APPLYING MULTI AGENT SYSTEM TO TRACK UAV MOVEMENT

by

Shulin Li

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science

Department of Computer & Information Technology

West Lafayette, Indiana

December 2019

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dedicated to my grandparents, the most significant people in my life.

ACKNOWLEDGMENTS

Study abroad is harder than I thought. Thanks to my parents (Xianjun and Yun), Professor Eric T. Matson and Ms. Lorelie Cruthers, who make it happen. I cherish the opportunities generously provided by them, so that I could learn to be a better person, learn to stand on my feet, learn to understand the kindness from people, learn to see the bigger picture and learn to leave my mark on the world.

I would like to my genuine thankfulness to Professor Anthony Smith and Professor John A Springer, who provide professional insights and advices on my research. My committee supports me all the way. I appreciate that Professor Matson, Professor Smith and Professor Springer always make their time for meeting with me. I would also express genuine gratitude to Stacy Lane and Cynthia Salazar for their generous help on every question that I asked.

Eggy, you are a special exist once in my life. You were with me on the journey of the graduate school. Thank you Chenxi for always being there for me. Thanks Eunsuh and Austin who help me with their specialties. Thanks my dear friends and labmates from M2M. I will miss the time that we study, travel and attend the conferences together. Those will be my most precious memories.

TABLE OF CONTENTS

LIST OF TABLES 7
LIST OF FIGURES 8
GLOSSARY 10
LIST OF ABBREVIATIONS 11
ABSTRACT 12
CHAPTER 1. PURPOSE AND PROBLEM 13
1.1 Background 13
1.2 Problem Statement 14
1.3 Significance 14
1.4 Purpose 15
1.5 Research Question 15
1.6 Assumptions 15
1.7 Delimitations 15
1.8 Limitations 16
1.9 Key Definitions 16
1.10 Summary 17
CHAPTER 2. REVIEW OF LITERATURE 18
2.1 Findings pertaining to the Problem & Purpose 18
2.2 Findings pertaining to the Methodology 19
2.2.1 Counter UAV technology 19
2.2.2 UAV moving route 23
2.2.3 Multi-agent system simulation 23
2.3 Studies & Proposed Methodology 26
2.4 Summary 29
CHAPTER 3. PROPOSED METHODOLOGY 30
3.1 Research Type 30
3.2 Experimental Design 30
3.3 Population & Sample 30
3.4 Agent Rules 31

3.5 Instrumentation ... 36
3.6 Variables ... 37
3.7 Data Collection ... 37
3.8 Scenario Examples ... 38
3.8.1 Types of agent clusters ... 38
3.9 Time Action Plan ... 39
3.10 Summary ... 40
CHAPTER 4. ACTUAL EXPERIMENT ... 41
4.1 Introduction ... 41
4.2 Initial Test of Proposed Methodology ... 41
4.3 Final Test Introduction ... 43
4.3.1 Why did the model change? ... 45
4.4 Modified Agent Actions ... 48
4.5 Modified Scenarios Breakdown ... 52
4.5.1 Track with Two Stations ... 53
4.5.2 Track with Three stations ... 58
4.6 Experiment Results ... 59
4.6.1 Expected Results ... 59
4.6.1.1 Simulation 1 ... 59
4.6.1.2 Simulation 2 ... 61
4.6.1.3 Simulation 3 ... 62
4.6.2 Overall Performance ... 64
4.6.3 Unusual Results ... 64
CHAPTER 5. CONCLUSION AND FUTURE WORKS ... 67
5.1 Conclusion ... 67
5.2 Future Works ... 67
REFERENCES ... 69
APPENDIX A. ALGORITHM OF DETECTION BY THREE STATIONS ... 74

LIST OF TABLES

Table 1. Evaluation on different ABM simulators 26
Table 2. Variables in the experiment 37
Table 3. Observed and calculated data list 38
Table 4. Algorithm of detection by two stations 56
Table 5. Variables setup 59
Table 6. Simulation 1 result 59
Table 7. Simulation 1 data 60
Table 8. Simulation 2 result 61
Table 9. Simulation 2 data 61
Table 10. Simulation 3 result 62
Table 11. Simulation 3 data 63
Table 12. Average performances 64
Table 13. Flaw insider 64
Table 14. Parameters when unexpected results showed 65
Table 15. Observation of different range of detect_radius 65
Table 16. Algorithm of detection by three stations 74

LIST OF FIGURES

Figure 1. The Number of UAVs in USA 13
Figure 2. Park et al. low-cost radar system hardware 20
Figure 3. Matson et al. acoustic detection node hardware 20
Figure 4. Wuhan police testing the UAV gun 22
Figure 5. Dutch company taking down UAVs using trained eagle (Castle, 2016) 22
Figure 6. Agents' actions in a sequence diagram (Kim & Matson, 2016) 25
Figure 7. NetLogo GUI (Kim & Matson, 2016) 27
Figure 8. Simple agent logic (Atmaram, 2006) 28
Figure 9. Intelligent learning agent logic (Atmaram, 2006) 28
Figure 10. Circle agent cluster detecting UAV scenario 33
Figure 11. Agent 1, 2, 16 detecting UAV 33
Figure 12. Green cross marks where UAV is at 34
Figure 13. Blue cross marks where UAV is at 34
Figure 14. Agent 2, 3, 4 detecting UAV 35
Figure 15. Purple cross marks where UAV is at 35
Figure 16. Circle agent cluster drawing UAV route 36
Figure 17. Rectangle agent cluster waiting for UAVs 39
Figure 18. Circle agent cluster waiting for UAVs 39
Figure 19. Agile methodology of developing 40
Figure 20. Initial test setup 41
Figure 21. Normal two marks intersections 42
Figure 22. Abnormal two marks intersections 43
Figure 23. Final test setup 44
Figure 24. Final test running 45
Figure 25. Proposed rectangle agent cluster 46
Figure 26. Proposed circle agent cluster 46
Figure 27. Proposed model, long-distance sensors required 47
Figure 28. Final model, short-distance sensors 47
Figure 29. Grey area, the detection range of the station 49

Figure 30. Two stations detecting the UAV 49
Figure 31. Red cross drew at the intersection of two circles 50
Figure 32. The UAV track drew by the system 51
Figure 33. Modified sequence diagram 52
Figure 34. 4*4 Agent cluster with 1*1 marked.. 53
Figure 35. Normal two marks intersections... 53
Figure 36. Two vertical marks middle point... 54
Figure 37. Horizontal two marks middle point.. 55
Figure 38. Three marks intersections.. 58
Figure 39. Three marks out of four stations.. 58
Figure 40. Radius 20, one jumpy cross.. 66
Figure 41. Radius 25, many jumpy crosses 66

GLOSSARY

Unmanned Aerial Vehicle - Unmanned Aerial Vehicle is usually referred to as a class I UAV, which is powered, can fly autonomously or be piloted and can carry some payload (Department of Defense, 2011).

Counter UAV System - A system which detects and/or captures or attacks the UAVs (Michel, 2018).

Multi-agent System - A group of software agents interacting with each other to solve problems based on everyone's knowledge (Sycara, 2012).

Agent - According to Shoham, the agent is an entity, which continuously performs in a certain environment without constant human instruction and intervention (Shoham, 1993). In the research, the agents are a group of entities which share the similar characteristics and behaviors. When they face same situations, they are supposed to perform similar behaviors.

Node - According to Matson et al., an acoustic node receives and analyzes the UAV's sound information using a soundcard and the machine learning model (Matson et al., 2019). The node in this study has a more general meaning. It is a unit or a station detecting UAVs by sensors and data-analysis techniques. The node includes but is not limited to the acoustic sensing nodes.

LIST OF ABBREVIATIONS

UAV	Unmanned Aerial Vehicles
CUAS	Counter Unmanned Aerial System
FAA	Federal Aviation Administration
MAS	Multi-agent System
ABM	Agent-based modeling
ROS	Robot Operating System
AI	Artificial Intelligence
m	meter, a measuring unit for length
s	second, a measuring unit for time
cor	Coordinate

ABSTRACT

The thesis introduces an innovative UAV detection system in five chapters. The first chapter introduces the current UAV market situation, the importance of the Counter UAV system (CUAS) and the reason to improve UAV detection. The commercial UAV market is booming. Meanwhile, the risks and threats from improper UAV usages are also booming. Although the government updates policies, there are other ways like CUAS to protect the public and facilities. The problem is a lack of an intelligent platform which can adapt many sensors for UAV detection. The hypothesis is that, the system can track the UAV’s movement by applying the multi-agent system (MAS) to UAV detection and track. The second chapter includes the literature reviews of the problems and multiple CUAS techniques. There are sensors for UAV detection and techniques for UAV interception. The following chapter proposes the investigation method overview, discussing the experiment design and the agent rules. The methods are inspired by the reviewed studies in the previous chapter. The last two chapter state the modifications for the final simulation. The processes, the scenarios, the logic flows and the result statistics of the experiment are explained. The experiment proves that the multi-agent system benefits the UAV track. At last, there is a brief discussion on conclusion and the future work of the project.

CHAPTER 1. PURPOSE AND PROBLEM

1.1 Background

Unmanned Aerial Vehicles, or UAVs, are usually small and fast quadcopters aircrafts controlled by the operator. Originally, UAVs were invented for military purposes. Due to the affordability and accessibility of a commercial UAV, the UAV market is emerging in the decade. More and more people purchase UAVs for leisure activities, such as photo taking, video filming, and so on. Besides the recreational purposes, UAVs can be used for academic research, surveillance and data collection. The non-military usage of UAV is the key to the emergence of the UAV market. By 2017, there are approximately 1.1 million UAVs in the United States. The UAV's quantity is expected to reach 2 million at the end of 2019 (FAA, 2019), which is near double quantity compared the quantity in 2017, shown in Figure 1 (Philly by Air, 2018). By 2025, the revenue of the UAV industry will grow to $82 billion. The military UAV market may reach $13 billion by that time (AP News, 2018), while most of the market belongs to the commercial UAV market (Jenkins & Vasigh, 2018).

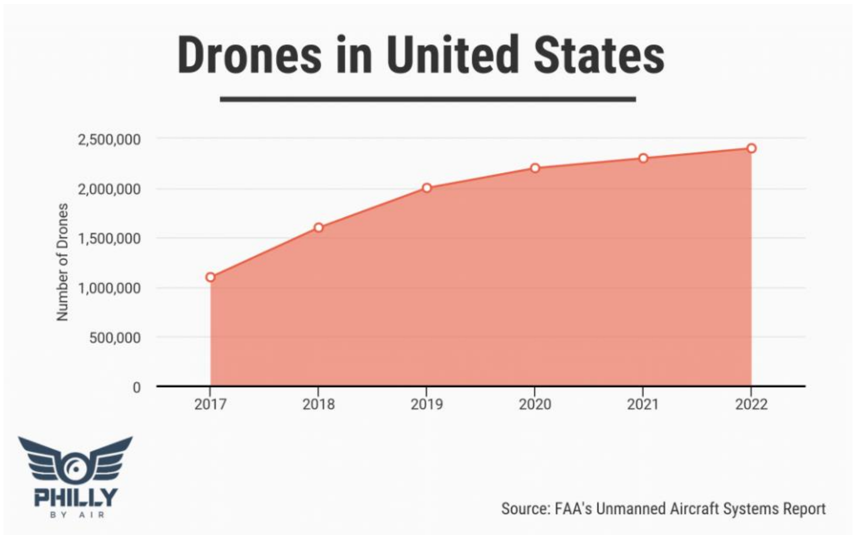


Figure 1. The Number of UAVs in USA

There are always risks in an emerging industry. UAVs are small, fast and hard to detect. Those characteristics causes the issues for illegal potentials. Some people take advantage of the

UAV camera to invade others' privacies (Christian, 2019) (Reitmeyer, 2019). Some criminals use UAVs to smuggle drugs (McVicker, 2015) (Stelter, 2016). Some intend to load the explosive weapons threatening public safety (Blodgett, 2019). Besides being used for the malicious purposes, there are accidents that the recreational UAVs flew in sensitive facilities. In 2015, a DJI Phantom commercial UAV crashed on the lawn of the White House yard due to the operator's mistakes (Hennigan, 2018). The terrifying thing was that the DJI Phantom UAV would not be noticed easily by the security. In 2017, a pilot spotted a UAV outside the plane when the plane was landing at Chicago O'Hare International Airport (Hennigan, 2018). Those accidents brought up public's concerns on UAV, making it clear that a solution, which can be the counter UAV technology, must be developed.

1.2 Problem Statement

The problem addressed by the proposed study is the lack of a multi-agent counter UAV detection system to track the UAV movement and detect a UAV's route.

1.3 Significance

The FAA announced the regulation of UAV ownership and the UAV-flying certificate in 2015; however, it did not turn out well as expected. By 2017, there is only 40% of the UAV owners who followed the registration and examination processes (FAA, 2019). Further, the local laws also address on the forbidden UAV-flying areas. Although the law is trying to keep up with the UAV industry, there is a long way to fully regulate proper usages of UAVs. Before the law catching up with the industry, the gap always remains. Even the gap will decrease a little every year, it stays remaining for those years. If the gap between the law and the UAV operation exists, the illegal handling will continue to exist. To prevent malicious usage of UAV, the Counter UAV system (CUAS) becomes very important to fill the gap.

Currently, CUAS is not systematic nor standardized. Every aspect of CUAS needs to develop further. Considering the situation, it is worth to try for new solutions, which is applying the multi-agent artificial intelligence system on UAV detection. The combination may bring new possibilities and specifications.

1.4 Purpose

The purpose is to test the hypothesis that by applying MAS to UAV detection, the system will track the UAV movements and routes. To fulfill the test, the deliverables will be a development of the MAS framework and a simulation and the evaluation of the system performance.

1.5 Research Question

Can MAS-assisted Counter UAV Detection system track the suspicious UAV movement?

1.6 Assumptions

- When the UAV flies, it has constant weight, height and other physical conditions.
- The UAV model used in the simulate represents the general commercial UAVs which are small, round, fast. For example, it can be the DJI Phantom.
- The UAV can be detected by the proximity sensors if the UAV is in the detection range.
- The system can adapt with different sensors. To fulfill the current study goal, only the proximity sensors are considered in the study.
- The agent sensor detection range covers all the detection area. There is no detection gap in the detection area.
- The network of the simulation will be pre-set to mimic the realistic environment.
- The stations are small, so the size of stations will not be deducted from the distance to the UAV.

1.7 Delimitations

- The research focuses on improving the functional level of the UAV detection module in a CUAS. The experiments will be designed as testing the performance of the system, not comparing between the simulated system performance and the realistic system performance.
- The simulation is a 2 Dimension (2D) model. Only x and y coordinates are considered.

- The simulation provides an ideal environment (no wind, no controlling mistakes, etc.) to collect data.
- In the simulation, only proximity sensors will be used for measuring the distance.

1.8 Limitations

- The limitation of simulating MAS will heavily depend on the simulating software, NetLogo. For example, if the software is more compatible with the programming package extension, it can realize more functions. The limitations will be documented along with the development stage.

1.9 Key Definitions

- Unmanned Aerial Vehicle – Unmanned Aerial Vehicle is usually referred to as a class I UAV, which is powered, can fly autonomously or be piloted and can carry some payload (Department of Defense, 2011).
- Counter UAV System – a system which detects and/or captures or attacks the UAVs (Michel, 2018).
- Multi-agent System – A group of software agents interacting with each other to solve problems based on everyone's knowledge (Sycara, 2012).
- Agent – According to Shoham, the agent is an entity, which continuously performs in a certain environment without constant human instruction and intervention (Shoham, 1993). In the research, the agents are a group of entities which share the similar characteristics and behaviors. When they face same situations, they are supposed to perform similar behaviors.
- Node – According to Matson et al., an acoustic node receives and analyzes the UAV's sound information using a soundcard and the machine learning model (Matson et al., 2019). The node in this study has a more general meaning. It is a unit or a station detecting UAVs by sensors and data-analysis techniques. The node includes but is not limited to the acoustic sensing nodes.

1.10 Summary

The first chapter introduces the current UAV market environment. As mentioned before, the commercial UAV market is growing fast, while the laws and regulations take longer to catch up with the market. Many people do not follow the rules and regulations. Some people even use the UAV for purposeful, illegal behaviors. To protect a facility, the research proposes to improve CUAS detection intelligence level by applying MAS. It is a new approach which is challenging and worth to investigate.

CHAPTER 2. REVIEW OF LITERATURE

This chapter reviews the relevant literatures on the topics such as the existing problems, counter UAV technologies, UAV movement and multi-agent system simulation.

2.1 Findings pertaining to the Problem & Purpose

The commercial UAVs are applied to recreational usages, research purposes as well as illegal acts. There is a lack of complete regulations on UAV usage, as well as standardized commercial UAV defense technology (Michel, 2018). According to Peacock, "…methods of generating mitigations related to specific protocol controlled small UAVs, such as those running on radio frequency would expand the body of knowledge, as there is currently little identified research exploring this approach" (Peacock, 2014, p.70). Notably, "…world governments realized that there is a lack of security protocols in place for such small, yet potentially dangerous, threats." (Goppert et al., 2017, p.1). When the detection involves many nodes, sensors, processors, and human commanders, system integration becomes especially important. So far, a counter UAV detection program usually focuses on one or more sensors. For example, a system may use radar, camera, or audio sensors. However, applying a single kind of sensor may have limitations. For instance, according to a current study, the listening nodes with acoustic sensors cannot make the decision individually. "Machine learning and rule-based methods can be combined to find which node is closer to the UAV. It is hard to find where the UAV is when louder UAV approaches and multiple nodes detect it" (Matson et al., 2019, p.71). The nodes send the recorded data to a central server, then the server detects if there is a UAV or not (Matson et al., 2019). In this case, the nodes collect information separately. If the nodes were able to talk to each other instead of sending data to a central server for processing, the communication efficiency may be improved. If the system is able to engage in communication among each node, then it would be possible to develop a sensor-fusion standardized intelligent UAV detection system. To enhance the integration and efficiency of detection, agent-oriented programming will be applied to the nodes to perform the MAS features.

Developing CUAS is significant to protecting the public from the malicious usages of UAVs. According to Lim, "…detecting and categorizing UAVs has become and will continue to become extremely important as terrorists find new ways to exploit emerging technologies" (Lim, 2018, p.242). This research proposes a new approach for UAV detection, which is the crucial phase to successfully operate the UAV measurements (Matson et al., 2019).

2.2 Findings pertaining to the Methodology

The section reviews related technologies, such as CUAS, UAV prediction, MAS and simulator evaluation. The strengths, weaknesses and potentials will be mentioned as well.

2.2.1 Counter UAV technology

Counter UAV technology originally was developed for military usage in this century. At that time, the military used a ground-based defense system to detect the low-altitude flying, small and slow unmanned aerial vehicles (Michel, 2018). After commercial UAVs launching to the markets, people realize the danger of the UAVs. Counter UAV technology is the common name for the UAV detection, tracking, and interdiction technologies. CUAS contains one or more counter UAV technologies to protect the target from UAV harassment and attack. Generally, there are three kinds of CUAS, for detection only, for interdiction only, or both detection and interdiction (Michel, 2018). Detection-only CUAS and interdiction-only CUAS work towards to the same purpose, which is to defend against the UAV invasion. The third kind of CUAS system is highly comprehensive. It is challenging to make all functions work well. The existing detection solutions and some interdiction solutions will be introduced following.

Radar is one of the most known approaches to detect UAVs. For military purposes, radar is applied to detect flying vehicles. However, it is expensive in cost and the detection mechanism limits radar application to UAV detection. The radar receives the RF pulses from the suspicious vehicle, so that radar is the expert in detecting large and angular objects. Commercial UAVs, conversely, are small and round. The shape of UAV increases the difficulty in distinguishing it from other small flying objects like birds. Park et al., scholars from Purdue University, designed a low-cost radar (Figure 2). which can detect the small flying vehicles in a short range around 10 meters (2015). However, averagely the UAV speed is 5 m/s. When a UAV carries explosive weapons, if the detection range is a circle area with a 10 meters radius surrounding the protected facility, it only takes 2 seconds for the UAV arriving the facility and for the system to detect and take down the UAV. In this case, it is better to save more time for the system to conduct the detection and the attack moves, indicating that the detection area radius should be larger than 10 meters.

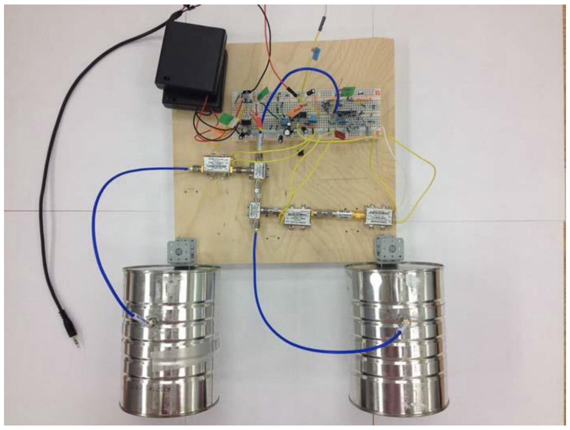


Figure 2. Park et al. low-cost radar system hardware

Acoustic detection (Figure 3) is a low-cost UAV detection solution. This approach requires a database of the UAV motor sound data. A microphone records the UAV sound, then the system will match the features in the database. The limitation is that, because different UAV models sound differently, one database only serves for one type of UAV (Li, 2018). Further, the sound signature changes when the UAV gets loaded. As a result, for any type of UAV, if the UAV load changes, the database needs to change too. The other challenge is to distinguish the motor sound with the background sound and noise (Matson et al., 2019).

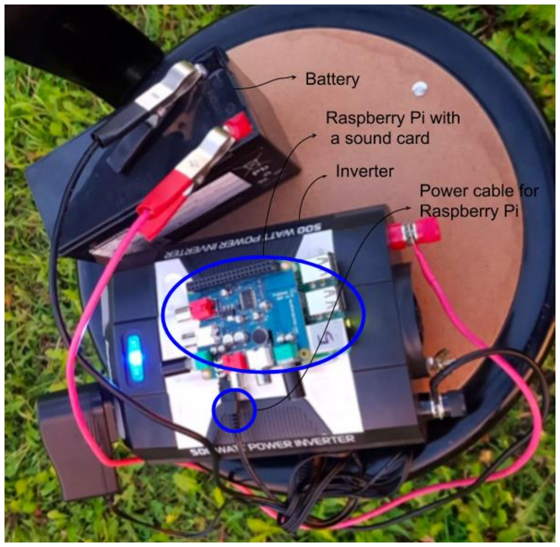


Figure 3. Matson et al. acoustic detection node hardware

The camera may track the moving UAV by vision. The range of detection heavily depends on the camera’s quality. The camera has a hard time to distinguish UAVs and seagulls (Naboulsi, 2015). The other limitation is the data transmission. Since the image and video are much larger than audio, the transmission requires good bandwidth of the internet.

Infrared sensors detect on gas-powered UAVs with large payload well, because they produce a lot of heat. The limitation is that most recreational UAVs do not produce much heat, which is difficult for IR sensors to detect (Naboulsi, 2015).

As for the UAV interception, common approaches are jamming, hacking and net-catching. Jamming is the most popular approach to interrupt the radio frequency communication between UAVs and the operator or the satellite (Michel, 2018). Some anti-UAV devices are invented based on the jamming principle. In 2017, the police of Wuhan China began to use the anti-UAV gun to defend the unregulated UAVs (Ye, 2017). The UAV rifle gun can affect to a 1 km range (Figure 4). Despite the price ($36,000), the anti-UAV gun is effective for UAV control. Jamming is only applied for military purposes, because jamming any authorized radio communication is banned in the USA (Jensen, 2019). In this case, the UAV jammer has a very limited market. There is a low-tech innovative approach to take down the UAVs. According to the Dutch company Guard from Above, they have trained large birds of prey for capturing the UAVs in the sky (Figure 5) (Morby, 2016). This approach does not have the risk of intercepting the airplanes, furthermore, it is direct, fast and accurate. The main concern is on the eagles’ health condition when they attack the UAV propeller.

Figure 4. Wuhan police testing the UAV gun

Figure 5. Dutch company taking down UAVs using trained eagle (Castle, 2016)

Since each detection method has their advantages and limitations, the combined-sensors system is the future solution. For instance, the UAV Dome defense system includes the radar, a surveillance system and a jamming system (Lappin & Binnie, 2016). This study will build the foundation of a multi-node adaptive system by applying the MAS to UAV detection, to add more bits of intelligence to the traditional detection system.

2.2.2 UAV moving route

The recent studies on predicting the UAV movement and route are few. By applying the search strategy, most researches study on using algorithms to plan the route of flying to avoid obstacles (Lin & Saripalli, 2017) (Dolicanin et al., 2018). To mimic the UAV’s flying characteristics, it is better to study what flying modes a UAV has. The goals of path planning are maximum safety and the shortest route (Jun & D’Andrea, 2003). To simulate the UAV movement, path-planning strategies may be considered. The mainstream path-planning functions are Follow Me, Course Lock, Waypoints, Home Lock and Point of Interest, according to the largest commercial UAV hardware producer DJI (DJI, 2018) (UAV Coach, 2019). The types of flying modes of the commercial UAV are manual and autopilot. Manual mode allows the operator to completely control the UAV’s movement. The autopilot mode frees the user’s hands. For example, the Follow Me mode allows the user to be hands-free, while the UAV following the user at a distance of 20m and a height of 30m. If the UAV is in Point of Interest function, it will fly around a center circularly. This flying mode can be used for investigating an area. Course Lock enables the UAV to fly along one direction straightly. Home Lock provides UAV the ability to return to the original point (DJI, 2018). The obstacle avoidance function is the key feature to keep the UAV safe, however, the function is also a challenge to maintain the cost of the UAV (Oliver, 2019). Many approaches use obstacle avoidance sensors to realize the function. At first, the obstacle detection sensors only install at the front of the UAV. Later on, the sensors are placed at the top, bottom, side and back of the UAV to maximize the safe level (Corrigan, 2019). Common sensors detecting obstacles are the camera, ultrasonic sensors, lidar, IR sensors and Time-of-Flight (ToF) sensors. There is usually a combination of sensors installed to a UAV. For example, a DJI Inspire 2 uses IR sensors, ultrasonic sensors and vision sensors. The vehicle can scan for obstacles 30m ahead and 5m upwards, while maintaining a speed of 34mph flying safely (Oliver, 2019).

2.2.3 Multi-agent system simulation

MAS is a system consisting of many agents working together to achieve a goal. The AI term “agent” means an entity which can perform continuously in a certain environment without constant human instruction and intervention (Shoham, 1993). Agent-based modeling is a great way for simulating complicated dynamic system (Macal & North, 2010) and emergent phenomena (Bonabeau, 2002). Imagine if a UAV attacks a building or smuggles drugs to a jail, it is hard to gather the real-life information from those uncommon events. Those events do not happen regularly, compared with earthquakes, storms, etc., resulting in difficulty to collect data and analyze the situations. The simulation, however, changes the game. Not only can the simulation repeat the abnormal emergent situations, but also the researcher can collect detailed characteristics of the system by controlling specific variables and environment parameters. According to Bonabeau, if the "individual behavior is nonlinear and can be characterized by thresholds, if-then rules, or nonlinear coupling", agent-based modeling has a potential for such circumstances (Bonabeau, 2002, p.1). The simulation benefits studying both theories and real systems (Uhrmacher & Weyns, 2009). Eventually, CUAS will apply to the real-life environment, so that the simulation will be a good start to test the theories before applying it practically. An agent-based model has three basic components: a set of agents describing their attributes and actions; a set of agent relationships and topology defining how agents interact, and the agents' environment where agents interact (Macal & North, 2010).

In the multi-agent system, the task allocation is very important to help agents reach the highest performance. The existing challenges include distributed intelligence and heterogeneity between agents. Kim and Matson's paper proposed an algorithm that can find the best-fit agent for a task (Kim & Matson, 2016). The assumption considered the job-allocation procedure as acquisition, selection and delegation, while ignored other debatable facts to concentrate on the scheduling problem. It assumed that the communication between agents is smooth. The variables included the utility, number of tasks, efforts, capability and time. The study intended to limit the number of variables so that the system would perform well. When the system scheduled the tasks, it started with agents checking if it was eligible to conduct the task and if it could complete the task within the deadline. Then the virtual task queue went through all tasks. After the agent responded to the virtual task queue, the capability value will be measured and the task would be assigned to the proper agent. The communication process shows in the following picture:

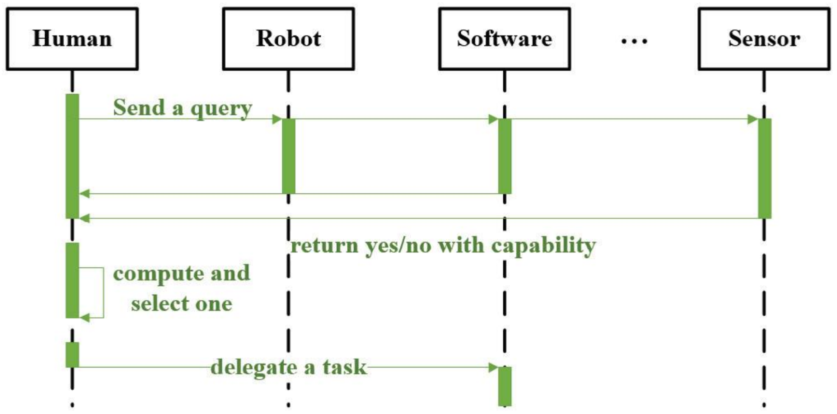


Figure 6. Agents' actions in a sequence diagram (Kim & Matson, 2016)

MAS simulation software observes the tactical decision-making process of the complex and dynamic systems (Siebers & Aickelin, 2018). An analytical model has a closed function; however, a simulation model provides views to observe any point of the system changing. According to Siebers and Aickelin, the purpose of the MAS simulation is to understand a complex system better or to predict and evaluate a system's performance. There are more than 80 agent-based modeling and simulation tools on the market. In summary, Table 1 lists the famous simulators and their attributes for evaluation (Mualla et al., 2018) (Allan, 2010) (Abar et al., 2017) (Schank, 2013) (Gonzalez, 2013). Confucius said, "The expectations of life depend upon diligence; the mechanic that would perfect his work must first sharpen his tools". To select the proper simulation tool to use, the characteristics of a simulator's learning ease, functionalities, user interfaces, supports and so on will be weighted in a scale of 1-5, where 1 represents the worst and 5 stands for the best. The explanations of each measurement see the following:

- Language – the kinds of programming languages supported by the simulator
- Highlight – the special benefits of the simulator
- Learning easiness – if the simulator is first-time user-friendly (clear tutorials, time to make the first project)
- Capability – if the simulator can handle large scale agent modeling (computing speed and computing memory required)
- Community – a forum or community where the developers and users can discuss problems (community existence, community update, community popularity)

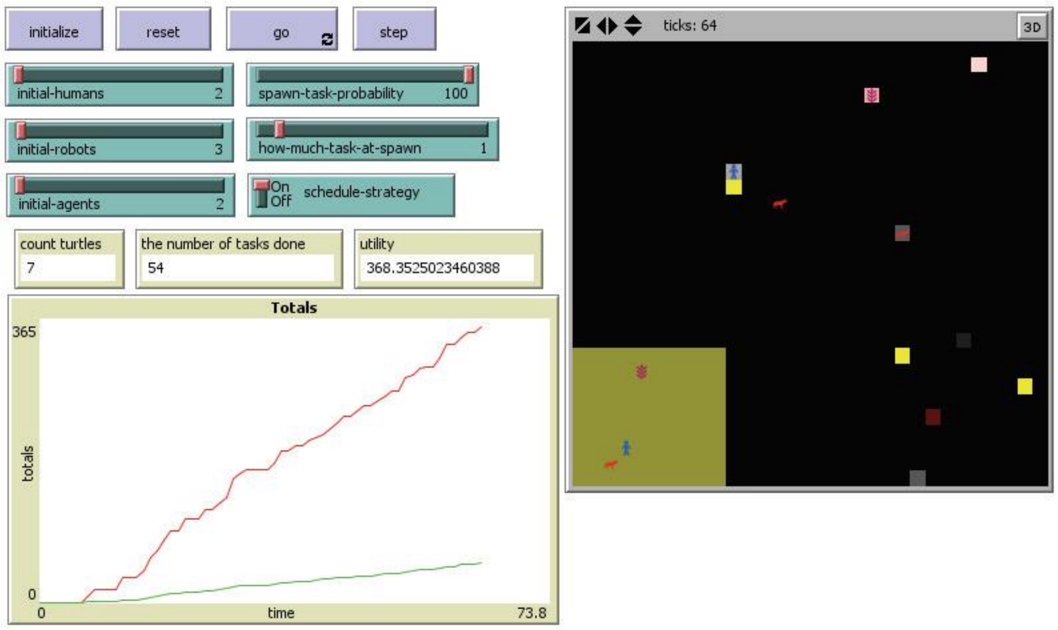
- Maintenance – see how much devotion the developers input to the simulator and how often they maintain the software, website, documents, events and everything

Table 1. Evaluation on different ABM simulators

Attributes /Name	Language	Highlight	Learning easiness (1-5)	Capability (1-5)	Community (1-5)	Maintenance (1-5)	**Total** (out of 20)
Gazebo	C++, Python, Java	Free, open source, ROS, high-quality graphics, 2D/3D	4	5	5	5	19
Repast	Java, C#, C++, etc.	Free, open source, 2D/3D modelling	1	5	2	4	12
Flame	Sample libraries & tutorials	Free, open source, agents characterized by conditions	2	5	1	1	9
NetLogo	Logo	Free, open source, 2D/3D modelling,	5	4	5	5	19
JADE	Java	Free, open source, graphical tools, remote GUI	3	2	3	3	11
MASON	Java	Open source, 2D/3D modelling	1	4	1	3	9
Swarm	Java, C, Swarm-code	Free, open source, collection of agents interacting	1	5	2	1	9

2.3 Studies & Proposed Methodology

According to the evaluation of multiple agent-based modeling simulators in the past chapter, NetLogo will be used developing MAS simulation. Compared to other competitors, NetLogo is more user-friendly, supportive and popular among researchers, which indicates that there are more tutorials and papers compared to other simulators. NetLogo is also free and open source software which supports Java extension. Based on the researcher's experience, Java will be the ideal programming language to develop the simulation. According to Kim and Matson, they used NetLogo GUI to improve the display of the simulation. In NetLogo environment, different variable values were simulated. The graph showing the relationship between time and progress was plotted (Figure 7) (Kim & Matson, 2016). Based on the previous researchers' experiences and NetLogo's outstanding features, NetLogo was decided to be the simulator. Although Gazebo weights the same score as NetLogo according to Table 1, Gazebo benefits ROS more so that NetLogo remains to be the best choice to simulate MAS.



initial-humans 2
spawn-task-probability 100
initial-robots 3
how-much-task-at-spawn 1
initial-agents 2
On Off schedule-strategy
count turtles 7
the number of tasks done 54
utility 368.3525023460388

Figure 7. NetLogo GUI (Kim & Matson, 2016)

The intelligent agents will be developed in the simulation. In MAS, there are simple agents (Figure 8) and learning agents (Figure 9). The main differences are their behaviors. According to Siebers & Aickelin, an intelligent learning agent needs to perform those behaviors:

- Responsive behavior – perceiving the environment and responding timely
- Pro-active behavior – acting goal-directed behaviors, being driven and opportunistic
- Social behavior – interacting (eg. commanding, listening, receiving) with other agents

- Flexible behavior – adjusting the approach to achieve the goals; dealing with failures (Siebers & Aickelin, 2018)

According to the goal of the UAV detection system, the agents with sensors will perceive the environment where the UAV is approaching. When an agent detects UAV, the agent will remind the neighbor agents to be alert because one agent's information may not be enough to accurately predict the UAV's heading directions. After the other agents receive the alert and detect the UAV, those agents begin to exchange and generate the information, then they predict the UAV's direction while alerting the agents near that direction. This logic will become the foundation of developing the intelligent agents.

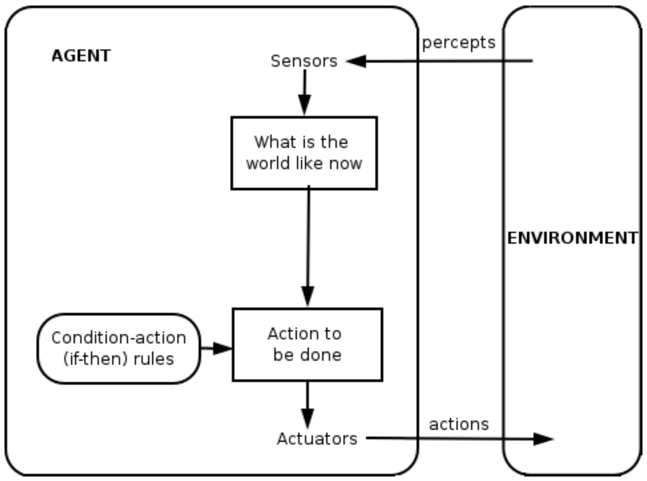


Figure 8. Simple agent logic (Atmaram, 2006)

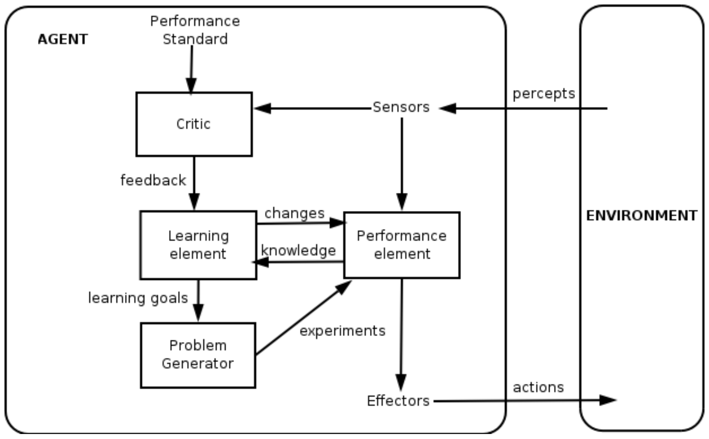


Figure 9. Intelligent learning agent logic (Atmaram, 2006)

2.4 Summary

In this chapter, literature reviews had been done which addressed the searching strategy, the significance of the problem and the other researchers’ approaches. Since applying MAS to UAV route track is a new combination in academics, the researcher introduced the background information and recent applications on each subject (CUAS, UAV route plan and MAS) with necessary logical interpretations and hypotheses.

CHAPTER 3. PROPOSED METHODOLOGY

The goal of this research is to simulate the counter UAV detection system with built-in intelligent agents. In this chapter, the research type will be explained. Then, the experiment design with its data type, instruments and data collection approach will be listed. At last, a development cycle and the time action plan will demonstrate the arrangement and expectation of the study.

3.1 Research Type

The research type is a quality descriptive research due to the hypothesis needed to be proven. An experiment will be designed to evaluate the system performance. After analyzing the data, the question, if applying MAS to UAV detection will be able to track the UAV movements, can be answered. In this case, since the goal is to prove a hypothesis, the study will be considered as quality descriptive research.

3.2 Experimental Design

The experiment will conduct in the NetLogo simulation environment. Previously, some intelligent agents were designed. The communication between agents will be enabled and tested. To set up the experiment, an observing tool will be designed according to the simulator's features. The observing tool is to observe the complete experiment process and data, which includes the initial UAV accessing angles, observed UAV leaving angles, predicting UAV leaving angles, etc. The observing tool will record all the data in a script for the further analysis. Evaluation focuses on the UAV prediction accuracy. By comparing the predicting UAV leaving angles with the observed UAV leaving angles and comparing the tracked UAV route with the real UAV route, the statistic error rate and the accuracy will be obtained.

3.3 Population & Sample

In a condition that the experiment will run 100 times, then the population will be all data collected during the 100 experiment executions. Considering one experiment data as a group of data, then there will be 100 groups of data. 10 groups of data will be used for analysis, so that the sample data will be the 10 groups of data which will be randomly drawn from the population. Because of random draws, the system performance evaluation will be more objective and less biased.

3.4 Agent Rules

The main goal is to design intelligent agents (nodes) which will constantly listen to the UAV and track the locations based on the overall information collected by each involved agent. Agents expectations:

- Alert – The nodes should actively listen to the UAV. Once the UAV reaches the detection area, the nodes can detect it immediately. “Listening” indicates “watching”, “focusing” or “actively waiting”. It is important to always have one or more agents actively listen to the UAV. Just like the iron fillings always point to the magnet no matter where the magnet moves, the agents also should be alert of UAV’s movement. Constantly listening will benefit on the completeness of data collection, and later, the accuracy of track.
- Accurate – At one moment, the nodes should detect the UAV’s distance accurately.
- Adaptive – The nodes should be adaptive with multiple kinds of sensors. The proximity sensors will be used in the experiment due to the detection methodology. In the future, it is possible to develop the adaptivity by adding more sensors.
- Communicative – The agents talk to each other to share the information and make decisions timely.

Agent rules can be explained in a detection scenario example (Figure 10 - 16):

a) **Agent detection time frame** – The agent cluster surrounds the detection area. The detection will begin when the UAV accesses the edge of the circle, and it will end when the UAV leaves the edge of the circle.
b) **Detect** – The agent can detect the distance with the UAV.
c) **Talk** – Agents will report their distance with UAV every t seconds, $t > 0$, then the three closest agents will be active. In Figure 10, at the green point, the agent 1, 2 and 16 are the closest with the UAV. After t seconds, when the UAV flies to the blue point,

the agent 1, 2 and 3 are the closest with the UAV. Things change at the next 2 seconds showing at the purple point, where the agent 2, 3 and 4 are the closest.

d) **Report** – After talking, the 3 agents which have the smallest distance with the UAV will be reported. For example, at the green point, the agent 1, 2 and 16 will be reported. After 2 seconds, the agent 2, 3 and 4 will be reported since they are the closest. Every 2 seconds, there will be 3 agents reported near the UAV.

e) **Record** – The reported agents will measure the distances from them to the target UAV using proximity sensors. For example, the first measurement is at the green point. The distance a between agent 1 and the UAV is 5. The distance b between agent 2 and the UAV is 8. The distance c between agent 16 and the UAV is 10.

f) **Mark** – According to trilateration (Figure 11), a green circle is drawing with agent 1 as the origin and distance a as the radius. Similarly, agent 2 is the origin of the second green circle. Agent 16 is the origin of the third green circle. The intersection point of the three circles will be marked as the green cross (Figure 11). Similarly, the blue and purple crosses is marked accordingly (Figure 13, 15).

g) **Sketch route -** After marking crosses, a curve will be drawn to connect those intersections (Figure 16). The curve will track the route of the UAV when it goes through the detection area.

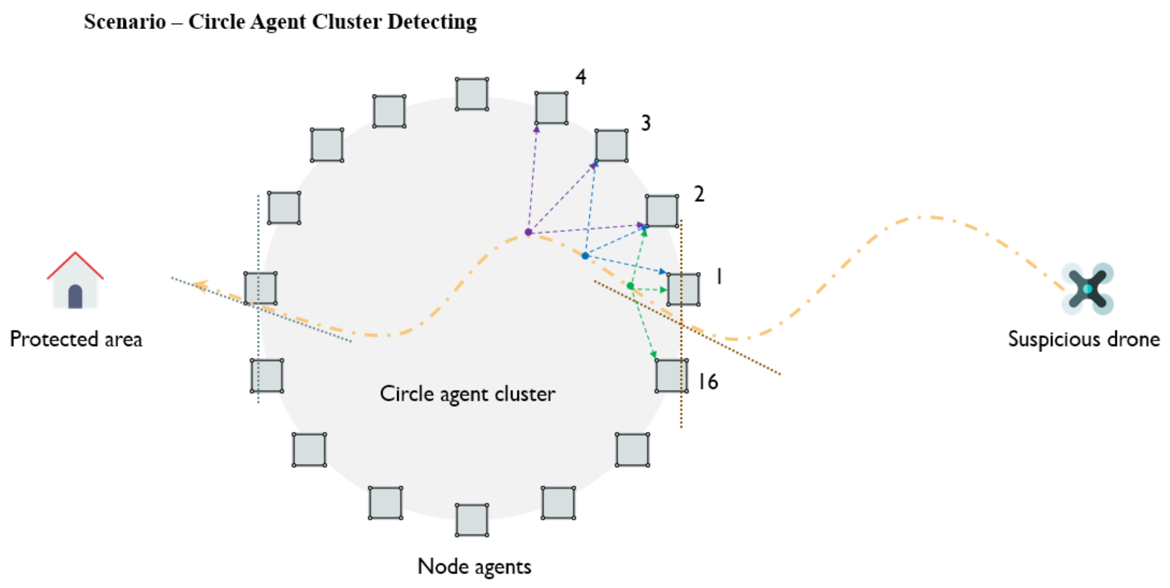


Figure 10. Circle agent cluster detecting UAV scenario

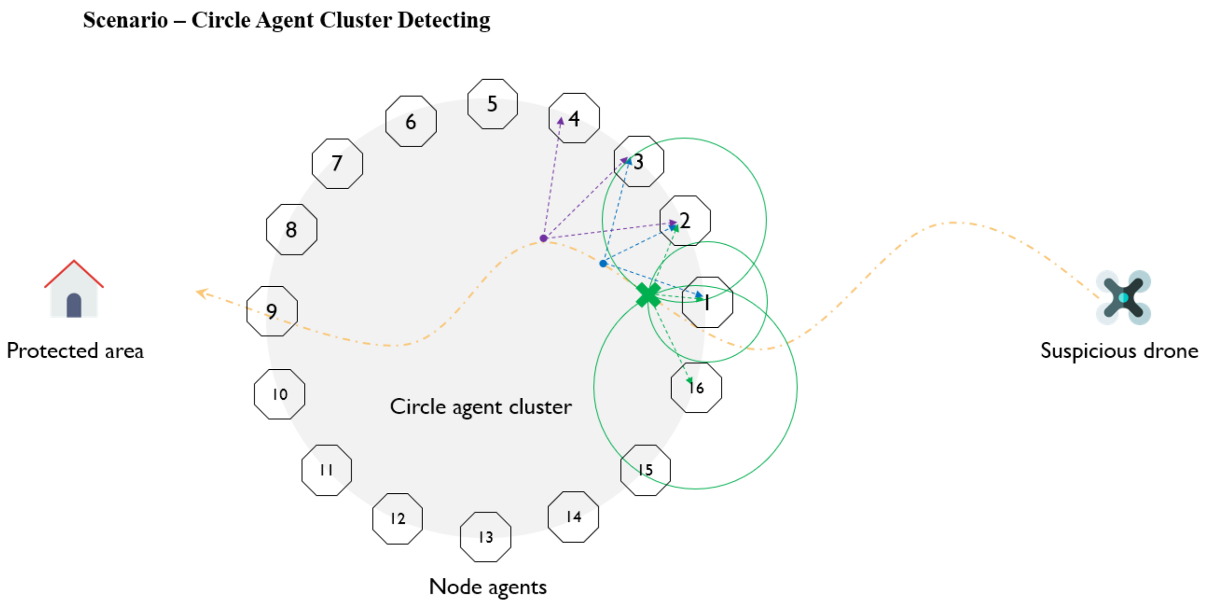


Figure 11. Agent 1, 2, 16 detecting UAV

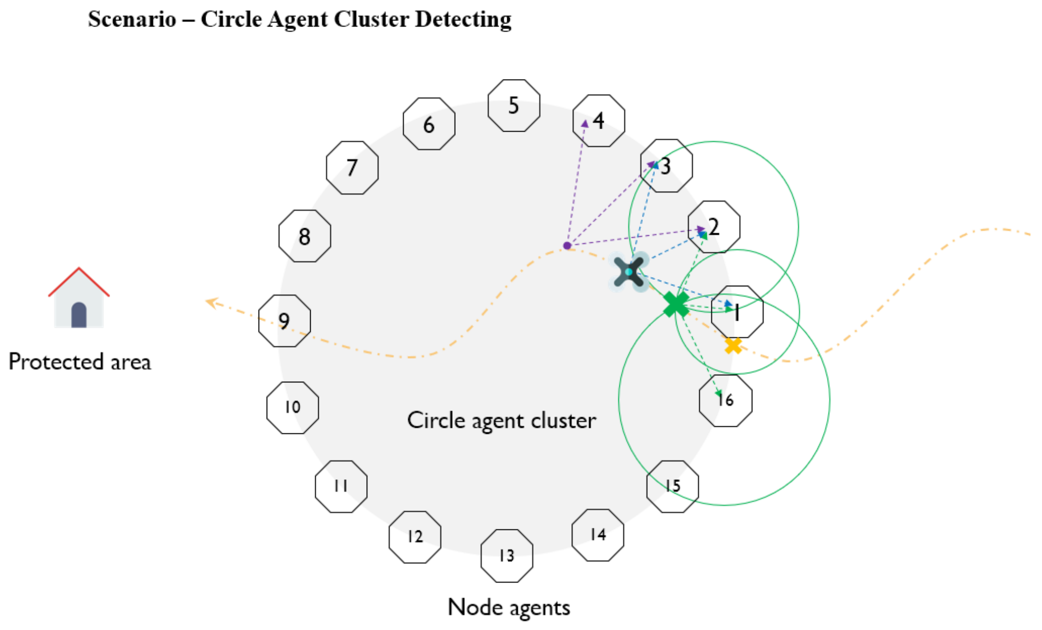


Figure 12. Green cross marks where UAV is at

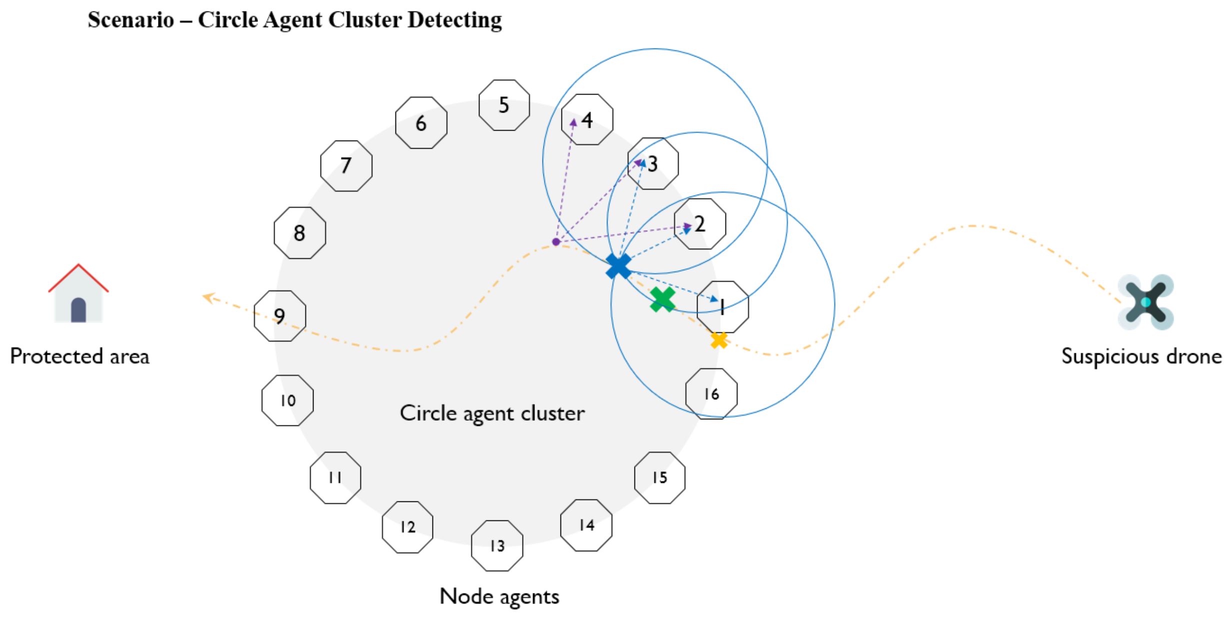


Figure 13. Blue cross marks where UAV is at

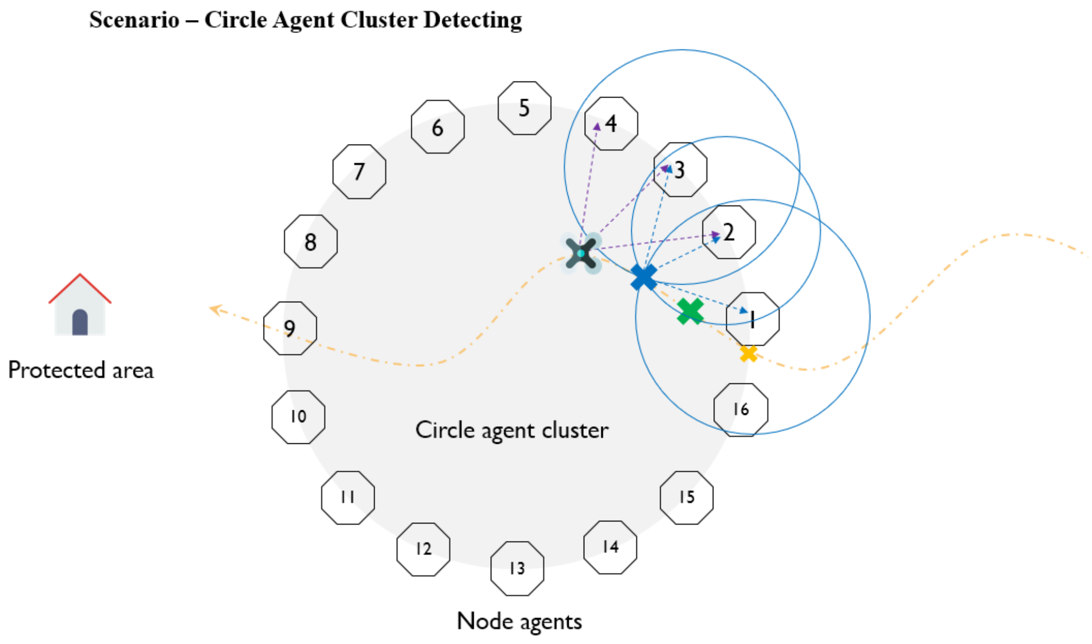


Figure 14. Agent 2, 3, 4 detecting UAV

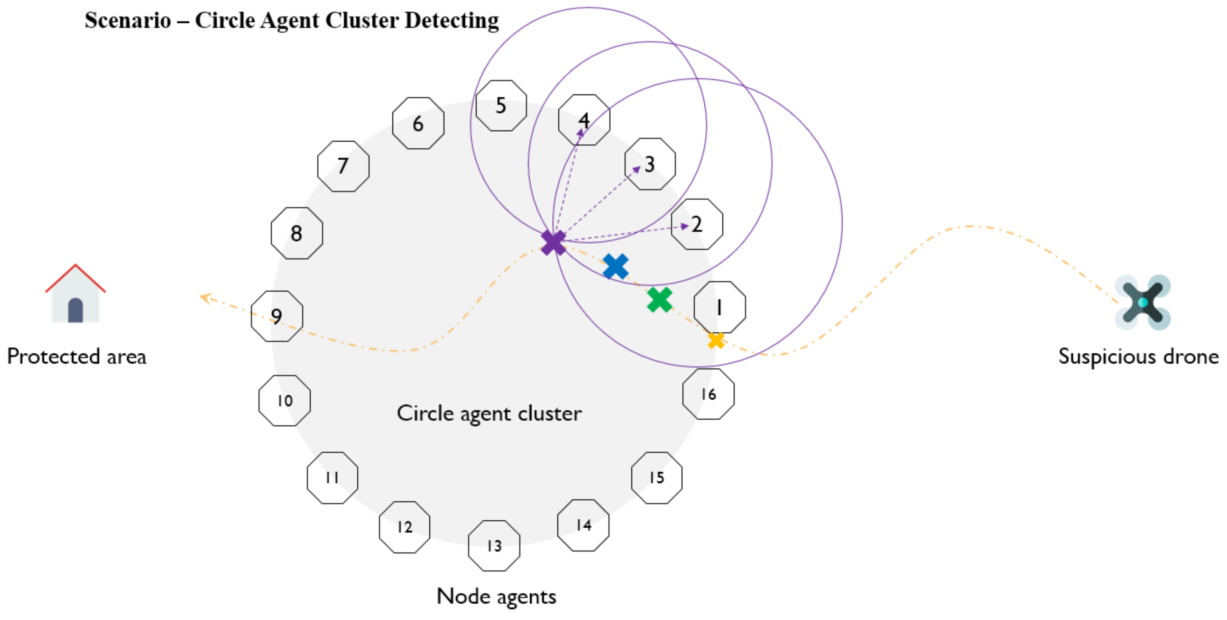


Figure 15. Purple cross marks where UAV is at

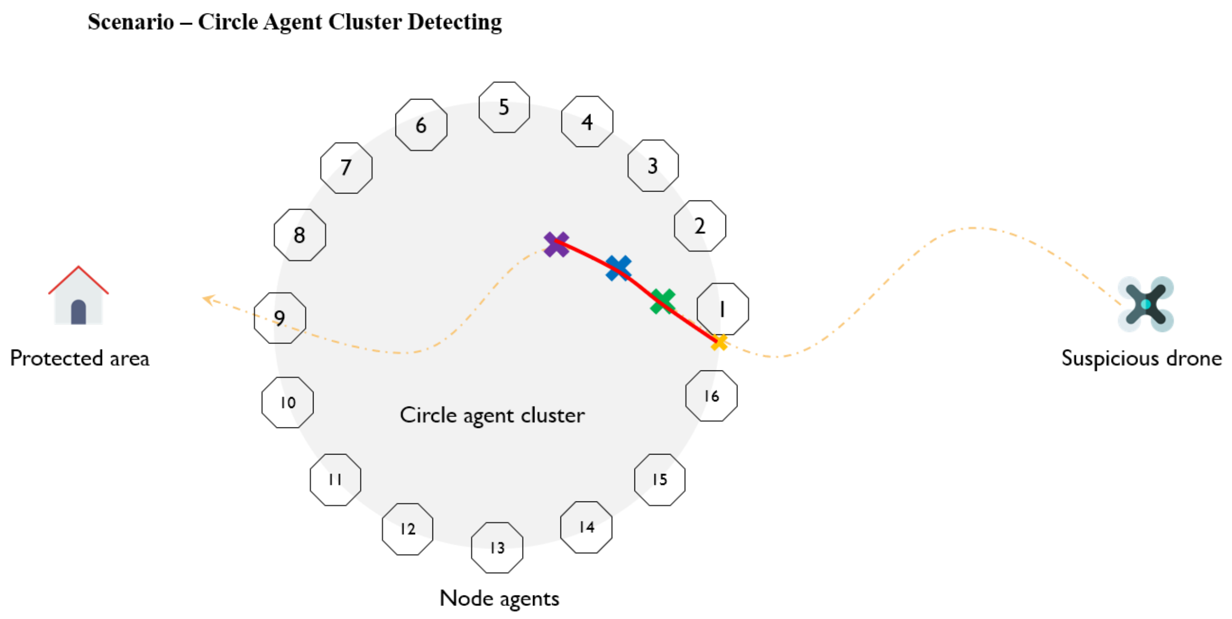


Figure 16. Circle agent cluster drawing UAV route

3.5 Instrumentation

To build the UAV detection simulation, the NetLogo simulator software was used. The Java extension package may be used to achieve complex features of the system and the intelligence of the agents. For the hardware, a laptop with the Intel Core i5 CPU and 8GB RAM will be used.

3.6 Variables

Table 2. Variables in the experiment

Name	Symbol	Comment
	Independent Variables	
Agent cluster base	x_base, y_base	Where the cluster baselines are
Station agent gap	h_gap, v_gap	Horizontal gap value and vertical gap value between stations
Sensor detect radius	detect_radius	Within the radius, the sensor on the station can detect the distance of the UAV
	Dependent Variables	
Mark intersection x coordinate	i1_x, i2_x, i3_x … in_x	
Mark intersection y coordinate	i1_y, i2_y, i3_y … in_y	

To test the reliability and validity of the data, the simulation will be repeated fifty times to see how many accuracies the system tracks the UAV's route. If some results hit the goal while some does not, the reliability of the data needs to be checked, seeing if the weights are consistent.

The system performance will be evaluated in the detection accuracy. The detection accuracy measures how close the track is with the real UAV movements in coordinates.

3.7 Data Collection

There are two main categories data which will be collected: the observed data and the calculated data. The observed data will be generated from the agents, which will help to track the UAV movements. The observed data will result from the built-in observer feature in NetLogo, so that the observed data will contain every available data during the experiments. The agents and the simulation observer will manage to collect the data. Ideally, the observer will generate a script consisting of all formatted data.

Observed data comes from the direct observation of the experiment. The observed data is generated by the simulation software. The calculated data results from the data observed. Here are tables of variables:

Table 3. Observed and calculated data list

Name	Symbol	Comment
	Observed data	
Station color	color	Black – non-detected Yellow – detected
Station x coordinate	ox1, ox2, ox3 … oxn	
Station y coordinate	oy1, oy1, oy3 … oyn	
Mark agent size	r1, r2, r3 … rn	Distance between the station and the UAV
Mark intersection	i1, i2, i3 … in	
Ticks	t	Time counted by the simulation observer
	Calculated data	
Intersections' x coordinates	i1_x, i2_x, i3_x … in_x	
Intersections' y coordinates	i1_y, i2_y, i3_y … in_y	

3.8 Scenario Examples

In the section, the sample scenarios will be introduced to provide a better idea of the experiment design and concept.

3.8.1 Types of agent clusters

There are two models of detection areas for comparison. One is the rectangle agent cluster, where the agents arrange in a sequence on the edge of a rectangle (Figure 17). The other is the circle agent cluster, where the agents arrange like a circle cluster (Figure 18).

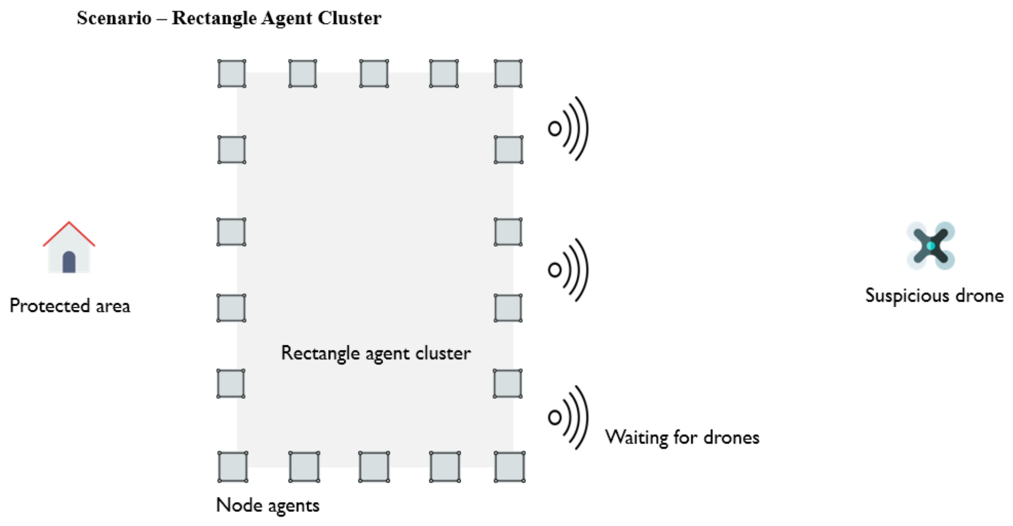


Figure 17. Rectangle agent cluster waiting for UAVs

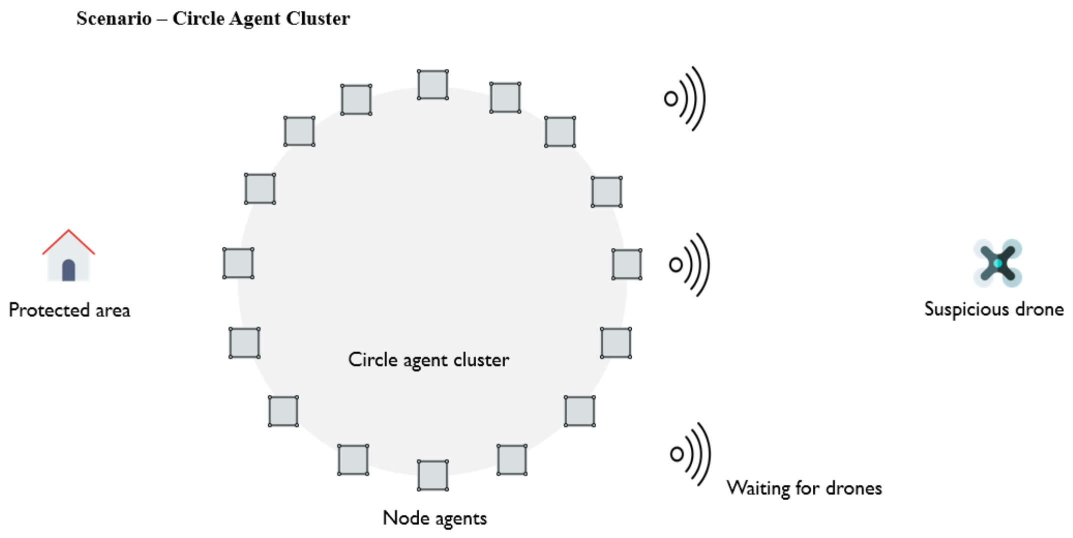


Figure 18. Circle agent cluster waiting for UAVs

3.9 Time Action Plan

The time for this study is limit but manageable. The agile development cycle will be performed on the study, so that consulting with committees, designing, developing and testing phases will be iterated as need, showing in the agile graph below:

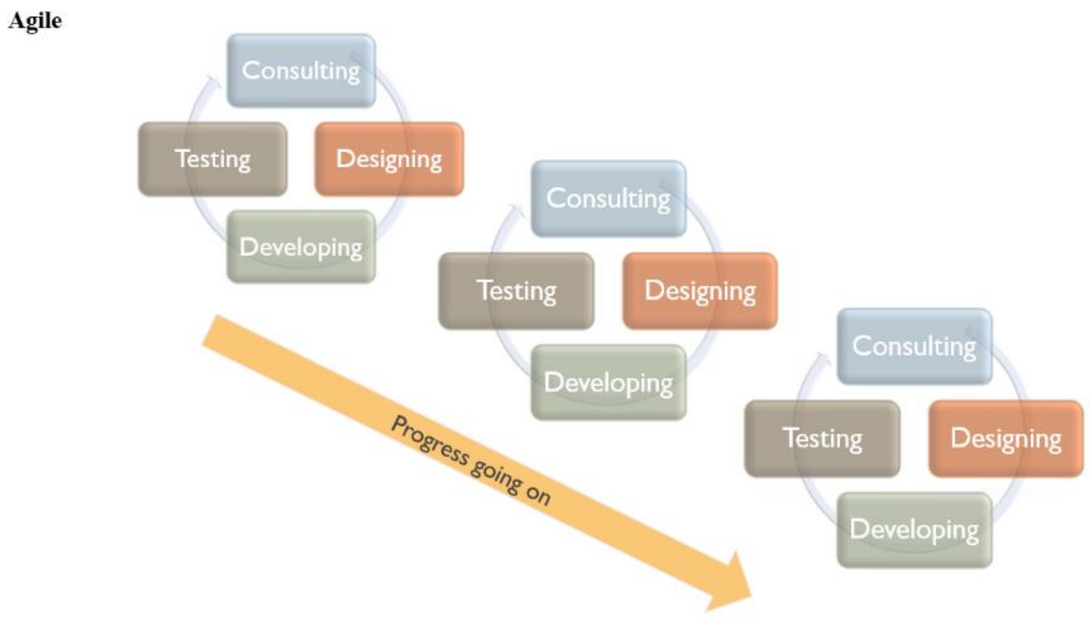


Figure 19. Agile methodology of developing

3.10 Summary

The chapter 3 introduces the overall proposed methodologies. For example, the research type, the experiment blueprint, the agent rules, data to be collected, data sampling method, expected experiment scenarios and the time action plan are discussed in this chapter.

CHAPTER 4. ACTUAL EXPERIMENT

4.1 Introduction

The chapter explained the process of the simulation experiment scenarios, logic flows, agent actions and the simulation GUI. Compared with the proposed methodology stated in Chapter 3, the real methodology followed the proposal methodology, however, focusing on more details and involving more calculations, which were not recognized in the previous chapter. The reason was because of the inaccurate estimation of what the simulation software can or cannot do. In proposed methodology, the intersection coordinate should be available from the observer. However, after the initial test, the simulation did not provide any intersection information. In this case, it was decided to calculate the coordinate using geometry.

4.2 Initial Test of Proposed Methodology

The baseline experiment which tested a single line of an agent cluster (Figure 20). The goal for the test was to identify when the agent cluster should start to detect the UAV when it flew close to the detection area.

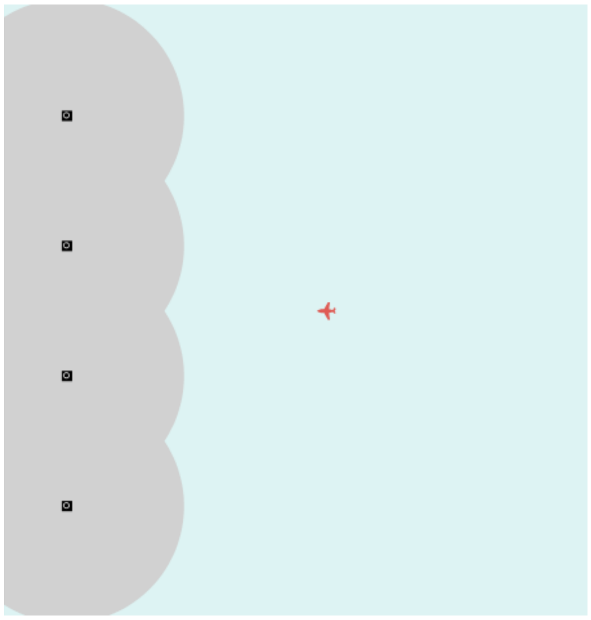


Figure 20. Initial test setup

The initial test only involved two intersections (i1, i2). Since the agents ranked in a line, there were only two efficient marks working as defining the distance. Two marks mean that there would be one or two intersections involved, so that both intersections would be marked with a red cross (Figure 21). The situations that which intersection was the valid one would be clarified in the final simulation test.

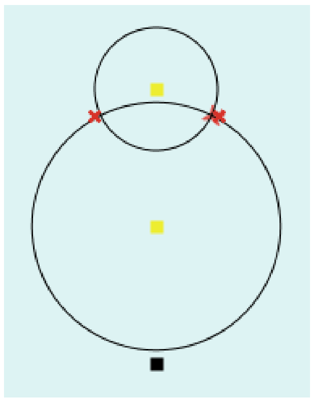


Figure 21. Normal two marks intersections

According to the initial test, some problems showed, so another look was taken to the proposed methodology logic. Sometimes the marks did not show up at the correct stations. In this case, a mark for each station was built. At where every station stays, there would be a built-in mark. When the UAV approached to one of the stations, the station’s mark would be visible, tracking the distance between station and the UAV. Another problem was that the crosses were not mark the intersections appropriately (Figure 22). To fix this problem, the program was troubleshot by hand-calculating where the intersection should be, then compared with the simulation results. A constant was added to improve the result as a trial. Turning out, the formula itself needed to be improved. Solving problems from the initial test was a great to build and conduct the final experiment.

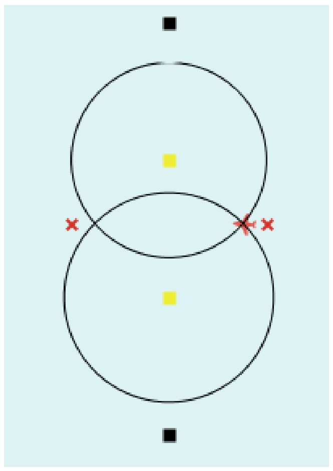
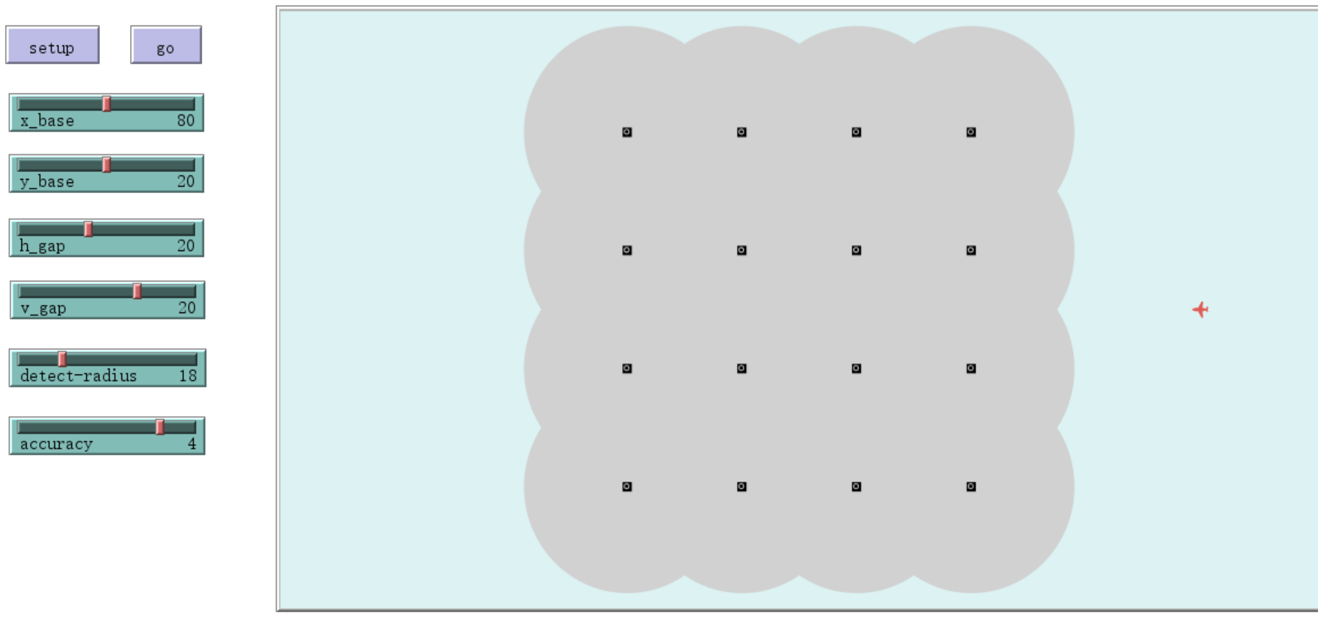


Figure 22. Abnormal two marks intersections

4.3 Final Test Introduction

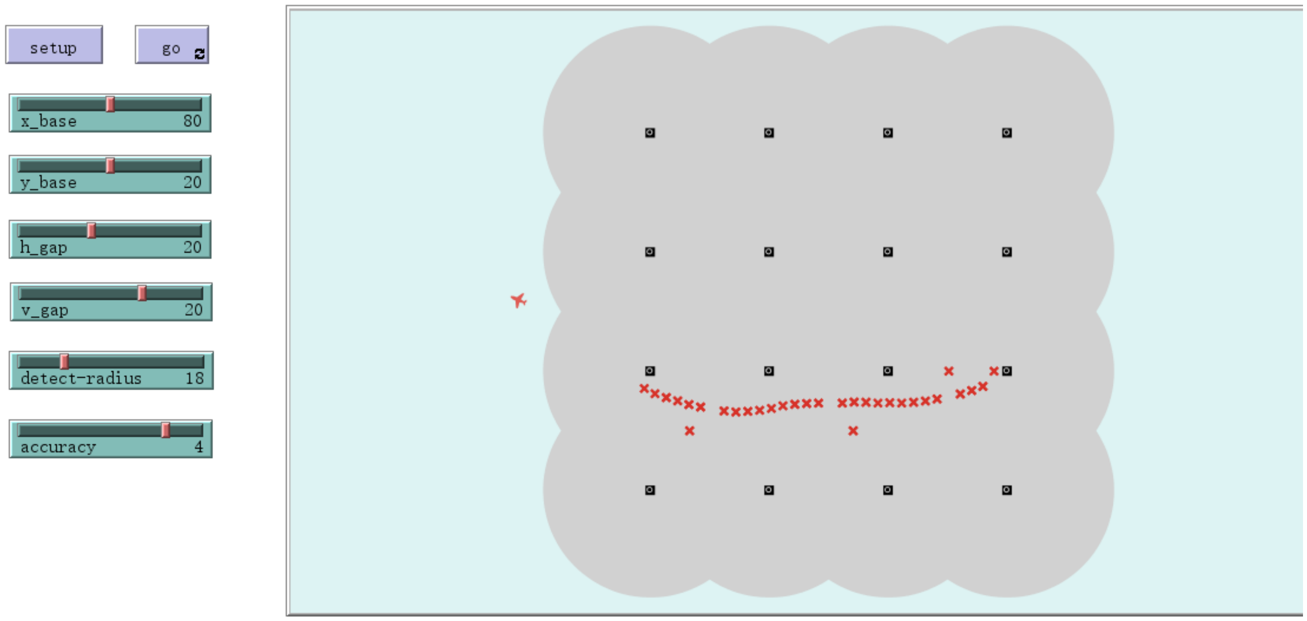
In the final test, there were total sixteen stations and one UAV participating as the agent cluster for detection. The layout of the GUI setup showed in Figure 23. The running simulation showed in Figure 24. The UAV was presented by a red airplane. The stations were the black squares. The gray area surrounding stations were their detection range. Due to the analogous characteristics shared by a group of agents, the detection range for each station was set to be the same. Indicated that all the stations loaded at least one kind of the proximity sensor. The model was proposed to be a framework which can adapt with different sensors tracking the UAV. It was still possible with the current model, because one station can load one or more sensors. The sensor-layer idea would be discussed in the future work session. In this experiment, the focus was on the single layer of proximity sensor on the stations.

On the left of GUI, there are basic setup values and a go button to start the simulation. Since the origin (0, 0) at the left bottom corner was determined, so x_base and y_base used for setting up the baseline of the agent cluster. The horizontal distance between lines of agent was h_gap. Similarly, the vertical distance between lines of agent was v_gap. The user can setup the detection range of every station by sliding the bar of detect-radius. There must not be any detection gap in the cluster, otherwise the cluster may lose the track of the UAV.



x_base 80
y_base 20
h_gap 20
v_gap 20
detect-radius 18
accuracy 4

Figure 23. Final test setup



x_base 80
y_base 20
h_gap 20
v_gap 20
detect-radius 18
accuracy 4

Figure 24. Final test running

4.3.1 Why did the model change?

Compared with the scenarios in the proposed methodology (Figure 25 and 26), the station setup changed. Since the initial experiment was conducted to test if the simulation performed the trilateration well, suggested by the committee, the model with four stations lined up vertically was tested. The final model was extended from the initial test model. As a result, in the final model, the stations setup in a rectangle grid.

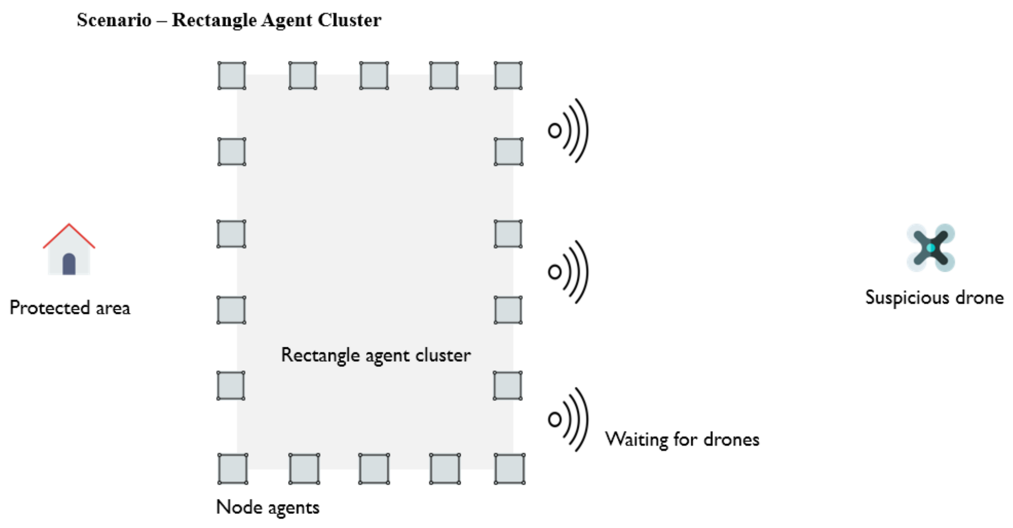


Figure 25. Proposed rectangle agent cluster

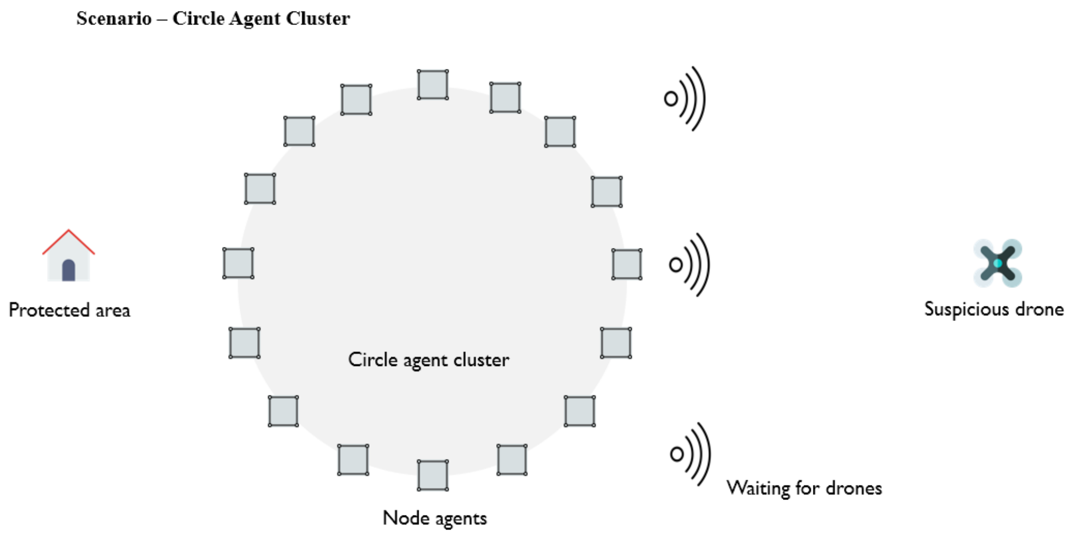


Figure 26. Proposed circle agent cluster

The main difference between the proposed models and the final model was that there were no stations inside of the detection area according to the proposed model (Figure 25 and 26); however, in the final model, not only the stations surrounded the detection area, but also there were stations inside of the detection area. The reasons to make the change were:

- To reduce the needs of long-distance proximity sensor

- To improve the detection efficiency when the UAV flew into the detection area
- To make the research process more tangible
- To break down the research scope

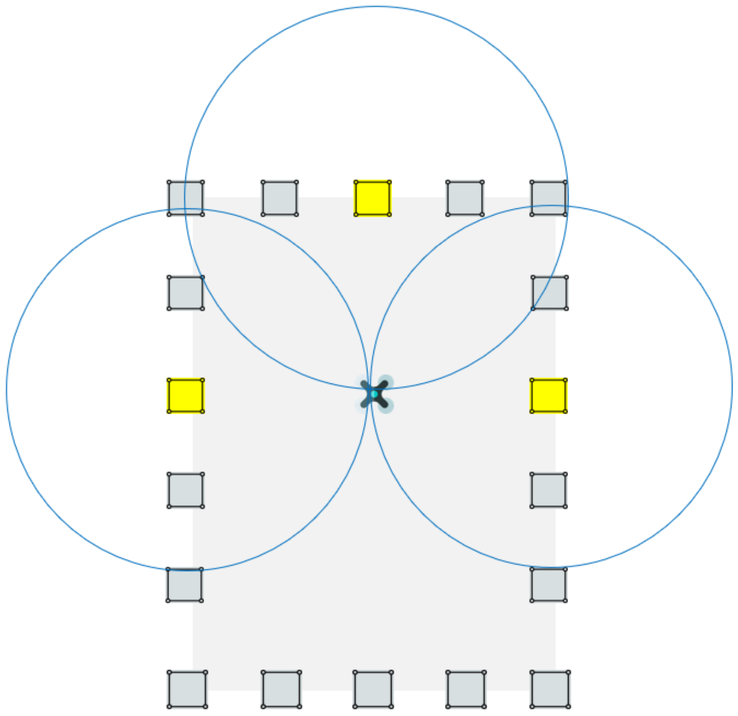


Figure 27. Proposed model, long-distance sensors required

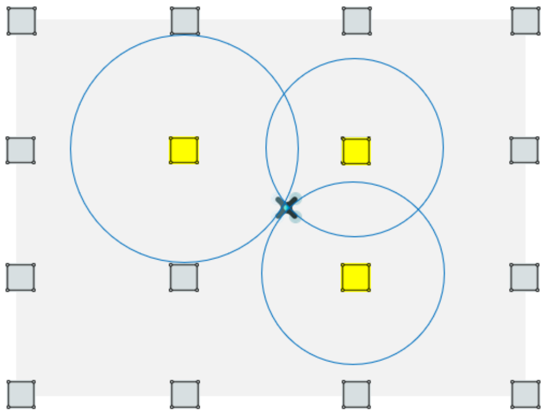


Figure 28. Final model, short-distance sensors

If the stations only surrounded the detection area and no detection gap was supposed to be in the detection area, it indicated that the sensor detection range had to be large enough (Figure 27). Wherever the UAV located in the detection area, there always should be some stations watching the UAV. However, when the model changed to the grid model, more stations were used, so that the distances between stations were smaller than the proposed model (Figure 28). Thus, the proximity sensor range can be smaller. Since the stations became closer, the detection gap would be hardly found, resulting higher efficiency to locate the UAV inside of the detection area. The third reason stated that the design of the final model was pretty much based on the initial test model. In this case, the research process seemed more tangible because each development phases depended on the previous phases. The fourth reason to change the model was because of an adjust of the research scope. Originally, both proposed models were going to be tested. Due to the initial test, a vertical line of stations worked well using the trilateration to detect the UAV, so that only the grid model was developed for the final test, instead of the circle model. The principle to calculate the intersections was the same, so it was possible that the circle model would work using the trilateration. It could be a good direction to continue this research.

4.4 Modified Agent Actions

Compared with the proposed agent actions, the modified version of agent actions was more accurate to describe what agents did in the experiment to achieve the goal of tracking. The agent actions were showing as following:

a) **Agent detection time frame** – The agent cluster surrounded and filled in the detection area. The detection would begin when the UAV was detected by two stations, and it would end when the UAV cannot be detected by at least two stations anymore. Once the UAV flew into the station's detection range, which showed as the grey area in Figure 29, the according station would turn to yellow color.

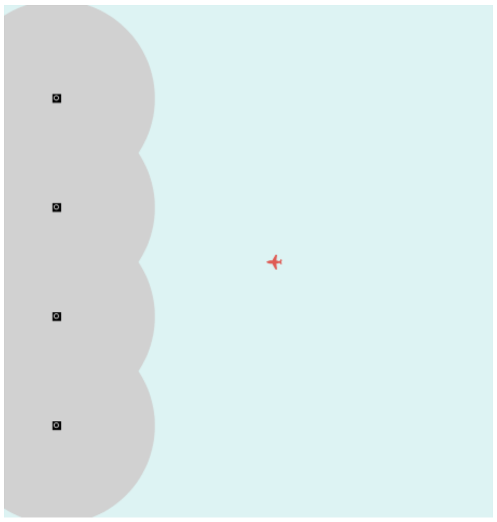


Figure 29. Grey area, the detection range of the station

b) **Ask** – The system observer asked the station agents whether they saw a UAV.

c) **Detect** – If stations found a UAV, the station changed its color from black to yellow (Figure 30). The station agent detected the distance between itself and the UAV. The distances would record as values r ∈ { r1, r2, r3 … rn }.

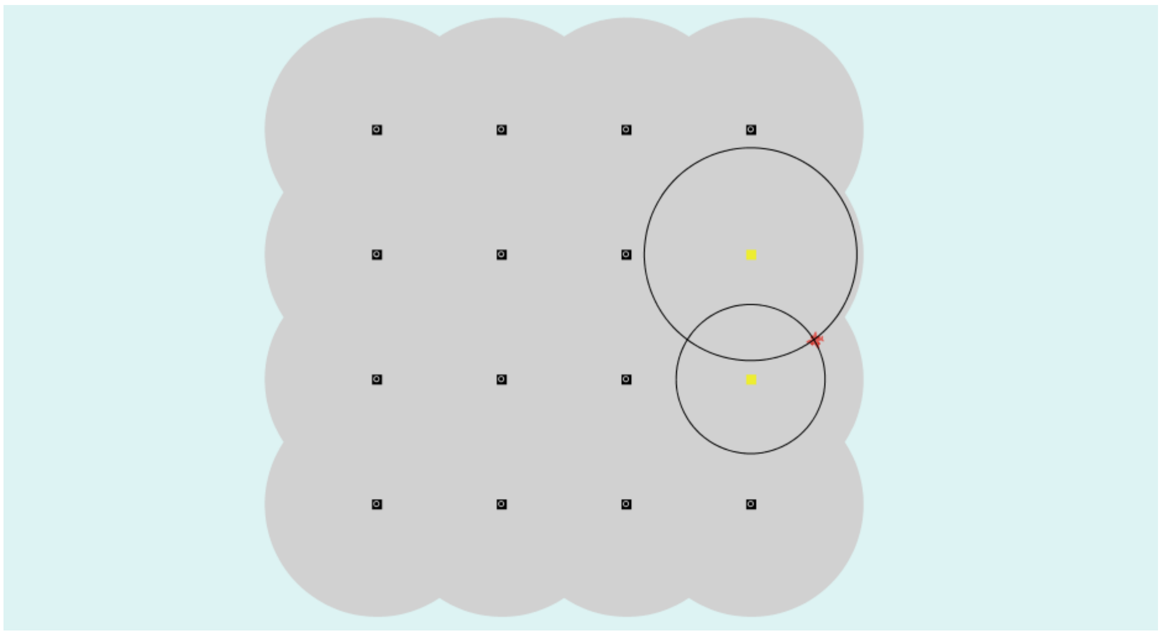


Figure 30. Two stations detecting the UAV

d) **Select** – If there were more than three stations in the list, according to the distances, the three stations with closest distances would be considered.

e) **Report** – The system would report the selected stations and their distances into a list. The stations were where the center of the circles located, recorded as o ∈ { o1, o2, o3 … on}.

f) **Mark** – Based on the distance, the station drew a black circle with the radius equal to the distance, to track the UAV (Figure 31).

g) **Draw Intersections** - According to trilateration, a red cross would draw at the intersection of the marked circles (Figure 31). The intersections were recorded as i ∈ { i1, i2, i3 … in }. The x coordinates of intersections were intersection number_x ∈ { i1_x, i2_x, i3_x … in_x }, where the y coordinates of intersections were intersection number_y ∈ { i1_y, i2_y, i3_y …in_y }.

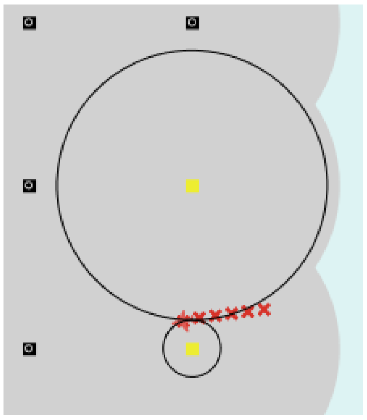


Figure 31. Red cross drew at the intersection of two circles

h) **Track** – Along with the UAV moving through the detection area, there would be more and more red crosses, which formulated the UAV track detected by the system (Figure 32).

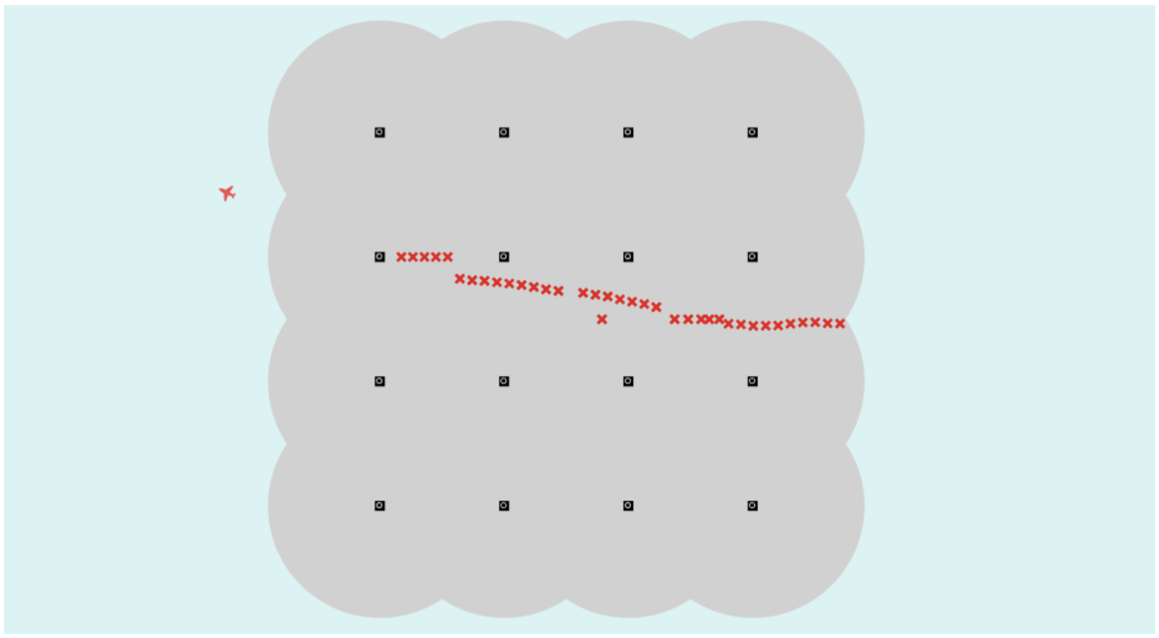


Figure 32. The UAV track drew by the system

There was not a Sketch step. The reason was that once the crosses were drawn at the intersections along with the UAV movements, the crosses were quite concentrated and could describe the UAV movement well. An extra curve line could make it clearer, but it was not necessary. The Predict process was removed as well. The reason was that the data gathered would not be enough data to make an accurate prediction. When the UAV was leaving the detection area, the detected stations would be less and less. At last, there would be only one station detected the UAV, which did not provide enough data for tracking and predicting.

The interactions between agents showed in the following sequence diagram:

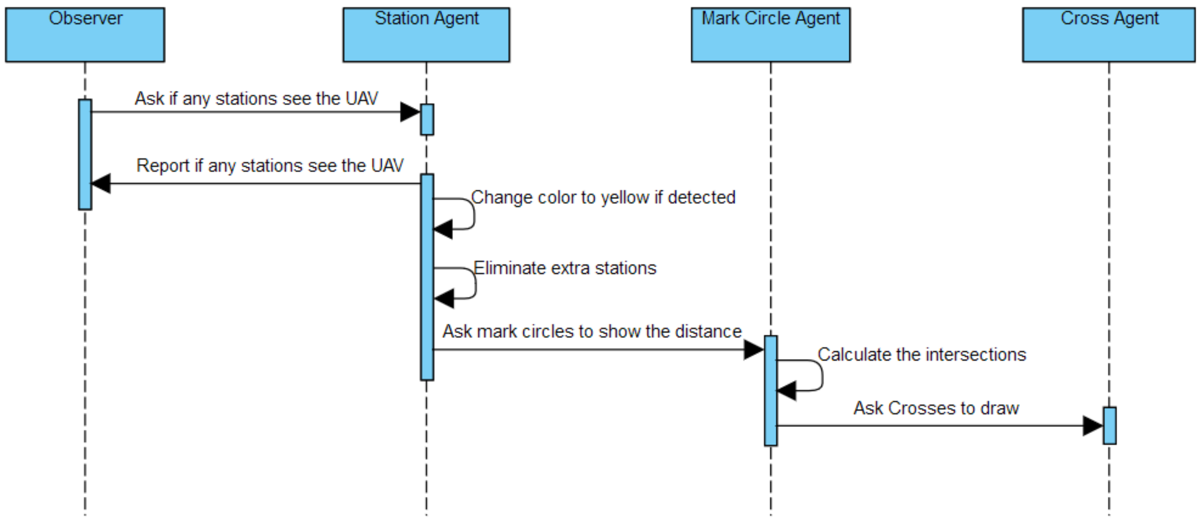

Observer
Station Agent
Mark Circle Agent
Cross Agent
Ask if any stations see the UAV
Report if any stations see the UAV
Change color to yellow if detected
Eliminate extra stations
Ask mark circles to show the distance
Calculate the intersections
Ask Crosses to draw

Figure 33. Modified sequence diagram

4.5 Modified Scenarios Breakdown

To mimic the sensor tracking the UAV movement in the detection area, the trilateration was proposed to locate the UAV. In the simulation, the trilateration method was broken down to adapt with different scenarios to make a better track of the UAV's location. Since the 4*4 agent cluster ranking as:

$$\begin{vmatrix} 13 & 9 & 5 & 1 \\ 14 & 10 & 6 & 2 \\ 15 & 11 & 7 & 3 \\ 16 & 12 & 8 & 4 \end{vmatrix}$$

and was combined of nine 1*1 agent cluster (Figure 34), the 1*1 cluster was the one to discuss about.

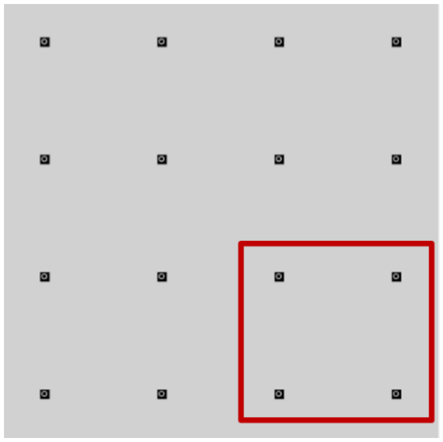


Figure 34. 4*4 Agent cluster with 1*1 marked

For every 1*1 cluster, there could be two or three marks to jointly identify the UAV position. The following subchapters explained how the algorithm worked in the different scenarios.

4.5.1 Track with Two Stations

When the UAV first approached the agent cluster, usually the first column of agent on the right, would detect the UAV and alarm the detection area. When the UAV was about entering the detection area, it was always tracked by two stations from the first column of agents, which was the same case in the initial test. However, there were two intersections with the two marks (Figure 35).

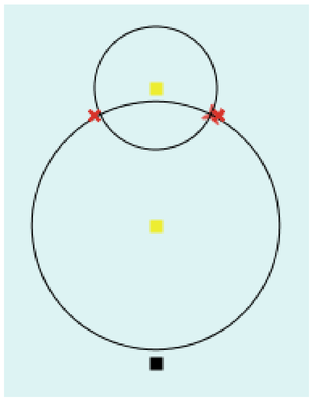


Figure 35. Normal two marks intersections

To define which intersection was the correct track, the middle-point concept was developed. Assuming the UAV always entering the detection from left to right, then before the UAV hit the middle point of the two stations, the right intersection would be regarded as correct track. Once the UAV passed the middle point of the two stations, the intersection on the left would be considered as correct track. In Figure 36, it showed that the UAV was passing the middle point of the two yellow stations.

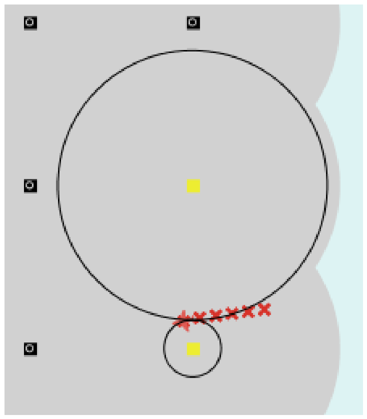


Figure 36. Two vertical marks middle point

There was another possibility when only two marks detected the UAV. When the UAV was in middle of the detection area, however, it was exposed to only two stations, which was possible (Figure 37). In this case, there would be either one or two intersections within those two marks. If there was one intersection like Figure 37, then that intersection would be considered. However, if there were two intersections, the system would consider the middle point of the two points. It may be inaccurate. However, the simulation should respect the realistic situation. In this case, taking the middle of two intersections was the acceptable assumption.

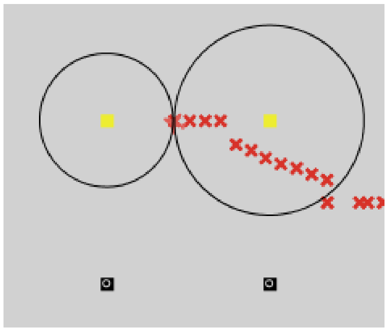


Figure 37. Horizontal two marks middle point

Table 4. Algorithm of detection by two stations

```
if n = 2 [
  show n
  let ox sort [xcor] of black_mark
  let oy sort [ycor] of black_mark
  ; same xcor
  if item 0 ox = item 1 ox [
    let ox1 item 0 ox
    let ox2 item 1 ox
    let oy1 item 1 oy ; o1 to o2 -> up to down, so y1>y2
    let oy2 item 0 oy
    let r1_twice first [size] of black_mark with [ycor = oy1]
    let r1 r1_twice / 2
    let r2_twice first [size] of black_mark with [ycor = oy2]
    let r2 r2_twice / 2
    print ("xcor should be same")
    ; if two intersections,
    ; i1 - intersection 1 left; i2 - intersection 2 right
    ; i2_y = i1_y
    let i1_x (ox1 - sqrt ( abs (r1 ^ 2 - (r1 ^ 2 + v_gap ^ 2 - r2 ^ 2) ^ 2 /
(4 * v_gap ^ 2) ) ) )
    let i1_y (oy1 - ( (r1 ^ 2 + v_gap ^ 2 - r2 ^ 2) / ( 2 * v_gap ) ) )
    let i2_x (ox1 + sqrt ( abs (r1 ^ 2 - (r1 ^ 2 + v_gap ^ 2 - r2 ^ 2) ^ 2 /
(4 * v_gap ^ 2) ) ) )
    let i2_y i1_y
    let d_x [xcor] of drone 0
    (ifelse
      d_x >= ox1 [  ; when drone access the detection area
                    ; only mark i2 on the right
        print ("when drone access the detection area, only mark i2 on the
right")
        hatch-crosses 1 [
          set color red
          set size 2
          setxy (i2_x) (i2_y)
          show list xcor ycor
          file-print csv:to-row (list who xcor ycor)
          stamp
          die
        ]
      ]
      [ ; only mark i1 on the left
        print ("only mark i1 on the left")
        hatch-crosses 1 [
          set color red
```

Table 4 continued

```
          set size 2
          setxy (i1_x) (i1_y)
          show list xcor ycor
          file-print csv:to-row (list who xcor ycor)
          stamp
          die
        ]
      ] )
    ]
  if item 0 oy = item 1 oy [ ; same ycor
    let ox1 item 1 ox
    let ox2 item 0 ox
    let oy1 item 0 oy
    let oy2 item 1 oy
    let r1_twice first [size] of black_mark with [xcor = ox1]
    let r1 r1_twice / 2
    let r2_twice first [size] of black_mark with [xcor = ox2]
    let r2 r2_twice / 2
    print ("ycor should be same")
    ; i1 - intersection 1 up; i2 - intersection 2 down
    ; cannot tell which one, so get the middle point
    ; i2_x = i1_x
    let i1_x (ox1 - v_gap + (r2 ^ 2 + v_gap ^ 2 - r1 ^ 2) / ( 2 * v_gap ) )
    let i1_y (oy1 + sqrt ( abs (r2 ^ 2 - (r2 ^ 2 + v_gap ^ 2 - r1 ^ 2) ^ 2 /
(4 * v_gap ^ 2) ) ) )
    let i2_x i1_x
    let i2_y (oy1 - sqrt ( abs (r2 ^ 2 - (r2 ^ 2 + v_gap ^ 2 - r1 ^ 2) ^ 2 /
(4 * v_gap ^ 2) ) ) )
    hatch-crosses 1 [
      set color red
      set size 2
      setxy i1_x oy1
      show list xcor ycor
      file-print csv:to-row (list who xcor ycor)
      stamp
      die
    ]
  ]
]
```

4.5.2 Track with Three stations

The scenarios which happened mostly were the UAV tracking by three marks (Figure 38). The way to calculate the intersection is using the trigonometric functions:

$$a^2 = b^2 + c^2 - 2bc\cos A$$

$$\cos A = \frac{b}{c}\text{, when C = 90 degrees}$$

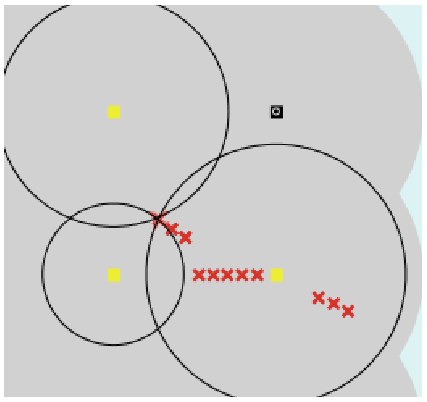


Figure 38. Three marks intersections

It was possible that the UAV was in four nearby stations' detection ranges at the meanwhile. If it was the case, then the system would rule out the furthest station and consider the rest three (Figure 39), due to the nature of Trilateration.

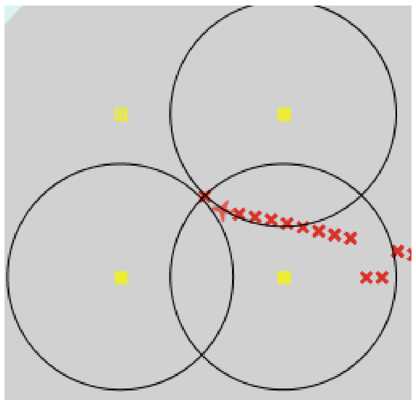


Figure 39. Three marks out of four stations

4.6 Experiment Results

There were twenty simulations randomly selected from over fifty runs to be the samples for the data analysis. The data of the UAV coordinates and the crosses coordinates reported by the simulation using a csv. file. Comparing the UAV coordinates and where the crosses were being marked at the same time, how well the agent-based model worked for the UAV tracking could be determined. For every UAV coordinate and the cross coordinate recorded at the same time, it became a pair of data. The following chapters listed the simulation results that went well and the simulation results that were not satisfied enough.

4.6.1 Expected Results

The expected simulation results had high accuracy to track the UAV route. Due to the approximation made to calculate the intersections, a few inaccurate detections were allowed. For the expected results, all the simulations had the setup showing in the table:

Table 5. Variables setup

Variable Name	Range	Value
x_base	60 – 100	80
y_base	10 – 30	20
h_gap	0 – 50	20
v_gap	0 – 30	20
detect_radius	14 – 30	18

4.6.1.1 Simulation 1

Table 6. Simulation 1 result

Pairs of data	Correct Track (%)	Flaws (%)	
41	100	0	
		Inaccuracy (%)	Undetected (%)
		0	0

Table 7. Simulation 1 data

Agent ID	xcor	ycor
49	132.255	48.439
50	132.255	48.439
0	132.255	48.439
51	130.306	47.990
52	130.306	47.990
0	130.306	47.990
53	128.350	47.574
54	128.350	47.574
0	128.350	47.574
55	126.401	47.124
56	126.401	47.124
0	126.401	47.124
57	124.479	46.573
58	124.479	46.573
0	124.479	46.573
59	122.556	46.021
60	122.556	46.021
0	122.556	46.021
61	120.607	45.571
62	120.607	45.571
0	120.607	45.571
	……	
147	54.936	45.226
148	54.936	45.226
0	54.936	45.226
149	53.004	44.708
150	53.004	44.708
0	53.004	44.708

4.6.1.2 Simulation 2

Table 8. Simulation 2 result

Pairs of data	Correct Track (%)	Flaws (%)	
37	83.8	16.2	
		Inaccuracy (%)	Undetected (%)
		16.2	0

Table 9. Simulation 2 data

Agent ID	xcor	ycor
49	134.057	49.304
50	134.057	49.304
0	134.057	49.304
51	132.057	49.339
52	132.057	49.339
0	132.057	49.339
	……	
69	112.996	50.000
70	112.996	50.000
71	112.996	50.000
0	114.114	49.304
72	111.705	50.000
73	111.705	50.000
74	111.705	50.000
0	112.144	49.651
75	109.622	50.000
76	109.622	50.000
77	109.622	50.000
0	110.212	50.169
	……	
136	65.311	60.000
137	65.311	60.000
0	65.311	59.106
138	63.457	60.000
139	63.457	60.000
0	63.457	59.855

There are several reasons causing the inaccurate tracks. The first reason was due to a reasonable hypothesis which made by the program. When there were three marks detecting the UAV, the trilateration can be used to calculate where the UAV was. However, when there were only two stations detecting the UAV, the system was only aware of the one or two intersections of the two marks. If it was one intersection, there would be no assumption made. The cross would be drawn at that intersection. However, if there were two intersections, the system did not have enough data to determine which intersection was the correct one. In this case, the system would assume that choosing the middle point between the two intersections. Then, a cross would be drawn at the middle point, pretending the UAV location.

The other possible reason was due to the limitation of the system. When the detection range of each station set to cover all the detection area, however, smaller to cover other stations, the track had a high accuracy marking the route of the UAV. When the detection range of each station enlarged, till it can cover its neighbor stations, the system acted unstably so that the marks were unable to correct showing the distances between the stations and the UAV. To improve the system stability, the algorithm would be analyzed to delimit the application range of the system and trilateration.

4.6.1.3 Simulation 3

Table 10. Simulation 3 result

Pairs of data	Correct Track (%)	Flaws (%)	
36	88.9	11.1	
		Inaccuracy (%)	Undetected (%)
		5.55	5.55

Table 11. Simulation 3 data

Agent ID	xcor	ycor
49	117.514	60.000
50	117.514	60.000
0	117.514	60.239
51	115.533	60.000
52	115.533	60.000
0	115.533	60.518
	……	
0	101.585	61.493
0	99.590	61.632
65	97.634	60.000
66	97.634	60.000
0	97.634	62.048
67	95.649	60.000
68	95.649	60.000
0	95.649	62.292
69	93.686	60.000
70	93.686	60.000
0	93.686	62.674
	……	
118	54.243	64.679
119	54.243	64.679
0	54.243	64.679
120	52.331	64.094
121	52.331	64.094
0	52.331	64.094

The detection area was fully covered by the stations. In this case, when the UAV flew through the area, the UAV should always be watched. However, there were times when the UAV was only detected by one single station, indicating that the trilateration cannot perform under this situation. According to the experiment design rule, at this specific position, the UAV cannot be tracked by the system, causing the misses or the undetected result. Even though it resulted in a false positive situation that the UAV cannot be detected correctly, the result did not affect much on the UAV route track. The reason was because the system performed tracking very often. If the system was not able to track the UAV at few times, the route would still represent the UAV movement.

4.6.2 Overall Performance

The statistics focused on the average performance, for example, how long to finish a simulation, how many pairs of data in a simulation, how accurate of the track, and if there were some flaws, whether they were inaccurate tracks or complete misses of the UAV.

According to the statistics (Table 12), for inaccuracy tracks, the average deviation was around 3.23, which meant averagely, a bump caused 3.23 away from the UAV on the x coordinate. In Table 12, every 100 pairs of data, there could be 8.65 pairs with flaws. 2.7 of the flaws were that the stations undetected the UAV at all. While the other 5.95 flaws happened because of the inaccuracy track. Overall, at any times, the correct track rate was higher than 90%, which indicated that the agent-based model did the job tracking the UAV through the detection area.

Table 12. Average performances

Average Performance of Every Simulation			
Time (s)	Pairs of Data	Correct Track (%)	Flaws (%)
3.79	37	91.35	8.65

Table 13. Flaw insider

Flaw Analysis	
Inaccuracy (%)	Undetected (%)
5.95	2.7
Estimated Deviation of Inaccuracy	
3.23	

4.6.3 Unusual Results

The unexpected results did not happen previously, when the detect_radius variable was set as 18. It happened when the program was tested by one of the committee members. The problem was that when the detect_radius was larger than 25, the inaccuracy became visually recognizable (Table 14). Later, more tests were performed to determine whether the detect_radius affected the system or not. The observation was recorded in Table 15:

Table 14. Parameters when unexpected results showed

Variable Name	Range	Value
x_base	60 – 100	80
y_base	10 – 30	20
h_gap	0 – 50	20
v_gap	0 – 30	20
detect_radius	14 – 30	>25

Table 15. Observation of different range of detect_radius

Variable Name	Value	Observation
detect_radius	< 20	The system tracked the UAV route with high accuracy
	20 – 25	The system can track the UAV route with less accuracy. The crosses began a little “jumpy” (Figure 40)
	>25	The system barely tracked the UAV route. The route showed low accuracy. The crosses were “jumping” most of the time (Figure 41)

According to the tests, a hypothesis that the sensor detection radius can affect the system performance. To figure out the cause, further research would be required. For now, one possible explanation was the algorithm. Due to the detect_radius was large, there were more overlaps between stations, so that there would be many stations detecting the UAV at the same time. When the system selected the three closest stations to conduct the trilateration, the system did not consider the situation where there were four, five or more closest stations. Why were there four or five closest stations? Because the distances could tie. When there were many overlaps between stations, it was very possible to have the same distances to the UAV. However, the system had not been developed to deal with the distance-tie situation, which caused the system confused which three stations to choose. Thus, the inaccurate detection happened.

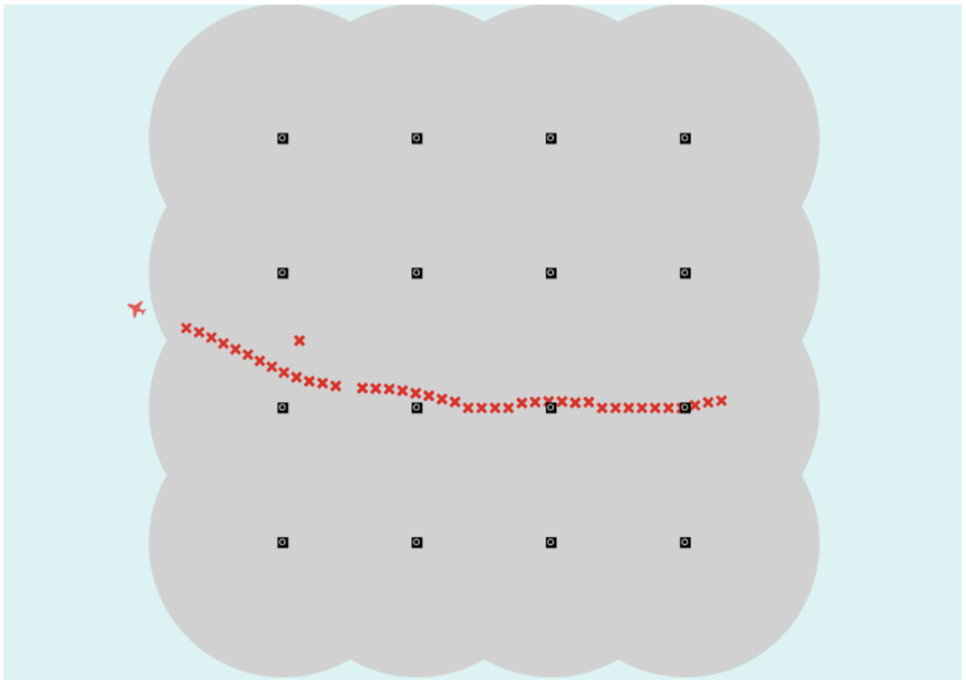


Figure 40. Radius 20, one jumpy cross

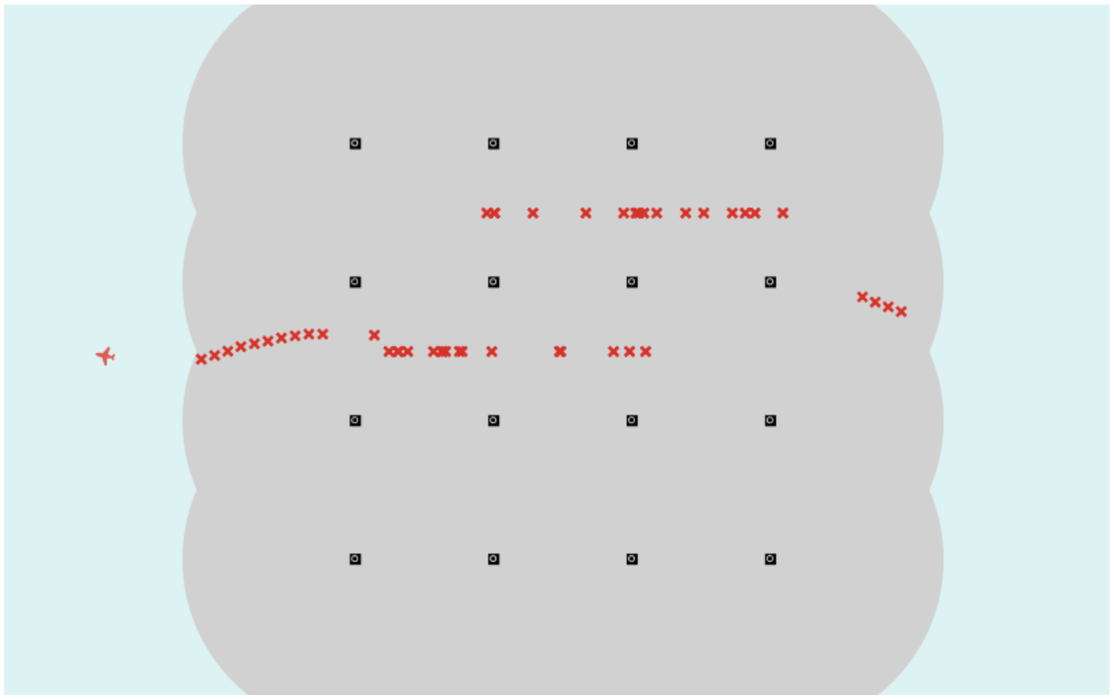


Figure 41. Radius 25, many jumpy crosses

CHAPTER 5. CONCLUSION AND FUTURE WORKS

5.1 Conclusion

The research topic was originated by applying the multi-agent system to the UAV detection system. In the research, an agent-based model was developed to mimic the scenarios tracking the UAV movement in the detection area. The station agents assumed to load the proximity sensor on them, in order to measure the distance between the UAV and the stations themselves. There were groups of agents using in the simulation. According to the result analysis, the track benefited from the similarity of agents. Overall, the correct and accurate track rate was higher than 90%, which was a promising sign that the multi-agent system could help with the UAV track. Even though there were flaws, however, there were more potentials than flaws in the project. Again, applying multi-agent system to the counter UAV technology was bold, however, understandable. More and more counter UAV technologies involved sensor-fusion. Agent-based model would be the great platform for the sensor-fusion detection system. When there were sensor-layers composed by multiple categories of sensors, the track accuracy must be higher. The research was the firm first step to the future of the counter UAV technologies.

5.2 Future Works

As mentioned, the research is the first step to a multi-agent UAV detection system. There are possibilities in every aspect to continuously work on the research.

The station layout can be changed. Then, the evaluation of the tracking accuracy can be performed among different layouts. In the research, only the grid layout was simulated. As proposed, the circle layout could be considered in the simulation. Besides, the station can also form as triangles and other shapes.

The performance could be improved by involving more kinds of sensors and algorithms. Since the research is in 2-D, the proximity sensor along can handle the situations. However, as the program being developed more and more close to the reality, the 3-D modelling must be taken in consideration. In this case, other sensors should be added for the simulation.

In the research, one assumption is that the UAV can be detected by a proximity sensor. According to the result, there are flaws when tracking the UAV movement. To improve the tracking, sensor-layers could be considered. Different sensors monitor different areas and heights, so that the system will take advantage of the variable ranges and different sensitivities from all kinds of sensors. As a result, the accuracy of the tracking result and the reliability of the system will be improved.

REFERENCES

"Confucius Quotes." Quotes.net. STANDS4 LLC, 2019. Web. 27 Jul 2019. https://www.quotes.net/quote/1227

Abar, S., Theodoropoulos, G. K., Lemarinier, P., & O'Hare, G. M. P. (2017). Agent Based Modelling and Simulation tools: A review of the state-of-art software. Computer Science Review, 24, 13–33. https://doi.org/10.1016/j.cosrev.2017.03.001

Allan, R. (2010). Survey of Agent Based Modelling and Simulation Tools. Science and Technology Facilities Council, 48.

AP News. (2018, June 28). Global Counter-UAV (C-UAV) Systems Market Forecast to 2026: A $13.9 Billion Opportunity - Drivers, Constraints, Opportunities and Threats - ResearchAndMarkets.com. Retrieved July 12, 2019, from https://www.apnews.com/d90d975664394fe08f9c23dc8dc7d308

Atmaram, U. (2006). Learning agent, based on Artificial Intelligence: A Modern Approach. Retrieved from https://commons.wikimedia.org/wiki/File:IntelligentAgent-Learning.png

Atmaram, U. (2006). Simple reflex agent, based on Artificial Intelligence: A Modern Approach. Retrieved from https://commons.wikimedia.org/wiki/File:IntelligentAgent-SimpleReflex.png

Bonabeau, E. (2002). Agent-based modeling: Methods and techniques for simulating human systems. Proceedings of the National Academy of Sciences, 99(Supplement 3), 7280–7287. https://doi.org/10.1073/pnas.082080899

Castle, S. (2016, May 28). Dutch Firm Trains Eagles to Take Down High-Tech Prey: UAVs. The New York Times. Retrieved from https://www.nytimes.com/2016/05/29/world/europe/UAVs-eagles.html

Christian, J. (2019, February 9). Someone is terrorizing people by peering in windows with a UAV. Retrieved July 29, 2019, from Futurism website: https://futurism.com/the-byte/terrorizing-people-peering-windows-UAV

Corrigan, F. (2019, May 22). 12 Top Collision Avoidance UAVs And Obstacle Detection Explained [Text]. Retrieved July 25, 2019, from UAVZon website: https://www.UAVzon.com/learn-about-UAVs-quadcopters/top-UAVs-with-obstacle-detection-collision-avoidance-sensors-explained/

Department of Defense (2011). *Unmanned Aircract System Airspace Integration Plan, Version 2.0, March 2011*. Retrieved from https://www.hsdl.org/?abstract&did=

DJI. (2018). The Era of Intelligent Flight is Now | DJI. Retrieved July 25, 2019, from Intelligent Flight Modes website: http://www.dji.com/product/intelligent-flight-modes

Dolicanin, E., Fetahovic, I., Tuba, E., Capor-Hrosik, R., & Tuba, M. (2018). Unmanned Combat Aerial Vehicle Path Planning by Brain Storm Optimization Algorithm. Studies in Informatics and Control, 27(1). https://doi.org/10.24846/v27i1y201802

Elsevier. (2019). Improve engineering research success. Retrieved July 7, 2019, from https://www.elsevier.com/solutions/engineering-village/features

FAA. (2019). *Unmanned Aircraft Systems* (Rep.). Retrieved July 12, 2019, from https://www.faa.gov/data_research/aviation/aerospace_forecasts/media/unmanned_aircraft_systems.pdf

Gonzalez, R. (2013, August 6). Which simulator is the best for Multi-Agent Systems? - Answer by Rafael A Gonzalez [Questions]. Retrieved July 29, 2019, from ResearchGate website: https://www.researchgate.net/post/Which_simulator_is_the_best_for_Multi-Agent_Systems

Goppert, J. M., Wagoner, A. R., Schrader, D. K., Ghose, S., Kim, Y., Park, S., … Hopmeier, M. J. (2017). Realization of an Autonomous, Air-to-Air Counter Unmanned Aerial System (CUAS). 2017 First IEEE International Conference on Robotic Computing (IRC), 235–240. https://doi.org/10.1109/IRC.2017.10

Hennigan, W. J. (2018, May 31). Security Experts Say UAVs Pose a National Security Threat. Retrieved July 12, 2019, from https://time.com/5295586/UAVs-threat/

Jenkins, D., & Vasigh, B. (2018, November 11). The economic impact of unmanned aircraft systems integration in the United States. Retrieved July 12, 2019, from https://www.worldcat.org/title/economic-impact-of-unmanned-aircraft-systems-integration-in-the-united-states/oclc/840129032

Jensen, J. (2019, May 30). UAV Jammer 101 — An Inside Look at Counter UAS Technology. Retrieved July 29, 2019, from UAV Coach website: https://uavcoach.com/UAV-jammer/

Jun, M., & D'Andrea, R. (2003). Path Planning for Unmanned Aerial Vehicles in Uncertain and Adversarial Environments. In S. Butenko, R. Murphey, & P. M. Pardalos (Eds.), Cooperative Control: Models, Applications and Algorithms (Vol. 1, pp. 95–110). https://doi.org/10.1007/978-1-4757-3758-5_6

Kim, Y., & Matson, E. T. (2016). A Realistic Decision Making for Task Allocation in Heterogeneous Multi-agent Systems. Procedia Computer Science, 94, 386–391. https://doi.org/10.1016/j.procs.2016.08.059

Lappin, Y., & Binnie, J. (2016). Rafael unveils UAV Dome counter-UAV system. IHS Jane's Defence Weekly, p. IHS Jane's Defence Weekly, Apr 20, 2016, Vol.53(24). https://techtime.news/2016/04/11/rafael/

Li, S., Kim, H., Lee, S., Gallagher, J. C., Kim, D., Park, S., & Matson, E. T. (2018, October). Convolutional Neural Networks for Analyzing Unmanned Aerial Vehicles Sound. In 2018 18th International Conference on Control, Automation and Systems (ICCAS) (pp. 862-866). IEEE.

Lifehack & Experimenter. (2018, January 25). Magnet Iron filings = Fun Magic & Satisfying Experiment. Retrieved July 29, 2019, from https://www.youtube.com/watch?v=LMFAOLKaYd0

Lim, D., Kim, H., Hong, S., Lee, S., Kim, G., Snail, A., . . . Gallagher, J. (2018). Practically Classifying Unmanned Aerial Vehicles Sound Using Convolutional Neural Networks. 2018 Second IEEE International Conference on Robotic Computing (IRC). doi:10.1109/irc.2018.00051

Lin, Y., & Saripalli, S. (2017). Sampling-Based Path Planning for UAV Collision Avoidance. IEEE Transactions on Intelligent Transportation Systems, 18(11), 3179–3192. https://doi.org/10.1109/TITS.2017.2673778

Macal, C. M., & North, M. J. (2010). Tutorial on agent-based modelling and simulation. Journal of Simulation, 4(3), 151–162. https://doi.org/10.1057/jos.2010.3

Matson, E., Yang, B., Smith, A., Dietz, E., & Gallagher, J. (2019). UAV Detection System with Multiple Acoustic Nodes Using Machine Learning Models. 2019 Third IEEE International Conference on Robotic Computing (IRC), 493–498. https://doi.org/10.1109/IRC.2019.00103

Michel, H. (2018, February 20). Report: Counter-UAV Systems. Retrieved June 15, 2019, from https://UAVcenter.bard.edu/counter-UAV-systems/

Morby, A. (2016, February 10). Eagles are being trained to take down UAVs from the sky. Retrieved July 29, 2019, from Dezeen website: https://www.dezeen.com/2016/02/10/eagles-attacking-UAV-crime-london-metropolitan-police-netherlands-guard-from-above/

Mualla, Y., Bai, W., Galland, S., & Nicolle, C. (2018). Comparison of Agent-based Simulation Frameworks for Unmanned Aerial Transportation Applications. Procedia Computer Science, 130, 791–796. https://doi.org/10.1016/j.procs.2018.04.137

Naboulsi, Z. (2015, May 28). UAV detection: What works and what doesn’t. Retrieved July 29, 2019, from Help Net Security website: https://www.helpnetsecurity.com/2015/05/28/UAV-detection-what-works-and-what-doesnt/

National University Library System. (2019, June 22). Database Tutorials and How-to Guides: IEEE Xplore. Retrieved July 7, 2019, from https://nu.libguides.com/how/ieee

Oliver. (2019, June 30). Best Obstacle Avoidance UAVs: Features, Technology & Reviews. Retrieved July 25, 2019, from FPV Frenzy website: https://fpvfrenzy.com/best-obstacle-avoidance-UAVs-features-technology-reviews/

Peacock, M. (2014). *Detection and control of small civilian UAVs*. Retrieved from https://ro.ecu.edu.au/theses_hons/120

Philly by Air. (2019, March 12). 33 Eye-Opening UAV Stats - Key Trends for 2019. Retrieved July 12, 2019, from https://www.phillybyair.com/blog/UAV-stats/

ProQuest. (2019, June 28). ABI/INFORM Collection: About. Retrieved July 7, 2019, from https://proquest.libguides.com/abiinformcollection

Reitmeyer, J. (2019, February 21). Move to Cover UAVs in Trespassing, Invasion of Property Laws - NJ Spotlight [Newsletters]. Retrieved July 29, 2019, from NJSPOTLIGHT website: http://www.njspotlight.com/stories/19/02/20/bill-would-extend-trespassing-invasion-of-property-laws-to-cover-UAVs/

S. Park, S. Shin, Y. Kim, E. T. Matson, K. Lee, P. J. Kolodzy, J. C. Slater, M. Scherreik, M. Sam, J. C. Gallagher et al., “Combination of radar and audio sensors for identification of rotor-type unmanned aerial vehicles (uavs),” in SENSORS, 2015 IEEE. IEEE, 2015, pp. 1–4.

Schank, J. (2013). Agent-Based Models methodology and philosophy - Simulators. Retrieved July 29, 2019, from Agent-Based Models methodology and philosophy website: http://www.agent-based-models.com/blog/resources/simulators/

Shoham, Y. (1993). Agent-oriented programming. Artificial Intelligence, 60, 42.

Siebers, Peer-Olaf, & Aickelin, U. (2018). INTRODUCTION TO MULTI-AGENT SIMULATION. Encyclopaedia of Decision Making and Decision Support Technologies, 554–564.

Sycara, K (2012). Carnegie Mellon University: Advanced Agent-Robotics Technology Lab: Multi-Agent Systems. Retrieved from https://www.cs.cmu.edu/~softagents/multi.html

Techopedia. (n.d.). What is an Unmanned Aerial Vehicle (UAV)? - Definition from Techopedia. Retrieved July 12, 2019, from https://www.techopedia.com/definition/29896/unmanned-aerial-vehicle-uav

UAV Coach. (2019). Top 100 UAV Companies to Watch in 2019. Retrieved July 25, 2019, from UAV Coach website: https://uavcoach.com/UAV-companies/

Uhrmacher, A., & Weyns, D. (Eds.). (2009). Multi-agent systems: Simulation and applications. Boca Raton: CRC Press/Taylor & Francis.

Ye, J. (2017, March 15). Chinese police armed with anti-UAV ‘guns’ to scramble signals and force landings | South China Morning Post [Newsletters]. Retrieved July 29, 2019, from South China Morning Post website: https://www.scmp.com/news/china/society/article/2079045/chinese-police-force-equipped-anti-UAV-guns

APPENDIX A. ALGORITHM OF DETECTION BY THREE STATIONS

Table 16. Algorithm of detection by three stations

```
if n = 3 [

  let ox sort [xcor] of black_mark
  let oy sort [ycor] of black_mark
  let d sqrt(v_gap ^ 2 + h_gap ^ 2)
  let B precision (acos( h_gap / d ) ) 3

  if (item 1 ox = item 2 ox and item 0 oy = item 1 oy) [ ; situation 1 -----
-----------------------

    print ("situation 1 ###############")

    let ox3 item 0 ox
    let ox1 item 1 ox
    let ox2 item 2 ox
    let oy3 item 0 oy
    let oy2 item 1 oy
    let oy1 item 2 oy
    let r1_twice first [size] of black_mark with [xcor = ox1 and ycor = oy1]
    let r1 r1_twice / 2
    let r2_twice first [size] of black_mark with [xcor = ox2 and ycor = oy2]
    let r2 r2_twice / 2
    let r3_twice first [size] of black_mark with [xcor = ox3 and ycor = oy3]
    let r3 r3_twice / 2
    ; i1 & i2
    let i1_x (ox1 - sqrt ( abs (r1 ^ 2 - (r1 ^ 2 + v_gap ^ 2 - r2 ^ 2) ^ 2 /
(4 * v_gap ^ 2) ) ) )
    let i1_y (oy1 - ( (r1 ^ 2 + v_gap ^ 2 - r2 ^ 2) / ( 2 * v_gap ) ) )
    let i2_x (ox1 + sqrt ( abs (r1 ^ 2 - (r1 ^ 2 + v_gap ^ 2 - r2 ^ 2) ^ 2 /
(4 * v_gap ^ 2) ) ) )
    let i2_y i1_y
    ; i3 & i4 up to down
    let i3_x (ox2 - v_gap + (r3 ^ 2 + v_gap ^ 2 - r2 ^ 2) / ( 2 * v_gap ) )
    let i3_y (oy2 + sqrt ( abs (r3 ^ 2 - (r3 ^ 2 + v_gap ^ 2 - r2 ^ 2) ^ 2 /
(4 * v_gap ^ 2) ) ) )
    let i4_x i3_x
    let i4_y (oy2 - sqrt ( abs (r3 ^ 2 - (r3 ^ 2 + v_gap ^ 2 - r2 ^ 2) ^ 2 /
(4 * v_gap ^ 2) ) ) )
    ; i5 & i6
;        let A precision (acos( (r3 ^ 2 + d ^ 2 - r1 ^ 2) / (2 * r3 * d) ) )
3
;        ifelse (A = B) [
;          let i5_x ox3
;          let i5_y (oy3 + r3)
;          let i6_x (ox3 + r3)
;          let i6_y oy3
;        ]
;        [
;          let i5_x ( ox3 + cos (A + B) * r3 )
;          let i5_y ( oy3 + sin (A + B) * r3 )
;          let i6_x ( ox3 + cos (B - A) * r3 )
;          let i6_y ( oy3 + sin (B - A) * r3 )
;        ]
    ; situation 1, i1
    hatch-crosses 1 [
      set color red
      set size 2
      setxy i1_x i1_y
      show list xcor ycor
      file-print csv:to-row (list who xcor ycor)
      stamp
      die
    ]
  ]

  if (item 0 ox = item 1 ox and item 0 oy = item 1 oy) [ ; situation 2 -----
-----------------------

    print ("situation 2 ###############")
    let ox1 item 0 ox
    let ox2 item 1 ox
    let ox3 item 2 ox
    let oy2 item 0 oy
    let oy3 item 1 oy
    let oy1 item 2 oy
    let r1_twice first [size] of black_mark with [xcor = ox1 and ycor = oy1]
    let r1 r1_twice / 2
    let r2_twice first [size] of black_mark with [xcor = ox2 and ycor = oy2]
    let r2 r2_twice / 2
    let r3_twice first [size] of black_mark with [xcor = ox3 and ycor = oy3]
    let r3 r3_twice / 2
    ; i1 & i2
    let i1_x (ox1 - sqrt ( abs (r1 ^ 2 - (r1 ^ 2 + v_gap ^ 2 - r2 ^ 2) ^ 2 /
(4 * v_gap ^ 2) ) ) )
```

```
    let i1_y (oy1 - ( (r1 ^ 2 + v_gap ^ 2 - r2 ^ 2) / ( 2 * v_gap ) ) )
    let i2_x (ox1 + sqrt ( abs (r1 ^ 2 - (r1 ^ 2 + v_gap ^ 2 - r2 ^ 2) ^ 2 /
(4 * v_gap ^ 2) ) ) )
    let i2_y i1_y
    ; i3 & i4
    let i3_x (ox3 - v_gap + (r2 ^ 2 + v_gap ^ 2 - r3 ^ 2) / ( 2 * v_gap ) )
    let i3_y (oy2 + sqrt ( abs (r2 ^ 2 - (r2 ^ 2 + v_gap ^ 2 - r3 ^ 2) ^ 2 /
(4 * v_gap ^ 2) ) ) )
    let i4_x i3_x
    let i4_y (oy2 - sqrt ( abs (r2 ^ 2 - (r2 ^ 2 + v_gap ^ 2 - r3 ^ 2) ^ 2 /
(4 * v_gap ^ 2) ) ) )
    ; i5 & i6
;         let A precision (acos( (r3 ^ 2 + d ^ 2 - r1 ^ 2) / (2 * r3 * d) ) )
3
;         ifelse (A = B) [
;           let i5_x ox3
;           let i5_y (oy3 + r3)
;           let i6_x (ox3 - r3)
;           let i6_y oy3
;         ]
;         [
;           let i5_x ( ox3 - cos (A + B) * r3 )
;           let i5_y ( oy3 + sin (A + B) * r3 )
;           let i6_x ( ox3 - cos (B - A) * r3 )
;           let i6_y ( oy3 + sin (B - A) * r3 )
;         ]
    ; situation 2, i2
    hatch-crosses 1 [
      set color red
      set size 2
      setxy i2_x i2_y
      show list xcor ycor
      file-print csv:to-row (list who xcor ycor)
      stamp
      die
    ]
  ]

  if (item 1 ox = item 2 ox and item 1 oy = item 2 oy) [ ; situation 3 -----
-----------------------

    print ("situation 3 ###############")
    let ox3 item 0 ox
    let ox2 item 1 ox
    let ox1 item 2 ox
    let oy2 item 0 oy
    let oy3 item 1 oy
    let oy1 item 2 oy
    let r1_twice first [size] of black_mark with [xcor = ox1 and ycor = oy1]
    let r1 r1_twice / 2
    let r2_twice first [size] of black_mark with [xcor = ox2 and ycor = oy2]
    let r2 r2_twice / 2
    let r3_twice first [size] of black_mark with [xcor = ox3 and ycor = oy3]
    let r3 r3_twice / 2
    ; i1 & i2
    let i1_x (ox1 - sqrt ( abs (r1 ^ 2 - (r1 ^ 2 + v_gap ^ 2 - r2 ^ 2) ^ 2 /
(4 * v_gap ^ 2) ) ) )
    let i1_y (oy1 - ( (r1 ^ 2 + v_gap ^ 2 - r2 ^ 2) / ( 2 * v_gap ) ) )
    let i2_x (ox1 + sqrt ( abs (r1 ^ 2 - (r1 ^ 2 + v_gap ^ 2 - r2 ^ 2) ^ 2 /
(4 * v_gap ^ 2) ) ) )
    let i2_y i1_y
    ; i3 & i4
    let i3_x (ox1 - v_gap + (r3 ^ 2 + v_gap ^ 2 - r1 ^ 2) / ( 2 * v_gap ) )
    let i3_y (oy2 + sqrt ( abs (r3 ^ 2 - (r3 ^ 2 + v_gap ^ 2 - r1 ^ 2) ^ 2 /
(4 * v_gap ^ 2) ) ) )
    let i4_x i3_x
    let i4_y (oy2 - sqrt ( abs (r3 ^ 2 - (r3 ^ 2 + v_gap ^ 2 - r1 ^ 2) ^ 2 /
(4 * v_gap ^ 2) ) ) )
    ; i5 & i6
;        let A precision (acos( (r2 ^ 2 + d ^ 2 - r3 ^ 2) / (2 * r2 * d) ) )
3
;        ifelse ( A = B ) [
;          let i5_x ox2
;          let i5_y (oy2 + r2)
;          let i6_x (ox2 - r2)
;          let i6_y oy2
;        ]
;        [
;          let i5_x ( ox2 - cos (A + B) * r2 )
;          let i5_y ( oy2 + sin (A + B) * r2 )
;          let i6_x ( ox2 - cos (B - A) * r2 )
;          let i6_y ( oy2 + sin (B - A) * r2 )
;        ]
    ; situation 3, i1
    hatch-crosses 1 [
      set color red
      set size 2
      setxy i1_x i1_y
      show list xcor ycor
      file-print csv:to-row (list who xcor ycor)
      stamp
      die
    ]
  ]

  if (item 0 ox = item 1 ox and item 1 oy = item 2 oy) [ ; situation 4 -----
-----------------------

    print ("situation 4 ###############")
    let ox1 item 0 ox
    let ox2 item 1 ox
    let ox3 item 2 ox
    let oy2 item 0 oy
    let oy1 item 1 oy
    let oy3 item 2 oy
    let r1_twice first [size] of black_mark with [xcor = ox1 and ycor = oy1]
    let r1 r1_twice / 2
    let r2_twice first [size] of black_mark with [xcor = ox2 and ycor = oy2]
    let r2 r2_twice / 2
    let r3_twice first [size] of black_mark with [xcor = ox3 and ycor = oy3]
    let r3 r3_twice / 2
    ; i1 & i2
    let i1_x (ox1 - sqrt ( abs (r1 ^ 2 - (r1 ^ 2 + v_gap ^ 2 - r2 ^ 2) ^ 2 /
(4 * v_gap ^ 2) ) ) )
    let i1_y (oy1 - ( (r1 ^ 2 + v_gap ^ 2 - r2 ^ 2) / ( 2 * v_gap ) ) )
    let i2_x (ox1 + sqrt ( abs (r1 ^ 2 - (r1 ^ 2 + v_gap ^ 2 - r2 ^ 2) ^ 2 /
(4 * v_gap ^ 2) ) ) )
    let i2_y i1_y
    ; i3 & i4
    let i3_x (ox3 - v_gap + (r1 ^ 2 + v_gap ^ 2 - r3 ^ 2) / ( 2 * v_gap ) )
    let i3_y (oy2 + sqrt ( abs (r1 ^ 2 - (r1 ^ 2 + v_gap ^ 2 - r3 ^ 2) ^ 2 /
(4 * v_gap ^ 2) ) ) )
    let i4_x i3_x
    let i4_y (oy2 - sqrt ( abs (r1 ^ 2 - (r1 ^ 2 + v_gap ^ 2 - r3 ^ 2) ^ 2 /
(4 * v_gap ^ 2) ) ) )
    ; i5 & i6
;        let A precision (acos( (r2 ^ 2 + d ^ 2 - r3 ^ 2) / (2 * r2 * d) ) )
3
;        ifelse (A = B) [
;          let i5_x ox2
;          let i5_y (oy2 + r2)
;          let i6_x (ox2 + r2)
;          let i6_y oy2
;        ]
;        [
;           let i5_x ( ox2 + cos (A + B) * r2 )
;           let i5_y ( oy2 + sin (A + B) * r2 )
;           let i6_x ( ox2 + cos (B - A) * r2 )
;           let i6_y ( oy2 + sin (B - A) * r2 )
;         ]
    ; situation 4, i2
    hatch-crosses 1 [
      set color red
      set size 2
      setxy i2_x i2_y
      show list xcor ycor
      file-print csv:to-row (list who xcor ycor)
      stamp
      die
    ]
  ]
]
```