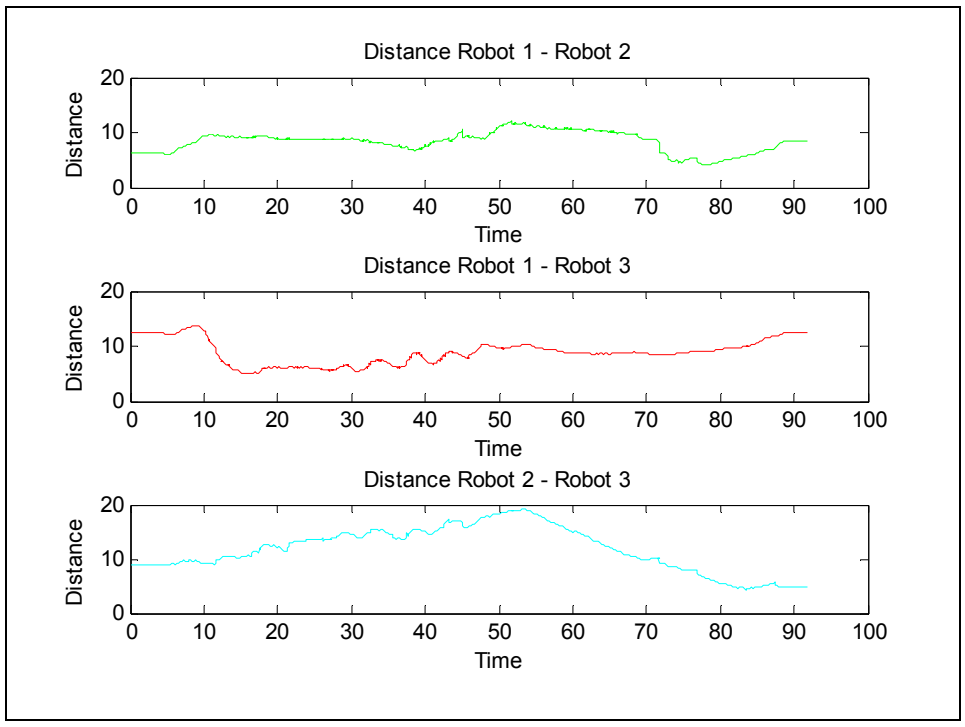

Figure 9.23. Experiment 6: Distance between swarm members.

Chapter 10

Future Approach Utilizing Deformable Ellipses

10.1 Technical Approach

The objective of this expanded formation control methodology is to attract elements of a swarm into a bounded formation and allow the swarm to stay in that formation as it moves around the mission space. In Chapter 4, a vector field is designed to attract swarm members to an ellipse surrounding a convoy of vehicles. By modifying the weights on the vectors in the field, the minimum distance between the swarm members and the convoy were controlled.

The location of the center (x_c, y_c), relative to the initial location of the swarm members gives some control over the shape of the final formation. If the center of the ellipse is set below the initial locations of the swarm members, the swarm members eventually form into a shallow wedge formation. By locating the center above the initial location of the swarm members, the same vector field produces a shallow "vee" formation. All formations derived from the approach in Chapter 4 are subsets of elliptical curves – formation vertices (if they exist) are rounded and the formations could be considered sloppy in some applications. In the next section, the formations are sharpened by changing the metric in Equation (4.1).

10.2 Improving Static Formations

In order to generalize the formation control strategy discussed in Chapter 4, Equation (4.1) will be generalized as:

$$z(x, y) = e^{-\alpha N(x,y)} \tag{10.1}$$

With

$$N(x,y) > 0 \; for\; all \; (x,y) \in R^2 \tag{10.2}$$

In Chapter 4, a modified Euclidian distance metric as the function N was used. However any function satisfying the conditions in Equation (10.2) could be used. The vector field given in Chapter 4 can be generalized as:

$$\begin{pmatrix} v_x \\ v_y \end{pmatrix} = \frac{-w_{out}(x,y)}{\sqrt{N_x^2 + N_y^2}} \begin{pmatrix} N_x \\ N_y \end{pmatrix}, \; for\; all \; (N_x, N_y) \neq (0,0) \tag{10.3}$$

with the weight $w_{out}(x,y)$ given by:

$$w_{out}(x,y) = \frac{1}{1 + e^{-\alpha_{out}(N(x,y)-(R^* + \Delta R))}} \tag{10.4}$$

The sharpness of the formations can be adjusted by using an alternate metric such as a modified absolute metric given in Equation (10.5):

$$N(x,y) = |x - x_c| + \gamma |y - y_c|, \quad \gamma > 0 \tag{10.5}$$

In Figures 10.1a and 10.1b wedge formations based on two different norms are compared– the modified Euclidean metric (Figure 10.1a) as in Chapter 4 and the modified absolute metric (Figure 10.1b). The formation in Figure 10.1a does not have the crisp lines shown in Figure 10.1b. However, the absolute metric also has some undesirable characteristics. Following the paths of individual swarm members shown as black lines in the figure, several members are attracted to the vertex of the formation. In practice, swarm members use both the formation vector field and an obstacle vector field to control their movements so they avoid each other and spread out along the formation.

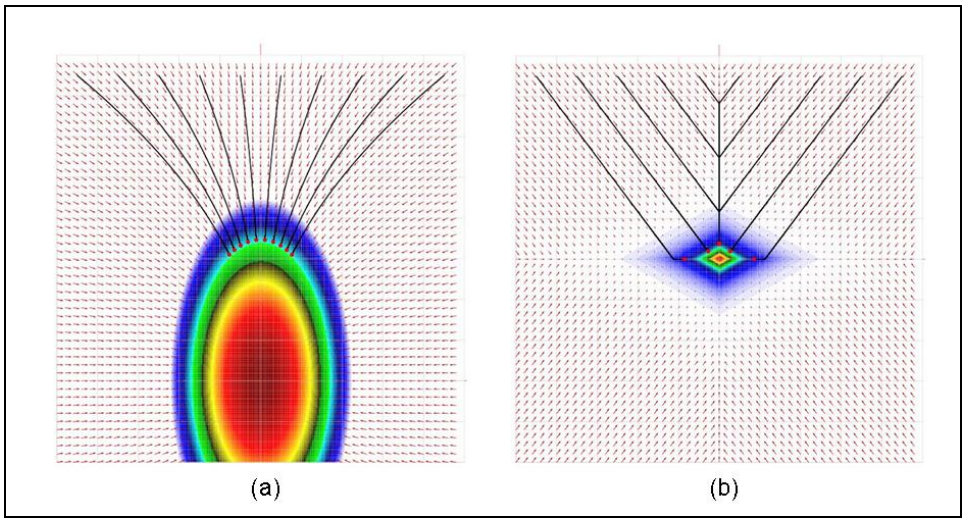

Figure 10.1. Examples of wedge formations. (a) the shallow wedge is produced by using a modified Euclidean norm, and (b) the Euclidean norm was replaced with the absolute norm to produce a sharper wedge.

Using a large positive power of Equation (4.1) as *N(x,y)*, results in the box-like surface shown as a contour map in Figure 10.2. In this case, swarm members are attracted to well defined, bounded line segments. Swarm members are attracted to bounded vertical and horizontal line segments. A box distribution is defined by:

$$z(x,y) = e^{-\alpha((x-x_c)^2+\lambda(y-y_c)^2)^\rho}, \rho > 1 \tag{10.6}$$

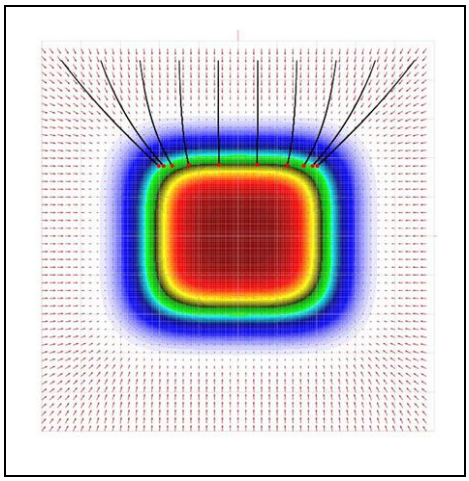

Figure 10.2. A box distribution formation.

If it can be assumed that swarm members are initially in some bounded neighborhood, the center of the surface defined by Equation (10.1) can be chosen to attract swarm members into "vee", wedge, line or column formations. Unfortunately, some of these formations can be hard to control as the formation moves through a dynamic environment since the movement of the formation may push some of the individual swarm members out of alignment. The most predictable control of the swarm is obtained when the movement of the center leads the movement of the swarm members. Examining the vector fields in the figures above, it is clear that swarm members initially located in the lower half of the graph will always stay below the center of the surface so formations based on the lower half of the surface can be lead through a dynamic environment. In Section 10.3, the surface described in Equation (10.1) will be transformed so that swarm members starting below the center can achieve and maintain any of the 4 formations of interest.

10.3 Bending the Ellipsoid

Examining Figure 10.1a, it is easy to see that swarm members initially located in the lower half plane can be attracted into horizontal line or "vee" formations. Intuitively, if the formation surface and the associated vector field could be "bent" along the y-axis, the swarm members would be attracted into additional formations including wedge and column formations. Mathematically, bending the surface involves changing the function *N(x,y)* from Equation (10.1).

Using Equation (4.1) as a model, Let z(X,Y) be an ellipsoid is defined in a reference domain with variables *X* and *Y* with a center *(*0, 0*)*:

$$z(X,Y) = e^{-\alpha(X^2+\gamma Y^2)} \tag{10.7}$$

The function z(X,Y) will be transformed it to the desired surface in the physical domain. In this section, it is assumed that the bent ellipse has a center *(0, 0)*. In Section 10.4, translation and rotation about the axis and the center will be shown.

Let $\psi(x)$ be the "bending" function in the x-y plane. In this case the bending function is a parabola:

$$\mathcal{Y}(x) = X_L^2 - r^2 x^2 \tag{10.8}$$

The bending function could have other forms depending on the formations desired. In a sense, the function $\mathcal{Y}$ bends the X-axis of the original ellipse. Suppose that the variable X from Equation (10.7) is defined as:

$$X = x \tag{10.9}$$

and

$$Y = y - X_L^2 + r^2 x^2 \tag{10.10}$$

Then Equation (10.7) can be rewritten as:

$$z(x,y) = e^{-\alpha(X^2+\gamma Y^2)} = e^{-\alpha(x^2+\gamma(y-X_L^2+r^2x^2)^2)} \tag{10.11}$$

The parameters X_L and r control the bend of the ellipse. An example of a bent ellipsoid is given in Figure 10.3a. The surface gradient vector is given by:

$$\begin{pmatrix} g_x \\ g_y \end{pmatrix} = -2\alpha z(x,y) \begin{pmatrix} x + 2r^2 x\lambda(y - X_L^2 + r^2x^2) \\ \gamma(y - X_L^2 + r^2x^2) \end{pmatrix} \tag{10.12}$$

As before, a weighted vector in the field of the form:

$$\begin{pmatrix} v_x \\ v_y \end{pmatrix} = \frac{-w_{out}(X(x,y),Y(x,y))}{\left\|(g_x,g_y)^t\right\|} \begin{pmatrix} g_x \\ g_y \end{pmatrix} \tag{10.13}$$

is used to attract swarm members to the surface. Here the weighting function w is defined in the reference domain by Equation (4.13) as a sigmoid function that dies off in the interior of a user-specified elliptical contour. In the physical domain, a substitution for the variables X and Y must

expressed in terms of the physical variables x and y from Equations (10.9) and (10.10). Figure 10.3b shows a vector field for a bent ellipse.

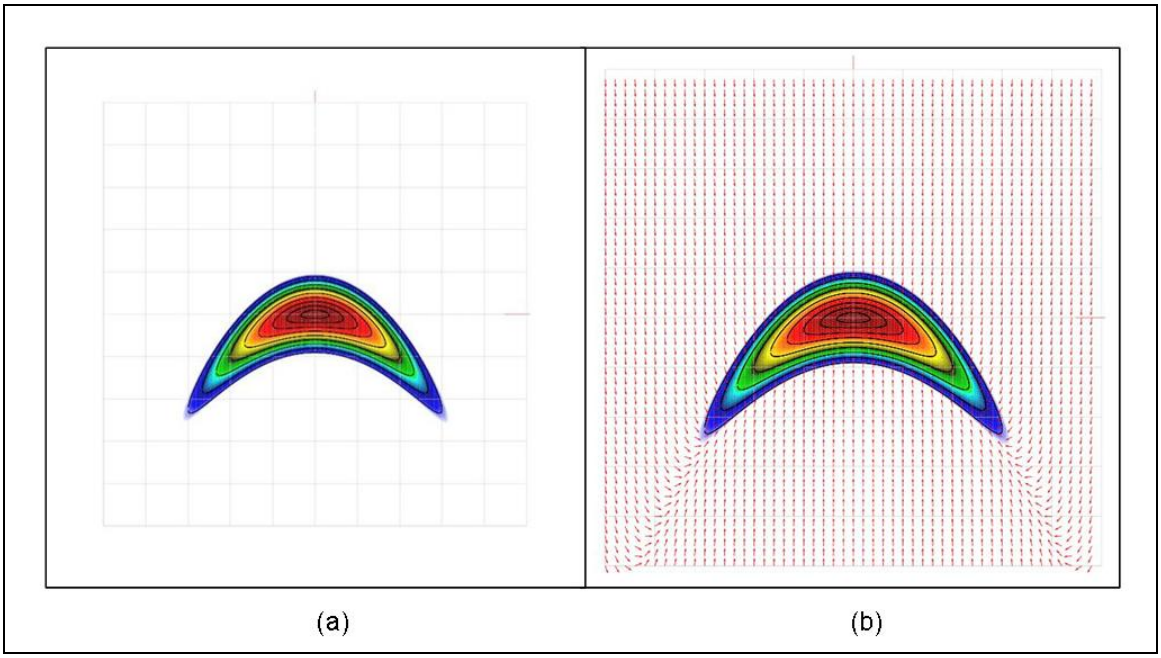

Figure 10.3. Bent ellipse. (a) without vector field, (b) with vector field.

Following the arrows in the field, most swarm members, regardless of their initial location, are attracted to the lower edge of the surface and, in this case, into a wedge formation. Also swarm members initially located within the parabolic arc described by Equation (10.8) maintain their initial separation in the x-direction.

Figure 10.4 shows 2 addition examples of vector fields based on this surface. By changing the X_L and r parameters, the distance between the 2 roots is controlled,

$$x = \pm\frac{X_L}{r} \tag{10.14}$$

In Figure 10.4a, the value of r is small relative to the value of X_L, resulting in a relatively flat parabola, so the swarm members will form a column. In Figure 10.4b, r is large relative to X_L , the roots are very close to each other, so the parabola grows rapidly, and the swarm members will form a wedge.

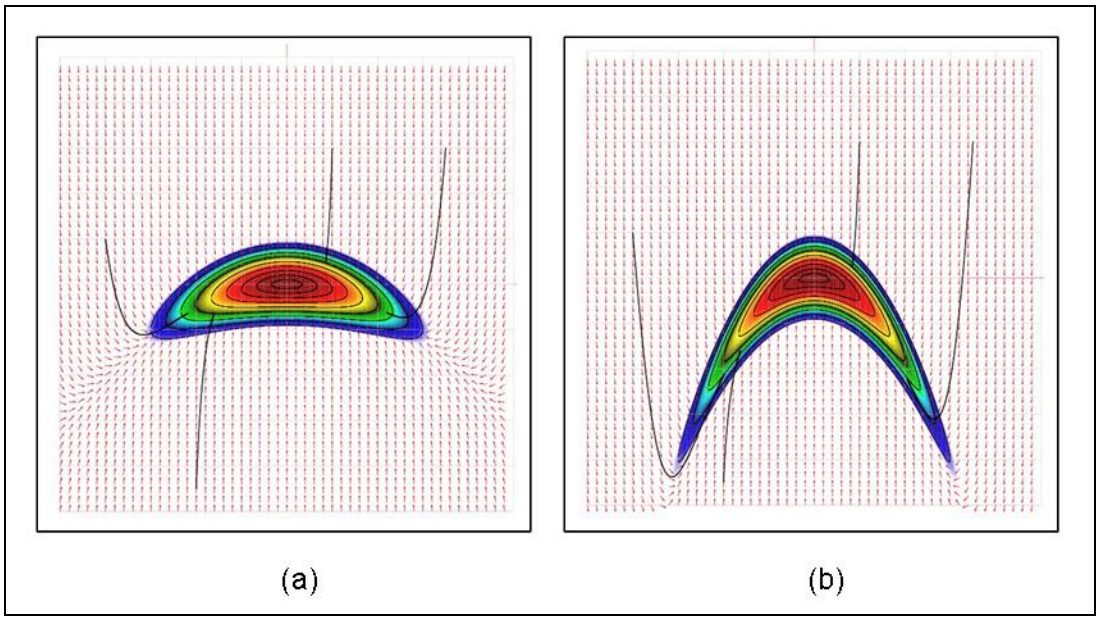

Figure 10.4. Bent ellipse examples. (a) column formation, (b) wedge formation

10.4 Translating and Rotating the Ellipsoid

The goal is to move the swarm in formation along an arbitrary path which will require translation and rotation of the formation. Translation along a path is straightforward – the center of the surface described by Equation (4.1) (or Equation (10.1) in the more general case), (x_c, y_c) can be a function of time. However, to follow a path in formation, the formation must be rotated. The simplest case is considered first by rotating the ellipsoid described in Equation (4.1). Consider a reference domain, with variables $\mathfrak{x}$ and $\mathfrak{y}$ in which the ellipsoid is centered at (*0, 0*) with an axis parallel to the $\mathfrak{x}$-axis. The equation of the surface in the $\mathfrak{x}$-$\mathfrak{y}$ domain is:

$$z(\mathfrak{x},\mathfrak{y}) = e^{-\alpha(\mathfrak{x}^2+\lambda\mathfrak{y}^2)} \tag{10.15}$$

A rotation matrix is used to map (x,y) from the physical domain into the corresponding point ($\mathfrak{x}$, $\mathfrak{y}$) from the reference domain.

$$\begin{pmatrix} \mathfrak{X} \\ \mathfrak{Y} \end{pmatrix} = \begin{pmatrix} \cos\phi_t & -\sin\phi_t \\ \sin\phi_t & \cos\phi_t \end{pmatrix} \begin{pmatrix} x - x_{c_t} \\ y - y_{c_t} \end{pmatrix} = \begin{pmatrix} (x - x_{c_t})\cos\phi_t - (y - y_{c_t})\sin\phi_t \\ (x - x_{c_t})\sin\phi_t + (y - y_{c_t})\cos\phi_t \end{pmatrix} \tag{10.16}$$

Here the angle φ_t and the center (x_{ct},y_{ct}) are functions of time. Substituting into Equation (10.14) yields:

$$z(x,y) = e^{-\alpha([(x-x_{c_t})\cos\phi_t - (y-y_{c_t})\sin\phi_t]^2 + \gamma([(x-x_{c_t})\sin\phi_t + (y-y_{c_t})\cos\phi_t]^2)} \tag{10.17}$$

where z is expressed in terms of the physical variables x and y. The gradient of Equation (10.17) can be found directly:

$$= 2\alpha z(x,y) \begin{pmatrix} (x - x_{c_t})(\cos^2\phi_t + \gamma\sin^2\phi_t) + (y - y_{c_t})\sin\phi_t\cos\phi_t(\gamma - 1) \\ (x - x_{c_t})\sin\phi_t\cos\phi_t(\gamma - 1) + (y - y_{c_t})(\cos^2\phi_t + \gamma\sin^2\phi_t) \end{pmatrix} \tag{10.18}$$

However, the gradient can better be expressed in terms of matrix operations that describe the translations and rotations from the reference to the physical domain. Returning to Equation (10.7), the gradient vector, in the physical domain, for the surface z can be written as:

$$\nabla z(x,y) = -2\alpha z(x,y)(\mathfrak{X}(x,y)\nabla\mathfrak{X} + \gamma\mathfrak{Y}(x,y)\nabla\mathfrak{Y}) \tag{10.19}$$

Substituting the values of $\mathfrak{X}$ and $\mathfrak{Y}$ from Equation (10.17) gives:

$$\nabla z(x,y) = -2\alpha z(x,y)[[(x - x_{c_t})\cos\phi_t - (y - y_{c_t})\sin\phi_t] \begin{pmatrix} \cos\phi_t \\ -\sin\phi_t \end{pmatrix} + \gamma[(x - x_{c_t})\sin\phi_t + (y - y_{c_t})\cos\phi_t] \begin{pmatrix} \sin\phi_t \\ \cos\phi_t \end{pmatrix}] \tag{10.20}$$

An easier way to look at this is to employ the rotation matrix used in Equation (10.17):

$$\nabla z(x,y) = 2\alpha z(x,y)\begin{pmatrix}\cos\phi_t & -\sin\phi_t \\ \sin\phi_t & \cos\phi_t\end{pmatrix}\begin{pmatrix}\mathfrak{X}(x,y) \\ \gamma\mathfrak{Y}(x,y)\end{pmatrix} \tag{10.21}$$

In this case, the gradient vector is computed in the reference domain and then rotated into the physical domain. Since the term $2\alpha z(x,y) > 0$ for all (x,y) and the gradient vector is normalized when the final field vector is computed, this term can be dropped from Equation (10.21). The vector used to control the movement of the rotated swarm is:

$$\begin{pmatrix}v_x \\ v_y\end{pmatrix} = \frac{-w_{out}(\mathfrak{X}(x,y),\mathfrak{Y}(x,y))}{\|\nabla z(x,y)\|}\nabla z(x,y), \quad \nabla z(x,y) \neq 0 \tag{10.22}$$

In practice, it is easiest to compute the gradient vector and the weighting function in the reference domain, then rotate it into the physical domain.

For the case of the bent ellipse, there are two coordinate transformations – one to bend the ellipse and the next to translate and rotate the ellipse into the desired position. Figure 10.5 shows a diagram of the coordinate systems involved. In this case, there are two intermediate coordinate systems. The reference coordinate systems, just as in the previous discussion in Chapter 4, contains an ellipsoid with center (0,0) oriented with 1 axis parallel to the x-axis. The "bent ellipse" coordinate system leaves the center and orientation unchanged from the reference, but bends the ellipse into the desired shape. Finally, the physical coordinate system places the surface into the desired position and orientation.

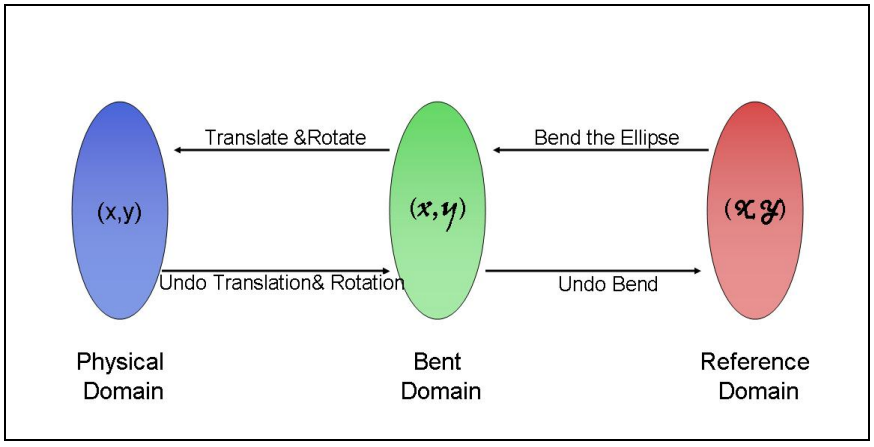

Figure 10.5. Diagram of transformations applied to vector field for the bent ellipse.

Again starting with the reference domain, with variables $\mathcal{X}$ and $\mathcal{Y}$, the equation for the ellipsoid surface is in Equation (10.14). The variables X and Y are the result of the bending operation so they are functions of the bent domain variables x and y as in Equation (10.23).

$$z(\mathcal{X}(x,y),\mathcal{Y}(x,y)) = e^{-\alpha(\mathcal{X}(x,y)^2+\lambda\mathcal{Y}(x,y)^2)} \tag{10.23}$$

Finally, the variables x and y are the result of a translation and a rotation in the physical domain with variables x and y.

$$z(\mathcal{X}(x(x,y),y(x,y)),\mathcal{Y}(x(x,y),y(x,y))) = e^{-\alpha(\mathcal{X}(x(x,y),y(x,y))^2+\gamma\mathcal{Y}(x(x,y),y(x,y))^2)} \tag{10.24}$$

To compute the vector field that controls the movement of the swarm, the chain rule is applied. Ignoring the scalar term, $\nabla z(\mathcal{X},\mathcal{Y})$ becomes:

$$\nabla z(x,y) = \begin{pmatrix} \mathcal{X}(x(x,y),y(x,y)) \\ \gamma\mathcal{Y}(x(x,y),y(x,y)) \end{pmatrix} \tag{10.25}$$

Expanding the derivatives with the chain rule and writing the matrix elements as dot products produces:

$$\nabla z(x,y) = \begin{pmatrix} \mathcal{X}\mathcal{X}_x x_x + \gamma\mathcal{Y}\mathcal{Y}_x x_x + \mathcal{X}\mathcal{X}_y y_x + \gamma\mathcal{Y}\mathcal{Y}_y y_x \\ \mathcal{X}\mathcal{X}_x x_y + \gamma\mathcal{Y}\mathcal{Y}_x x_y + \mathcal{X}\mathcal{X}_y y_y + \gamma\mathcal{Y}\mathcal{Y}_y y_y \end{pmatrix} \tag{10.26}$$

Again, recognizing the multiplication of 3 matrices gives Equation (10.26) which can further be simplified:

$$\nabla z(x,y) = \begin{pmatrix} \xi_x & \eta_x \\ \xi_y & \eta_y \end{pmatrix} \begin{pmatrix} \mathfrak{X}_\xi & \mathfrak{Y}_\xi \\ \mathfrak{X}_\eta & \mathfrak{Y}_\eta \end{pmatrix} \begin{pmatrix} \mathfrak{X} \\ \gamma\mathfrak{Y} \end{pmatrix} \tag{10.27}$$

Now the definitions of the coordinates are applied to express Equation (10.27) in terms of the physical coordinate system. First, as in Equation (4.3) the rotated x and y matrix is defined by:

$$\begin{pmatrix} \xi \\ \eta \end{pmatrix} = \begin{pmatrix} (x - x_{c_t})\cos\phi_t - (y - y_{c_t})\sin\phi_t \\ (x - x_{c_t})\sin\phi_t + (y - y_{c_t})\cos\phi_t \end{pmatrix}. \tag{10.28}$$

so

$$\begin{pmatrix} \xi x & \eta x \\ \xi y & \eta y \end{pmatrix} = \begin{pmatrix} \cos\phi_t & \sin\phi_t \\ -\sin\phi_t & \cos\phi_t \end{pmatrix} \tag{10.29}$$

As stated previously, $\mathfrak{X}$ and $\mathfrak{Y}$ are:

$$\begin{pmatrix} \mathfrak{X} \\ \mathfrak{Y} \end{pmatrix} = \begin{pmatrix} \xi \\ \eta - X_L{}^2 + r^2\xi^2 \end{pmatrix} \tag{10.30}$$

so

$$\begin{pmatrix} \mathfrak{X}_\xi & \mathfrak{X}_\eta \\ \mathfrak{Y}_\xi & \mathfrak{Y}_\eta \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 2r^2\xi & 1 \end{pmatrix} \tag{10.31}$$

Substituting in the definition of ξ in Equation (10.28) gives:

$$\begin{pmatrix} \mathfrak{X}_\xi & \mathfrak{X}_\eta \\ \mathfrak{Y}_\xi & \mathfrak{Y}_\eta \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 2r^2[(x - x_{c_t})\cos\phi_t - (y - y_{c_t})\sin\phi_t] & 1 \end{pmatrix} \tag{10.32}$$

Finally, expressing the vector from Equation (10.27) in terms of x and y yields:

$$\nabla z(x,y) = \begin{pmatrix} \cos\phi_t & \sin\phi_t \\ -\sin\phi_t & \cos\phi_t \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 2r^2[(x-x_{c_t})\cos\phi_t - (y-y_{c_t})\sin\phi_t & 1 \end{pmatrix} \begin{pmatrix} (x-x_{c_t})\cos\phi_t - (y-y_{c_t})\sin\phi_t \\ \lambda((x-x_{c_t})\sin\phi_t + (y-y_{c_t})\cos\phi_t) \end{pmatrix} \tag{10.33}$$

Multiplying Equation (10.29), (10.32) and (10.33) together gives the gradient vector, which can be substituted into Equation (10.22) to get the final field vector. To understand the weighting function, it is easiest to consider this function in the reference domain, in that domain; the weight is a sigmoid that depends on the modified distance to the center of the ellipse. Figure 10.6 illustrates a rotated and bent ellipse.

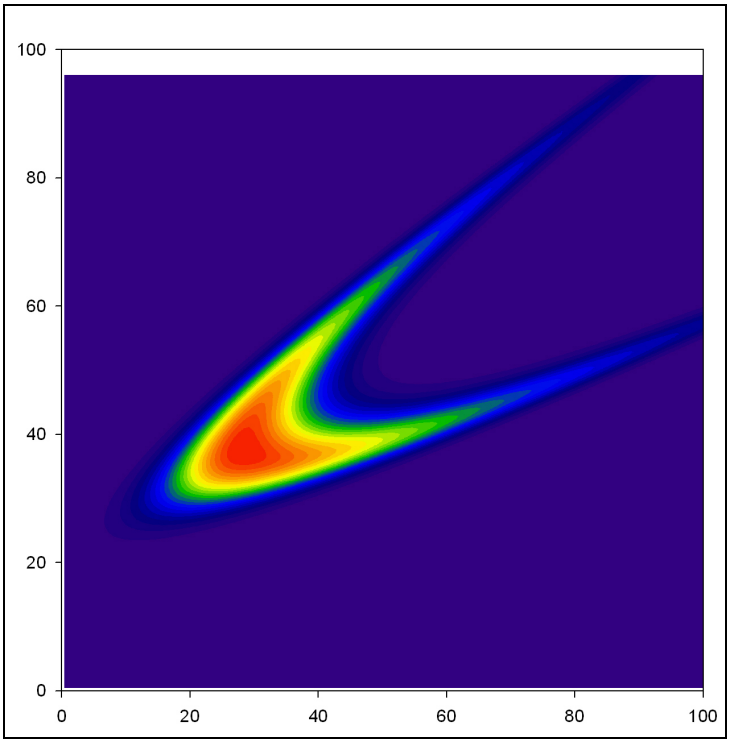

Figure 10.6. A rotated and bent ellipse.

Chapter 11

Conclusions and Future Work

11.1 Conclusions

In this work, a methodology for attracting members of a swarm or other multi-agent system into a formation has been presented. Potential functions together with limiting functions can be successfully utilized to control robot swarm formation, obstacle avoidance and the overall swarm movement. The presented method supports scalability, different swarm sizes, multiple formations, heterogeneous swarm member teams, centralized and decentralized formation control.

These formations can move as a unit, adapt to non-uniform surfaces and change dynamically. A mathematical surface pulls the swarm members around in formation much like a magnet. By adjusting the parameters on the surface, the shape and extent of the formation can be controlled. Formations can change dynamically by making the surface parameters functions of time. Since our approach is based on potential field methodology, it is scalable to very large swarms and it is relatively tolerant to positional uncertainties for individual swam members. Finally, our approach requires relatively little information to be transmitted to the swarm

The advantage of this approach in comparison to others is the simplicity of the functions and the ease of formation changes. There is no expensive tuning involved and little information is required for each robot to adhere to formation constraints.

11.2 Future Work

In the future, the work will be expanded to include more formations. In addition, the fuzzy logic tuner for the control variables for the swarm functions will be expanded. The approach will also be expanded to unmanned aerial vehicles (UAVs) by making the formation function a trivariate normal.

The ongoing research objective is to develop a framework and methodology for swarm formation control by combining control and learning techniques to model swarms and allow heterogeneous swarms of ground and/or aerial vehicles to maintain formation while avoiding collisions and handling dynamic changes in the environment. The goal of the design is to easily maintain formation of a group of UGVs/UAVs which is not dependent on the size of the swarm or the type of platform and is transferable to multiple formation types.

The long term goal is to apply this technique to large numbers of autonomously functioning vehicles which can be used for critical missions such as convoy support. Learning can also be applied in the environment (e.g. terrain changes, potential threats) where the swarm formation can modify based on environmental parameters.

References

[1] M. Fields, "Modeling large scale troop movement using reaction diffusion equations," U.S. Army Research Laboratory, Aberdeen Proving Ground, MD 1993.

[2] M. Fields, "Preliminary applications of the variable resolution terrain model to a troop movement model," U.S. Army Research Laboratory, Aberdeen Proving Ground, MD 1995.

[3] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proceedings of IEEE International Conference on Robotics and Automation*, 1991.

[4] W. Kang, N. Xi, Y. Zhao, J. Tan, and Y. Wang, "Formation control of multiple autonomous vehicles: Theory and experimentation," in *Proceedings of IFAC 15th Triennial World Congress*, 2002.

[5] K. Fujibayashi, S. Murata, K. Sugawara, and M. Yamamura, "Self- organizing formation algorithm for active elements," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2000.

[6] T. Balch and M. Hybinette, "Social potentials for scalable multi-robot formations," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2000.

[7] N. Leonard and E. Fiorelli, "Leaders, artificial potentials and coordinated control of groups," in *Conference on Decision and Control*, 2001.

[8] H. R. Everett, "Robotic security systems," *IEEE Instrumentation & Measurement Magazine,* vol. 6, pp. 30-34, 2003.

[9] F. Miyawaki, K. Masamune, S. Suzuki, K. Yoshimitsu, and J. Vain, "Scrub nurse robot system-intraoperative motion analysis of a scrub nurse and timed-automata-based model for surgery," *IEEE Transactions on Industrial Electronics,* vol. 52, pp. 1227-1235, 2005.

[10] J. L. Jones, "Robots at the tipping point: the road to iRobot Roomba," *IEEE Robotics & Automation Magazine,* vol. 13, pp. 76-78, 2006.

[11] H. Sahin and L. Guvenc, "Household robotics: autonomous devices for vacuuming and lawn mowing [Applications of control]," *IEEE Control Systems Magazine,* vol. 27, pp. 20-96, 2007.

[12] M. J. Mataric, M. Nilsson, and K. T. Simsarin, "Cooperative multi-robot box-pushing," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1995, pp. 556-561.

[13] Y. U. Cao, A. S. Fukunaga, and A. B. Kahng, "Cooperative Mobile Robotics: Antecedents and Directions," *Autonomous Robots,* vol. 4, pp. 1-23, 1997.

[14] A. Shmilovici, F. Ramkddam, B. Lopez, and J. L. de la Rosa, "Measuring progress in multirobot research with rating methods - the RoboCup example," *IEEE Transactions on Systems, Man and Cybernetics, Part B,* vol. 34, pp. 1305-1308, 2004.

[15] H. Sugiyama, T. Tsujioka, and M. Murata, "QoS routing in a multi-robot network system for urban search and rescue," in *International Conference on Advanced Information Networking and Applications*, 2006.

[16] X. Zheng, S. Jain, S. Koenig, and D. Kempe, "Multi-robot forest coverage," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 3852-3857.

[17] K. Sugawara and T. Watanabe, "Swarming robots-foraging behavior of simple multirobot system," in *IEEE/RSJ International Conference on Intelligent Robots and System*, 2002, pp. 2702-2707.

[18] L. E. Parker, "ALLIANCE: an architecture for fault tolerant multirobot cooperation," *IEEE Transactions on Robotics and Automation,* vol. 14, pp. 220-240, 1998.

[19] Z. Cao, M. Tan, S. Wang, Y. Fan, and B. Zhang, "The optimization research of formation control for multiple mobile robots," in *Proceedings of the 4th World Congress on Intelligent Control and Automation*, 2002, pp. 1270-1274.

[20] L. Chaimowicz and V. Kumar, "Aerial Shepherds: Coordination among UAVs and Swarm Robots," in *International Symposium on Distributed Autonomous Robotic Systems*, 2004.

[21] L. Chaimowicz and V. Kumar, "A Framework for the Scalable Control of Swarms of Unmanned Ground Vehicles with Unmanned Aerial Vehicles," in *Proceedings of the International Conference on Robotics and Remote Systems for Hazardous Environments*, 2004.

[22] M. Egerstedt and H. Xiaoming, "Formation constrained multi-agent control," *IEEE Transactions on Robotics and Automation,* vol. 17, pp. 947-951, 2001.

[23] W. Kowalczyk, "Target assignment strategy for scattered robots building formation," in *Proceedings of the Third International Workshop on Robot Motion and Control*, 2002, pp. 181-185.

[24] S. Zelinski, T. J. Koo, and S. Sastry, "Optimization-based formation reconfiguration planning for autonomous vehicles," in *IEEE International Conference on Robotics and Automation*, 2003, pp. 3758-3763.

[25] T. Balch and M. Hybinette, "Social potentials for scalable multi-robot formations," in *Proceedings IEEE International Conference on Robotics and Automation* 2000, pp. 73-80.

[26] J. P. Desai, "Modeling multiple teams of mobile robots: a graph theoretic approach," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001, pp. 381-386.

[27] D. D. Dudenhoeffer and M. P. Jones, "A formation behavior for large-scale micro-robot force deployment," in *Proceedings of the Simulation Conference* 2000, pp. 972-982.

[28] J. Fredslund and M. J. Mataric, "A general algorithm for robot formations using local sensing and minimal communication," *IEEE Transactions onRobotics and Automation,* vol. 18, pp. 837-846, 2002.

[29] K. Fujibayashi, S. Murata, K. Sugawara, and M. Yamamura, "Self-organizing formation algorithm for active elements," in *Proceedings of the IEEE Symposium on Reliable Distributed Systems*, 2002, pp. 416-421.

[30] A. Jadbabaie, L. Jie, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control,* vol. 48, pp. 988-1001, 2003.

[31] P. Kostelnik, M. Samulka, and M. Janosik, "Scalable multi-robot formations using local sensing and communication," in *Proceedings of the Third International Workshop on Robot Motion and Control*, 2002, pp. 319-324.

[32] N. E. Leonard and E. Fiorelli, "Virtual leaders, artificial potentials and coordinated control of groups," in *Proceedings of the IEEE Conference on Decision and Control*, 2001, pp. 2968-2973.

[33] M. Lindhe, P. Ogren, and K. H. Johansson, "Flocking with Obstacle Avoidance: A New Distributed Coordination Algorithm Based on Voronoi Partitions," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2005, pp. 1785-1790.

[34] R. Olfati-Saber and R. M. Murray, "Graph rigidity and distributed formation stabilization of multi-vehicle systems," in *Proceedings of the 41st IEEE Conference on Decision and Control*, 2002, pp. 2965-2971.

[35] K. Sugihara and I. Suzuki, "Distributed algorithms for formation of geometric patterns with many mobile robots," *Robot Systems,* vol. 13, 1996.

[36] T. Fukuda, S. Ito, F. Arai, Y. Yokoyama, Y. Abe, K. Tanaka, and Y. Tanaka, "Navigation system based on ceiling landmark recognition for autonomous mobile robot-landmark detection based on fuzzy template matching (FTM)," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1995.

[37] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "A taxonomy for swarm robots," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1993, pp. 441-447.

[38] T. Fukuda, D. Funato, K. Sekiyama, and F. Arai, "Evaluation on flexibility of swarm intelligent system," in *IEEE International Conference on Robotics and Automation*, 1998, pp. 3210-3215.

[39] E. Bonabeau and G. Théraulaz, "Swarm smarts," in *Scientific American*, March 2000, pp. 72-79.

[40] K. Donahue, L. A. Lewis, and S. S. Schneider, "The role of the vibration signal in the house-hunting process of honey bee (Apis mellifera) swarms," *Behavioral Ecology and Sociobiology,* vol. 54, pp. 593-600, October 2003.

[41] J. A. Freund, L. Schimansky-Geier, B. Beisner, A. Neiman, D. F. Russell, T. Yakusheva, and F. Moss, "Behavioral Stochastic Resonance: How the Noise from a Daphnia Swarm Enhances Individual Prey Capture by Juvenile Paddlefish," *Journal of Theoretical Biology,* vol. 214, pp. 71-83, January 2002.

[42] L. Barnes, W. Alvis, M. Fields, K. Valavanis, and W. Moreno, "Heterogeneous Swarm Formation Control Using Bivariate Normal Functions to Generate Potential Fields," in *IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications*, 2006, pp. 85-94.

[43] L. Barnes, W. Alvis, M. A. Fields, K. Valavanis, and W. Moreno, "Swarm Formation Control with Potential Fields Formed by Bivariate Normal Functions," in *14th Mediterranean Conference on Control and Automation*, 2006, pp. 1-7.

[44] L. Barnes, M. A. Fields, and K. Valavanis, "Unmanned Ground Vehicle Swarm Formation Control Using Potential Fields," in *15th Mediterranean Conference on Control and Automation*, 2007, pp. 1-8.

[45] L. Barnes, M. A. Fields, and K. Valavanis, "Heterogeneous Swarm Formation Control Using Bivariate Normal Functions to Generate Potential Fields," *International Transactions on Systems Science and Applications,* vol. 2, pp. 346-359, Feb. 2007.

[46] D. Zarzhitsky and D. F. Spears, "Swarm approach to chemical source localization," in *IEEE International Conference on Systems, Man and Cybernetics*, 2005, pp. 1435-1440.

[47] D. Galzi and Y. Shtessel, "UAV formations control using high order sliding modes," in *American Control Conference*, 2006.

[48] H. G. Tanner, "Switched UAV-UGV Cooperation Scheme for Target Detection," in *IEEE International Conference on Robotics and Automation*, 2007, pp. 3457-3462.

[49] K. M. Passino, *Biomimicrcy for optimization, control, and automation*. London: Springer, 2004.

[50] T. Balch and R. C. Arkin, "Behavior-based formation control for multirobot teams," *IEEE Transactions on Robotics and Automation,* vol. 14, pp. 926-939, 1998.

[51] H. Hsu and A. Liu, "Multiagent-Based Multi-team Formation Control for Mobile Robots " *Journal of Intelligent and Robotic Systems,* vol. 42, April 2005.

[52] A. K. Das, R. Fierro, V. Kumar, J. P. Ostrowski, J. Spletzer, and C. J. Taylor, "A vision-based formation control framework," *IEEE Transactions on Robotics and Automation,* vol. 18, pp. 813-825, 2002.

[53] J. P. Desai, "A graph theoretic approach for modeling mobile robot team formations," *Journal of Robotic Systems,* vol. 19, pp. 511-525, 2002.

[54] M. A. Lewis and K. H. Tan, "High Precision Formation Control of Mobile Robots Using Virtual Structures," *Autonomous Robots,* vol. 4, 1997.

[55] H. C. Hsu and A. Liu, "Applying a Taxonomy of Formation Control in Developing a Robotic System," in *IEEE International Conference on Tools with Artificial Intelligence*, 2005, pp. 3-10.

[56] H. Ando, I. Suzuki, and M. Yamashita, "Formation and agreement problems for synchronous mobile robots with limited visibility," in *Proceedings of the IEEE International Symposium on Intelligent Control*, 1995, pp. 453-460.

[57] J. Baillieul, "Remarks on a Simple Control Law for Point Robot Formations with Exponential Complexity," in *45th IEEE Conference on Decision and Control*, 2006, pp. 3357-3362.

[58] T. D. Barfoot and C. M. Clark, "Motion planning for formations of mobile robots," *Robotics and Autonomous Systems,* vol. 46, pp. 65-78, 2004.

[59] C. Belta and V. Kumar, "Motion generation for formations of robots: A geometric approach," in *IEEE International Conference on Robotics and Automation*, 2001, pp. 1245-1250 vol.2.

[60] R. Bhatt, T. Chin Pei, and V. Krovi, "Geometric motion planning and formation optimization for a fleet of nonholonomic wheeled mobile robots," in *IEEE International Conference on Robotics and Automation*, 2004, pp. 3276-3281 Vol.4.

[61] E. Bicho and S. Monteiro, "Formation control for multiple mobile robots: a non-linear attractor dynamics approach," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003, pp. 2016-2022.

[62] L. Chaimowicz, N. Michael, and V. Kumar, "Controlling Swarms of Robots Using Interpolated Implicit Functions," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2005, pp. 2487-2492.

[63] Y.-L. Chuang, Y. R. Huang, M. R. D'Orsogna, and A. L. Bertozzi, "Multi-Vehicle Flocking: Scalability of Cooperative Control Algorithms using Pairwise Potentials," in *IEEE International Conference on Robotics and Automation*, 2007, pp. 2292-2299.

[64] T. J. Koo and S. M. Shahruz, "Formation of a group of unmanned aerial vehicles (UAVs)," in *Proceedings of the American Control Conference*, 2001, pp. 69-74.

[65] N. Michael, C. Belta, and V. Kumar, "Controlling three dimensional swarms of robots," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2006, pp. 964-969.

[66] S. Kalantar and U. R. Zimmer, "Contour shaped formation control for autonomous underwater vehicles using canonical shape descriptors and deformable models," in *MTS/IEEE TECHNO-OCEAN*, 2004, pp. 296-307 Vol.1.

[67] J. Shao, J. Yu, and L. Wang, "Formation Control of Multiple Biomimetic Robotic Fish," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 2503-2508.

[68] R. W. Beard, J. Lawton, and F. Y. Hadaegh, "A feedback architecture for formation control," in *Proceedings of the American Control Conference*, 2000, pp. 4087-4091.

[69] J. Lawton and R. W. Beard, "Synchronized Multiple Spacecraft Rotations," *Automatica,* vol. 38, pp. 1359-1364, 2000.

[70] C. M. Saaj, V. Lappas, and V. Gazi, "Spacecraft Swarm Navigation and Control Using Artificial Potential Field and Sliding Mode Control," in *IEEE International Conference on Industrial Technology*, 2006, pp. 2646-2651.

[71] X. Yun, G. Alptekin, and O. Albayrak, "Line and Circle Formation of Distributed Physical Mobile Robots," *Journal of Robotic Systems,* vol. 14, pp. 63-76, 1997.

[72] M. Egerstedt and H. Xiaoming, "Formation constrained multi-agent control," in *IEEE International Conference on Robotics and Automation*, 2001, pp. 3961-3966.

[73] H. C. H. Hsu and A. Liu, "Multiple teams for mobile robot formation control," in *Proceedings of the IEEE International Symposium on Intelligent Control*, 2004, pp. 168-173.

[74] V. Gazi, "Swarm aggregations using artificial potentials and sliding-mode control," *IEEE Transactions on Robotics and Automation,* vol. 21, pp. 1208-1214, 2005.

[75] W. Ren and R. W. Beard, "Decentralized scheme for spacecraft formation flying via the virtual structure approach," *Journal of Guidance, Control, and Dynamics,* vol. 27, pp. 73-82, 2004.

[76] G. H. Elkaim and R. J. Kelbley, "A lightweight formation control methodology for a swarm of non-holonomic vehicles," in *IEEE Aerospace Conference*, 2006.

[77] M. A. Hsieh, S. Loizou, and V. Kumar, "Stabilization of Multiple Robots on Stable Orbits via Local Sensing," in *IEEE International Conference onRobotics and Automation*, 2007, pp. 2312-2317.

[78] M. Ji and M. Egerstedt, "Distributed Coordination Control of Multiagent Systems While Preserving Connectedness," *IEEE Transactions on Robotics and Automation,* vol. 23, pp. 693-703, 2007.

[79] S. S. Ge and C. H. Fua, "Queues and Artificial Potential Trenches for Multirobot Formations," *IEEE Transactions on Robotics and Automation,* vol. 21, pp. 646-656, 2005.

[80] G. Antonelli, F. Arrichiello, S. Chakraborti, and S. Chiaverini, "Experiences of formation control of multi-robot systems with the Null-Space-based Behavioral Control," in *IEEE International Conference on Robotics and Automation*, 2007, pp. 1068-1073.

[81] C. Belta and V. Kumar, "Abstraction and control for Groups of robots," *IEEE Transactions on Robotics and Automation,* vol. 20, pp. 865-875, 2004.

[82] Z. Cao, L. Xie, B. Zhang, S. Wang, and M. Tan, "Formation constrained multi-robot system in unknown environments," in *IEEE International Conference on Robotics and Automation*, 2003, pp. 735-740.

[83] Y. Guohua, H. O. Wang, K. Tanaka, and G. Zhi-Hong, "Managing group behaviors in swarm systems by associations," in *American Control Conference*, 2006.

[84] B. Liu, R. Zhang, and C. Shi, "Formation Control of Multiple Behavior-based robots," in *International Conference on Computational Intelligence and Security*, 2006, pp. 544-547.

[85] J. H. Reif and H. Wang, "Social potential fields: a distributed behavioral control for autonomous robots," in *Proceedings of the workshop on Algorithmic foundations of robotics*, 1995, pp. 331-345.

[86] V. Gazi, B. Fidan, Y. S. Hanay, and M. İ. Köksal, "Aggregation, foraging, and formation control of swarms with non-holonomic agents using potential functions and sliding mode techniques," *Turkish Journal of Electrical Engineering and Computer Sciences,* July 2007.

[87] S. S. Ge, C.-H. Fua, and W.-M. Liew, "Swarm formations using the general formation potential function," in *IEEE Conference on Robotics, Automation and Mechatronics*, 2004, pp. 655-660.

[88] D. H. Kim, H. O. Wang, Y. Guohua, and S. Seiichi, "Decentralized control of autonomous swarm systems using artificial potential functions: analytical design guidelines," in *43rd IEEE Conference on Decision and Control*, 2004, pp. 159-164.

[89] R. Olfati-Saber and R. M. Murray, "Distributed Cooperative Control of Multiple Vehicle Formations using Structural Potential Functions," in *15th IFAC World Congress*, July 2002.

[90] J. Yao, R. Ordonez, and V. Gazi, "Swarm Tracking Using Artificial Potentials and Sliding Mode Control," in *45th IEEE Conference on Decision and Control*, 2006, pp. 4670-4675.

[91] J. R. T. Lawton, R. W. Beard, and B. J. Young, "A decentralized approach to formation maneuvers," *IEEE Transactions on Robotics and Automation,* vol. 19, pp. 933-941, 2003.

[92] S. Monteiro and E. Bicho, "A dynamical systems approach to behavior-based formation control," in *IEEE International Conference on Robotics and Automation*, 2002, pp. 2606-2611.

[93] S. S. Ge, C.-H. Fua, and K. W. Lim, "Multi-robot formations: queues and artificial potential trenches," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2004, pp. 3345-3350.

[94] G. L. Mariottini, F. Morbidi, D. Prattichizzo, G. J. Pappas, and K. Daniilidis, "Leader-Follower Formations: Uncalibrated Vision-Based Localization and Control," in *IEEE International Conference on Robotics and Automation*, 2007, pp. 2403-2408.

[95] J. Shao, G. Xie, and L. Wang, "Leader-following formation control of multiple mobile vehicles," *IET Control Theory & Applications,* vol. 1, pp. 545-552, 2007.

[96] R. Olfati-Saber, W. B. Dunbar, and R. M. Murray, "Cooperative control of multi-vehicle systems using cost graphs and optimization," in *Proceedings of the American Control Conference*, 2003, pp. 2217-2222.

[97] J. Fredslund and M. J. Mataric, "Robot formations using only local sensing and control," in *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 2001, pp. 308-313.

[98] J. P. Desai, J. Ostrowski, and V. Kumar, "Controlling formations of multiple mobile robots," in *Proceedings of IEEE International Conference on Robotics and Automation*, 1998, pp. 2864-2869.

[99] M. Sisto and D. Gu, "A Fuzzy Leader-Follower Approach to Formation Control of Multiple Mobile Robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 2515-2520.

[100] R. Fierro and A. K. Das, "A modular architecture for formation control," in *Proceedings of the Third International Workshop on Robot Motion and Control*, 2002, pp. 285-290.

[101] K.-H. Tan and M. A. Lewis, "Virtual structures for high-precision cooperative mobile robotic control," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996, pp. 132-139.

[102] W. B. Dunbar and R. M. Murray, "Model predictive control of coordinated multi-vehicle formations," in *Proceedings of the 41st IEEE Conference on Decision and Control*, 2002, pp. 4631-4636.

[103] H. Yamaguchi, T. Arai, and G. Beni, "A distributed control scheme for multiple robotic vehicles to make group formations," *Robotics and Autonomous Systems,* vol. 36, pp. 125-147, 2001.

[104] S. Spry and J. K. Hedrick, "Formation control using generalized coordinates," in *43rd IEEE Conference on Decision and Control*, 2004, pp. 2441-2446.

[105] M. Yamakita and M. Saito, "Formation control of SMC with multiple coordinate systems," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004, pp. 1023-1028.

[106] F. Zhang, M. Goldgeier, and P. S. Krishnaprasad, "Control of small formations using shape coordinates," in *IEEE International Conference on Robotics and Automation*, 2003, pp. 2510-2515.

[107] F. Kobayashi, N. Tomita, and F. Kojima, "Re-formation of mobile robots using genetic algorithm and reinforcement learning," in *SICE Annual Conference*, 2003, pp. 2902-2907.

[108] K. Hirota, T. Kuwabara, K. Ishida, A. Miyanohara, H. Ohdachi, T. Ohsawa, W. Takeuchi, N. Yubazaki, and M. Ohtani, "Robots moving in formation by using neural network and radial basis functions," in *IEEE International Joint Conference on Fuzzy Systems and Fuzzy Engineering Symposium*, 1995, pp. 91-94.

[109] F. Michaud, D. Letourneau, M. Guilbert, and J. M. Valin, "Dynamic robot formations using directional visual perception," in *IEEE/RSJ International Conference on Intelligent Robots and System*, 2002, pp. 2740-2745.

[110] N. Moshtagh, A. Jadbabaie, and K. Daniilidis, "Vision-based control laws for distributed flocking of nonholonomic agents," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2006, pp. 2769-2774.

[111] P. Renaud, E. Cervera, and P. Martiner, "Towards a reliable vision-based mobile robot formation control," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004, pp. 3176-3181.

[112] R. Vidal, O. Shakernia, and S. Sastry, "Formation control of nonholonomic mobile robots with omnidirectional visual servoing and motion segmentation," in *IEEE International Conference on Robotics and Automation*, 2003, pp. 584-589.

[113] J. K. Wald and C. J. Patterson, "A variable resolution terrain model for combat simulation," U.S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, MD 1994.

[114] R. D. Garcia and K. P. Valavanis, "On-Board Processing Systems for Small Unmanned Vehicles," in *European Control Conference* Kos, Greece, July 2007.

[115] R. D. Garcia, "Designing an Autonomous Helicopter Testbed: From Conception through Implementation," in *Computer Science and Engineering* Tampa: University of South Florida, 2008.

[116] J. Snyder, "Map Projections - A Working Manual," in *USGS Professional Paper 1395*, 1987.

Appendices

Appendix A Nomenclature for Mathematical Variables

Table A.1. Swarm equation variables

$f(x, y)$	main swarm function
α	controls distance vector field is aloud to exist
γ	control variable in y direction (*YLength* / *XLength*)
x_c, y_c	swarm surface center in world reference frame
XLength (2A)	major axis of the swarm surface
YLength (2B)	minor axis of the swarm surface
d_x, d_y	velocity vectors in the x and y directions
x_{rot}, y_{rot}	x & y coordinates in the rotated reference frame
φ	heading between the swarm formation x-axis and the center (x_c,y_c)
R^*,R_{in} ,R_{out}	optimal, inner , and outer elliptical rings
ΔR_{in} , ΔR_{out}	Distance from the R^* band to the inner and outer boundaries of the formation band
r	Euclidean distance from swarm member to center (x_c,y_c)
x_{co}, y_{co}	obstacle center location
S_{out}, S_{in},, $N_{\perp}$	limiting functions away from the center, towards the center, and perpendicular to the center respectively
SGN	multiplier to change direction of perpendicular field about the x-axis
$\alpha_{out,}$ $\alpha_{in,}$ $\alpha_{\perp}$	control variables used in respective limiting functions $S_{out,}$ $S_{in, and}$ $N_{\perp}$
ΔR_{avoid}	desired distance to maintain from obstacles and/or other swarm members
r_{avoid}	Euclidean distance from swarm member to obstacle
S_{avoid}	limiting function to control obstacle avoidance

Table A.1 (Continued)

α_{avoid}	control variables used in S_{avoid} to control the range in which vector field around the obstacle extends
$d_{x\_avoid,,}$ d_{y_avoid}	velocity vectors around obstacles
v_x, v_y	combined velocity vectors
S_{speed}	fuzzy output variable to modify magnitude of v_x and v_y in order to control speed
dCenter	fuzzy input variable (distance from center)
dObst	fuzzy input variable (distance to nearest obstacle / swarm member)
R_{short}	minimum distance from obstacles / other swarm members
R_{long}	maximum distance from obstacles / other swarm members
dMembers	fuzzy input variable (distance to nearest neighbor)
R_{begin}	minimum acceptable distance from each swarm member
R_{end}	maximum acceptable distance from each swarm member

About the Author

Laura Barnes received a Bachelors Degree in Computer Science from Texas Tech University in 2003 and a M.S. in Computer Science from the University of South Florida in 2007. During her undergraduate program she worked as an intern IBM Global Services. During her time in graduate school she worked at the Army Research Lab at Aberdeen Proving Ground in Maryland. In addition, she worked as a graduate research assistant for the Unmanned Systems Lab and the Center for Robot-Assisted Search and Rescue. She also received a fellowship from the Oak Ridge Institute for Science and Education (ORISE).

While in the Ph.D. program at the University of South Florida, Ms. Barnes has published numerous pieces of work on swarm formation control. Her research interests encompass distributed, intelligent robotic and agent systems.