2002-06

Modeling robot swarms using agent-based simulation

Dickie, Alistair James

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

MODELING ROBOT SWARMS
USING AGENT-BASED SIMULATION

by

Alistair Dickie

June 2002

Thesis Advisor:	Gordon Bradley
Thesis Advisor:	John Hiles
Second Reader:	Arnold Buss

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2002	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Modeling Robot Swarms Using Agent-Based Simulation			**5. FUNDING NUMBERS**
6. AUTHOR Alistair Dickie			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Office of Scientific Research 801 North Randolph Street Arlington, VA 22203-1977			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release: distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT

In the near future advances in mechanical and electrical engineering will enable the production of a wide variety of relatively low cost robotic vehicles. This thesis investigates the behavior of swarms of military robots acting autonomously. The Multi-Agent Robot Swarm Simulation (MARSS) was developed for modeling the behavior of swarms of military robots. MARSS contains state, sensing, and behavioral model building tools that allow a range of complex entities and interactions to be represented. It is a model-building tool that draws theory and ideas from agent-based simulation, discrete event simulation, traditional operations research, search theory, swarm theory, and experimental design. MARSS enables analysts to explore the effect of individual behavioral factors on swarm performance. The performance response surface can be explored using designed experiments. A model was developed in MARSS to investigate the effects of increasing behavioral complexity for a search scenario involving a swarm of Micro Air Vehicles (MAV's) searching for mobile tanks in a region. Agreement between theoretical and simulated search scenarios for simple searchers was found. The effect of increased MAV sensory and behavioral capability was demonstrated to be important. Little improvement was observed in swarm performance with these capabilities, however agent performance was adversely affected by reacting to increased knowledge in the wrong way. The utility of MARSS for conducting this type of analysis was demonstrated.

14. SUBJECT TERMS Robot, Swarming, Swarm, Multi-Agent, Agent-Based, Simulation			**15. NUMBER OF PAGES** 109
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release: distribution is unlimited.

MODELING ROBOT SWARMS
USING AGENT-BASED SIMULATION

Alistair James Dickie
Captain, Australian Army
BSc(Hons), University College (University of New South Wales), 1995

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
June 2002

Author: Alistair Dickie

Approved by: Gordon Bradley
Thesis Advisor

John Hiles
Thesis Advisor

Arnold Buss
Second Reader

James Eagle
Chairman, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

In the near future advances in mechanical and electrical engineering will enable the production of a wide variety of relatively low cost robotic vehicles. This thesis investigates the behavior of swarms of military robots acting autonomously. The Multi-Agent Robot Swarm Simulation (MARSS) was developed for modeling the behavior of swarms of military robots. MARSS contains state, sensing, and behavioral model building tools that allow a range of complex entities and interactions to be represented. It is a model-building tool that draws theory and ideas from agent-based simulation, discrete event simulation, traditional operations research, search theory, swarm theory, and experimental design. MARSS enables analysts to explore the effect of individual behavioral factors on swarm performance. The performance response surface can be explored using designed experiments. A model was developed in MARSS to investigate the effects of increasing behavioral complexity for a search scenario involving a swarm of Micro Air Vehicles (MAV's) searching for mobile tanks in a region. Agreement between theoretical and simulated search scenarios for simple searchers was found. The effect of increased MAV sensory and behavioral capability was demonstrated to be important. Little improvement was observed in swarm performance with these capabilities, however agent performance was adversely affected by reacting to increased knowledge in the wrong way. The utility of MARSS for conducting this type of analysis was demonstrated.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

This work is by no means a solo effort. Assistance was provided from many fronts in order to produce this thesis. Perhaps most important has been the support provided by my wife, Narelle. Without her I would have to actually think about administrating my life. My children, Alexandra and Samuel, also provided support, for without them there is little point to do anything.

Gordon Bradley, John Hiles and Arnie Buss have been instrumental in turning ideas and concepts into this thesis. All have devoted many hours to discussion and support and are recognized in a more formal sense on the front page of this thesis. All professors that I interacted with whilst at NPS were helpful. The following deserve special mention for contributions they have made: Paul Sanchez, Susan Sanchez, Steven Pilnick, Tom Lucas, Wayne Hughes, Alan Washburn, Patricia Jacobs, Donald Gaver, John Arquilla, Eugene Paulo and Samuel Buttrey. In addition Andrzej Kapolka was particularly helpful with the 3D graphics.

The US Department of Defense provided financial support. In particular the Office of the Secretary of Defense Net Assessment, and the Air Force Office of Scientific Research provided financial support for my advisors and myself. The Defense Advanced Research Projects Agency and the Naval Research Laboratory assisted with the development of concepts.

Finally, the Australian Army deserves mention for employing me to attend the Naval Postgraduate School.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

In the near future advances in mechanical and electrical engineering will enable the production of a wide variety of relatively low cost robotic vehicles. These vehicles will be physically capable of performing many military tasks in all spheres of the battlespace. Most current and planned military robotic vehicles involve a single person controlling many vehicles. When the battlespace has thousands of robots this will become impractical. Humans will instead interact with groups of robots. Individual robots within the group will act autonomously to achieve a common goal. These groups of robots are known as swarms.

Modeling can help determine important behavioral and sub-system design considerations. Analytical models do not have the ability to answer the most pressing issues, such as how an individual robot should behave and how they should interact with each other in order to produce a desired swarm goal. Simulation can help answer these questions; in particular, agent-based simulation has constructs for representing knowledge, behavior, and interactions. The representation of these aspects of a swarming robot system is vital to understand the system as a whole.

The primary aim of the work reported in this thesis was to develop, implement, and test a model for investigating the behavior of swarms of robots. A simulation tool called the Multi-Agent Robot Swarm Simulation (MARSS) was developed. MARSS is a sophisticated simulation model-building tool that can be used by analysts to understand the contribution that individual behavioral characteristics make to group performance.

The modeling methodology described in this thesis uses ideas and technologies from agent-based simulation, discreet-event simulation, stochastic models, swarming theory, search theory, design of experiments, and statistics. No proper subset of these technologies is adequate to address the modeling questions.

The modeling of a robot swarm scenario in MARSS starts with defining the problem and understanding the system that is to be studied. The sensing process in

MARSS models agent interaction. The aim of the sensing model in MARSS is to transfer information about one agent's state to another. It consists of modeling the physical process involved with transferring energy through the environment.

The behavioral process in MARSS models agents' actions. Factors control the operations of the behavioral function. The actions of many agents produce an emergent group behavior. This behavior is measured and recorded, together with the factors that produce that response. Experiments are designed to get a good spread of factor levels over the response surface. Statistics, in particular regression trees, are then used to understand what factors contributed to the response.

This modeling method was tested on a search scenario involving Micro Air Vehicles (MAV's). The results from basic MAV search scenarios implemented in MARSS were validated against analytical results for exhaustive and random search for a moving target. In both cases the results from MARSS matched those determined analytically.

More complex scenarios searchers were created where MAV's were given a more involved behavior, allowing them to react to observed targets, each other, and targets observed by fellow searchers. The searchers were conducting a random search with these modifications. Summing the components of acceleration in different directions controlled movement. It was found that by using the movement mechanism involving accelerating towards the current way point that search performance improved over the pure random search by at least 10% regardless of the configuration of the targets.

When targets were moving in a group the most important factor affecting good search performance was acceleration away from an observed target. This was an artificial result based on the configuration of the sensor. Acceleration away from other searchers, and towards targets observed by other agents was found to have only a slight affect on performance.

The research question addressed for the MAV scenario was "How should individual agents behave to produce a desired swarm behavior?" This question was

answered for the MAV scenario by determining what factor levels contribute to good search performance.

Insight was provided into how the level of swarm performance is dependant on the level of communication by investigating the effects of being able to react to fellow searchers, and to targets found by fellow searchers. The results of this thesis suggest that the sharing of this information does not have a marked impact on the best swarm performance observed. A more interesting result is that reacting to that information in the wrong way can drastically reduce swarm performance.

The difference in swarm performance between Multi-Agent (distributed) and Single-Agent (centralized) swarm control was addressed by comparing the exhaustive search results to the distributed control of the complex scenarios. When targets were moving in a group, distributed control appeared to be much better; this result is somewhat artificial due to the sensor configuration. When targets were spread over the search area, distributed control did not achieve as well as central control, however the increased in performance observed does suggest that this may be possible.

Implementing and testing the MARSS model achieved the primary aim of this work. The utility of MARSS for conducting analysis of the behavior of robot swarms was demonstrated. Researchers that are considering investigating groups of robots have the MARSS tool available.

MARSS is available for download from http://diana.or.nps.navy.mil/~ajdickie/marss.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

1. Robots on the Future Battlefield

The practical projection of military power involves identification of targets and the delivery of energy to them. Achieving this delivery often requires placing military personnel at risk. One of the main aims of pursuing military technology is to reduce this risk, while retaining or enhancing the capability to identify and destroy targets. The development of robotic vehicles to further reduce human risk is a continuation of this work.

In the near future advances in mechanical and electrical engineering will enable the production of a wide variety of relatively low cost robotic vehicles. These vehicles will be physically capable of performing many military tasks in all spheres of the battlespace. The vehicles will have many capabilities. They will be able to navigate and move without direct humans control, process sensory information and use this to control actions, and perform any task that a machine can perform today. Indeed, many military robotic vehicles currently exist or are in the latter stages of development.

2. Towards Robot Swarms

Most current and planned military robotic vehicles involve a human in the decision making cycle while the vehicle is in operation. Indeed, in many cases there are teams of humans controlling only one robot. This ratio of robots to human controllers will be impractical when there are tens of thousands, or even hundreds of thousands, of robots operating in a future battlespace.

Direct human control also requires some communication link between the human and the vehicle. In many cases, particularly for small robotic vehicles, the size of the vehicle prohibits the allocation of weight, volume and power resources to a communication subsystem that has the capability to link to a central controller.

Both of these issues can be addressed by grouping robots together. Robots with common goals could be grouped into autonomous swarms. Humans would control the allocation of tasks to the swarm and then the swarm would act autonomously. Individual vehicles control their own actions in order to produce a desired group outcome. Humans would interact with the swarm, rather than individual vehicles.

A swarm of robots could self-organize, provided there is sufficient local communications capability for inter robot communication. Some current robot research involves giving individual vehicles behaviors so that they can navigate their environment and act autonomously. With this capability the prospect of an autonomous swarm is a reality.

3. Modeling Swarms of Robots

The development of robots capable of acting in a swarm could be very costly. To build and test tens, hundreds, or even thousands, of robots consumes scarce resources. It is important to conduct as much development as possible without actually building devices. Modeling can help determine design parameters in a number of areas.

The physical characteristics of a potential vehicle can be modeled. The cost in weight, volume and power resources for each subsystem can be used in conjunction with the payoff that each subsystem gives. This may help determine an optimum configuration for an individual robot. Behavior or tactics may be modeled to help determine how individuals should act in order to produce a desired group outcome. Complex models may take into account both physical characteristics and behavior of individuals in order to produce an overall system design.

4. Simulation as a Modeling Tool

Analytical models can be used to help answer some questions regarding swarms of robots. For example, optimization models may be used to determine the best balance between subsystems. Stochastic models may be used to analyze some aspect of a search tactic. A high level of abstraction and aggregation characterizes these analytical models.

Simulation is the only tool currently available that allows researchers to model the complex systems faced in the design of robots to operate in swarms. A characteristic of autonomous operation is the complexity associated with robot sensing, knowledge, and behavior systems. Analytic models cannot be used to provide insight to basic questions in this regard. The cheaply available power of current computing technologies allows simulation to provide insight into many such problems.

5. Agent-Based Simulation

Agent-based simulation concerns using simulation to model entities with intent. A suite of reasonably well-developed architectures exists for modeling sensing, knowledge and behavior. In most agent-based simulations entities control their own actions. Encapsulation of entity knowledge and actions about the world is typical.

Agent-based simulation is the only tool for modeling many kinds of interactions involving autonomous robots. Giving individual entities behavior rules results in individual actions. By making these rules dependant on the state of other entities, or some group goal, an overall swarm performance become evident, or emerges.

Although there exists a well-developed architecture and theory for using agent-based simulation, there is no tool to answer questions about a system involving swarms of robots.

B. AIM

The aim of this research reported in this thesis is to develop, implement, and test a model for investigating the behavior of swarms of robots.

C. APPROACH

A model was developed and implemented as the Multi-Agent Robot Swarm Simulation (MARSS). MARSS was designed to have the following characteristics:

- Have a broad application to a range of different entities, environments, and scenarios.

- It should be quick and easy to implement a new scenario.
- Data collection and experimentation should be automated.
- The implementation should be deployable.

MARSS was then tested using a Micro Air Vehicle (MAV) search scenario. The objective for the MAV scenario is to gain insight in to the behavioral factors controlling individuals that are important to the success of a swarm. Analysis of the MAV scenario provided useful information regarding the important behavioral factors that contribute to group performance. The research reported here involves modeling a swarm of MAV's to optimize group performance.

D. RESEARCH QUESTIONS

Of general interest are the techniques used to model and understand robot swarms. The research question is "What techniques can be used to effectively model robot swarms?". The general suitability of the modeling methodology implemented in MARSS will be addressed.

The research question addressed for the MAV scenario is "How should individual agents behave to produce a desired swarm behavior?" This question is very difficult to answer in general terms for the range of scenarios it encompasses. Insight can be provided into the broad question by quantitatively answering the following for a number of specific scenarios:

- How is the performance of the swarm dependent on the level of communication?
- What is the difference in swarm performance between distributed and centralized swarm control?

E. SCOPE

1. General

Much work with agent-based simulation has involved building agent models that accurately reflects some real agent. The aim of this work is different. The aim is to find

out what aspects of an individual agent's behavior are important to achieving a group goal. Instead of trying to model behavior, robots behavioral capabilities are assumed. The aim is to find how a swarm of such robots should use these capabilities to meet a group goal. By modeling behavioral capabilities we can create behavior rather than modeling it. The subtle difference is that the only measure of how well a group of agents perform is not how accurately the created behavior represents some real behavior, but how well the swarm performs with the created behavior.

Although MARSS was built to represent a broad range of scenarios, it was designed with some specific scenarios in mind. These included a MAV scenario where a swarm of MAV's is searching for a group of tanks in an area, a mobile surface sea mine scenario where a swarm of mobile mines coordinate attacks on a ship, and a missile defense scenario where a swarm of anti-missile interceptors attempt to attack and destroy incoming threats. Although the MAV scenario was the only one of these implemented and analyzed in detail, the modeling methodology implemented in MARSS was designed with all in mind.

2. Model Inclusions

Like all models MARSS is an abstraction of reality. Some aspects of reality are included in the model and others are not. The model has the capability to represent the following information:

- Almost any conceivable entity that has a state and in some way interacts with other entities.
- The transmission and sensing of information from one entity to the other. This includes the sensing of another entities state and the transmission of information deliberately (messages).
- A wide range of complex entity behaviors.
- The grouping of entities into swarms.
- Entity objectives, both for individuals and groups.
- A basic physical based model of movement.

3. Model Exclusions

MARSS does not explicitly include the ability to represent combat process. That is, the ability of one entity to inflict harm, damage, or attrition on another is not modeled. It could conceivably be included as part of the sensing process; however, the part of the model that represents this process is not designed for that purpose.

Terrain or other obstacles to movement and the explicit transmission of messages are not represented. While the model has the capability to be expanded to represent these aspects they are not essential for the range of scenarios that the model was designed for.

F. PREVIOUS WORK

1. Swarming Theory

Swarming is defined in different ways by different disciplines. The word originated in the biological field from the Old English *swearm*, meaning group of bees. Today a swarm to biologists is the collective term given to a group of insects or similar small animals.

Researchers in the Command, Control Communication, and Intelligence (C3I) area use swarming to describe a way of war fighting [Arquilla and Ronfeldt 2000]. For many years military theorists have made attempts to characterize the way wars are fought. They describe the tactics and doctrine of past engagements in order to gain some insight about how military forces should best operate in the future. Recently swarming has been described as one of four fundamental forms of engagement. The others are the chaotic melee, brute-force massing, and maneuver.

The chaotic melee was the first form of warfare to emerge. It is a primeval state of war with no discernible organization. Ancient clashes between unorganized forces and aerial dogfights in both world wars are examples. As communication improved, massing enabled centralized commanders to direct large bodies of troops. Massing is seen on the linear battlefields of many wars such as the trench warfare in Europe during WWI. Maneuver warfare is that which most modern forces attempt to employ today. It involves

engaging in conflict only when the conditions are favorable. Forces are maneuvered in such a way that they are massed at a decisive point. Operation Desert Storm is a modern example of maneuver warfare.

Swarming as a way of war fighting has been characterized as "*a seemingly amorphous, but deliberately structured, coordinated, strategic way to strike from all directions, by means of a sustainable pulsing of force and/or fire*". [Arquilla and Ronfeldt 2000] Swarming involves distributing autonomous, or semi-autonomous forces about the battlefield that come together, either in force or by fire, to strike at targets before dissolving and redistributing themselves. Many examples of swarming can be found throughout history and in nature. With the recent advent of network centric warfare has it emerged as a contender to maneuver warfare for modern military forces.

Some researchers that use multi-agent systems describe yet another type of swarming. The word swarm is used to describe a collection of agents that are homogeneous in physical characteristics and behavioral properties.

For the purposes of this thesis the word swarm encompasses many of these ideas. It will be used to describe a group of military robotic vehicles that are operating in the same battle space and have a common mission. The swarm may or may not be homogeneous in behavioral characteristics; however physical characteristics will at least be similar, if not the same. Individual robots may move in a pack or may be geographically dispersed. In general members of a swarm will have some form of direct or indirect communication with each other. The key to a group of agents being a swarm is a common goal that the group performance can be measured against.

2. Multi-Agent Systems

Multi-agent systems abound in many aspects of modern computing [FERB99]. The use of multi-agent systems for modeling and simulation has become common. Some of the more commonly cited examples include the following:

- The El Farol Problem – Using agents to model a bounded decision making process [Arthur 1994]

- BOIDS – A distributed behavioral model of flocks of birds [Reynolds 1987].
- ISAAC – Irreducible Semi-Autonomous Adaptive Combat model [Illachinski 1997].

Recent work from the Modeling Virtual Environments and Simulation (MOVES) Institute at the Naval Postgraduate School has focused on creating agent architectures that enable complex behaviors to emerge [Hiles et al 2001]. This work has led to techniques for agent construction including a social and organization relationship management engine, a composite agent architecture, an agent goal apparatus, a structure for capturing and applying procedural knowledge (tickets), and the ability to bring these technologies to bear at the right time and in the proper context using connectors. The work described in this report borrows much of the terminology and agent architecture from that conducted in the MOVES Institute. In particular the concept of an inner environment is used here. This is a construct that contains an agent's knowledge of itself and the world. The use of tickets and connectors to create behavior is adapted for work with MARSS.

Project Albert is an initiative of the Marine Corps Combat Development Command. It is an effort to provide quantitative answers to decisions methods using the "new sciences" [Horne and Leonardi 2001]. Part of the work involves the use of agent-based simulation to provide insight into particular problems. This focus on using agent-based simulation for detailed analysis by Project Albert is rare amongst the range of agent-based simulations developed over the past two decades.

3. Operations Research Techniques

Many of the techniques developed for use by operations research analysts have been incorporated in the MARSS modeling methodology. Discrete event simulation is used to form the basis for the program flow of MARSS. The discrete event simulation methodology helps to remove some of the anomalies associated with time step simulation, and to improve program execution times. Simkit is a discrete event modeling Application Programming Interface (API) developed to assist in the creation of discrete event simulations [Buss 2001]. By using aspects of the Simkit model, techniques created in its development were inherited by MARSS.

Established principles from design of experiments provide robust methods for the treatment of the simulation results [Box, Hunter and Hunter 1978]. More recently work has continued on experimental designs for simulations with many input factors. Evidence of the application of these techniques to agent-based simulation is limited. Project Albert is however undertaking some work in this area.

Search Theory has developed analytical models for conducting optimal search for targets, particularly in the military arena [Wagner, Mylander and Sanders 1999]. Of particular interest to scenarios developed for MARSS is the theory underlying exhaustive and exponential search for moving targets. Related to this is the development of stochastic models for search scenarios.

4. Relation To This Work

The research reported in this thesis is not an extension of any one particular area. Rather it draws tools, techniques, technologies and ideas from many disciplines together to create a modeling process for investigating swarms of military robots. Swarming theory provides the concepts and ideas that form the basis. Multi-agent and discrete event simulation provide the tools and techniques for modeling entities, behavior and interactions. Search theory provides the analytic underpinning for the processes that are investigated. Design of experiments provides the experimental procedures that allow the model to provide useful quantitative information. Finally, statistical data mining tools such as regression trees are used to analyze the simulation output. These techniques are bought together to create MARSS.

It is appropriate to distinguish between the approach taken by many researchers using agent-based systems and the approach this thesis reports. For a variety of reasons many researchers, including some of those mentioned above, attempt to build models that mimic an entity or group behavior. This may be to understand the factors that influence that behavior, to replicate it in some interactive simulation, or to study entity behavior in a different context. The task of creating such models is particularly difficult and important for a range of reasons. The work reported in this thesis does not attempt to take on this task. Rather the measure of the fitness of a created individual behavior is solely

dependant on how it performs in a group context. There is no preconceived notion of what that behavior should look like; only its performance matters.

G. DEFINITIONS

1. MARSS

MARSS is the term given to the Multi-Agent Swarm Simulation System. This tool is a Java implementation of the modeling methodology described in this report. It was developed in conjunction with the methodology, however should not be seen as the only, or best, way to implement the model.

MARSS contains state, sensing, and behavioral model building tools that allow a wide range of complex entities and interactions to be represented. It is a model-building tool that draws theory and ideas from agent-based simulation, discrete event simulation, traditional operations research, search theory, swarm theory, and experimental design.

2. Agent

An agent is a general term given to an entity with intent. For this model agents are mobile entities that sense and react to their environment. The term “agent” is used to describe the decision-making entity. In this sense it would be possible for a swarm of robotic vehicles to be controlled by a single agent (a central controller). This research will focus on the situation where each robotic vehicle acts as an individual agent and makes its own decisions, albeit within mission parameters. Hence the term ”multi-agent swarm” arises

3. Swarm

A swarm is defined as a group of agents with a common goal. This definition draws ideas of swarming from those presented earlier in this chapter. While this definition is used throughout this thesis swarming should be thought of as a concept rather than something that is precisely defined.

4. MAV

A Micro Air Vehicle (MAV) is an ultra small aircraft, no more than 15 cm in any dimension. Such vehicles are being developed by the Defense Advanced Research Projects Agency (DARPA) to perform useful military missions. The figure below is a picture of a prototype MAV built by AeroEnvironment for DARPA. [Grasmeyer and Keennen, 2000]. This particular vehicle flew for 30 min transmitting a real time color video image the entire time.

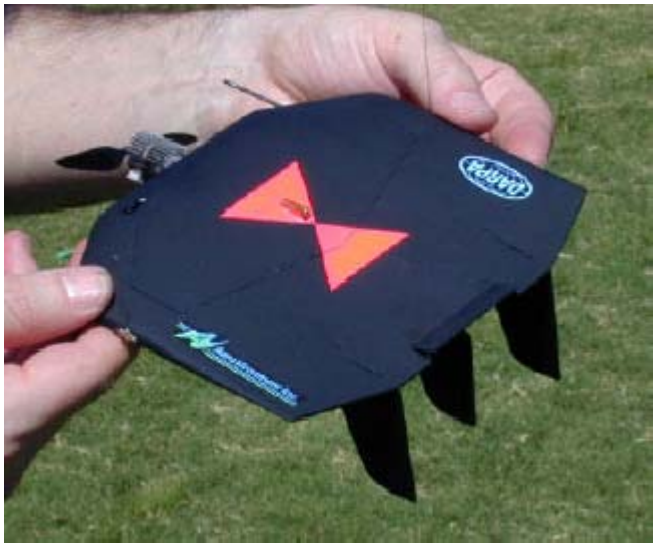

Figure 1. AeroEnvironments "Black Widow" Micro Air Vehicle

H. THESIS ORGANIZATION

1. General

The chapters of the thesis are designed so they can mostly be read independently. The remainder of this section describes briefly what is reported in each chapter.

2. Modeling Mobile Robots In A Swarm

A general description of the agent model is presented. The relationship between swarming theory as it applies to military operations and the behavior of a group of robots with a common goal is explored. More detail on the theory of Multi-Agent systems is then presented as it relates to robots.

3. The Sensing Model

Development of MARSS involved creating a complex sensing sub-model. This chapter describes the sensing sub-model in detail including the emission process. The reasons for the complex sensing process and comment on its limitations are reported.

4. Creating Robot Behavior

This chapter describes how the behavioral process is modeled. It emphasizes that the aim of this model is not to mimic a behavior but to create one. A description is given of how an agent's sensed view of the world is mapped to things that it can control.

5. MARSS Design And Features

The program design and programming principles of MARSS are presented and a description of the features available to the user is given. Brief comments on how MARSS can be used to help design robot behaviors are made; they are followed by suggestions for other possible uses. Possible future enhancements are explored.

6. A MAV Scenario

This chapter reports how MARSS was used to explore the important factors governing the behavior of a swarm of Micro Air Vehicles searching for tanks. A detailed description of the scenario and how it is implemented in MARSS is described.

7. MAV Scenario Results and Analysis

The experimental design, data collection, and analysis are reported. A comparison of the experimental results to theoretical performance is made for both exhaustive and random search. The development of more complex MAV behaviors is described and the results with such experiments analyzed.

8. Appendices

Two appendices are included. The first contains a pictorial description of the MARSS tool and should be read in conjunction with the MARSS design chapter. The second contains a detailed description of the search algorithm used to ensure uniform random coverage of a rectangular search area.

THIS PAGE INTENTIONALLY LEFT BLANK

II. MODELING MOBILE ROBOTS IN A SWARM

A. INTRODUCTION

The first question faced in developing a model to represent mobile robots in a swarm is "What do we want the model to tell us?" This is relatively easy for a specific scenario; however MARSS should be applicable to many robot swarm scenarios. The answer for MARSS is that it should help provide insight into the important factors that control robot behavior. The next question is what kind of model should be built.

A range of methods available to Operations Research analysts was considered as candidates to help understand robot behaviors. For a variety of reasons simulation appeared to hold the most promise for achieving the stated aim. Agent-based simulation methodologies and ideas were most appropriate to answer the questions posed. An agent-based architecture was then developed.

The remainder of this chapter is devoted to describing a number of aspects of the model that has been built. Central to this is what kind of swarming is being modeled. The remaining sections then describe how individual agents are modeled in this agent-based simulation.

B. SWARM THEORY APPLIED TO ROBOTS

The definition of swarming by researchers in the C3I area is closest to that used here to describe a robot swarm. That definition is mostly concerned with swarming as a method of engaging in combat. With regard to robot swarming the MARSS model is designed for analyzing other military operations, as surveillance and reconnaissance by a swarm. In these operations we are interested in a swarm of robots working together to achieve a common goal. The swarming is not necessarily geographic. In many cases the swarm behavior is a function of cooperation and distributed coordinated effort.

Recent advances in robotics, artificial intelligence, sensing, and communications promise to deliver capable low cost robotic weapon systems in the near future. Most robots will use swarm tactics in all spheres of the future battlespace. Swarms of crawling robots may be used to clear landing zones [WEBER 1995]. Robotic high-speed mobile

sea mines could effectively prevent areas of the sea from being used, or could even swarm to attack large vessels. Swarms of small aircraft with small warheads may be used to provide point missile defense, massing together to form a cloud in front of an incoming threat. The possibilities are only limited by imagination. Science fiction will gradually become a reality as technologies enable these ideas. In many cases, especially for air and sea surface craft, the technologies are already in place and relatively small investments will result in very capable weapon systems.

C. AGENT STATE

The description of an agent at any point in time is its state. The crux of what any simulation does is to vary the state over some domain using a function. In general these functions are too complex to be understood using analytical techniques. Although agent-based and other simulations can vary the state of an agent over domains other than time, for this model the properties of an agent that represent its state are only varied with time.

Not all properties of an agent need to be represented explicitly by its state. Indeed if we were to model every aspect of a robot we would need to capture the position and energy of every subatomic particle that made up the robots matter. Even with the unlikely assumption that we could somehow capture that information in the foreseeable future, to represent it in a way that we could use for simulation is well beyond the capabilities of any current or envisaged computing machine. To overcome this limitation the simulation designer must simplify the representation of the agent. In general the designer should choose properties to represent an agent that are believed to have some impact on the outcome being studied. Simple properties such as location, size, velocity, health, etc. are used to describe the state of agents in the MARSS model.

The concept of implied state allows us to simplify the model even further. Not all properties that are used by the simulation need be represented if one or more of these properties are functions of some other state variables. Such variables make up the implied state of an agent. In a similar vein some properties of an agent may be combined to form some artificial property. The health of a robot is an example. This property may be used

to represent the remaining time a robot can be used (battery life) or to represent some other property such as the reduction in physical capability due to battle damage.

In MARSS every aspect of an agent's state is modeled by either a vector or scalar property. Three-dimensional vectors are used to model properties such location, orientation, velocity, and acceleration. Scalars are used to model things like size, health, and age. Linked to each property is a name. To determine the state of an agent at any point in time there is a mapping from the property name to the value of that property.

Not all of these properties will vary over time. Some are constant for the duration of the simulations; others are modeled as properties so the agent can represent them in its inner environment. Those properties that change over time are distinguished in the MARSS as "changeable" properties. In discrete event simulation these changeable properties are known as state variables. Properties that are not changeable would be known as parameters.

Changeable properties may be unbounded or may be given bounds. For example there may be a limit on an agent's location property to ensure it does not go below ground level, or on a velocity property to ensure it does not exceed a particular value. The types of bounds that can be given to a scalar property are an upper and lower limit. For vector properties there may be an upper and lower limit on the vectors magnitude (for a velocity for example), or upper and lower limits on each of the three dimensions (for a location for example).

In MARSS there may be many forces working to affect a changeable property. Property change suggestions are made by the behavioral sub-model about the new value of the property. Each suggestion has a weight associated with it. The properties manage these suggested changes by simply performing a weighted addition of them and applying this suggestion to change the property. Before the property is finally changed the summed suggestion is restricted by the set bounds. The restricted suggestion is then applied, and the agent now has a new state.

D. THE INNER ENVIRONMENT

In the real world humans and other decision-making animals perceive the world around them and act on this representation of the world. Part of this perception is a perception of their own state. Together this perception of the external world and their own state is known as the inner environment. The outer environment is the world external to decision-making animal that it is making perceptions about. This model of the perception process is used in MARSS.

The main reason for distinguishing the information in the world between inner and outer environments is to encapsulate that information that an agent's behavior may be based. Simulations without this or a related structure may have a hard time keeping track of what an agent "knows" and can therefore base behavioral decisions on.

The inner environment is comprised of two distinct parts, the perception of self, and the perception of the outer environment. These parts are distinct due to the origin of the information. In the MARSS model there is a separation between these two parts of the inner environment as they are stored slightly differently in the agent model. However, an agent's behavioral mechanism makes no distinction.

An agent's perception of its own state is the first part of the inner environment. In some models it may be assumed that this information is available for the agent's decision-making process, however this is not always the case. In some cases an agent may know nothing about its own state and in other cases the information may be filtered in some way. For example, if an agent can somehow perceive its own location its representation of that property may be filtered or perturbed in some way to induce an effect of uncertainty in a location sensing system being modeled. It is possible that an agent's behavior be based on nothing more than information about the external environment it receives. In the MARSS model presented here it is assumed that agents have perfect information on those aspects of their state that are used for the decision making process.

The other part of the inner environment is the information that has been gathered by an agent's sensors from the outer environment. This information comes in two forms, information about other agents, and information about non-agent entities in the

environment such as terrain. The latter is not explicitly represented in the MARSS model, however the concept of stationary entities with no behavior can be used to represent objects that other agents can sense. The lack of an explicit representation of non-entities in the environment was a deliberate design decision due to the complexity of such a model.

The information that one agent can get about another agent consists only of the sensed agent's state. The creation of this part of the inner environment involves the transfer of information about one agent's state to another. One agent gets to know something about the properties of another. Similar to the information about an agent's own state, this information may be inaccurate in some way depending on the process being modeled. For the MARSS model the sensing process always transfers information about a particular property with complete accuracy. There is an entire chapter of this work devoted to the operation of the sensing model and it will not be discussed further in this section.

A depiction of the inner environment is shown in the following figure. Here we can see the two sections of the inner environment that make up an agent's inner environment.

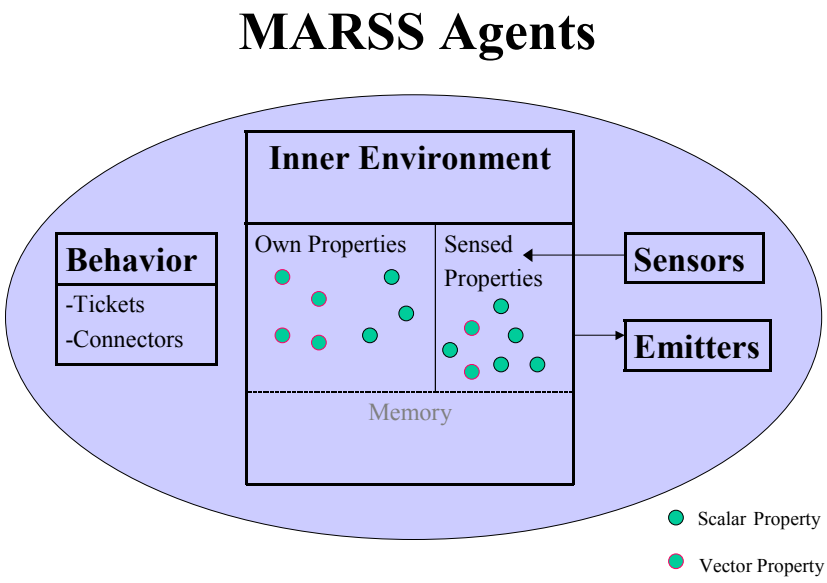

Figure 2. Graphical Depiction Of The Inner Environment

Note also in the above figure the depiction of memory. As both an agent's state and its sensed representation of the outer world change over time the inner environment also changes. In some agent systems it is appropriate to retain a history of information contained in the inner environment. Such a history can be described as an agent's memory. The other part of the memory that may be stored is some information about how an agent has behaved. A memory has not been implemented for any of the behaviors that have been constructed in MARSS, however it would be relatively easy to implement an agent with a memory.

The above figure also has areas that represent an agent's sensing system and behavior. The operation of the sensors and emitters depicted is covered in the next chapter, and behavior in the chapter following that.

III. THE SENSING MODEL

A. GENERAL

This chapter describes how the sensing process that creates an agent's perception of the outer environment operates in the MARSS model. The first part of the inner environment, that part formed from an agent's own properties, is covered in the previous chapter. The task of the sensing process is to take information about the outer environment, process it in a logical manner, and pass it to an agent for use in its behavioral system. In determining how to model this sensing process we must first decide what the information in the outer environment actually is, and where it resides.

B. CONSTRUCTING THE INNER ENVIRONMENT

The information external to an agent can be characterized in two ways; that information about other agents' state, and that information about non-agent state parameters of the model. An example of the latter is terrain and obstacles, which are not represented in MARSS. Information about other agents' state includes the current value of the other agents' properties such as its location or velocity.

The entire state of the simulation is described as the sum of states of all of the entities. The information external to an individual agent that has to be processed by the sensing model is the state of all of the other agents in the model. It would be possible to create a simulation where each agent could access every bit of information about the state of every other agent, however there are very few situations where this would accurately model a physical process. In the MARSS modeling process the only way one agent can gain information another agent is via the sensing process.

The task of the sensing model is to process information about an agent's state and pass this to other agents. To create a model for this process there are a number of other constraints that must be observed. One of the driving factors was to create a model where the information contained in both parts of the inner environment is indistinguishable to the agent. The information in the inner environment from the sensing process is stored as either vector or scalar values with a string representing its name; just like an agent's own

properties. Another factor considered in creating the MARSS model is that in many cases only a limited subset of information sensed by an agent will actually be used to control its behavior.

The sensing process is modeling sensors onboard robots and any other entities being represented. In creating a model to represent this process ideas were drawn from the physical world. In the physical world entities are not directly sensed by other entities, rather some signal is emitted, transferred through space, and received and processed by other entities. A description of this can be divided into the emission and sensing processes. The following sections discuss these in more detail.

C. THE EMISSION PROCESS

1. Bands

In the physical world energy is constantly being emitted by all entities. In most cases these emissions occur regardless of whether there is something capable of sensing and processing the emitted signal or not. These signals may electromagnetic, acoustic or chemical. We can characterize the signals by the kinds of sensors that can detect them; radio, visual and radar are examples. These characterizations can be thought of as channels through which information is passed. These channels bands are referred to as bands. This term is borrowed from the electromagnetic spectrum. A MARSS band may be used to model any information channel however.

One of the characteristics of signals in the real world is that at any point in space they have a strength (or intensity). The signal strength is generally greatest at the origin of the signal, and then degrades with distance. This attenuation function is specific to each band. Some signals attenuate linearly, or exponentially for example. Others may have more complex attenuation functions such as that attributed to the transmission of acoustic signals underwater. By specifying a band for a signal its degradation function is assumed. In MARSS there is no built in restriction on how many bands may be used for a particular scenario. In practice it is wise to limit this number to only those bands that are required for computational efficiency.

It is possible to have a band where a signal is not attenuated by distance. Such a process can be used in MARSS to represent a signaling process where there is a requirement to transmit information globally.

There are two main reasons for representing transmitted information in bands. The first is to model different attenuation functions. By doing so the strength of range of signals that depend on distance can be imitated. The second case for bands is to simplify the construction of sensors. Sensors that operate in a particular band need only consider signals in their particular band.

2. Signal Types

It has been previously stated that the type of information being transmitted are the properties of agents. A particular agent will generally not transmit a signal that contains all of its properties. Rather, signals will include a subset of its properties. An entity may have more than one kind of signal that is emitted. These groupings of properties into signals are known as signal types. An entity may have many signal types modeled. A signal type is a mapping from a signal type name to property names. Every entity has a signal type called “entity” that contains only one property, the entities location. Signal types can be thought of as how a particular entity can be observed as by other entities.

A concrete example of the signal type concept can be given for a tank. Suppose a tank is modeled with properties such as location, velocity, size, make, team, unique identification and armament. Examples of signal types include “entity”- (location), “tank” - (location, make), “teamTank” - (location, make, team) and “teamTank(i)” - (location, make, team, unique identifier). In each of these examples the signal type name is followed by the properties that are mapped to that signal type. It is not necessary that signal types be ordered in any way where properties are added for greater resolution of a particular entity. For example another valid signal type for a tank may be “weaponSystem” – (location, unique identifier, armament). Note that in MARSS all of these properties are either vector or scalars. The modeling of a particular entity in MARSS requires the analyst to define the signal type/properties mapping for each entity.

Signal types allow for a measure of control over what information is being transmitted in the environment. A signal type architecture may be created to ensure that as the strength of the received signal increased the sensor gained more information about an entity being sensed. An example of this is the gradual increase in visual information that a human receives as we become closer to an object being sensed.

3. Emitters and Emissions

To facilitate the transmission of signals, entities have emitters. An entity may have zero or more emitters. Each emitter is designed to emit a particular kind of signal, an emission. Emissions are the carriers of information in the model. Each emission consists of four pieces of data: its originating entity, what band it is transmitted in, its initial strength, and its originating location. The role of entity's emitter is to ensure that as the entity's state changes its emissions also change. Signal strength units in the model are arbitrary. Any units can be used for any band as long as the model is consistent throughout. In fact, in many cases the units will just be arbitrary rather than some common physical measurement. What is important is the relative values of signal strengths.

Emissions are created and destroyed only when required. This is not every time the state changes, rather the emissions parameters are updated in response to a change in state. The originating entity and band of an emission will never change. The initial strength of an emission may change if the agent's state somehow controls it. These state changes may be under the agent's control or may be a result of some other process. The main part of an emission that changes will be the emission location.

An agent may have many emitters and may therefore be responsible for signals in many different bands at any point in time. Information about an agent's signal type is not contained in an emission. Rather, a link to the originator is maintained by the emission. With many agents there may be a large number of emissions being modeled at any point in time. A description of how these are managed is provided in the next section.

4. The Ether

The term ether is borrowed from antique physics. Also known as the aether, the ether was supposedly a medium pervading all space that supported the transmission of electromagnetic waves. In MARSS the term ether is used to refer to the model construct that holds and manages emissions.

The ether manages these emissions in a relatively simple way. Each new emission is placed into a bin according to its band. The responsibility for updating an emission rests with its creating emitter. All the ether does is to provide a convenient mechanism to get all of the emissions in a particular band.

A more sophisticated approach to emission management may be to further bin the emissions according to some area of interest management rule. Emissions originating in some geographic area would be grouped together. The dimensions of the geographic volumes could be based on some function of the most sensitive sensor, the most energetic emission, and the band attenuation function. This area of interest management is not implemented in MARSS, but would be relatively simple to achieve.

5. Modeling Reflected Signals

In the physical world the energy that contains signaled information does not always originate at the entity, but is often reflected. The reflection of light by a tank, reflection of a radar signal by an aircraft, or reflection of sound by a submarine are all examples. The emission model presented here can easily handle the case where these reflected signals are either constant or some function of the reflecting entities state. To do so we imagine that the entity is the originator of the emission and ignore the process of the energy getting to it. With a little more difficulty ‘ping’ type reflections can also be modeled. In this case the emitter must be set to send an emission at appropriate times.

Where we wish to model a process where an entity sends a signal, it is reflected by one entity, and then received by the originator, or even some other entity, the MARSS model must be bent a little. The originator can model sending of the signal as an emission. The entity to reflect it must sense that signal (using the sensing process to be

described in the next section), and then emit a reflected emission. This reflected emission could then be sensed. For many purposes this convoluted process is not required and reflected emissions can just be modeled as normal emissions.

6. Signals For Communication

The emission process is the only means built in MARSS to allow communication between agents. This is a deliberate design decision so that all information presented to an agent that may be used for decision-making is in the same format. In a way this idea is close to the real world, as all communication transmissions involve some form of energy transfer.

For communication to be effective an agent should be modeled in a particular way. First, properties that are to be communicated, or message properties, need to be modeled. These may be some properties an agent has already, such as location or current waypoint. A particular message property may be created, such as a scalar that represents the agent's intentions. Next, the agent needs a property that it uses as a switch for communication. This may be extended to a property that controls the transmitted signal strength. The agent also needs a signal type for each type of message that it wishes to send. Rather than the signal type representing some image of the entity, it represents a signal that the agent will send. Finally, an emitter must be added to send the signal and a band conceived in which the signals are sent.

Agents that receive these communicated signals need only have a sensor that is capable of receiving information in the particular band. The sensing system will automatically process the signal for use by the agent in the decision making process. The important thing here is that a communicated message is modeled as just another piece of sensed information by the agent.

7. Signals For Attrition

MARSS is not designed to represent large scale and complex attrition between entities. There may still be a need however to model attrition to reflect some real process.

One way to do this in MARSS is to have the shooting process modeled as an emission, much like the communication process previously described. The shooting agent needs appropriate properties, signal types and emitters. The model needs an appropriate band, and agents subject to attrition need attrition 'sensors' and a mechanism to deal with the effects of the weapon systems.

In a sense this model is not all that far from reality. The operation of any weapon in the modern battle space involves the transfer of energy from one entity to another. By modeling this transfer as an emission and sensing process a variety of other effects built into more complex combat models can be represented. This includes taking into account probability of kill and probability of hit parameters used in many combat models

D. THE OPERATION OF SENSORS

1. Sensors

With an ether full of emissions the final task in the sensing process is to take information the emissions link to, and complete the construction of the inner environment. To accomplish this agents have zero or more sensors. Each sensor has a number of parameters that control what signals it receives from the ether. A sensor will typically process a number of emissions, accepting some as received, and rejecting others. There is no notion of partial receipt of a signal; it is all or nothing.

The first parameter that controls what emissions a sensor processes is the band. Each sensor is allocated only one band in which it can receive information. The remaining parameters are the sensors sensitivity, geometric constraints, and signal type capability. These are covered in turn.

2. Sensitivity

Each sensor has two parameters that control the sensitivity associated with it. One or more of the sensing agent's properties may dynamically control these parameters. They are the current, or 'entity', sensitivity (this is described more fully in signal type

capability), and the maximum sensitivity. A simple interpretation of the sensitivity is a level against which the strength of received signals is measured. If the received signal strength is greater than the sensitivity, then the signal is received. A higher sensitivity value will mean that more signal strength is required for the signal to be received. This is a little counter intuitive as a higher sensitivity value results in a less sensitive sensor, and a lower sensitivity value results in a more sensitive sensor.

3. Geometric Constraints

In reality some sensors are directed. Real sensors may only be able to receive signals from a particular direction, or from a specific minimum or maximum distance. In MARSS each sensor has a function that checks the location of the incoming signal against the orientation of the sensor and returns a Boolean value that controls if that signal can be measured. This geometric constraint check must be defined for each sensor.

4. Signal Type Capability

When a sensor is constructed the signal types that it has the capability to detect are defined. By default each sensor has the capability to detect an ‘entity’ signal type. This is the default signal type associated with each entity. Associated with each signal type capability is a scalar value termed a ‘sense threshold factor’. This factor is used to modify the base, or ‘entity’, sensitivity of a sensor in a multiplicative way. This defines the actual sensitivity for the particular signal type the threshold factor is associated with. The sense threshold factor associated with the ‘entity’ signal type is equal to one. Other signal types typically will have a value greater than one. A sensor with more than one signal type capability will therefore have a sensitivity level for each signal type it can sense. These adjusted sensitivities will all be some multiple of the ‘entity’ sensitivity.

5. Putting It All Together

To summarize the above process a sensor first checks the ether and gets all emissions from the band that it senses. It then looks at each emission, rejecting the emission if the sensors owner was the originator of the emissions. Next, the sensor does a

geometry check to determine if it is possible that it sensed the emission, rejecting it if not possible. The received emissions original strength is then attenuated according to the relevant band attenuation function to get received signal strength.

The sensor must then determine if it has the capability to receive a signal from the emission, based on the emission's originator signal types, the sensor's signal type capabilities, and the associated sensitivity. The first check here is to iterate over the sensor's sensitivities for each signal type until a sensitivity value is below or equal to the received signal strength. When this is found the emission's originator is queried to determine if it emits a signal type the same as the relevant signal type capability of the sensor. If so the sensor senses the signal type, the appropriate information is transferred to the entities inner environment, and the next emission is considered. If not the sensor proceeds to the next sensitivity value and checks again. Each emission can only be received by a sensor as one signal type.

6. Managing The Sensed Information

Each agent has a manager to take care of and operate all of its sensors. This manager also manages the sensed information. The manager contains a bin for each signal type capability that its sensors have. Each sensed signal is a mapping of property names to values. These values represent some portion of the state of the entity being sensed. Note that this sensing process does not give a sensing entity direct access to the state of another entity, as this would violate the general policy of information encapsulation. Instead state information is copied and passed into the sensing entities inner environment. With the completion of this process the agent now has a complete inner environment that its behavioral model can use for decision-making.

E. A SIMPLE SENSING EXAMPLE FOR SEARCH

The simplest of all sensors used in search scenarios is the cookie cutter sensor. A target is sensed if it is less than some range r, and is not if it is further than that range. One example of how to construct such a relationship between a target and a sensor in MARSS is given here. There are other ways to implement a cookie cutter sensor. In

effect this relationship will allow a sensor to know the location of the target if it is at a range less than r.

First one generic band is used. The attenuation function is linear. The signal strength received (S_r) is equal to the signal strength emitted (S_e) minus a constant (k) times distance (d) ($S_r = S_e - kd$, $d > 0$). Targets are left with the default signal type ("entity" – maps to location) and are given a constant signal strength (S_e) emitter. Searchers are given a sensor with a default signal type capability (entity) and a constant sensitivity (S_s). The range at which detection will occur is then given by the relationship, $r = (S_e - S_s)/k$.

Although this implementation of a cookie cutter sensor model in MARSS may seem complex, the modeling process is capable of modeling much more involved relationships between targets and searchers.

F. FLEXIBILITY AND LIMITATIONS OF THE SENSING MODEL

The sensing modeling process described here has the capability to be easily expanded to take into account many process not yet described. For example, by designing emitters and sensors so that emitted signal strength, and a sensor's sensitivity, are some random process, uncertainty can been introduced into the sensing process. This uncertainty is slightly more sophisticated than a random perturbation of the attenuation function (which is also easy to implement), since there is a relationship between two otherwise unconnected sensors and a single emitter. If the emitter is emitting with a large signal strength (perhaps corresponding to being at the top of a hill), then the signal is more likely to be received by both sensors. Likewise multiple emitters and a single sensor can create similar relationships.

The sensing model could also be made with a more sophisticated interpretation of geometries. The current geometry check for a sensor just checks to see if the relative orientation of the sensing entity is such that sensing is possible. A more sophisticated approach could vary the sensitivity continuously based on the orientation, more closely modeling reality. On the emission side the initial signal strength could be modified due to

the originators' orientation. This would require a simple enhancement to the sensing model.

The policy of information encapsulation is strictly adhered to in the MARSS model of sensing. Agents do not have direct access to information about other agents. Rather they have to go through the sensing process to get to it, and then they only get a copy. This is an important design consideration to help prevent errors in implementations of the model. It also allows imperfections, or noise, to be introduced in the transmission of information.

G. SUMMARY

The MARSS sensing model takes information about the outer environment and transfers it to an agent's inner environment. It does this by modeling the emission and receipt of signals containing information about other agents' properties. The sensing model is flexible enough to allow a wide range of complex agent interaction processes to be represented, yet simple enough to model sensing process used in other models. With a representation of the world in their inner environment an agent must make decisions on what to do. The behavioral model discussed in the next chapter handles this process.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. CREATING ROBOT BEHAVIOR

A. CREATING VS. MODELING BEHAVIOR

Many agent-based simulation models attempt to create a behavior of a group of agents that in some way mimics the behavior of real world entities. Examples of such simulations include models that mimic the behavior of a flock of birds or a swarm of ants. Simulations have been built that attempt to model some aspect of individual or collective human behavior. In some cases an attempt was made to model an emergent group behavior by creating individual behaviors.

The philosophy behind MARSS is very different. Rather than attempting to mimic a given behavior we are trying to create one. We have no preconceived notion of what the emergent behavior will look like; however we have a way to compare the performance of different emergent behaviors. The model is used to represent a behavioral process rather than a particular behavior. This subtle difference is very important as it markedly changes the way a simulation is designed and used.

B. THE BEHAVIORAL TASK

An agent behavioral model takes input from the inner environment that is formed from the sensing model and from the agent's own state. An output generated as a function of the inner environment and is used to affect some aspect of the agent's own state. Many behaviors that can be conceived may be generalized to fit into the MARSS model. The following sections cover a particular way of mapping information in an agent's inner environment to changes in an agent's state.

C. BEHAVIOR MEASURES OF EFFECTIVENESS

To compare the behavior of agents we must have some way of measuring it. This measurement is specific to each agent, and each task it is allocated to undertake; however, there are some consistencies between measures. Each measure of an agent's performance will have a value associated with it. For MARSS a measure also has an

array of information that was used to measure that agent's performance. The structure of this information is unique to each type of agent measure designed. The information contained in a measure is used for the measurement reporting process.

A distinction must be made between two different kinds of ways to measure an agent's performance. An agent's performance may be measured either internally or externally. An agent's internal measure its limited to the information in its inner environment over time. Measuring an agent's performance externally allows a much richer pool of information to be used; however such a measure should not be used by an agent's own behavioral system. To do so would violate the information encapsulation principles that apply to the agent system being modeled. An agent may use an internal measure and compare it to its goals, adjusting its behavior. External measures are only of interest to the analyst, however they may be used by the system as part of some behavior optimization algorithm. A distinction between internal and external measures of performance can be made at every level of performance measurement.

In MARSS we are concerned with measuring the performance of a swarm of agents. While not required it seems logical to measure the performance of a swarm based on the performance of the individual agents. In MARSS the performance of a swarm is an external measure that is the sum of the internal measures each agent makes of their own performance.

An agent may use internal measures of performance to dynamically change its behavior. Measures may be made of how well an agent is achieving a particular goal. Such measures are used to create adapting behaviors. Although no such behaviors were created for the work reported here, the building blocks are there to do so.

D. DEGREES OF FREEDOM

The part of an agent's state that the behavioral process can affect is called its degrees of freedom. In addition to characterizing an agent's properties as vector or scalar they are characterized as changeable or not changeable. Some of the changeable properties may be able to be affected by an agent's behavioral model. These properties are the agent's degrees of freedom and represent what the behavioral process can affect.

The degrees of freedom are the only properties that an agent's behavior may change. The model may change other properties. An example of this is the relationship between the acceleration, velocity and location properties that are implemented for mobile entities in the MARSS model. All of these properties are changeable, however in most cases the only one that is part of an agent's degrees of freedom is acceleration. The velocity and location are updated by the MARSS model based on the current velocity, the location, and the agent's intended acceleration.

E. BEHAVIOR MAPPING

It is possible to describe a behavior where there is some complex mapping of the inner environment to an agent's degrees of freedom. This quickly results in a behavioral model that is complex and specific to a particular agent. Any slight change in the behavior that we wish to model may require a major change to the code we have developed to represent that. To help alleviate this problem a generic and expandable way of mapping the inner environment to degrees of freedom has been developed. This method is similar to that used in related research [Hiles et al. 2001].

1. Connectors

The connector is the main construct to link some properties from the inner environment to a degree of freedom. A connector is designed so that it has many inputs and only one output (i.e. it affects only one of the agent degrees of freedom). The types of functions that a connector may perform are only limited by imagination. Some generic connecters have been developed for the MARSS model; however these are by no means the limit on what could be produced. For example, a connecter has been developed that takes a two vector inputs from the inner environment, determines their difference, normalizes and scales the result, and applies this as a property change suggestion to a changeable vector property. This connector may be used to cause an agent to accelerate towards a particular object is senses. Many other connectors have been developed.

Note that the output from an individual connector is a weighted suggested change to a particular property. All suggestions for that property are added and the result restricted by the property bounds before being applied.

2. Tickets

Most agent behaviors will have more than one connector. A structure has been developed to hold many connectors. This structure is called a ticket. [Hiles et al. 2001] The tickets developed for MARSS are sequential. Each connector is executed in turn for a particular ticket.

The ticket construct could be used to create much more complex behaviors. Multiple tickets could be applied to different scenarios that an agent faces to create some type of adaptive behavior.

3. Regulators

To control the execution of a group of connectors a model construct called a regulator was developed. A connector may or may not have a regulator. Regulators are grouped together in a regulator group. Regulators within a regulator group are designed to control the execution of connectors to which other regulators within the group belong. The normal way this works is that connectors on an individual ticket belong to a single regulator group.

Regulators have a type, and a list of types that, if activated, will stop the execution of the connector to which it belongs. This allows connectors to be conditionally executed based on the prior activation of regulators by other connectors in the sequential ticket. Generally a regulator will only extend (activate) if the execution of a connector achieved some standard.

The regulator framework allows complex behaviors to be built. For example an agent may try to do A, and if that is successful, skip B and C, however if A was not successful then it will try B, etc. This degree of flexibility in model design is important to be able to capture a wide variety of behaviors.

4. Factors

The discussion of the operation of a connector noted that the vector was scaled by a factor before applying it as a suggested property change. A factor is a value that controls some aspect of the operation of a connector. The level of the factor is just one of the decisions that must be made in constructing a function for a connector. These factors are the key to controlling an agent's behavior. A factor could perform any function in a connector however the most common is to weight the suggested change by a certain amount. Factors may be used as a yardstick to compare some aspect of the agent's inner environment against. For example, the connector's function may have a conditional statement that controls the operation of the connector where the magnitude of its MOE is tested against a factor value.

The levels that factors are set at define how an agent behaves. Once a behavior has been designed, the task is to determine at what levels to set the factors in order to get the best behavior from the agent or, in the case of MARSS, a swarm of agents. Each agent has a behavior factor manager to manage the factors that are used to control behavior. For each simulation run a factors level is held constant.

It is possible to use factors that are not associated with connectors if some other kind of model for a behavior is being developed. In such cases the factors would still control some aspect of the behavior that the agent undertook.

The same factor model may be used to control non-behavior parameters of an agent. A factor may for example, represent some physical capability of an agent, such as sensor sensitivity.

F. SWARM BEHAVIOR

The behavior previously described was for an individual agent. In MARSS we are interested in the behavior of a swarm of agents. Swarm behaviors are not explicitly designed. Rather they emerge from the interactions of individual agents. We can however characterize some aspects of a swarm behavior at the building stage. Swarms can be either homogeneous or heterogeneous. There are different levels of homogeneity. In a

completely homogeneous swarm each agent would have the same properties, emitters, sensors, behavior, and the levels that the factors are set at would be exactly the same. Swarms may be partly heterogeneous if one or more of the above properties are not constant throughout the group. A completely heterogeneous swarm would be a group of agents that have no common similarities other than a common measure of effectiveness.

MARSS scenarios have only been built for mostly homogeneous swarms. The swarms are heterogeneous in the levels that their factors are set at, and are homogeneous in physical capability and behavior construct. The possibilities are limitless for creating swarms at various levels of heterogeneity.

G. COOPERATIVE BEHAVIOR

Agents may cooperate in a deliberate or emergent manner. For deliberate cooperation there is some explicit communication between the agents that controls their behavior to help move towards a group goal. Emergent cooperation is subtler and occurs when the individual behaviors are such that agents share tasks or do not duplicate effort, for example. Although a scenario could be easily implemented in the MARSS model to force deliberate cooperation between agents, work so far has focused on emergent cooperation.

The beauty of emergent cooperation is that there is no requirement for a leader, just a set of cooperative rules that each agent possesses in order to get a job done. This kind of cooperation is often seen in nature [Franks, Sendova-Franks and Anderson 2001]. Scenarios implemented in the MARSS model have considered such basic cooperation between searchers. In this case a set of rules is developed to help reduce duplication of effort, and increase effort of the group in a direction most likely to produce the desired swarm outcome.

Rather than emergent or deliberate cooperation mechanisms being absolute, a spectrum can be envisaged. At one end there is completely emergent cooperation, and at the other completely deliberate. In between, agents may have both cooperative rules and a mechanism to transfer intent between members of the group.

H. SEARCHING FOR BEHAVIORS

1. General

When using the ticket, connector, and factor architecture described in this chapter to model behavior, the task of determining an optimum behavior is reduced to determining the optimal levels that the behavioral factors should be set to. This assumes that the designer of the connectors and tickets is representing enough behavioral flexibility based on the available inputs from the inner environment and the degrees of freedom of the agent. Couched in this context, of determining optimal levels for a range of factors to produce a desired (or optimal) swarm response, the problem is a typical search problem that is faced by analysts. A common approach to solving this problem in agent-based design is to use some genetic algorithm to adjust the levels between simulation runs, with the aim being to evolve to a good solution. In addition to using that technique the work reported here used designed experiments to explore the response surface in a more systematic way. These two techniques are described and compared below.

2. Evolving Behaviors

In order to use search methods involving genetic or evolutionary type algorithms to evolve the behavior of a swarm, some degree of heterogeneity between swarm members is required. The selection process of evolutionary algorithms requires this variation if we are to link good actions to some aspect of the agent. There must be a way of measuring individual performance contribution to the overall group goal. In MARSS the heterogeneity is produced by having a mix of factor levels for a particular factor that every member of the swarm has. The measures of swarm and entity performance that have been previously described are suited to evolving behaviors.

After a simulation run the entities in a swarm are ordered according to their performance (as defined by their individual measure of effectiveness). The top half performing entities are used as a ‘mating pool’. A new swarm is formed with the same number of entities as the old swarm. Two entities are selected from the mating pool at

random (with replacement). For each of the factor levels to be set on the new entity a level is chosen from one of the two selected from the mating pool (again at random). This is applied to the new entity with a possible mutation. Each factor on each entity is set using this method. In theory after many mutations those factor levels that result in good swarm performance are preferred and the swarm performance improves.

The mutation mechanism in the MARSS model is slightly different from a standard genetic algorithm. Factors are allocated a random variate and a mutation probability. When a factor is copied a uniform random number is compared against the mutation probability. If mutation occurs then the level of the factor is set to a new value generated from the random variate. The random variate can be from any distribution for each particular factor.

Searching a behavioral response surface using evolutionary theory can be likened to many concurrent hill-climbing algorithms with random jumps. It can only be used in the MARSS model when swarm behavior is heterogeneous with regard to factor levels. Heterogeneous factor levels would be difficult to analyze using designed experiments since the number of factors that control the response surface would be the number of agents in the swarm, multiplied by the number of factors for each agent.

3. Designed Experiments

Using designed experiments to explore response surfaces has not been a common procedure with agent-based simulations. For the MARSS model it has been the primary response investigation tool so far. A number of experimental design principles have been implemented in the MARSS model. These include the ability to use common random numbers over a block of simulation runs. The ability to conduct full factorial, grid search, random sampling, and Latin hypercube setting of factors has been implemented. Entities may have a factor level setter that manages the setting of factors using one of these methodologies between simulation runs. This allows those factors, and in some cases interactions between the factors, that are not significant to the response to be identified.

A more complete description of some of the designed experiments conducted so far is reported with the results of those experiments.

I. FUTURE WORK WITH BEHAVIOR

The agent behavioral mechanism described here produces reactive rather than adaptive agents. Adaptive agents could also be modeled in MARSS with straightforward enhancements.

The main enhancement that could be undertaken would be to allow for grouping of tickets into goals, ticket exchange, ticket modification and construction, and the grouping of goals into roles. A proposed structure would involve many tickets, goals and roles in a similar fashion to previous work on adaptive agents. [Hiles et al 2001] This would allow agents to have a complex adaptive behavior, reacting to their environment and learning. This is not yet implemented in the MARSS model; however, it was a design consideration from the beginning.

In addition, there is scope to randomly generate connectors in order to implement some adaptive learning. These and other more complex mappings of the inner environment to agents' degrees of freedom would be a useful addition to the MARSS model.

Although evolving behaviors were designed and tested in the MARSS model some implementation problems prevented their use for production experiments. With more work these problems could be easily overcome. It is expected that a combination of evolving behaviors and designed experiments would be a particularly powerful analytical tool.

J. SUMMARY

The role of the behavioral sub-model is to map information from the inner environment to the degrees of freedom an agent has to change its own properties. This mapping is performed by the behavioral function. In MARSS these functions are implemented as connectors, tickets and regulators, and factors. By changing the factor levels the behavior of an agent will change. Once a general behavior is designed the task

is to find what levels to set the factors at to produce an emergent behavior. In MARSS this is done using genetic evolution and designed experiments

V. MARSS DESIGN AND FEATURES

A. GENERAL

The development of a modeling process to represent robot swarms and the MARSS tool occurred concurrently. This chapter is about how the model is implemented in MARSS and describes some features of the MARSS tool. Appendix A contains a graphical description of the MARSS tool and is designed to be read in conjunction with this chapter. This chapter expands on the graphical description where the pictures in Appendix A cannot tell the story.

The remainder of this chapter provides expansion on design aspects that are unique or important to MARSS, it describes how MARSS can be used to help answer questions and explore problems, and finally a description of enhancements that are possible are described.

MARSS is free for all to use. Further information, downloads, and up to date information may be found at http://diana.or.nps.navy.mil/~ajdickie/marss.

B. DEVELOPMENT ENVIRONMENT

1. General

MARSS was developed using the Java Software Development kit version J2SE 1.4 provided by Sun Microsystems. A number of extension Application Programming Interfaces (API's) were used to enhance the capabilities of the basic development kit provided by Sun. The tool used for all development was the integrated development environment provided by Sun, Forte for Java, Community Edition. All development tools were free and well documented. Without this capability MARSS would not have been developed.

2. Java3D

The Java3D API provided by Sun Microsystems was used to provide a 3D graphical interface. This scene graph orientated API was found to be easy to use and well supported. In conjunction with Java3D some code was used from the www.j3d.org code repository. The vrml97.jar scene graph loader provided by Sun was used to load the 3D models that represented entities.

3. Simkit

Simkit is an event driven simulation API designed by Professor Arnold Buss at the Naval Postgraduate School. Not all aspects of Simkit were used. It is the primary driver to run the simulation. The Simkit engine also handles all randomness in MARSS.

4. JDOM

JDOM is an API for using the eXtensible Markup Language (XML) in Java. It was used to load, save and process data in XML formats.

C. DESIGN

1. Design As A Tool

From the outset MARSS was designed for use as a tool for analysts. Many agent-based simulations are designed primarily to simulate some scenario to either visualize that scenario, or to mimic some observed behavior. MARSS can be used for both of these purposes; however, its strength lies in its ability to help answer questions. Once an analyst has described a problem, if that problem concerns trying to determine some tactic or behavior for a group of robots, then MARSS may be a suitable tool to provide insight. MARSS cannot provide any absolute answer. It does however present a unique capability amongst the range of tools available.

2. Information Encapsulation

When programming in an object orientated language such as Java some form of information encapsulation is routine. Without careful design it would still be possible for some programming construct to get information about another inadvertently. MARSS has been designed to try to prevent this. In a valid scenario, agents cannot gain any information about other agents except through the sensing mechanism.

3. Development of an API

MARSS is more than just a graphical tool that analysts can use. Much of the programming construct is suitable for use by other programmers. Parts of the program can be used for the creation of more elaborate, or even simpler, simulations. The ability of Java to automatically generate documentation, and the intention of making the source code freely available are important factors that enhance the use of MARSS as an API.

4. Deployability

Although designed to be used by analysts, it is recognized that not all analysts are Java programmers. With this in mind the intention from the outset has been to create a program that is easy for the analyst to install and use. This has affected the way that code is written in that it must be compatible with the deployed system, especially in relation to file input and output. InstallAnywhereNow was chosen as the deployment solution. InstallAnywhereNow is a free product available from www.zerog.com. The product provides an interface that allows the creation and deployment of install programs that shield the user from many of the problems associated with getting Java programs running. Without this product the deployment of MARSS would be much more difficult.

5. Model –View Separation

The main use of MARSS is to run simulation experiments. This is a computationally intensive process. The display of what is going on in the simulation is also computationally intensive. It is important to be able to execute the simulation

without any graphical display. The code to run the simulation model is completely separate from the code that runs the graphical view. There is a very thin interface that allows the graphical view to poll the simulation to check its state in order to represent it. It is possible to switch MARSS to 'text' mode for rapid scenario execution. This strips the entire graphical environment overhead from using memory and processor resources.

6. GUI Environment

The Graphical User Interface (GUI) is designed primarily as a scenario debugging and visualization tool. Face validation by an analyst is an important part of the model development process. The 2D and 3D displays of the model have proved invaluable for doing this. A side benefit that the displays provide is the ability to quickly brief observers about a particular scenario. The GUI is described in more detail in Appendix A.

7. Rapid Scenario Creation

A common criticism of large-scale US Department of Defense simulations is that scenarios can take a tremendous amount of time to create. The entire structure of MARSS is such that scenarios can be rapidly created and debugged. Scenarios are defined externally in XML using an XML editor (for more information on XML visit www.w3.org/XML/). A scenario defined in XML can be thought of as a rooted tree with nodes representing entities. Further nodes represent behaviors, properties and sensors for each entity and so on until the entire scenario is described.

Once a scenario has been created in XML, it can be validated outside of the MARSS program against an XML Schema. It may also be shared between users of MARSS like any other file.

8. Random Seed Management

The use of the random number capabilities of Simkit allows randomness to be effectively managed in MARSS. One random seed is used to control all randomness in a particular simulation run. This random seed can come from a list of seeds that is iterated

over between simulation runs, the seed itself can be generated randomly, or the user may provide the seed. This management of seeds allows repeatability for debugging purposes.

The main reason to manage random seeds however is for designed experiments. Doing so can greatly reduce variation between simulation runs, and reduce the time it takes to get data with the same level of accuracy.

9. Data Collection

MARSS has some data collection techniques to gather data between simulation runs. Currently the data that is gathered consists of the information in a swarm MOE, the random seed used, and the levels that factors are set at. The procedure used to gather this data can be relatively easily modified to gather any type of data between simulation runs.

10. Variable Time Step

Most agent-based simulations use a time step methodology where time is moved forward by some constant increment and the state of the simulation recalculated. This happens over and over, in many cases with a time step that cannot be changed. It has been shown that the size of the time step can change the results that one observes in both a qualitative and quantitative way. MARSS uses Simkit to allow each entity to have its own time step.

Each entity updates its own state. Upon doing this it then determines when to update its state next. It does this by generating a value from a random variate that belongs to that entity. This random variate may be a ‘constant’ random variate in which case the time step will be constant for that entity at each stage. Any random variate that is defined in Simkit can be used as this time step.

Note that this allows entities to update their state at different rates. Some entities may update their state more often so there is greater resolution, others may have few significant state updates and as such there is no need to spend valuable computational resources updating their state.

11. Event Diagram

The event diagram for MARSS is shown in Figure 2. This is a graphical representation of how and when the states of entities change as the simulation progresses.

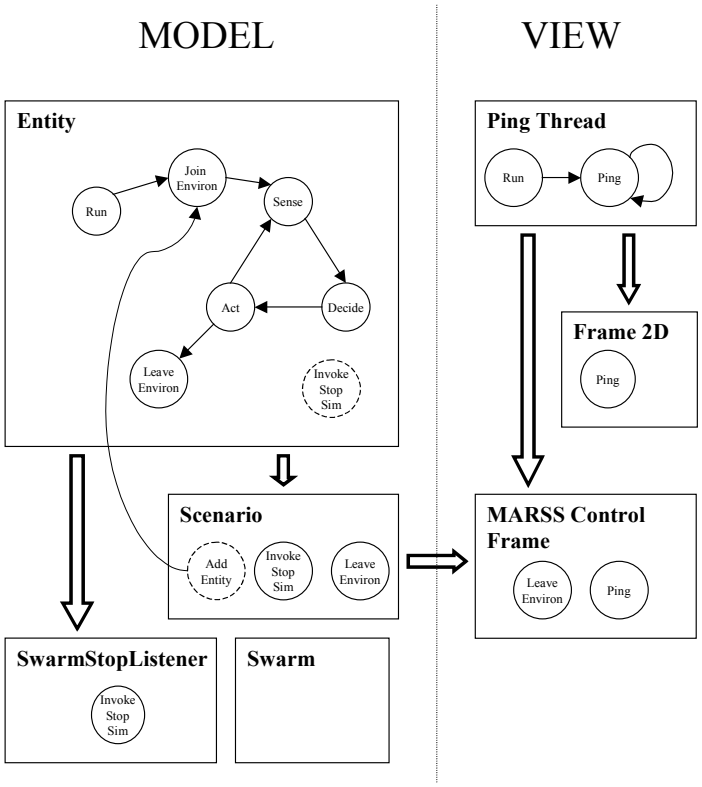

Figure 3. MARSS Event Diagram

Entities may enter the environment after the simulation has started. This allows an entity to 'appear' at a particular time in a simulation.

The main part of the simulation is the Sense-Decide-Act loop. These events are given priorities so that all Sense events happen before Decide events, which happen before Act events. In conjunction with the property change architecture, this allows entities to effectively carry out actions concurrently.

A Sense event schedules a Decide event to occur in zero simulation time. The Decide event will then schedule an Act event, again in zero simulation time. The Act event then schedules the next Sense event. The delay between Act and Sense is

determined on a per entity basis depending on the random variate allocated to control the step for that entity.

If two or more entities are scheduled to carry out a Sense event at the same time then they will all build their inner environment before the Decide event is undertaken. During the Sense event their sensors are asked to sense the ether and determine if there are any emissions that are detected.

During Decide events an entity's behavioral mechanism will make property change suggestions to those degrees of freedom that an entity decides to alter. All Decide events scheduled for a particular time will occur before the Act events.

The first thing to occur during an Act event is the agent's property change suggestions are applied and the property changes occur. The model then may also adjust properties such as location and velocity based on acceleration, for example. These model-adjusted properties are determined in code. The last thing an agent does before the Act event schedules a new Sense event is to have its emitters update any emissions that are in the ether to reflect its new state.

12. Coordinate System

A particular aspect of the MARSS environment is the coordinate system used. The coordinate system was chosen to enable easy integration with Java3D and is in nominal units (however meters are used for all scenarios). The coordinate system is right handed with y being height. This depicted in the figure below.

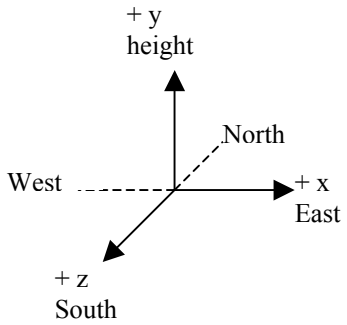

Figure 4. Coordinate System

13. Automated Designed Experiments

MARSS is designed primarily as a tool to conduct experiments. One important design consideration was to implement a FactorLevelSetter that controls what levels factors are set at between simulation runs. Such a device is built into MARSS. When factors are first defined an upper and lower level and an initial value is given for each. FactorLevelSetter's may be assigned to swarm's to control the factor levels for that swarm. Currently only a FactorLevelSetter for homogenous swarms is implemented.

The FactorLevelSetter can be assigned to control any factors that are important to the simulation response. The levels for factors not assigned remain at the initial value. The current FactorLevelSetter has four modes. The first is a full factorial mode. In this mode 2^n runs are conducted with n assigned factors set at upper and lower levels. Mode two sets the factors at random levels uniformly between the upper and lower level. Mode three is used for grid design. Assigned factors (n of them) are set at x levels for a design with x^n runs.

The last mode, mode four, is used to set factor levels according to a Latin Hypercube design. Each factor is split into n levels, where n is the number of factors assigned. A design is constructed so that no level appears twice for any factor over n runs, and so that no run has the same level for any pair of factors. An example of this design is given in the table below for an experiment with four factors.

Levels that each factor is set to in each run.		**Factor**			
		A	**B**	**C**	**D**
Run	**1**	3	2	1	4
	2	1	4	3	2
	3	4	1	2	3
	4	2	3	4	1

Table 1. An example of a 4x4 Latin hypercube

The FactorLevelSetter built into MARSS will automatically generate Latin hypercubes at random reasonably effectively for up to 30x30 size cubes (<3 seconds). Larger cubes can still be generated but it becomes very slow for cubes greater than 40x40.

Randomness is controlled for the full factorial, grid, and Latin hypercube by a set of seeds assigned to the FactorLevelSetter. For each set of full factorial, grid or Latin hypercube runs the seed is kept constant. This may be turned off and a random seed used for each run if desired.

D. USE OF MARSS

This section is not meant to be a full tutorial on how to build, debug and implement a particular scenario in MARSS. Rather it is should give the reader an idea of the scope and magnitude of the task.

The first step is to define the scenario in broad terms. How many entities, what are they doing, what is being represented. Before going to the next step the analyst should determine what he or she wishes to determine from the model. This can be the most time consuming step.

Before starting to implement a scenario it should be designed. This includes defining the following information:

- How many entities are being modeled and what signal types each entity should have.
- What bands are to be modeled and how does attenuation work for each band.
- What sensors are to be modeled and what capabilities each should have.
- What emitters need to be modeled.
- A behavioral mechanism for each entity.

Once the scenario is fully defined the time to implement it in MARSS is dependant on how much programming will need to be done. For most simple scenarios

there will be minimal programming. The following steps need to be undertaken to complete the implementation.

- Java programming of bands, sensors, emitters and measures of effectiveness that have not already been defined.
- Programming behaviors and connectors that cannot be represented by those already defined.
- Building an XML document in an external editor that describes the scenario. In many cases this will be merely a modification of an existing document.
- Debugging and tuning of the scenario.
- Analysis of data output and modification of the scenario.

The basic MARSS scenario is defined in an XML document that is validated against an XML schema. The details of how this is done are not included in this report. A more complete tutorial and description of how to implement a scenario in MARSS may be found at http://diana.or.nps.navy.mil/~ajdickie/marss.

E. FUTURE ENHANCEMENTS

A variety of future enhancements have been envisaged for MARSS. This include the following:

- Expanding the behavioral capabilities.
- Writing more bands, sensors, emitters, and measures for generic use.
- Incorporating an XML editor in MARSS specifically for developing MARSS scenarios.
- Creation of a more general XML schema specification and its associated loader that allows a wider range of scenarios to be developed in XML.

This is left as future work for any interested parties.

VI. A MICRO AIR VEHICLE SCENARIO

A. BACKGROUND

The Defense Advanced Research Projects Agency (DARPA) is managing a program to develop Micro Air Vehicles (MAV's). The charter is to develop technologies that will enable the production of very small aircraft capable of performing military missions. MAV's are defined as aircraft that are no more than 15cm in any dimension. Much of the research to date regarding MAV's has focused on overcoming the engineering challenges faced in designing such a small aircraft. More information on the DARPA MAV program, including papers describing their possible employment may be found at http://www.darpa.mil/tto/programs/mav.html.

The first MAV's that will be developed will probably have a human flying the vehicle by remote control, and will probably have relatively simple sensors that report information back to the controller. It is likely that future generations of MAV's will be able to fly autonomously and will have much more complex sensors with onboard processing capability. It is also likely that MAV's will have the capability to communicate some limited information.

With the capability to fly autonomously and communicate, MAV's will be able to act together to achieve a common mission. The individual behavior that enables a group goal to be pursued is the subject of this work. Important individual behavioral factors are explored. This work also investigates the effect of increasing an individual's knowledge about other searchers location and contacts.

B. SCENARIO DESCRIPTION

1. General

A potential use for MAV's will be to conduct search and reconnaissance. In order to conduct useful analysis a particular scenario was envisaged. It was important to select a scenario that had enough detail to investigate swarming behavior, yet was simple enough that it can be understood using analytical techniques. The scenario chosen consists of 25 MAV's searching a rectangular area for four tanks. A number of variations on this scenario are made to investigate various aspects of the model.

The general scenario presented is simple. This simplicity is required in order to compare the simulation results to those that can be generated analytically. As the specific scenario was enhanced to include more complex MAV behaviors it was decided to retain the simple scenario, so that MAV performance with more complex behavior could be compared to the base case.

The remainder of this section describes the basic scenario and its variations in more detail. The implementation of the scenarios in MARSS is described in the next section.

2. The Environment

The search environment is a flat 5000 x 5000 m area. There are no obstacles to movement or observation, either on the ground, or in the air. This model of the environment is a simplification of a flat desert scenario. While not proven in this work it is expected that the behavioral results obtained will not be overly sensitive to this simplification.

3. Target Tanks

The targets of the MAV's search effort are four "tanks". It is assumed that the tanks location probability is uniformly distributed over the search area. The tanks are moving at a constant speed of 10 m/s and are located in the search area at all times

(except for the follow tanks in Tank Behavior 3 which may sometime stray temporarily). Tanks move from waypoint to waypoint in the area. In order to ensure that the tanks location probability was uniformly distributed the general movement pattern of a tank follows that described in Appendix B. Three different behaviors were used for the tanks. These are as follows:

- **Tank Behavior 1**. Each tank is moving independently and randomly over the entire area. Each tanks start location is uniformly distributed over the entire area.
- **Tank Behavior 2**. Each tank is moving independently and randomly in a separate 2500 x 2500 m quadrant of the search area. Each tanks start location is uniformly distributed over its quadrant.
- **Tank Behavior 3**. A "lead tank" is moving in a random manner with the other three tanks following. The "follow tanks" have a behavior such that they will follow the closest tank they see in front of them at 100m at a 22.5 degree angle (two left and one right). Each follow tank has a sensor such that they can sense other tanks in front of them (90 degrees to each side) The lead tank notifies the follow tanks of a new way point. If for some reason a follow tank does not have a tank to follow, it will proceed to the way point and stop just prior to reaching it. There are many more subtleties to the tanks behavior however overall the emergent behavior has the effect of moving the group of tanks in a clump, roughly spaced at 100m. The start location of the lead tank is uniformly distributed over the entire area. The start location of the follow tanks is uniformly distributed in a 200 x 200 m square centered on the lead tank.

Tanks start moving as soon as the simulation is started and continue to do so until all are found. Tanks continue undertaking the behavior given to them even after being found. A tank has no way to know if it has been found, nor any way to evade being found. It is recognized that none of the tank behaviors described above represent the way real tanks operate on a field of battle. The tank behaviors described perhaps more closely represent the tactics undertaken by SCUD missile launchers in the desert, without the ability to hide.

4. Searcher MAV's

A real search mission would consist of a number of phases, including preparation for launch, launch, transit to search area, search, return, etc. For this analysis only the search phase of the MAV's mission is modeled.

In order for the simulation results to be compared with analytical results, starting conditions were abstracted from where searchers may be physically located in reality. In all cases searchers were initially located at 50m height.

Waypoints were located on the boundary and were also at 50m height in all cases. For all simple scenarios the searchers stayed at this height and paths were linear between waypoints. For some more complex scenarios the searchers were allowed to move between 0 and 300 m height, and were allowed to veer from a linear path; however would always eventually proceed to their current way point.

In all scenarios each MAV had an identical "tank sensor" this sensor is a conical sensor that always faces downwards with a cone angle of 90 degrees. The sensor is a cookie cutter sensor such that the probability of detection if a tank is inside the sensor cone is 1 and 0 otherwise. At 50m height this sensor has an effective sweep width of 100 m at ground level. Other sensors were added for the more complex scenarios to allow the MAV's to communicate, and to sense other MAV's locations.

MAV behaviors are divided into simple behaviors, where speed is constant and paths are linear, and complex behaviors, where searchers are using connectors and their acceleration property to control movement. For all simple behaviors the MAV's speed is constant at 30 m/s. The simple MAV behaviors are as follows:

- **Search Behavior 1**. The MAV's search in an exhaustive manner. Each MAV is assigned an area 5000 x 200 m. With 25 searchers the entire search area is covered. Each MAV splits its search area into two "sweeps" of 5000 x 100 m (based on the tank sensor sweep width). The search path is depicted in the figure below. Each MAV continues to conduct sweeps until all tanks are found.
- **Search Behavior 2**. The MAV's search in a random manner. Each MAV starts on the boundary and patrols the area in a random fashion using the movement strategy described in Appendix A. MAV movement is independent from other MAV's.
- **Search Behavior 3**. As for Search Behavior 2 however the start location is uniformly distributed over the search area.

Figure 5. Search Behavior 1 Individual MAV Path

Complex behaviors are discussed in the next section in conjunction with a description of how the scenarios were implemented in MARSS.

5. Searcher Measure of Effectiveness

The overall MOE used for all scenarios is the time taken by the MAV's to find all four tanks. In order to more fully understand the dynamics of a search, the time taken to find the first tank, the time from the first to the second, the time from the second to the third, and the time from the third to the fourth is also measured. This provides particular insights when comparing the results of the basic searches to analytical results. These inter-detection times also provide insight when trying to understand why searchers with complex behaviors perform as they do.

C. MARSS IMPLEMENTATION

1. General

All scenarios described were defined using XML. In some cases special java classes were written to implement particular characteristics of a searcher or target. The information contained in this section is a description of how the MAV scenarios were implemented in MARSS. Future scenarios created in MARSS could follow the considerations given in this section to create a successful scenario.

2. Bands

The first consideration made in implementing the MARSS scenario was to determine what information was being transferred between agents, and how to split this information flow into channels. The bands created for the MARSS scenario are as follows:

- **Visual Band**. This band was used to transmit information that would be received from visual sensors. The attenuation function for the visual band follows a square law. The received signal strength is related to the emitted signal strength by the following relationship, $S_r = S_s / d^2$. Nominal units are used for signal strengths. The important fact is that at d = 2 the amount

of signal strength remaining is one quarter of that at d = 1. This degradation of a visual strength assumes that the area of a received visual signal (corresponding to the number of pixels on an electronic sensor) is the measure of its strength.

- **Radio Communications Band**. The tanks used this band to transmit global information. It has no attenuation function i.e. $S_r = S_s$. The band is meant to represent a communication medium such as satellite communications.
- **Proximity Band**. For the more complex scenarios MAV's had to have a communications mechanism that was dependant on distance. The proximity band was used for this. The attenuation function was $S_r = S_e - d$ if $S_e - d > 0$ and $S_r = 0$ otherwise.

3. Properties

Both the MAV's and the tanks were implementations of the MobileEntity built into MARSS. Such MARSS agents have a standard set of properties. Some additional properties were required to implement the more complex behaviors. Not all properties listed below were used for all scenarios. The properties of the tanks and MAV's are summarized the tables below.

Property	Vector/ Scalar	Value	Bound	Use
location	vector	changeable	n/a	Tank location
size	scalar	10.0	n/a	By the visual band. The size is in nominal units.
health	scalar	n/a	n/a	not used
identifier	scalar	fixed positive integer	n/a	To uniquely identify the agent
orientation	vector	changeable	fixed to tank velocity	not used
velocity	vector	changeable	magnitude bounds – vector length constrained from 0 to 10	Tank velocity
acceleration	vector	changeable	n/a	not used
wploc	vector	changeable	n/a	Used to keep track of the current way point location of a tank for communication to other tanks.
transmit	scalar	changeable	n/a	Used to control the transmission strength in the Radio Communications Band

Table 2. Tank Properties

Property	Vector/ Scalar	Value	Bounds	Use
Location	vector	changeable	x (-3000, 3000) y (-1, 300) z (-3000, 3000)	Tank location
Size	scalar	10.0	n/a	By the visual band. The size is in nominal units.
Health	scalar	n/a	n/a	not used
identifier	scalar	fixed positive integer	n/a	To uniquely identify the agent
orientation	vector	fixed at (0.0,-1.0,0.0)	n/a	Used for the orientation of the visual sensor
Velocity	vector	changeable	magnitude bounds – vector length constrained from 0 to 30	Tank velocity
acceleration	vector	changeable	magnitude bounds – vector length constrained from 0 to 30	Tanks acceleration
Wploc	vector	changeable	n/a	Used to keep track of the current way point location of a MAV
foundtankloc	vector	changeable	n/a	Keeps the location of the closest tank that is currently being sensed. Not used for all scenarios. Note that the property is set to a large value if there is no tank found.

Table 3. MAV Properties

4. Emitters

In order to emit a signal that can be received each agent must have one or more signal types. The signal types of the tanks and MAV's are shown in the tables below.

Signal Type	Properties Transmitted	Notes
Entity	location	Default signal type
Tank	location, orientation, velocity	Included for illustrative purposes only. Not actually used
tank(i)	location, orientation, velocity, identifier	
tankmessage	location, wploc	used only by the "lead tank" to transmit a new way-point location

Table 4. Tank Signal Types

Signal Type	Properties Transmitted	Notes
entity	location	Default signal type
swarmer	location, foundtankloc	Not used for all scenarios

Table 5. MAV Signal Types

To generate emissions both the MAV's and tanks require emitters. The emissions generated have the capability to be seen as any of the signal types allocated to that agent. The signal type that an emission is seen as is dependant on an agent's sensors. The emitters that each agent has are listed in the tables below. Note that the signal strength is in no particular units.

Band	Initial Signal Strength	Notes
Visual	314	Based on the "size" property
Radio communications	1.0	Only on the "lead tank"

Table 6. Tank Emitters

Band	Initial Signal Strength	Notes
Visual	314	Based on the "size" property
Proximity	1000	Used only in the more complex scenarios

Table 7. MAV Emitters

5. Sensors

Sensors give an agent the ability to retrieve information from emissions and place that information in their inner environment. The sensors on the tanks and MAV's are listed in the tables below.

Sensor Band	Signal Type Capabilities (multiplier)	Base Sensitivity (nominal units)	Notes
Visual	entity (1.0), tank(1.2), tank(i)(1.4)	0.0005	The geometry is such that a tank will sense all visual signals in front of it.
Radio communications	entity(1.0), tankmessage(1.0)	0.0	Setting the signal type capability equal to that of entity effectively overrides that capability. The sensitivity is such that all emissions will be received.

Table 8. Tank Sensors

Sensor Band	Signal Type Capabilities (multiplier)	Base Sensitivity (nominal units)	Notes
Visual	entity (1.0), tank(1.2), tank(i)(1.4)	0.0005	The geometry is conical. A MAV will sense only in a cone centered on its orientation with a cone angle of 90 degrees
Proximity	entity(1.0), swarmer(1.0)	0.0	Setting the signal type capability equal to that of entity effectively overrides that capability. The base sensitivity is such that all emissions will be received.

Table 9. MAV Sensors

6. Behaviors

Each scenario had slightly different behaviors. The tank behaviors (Tank Behavior 1 through 3), and basic MAV behaviors (Search Behavior 1 through 3) are described in the previous section. These behaviors were implemented in MARSS by explicitly writing behavior classes to implement them. The specifics of how the behavior was implemented will not be discussed in more detail here.

The more complex MAV behaviors were implemented with the connector methodology. Although regulators were tested on some behaviors they were not used for any production runs. The table below describes the connectors used for complex MAV behaviors. Not all connectors were used for all scenarios. By combining various connectors Search Behaviors 4 through 7 are described in the table.

Input properties	Affects property	Factors (bounds)	Used with Search Behavior	Notes
location wploc	wploc	wpDistance (10,200)	4,5,6,7	Determines the distance from the MAV's current location to wploc and if less than wpdistance replaces wploc with a new way point generated at random (using the methodology in Appendix B)
location wploc	acceleration	accToWP (-100,100)	4,5,6,7	Determines the vector from current location to wploc, normalizes it and scales by factor accToWP. The result is suggested as a change to acceleration.
location tank(i) – location (sensed property)	acceleration	accToTank (-100,100)	5,6,7	Determines the vector from current location to the closest sensed tank, normalizes it and scales by factor accToTank. The result is suggested as a change to acceleration.
location swarmer – location (sensed property)	acceleration	accToNear	6,7	Determines the vector from current location to the closest sensed MAV, normalizes it and scales by factor accToMAV. The result is suggested as a change to acceleration.
location tank(i) – location (sensed property)	foundtankloc	n/a	7	Maps the location of the closed found tank to foundtankloc so this property will be transmitted. If no tank is found sets foundtankloc to a magnitude greater than the area of interest.
location swarmer – foundtankloc (sensed property)	acceleration	accToFound	7	Determines the vector from current location to the all sensed swarmers foundtankloc property, normalizes it and scales by factor accToFound. The result is suggested as a change to acceleration. If the foundtankloc magnitude is greater than the area of interest the foundtankloc is ignored.

Table 10. MAV Connectors

7. MOE's

Each searcher has an internal MOE that records the time a tank was found and its identifier. Future sightings of the same tank are ignored. The individual MOE's are associated with an external swarm MOE that takes the information from all tanks and again ignored duplicate sightings. In effect the swarm MOE measure when four tanks have been uniquely identified. When this occurred the swarm MOE invokes the stop simulation event in the entity that contributed to the find.

8. Other MARSS details

A "SwarmStopListener" is added to each scenario to listen for the stop simulation event. This construct outputs the information contained in the swarm MOE to a file, including the time of first detection of each tank. The random seed used for a particular run is also recorded.

Various Geo-Referenced Maps were used to provide a background for the visual simulation. This was helpful in the debugging stage of scenario development.

Simulations were generally set for over 10,000 runs and stopped after a few thousand as time permitted. The simulations were designed to stop at time 20,000 if all tanks had not been detected although such an event was not observed in production runs.

A constant time step of 0.5 was generally used for both tanks and MAV's. This provided enough performance without seriously affecting the model of the sensors.

A standard list of 1000 seeds was used for production runs of the more complex scenarios where behavioral factors were being investigated.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. MAV SCENARIO RESULTS AND ANALYSIS

A. GENERAL

A range of scenarios were created using combinations of the tank and searcher behaviors described in the previous section. The scenarios can be split into simple scenarios and complex scenarios. The simple scenarios were designed to produce results that could be compared to analytical results, and to provide a base case to measure MAV performance in the complex scenarios against. The simple scenarios were either exhaustive search or random search.

The complex scenarios were designed to provide information about how improving individual searchers knowledge will improve the performance of the swarm. The analysis of these scenarios consisted of first finding the best behavior within the sensing and behavioral bounds given, and then determining and understanding the swarm performance with that behavior.

B. EXHAUSTIVE SEARCH

1. Analytical Results

Analytical results were determined for a simplified version of the exhaustive search scenarios implemented in simulation. The exhaustive search scenario consists of 25 searchers searching for four tanks in a 5000 x 5000 m area. The tanks are moving at 10m/s and the searchers at 30 m/s. The tanks movement is such that they are equally likely to be at any point in the area. The searchers each have an area 5000 x 200 m that they are sweeping with their 100m sweep width sensor in an exhaustive way. The tanks start at random locations and the searchers start 50m from the boundary of their first sweep. The following assumptions are made to investigate this scenario analytically:

- The searchers move at constant velocity and turn instantly.
- The tanks move at constant velocity and their movement is such that they are equally likely to be at any point in the area at any time.

- If a searcher sweeps a lane on one pass then by the time the next pass occurs, a tank is equally likely to be in either lane (this assumption is made based on the speed of the tanks and their random movement).
- A tanks movement is independent from other tanks and from the searchers.
- The sensor has a perfect sweep width of 100m and the searcher can exactly position itself between sweeps.

The task of the analysis is to determine the expected value and variance of the time to detect each of the four tanks, and to determine the same moments for the total time until all four tanks are detected. This is approached by considering the inter-detection times for each tank.

Initially the problem is simplified by determining the solution for a single tank. The expected time to detect a single tank can be determined from the expected number of sweeps and the expected time to detect a tank during a sweep. The probability of detecting a single tank on a given sweep is 0.5. The probability of detecting it on the next sweep is 0.25 (probability of not detecting on the first sweep multiplied by the probability of detecting on the next sweep 0.5 x 0.5). It is recognized that the number of sweeps are geometric with probability 0.5. The expected number of sweeps is equal to 2.

The time taken for one sweep is equal to the distance (5000m) divided by the velocity (30 m/s), or 166.67 s. As the tanks are equally likely to be found at any point in the sweep the expected time to detect in a sweep is uniformly distributed with a mean of 83.33s. The overall expected time to detect is equal to the time taken for one less than the expected number of sweeps, plus the expected time to detect on the final sweep. In addition 3.33s for the inter sweep times must be added. The resultant expected time to detect a single tank is 253.3 s.

Rather than explicitly calculating the variance we can make the assumption that the time to detection is roughly exponential. The lack of memory property of the geometric distribution allows this. Using the exponential assumption for time to detect the detection rate is 1/253.3 = 0.003947. The rate of detection of the first of four tanks (T1) is therefore 1/(4 x 0.00394) . Because of the memory less property of the exponential distribution the rate of detection for first of the remaining three tanks (T2) is

1/(3*0.003947) and so on for the rate for T3 and T4. This produces the expected values for T1 through T4 in the table below. With the exponential assumption the standard deviation is the same as the expected value. The values T1 to T4 are refereed to collectively as inter-detection times (even though T1 is the time until the first detection).

E [T1]	**E [T2]**	**E [T3]**	**E [T4]**
63.3s	84.4s	126.7s	253.3s

Table 11. Expected detection times for each tank using exhaustive search

To calculate the moments associated with the total time to detection we continue using the assumption that the inter-detection times are independently distributed according to the exponential distribution. The distribution of the total time to detection is the sum of the four exponential inter-detection times. It can alternatively be thought of as the maximum of four independent exponential variables with rate parameter equal to the underlying detection rate of a single target. Using the former reasoning, and the expected values and variances given above, the expected value for the total time to detect is 527.8s and the standard deviation is 302.3s.

2. Experimental Results

Three exhaustive search experiments were conducted. In all cases Search Behavior 1 was used. The performance of the searchers was recorded with each of the tank behaviors. A summary of the results for each experiment is shown in the tables below. Means, standard deviations and standard errors of the means were determined using traditional statistical methods. Standard errors of standard deviations were determined using the bootstrap function in the statistical package S-Plus with default parameters (1000 replications)

Experiment A used Search Behavior 1 and Tank Behavior 1. This experiment is expected to most closely represent a real exhaustive search scenario where the targets

location probability is uniformly distributed over the area. The results of experiment A are summarized in the table below. Experiment A consisted of 9848 runs.

	T1	T2	T3	T4	Total
Mean	67.2	78.5	117.8	241.3	504.7
Std Error (Mean)	0.6	0.8	1.2	2.5	2.9
Std Deviation	58.1	79.1	121.6	244.5	290.4
Std Error (Std Deviation)	1.0	1.3	1.8	3.9	3.6

Table 12. Experiment A results

Quantile-Quantile exponential plots were used for visual comparison of the exponential assumption of inter-detection times. These plots are shown in the Figure below. It is observed that the exponential assumption holds relatively well for T2 through T4. It holds less well for T1 however is still relatively good.

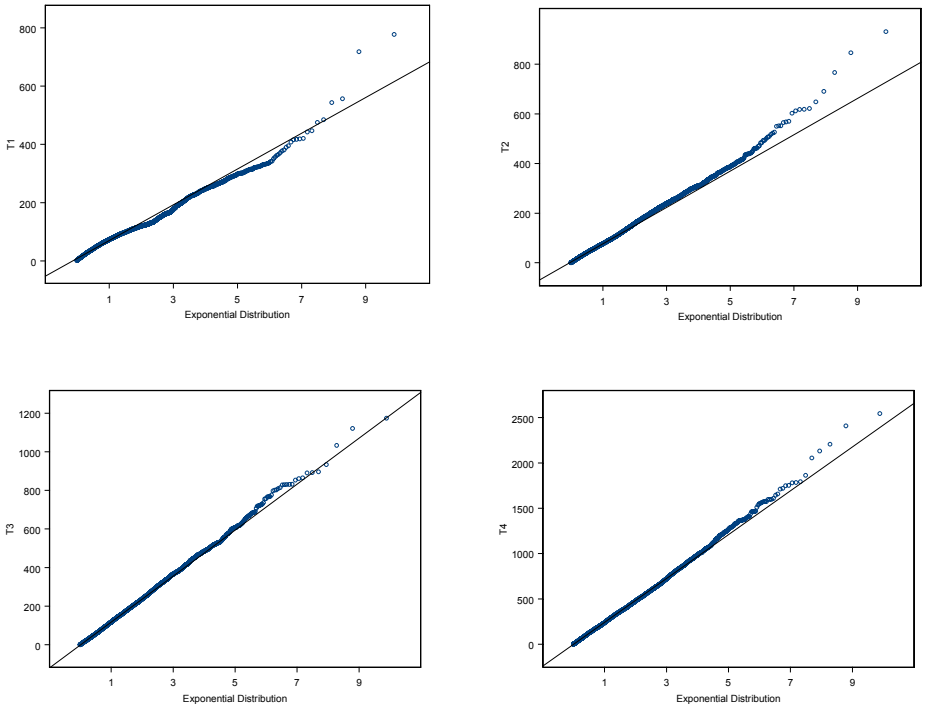

Figure 6. Plots to check the exponential assumption for experiment A

Experiment B used Search Behavior 1 and Tank Behavior 2. The results are summarized in the table below. Experiment B consisted of 4277 runs.

	T1	T2	T3	T4	Total
Mean	66.5	86.9	113.7	233.9	501.0
Std Error (Mean)	1.0	1.1	1.7	3.8	4.5
Std Deviation	65.7	74.5	109.3	247.8	294.2
Std Error (Std Deviation)	1.5	1.6	3.0	5.7	5.7

Table 13. Experiment B results

Experiment C used Search Behavior 1 and Tank Behavior 3. The results are summarized in the table below. Experiment C consisted of 1065 runs.

	T1	T2	T3	T4	Total
Mean	86.5	51.0	127.0	252.1	516.7
Std Error (Mean)	1.6	3.0	4.8	8.2	9.0
Std Deviation	52.8	99.5	156.8	266.2	293.2
Std Error (Std Deviation)	1.8	4.0	6.0	10.2	12.3

Table 14. Experiment C results

3. Discussion

The results suggest that Experiment A and Experiment B are a reasonable approximation to the analytical results determined for exhaustive search. The confidence intervals for the mean inter-detection times observed do not generally fall over the mean inter-detection time calculated. In general observed inter-detection times are slightly less than that calculation, however they are relatively close. There are a number of reasons why observed detections would occur sooner than the analytical predictions. The detection assumes a sweep width about a point. In reality the sensor is sensing in front of the searchers position and detection may occur before the searcher reaches the targets location. In addition the search effort between sweeps is not taken into account in the analytical result.

The exponential assumption for inter-detection holds relatively well for experiments A and B. In addition to the relationship between the moments we observe appropriate QQ-plots. This is especially for T2 through T4 and the total detection time. The observations for T1 do not meet the exponential assumption quite as well. This result is perhaps due to the start conditions in each experiment.

In addition the predicted expected moments for total detection time agree reasonably well with those observed. This is shown in the figure below.

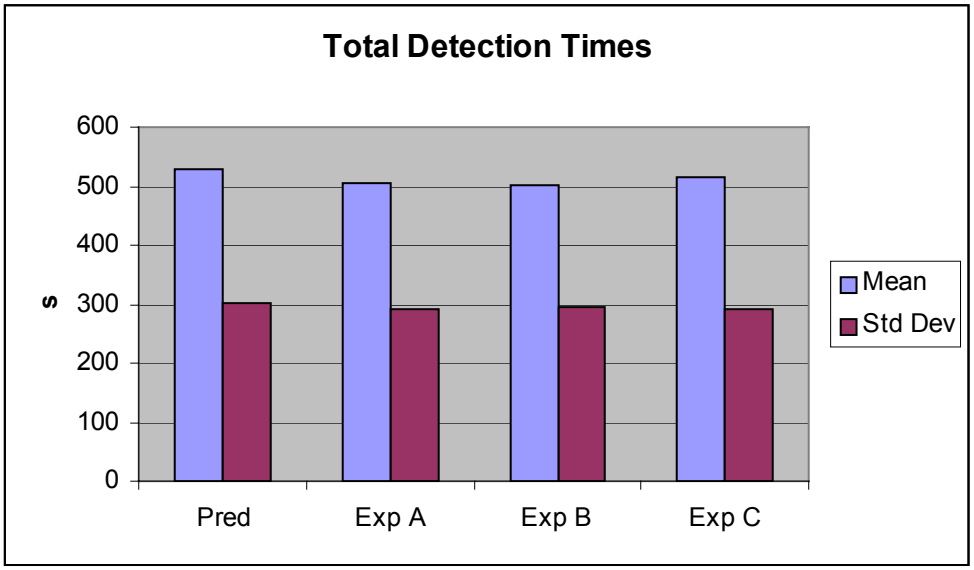

Figure 7. Predicted and observed total detection times for exhaustive search

The results from Experiment C show that clumping the tanks together increased the value of T1 however decreased the value for T2. This makes intuitive sense. Because the tanks locations are dependant and close together, the conditional probability of finding a second tank is dependant on whether one has already been found on that sweep.

The exhaustive search pattern provides the best theoretical coverage of the search area. For an exhaustive search to be effective searchers must coordinate their efforts fully. When a central controller can direct the movement of every individual this is possible. Due to their size MAV's may not have the communication capabilities to receive directions from a central controller. The random search scenario is developed to provide a base case assuming that the MAV's have no communication capability and proceed on a purely random search.

C. RANDOM SEARCH

1. Analytical Results

For the random search scenarios the MAV's move in a random manner such that they covered each portion of the search area evenly. The following assumptions are made to investigate the random search scenario analytically:

- The searchers move at constant velocity and turn instantly.
- The tanks move at constant velocity and their movement is such that they are equally likely to be at any point in the area at any time.
- The tanks movement is independent from other tanks and from the searchers.
- The searcher sensor has a perfect sweep width of 100m.

The analytical argument for a random search is well documented in the literature [Naval Ops Analysis]. Detection times follow the exponential distribution with rate (W x v)/A where W is the sweep width, v is the searcher velocity and A is the area to be searched.

The search rate for a single target is determined first. The sweep width for all 25 searchers is 2500m. Traveling at velocity 30 m/s and covering area 5000 x 5000 m the predicted rate of detection for a single target is 0.003. A similar argument to that made for the exhaustive search for determining the inter-detection times can be made with random search. The expected inter-detection times are shown in the table below.

E [T1]	**E [T2]**	**E [T3]**	**E [T4]**
83.3	111.1	166.7	333.3

Table 15. Expected detection times for each tank using random search

The moments for the total time of detection are determined in a similar manner to that for the exhaustive search. The expected value for the total time to detect is 694.4s and the standard deviation is 397.7s.

2. Experimental Results

Four experiments were conducted using searchers moving at random. The general set up of the scenario was the same as for the exhaustive search experiments, however the movement of the searchers followed the random waypoint generation method described in Appendix B.

Experiment D used Search Behavior 3 and Tank Behavior 1. The searchers start internally and search randomly, and the tanks movement is independent and random. This experiment is as close as possible to the scenario investigated analytically. The results are summarized in the table below. A total of 3341 runs were made for this experiment.

	T1	T2	T3	T4	Total
Mean	80.7	109.1	164.8	323.1	677.7
Std Error (Mean)	1.5	1.9	2.8	5.4	6.6
Std Deviation	84.2	108.9	161.4	313.1	381.0
Std Error (Std Deviation)	1.9	3.0	3.5	6.9	7.1

Table 16. Experiment D results

A slight variation on the start location of the searchers was made for Experiment E. This experiment used Search Behavior 2 and Tank Behavior 1. The searchers start on the boundary and search randomly, and the tanks movement is independent and random. The results are summarized in the table below. A total of 2725 runs were made for this experiment.

	T1	T2	T3	T4	Total
Mean	79.9	102.5	162.7	332.4	677.4
Std Error (Mean)	1.5	2.1	3.1	6.6	7.7
Std Deviation	78.8	107.7	161.4	342.6	401.4
Std Error (Std Deviation)	2.3	3.0	3.9	10.1	9.5

Table 17. Experiment E results

Experiment F Search Behavior 2 and Tank Behavior 2. The searchers start on the boundary and search randomly, and the tanks movement is random in an assigned

quadrant. The results are summarized in the table below. A total of 5390 runs were made for this experiment.

	T1	T2	T3	T4	Total
Mean	81.2	109.9	171.5	336.2	698.8
Std Error (Mean)	1.1	1.5	2.3	4.7	5.6
Std Deviation	80.9	109.9	170.3	343.7	409.8
Std Error (Std Deviation)	1.5	2.0	3.8	9.2	10.9

Table 18. Experiment F results

Experiment G used Search Behavior 2 and Tank Behavior 3. The searchers start on the boundary and search randomly, and the tanks move together in a group. The results are summarized in the table below. A total of 1718 runs were made for this experiment.

	T1	T2	T3	T4	Total
Mean	130.1	83.3	136.4	280.9	630.7
Std Error (Mean)	3.1	3.3	4.6	7.9	10.2
Std Deviation	129.2	135.8	189.6	328.0	422.0
Std Error (Std Deviation)	4.3	5.2	6.7	11.4	10.6

Table 19. Experiment G results

3. Discussion

The results for experiments D, E and F agree with the results predicted analytically. In all cases the exponential assumption for inter-detection times holds. Comparing the relationship between the observed mean and standard deviation confirmed this. QQ-Plots were also used to check the exponential assumption.

The inter-detection times are slightly less than predicted by analysis. The searchers are finding the targets a little earlier than predicted by the analysis. If a target is directly in a searchers path the sensor will detect the target about 1.7 seconds prior to the sensor being directly over the target. In addition it is expected there is some increased search effect at the edge. These effects were not taken into account for the analytical determination of the expected moments. The slightly better inter-detection times can be explained with these effects.

The results for experiment G represent a significant departure from the exponential inter-detection times. By comparing the observed moments and the QQ plot in the figure below T1 looks to be exponential, however the expected detection time is much greater than that predicted by theory (~130s observed compared to ~83s predicted). T2 to T4 are not exponential. It is expected that the cause of this departure is a violation of the independence assumption for the tanks movement.

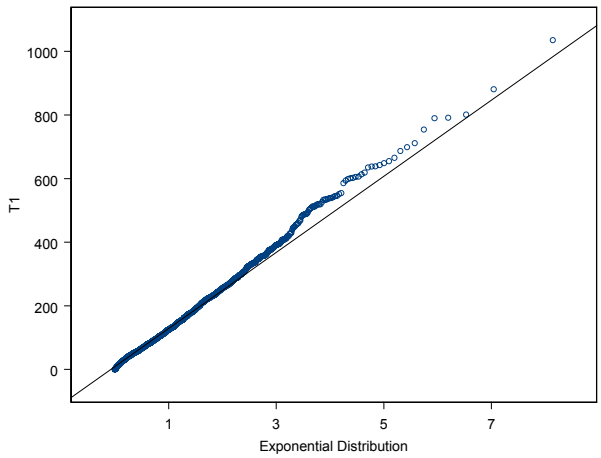

Figure 8. QQ-plot of observed values for T1

In experiment G the tanks are following each other at about 100 m. The problem to find the first tank is really to find the group of tanks, however the target is now no longer a point, rather it is the distribution of the group. This effectively increases the theoretical sweep width of the sensor. An exact calculation of the T1 based on this increased theoretical sweep width is not presented. Working backwards from the observed detection rate the sweep width is about 250 m. This seems to be about what would be expected for the theoretical sweep width. Once the first tank is found it is much more likely to find the second and so on for the third and fourth. This is most likely the cause of the departure for the experimental assumption for T2, T3 and T4.

The total detection times for all random search experiments are shown in the figure below. Again experiment D, E and F match what is predicted by theory.

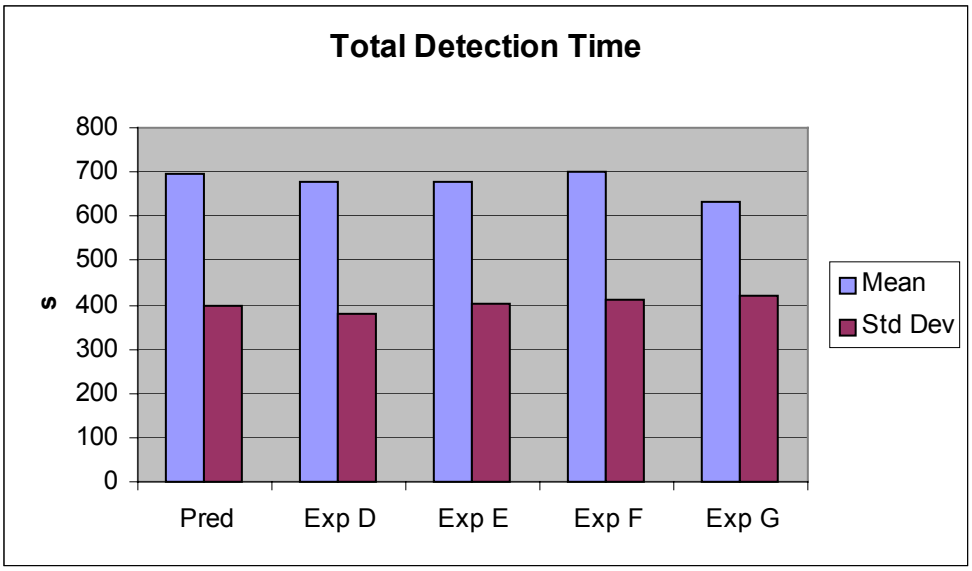

Figure 9. Predicted and observed total detection times for random search

It is interesting that the results of experiment G are close to that predicted by the exponential theory even though the observed inter-detection times were not exponential. There is perhaps some limiting behavior as the number of targets in the group expands. It is expected that the group separation is important in determining the departure from the exponential assumption. Further work could investigate the effect of group separation on exponential inter-detection times and the overall detection time.

Experiments F and G are used as a basis to determine more complex behaviors. These more complex experiments are discussed in the next section.

D. MORE COMPLEX SEARCH BEHAVIORS

1. Experiment Design

The more complex search behaviors all have the random search behavior as a base. Searchers roughly follow the random movement algorithm of Appendix B. Most movement is by accelerating to a waypoint with a capped velocity. Movement is modified by adding acceleration components in the direction of (or away from) sensed tanks, other MAV's, and tanks sensed by other MAV's.

Experiments H through K build on the base provided by experiment G. In all those cases the tanks are using the same behavior (following a lead tank). Experiments L through O build on experiment F. In those cases the tanks are moving in separate quadrants of the search area.

The aim of the experiments conducted with more complex search behaviors was to determine the effect of behavioral control factors on swarm performance. In total eight experiments are reported here. The table below provides a summary of the configuration of targets and searchers for each experiment.

Experiment	**Target Configuration**	**Searcher Configuration**	**Factors to measure**	**Experiment Design**
H	Tank Behavior 2	Search Behavior 4	wp, dist	Grid – four levels
I	Tank Behavior 2	Search Behavior 5	wp, dist, tank	Grid - four levels
J	Tank Behavior 2	Search Behavior 6	wp, dist, tank, near	Latin Hypercube
K	Tank Behavior 2	Search Behavior 7	wp, dist, tank, near, found	Latin Hypercube
L	Tank Behavior 3	Search Behavior 4	wp, dist	Grid – four levels
M	Tank Behavior 3	Search Behavior 5	wp, dist, tank	Grid - four levels
N	Tank Behavior 3	Search Behavior 6	wp, dist, tank, near	Latin Hypercube
O	Tank Behavior 3	Search Behavior 7	wp, dist, tank, near, found	Latin Hypercube

Table 20. Target and Searcher Configuration for Complex Experiments

The design of each experiment was chosen to ensure a good coverage of the response surface, while ensuring that each design point would have enough simulation runs to get an accurate picture of the response at that point.

A summary of the meaning for each of the factors is given in the table below. Note that the factors related to acceleration control property change suggestions for the acceleration property. The property change suggestions are added (using standard vector addition) and then the resultant vector restricted by the magnitude bounds placed on the acceleration property (<30).

Factor	Abbreviation	Meaning	Level Range
accToWP	wp	The magnitude of acceleration towards the current way point	(3, 20)
wpDistance	dist	The distance from the current way point where it is discarded and another chosen using the algorithm described in Appendix B	(10, 200)
accToTank	tank	The magnitude of acceleration towards the closest tank that is currently being sensed	(-100, 100)
accToNear	near	The magnitude of acceleration towards each MAV currently being sensed	(-80, 20)
accToFound	found	The magnitude of acceleration towards each tank that is being sensed by each "near" tank.	(-100,100)

Table 21. Summary of factors that are considered

The range of the factor levels described above were chosen by trial and error by visually determining the effect of each factor. Here the visual display associated with MARSS was particularly useful. Note that the much larger magnitude levels for *tank, near*, and *found*, when compared to *wp,* were effective at overcoming any acceleration towards the current waypoint without the need for regulators.

2. Experimental Results

Each of the experiment described above were run overnight on a Pentium 4 computer. The number of runs for each experiment is given in the table below. Note that this is the raw number of runs, not the number of "grids" or "cubes".

Experiment	H	I	J	K	L	M	N	O
Number of Runs	3653	4876	902	1207	5274	5026	1189	1587

Table 22. Number of runs for each complex experiment

The aim of the analysis is to identify factors that contribute to the response. Initially this was attempted using classic linear regression models. While linear regression techniques provided some information there were limitations to this analysis. These included the following:

- The assumptions of normal residuals and homoscedasticity do not hold for the nature of the response observed. By transforming the response (using log(y) or $y^{\wedge 1/3}$ for example) the residuals can be made to look more normal.
- In many cases the response was not linear. In particular the response was stepped for particular predictor variables. In such cases even a polynomial regression model could not capture the observed means.
- A linear regression model does not readily produce output that helps identify breakpoints in the levels for a particular factor.

Examples of the complexity of the response are given in the figure below for experiment J. A boxplot is given for each level by factor. The complexity is particularly apparent for the factors *tank* and *near*.

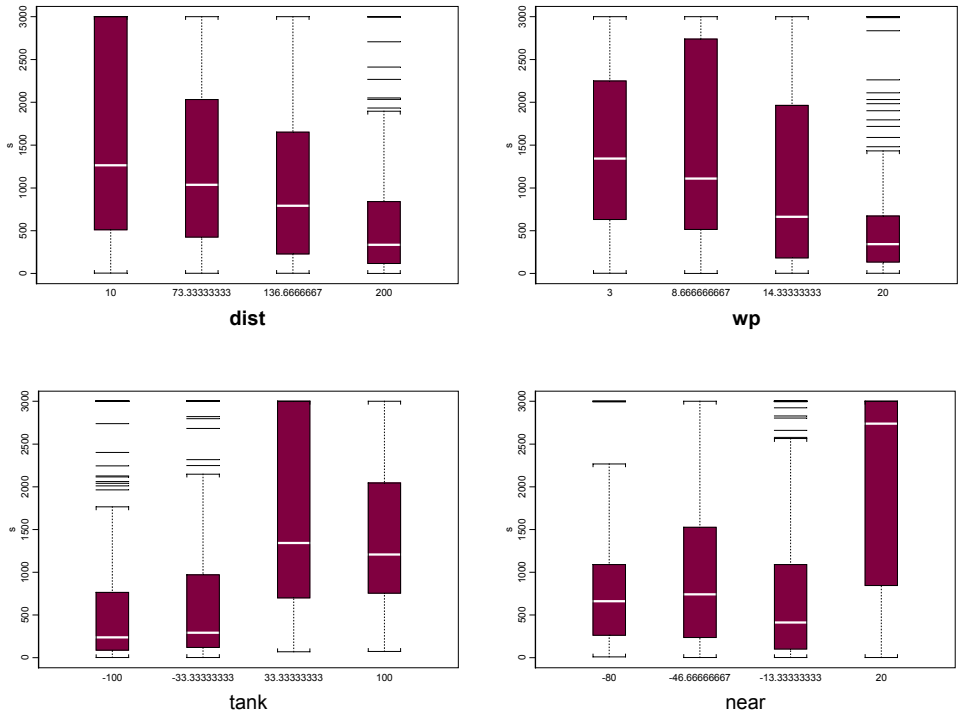

Figure 10. Response for Experiment J by Factor

Other experiments exhibit a response with similar complexity. What are not shown in the figure above are interactions between the various factors. In addition to this being hard to represent, interactions are deliberately confounded in the Latin hypercube design in order to better understand the main effects.

It is easily observed from the figure above that a model assuming linearity for *tank* or *near* would not appropriately find the step in *tank* between –33 and 33, or the dip in *near* at around –13. Indeed even a third order polynomial model may not accurately be able to describe the response.

After trying many techniques for describing the data a Regression Tree proved to be the most interpretable, and the best technique to give the information sought regarding important factors. The "tree" function in S-Plus was used for this analysis. A workflow was created to treat the output from all experiments in a similar way.

Initial a tree is created that attempts to predict *total* from all other factors. The tree is cross validated using the "cv.tree" function, and the size of tree with the minimum

deviance determined. The tree is pruned using the "prune.tree" function to the appropriate size, and the leaves ordered so the minimum response is on the left. The tree for experiment J is shown in the figure below.

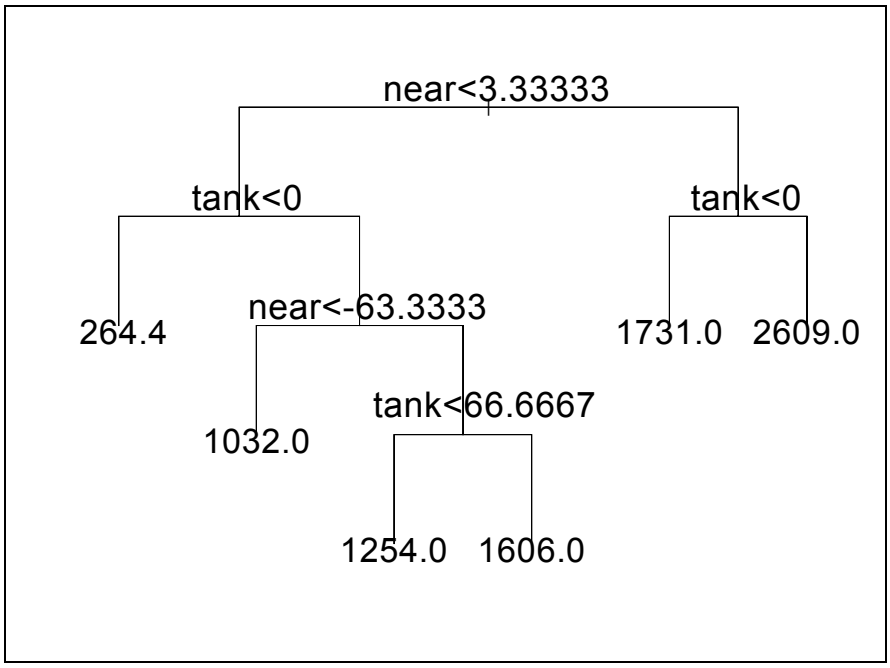

Figure 11. Regression tree for Experiment J

The tree above predicts that if *near* is less than 3.3 and *tank* < 0 then the observed mean response will be 264.4. By filtering the simulation observations for this experiment to only those data points that meet these criteria the following summary statistics are observed. Count = 292, mean = 264.4 (as predicted by the tree), standard deviation = 258.4, and the standard error of the mean = 15.0.

The use of regression tree has assumptions of normal residuals and homoscedasticity just like linear regression. Although these assumptions are not met any better for the regression tree they are much less important to the outcome. Again by taking the log of the response these assumptions can be met slightly better. However, doing so has a cost of interpretability and the leaves of the tree no longer match with the observed values.

Using the procedure described, regression trees were constructed for each experiment. These trees are shown in the figure below.

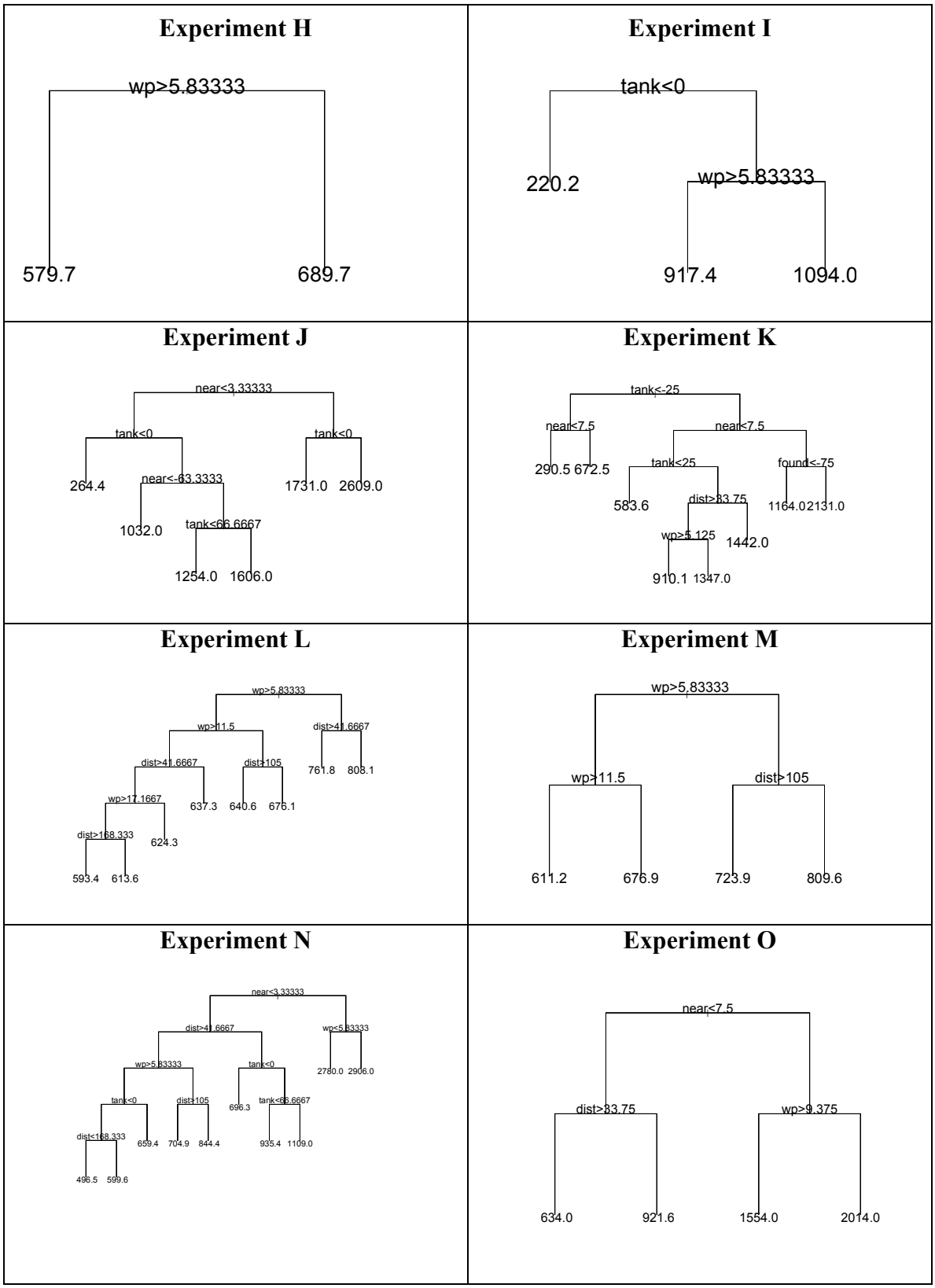

Figure 12. Regression Trees for all Complex Experiments

Summary statistics were generated for each data point that met the tree conditions pertaining to the lowest total (the far left leaf on each of the above trees). These statistics are summarized in the table below.

Experiment	**H**	**I**	**J**	**K**
Factor Levels	wp > 5.8	tank < 0	near < 3.3 tank < 0	tank < -25 near < 7.5
Number of Runs That Satisfy	2739	2448	297	370
Mean Total Time	579.7	220.2	264.4	290.5
Std Error of the Mean	7.6	4.6	15.0	15.6
Standard Deviation of Total Time	399.5	227.8	258.4	300.0
Experiment	**L**	**M**	**N**	**O**
Factor Levels	wp > 17.6 dist > 168	wp > 11.5	near < 3.3 tank < 0 wp > 5.8 41.7<dist<168.3	near < 7.5 dist > 33.75
Number of Runs That Satisfy	329	2512	151	1037
Mean Total Time	593.4	611.2	496.5	634.0
Std Error of the Mean	18.0	7.2	21.0	11.1
Standard Deviation of Total Time	325.8	358.7	258.2	356.8

3. Discussion

The results from Experiments H through K will be considered first. In these cases the searchers were looking for four tanks that were moving in a group. Experiment H was designed to closely replicate experiment G. The only factors controlling movement were the magnitude of acceleration towards the next waypoint (*wp*) and the distance from that waypoint where another would be selected (*dist*). The resultant behavior has searchers executing a smooth turn starting before the boundary.

The regression tree for experiment H suggests that the only significant factor that contributes to the response is *wp*. In this case if *wp* is greater than 5.8 the mean of total

time to detect all four tanks is observed as about 580. This represents about 10% decrease in detection time when compared to experiment G (mean = 630), however the search is not as good as the exhaustive search in experiment C (mean = 516). This decrease in detection time may be explained by searcher no longer wasting search effort at the boundary. Rather, the level for *wp* controls movement in such a way that a searchers search cone merely touches the boundary before proceeding to the next waypoint.

For experiment I searchers were given the ability to accelerate towards or away from the closest tank they observe. The factor controlling this magnitude (*tank*) dominated the response. By accelerating away from a tank (i.e. tank <0) the response was reduced from the pure random search of experiment G by around 65%. By accelerating away from a tank the MAV slows and climbs. This has the effect of both increasing the effective search radius, and of remaining in the general area of the observed tank for longer. Since in this scenario the tanks are moving together it is obvious that this increases the chance of finding more tanks. The only other split in the regression tree again confirms that the higher the acceleration towards a way point the better.

Experiment J built on experiment I by giving searchers the ability to accelerate towards or away from other searchers (within about a 500m radius). It was expected that by doing this searchers would maintain a separation so that they more effectively covered the search area. Instead it was found that if searchers accelerated towards each other at all, detection time drastically increased. This, combined with any acceleration towards a found tank, confounded many of the observed results. The resultant total detection time still represents an achievement over the pure random search, however is not as good as that observed for experiment I.

Experiment K allows the searchers to accelerate towards or away from all tanks found by another searcher (again limited to searchers < 500m away). Again low values of *near* and *tank* dominated the response. Any high values of these factors confounded the response. The resultant total detection time is similar, although a little higher, to the response that was observed for experiment J.

Over all of the experiments H through K the most important factor at improving detection time was the acceleration away from a detected tank. For the latter two

experiments additional factors did not improve the total detection time. It is expected that if the factor levels for near and found were constrained to reasonable ranges that the results may be more telling.

Experiments L through O had tanks moving randomly in separate quadrants. These experiments most closely match experiment F. Again with just *wp* and *dist* experiment L showed a significant improvement (around 15%) over the pure random search of experiment F. This confirmed the result obtained from experiment H that the greater the magnitude of acceleration towards the current way point, the better the overall searcher performance. It appears from experiment M that the addition of the factor *tank* does not significantly affect the performance. The deliberate geographic dispersal of the tanks explains why the factor *tank* has does not have the same effect as for experiment I.

The inclusion of a searcher's ability to react to other tanks does appear to improve performance in experiment N. A complex mix of factor levels produces a performance that is on par with the exhaustive search of experiment C. However, the small number of observations that meet the complex factor level criteria makes this result somewhat suspect. Adding the factor *found* in experiment O messes everything up and produces the worst response from the experiments L through O.

In summary, experiments L through O show that again *near* is the dominant factor where it is present. The results suggest that it is possible to approach the exhaustive search performance by having a complex movement mechanism based on accelerations such as those considered.

E. SUMMARY OF RESULTS

The basic scenarios (experiments A through G) showed that the search implemented in MARSS agrees with analytical results. This agreement validates the MAV search scenario that was implemented. The results provided a base to compare the complex scenarios against.

The results from the complex scenarios were surprising and insightful. The increased search performance based just on the ability of searchers to turn before the boundary of the search area was surprising. Also surprising was the lack of increased performance based on the factors *near* and *found*. A better range for the levels of *tank*, *near* and *found* could have been made where these factors were present. Only negative values of *tank* and *near* should have been considered. The appropriate levels for *found* are not clear.

F. FUTURE WORK

Future work with the MAV scenario could include the following:

- Rerunning the current experiments at more appropriate factor levels.
- The nature of the tank sensor made the current scenario artificial in that for the basic scenarios the MAV's were not flying at optimum height to maximize the sensor sweep width. A more complex sensor that was not a cookie cutter and had an optimum sweep width of 100 m at 50 height would be useful, especially if the nature of the sensor was such that the sweep with decreased with any change in height.
- A more realistic consideration of the communication mechanism to communicate location and sensed tank information between searchers.

THIS PAGE INTENTIONALLY LEFT BLANK

VIII. CONCLUSIONS

A. USING MARSS TO MODEL ROBOT SWARMS

The primary aim of the work reported in this thesis was to develop, implement, and test a model for investigating the behavior of swarms of robots. The construction of MARSS developed and implemented such a model. MARSS is a sophisticated simulation model-building tool that can be used by analysts to understand the contribution that individual behavioral characteristics make to group performance.

The modeling methodology described in this thesis uses ideas and technologies from agent-based simulation, discrete-event simulation, stochastic models, swarming theory, search theory, design of experiments, and statistics. No proper subset of these technologies is adequate to address the modeling questions. The application of all of these techniques to the broad problem of understanding the behavior of robot swarms was particularly effective.

The modeling of a robot swarm scenario in MARSS starts with defining the problem and understanding the system that is to be studied. Properties of agents must be defined and understood. Each agent's ability to affect its destiny by controlling its state must be defined.

The sensing process models agent interaction. The aim of the sensing model in MARSS is to transfer information about one agent's state to another. It consists of modeling the physical process involved with transferring energy through the environment. Agents emit emissions in particular bands. These emissions reside in the ether. Agents have sensors that detect emissions in the ether. Appropriate emissions are sensed if their signal strength has not been attenuated below the sensitivity of the sensor. With the sensing of an emission, information about the originator is transferred to the owner of the sensor. This information, together with some part of the agent's own state, forms the inner environment.

The behavioral process models agent actions. Functions are executed on the inner environment and the output applied to the agent's properties within their degrees of freedom. Factors control the operations of the behavioral function. This will cause the agent to act in a specific way.

The actions of many agents produce an emergent group behavior. This behavior is measured and recorded, together with the factors that produce that response. Experiments are designed to get a good spread of factor levels over the response surface. Statistics, in particular regression trees, are then used to understand what factors contributed to the response.

It would be nice to say that the modeling process described in this thesis ended there. In reality the results from this process are then used to refine the simulation model, redesign the experiment, and better understand the operation of the system being investigated. MARSS is just one tooth in the cog that helps to grind this process towards a better understanding of robot swarms. It does however represent a capability that did not previously exist.

B. MAV SCENARIO

MARSS and the associated modeling method were tested on a search scenario involving Micro Air Vehicles. In addition to testing MARSS this provided an opportunity to answer questions about the behavior of a conceptual system.

The results from basic MAV search scenarios implemented in MARSS were validated against analytical results for exhaustive and random search for a moving target. In both cases the results from MARSS matched those determined analytically. This showed that the movement and sensing models in MARSS were working as designed.

For more complex scenarios, searchers were given more involved behavior that allowed them to react to observed targets, each other, and targets observed by fellow searchers. The searchers were conducting a random search with these modifications. Summing the components of acceleration in different directions controlled movement. It was found that just by using the movement mechanism involving accelerating towards

the current way point that search performance improved over the pure random search by at least 10% regardless of the configuration of the targets.

When targets were moving in a group the most important factor affecting good search performance was acceleration away from an observed target. This was an artificial result based on the configuration of the sensor. Acceleration away from other searchers', and towards targets observed by other agents was found to have only a slight affect on performance. This was surprising, it was expected that reaction to other searchers location would produce a significant increase in performance, perhaps even approaching exhaustive search.

The research question addressed for the MAV scenario was "How should individual agents behave to produce a desired swarm behavior?" This question was answered for the MAV scenario by determining what factor levels contribute to good search performance.

Insight was provided into how the level of swarm performance is dependant on the level of communication by investigating the effects of being able to react to fellow searchers, and to targets found by fellow searchers. The results of this thesis suggest that the sharing of this information does not have a marked impact on the best swarm performance observed. Perhaps a much more interesting result is that swarm performance can be drastically reduced by reacting to that information in the wrong way.

The difference in swarm performance between distributed and centralized swarm control was addressed by comparing the exhaustive search results to the distributed control of the complex scenarios. When targets were moving in a group distributed control appeared to be much better, although this result is somewhat artificial due to the sensor configuration. When targets were spread over the search area distributed control did not achieve as well as central control. However, the increase in performance observed does suggest that central control may be possible.

C. FUTURE WORK

Many references to future work have been made throughout this thesis. Future research could focus on many areas. The MARSS model could be enhanced in almost all areas. Further work on the particular MAV scenario could involve modifications of the implementation in MARSS to more closely represent real vehicles.

The primary aim of the work reported in this thesis was to develop a process for modeling swarms of robots. Any researchers that are considering investigating groups of robots will have the MARSS tool available from http://diana.or.nps.navy.mil/~ajdickie/marss.

APPENDIX A

This appendix is a visual depiction of the MARSS simulation tool. Each figure contains a description of the functionality that is provided by that part of MARSS.

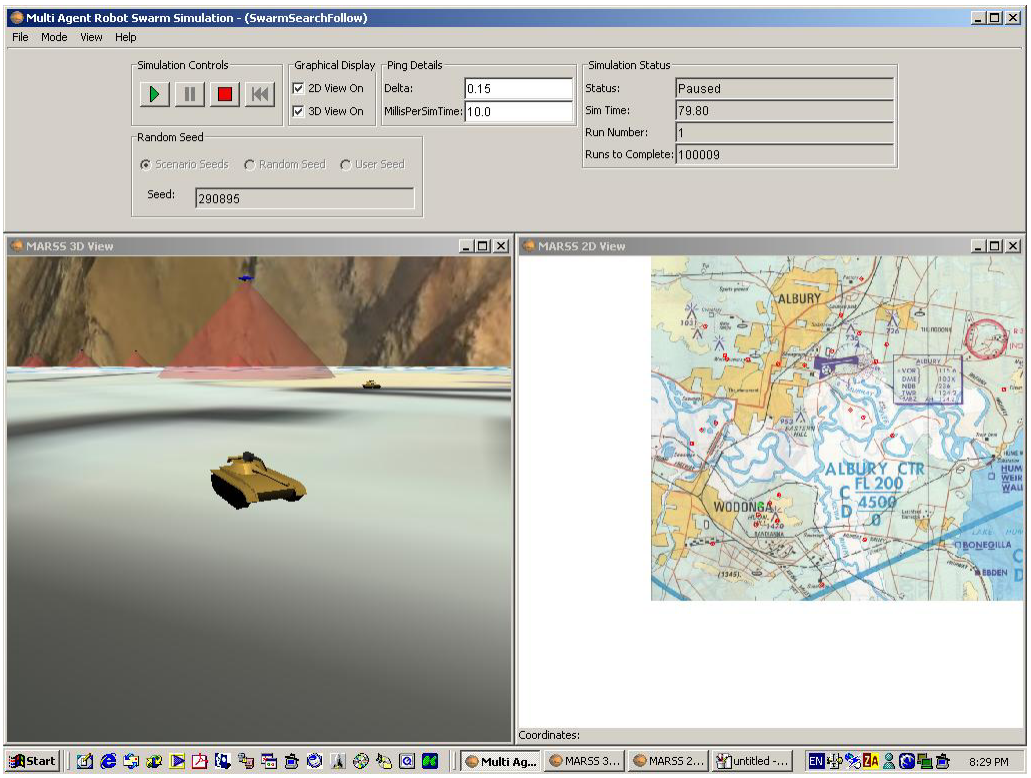

Figure 13. The Main View

This is the standard view provided by MARSS. The top panel contains controls for starting, pausing, stopping and resetting the simulation. The 2D and 3D view can be turned on and off. The "ping" delta and milliseconds per simulation time can be set. Together these controls adjust the speed of the simulation and the speed the model runs at. The simulation status shows what MARSS is doing. The random seed options give a runtime control of the seed to use when the simulation is reset between runs.

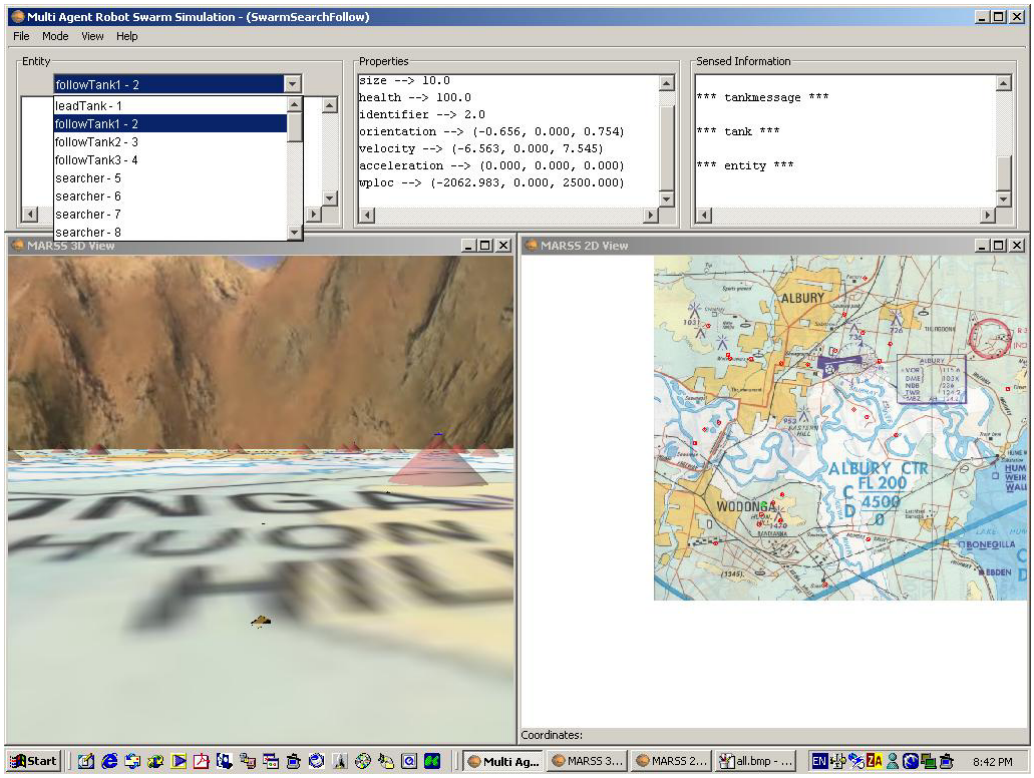

Figure 14. The State of the Simulation

This view provides information on the state of the simulation. While it can be used dynamically it is best to pause the simulation in the main view first. The value of each entity's properties can be viewed in the top center, and the information that an entity is sensing can be view in the top right. The current entity can be selected from the drop down list, by double clicking in the 2D view, or by clicking on the entity in the 3D view. Selecting an entity centers both views on that entity and shows the entity properties in the top entity panel. Just below the entity selection drop down list is a text area that displays the factor levels for that entity if they are being modeled.

Figure 15. The 3D View

The 3D view is good for visually determining state parameters such as location and lateral acceleration. A VRML file that may be associated with an entity controls the display of that entity in the 3D view. In the picture above we also see the overlay in the top left corner that provides more information about the current view. The background displayed here is the "sky" background. Backgrounds are selectable dynamically.

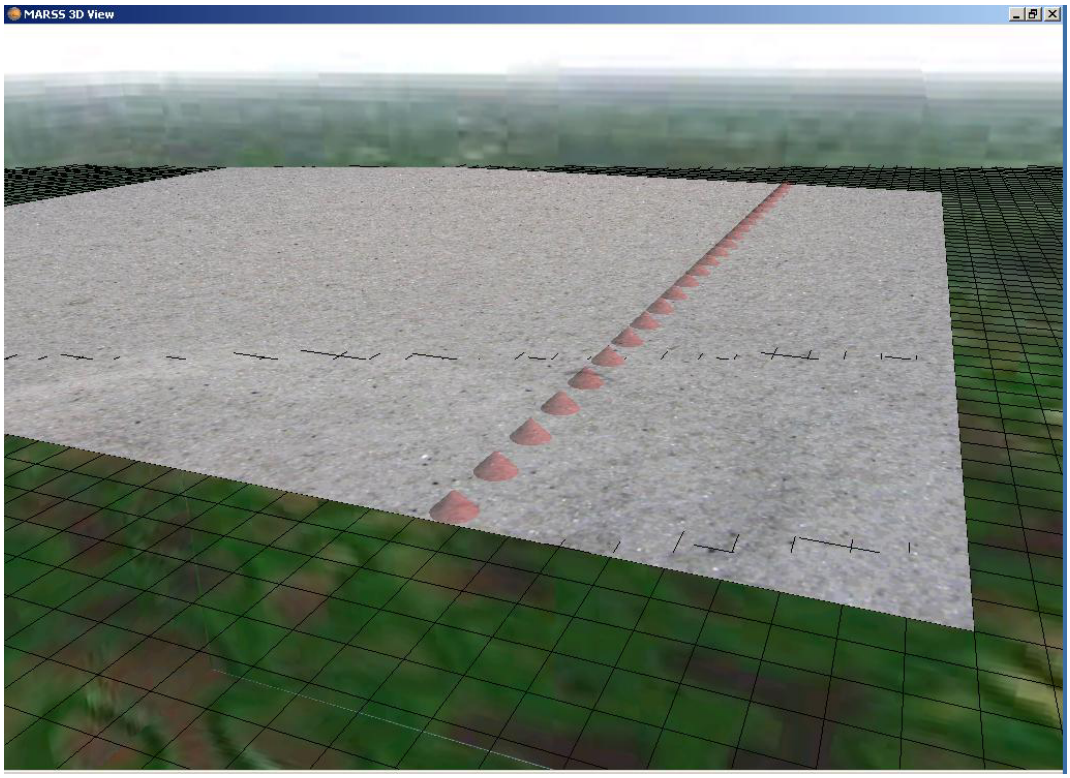

Figure 16. More 3D View

In this 3D display we see the grid lines drawn. The grid is a 100m grid at height 0 that is drawn for 100,000 m in each direction from the origin. Note the "map" displayed here is a picture of sand. Any .jpg image may be used as a map to base a scenario on. The image is drawn at height 0 in all cases.

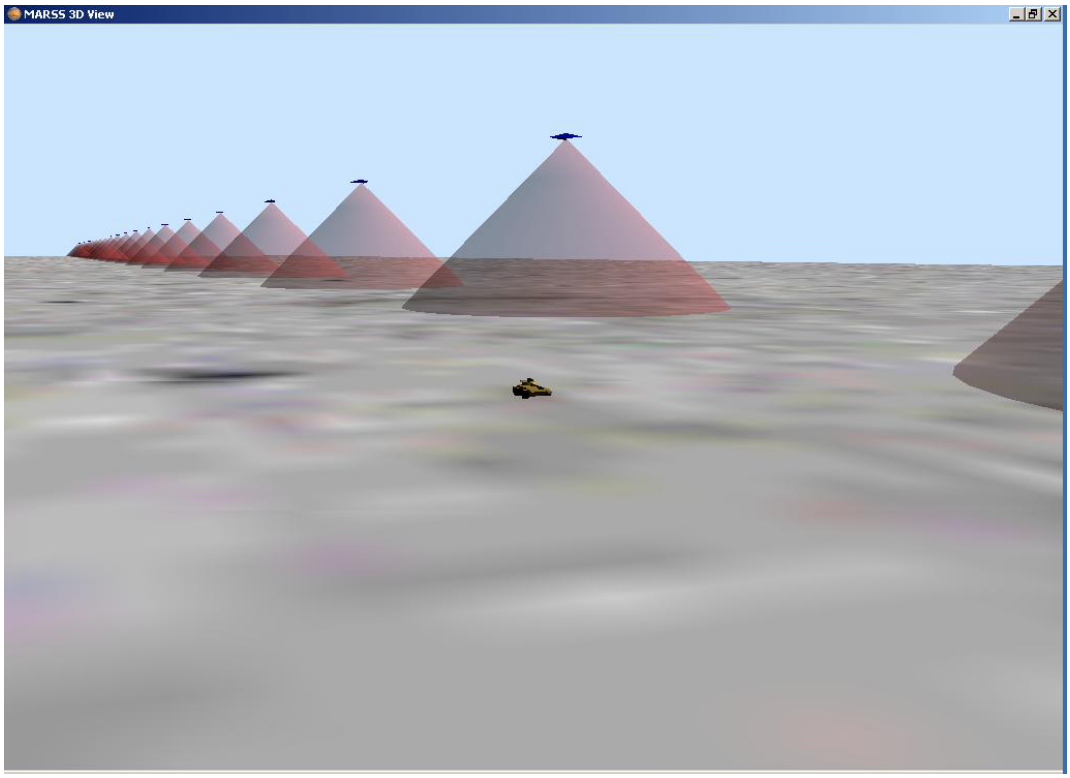

Figure 17. And More 3D View

This display shows an exhaustive search scenario with the searchers looking for tanks. The search cones are part of the VRML description of an entity that replicates effect of a sensor in the model. The 3D view can be controlled using the arrow keys and the mouse. The help menu provides more information on the function of the controls.

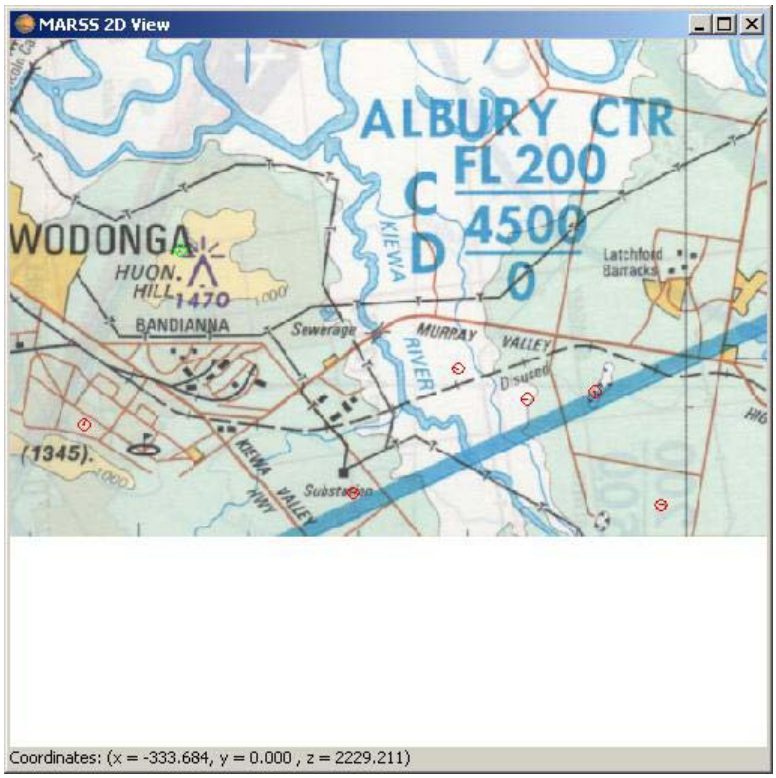

Figure 18. 2D View

The 2D displays the same map as the 3D display. All entities are drawn as a small circle with a line indicating the orientation of the entity. The currently selected entity is displayed in green with all other entities being displayed as red. Holding the mouse over the display gives the user feedback on the coordinates of that location. The display can be zoomed and translated using mouse controls. More details of the controls in this display can be found in the help.

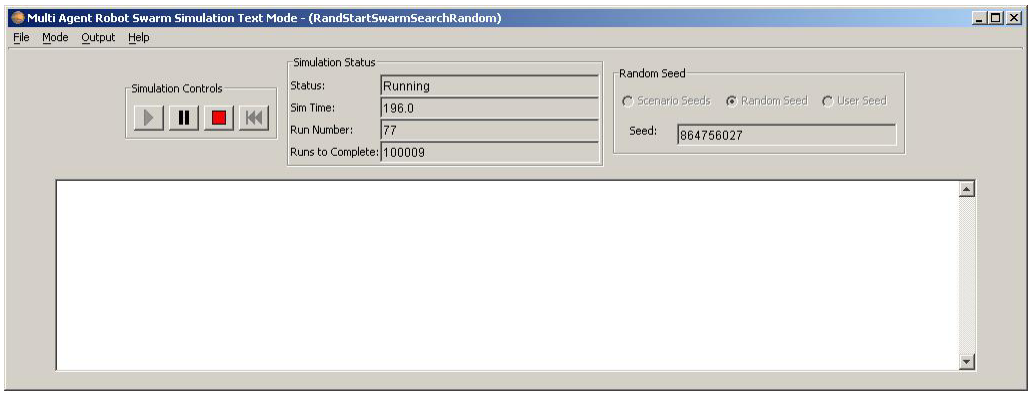

Figure 19. Text Mode

After debugging and face validating the scenario using the graphical mode the simulation can be switched to text mode for multiple runs using the mode menu. Although most output data is directed to a file, this mode can provide further text output. The same controls over the function of the model are provided in both text and graphical mode.

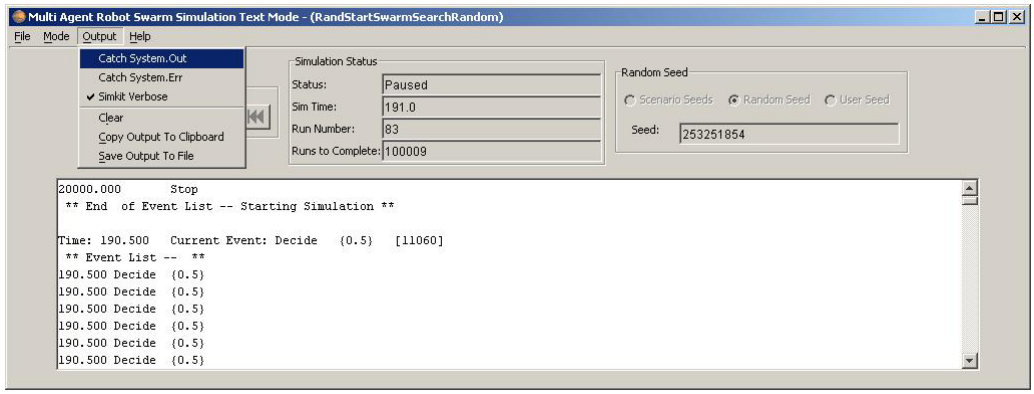

Figure 20. Text Mode Options

The output menu may be used to direct certain simulation output to the text area. This includes the verbose Simkit output. In addition to use for further debugging this mode allows the user a simple mechanism to capture output.

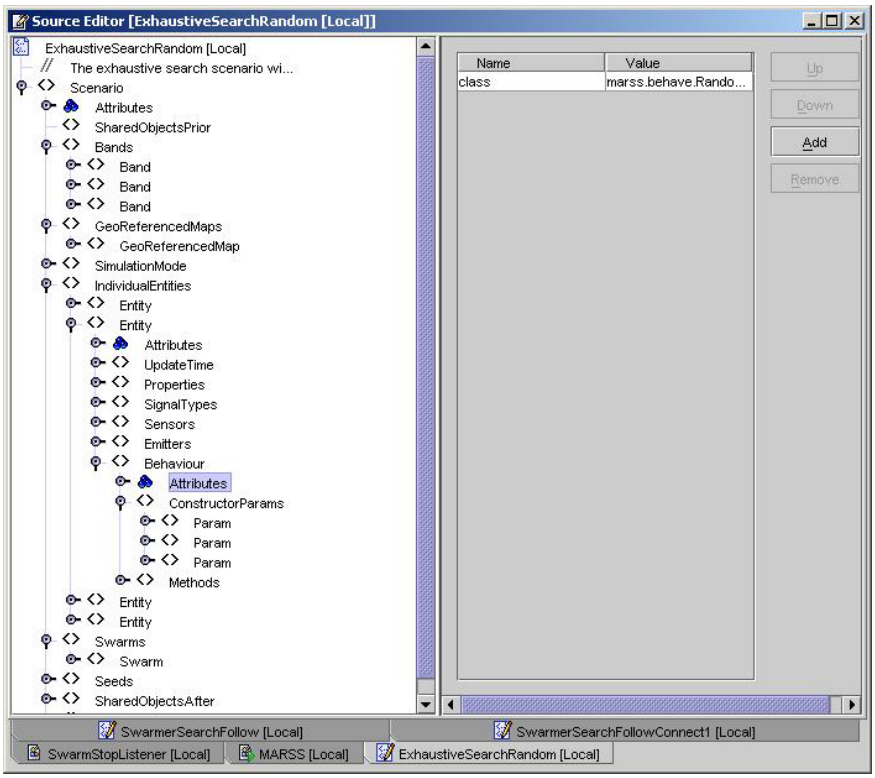

Figure 21. XML Scenario Design

Scenarios for MARSS may be fully defined in XML. This screenshot is a picture of the XML editor that is built into Sun Microsystems Forte for Java. Any XML editor may be used. The use of XML allows for easy creation of scenarios from previous ones, and the sharing of scenarios between machines. Users can define sensors, behaviors and other elements of the simulation that are user defined java classes. These classes are loaded dynamically at runtime.

APPENDIX B -UNIFORM RANDOM SEARCH OF A RECTANGULAR AREA

BACKGROUND

One assumption encountered in the development of random search theory is that the probability of a searcher being at any location in a search area is equal. [Wagner, Mylander and Sanders 1998] Similar assumptions may be made in the search for a moving target. A practical movement strategy that resulted in such a uniform distribution of location probability is required in order to compare simulation and analytical results. A rectangular area is particularity useful as such areas can be easily tiled and are commonly encountered in real searches.

Although a real search may consist of many targets and searchers, the simplified problem is to consider the case for a single mover. In this case a mover is traveling at constant velocity from waypoint to waypoint. The waypoints are to be generated using some algorithm.

AIM

The aim is to create an algorithm that generates waypoints in an area such that a mover traveling between such waypoints is equally likely to be at any point in the area at any time.

PREVIOUS WORK

The problem of creating a movement strategy to ensure uniform coverage of a region has been previously considered. Most approaches focus on a movement strategy where the mover travels from point to point on the boundary of the region. The task is to determine the reflection angle from the boundary. Lalley and Robbins considered the case for search of a circular disk [Lalley and Robbins 1989]. The article describes a procedure for uniform coverage of a circular disk. In this case a searcher moves between boundaries along chords. When the searcher must decide what point on the boundary to

head towards an angle q between 0 and 180 degrees should be generated randomly according to the following:

$$\text{Prob}(q) = 1/2 \sin(q)\, dq$$

where q measures the angle from the tangent to the boundary at the current point and the chord to be taken. Lalley and Robbins proved that using this procedure the disk would be covered uniformly. This kind of reflection is known as diffuse reflection.

Gage generalized the work of Lalley and Robbins to other convex regions[Gage1]. His effort provides a good summary of work related to this problem with appropriate references. In particular he stated that the probability of reflecting from the edge of a region in a particular direction should be proportional to the distance to the boundary in that direction. This is developed by asserting that the effort attributed to a particular area between two chords to an opposing boundary should be proportional to that area. This algorithm was implemented for a rectangular region [Gage 2]. He stated that for a circular region the diffuse reflection algorithm of Lalley and Robbins agrees with his generalization.

A SIMPLE ALGORITHM

In real search problems a more practical approach is to determine a sequence of way points rather than reflection angles. The problem is considered by assuming the searcher is located on the boundary and then determining where on the opposing three boundaries the next way point should be. According to the method proposed by Gage the probability of proceeding to a point on one of the opposing boundaries should be proportional to the distance to that point. For a rectangular region this means that the distribution of reflection angles is different from every point (although symmetric about the center of a boundary). Therefore no simple formula can be determined for the reflection angle.

Although there is no simple formula for determining the reflection angle it is nonetheless possible to determine the next waypoint using a simple algorithm. This algorithm relies on the fact that each portion of area should be transited with equal

probability. The algorithm presented can be generalized for any convex area, however its implementation is particularly efficient for a rectangular region. The algorithm is given below.

Algorithm for uniform distribution of search effort over a convex area

1. Assume that the mover is located on the boundary of the area.

2. Generate a continuous uniform random variable between 0 and the area of the region being considered.

3. Pick the next way point on the boundary such that the area of the region to the left of the chord from the searchers current location to the way point is equal to the random variable.

4. Do it again.

DISCUSSION

The discussion of this algorithm is restricted to a rectangular region, however provided the geometric calculations or integrals can be computed, it can be generalized to other convex regions. This algorithm is particularly attractive for rectangular regions since the area can be determined using basic geometry without the need for evaluating transcendental functions. Note that it does not produce an angle to reflect at, rather a way point to move towards. This is more attractive for operational searchers to use in movement management systems.

The algorithm assumes that the searcher is located on a boundary at the start, however the only requirement for continuing uniform coverage is that from the mover's current location it can move in directions that take it through every point in the region. This means that the mover may start at an internal point and move to a boundary for the first move. In fact if we want the mover's location to be uniform from time zero then we should first place the mover uniformly and move it to a boundary point. The determination of what point on the boundary to move to may be made using a similar algorithm.

The practical implementation of the area calculation in code involves a mover keeping track of which of the four edges it is on and how far it is to each of the closest

two corners along the current edge in each direction. Using this information with many conditional statements, the mover determines what edge to move to and how far to each of the corners from that edge. This detail will not be presented further.

It is intuitively easy to understand this approach when we consider the discrete case. Imagine dividing the area up into N equal area wedges from our current location. Next roll an N sided die to determine which wedge to head along. By doing this each portion of area is approximately (only because it is discrete) equally likely to be covered.

To understand the continuous version of this algorithm consider that as N approaches infinity the area of the wedges approaches zero and the area of the wedge becomes directly proportional to its length. This is the same as the result by Gage that the probability of proceeding in a particular direction is proportional to the distance to the boundary in that direction.

There are a few consequences of this algorithm. It seems that it is impossible to create a random movement strategy with internal way points and achieve uniform coverage on a per chord basis. Unless a searcher considers moving to the boundary at the end of a chord once that chord is chosen, different portions of the area under the chord will have different probabilities of being covered. This does not agree with the idea that by traveling along a chord each portion of area is equally likely to be covered.

Another consequence is that the distribution of angle of reflection is dependant on where a searcher is on the boundary. For the circular disk situation considered by Lalley and Robbins this is not the case as the region is regular. For a region of any other shape there is not one particular distribution that is appropriate. The shape of the distribution of angles must be calculated from each point on the boundary. This is not necessary in many cases (such as the rectangular region) since we can generate the random variable from the total area and then either integrate or geometrically calculate the area required to meet this random variable. This method does not require a calculation of the distribution from which to draw an angle.

A visual simulation of an implementation of this algorithm in a Java applet may be found at http://diana.or.nps.navy.mil/~ajdickie/marss/distribution. Also on this site are examples of a number of strategies that do not produce uniform coverage of the region.

CONCLUSION

A simple algorithm is presented to generate a movement strategy that ensures a mover's location is uniformly distributed over an area. The algorithm generates waypoints by generating a uniform random variable and then finding a point such that the area of a wedge is equal to the generated variable.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[Arquilla and Ronfeldt 2000] **Arquilla, J. and Ronfeldt, D.,** "*Swarming and The Future of Conflict*", RAND Corporation, 2000.

[Arthur 1994] Arthur, B., "*Inductive Reasoning and Bounded Rationality*", American Economic Association Papers 84, pp 406-411.

[Box, Hunter and Hunter 1978] Box, G. E. P., Hunter, W. G., Hunter, J. S., "*Statistics for Experimenters*", Wiley, 1978.

[Buss 2001] Buss, A., "*Discrete Event Programming with Simkit*", Simulation News Europe, 32/33, November 2001. http://diana.gl.nps.navy.mil/Simkit/.

[Franks, Sendova-Franks and Anderson 2001] Franks, N. R., Sendova-Franks, A. B., Anderson, C., "*Division of labours within teams of New World and Old World army ants*", Animal Behavior, 62, pp635-642, 2001.

[Gage 1993] Gage, D.W. *"Randomized search strategies with imperfect sensors".* Proceedings of SPIE Mobile Robots VIII, Boston, 9-10 September 1993, bol2058, pp 270-279.

[Gage 1998] Gage, D.W. *"Randomized Search Strategies"* http://www.spawar.navy.mil/robots/research/manyrobo/randomize.html, 1 December 1998

[Grasmeyer and Keennon, 2000] Grasmeyer, J. M., and Keennon, M. T., "*Development of the Black Widow Micro Air Vehicle*", AIAA Paper No. AIAA-2001-0127, Presented at the 39th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, January, 2000.

[Hiles et al 2001] Hiles, J., VanPutte, M., Osborn, B., Zyda, M., "*Innovations in Computer Generated Autonomy at the MOVES Institute*", Naval Postgraduate School Technical Report, NPS-MV-02-002, 2001.

[Horne and Leonardi 2001] Horne, G., Leonardi, M., "*Maneuver Warfare Science 2001*", Marine Corps Combat Development Command, Quantico, VA, 2001.

[Illachinski 1997] Illachinski, A., Center for Naval Analyses, "*Irreducible Semi-Autonomous Adaptive Combat(ISAAC): An Artificial Life Approach to Land Warfare*", CRM 97-61.10/August 1997.

[Lalley and Robbins 1989] **Lalley, S.P., and Robbins, H.E.** *"Uniformly Ergodic Search in a Disk".* Lecture notes in pure and applied mathematics, v. 112, Search Theory. 1989, pp 131-151.

[Reynolds 1987] Reynolds, C., *"Flocks, Herds and Schools: A Distributed Behavioural Model"*, Computer Graphics, 21(4) (SIGRAPH 87 Conference Proceedings), pp 25-34. 1987 http://www.red3d.com/cwr/papers/1987/boids.html

[Wagner, Mylander and Sanders 1998] Wagner, D. H., Mylander, W.C., and Sanders, T. J., *"Naval Operations Analysis 3rd Ed."*, Naval Institute Press, 1999.

[Weber 1995] Weber, T. R., "*An analysis of Lemmings: A swarming approach to mine countermeasures in the VSW/SZ/BZ*", Naval Postgraduate School Thesis, 1995.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California