MULTI-AGENT BASED SIMULATION OF AN UNMANNED AERIAL VEHICLES SYSTEM

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Karthiksivaram Murugesan

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

October 2011

Fargo, North Dakota

North Dakota State University
Graduate School

Title

MULTI – AGENT BASED SIMULATION OF

UNMANNED AERIAL VEHICLES SYSTEM

By

KARTHIK SIVARAM MURUGESAN

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

ABSTRACT

Murugesan, Karthiksivaram, M.S., Department of Computer Science, College of Science and Mathematics, North Dakota State University, October 2011. Multi-Agent Based Simulation of an Unmanned Aerial Vehicles System. Major Professor: Dr. Kendall Nygard.

The rapid growth of using Unmanned Aerial Vehicles (UAV) for civilian and military applications has promoted the development of research in many areas. Most of the unmanned aerial vehicles in use are manually controlled [4]. Often, UAVs require highly trained pilot operators. Hence, the main challenge faced by researchers has been to make UAVs autonomous or semiautonomous.

The goal of this research project is to develop and implement a simulation for a user-defined environment allowing UAVs to maneuver in free environments and obstacle-laden environments using Boid's algorithm of flocking with obstacle avoidance. The users are permitted to analyze the maneuvering area and coverage efficiency of the UAVs and to dynamically change environments. This project makes use of Boid's flocking algorithm to generate different kinds of movements for the flying agents, enabling the user to analyze the effectiveness of patrolling in that particular scenario.

The number of UAVs and the type of environment are set by the user. The set number of UAVs moves as a flock or swarm inside the set environment by using Boid's rules of flocking: cohesion, alignment, and separation. The coverage efficiency of the UAVs in that particular environment is reported based on the ratio between the area covered and the time when the search time reaches a threshold. The advantages and feasibilities of the approach are discussed with the simulation results.

ACKNOWLEDGEMENTS

I would like to acknowledge and express my deep gratitude to the following people who enabled and encouraged me to complete this paper successfully: my esteemed adviser, Dr. Kendall Nygard, for his continued support, help, and direction; my family and friends for their guidance and constant follow up; and everyone whose comments and suggestions aided the improvement of this project. My sincere thanks to Dr. Simone Ludwig, Dr. Tariq King, Dr. Kendall Nygard, and Dr. Limin Zhang for serving on the committee.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 - INTRODUCTION

During the past few years, the use of Unmanned Air Vehicles (UAV) has been growing tremendously for ensuring public safety and military operations. UAVs have found a place in a wide variety of applications, such as remote sensing, transport, and surveillance, as well as in other military operations, such as search and rescue. The usage of UAVs provides us a lot of advantages, such as better maneuverability, reduced cost, reduced radar signatures, longer endurance, and lower risks to crew members [1].

The growth in military applications of UAVs has been remarkable. This growth has demanded highly sophisticated manual controllers or pilots who can maneuver the UAV. This growth has led to demanding research for automating or semi-automating the UAVs across their application space [3].

Attempts to build such an autonomous or semi-autonomous UAV using artificial intelligence has often adopted a decentralized coordination of multiple UAVs using multi-agent systems [4]. This project is aimed at designing a simulation model of such an autonomous system in which UAVs can maneuver with less involvement of manual control using a multi-agent environment.

Multi-agent systems have been investigated extensively by many researchers. During the past few years, there has been a growth of interest in the potential of agent technology within the context of software engineering.

In this paper, I present a UAV simulation model based on the evolvement of multi-agent systems. This model was designed and developed using NetLogo. NetLogo is a cross-platform, multi-agent, programmable modeling environment that helps to define and model intelligent agent systems. A multi-UAV system involves searching an observation space by using UAVs flying across an environment.

In computer science, this problem is considered to be a challenging search problem because the observation space is complex, and it might have obstacles and other constraints such as time and space [9]. The area covered in a particular amount of time has been considered the prime factor in these search problems. It is very challenging because the obstacles keep changing. This simulation will offer a better way to analyze the number of pixels covered in different kinds of search spaces by applying Boid's flocking algorithm to maneuver UAVs. Finally, the simulation also provides a plot to concentrate upon the pixel coverage efficiency of the algorithm.

1.1. Problem Statement

The requirements of the system would be to fly the agents in a space and then observe their interaction with other agents [7]. This simulation would require us to set up the observation space which should be random because spaces are not the same at all places.

The foremost requirement of such a simulation system would be to simulate the environment close to real-world happenings and to still satisfy the atomicity and integrity properties of all agents.

UAV simulation is highly challenging, being in a huge and partially visible environment with more agents to be synchronized. This research also requires an analysis of the real-world scenario.

The simulation uses NetLogo, one of the premier agent simulation systems used to observe multi-agent interactions. The simulation design should enable us to

1. Set up the number of UAVs (flights).
2. Set up the environment (search space) with a collection of obstacles with various sizes and shapes.
3. Traverse the set group of UAVs across the pixels in the set environment.
4. Efficiently maneuver UAVs by avoiding the other UAVs and obstacles in the environment.
5. Mark the covered pixels of the environment and calculate the coverage percentage of the traversal (search).
6. Report the results of the simulation.

1.2. Objective

1. To study the problems in the prevailing methods of maneuvering UAVs.

2. To identify a maneuvering technique that can reduce collisions across a simulated environment.

3. To identify techniques to reduce the time taken to cover all pixels across the simulated environment.

4. To enable users to simulate an environment that is much closer to real-world environments with high buildings and other obstacles which are sometimes dynamic.

5. To study the advantages of the formulated simulation and its ability to address the problems identified using experimentation over the formulated system.

CHAPTER 2 - BACKGROUND

This chapter briefly discusses the concepts involving the multi-agent based UAV simulation system and, further, references and outlines the ongoing science and technology related to the project. The topics discussed in this section served as motivation to develop the simulation.

2.1. Introduction

UAVs have found their application in various domains. Several countries have extended research in this domain. Hence, the growth of UAV demand is tremendous and has grown at a significant rate as represented by Figure 2.1 on Department of Defense (DOD) annual funding for UAVs. The growing demand of UAVs has led to a larger demand for pilots and automation. Figure 2.2 show that UAV accidents occur at a rate several times that of commercial aircraft.

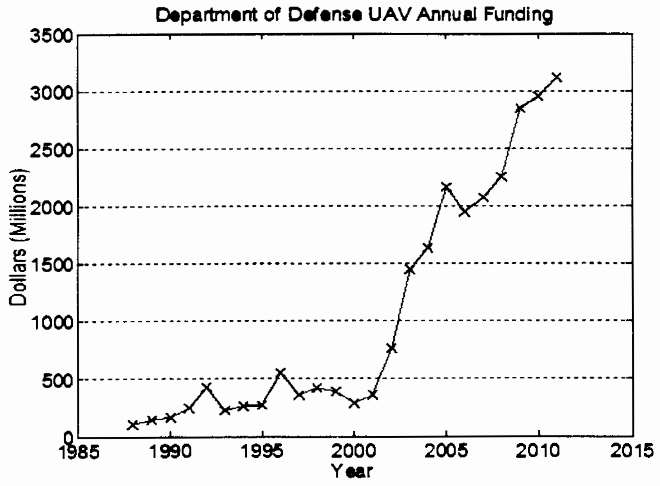


Figure 2.1: Department of Defense UAV Annual Funding [3].

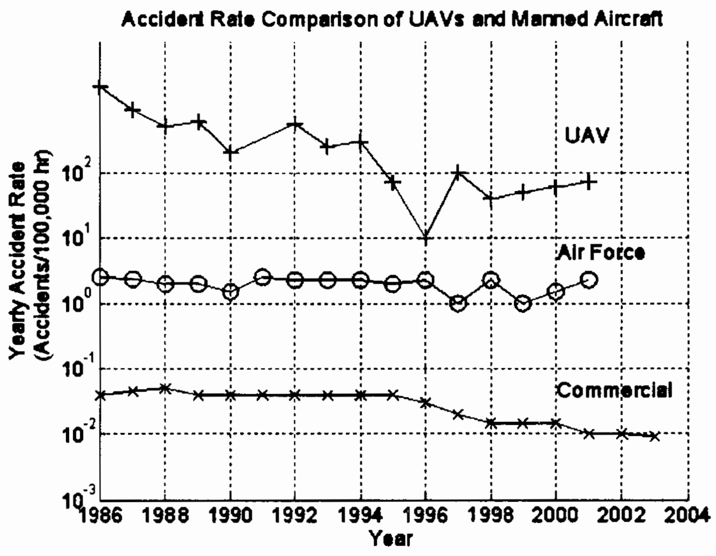


Figure 2.2: Comparison of Accident Rates [3].

The above findings have increased the demand of the research needed to formulate better maneuvering for UAVs. Usage of a manual controller across the UAV domain has introduced lot of constraints or problems for covering a particular area. Most searches resulted in collisions with obstacles, making the search performed by UAVs less reliable. In this paper, a novel simulation of UAVs is designed, developed, and tested with reference to the research introduced in this chapter.

2.2. Multi-Agent Systems

Intelligent systems are a class of multi-agent systems (MAS) that originated from the field of Distributed Artificial Intelligence (DAI) [13]. Artificial intelligence (AI) was defined by Russell and Norvig as the study and design of intelligent agents. Intelligent agents refer to software agents that can act as autonomous entities and interact with the surroundings [20]. It acts accordingly based on its learning from the environment.

$$F: P^* \dashrightarrow A \quad ,$$

Where, P* = Percept

A = Action

DAI is a term used to denote the research division of artificial intelligence that is dedicated to complex problems development of distributed solutions. Multi-agent systems originated from DAI, where the agents are autonomous and collectively use intelligence to achieve a solution for a complex problem.

An agent-based model (ABM) deals with a class of computational models for simulating the actions and interactions of autonomous agents with a view to assess their effects on the system as a whole.

2.3. Applications of a Multi-Agent System

2.3.1. Problem Solving

The MAS can serve as an alternative to centralized problem solving, either because problems are distributed or because the distribution of problem solving between different agents reveals itself to be a more efficient way to organize the problem solving. MAS will be flexible and allow failures in the system or it will be the only way to solve the problem [13].

2.3.2. Multi-Agent Simulation

Simulating MAS is widely used to enhance knowledge in biology or in the social sciences. MAS allow us to make small laboratories for testing theories about local behaviors called the artificial universes [13].

Construction of Synthetic Worlds: These artificial universes can be used to describe specific interaction mechanisms and to analyze their impact at a global level in the system.

The entities are mainly inspired by animal behaviors (hunting, searching, or gathering habits), so they are called as Animates [13].

2.3.3. Collective Robotics

Defining the robots as MAS where each subsystem has a specific goal and deals with that goal only. All the small tasks accomplished can result in solving a bigger problem. This approach can also be used in the coordination of different mobile robots in a common space [13].

2.3.4. Kinetic Program Design

MAS can also be seen as an efficient way to introduce modularity to a program [13]. Modularity makes the system design more flexible.

2.4. Swarm Approach

Swarming is a collective and emergent behavior exhibited by living organisms during movement. Some of the natural examples of swarms are an ant colony, animal herding, and birds flocking. This behavior gains interest in the area of artificial intelligence where such movements are analyzed to apply them to intelligent systems. The swarming approach is considered to be a viable way to control and coordinate the movement of robots which can act autonomously or semi-autonomously [8].

Swarming is defined to have a solution for a complex problem that can address almost all problems in emergent and self-organizing systems [12]. In computer science, one of the important examples of emergent and self-organizing systems is multi-agent systems. The simple interactions between each autonomous agent of the system can address complex patterns and problems similar to the interaction between the animals which result in an organized group solution. The coordination between smaller, autonomous agents can be addressed by swarm interactions. One such concept with applications that are yet to be explored is Stigmergy. Stigmergy helps with the coordination and communication between intelligent systems such as robotics, computer networks, and multi-agent systems [12].

2.5. Flocking

Flocking is a collective behavior exhibited by a group of birds in flight. Flocking behavior was first simulated by Craig Reynolds. Reynolds used the flocking metaphor in his seminal paper on boids in the context of computer animation [5].

2.5.1. Boid's Algorithm

In this project for the simulation of unmanned aerial vehicles, the famous Boid's Algorithm is used. The design of the algorithm involves studying and analyzing the Boids' separation, alignment, and cohesion relative with the position and velocities to other Boid's [5]. Reynolds et al. [5] showed that the flocking behavior of birds is controlled by three simple rules:

Separation: avoid crowding neighbors who fly at a particular distance

Alignment: steer towards the average heading of neighbors to achieve polarity

Cohesion: steer towards the average position of neighbors to form a group

The basic rules were identified for understanding the other behaviors of birds. The aggregate motion of the simulated flock created by Reynolds et al. [5] was a distributed behavioral model much like the one at work in a natural flock. Each simulated bird was considered an autonomous actor. These actors were navigated according to their local perception of the dynamic environment set up as an environment. Reynolds extended a simulation into an informal paper on another elaborate behavioral model of birds to avoid obstacles that were used by the earlier researchers. Following figure 2.3 shows an example flocking with cylindrical obstacles.

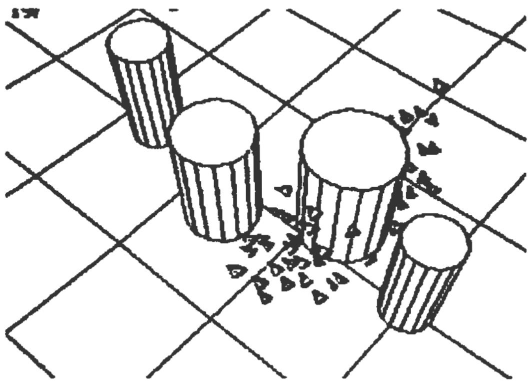


Figure 2.3: Boid's Flocking with Cylindrical Obstacles [5].

The implementation of the flocking algorithm took O (n^2) computational time. The obstacle avoidance performed by the Boid's used a 360-degree turn when an obstacle was found exactly in center of the path for a particular Boid [5].

2.6. Unmanned Aerial Vehicles

The class of aircraft with no pilot on board is generalized as unmanned aerial vehicles. UAVs can be remote-controlled aircraft or can fly semi-autonomously based on pre-programmed flight plans. One of the main challenges faced in the UAVs research space is to make them autonomous or semiautonomous because these systems require highly sophisticated controllers [16].

2.6.1. Advantages of UAVs

The military role of UAVs is growing at unprecedented rates. Some of the advantages that make UAVs viable for military roles are as follows [2]:

1. Reduced risk to human life
2. Maneuverability across varied landscapes
3. Reduced radar signatures
4. Longer endurance

2.6.2. Applications of UAVs

1. Military operations such as search and rescue, missile launch, and spy operations
2. Transport goods [6]
3. Scientific research

2.7. Problems Identified when UAVs Flock

The main constraints identified when UAVs flock are coverage, connectivity, and coordination [4]. These problems become more complex when UAVs are given autonomy. These problems can be solved with the introduction of more UAVs across the same environment. The drawback is that their high cost prohibits large-scale deployment at the current time. Hence, it is cost-effective to minimize the number of UAVs [10].

2.7.1. Approaches

This research investigates the problems and analyzes two approaches to solve them, the control system approach and the cognitive modeling approach. The first approach deals with adding a layer over the flight control system to control the mission and the second approach includes usage of the human metaphor of agency more seriously and implements an autonomous controller based on a model of human decision making widely referenced in the military command and control literature [7].

2.7.2. Flocking - Communication Problems in UAVs

This research investigates the UAV placement and navigation strategies with the end goal of improving network connectivity. Because the ground nodes can be mobile, a fixed placement strategy is either inadequate or wasteful; hence, this paper proposes the use of local flocking rules that aerial living beings, such as birds and insects, follow to address communication problems. It showed that the simulation using a flocking-based navigation strategy is adaptive to the motion of ground nodes and can, indeed, maintain high connectivity in a mobile ground network [4].

2.7.3. Cooperative Search

This research explains that a well-defined swarm, such as UAVs flocking, can distinctively enhance the sensing and detection operations of the system while minimizing the transmission of excessive control information for adaptation of the team's topology. This paper proved mathematically that such an algorithm will increase the probability of detection, minimize the expected time to detect the target and the number of UAVs employed, and can yield a better search plan [8].

2.7.4. NASA-Networked UAV Teaming Experiment

This experiment was performed by the engineers and technicians from NASA's Ames Research Center and Dryden Flight Research Center. They conducted flight tests over a "virtual" forest fire in early 2005 to evaluate new flight-control software that will allow unmanned aerial vehicles (UAV) to autonomously react to obstacles as they fly pre-programmed missions. The tests were conducted to investigate cooperative flight strategies for airborne monitoring, surveillance of natural disasters and for atmospheric sampling. Figure 2.4 shows an UAV used during networked teaming experiment.

The scientists believed that this emerging software technology may, one day, enable swarms of aircraft to move safely from one area to another as a flock or group, "stacked" in a vertical column with instruments to collect air samples for future science missions or to help ground personnel monitor forest fires and other natural disasters [17].

Figure 2.4: UAV Used in Network UAV Teaming Experiment [17]

2.8. Simulation Environments

The simulation environments which are used to counterfeit multi-agent systems are denoted as Agent-Based Modeling Systems (ABMS). There are several ABMSs used to address and model different types of agent modeling, each of which is custom suited based on the problem addressed [15].

2.8.1. Survey of Simulation Tools

This survey identified lot of agent-based modeling environments which are in use and outlined in the list as follows:

Agent Sheets, Andromeda, Any Logic, Ascape, Breve, Cormas, DEVS: Discrete Event System Specification, EcoLab, FLAME: Flexible Agent Modeling Environment, JAS: Java Agent Based Simulation Library, LSD: Laboratory for Simulation Development, MAML: Multi-Agent Modeling Language, MATSim, MASON: Multi-Agent Simulation of Neighborhoods, MASS: Multi-Agent Simulation Suite, MetaABM, MIMOSE, MobiDyc: ModelizationBaséesur les Individus pour la Dynamique des Communautés,Modelling4all, Net Logo, Open Star Logo, Repast: Recursive Porous Agent Simulation Toolkit, Repast Symphony, SimPack, SimPy, SOARS: Spot Oriented Agent Role Simulator, StarLogo, SugarScape, Swarm, VisualBots, Xholon, A-globe, ABLE: Agent Building and Learning Environment, Cougaar: Cognitive Agent Architecture, FIPA: Foundation for Physical Intelligent Agents, JAD;E: Java Agent Development Framework, Jason, MadKit, MAGSY, MASIF, SDML: Strictly Declarative Modeling Language, SeSAm: Shell for Simulated Agent Systems, SimAgent, Zeus [15]. Table 2.1 gives a survey of terminologies used in four different agent-modeling environments.

Table 2.1: Comparison of Agent-Based Modeling Systems [15]

Concept/Term	**MASON**	**NETLOGO**	**REPAST**	**SWARM**
Object that builds and controls simulation objects	Model	Observer	Model	Modelswarm
User-opened display of an agent's state	Inspector	Monitor	Probe	probe display
An agent behavior or event to be executed	Steppable	Procedure	Action	Action
Queue of events executed repeatedly	Schedule	Forever procedure	Schedule	Schedule

2.8.2. NetLogo

NetLogo, originally named StarLogoT, is a high-level platform providing a simple, powerful programming language imbibing with built-in graphical interfaces and extensive documentation. It is predominantly well adapted for the development of models involving complex systems evolving over time. One of its uses can be seen extensively for deploying models over the internet. Instructions can be given to huge number of agents operating parallel using these models. Exploring the relationship between the micro-level attributes of individuals and the macro-level patterns that result from the interaction of many individuals is possible [18].

Although NetLogo maintains the heritage of StarLogo in areas such as educational tool, its main architectural design objective is to operate easily. To reduce the programming effort greatly, NetLogo consists of high-level structures and primitives. The language inherits from Logo, a dialect of Lisp. Although it does not consist of structuring capabilities from a standard programming language, it contains most of them [18].

NetLogo is, more or less, a programmable modeling environment for simulating natural and social phenomena. It has been in continuous evolvement ever since it was made at the Center for Connected Learning and Computer-Based Modeling from its development by Uri Wilensky in 1999. NetLogo makes it possible for students to open simulations and "explore" with them, analyzing their attributes under several situations and conditions. Perhaps, it enables students, teachers, and curriculum developers to design their own models. Students and teachers can easily execute the simulations and can even build their own model using NetLogo. Researchers can use NetLogo for implementing advanced research in many fields [18]. A classroom participatory-simulation tool, HubNet, is also powered by NetLogo. By using networked computers or handheld gadgets, such as Texas Instruments graphing calculators, a student can access and control an agent in a simulation.

One of the next-generation multi-agent modeling languages that started with StarLogo is NetLogo. The functionality of our product, StarLogoT, is built, adding important new features, a redesigned language and user interface. NetLogo has been implemented on the Java virtual machine, so it is supported on all major operating systems (Mac, Windows, Linux, etc.). It can be executed as a standalone application or from a command line. Models and HubNet functions can be executed in the form of Java applets in a web browser [19].

2.8.2.1. Types of agents

Observer Agent: The observer does not have a location. You can imagine it as looking at the world of turtles, links, and patches.

Turtles: Turtles are agents that move around in the world.

Patches: The world is two dimensional and is divided into a grid of patches. Each patch is a square piece of "ground" over which turtles can move.

Links: Links are agents that connect two turtles. Links can be directed (*from* one turtle *to* another turtle) or undirected (one turtle *with* another turtle) [19].

2.8.3. Simulation Interface

The simulation interface of NetLogo is very user friendly with provisions to drag and drop controls as shown in Figure 2.5. The speed slider enables us to visualize the simulation at multiple speeds, either continuously or in ticks. The information tab contains the documented information about the model, and the procedures tab contains the code for the model.

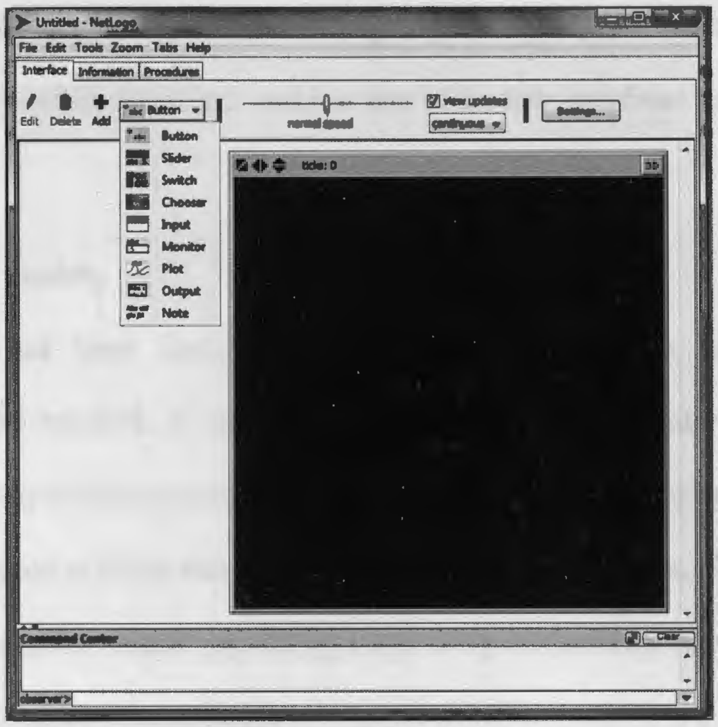


Figure 2.5: NetLogo Interface

2.8.3.1. Applications

NetLogo is apt for developing and modeling complex systems that evolve over time [5]. The developed applications can be deployed on the internet for the public to view. NetLogo defines models with mobile agents exhibiting parallel on a mesh with attributes presided by local interactions over a short period of time.

2.8.3.2. Advantages

NetLogo is considered the extensive professional platform for its documentation as well as its good look and feel. It is supported by various operating systems, such as Windows, Mac, Linux, etc., hence it is considered to be a cross-platform entity. For modeling multi-agent systems, one requires a full programmable interface, and NetLogo serves its full purpose accomplishing its features. The NetLogo platform ensures a robust and flexible plotting system. Using NetLogo, both the two-dimensional and three-dimensional views can be simulated, and it is one of the few interfaces supporting both the views [5].

2.8.3.3. Example models

NetLogo has been explored for its concepts with the help of thorough documentation and tutorials. A supplement known as Models Library is made from a collection of already-defined simulations that can be altered. NetLogo applications can be seen extensively used in fields like social sciences, biology, medicine, physics, chemistry, mathematics, economics, social psychology, and computer science. NetLogo is used for many model-based inquiry curricula [5].

CHAPTER 3 - METHODOLOGY

This chapter explains the analysis of the system, the algorithm, and some additional details of the interface. It also explains software flow of the system in detail.

3.1. Analysis

This section gives a brief description about the type of agents involved, the environment parameters, the input parameters, and the approach used for the simulation.

3.1.1. Agent Description

NetLogo is considered to be a two-dimensional agent world, consisting of different types of agents. Agents are the world entities that can follow instructions. In NetLogo, we can define four different types of agents: turtles, patches, links, and observers [5].

3.1.1.1. Turtles

Turtles are the moving agents of the NetLogo world. These agents move around the world based on their set instructions.

The moving agents of the UAV simulation will be the UAV's flying agents in the system. These agents use the reference of flying Boids which has to interact with each other to cover the NetLogo world.

3.1.1.1.1. Flying agents: UAV

These groups of turtles are defined in the NetLogo world for the UAV simulation. These agents form the flock and move across the world using the movements specified in Boid's Flocking algorithm.

3.1.1.1.2. Separation

Separation is steering the UAV away from each of the crowding local flock mates [5]. In the context of the UAV simulation, this movement will help the UAVs avoid colliding with each other. Figure 3.1 demonstrates the separation the birds implement during flight.

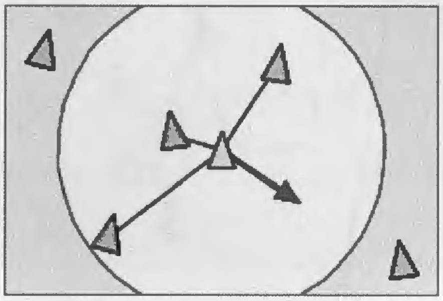


Figure 3.1: Separation [5]

3.1.1.1.3. Alignment

Alignment is steering towards the average heading of local flock mates. This movement will help a UAV to align towards a group of the other flying UAVs. Figure 3.2 shows the birds alignment during bird's flight.

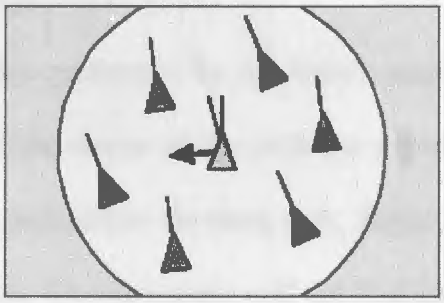


Figure 3.2: Alignment

3.1.1.1.4. Cohesion

Cohesion is steering to move towards the average position of local flock mates. This movement will help a UAV to maintain a consistent distance from other UAVs within a particular group of flights. Figure 3.3 shows the group formation by birds during usual flight.

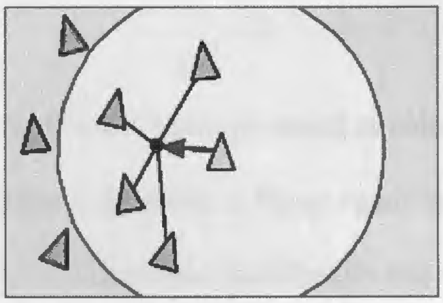


Figure 3.3: Cohesion [5]

3.1.1.1.5. Obstacle Avoidance

Obstacle avoidance is a specialized behavior of birds to steer away from static obstacles and to realign themselves to a flock. This movement finds a greater point of interest in flight simulations because there are greater possibilities of facing obstacles with flights when they fly at lower altitudes. Hence, this movement helps the UAVs to avoid the obstacles placed in the NetLogo world space.

The obstacle avoidance performed by the Boid's uses a 360-degree turn when an obstacle is found exactly in the center of the path for a particular boid [5]. An aircraft cannot do a 360-degree turn and follow the same path. Hence, the simulation is tailored to make a smooth turn. However, this movement will not find its place at higher altitudes, so this simulation will let the user set the altitude higher and lower. When a higher altitude is selected, the obstacles will not be taken into account during the flight [5].

3.1.1.2. Patches

The NetLogo world is a two-dimensional space which is divided into a grid of patches. Each patch is a square piece of "ground" over which turtles can move. In other words, we can refer to patches as the stationary agents of the NetLogo world. This class of agents will help us to define the environment where the UAVs maneuver [18].

3.1.1.2.1. Obstacles

The obstacles are defined in the NetLogo world as colored patches that are at lower altitudes. When a lower altitude is selected, a flying agent has to maneuver to avoid the obstacles in the world. The UAV simulation described in this project allows users to place the obstacles randomly or as defined by the user.

1. Random obstacles

   Obstacles are placed randomly across the NetLogo world by coloring patches at different places in the world. These obstacles are placed based on the number of obstacles specified by the observer agent.

2. Defined obstacles

   Observer agents are able to place obstacles at different places in the world by coloring the patches themselves. This obstacle will also enable the observer agent to draw a line of obstacles.

3.1.2. Input Description

This section will discuss interactions between the observer agent and the NetLogo interface. The NetLogo world is simulated based on the set parameters. These parameters are set through the NetLogo interface elements by using sliders and combo boxes. Once the agent setup and parameter setup are completed, the event handler buttons, setup and go, will allow observer agents to initiate the simulations. Input parameters for the UAV simulation consist of setting up the following parameters:

Number of UAVs (turtles): Slider that sets the number of moving flights across the world.

Number of Obstacles: Slider that sets the number of obstacles. At lower heights, obstacles illustrate entities like buildings, towers, and other objects that obstruct the normal flight path.

Height of Flight: Chooser that will have options to set lower altitudes and higher altitudes. Lower altitudes will set the behavior of avoidance to the UAVs and higher altitudes will allow UAVs to pass over an obstacle.

Placement of obstacles: Obstacles can be set to visualize the maneuverability of UAVs across the world. Clicking on the NetLogo space will create a square obstacle.

3.1.3. Approach

We identified the main problems that the system has to address:

1. The amount of time taken to cover a set environment
2. The ability of UAVs to maneuver through the set environment without colliding with each other and with obstacles set in the environment.

The ability of birds to fly without colliding while aligning towards the flock will ensure the ability of UAVs to maneuver in order to avoid collisions with each other. When obstacles are introduced into the environment, the UAVs have to maneuver without colliding with each other and also have to avoid obstacles. To avoid obstacles, UAVs have to make corresponding turns when they are in the vicinity of an obstacle. Another important parameter to consider is the UAVs' turn capability. They have to make a turn as an arc at a particular distance from the obstacle. To perform this arc, UAVs have to do a separation by an arc within some distance. Then, they would have to align and cohere to the flockmates of the nearest flock. At a higher level of flight, UAVs can be allowed to pass over obstacles because we assume the height of flight will be higher than the highest building or obstacle in the environment.

3.2. Design

The design of this model is aimed at building an autonomous or semi-autonomous model for UAV simulation with consideration of the constraints reviewed in the Approach section. Hence, the design focuses on dissipation of work from the observer agent to other agents.

The sequence diagram (Figure 3.4) and the use case diagram (Figure 3.5) emphasize the dissipation in the first three steps where the observer agent sets up the other agents and the environment. After this phase, the algorithm does the search and reports the results to the observer agent.

3.2.1. Sequence Diagram

Figure 3.4 illustrates the sequence of actions that are performed across the UAV simulation model. This sequence diagram shows the agent interactions arranged in time sequence. The observer agent performs the setup for the turtles, patches, and environment. The turtles (UAVs) then flock across the set environment and color the patches in the environment as they fly over each patch. The reporter agent gives the coverage percentage and plots the graph using the attained values.

3.2.2. Use Case Diagram

Figure 3.5 illustrates the use cases for each agent in the UAV simulation model.

This use case diagram presents a graphical overview of the functionality provided by the UAV simulation model in terms of each agent in the system.

The user or the observer agent performs the setup and initiates the simulation. The UAVs flock around the NetLogo world and make changes to the world as they cover of each of the patches. The NetLogo world and the reporting agent give results to the observer agent.

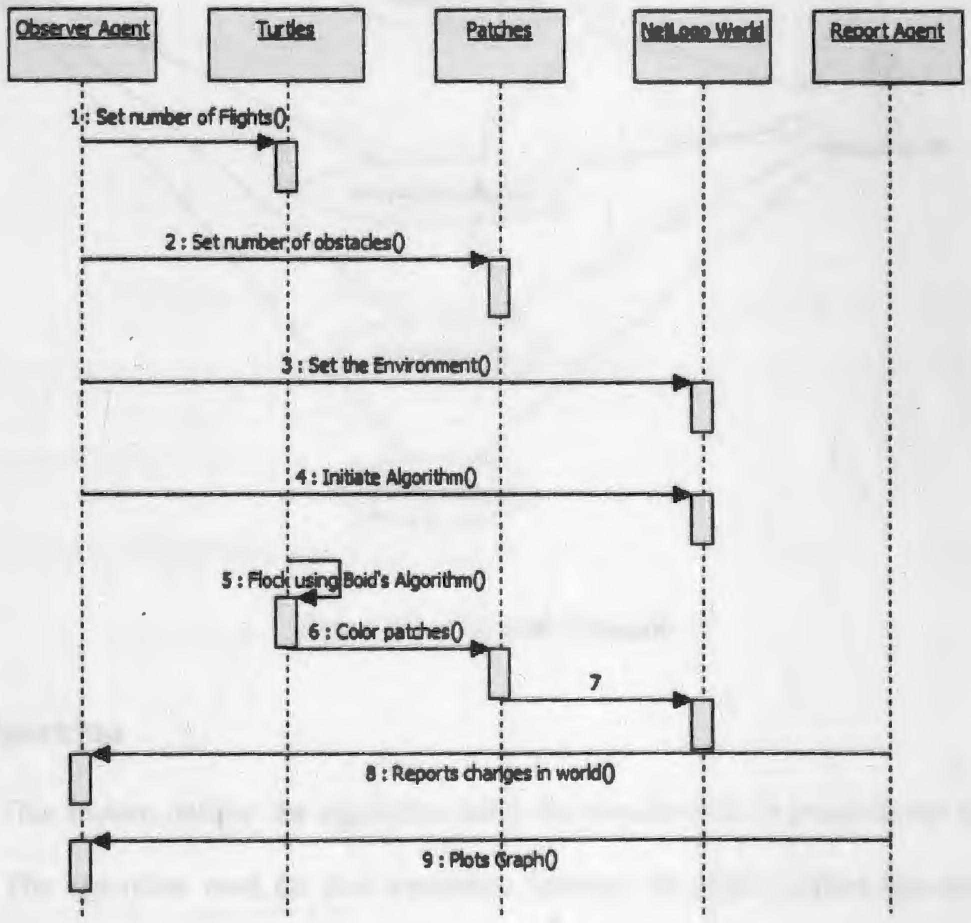

Observer Agent
Turtles
Patches
NetLogo World
Report Agent
1 : Set number of Flights()
2 : Set number of obstacles()
3 : Set the Environment()
4 : Initiate Algorithm()
5 : Flock using Boid's Algorithm()
6 : Color patches()
7
8 : Reports changes in world()
9 : Plots Graph()

Figure 3.4: Sequence Diagram

This use case diagram presents a graphical overview of the functionality provided by the UAV simulation model in terms of each agent in the system.

The user or the observer agent performs the setup and initiates the simulation. The UAVs flock around the NetLogo world and make changes to the world as they cover of each of the patches. The NetLogo world and the reporting agent give results to the observer agent.

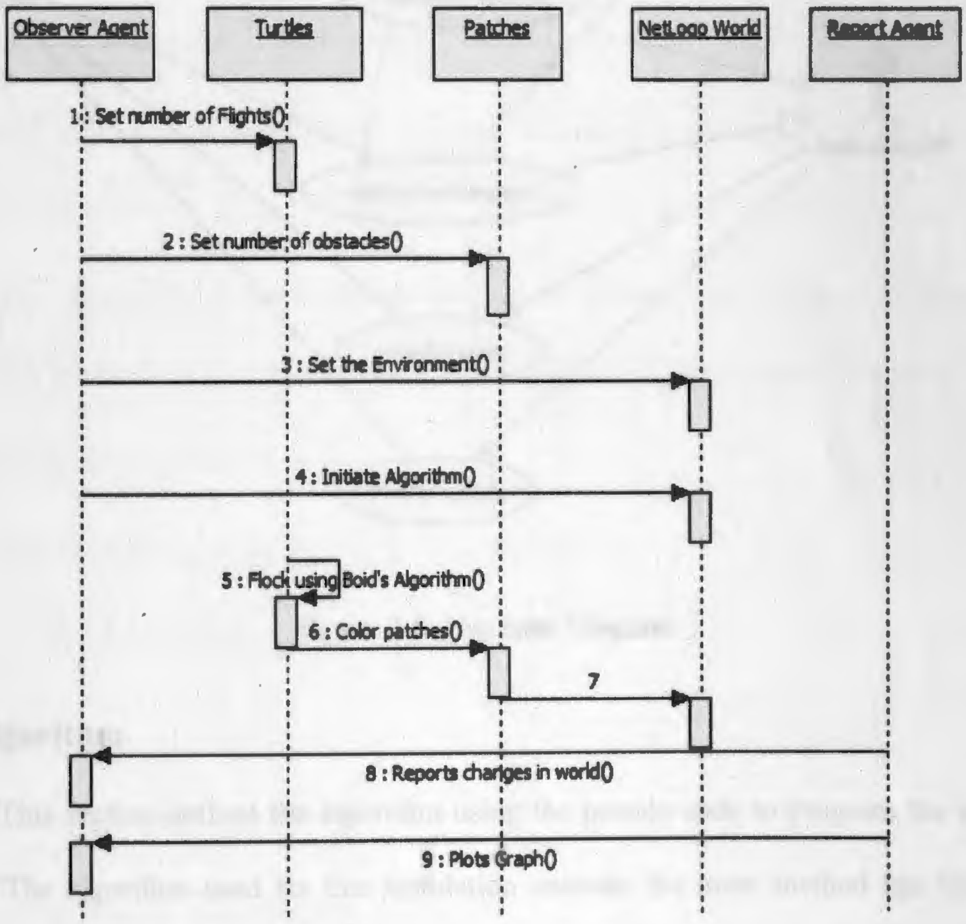

Observer Agent
Turtles
Patches
NetLogo World
Report Agent
1 : Set number of Flights()
2 : Set number of obstacles()
3 : Set the Environment()
4 : Initiate Algorithm()
5 : Flock using Boid's Algorithm()
6 : Color patches()
7
8 : Reports changes in world()
9 : Plots Graph()

Figure 3.4: Sequence Diagram

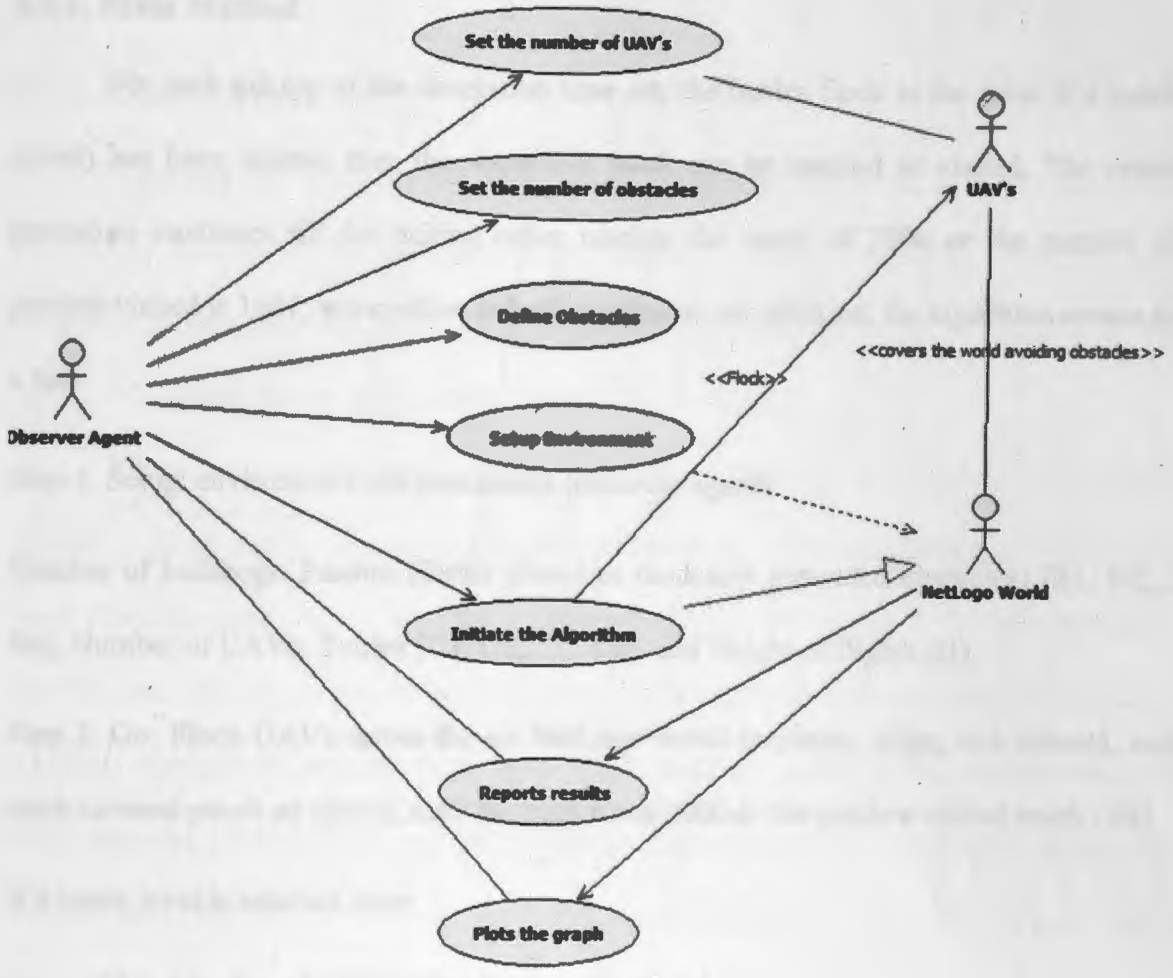


Figure 3.5: Use case Diagram

3.3. Algorithm

This section defines the algorithm using the pseudo code to program the system's agents. The algorithm used for this simulation contains the main method (go method in NetLogo) which initiates the implementation of the UAVs as a flock. The output of this algorithm is the coverage percentage calculated as the ratio between the number of pixels covered and the total number of pixels covered across the setup NetLogo world.

3.3.1. Main Method

For each ticking of the simulation time set, the turtles flock in the area. If a patch (pixel) has been visited, then the respective patch can be marked as visited. The entire procedure continues till the ticking either reaches the count of 2000 or the number of patches visited is 1681; when either or both conditions are satisfied, the algorithm comes to a halt.

```
Step 1. Setup: environment and parameters (observer agent)

Number of buildings: Patches (Either placed or randomly generated obstacles) [B1, B2,.,. Bn], Number of UAVs: Turtles [U1, U2,….., Un], and Height of flights (H)

Step 2. Go: Flock UAVs across the set NetLogo world (separate, align, and cohere), and mark covered pixels as visited, until the ticks reach 2000 or the patches visited reach 1681.

If a lower level is selected, then
    If there exists a building (Bs) in the path, then
        Avoid building (Bs)
    Else then
        Perform Go
Else if higher level is selected, then
    Perform Go

Step 3. Report: Number of pixels covered (NP) and the number of ticks (simulation time - NT) taken and coverage percentage (CP = NP / NT * 100)
```

Step 4. End

3.3.2. Flock Method

There are two modes of the selection for placing the obstacles, and they are either the random placing or the user-defined placement. With each selection, there are two categories of consideration: either lower altitudes of the objects or higher altitudes of the UAVs. We should discuss the scenario in both the categories.

When the mode is in random placement and the category is in the lower altitude, the chance of the collision between UAVs and obstacles is more. The UAVs can be turned away from the obstacles, and also, the chance of colliding with other UAVs must be taken into account. The algorithm finds other UAVs in the path and selects the nearest neighbor from the others. If the distance between the UAV and the neighbor UAVs is less than 2.00 units, then the UAVs follow any of the selection procedures, i.e., separation, alignment, and cohesion [18].

3.3.3. Procedure to Avoid Obstacles

If there is any obstacle present in the path, then UAVs follow the cone of the patch, and also, the patch is not black. The UAVs separate from the obstacle and align with the nearest neighbor. Later, they cohere as a group to form a flock. This procedure also includes the flockmates effort to avoid hitting a particular UAV. Here, all the flockmates that approaches a particular UAV with a heading angle more than 90 degrees will be avoided.

3.3.4. Procedure to Find Flockmates

The flockmate of a particular UAV is found using the Visibility parameter set by the observer agent. The formation of flock varies based on this parameter. The flockmates of a particular UAV are set as the other UAVs flying in the Visibility radius.

3.3.5. Procedure to Find Nearest Neighbor

To find the nearest neighbor of a UAV, we find the UAV with the minimum distance from the particular UAV. This procedure will return us the distance in terms of the number of patches [18].

3.3.6. Procedure to Separate

In order to separate the UAVs, the turtle procedure can be accessed. Turn away the UAV heading towards the nearest neighbor by minimum separation, set by the observer agent [18].

3.3.7. Procedure to Align

This procedure helps a particular UAV to form a flock by aligning themselves with other UAVs. The flockmates can be turned by assessing the average heading of the other flockmates [18].

3.3.8. Procedure to Cohere

This procedure will help the UAVs align towards the average heading of the flockmates. The turtles align towards an average heading with each other. After the average heading is calculated, each turtle is 180 degrees aligned to the average heading. Hence, the UAVs form a flock [18]. This algorithm has adaptations of the flocking algorithm defined by Craig Reynolds [5].

CHAPTER 4 – EXPERIMENTATION AND RESULTS

This chapter discusses how the model proposed for the UAV simulation works and carries out experiments which show the efficiency of the new research method to maneuver the UAVs. The total area of the NetLogo world, or the geographical area over which the UAVs are flown, is 1681 patches. This simulation requires us to set four parameters to perform an experiment. The experiment is performed by setting different values for the four parameters.

4.1. Methods Used

We consider the placement of obstacles in two different methods. The first method will help us simulate the static obstacles of an environment, and the second method would let the user to dynamically place obstacles in an environment.

4.1.1. Random Placement Method

This method involves random placement of obstacles in the simulation world and is performed by coloring the patches randomly across the NetLogo world. The observer agent chooses the number of obstacles from the Graphical User Interface (GUI); based on this number, the random colors are choosen from among all the colors excluding black, the world's default color.

4.1.2. Observer Placement Method

This method involves user placement of obstacles across the world by clicking over the NetLogo world. This method enables us to draw lines, walls, and other shapes to make the testing environment dynamic.

4.1.3. Altitude

Altitude defines the height of flight for the UAVs. The altitude setting is classified as higher and lower. Higher altitude is assumed to be higher than the highest obstacle in the world. Hence, the UAVs will cover the building when they pass over the obstacle.

When a lower altitude is selected, UAVs have to maneuver to make a successful coverage. Hence, UAVs will avoid the building by turning away from the obstacle.

4.1.4. Reporter Agent

The agents that report the results of a simulation run dynamically are reporter agents. This simulation includes the following reporters:

Number of Ticks: The total number of simulation time (tick) taken. This unit helps to analyze the time used for a simulation run.

Covered Area: The number of patches covered by the UAVs deployed in the NetLogo world is referred as the covered area.

Total Area: The total number of patches that are present in the NetLogo world

Building Area: The area covered by the buildings that are placed across the NetLogo world

Coverage Percentage: The percentage of patches (pixels) covered by the UAVs in the world is termed the coverage percentage.

As the UAVs move over the NetLogo world the reporters will detail the changes that occur in the environment. Based on these changes, the results can be analyzed.

4.1.5. Plot

A two-dimensional graph that lets us analyze the performance of each simulation run conducted across various inputs. The number of ticks taken is plotted across the x-axis, and the coverage percentage is plotted in the y-axis.

4.2. Results

This section includes the results of the simulation with different input values and input settings. The UAV simulation stops either when the entire NetLogo world is covered or when the number of ticks reaches 2000.

Here, all the results are obtained having an optimum visibility (6.0) and minimum separation (3.0) so that the UAV flocks as larger groups. The results are analyzed using four different settings.

4.2.1. Setting 1

In this setting, the environment is set as random, and the altitude is set as lower. This setting is a constrained setting because the UAVs have to maneuver to avoid obstacles and simultaneously coordinate with other UAVs. Because this environment is very constrained, the variation in the coverage percentage is at a higher rate.

Table 4.1 shows the results of this setting for different input values. These results show us variations that happen as we increase the constraints of the flight with more obstacles and fewer flights. Here, the variation is high because it is a more constrained setting.

4.2.2. Setting 2

In this setting, the environment is set as random, and the altitude is set as higher.

Here, the UAVs attempt to cover the entire world, including the obstacle-laden area because the altitude is higher.

Table 4.1: Results of Setting 1

Number of obstacles	**Number of UAVs**	**Altitude**	**Environment**	**Number of ticks**	**Percentage**
0	10	Lower	Random	1271	100%
1	10	Lower	Random	2000	97.501%
5	10	Lower	Random	2000	86.318%
5	5	Lower	Random	2000	85.121%
10	5	Lower	Random	2000	71.327%
10	1	Lower	Random	2000	10.648%

The coverage percentages do not vary a large extent, and the percentage always stays near 100. The last reading suggests that, even when only one UAV is set up, the environment is almost covered at this setting.

Table 4.2 shows the results of this setting for different input values. Since we have assumed that, at higher levels, the UAVs can pass over an obstacle; results remain closer to 100% coverage. The coverage percentage gradually decreases as we increase the constraints over the environment.

Table 4.2: Results of Setting 2

Number of Obstacles	Number of UAVs	Altitude	Environment	Number of ticks	Percentage
0	10	Higher	Random	1245	100%
1	10	Higher	Random	1544	100%
5	10	Higher	Random	1958	100%
5	5	Higher	Random	2000	99.286%
10	5	Higher	Random	2000	98.334%
10	1	Higher	Random	2000	93.813%

4.2.3. Setting 3

In this setting, the environment is placed by the observer agent (user), and the altitude is set as lower. This setting is a heavily constrained environment similar to Setting 1, but in this setting, the observer agent can place the obstacle at runtime or dynamically.

During the observation of this setting, obstacles were preplaced because this setting demanded a stable observation. Table 4.3 shows the results of this setting for different input values.

In this setting, the variation in results remains high because obstacles are avoided by the UAVs with the lower level of flight selected. The percentage varies from 100% to 5% as we increase the constraints. Figure 4.1 shows us the screen shot of the result shown when we place one obstacle and 10 UAV's.

Table 4.3: Results of Setting 3

Number of Obstacles	Number of UAVs	Altitude	Environment	Number of ticks	Percentage
0	10	Lower	Placed	1520	100%
1	10	Lower	Placed	2000	98.632%
5	10	Lower	Placed	2000	81.083%
5	5	Lower	Placed	2000	76.443%
10	5	Lower	Placed	2000	70.613%
10	1	Lower	Placed	2000	5.532%

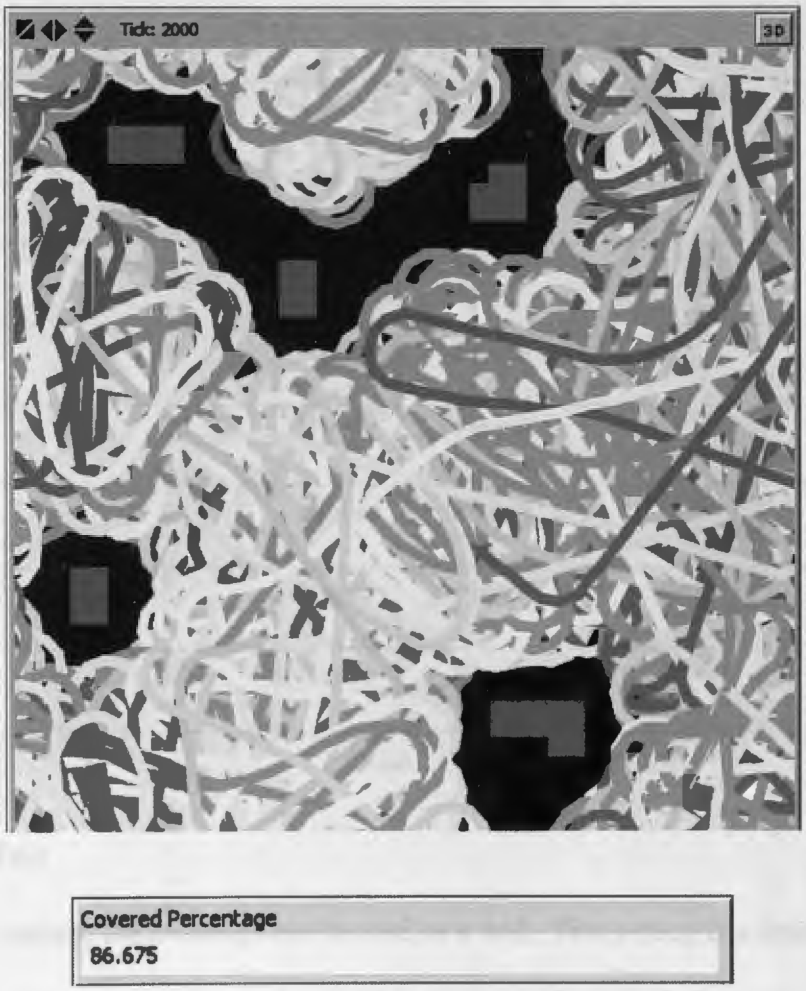
Covered Percentage
98.394

Figure 4.1: Screenshot Showing the Result for the Second Test of Setting 3.

Figure 4.2 shows the screen shot of the NetLogo world with 10 UAVs flown over an environment where 5 obstacles are placed as shown. The obstacles are placed here with different shapes to replicate a real world situation (Buildings).

On careful examination, one can see that the flocking patterns are simulated in this experiment and the coverage percentage was observed to be 86.68 in 2000 ticks (simulation time) as shown in the figure.

Figure 4.2 also shows the variation in the angle of turns of the fights when an obstacles is on its path. This angle variation is due to the adjustments a flock mate makes to preserve the formation.



Covered Percentage
86.675

Figure 4.2: Simulation Result (n=10)

4.2.4. Setting 4

In this setting, the environment is observer placed, and the altitude is set as higher. This setting is less constrained. Table 4.4 shows the results of this setting for different input values.

The results remain closer to 100% coverage because this setting is not constrained with obstacles as the UAVs can pass over. The coverage percentage gradually decreases as we increase the constraints over the environment.

Table 4.4: Results of Setting 4

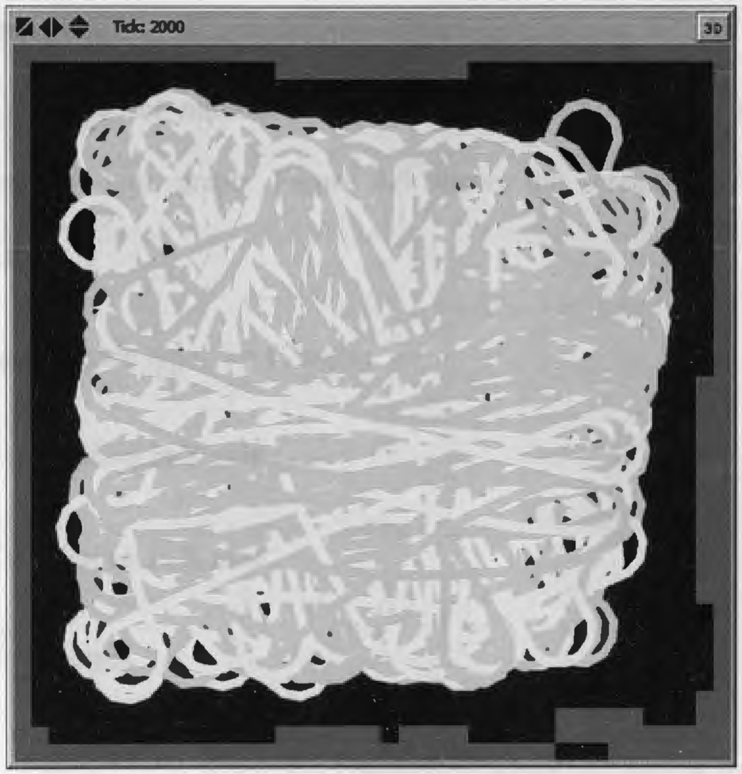
Number of Obstacles	Number of UAVs	Altitude	Environment	Number of ticks	Percentage
0	10	Higher	Placed	1498	100%
1	10	Higher	Placed	1544	100%
5	10	Higher	Placed	1958	100%
5	5	Higher	Placed	2000	99.286%
10	5	Higher	Placed	2000	98.334%
10	1	Higher	Placed	2000	93.813%

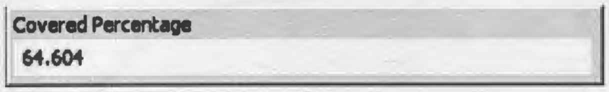
4.2.5. Wall Test

In this setting, the obstacles are formed as a wall. This setting is a constrained, and the UAVs have to maneuver themselves within the wall. Figure 4.3 shows the results of the wall test.

Figure 4.3 shows us the resulting NetLogo world after the wall test is performed. In this result, the coverage percentage was 64.604%, and the obstacle area was 208/1681, which formed approximately 13% of the entire area. If the obstacles are placed as a wall, the UAVs did not cover 22% approximately out of the designated area.

Figure 4.4 shows the plot generated during the wall test. The variation at the end is because of an increase in coverage towards the end of the simulation as the UAVs find paths that are not covered from the visited (Boolean) parameter.




Covered Percentage

64.604

Figure 4.3: Simulation Result: Wall Test.

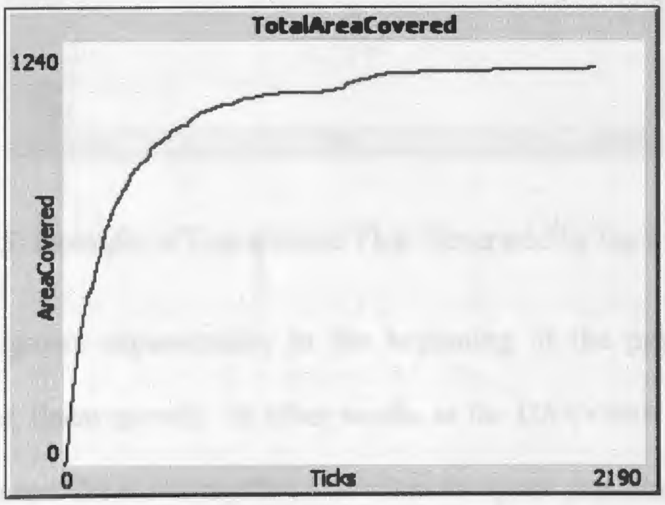


Figure 4.4: Plot Generated: Wall Test.

4.3. Plot Analysis

A graph showing the number of ticks and the area covered is plotted. This plot provides a logarithmic-type graph. Figure 4.5 shows the screenshot of plot from the simulation results.

Logarithmic type: Y= C log(x)

Y: Variation in Y axis

X: Variation in X axis

C: Constant.

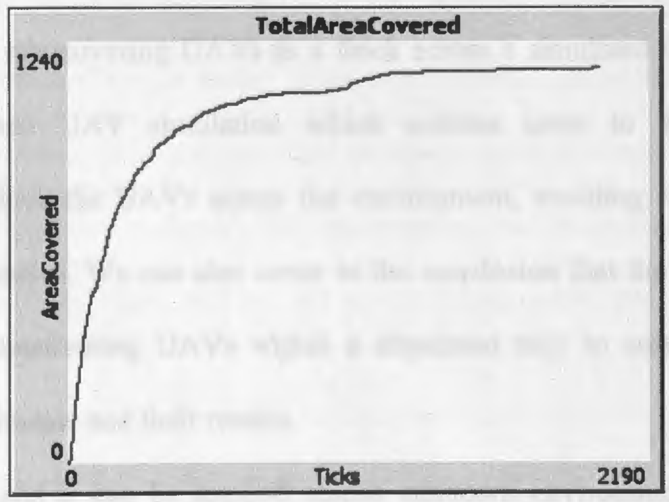


Figure 4.5: Example of Logarithmic Plot, Generated by the Simulation

This graph grows exponentially in the beginning of the process and gradually decreases to a stable, linear growth. In other words, as the UAVs start to move around the world, they would cover the world rapidly, and when the space gets limited, the UAVs start moving in the covered area.

4.3. Plot Analysis

A graph showing the number of ticks and the area covered is plotted. This plot provides a logarithmic-type graph. Figure 4.5 shows the screenshot of plot from the simulation results.

Logarithmic type: Y= C log(x)

Y: Variation in Y axis

X: Variation in X axis

C: Constant.

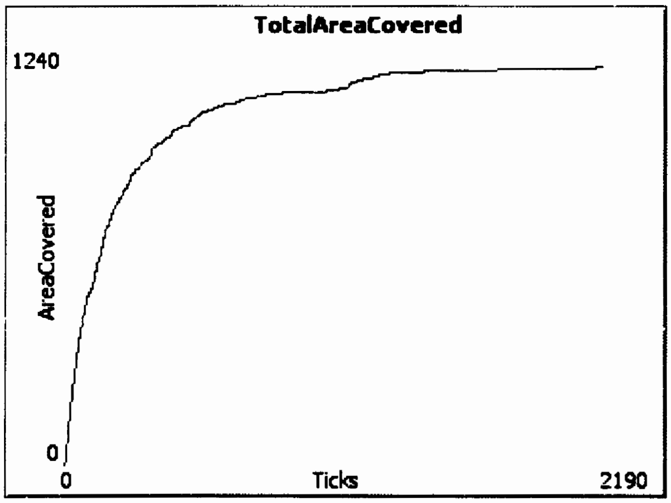


Figure 4.5: Example of Logarithmic Plot, Generated by the Simulation

This graph grows exponentially in the beginning of the process and gradually decreases to a stable, linear growth. In other words, as the UAVs start to move around the world, they would cover the world rapidly, and when the space gets limited, the UAVs start moving in the covered area.

CHAPTER 5 - CONCLUSION

A simulation framework for UAVs using multi-agent systems was designed, developed, and implemented. Now, the framework can enable users to analyze the UAVs' deployment in a dynamic environment utilizing the various settings provided by the model. This paper discussed the problems associated with the UAV simulation design and provided a solution based on the flocking algorithm to overcome these problems. The simulation also provided results that were generated based on four different settings.

5.1. Observations

Based on the results we obtained from various experiments while considering the goal of this paper, maneuvering UAVs as a flock across a simulated virtual environment, we have formulated UAV simulation which enables users to dynamically set an environment and flock the UAVs across the environment, avoiding collisions with other flights and set obstacles. We can also come to the conclusion that the Flocking algorithm serves better in maneuvering UAVs within a stipulated time to search an environment based on the experiments and their results.

Settings 1 and 3 can be applied across unknown environments; for example, a space-based UAV can maneuver across land forms on a different planet to explore the possibility of life. Settings 2 and 4 can be applied when spying across a known place to locate enemies because, in such an environment, flying at a lower altitude could be disastrous. The amount of time taken can be minimized by analyzing the results across a real-time environment in various settings.

CHAPTER 6 – FUTURE RECOMMENDATIONS

This simulation can be extended to use in some real-world environments, introducing birds and other flying entities to the environment. A geographical position system can be used for setting up the obstacles in the environment. Because the real world has lot more complexities, this simulation can be extended as real-world experiments such as experiments conducted by NASA.

There can be a reduction in simulation time by not using the same route that has already been taken. This time reduction can be done by storing the coordinates of paths taken in a database and then adding code to avoid the already-taken routes. This will improve the search methodology as well.

Further, there can be an added feature to analyze the communication between UAVs [14]. This feature can be implemented similarly to the information exchanged while sending packets through a network. The information can be routed using the nearest neighbor in the network. Thus, we can get a cumulative result generated by the agents themselves.

REFERENCES

[1] Department of Defense. "Unmanned Aircraft Systems Roadmap 2005-2030." http://www.fas.org/irp/program/collect/uav_roadmap2005.pdf, 2005.

[2] Roland E. Weibull, and John R. Hansman. "Safety Considerations for Operation of Unmanned Aerial Vehicles in the National Airspace System" ICAT-2005-1. MIT International Center for Air Transportation, Massachusetts Institute of Technology Cambridge, MA, USA 2005.

[3] James T. Hing, and Paul.Y. Oh, "A Motion Platform Integrated UAV Pilot Training and Evaluation System for Future Civilian Applications." Drexel Autonomous Systems Laboratory (DASL), Philadelphia, PA, USA 2008.

[4] Prithwish Basu, Jason Redi, and Vladimir Shurbanov. "Coordinated Flocking of UAVs for Improved Connectivity of Mobile Ground Nodes." Military Communications Conference, Monterey, CA, USA 2004.

[5] Craig W. Reynolds. "Flocks, Herds, and Schools: A Distributed Behavioral Model" SIGGRAPH Conference Proceedings. http://www.red3d.com/cwr/boids/. New York, NY, USA 1987.

[6] Zak Sarris. "Survey of UAV applications in civil markets". The 9th IEEE Mediterranean Conference on Control and Automation, MED '01, Croatia 2001.

[7] Samin Karim, and Clint Heinze. "Experiences with the Design and Implementation of an Agent based Autonomous UAV Controller". ACM AAMAS, Utrecht University, Netherlands, 2005.

[8] Patrick Vincent, and Izhak Rubin. "A Framework and Analysis for Cooperative Search Using UAV Swarms." ACM SAC, New York, NY, USA 2004.

[9] Hugo Santana, Vincent Corruble, and Bohdana Ratitch. "Multi-Agent Patrolling with Reinforcement Learning." ACM AAMAS. New York, NY, USA 2004.

[10] Jimmy Perron, Jimmy Hogan, Bernard Moulin, Jean Berger, and Micheline Belanger. "A Hybrid Approach Based on Multi-Agent Geosimulation and Reinforcement Learning to Solve a UAV Patrolling Problem." Winter Simulation Conference, Miami, FL, USA 2008.

[11] Paul De Jong. "Coalition Formation in Multi-Agent UAV Systems." B.A.Calvin College, Grand Rapids, MI, USA 1999.

[12] Gary B. Lamont. "UAV Swarm Mission Planning Development Using Evolutionary Algorithms – Part I". Air Force Institute of technology wright- Patterson AFB OH Department of Electrical and Computer Engineering. May 2008.

[13] Jacques Ferber. "Multi-Agent System: An Introduction to Distributed Artificial Intelligence." Addison Wesley Longman, 1999.

[14] Kendall E. Nygard, Dianxiang Xu, Jonathan Piklalek, and Martin Lundell. "Multi-Agent Designs for Ambient Systems." Ambi-Sys '08, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering. Quebec City, Quebec, Canada 2008.

[15] Rob Allen. "Survey of Agent Based Modeling and Simulation Tools.". Version 1.1, October 2009.

[16] "The UAV." http://www.theuav.com/ N.D. August 2011.

[17] "Networked UAV Teaming Experiment."

http://www.nasa.gov/centers/dryden/history/pastprojects/APV-3_NUAVT/index.html,

N.D. August 2011.

[18] Uri Wilensky. "NetLogo 4.1.3 User Manual." http://ccl.northwestern.edu/netlogo/, Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL. 1999.

[19] Luis R. Izquierdo. "Quick guide NetLogo 4.0." http://luis.izquierdo.name. ND. August 2011.

[20] Stuart J.Russell, and Peter Norvig. "Artificial Intelligence: A Modern Approach" (2nd ed.), Upper Saddle River, New Jersey: Prentice Hall. 2003.